

PHP

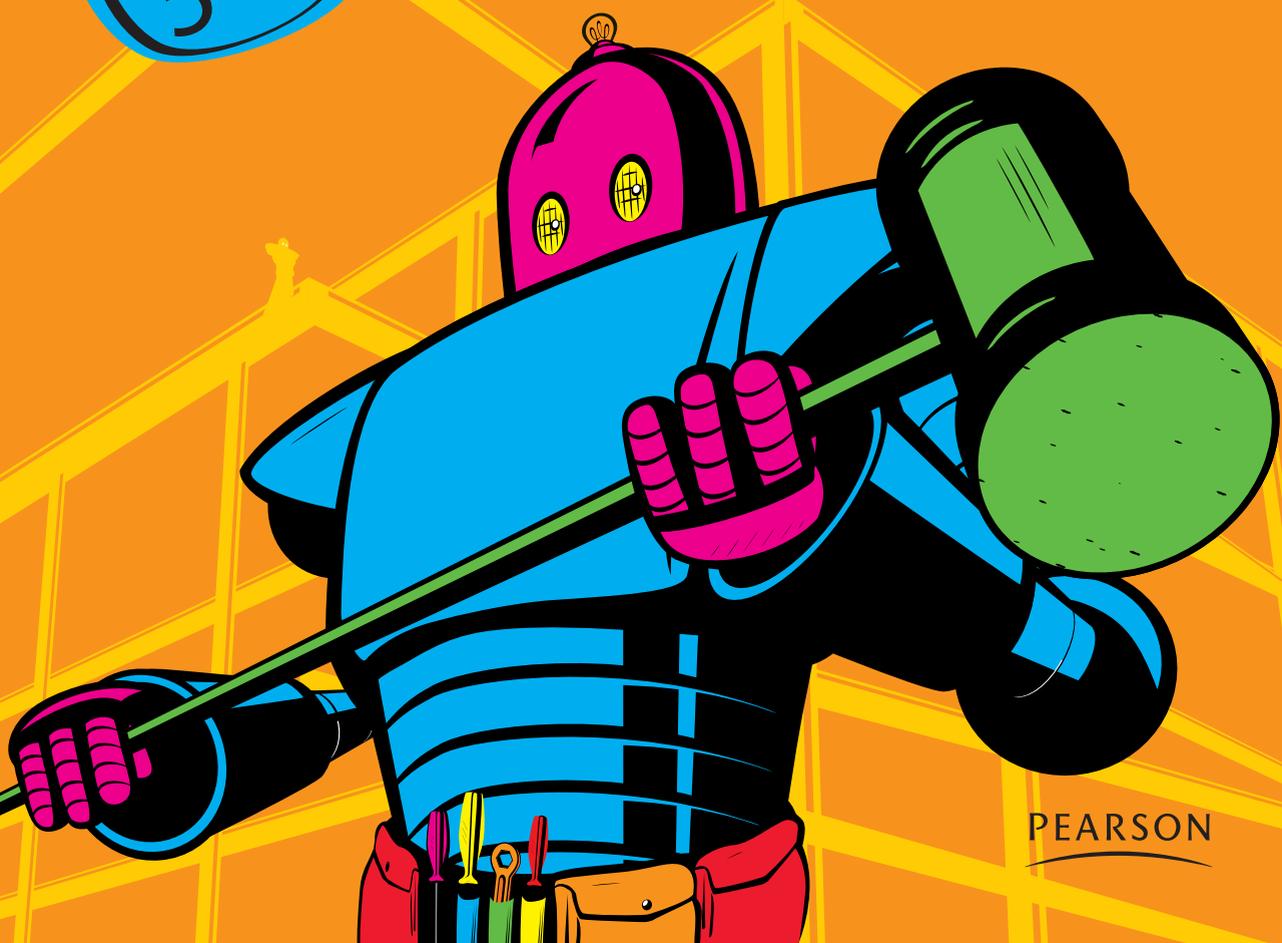
Clés en main

76 SCRIPTS EFFICACES POUR ENRICHIR
VOTRE SITE WEB

WILLIAM STEINMETZ ET BRIAN WARD

Conçu pour
les versions

5 et 6



PEARSON

PHP CLÉS EN MAIN

**76 scripts efficaces
pour enrichir vos sites web**

par William Steinmetz et Brian Ward

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

Titre original : *Wicked cool PHP*

Traduit de l'américain
par Éric Jacoboni

Réalisation PAO : TyPAO

Relecture technique : Jean-Marc Delprato

Collaboration éditoriale : Dominique Buraud

ISBN original : 978-1-59327-173-2
Copyright © 2008 by William Steinmetz
with Brian Ward
All rights reserved

ISBN : 978-2-7440-4030-6

Copyright© 2009 Pearson Education France

Tous droits réservés

Éditeur original : No Starch Press
www.nostarch.com

Aucune représentation ou reproduction, même partielle, autre que celles prévues à l'article L. 122-5 2° et 3° a) du code de la propriété intellectuelle ne peut être faite sans l'autorisation expresse de Pearson Education France ou, le cas échéant, sans le respect des modalités prévues à l'article L. 122-10 dudit code.

No part of this book may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, recording or by any information storage retrieval system, without permission from the publisher.

TABLE DES MATIÈRES

INTRODUCTION 1

1

TOUT CE QUE VOUS AVEZ TOUJOURS VOULU SAVOIR SUR LES SCRIPTS PHP SANS JAMAIS OSER LE DEMANDER 3

Recette 1 : Inclure un fichier extérieur dans un script	4
Problèmes éventuels	5
Recette 2 : Alternier les couleurs des lignes d'un tableau	7
Amélioration du script	8
Recette 3 : Créer des liens Précédent/Suivant	9
N'afficher qu'un sous-ensemble de lignes de votre base de données.	11
Compter le nombre total de lignes de l'ensemble résultat.	11
Utilisation du script	12
Recette 4 : Afficher le contenu d'un tableau	14
Recette 5 : Transformer un tableau en variable scalaire qui pourra être restaurée ultérieurement	15
Problèmes éventuels	15
Recette 6 : Trier des tableaux à plusieurs dimensions	16
Amélioration du script	17
Recette 7 : Créer des templates pour votre site avec Smarty	17
Installation de Smarty	18
Initiation rapide à Smarty	19
Problèmes éventuels	20
Amélioration du script	20

2

CONFIGURATION DE PHP 23

Les options de configuration et le fichier php.ini	23
Trouver l'emplacement de votre fichier php.ini	24
Recette 8 : Afficher toutes les options de configuration de PHP	25
Recette 9 : Obtenir la valeur d'une option de configuration particulière	25

Recette 10 : Signaler les erreurs	26
Messages d'erreurs classiques	27
Recette 11 : Supprimer tous les messages d'erreur	28
Recette 12 : Allonger le temps d'exécution d'un script	29
Problèmes éventuels	29
Recette 13 : Empêcher les utilisateurs de déposer de gros fichiers	29
Recette 14 : Désactiver les variables globales automatiques	30
Recette 15 : Activer les apostrophes magiques	30
Problèmes éventuels	31
Recette 16 : Restreindre l'accès de PHP aux fichiers	31
Problèmes éventuels	31
Recette 17 : Supprimer des fonctions précises	32
Recette 18 : Ajouter des extensions à PHP	32
Ajouter des extensions PHP	33
Installer des extensions avec un panneau de contrôle web	34
Problèmes éventuels	38

3

SÉCURITÉ ET PHP **39**

Options de configuration recommandées pour la sécurité	41
Recette 19 : Attaques par injection SQL	42
Recette 20 : Empêcher les attaques XSS basiques	43
Recette 21 : Utiliser SafeHTML	45
Problèmes éventuels	46
Recette 22 : Protéger les données avec un hachage non réversible	47
Amélioration du script	48
Recette 23 : Chiffrer les données avec Mcrypt	48
Amélioration du script	50
Recette 24 : Produire des mots de passe aléatoires	51
Utilisation du script	51

4

TRAITEMENT DES FORMULAIRES **53**

Mesures de sécurité : ne faites pas confiance aux formulaires	53
Stratégies de vérification	54
Utiliser \$_POST, \$_GET, \$_REQUEST et \$_FILES pour accéder aux données des formulaires	55
Recette 25 : Récupérer les données des formulaires en toute sécurité	55
Recette 26 : Supprimer les espaces inutiles	56
Recette 27 : Importer des données de formulaire dans un tableau	57
Recette 28 : S'assurer qu'une réponse fait partie d'un ensemble de valeurs	60
Recette 29 : Utiliser plusieurs boutons de validation	61
Recette 30 : Vérifier la validité d'une carte de crédit	61
Recette 31 : Vérifier la date d'expiration d'une carte de crédit	65
Recette 32 : Vérifier la validité des adresses de courrier électronique	66
Recette 33 : Tester la validité des numéros de téléphone	67

5

TRAITEMENT DU TEXTE ET DE HTML 69

Recette 34 : Extraire une partie d'une chaîne	69
Amélioration du script	71
Recette 35 : Mettre une chaîne en majuscules, en minuscules ou en capitales	72
Problèmes éventuels	72
Recette 36 : Rechercher des sous-chaînes	73
Problèmes éventuels	74
Recette 37: Remplacer des sous-chaînes	74
Problèmes éventuels	75
Recette 38 : Trouver et corriger les fautes d'orthographe avec <i>pspell</i>	76
Utiliser le dictionnaire par défaut	76
Ajouter un dictionnaire personnalisé à <i>pspell</i>	80
Problèmes éventuels	80
Recette 39 : Expressions régulières	81
Introduction aux expressions régulières	81
Caractères spéciaux	82
Itérateurs de motifs	82
Groupements	83
Classes de caractères	83
Construction d'une expression régulière	84
Recherches et extractions avec les expressions régulières	85
Remplacement de sous-chaînes avec les expressions régulières	86
Recette 40 : Réarranger un tableau	87
Recette 41 : Extraire des données des pages	88
Amélioration du script	89
Recette 42 : Convertir du texte normal en document HTML	90
Recette 43 : Créer des liens automatiques vers les URL	93
Recette 44 : Supprimer les balises HTML contenues dans une chaîne	93

6

TRAITEMENT DES DATES 95

Représentation du temps avec Unix	95
Recette 45 : Connaître l'instant courant	96
Recette 46 : Obtenir l'instant correspondant à une date du passé ou du futur ...	97
Création d'instants à partir d'une chaîne	97
Création d'instants à partir de dates	99
Recette 47 : Formater les dates et les heures	100
Formater les dates en français	102
Recette 48 : Calculer le jour de la semaine d'une date	103
Recette 49 : Calculer la différence entre deux dates	104
Utilisation du script	105
Amélioration du script	105
Format des dates MySQL	106

7

TRAITEMENT DES FICHIERS **107**

Permissions des fichiers	107
Permissions avec un client FTP	109
La ligne de commande	109
Problèmes éventuels	109
Recette 50 : Mettre le contenu d'un fichier dans une variable	110
Amélioration du script	111
Problèmes éventuels	112
Recette 51 : Écrire dans un fichier	112
Recette 52 : Tester l'existence d'un fichier	113
Recette 53 : Supprimer des fichiers	114
Recette 54 : Déposer des images dans un répertoire	114
Utilisation du script	118
Problèmes éventuels	118
Amélioration du script	119
Recette 55 : Lire un fichier CSV	119

8

GESTION DES UTILISATEURS ET DES SESSIONS **121**

Suivi des données des utilisateurs avec des cookies et des sessions	121
Les cookies	122
Les sessions	122
Recette 56 : Créer un message "Heureux de vous revoir <i>NomUtilisateur</i> !" avec les cookies	123
Problèmes éventuels	124
Recette 57 : Utiliser les sessions pour stocker temporairement des données	125
Problèmes éventuels	127
Recette 58 : Vérifier qu'un navigateur accepte les cookies	128
Recette 59 : Rediriger les utilisateurs vers des pages différentes	129
Recette 60 : Imposer l'utilisation de pages chiffrées par SSL	130
Recette 61 : Obtenir des informations sur le client	130
Recette 62 : Délais d'expiration des sessions	135
Recette 63 : Système de connexion simple	136

9

TRAITEMENT DU COURRIER ÉLECTRONIQUE **139**

Recette 64 : Envoyer du courrier avec PHPMailer	140
Installation de PHPMailer	140
Utilisation du script	141
Ajout de fichiers attachés	143
Problèmes éventuels	143
Recette 65 : Vérifier les comptes utilisateurs avec le courrier électronique	144

10

TRAITEMENT DES IMAGES **149**

Recette 66 : Créer une image CAPTCHA pour améliorer la sécurité	149
Recette 67 : Créer des vignettes	157

11

UTILISATION DE cURL POUR LES SERVICES WEB **163**

Recette 68 : Se connecter à d'autres sites web	164
Recette 69 : Utiliser les cookies	166
Recette 70 : Transformer du XML sous une forme utilisable	167
Recette 71 : Utiliser des services web de localisation géographique	169
Recette 72 : Interroger Amazon avec PHP et SOAP	172
Recette 73 : Construire un service web	174

12

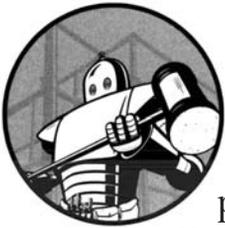
MISE EN APPLICATION **179**

Recette 74 : Un système de sondage	179
Création d'un formulaire pour les bulletins de vote	181
Traitement des votes	183
Récupération du résultat d'un sondage	184
Amélioration du script	187
Recette 75 : Cartes postales électroniques	188
Choix d'une carte	189
Envoi d'une carte	191
Visualisation d'une carte	195
Amélioration du script	197
Recette 76 : Un système de blog	198
Créations de billets	199
Affichage d'un billet	201
Ajout de commentaires	205
Création d'un index des billets	206
Amélioration du script	209

ANNEXE **211**

INDEX **213**

INTRODUCTION



Ce livre est destiné aux développeurs qui viennent de découvrir PHP et qui se demandent comment l'utiliser pour leurs applications. Vous avez sûrement des bases en programmation et vous avez probablement déjà rencontré un grand nombre d'exemples sur Internet, mais vous vous demandez peut-être pourquoi certains de ces exemples sont bien plus compliqués que d'autres qui font la même chose.

Nous avons essayé de faire en sorte que les exemples de ce livre soient les plus simples possibles et d'expliquer au maximum chaque extrait de code : pour réduire les risques de confusion entre les codes serveur et client, nous n'avons, par exemple, que peu utilisé JavaScript. Nous savons que vous êtes impatient et c'est la raison pour laquelle le **premier chapitre, "Tout ce que vous avez toujours voulu savoir sur les scripts PHP sans jamais oser le demander"**, présente des solutions rapides aux principaux petits problèmes quotidiens que tout le monde rencontre.

Lorsque vous serez rassasié, passez au **Chapitre 2, "Configuration de PHP"**, pour savoir comment installer et configurer PHP – beaucoup de problèmes sont dus à une mauvaise configuration.

Le **Chapitre 3, "Sécurité et PHP"**, poursuit dans cette voie en expliquant comment sécuriser vos scripts.

Le **Chapitre 4, "Traitement des formulaires"**, revient aux bases du langage. Il explique notamment comment récupérer ce qu'a saisi l'utilisateur à partir d'un formulaire ou d'autres sources dynamiques.

Le **Chapitre 5, "Traitement du texte et de HTML"**, montre comment traiter le texte et les chaînes de caractères à l'aide de certains outils comme les expressions régulières.

Le **Chapitre 6, "Traitement des dates"**, étudie comment gérer les temps et les dates avec PHP et MySQL et le **Chapitre 7, "Traitement des fichiers"**, est consacré à la manipulation des fichiers.

Après ces points essentiels, le **Chapitre 8, "Gestion des utilisateurs et des sessions"**, présente les détails de la gestion et du suivi des sessions. Lorsqu'un site complexe attire de nombreux utilisateurs, il est important de savoir ce que fait chacun d'eux afin que la session d'un utilisateur particulier n'interfère pas avec celle des autres.

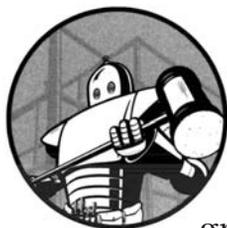
Le **Chapitre 9, "Traitement du courrier électronique"** et le **Chapitre 10, "Traitement des images"** expliquent, respectivement, comment manipuler les e-mails et les images. Ces traitements étant généralement mal adaptés aux scripts serveurs, ces chapitres présentent des traitements relativement légers, qui peuvent améliorer le comportement de votre site.

Le **Chapitre 11, "Utilisation de cURL pour les services web"** montre comment configurer votre serveur web pour qu'il interagisse *via* XML avec des services web fournis par d'autres sites.

Enfin, le **Chapitre 12, "Mise en application"**, contient trois petits projets amusants qui peuvent être intégrés à des sites plus importants. Ces projets mettent en pratique ce qui a été présenté auparavant dans cet ouvrage.

1

TOUT CE QUE VOUS AVEZ TOUJOURS VOULU SAVOIR SUR LES SCRIPTS PHP SANS JAMAIS OSER LE DEMANDER



Les scripts présentés dans ce chapitre répondent à plusieurs questions qui encombrant les forums et les groupes de discussions consacrés à PHP. Parmi elles, citons :

- Comment ajouter des liens *Précédent*/*Suivant* à mon panier virtuel ?
- Existe-t-il un moyen simple d'utiliser une couleur différente pour chaque ligne de mon tableau ?
- Je veux trier un gros tableau et je ne sais pas comment faire !
- Quel système de templates puis-je mettre en place pour que mes données soient formatées de la même façon sur toutes les pages ?

Bien que ce livre contienne des scripts assez compliqués et que d'autres vous sembleront plus intéressants, ceux qui sont présentés ici répondent aux questions que nous rencontrons sans cesse. Ces scripts pour débutants représentent ce que tout le monde *devrait* savoir ou *voudrait* savoir. Faites lire ce chapitre à un programmeur en PHP que vous appréciez et il vous en sera reconnaissant.

NOTE Si vous n'avez pas peur de jouer le rôle d'administrateur de votre serveur web, le Chapitre 2 vous aidera également à progresser si vous débutez en PHP ; il vous facilitera également la vie si vous avez déjà un peu programmé avec ce langage.

Recette 1 : Inclure un fichier extérieur dans un script

La plupart des applications sérieuses utilisent un ensemble de variables et de scripts qui seront repris par quasiment toutes les pages. Si vous concevez, par exemple, un panier virtuel qui se connecte à une base de données MySQL, vous pouvez déclarer le nom d'utilisateur et le mot de passe MySQL dans chaque page du panier, mais que se passera-t-il si vous devez changer ce mot de passe ? La modification et la mise à jour de chaque fichier sur le serveur pourront alors devenir un gros problème.

Au lieu de déclarer le mot de passe dans chaque page de script, vous pouvez stocker le nom d'utilisateur et le mot de passe dans un fichier séparé. Vous pourrez ensuite inclure ce fichier dans votre script et toutes les variables déclarées dans le fichier seront automatiquement déclarées dans le script !

En outre, vous pouvez également stocker des scripts ou des fonctions dans un fichier et ne les inclure que lorsque vous en avez besoin. La fonction permettant de suivre en temps réel les expéditions UPS, par exemple, représente 24 Ko de traitement XML, mais elle ne sert que lorsqu'un client choisit UPS comme type d'expédition. Pourquoi ne pas alors la stocker dans un fichier *suivi_ups.php* et ne l'appeler que lorsque cela s'avère nécessaire ?

En réalité, quasiment toutes les applications PHP professionnelles emploient un fichier portant un nom comme *config.php*, qui contient les déclarations des variables essentielles utilisées par toutes les pages : le nom d'utilisateur et le mot de passe MySQL, par exemple. Ces applications stockent également les scripts utilitaires dans des répertoires distincts : les programmeurs peuvent alors faire de savants mélanges en prenant le script qui vérifie-si-un-utilisateur-est-connecté dans un répertoire et le script qui récupère-les-données-de-la-base dans un autre. Il leur reste à écrire un script qui adapte la vue des données en fonction de la connexion ou non de l'utilisateur. Voici comment faire :

```
<?php
require_once("/chemin/vers/fichier.php");
?>
```

Le fichier passé à `require_once()` fait désormais partie du script, exactement comme si vous aviez copié son contenu pour le coller dans le script. Vous pouvez même inclure des fichiers HTML pour créer un système de templates rudimentaire.

Quel que soit le nom du fichier, PHP tentera de lire son contenu comme du code PHP. Comme pour tout fichier PHP, vous devez donc entourer le code PHP contenu dans ce fichier par les balises `<?php` et `?>` ; sinon, l'interpréteur se

contentera d'afficher le contenu brut de tout le fichier (même s'il est codé en binaire).

La fonction `require_once()` s'utilisant comme n'importe quelle autre instruction, vous pouvez donc l'intégrer dans une structure de contrôle :

```
if ($fichier_necessaire === true) {  
    require_once("fichier_necessaire.php");  
}
```

Problèmes éventuels

Plusieurs choses peuvent mal tourner lorsque l'on inclut un fichier.

Le chemin vers le fichier est incorrect.

En ce cas, le script se termine en signalant une erreur fatale. Si vous préférez qu'il poursuive son exécution même s'il n'a pas trouvé le fichier à inclure, utilisez `include_once()` à la place de `require_once()`.

Le chemin vers le script est correct, mais il se trouve dans un répertoire inaccessible en lecture.

Ce problème peut survenir si vous avez configuré la variable `open_basedir` pour restreindre les répertoires auxquels PHP a accès. Pour des raisons de sécurité, les développeurs web limitent l'accès aux fichiers et aux répertoires importants. Nous expliquerons comment modifier les droits d'accès aux répertoires à la recette n°16, "Restreindre l'accès de PHP aux fichiers".

Le fichier inclus contient une ligne blanche ou une espace avant ou après le code du script PHP.

Si votre script met en place des cookies ou effectue un traitement qui n'est pas conforme à HTTP, il doit le faire *avant* d'envoyer quoi que ce soit au navigateur. N'oubliez pas que PHP affiche tout ce qui ne se trouve pas entre les balises `<?php` et `?>` dans un fichier inclus : si une ligne blanche se trouve avant ou après ces balises, elle sera donc envoyée au navigateur comme s'il s'agissait de code HTML, ce qui empêchera la mise en place des cookies et le démarrage des sessions. Si vous incluez un script, vérifiez qu'il n'y a pas d'espace à l'extérieur des balises PHP. Faites particulièrement attention aux espaces après la balise de fin `?>` car généralement ils restent invisibles dans un éditeur de texte.

NOTE *Les cookies servent à suivre la trace des utilisateurs et à stocker des informations cachées. Consultez le Chapitre 8 pour plus de détails.*

Le fichier inclus peut être lu par des méthodes non PHP.

Vous pouvez stocker des variables PHP dans n'importe quel fichier, quel que soit son nom, mais si vous n'indiquez pas à Apache que ces fichiers doivent être protégés en écriture, il enverra leur contenu en texte brut à quiconque lui demande. Si vous n'y prenez pas garde, toute personne connaissant le nom de vos fichiers inclus pourra donc les lire.

Il est inutile de préciser que le stockage des mots de passe et des noms de comptes MySQL dans un emplacement qui peut être lu par Internet Explorer n'est pas considéré comme une bonne mesure de sécurité.

Pour améliorer la sécurité, vous pouvez placer les fichiers inclus à l'extérieur des répertoires du serveur web (de préférence, dans un répertoire dont l'accès est protégé par un mot de passe), afin que les scripts ne puissent être accédés que par FTP. Si vous manipulez des données sensibles, comme des données bancaires, vous devriez vous sentir obligé de prendre ces mesures.

NOTE *Nous verrons comment valider un numéro de carte bancaire à la recette n°30, "Vérifier la validité d'une carte de crédit".*

Vous êtes perdu dans les inclusions.

Un jour, j'ai acheté un programme de panier virtuel parce qu'il était écrit en PHP et que je comptais adapter les scripts pour mes besoins professionnels. Imaginez ma surprise lorsque j'ai constaté que son module principal (celui où les visiteurs ajoutent, suppriment et modifient les articles) contenait 7 inclusions, 12 lignes de code et 3 templates Smarty. J'ai ouvert l'un des fichiers inclus et j'ai découvert – vous l'aurez deviné – qu'il contenait à son tour trois inclusions.

Les fichiers inclus permettent de rendre votre code très compact mais, croyez-moi, si vous tentez de déchiffrer un script faisant appel à de très nombreux fichiers inclus, c'est l'enfer. Pour la santé mentale des autres programmeurs et des générations futures, *n'incluez pas un fichier sans ajouter un commentaire indiquant aux autres ce que fait ce fichier*, merci.

Vous utilisez des variables non vérifiées comme noms de fichiers inclus.

Bien que vous puissiez écrire `include($fichier)` pour inclure un script en fonction du choix de l'utilisateur, cette pratique permet à un pirate d'inclure n'importe quel fichier du site avec un minimum d'effort ou, selon la configuration de votre serveur, un fichier de son site qui s'introduira donc sur votre serveur. En fait, quelques virus PHP utilisent ce genre de faute de programmation. En outre, ce genre de scripts sont bien plus sujets aux bogues et sont généralement impossible à relire.

Si vous devez inclure des fichiers dont le nom provient d'une variable, utilisez plutôt un script comme celui de la recette n°28, "S'assurer qu'une réponse fait partie d'un ensemble de valeurs", afin de vérifier que les noms de fichiers sont corrects et éviter ainsi qu'un pirate puisse lire votre fichier de mots de passe.

Recette 2 : Alternier les couleurs des lignes d'un tableau

Si vous devez présenter un grand nombre d'informations sous la forme de lignes d'un tableau, comme les sujets d'un forum de discussion ou les articles d'un panier virtuel, les différentes lignes seront bien plus faciles à lire si chacune d'elles a une couleur légèrement différente de celles des lignes qui l'entourent. La première étape consiste à définir les couleurs des lignes de tableau dans votre feuille de style.

```
tr.lig1 {
    background-color: gray;
}

tr.lig2 {
    background-color: white;
}
```

Ici, on a défini deux classes de style, `lig1` et `lig2`, pour les balises de lignes de tableau (`<tr>`). Vous pouvez les intégrer dans un fichier CSS inclus par votre document ou les placer entre les balises `<style>` et `</style>` dans la partie `<head>` du document.

Définissons maintenant une fonction qui retourne alternativement ces classes. Nous utiliserons une petite astuce consistant à passer par référence une variable entière à cette fonction, qui pourra ainsi modifier cette variable qui servira de bascule pair/impair :

```
function formater_ligne_tableau(&$cpteur_lig) {
    // Renvoie la classe de style pour une ligne
    if ($cpteur_lig & 1) {
        $couleur_lig = "lig2";
    } else {
        $couleur_lig = "lig1";
    }
    $cpteur_lig++;
    return $couleur_lig;
}
```

Voyons maintenant comment utiliser cette fonction. On crée d'abord une requête SQL pour obtenir quelques lignes de données à partir de la table décrite dans l'annexe, puis on lance le formatage du tableau :

```
$sql = "SELECT nom_produit FROM infos_produits";
$resultat = @mysql_query($sql, $db) or die;

echo "<table>";
```

Le reste est assez simple ; il suffit d'initialiser la variable bascule `$i`, d'appeler `formater_ligne_tableau($i)` sur chaque ligne afin d'obtenir des classes de style alternées et d'associer à chaque ligne du tableau la classe ainsi obtenue. Enfin, il reste à fermer le tableau :

```
$i = 0;
while($lig = mysql_fetch_array($resultat)) {
    /* Affichage du résultat */
    $classe_lig = formater_ligne_tableau($i);
    echo "<tr class=\"\$classe_lig\"><td>$lig[nom_produit]</td></tr>";
}
echo "</table>";
```

Il est essentiel de comprendre comment fonctionne `formater_ligne_tableau()` : il faut initialiser une variable entière pour représenter l'état, mais la valeur en elle-même n'a aucune importance puisque la fonction s'en occupe pour vous.

Amélioration du script

Vous pouvez faire un million de choses avec les feuilles de style. Pour plus d'informations sur les CSS, consultez l'une des nombreuses ressources en ligne consacrées à ce sujet ou lisez le livre d'Éric Meyer, *CSS par Éric Meyer* (Pearson, 2005). En utilisant une approche objet, vous pouvez aisément convertir cette fonction en une méthode de formatage de tableau : au lieu de déclarer explicitement une variable d'état, il suffit de créer une classe avec une variable privée chargée de mémoriser cet état. Le constructeur et le destructeur ouvrent et ferment, respectivement, les balises du tableau. Voici à quoi ressemblera cette classe :

```
class TableauAlterne {
    function __construct() {
        $this->etat = 0;
        print "<table>";
    }
    function __destruct() {
        print "</table>";
    }
    function affiche_ligne($ligne) {
        if ($this->etat & 1) {
            $couleur_lig = "lig2";
        } else {
            $couleur_lig = "lig1";
        }
        print "<tr class=\"\$couleur_lig\">";
        foreach ($ligne as $valeur) {
            print "<td>$valeur</td>";
        }
    }
}
```

```

    }
    print "</tr>";
    $this->etat++;
}
}

```

Voici comment utiliser cette classe (en supposant que la requête SQL est identique à celle de l'exemple précédent) :

```

$montableau = new TableauAlterne;
while($ligne = mysql_fetch_row($resultat)) {
    /* Affichage du résultat. */
    $montableau->affiche_ligne($ligne);
}
unset($montableau);

```

Au premier abord, vous pourriez penser que c'est une belle amélioration car elle est un peu plus simple à utiliser – il n'est plus nécessaire de se préoccuper de la variable d'état, la méthode `affiche_ligne()` peut gérer un nombre quelconque de colonnes et vous n'avez plus non plus besoin d'écrire le moindre code HTML pour ouvrir et fermer le tableau. C'est, en effet, un avantage indéniable si tous vos tableaux se ressemblent, mais cette apparente simplicité se paye en termes de souplesse et d'efficacité. Vous pourriez bien sûr ajouter des méthodes et des attributs pour effectuer des traitements supplémentaires, comme l'ajout d'en-têtes aux tableaux, mais demandez-vous si cela en vaut réellement la peine.

Recette 3 : Créer des liens Précédent/Suivant

Si vous devez afficher un grand nombre d'articles sur une page, il peut être souhaitable de découper cet affichage en plusieurs pages contenant, chacune, un nombre limité d'articles. Vos résultats seront ainsi plus faciles à lire et vous améliorerez le temps de chargement des pages.

Une barre de navigation permet aux utilisateurs de contrôler le chargement de ces pages. Vous avez besoin d'une barre comportant des liens *Précédent* et *Suivant*, ainsi que des liens permettant d'atteindre une page précise d'après son numéro. Voici un script qui prend tout cela en charge :

```

<?php

function creer_navbar($num_deb = 0, $articles_par_page = 50, $nbre) {
    // Création d'une barre de navigation
    $page_courante = $_SERVER["PHP_SELF"];

    if (($num_deb < 0) || (! is_numeric($num_deb))) {
        $num_deb = 0;
    }
}

```

```

$navbar = "";
$navbar_prec = "";
$navbar_suiv = "";

if ($nbre > $articles_par_page) {
    $cpteur_nav = 0;
    $nb_pages = 1;
    $nav_passee = false;

    while ($cpteur_nav < $nbre) {
        // Est-on sur la page courante ?
        if (($num_deb <= $cpteur_nav) && ($nav_passee != true)) {
            $navbar .= "<b><a href=\"\$page_courante?debut=$cpteur_nav\">
                [$nb_pages]</a></b>";

            $nav_passee = true;
            // Faut-il un lien "Précédent" ?
            if ($num_deb != 0) {
                $num_prec = $cpteur_nav - $articles_par_page;
                if ($num_prec < 1) {
                    $num_prec = 0;
                }
                $navbar_prec = "<a href=\"\$page_courante?debut=$num_prec \">
                    &lt;&lt;Précédent - </a>";
            }

            $num_suiv = $articles_par_page + $cpteur_nav;

            // Faut-il un lien "Suivant" ?
            if ($num_suiv < $nbre) {
                $navbar_suiv = "<a href=\" \$page_courante?debut=$num_suiv\">
                    - Suivant&gt;&gt; </a><br>";
            }
        } else {
            // Affichage normal.
            $navbar .= "<a href=\"\$page_courante?debut=$cpteur_nav\">
                [$nb_pages]</a>";
        }

        $cpteur_nav += $articles_par_page;
        $nb_pages++;
    }

    $navbar = $navbar_prec . $navbar . $navbar_suiv ;
    return $navbar;
}
?>

```

Supposons que vous utilisiez ce script pour traiter des informations provenant d'une base de données. Il y a deux points à respecter pour que l'affichage soit précis et efficace :

N'afficher qu'un sous-ensemble de lignes de votre base de données.

Si vous souhaitez n'afficher que 25 articles par page, écrivez une requête SQL qui ne renvoie que 25 lignes (vous *pourriez* tout récupérer, puis parcourir les milliers de lignes de résultats pour ne garder que les 25 qui vous intéressent, mais il existe des méthodes bien plus efficaces). En outre, vous voulez ne montrer que 25 articles *spécifiques*. Récupérer les 25 premiers articles de la base ne vous aidera pas si l'utilisateur veut consulter les produits correspondant aux articles 176 à 200.

Heureusement, la clause `LIMIT` de SQL permet d'atteindre aisément ces deux objectifs puisqu'elle permet de ne récupérer qu'un certain nombre de lignes dans une base de données. La requête SQL sera donc de la forme :

```
SELECT * FROM votre_table
WHERE conditions
LIMIT $num_deb, $articles_par_page
```

Lorsqu'elle est ajoutée à la fin d'une requête, la clause `LIMIT` n'extrait que le nombre indiqué de lignes de l'ensemble résultat. En mettant, par exemple, `LIMIT 75, 25` à la fin d'une requête, on ne récupère que 25 lignes de la base à partir de la 75^e ligne.

NOTE *Si le numéro de la ligne de début est supérieur au nombre de lignes disponibles, MySQL ne renvoie aucune donnée. Si, par exemple, vous utilisez une clause `LIMIT 200, 25` alors que la table ne contient que 199 lignes, vous obtiendrez un ensemble résultat vide, pas une erreur. Les programmes bien écrits prennent en compte les ensembles vides en faisant un test `if (mysql_num_rows($resultat) > 0)`.*

Maintenant que vous n'avez que les lignes que vous souhaitiez, il vous reste une chose à faire.

Compter le nombre total de lignes de l'ensemble résultat.

Ensemble résultat est le terme employé par SQL pour signifier "toutes les données qui ont été renvoyées après le traitement de la clause `WHERE`". On a besoin du nombre total de lignes pour savoir sur quelle page on se trouve et combien il reste de pages avant la fin. Sans lui, nous pourrions afficher un lien *Suivant* alors qu'il n'y a plus d'articles à voir et nous ne pourrions pas indiquer à l'utilisateur le nombre de pages qu'il lui reste à consulter.

La requête SQL devrait être de la forme :

```
SELECT count(*) AS nombre FROM votre_table
WHERE conditions
```

Lorsque vous disposez de cette information, vous pouvez greffer les trois informations dans la fonction `creer_navbar()`. Celle-ci contient plusieurs variables :

\$page_courante : La page courante qui contient la barre de navigation. Dans ce script, nous utilisons la variable spéciale `$_SERVER["PHP_SELF"]` qui contient toujours la page courante sans le nom d'hôte ni les éventuels paramètres GET. Dans un script accessible à l'URL <http://exemple.com/navbar.php?debut=0>, par exemple, `$page_courante` vaudrait `/navbar.php`.

\$num_deb : Le premier numéro de ligne. Si, par exemple, l'utilisateur examine les lignes 100 à 125, ce numéro vaudra 100. Il est passé à l'URL *via* un paramètre GET. Par défaut, il vaut 0.

\$articles_par_page : Le nombre de lignes affichées sur chaque page. Si, par exemple, il y a 100 lignes de données, une valeur de 25 pour cette variable produira quatre pages de 25 lignes chacune. Ces mêmes 100 lignes avec une valeur de 50 pour `$articles_par_page` produirait deux pages de 50 lignes. Si cette variable vaut 200, un ensemble de 100 lignes n'aura pas de barre de navigation puisque tout tient sur une seule page. La valeur par défaut de cette variable est 50.

\$nbre : Le nombre total de lignes. Nous avons déjà expliqué comment obtenir cette information dans "Compter le nombre total de lignes de l'ensemble résultat".

\$cpteur_nav : Cette variable part de 0 et s'incrémente de `$articles_par_page` jusqu'à ce qu'elle devienne supérieure à `$nbre`, auquel cas le script a atteint la fin de l'ensemble résultat – et donc la fin de la barre de navigation.

\$nb_pages : Le nombre de pages dans l'ensemble résultat.

\$nav_passee : Une variable temporaire qui passe à vrai dès que `$cpteur_nav` dépasse `$num_deb` ; en d'autres termes, nous sommes sur la page courante. Cela permet de mettre en évidence le numéro de la page courante lors de l'affichage de la barre de navigation.

Utilisation du script

```
<?php

$num_deb = intval($_GET("debut"));
$articles_par_page = 100;
$categorie_ventes = null;

if ($num_deb >= 0) {
```

```

// Compte les articles de la catégorie.
$sql = "SELECT count(*) AS nombre FROM infos_produits
WHERE categorie = 'chaussures'";
$resultat = @mysql_query($sql, $chaine_connexion)
or die("Erreur : " . mysql_error());
while ($ligne = mysql_fetch_array($resultat)) {
    $nbre = $ligne['nombre'];
}

// Récupère le résultat à afficher pour le client.
$sql = "SELECT num_produit, nom_produit FROM infos_produit
WHERE categorie = 'chaussures'
LIMIT $num_deb, $articles_par_page";
$resultat = @mysql_query($sql, $chaine_connexion)
or die("Erreur : " . mysql_error());
if (mysql_num_rows($resultat) > 0) {
    while($ligne = mysql_fetch_array($resultat)) {
        // Parcourt des lignes et ajoute des balises HTML
        // dans $categorie_ventes.
        $categorie_ventes .= "Données provenant de la requête SQL";
    }
} else {
    $categorie_ventes = "Aucun article n'appartient à cette catégorie.";
}

$navbar = creer_navbar($num_deb, $articles_par_page, $nbre);
}

if (is_null($categorie_ventes)) {
    echo "Entrée incorrecte.";
} else {
    echo "$navbar<br />$categorie_ventes";
}

?>

```

Cet exemple montre comment utiliser la fonction `creer_navbar()` avec un ensemble de données provenant d'une table SQL (le contenu de cette table est détaillé en annexe). Ce script fonctionne de la façon suivante :

1. Le premier numéro de ligne est extrait d'un paramètre GET et placé dans la variable `$num_deb`.
2. Une première requête SQL permet d'obtenir le nombre total de lignes concernées dans la table.
3. Une seconde requête extrait au plus `$articles_par_pages` lignes de la table, à partir de la ligne `$num_deb`.
4. Les données de la ligne sont formatées.

5. La fonction `creer_navbar()` crée la barre de navigation ; la barre formatée est placée dans la variable `$navbar`
6. Le script affiche la barre de navigation et les données formatées.

Recette 4 : Afficher le contenu d'un tableau

Supposons que vous ayez ajouté et ôté des éléments d'un tableau mais que ces modifications provoquent quelques problèmes. Vous voulez consulter le contenu du tableau à un instant donné du code. Pour cela, il existe une solution très simple, pourtant *souvent* méconnue de nombreux didacticiels PHP : la fonction `print_r()`. Voici un exemple :

```
<?php

$alacarte = array("crème glacée au chocolat",
                 "pudding à la vanille",
                 "fraises à la crème fouettée");
$menu = array ("amuse-gueule" => "fruit",
              "entrée" => "rosbif",
              "dessert" => $alacarte);

print_r($menu);

?>
```

La fonction `print_r()` affiche le tableau sous un format lisible, ce qui est particulièrement pratique lorsque l'on manipule des *tableaux à plusieurs dimensions* (tableaux contenant d'autres tableaux). Il suffit de passer à `print_r()` un tableau en paramètre et l'on obtient un affichage sous un format simple. Comme ce format utilise des espaces et pas de balises HTML, vous devrez probablement consulter le format source de la page pour visualiser le contenu du tableau tel qu'il est produit. L'exemple précédent affichera :

```
Array
(
    [amuse-gueule] => fruit
    [entrée] => rosbif
    [dessert] => Array
        (
            [0] => crème glacée au chocolat
            [1] => pudding à la vanille
            [2] => fraises à la crème fouettée
        )
)
```

Recette 5 : Transformer un tableau en variable scalaire qui pourra être restaurée ultérieurement

Bien que les tableaux soient très utiles, ils ne peuvent pas être utilisés partout. Vous ne pouvez pas les stocker dans un cookie ni dans une session, par exemple. En outre, MySQL et XML ne savent pas non plus gérer les tableaux de PHP.

Heureusement, il existe un moyen de transformer les tableaux PHP en chaînes qui, elles, peuvent être stockées n'importe où : la fonction `serialize()`. Voici un script qui illustre son fonctionnement (on suppose que le tableau `$alacarte` est le même que dans la section précédente) :

```
<?php

$menu = array(
    "amuse-gueule" => "fruit",
    "entrée" => "rosbif",
    "dessert" => $alacarte);

$menu_s = serialize($menu);

echo $menu_s;

?>
```

L'exécution de ce script donnera le résultat suivant :

```
a:3:{s:12:"amuse-gueule";s:5:"fruit";s:7:"entrée";s:6:"rosbif";s:7:"dessert";
a:3:{i:0;s:26:"crème glacée au chocolat";i:1;s:21:"pudding à la vanille";i:2;s:30:
"fraises à la crème fouettée";}}
```

Vous pouvez stocker cette valeur à peu près n'importe où – dans des cookies, des bases de données, des paramètres cachés POST etc. Pour effectuer la manœuvre inverse, il suffit d'utiliser la fonction `deserialize()` de la façon suivante :

```
$menu = deserialize($menu_s);
```

Problèmes éventuels

`serialize()` est une fonction très pratique, mais vous devez connaître certains points. Les tableaux sérialisés sont relativement faciles à lire : si vous stockez des tableaux contenant des données sensibles dans des cookies ou des sessions, il est donc préférable de les chiffrer auparavant pour éviter que vos données ne soient récupérées par des utilisateurs malhonnêtes (voir la recette n°23, "Chiffrer les données avec `Mcrypt`").

N'abusez pas non plus de cette fonction. Si vous constatez que vous stockez souvent des tableaux dans des bases de données ou que vous récupérez fréquemment des chaînes de sérialisation, vous devriez sûrement revoir la structure de votre base de données ou le mécanisme de stockage.

Enfin, évitez de passer des tableaux sérialisés dans des paramètres GET car la limite de la taille des URL est assez courte.

Recette 6 : Trier des tableaux à plusieurs dimensions

Une tâche de programmation éternelle consiste à trier des tableaux complexes comme celui-ci :

```
$articles = array(
    array("nom" => "Mojo Nixon", "prix" => 19.96, "quantité" => 2),
    array("nom" => "ABBA", "prix" => 14.99, "quantité" => 1),
    array("nom" => "Iced Earth", "prix" => 12.96, "quantité" => 4),
);
```

Vous voulez trier ce tableau en fonction de la clé `nom` de chaque sous-tableau. PHP, comme de nombreux autres langages de scripts, dispose d'une fonction de tri nommé `usort()` qui permet de trier en utilisant une fonction de comparaison définie par l'utilisateur. En d'autres termes, vous pouvez trier un tableau sur le critère de votre choix et vous n'avez pas besoin de vous soucier des mécanismes des algorithmes de tri. Avec `usort()`, un script de tri selon les noms se ramène donc au code suivant :

```
function compare_noms($a, $b) {
    return strcasecmp($a["nom"], $b["nom"]);
}

usort($articles, "compare_noms");
```

Si ce script semble si simple, c'est parce qu'il... l'est. La fonction de comparaison personnalisée s'appelle `compare_noms()` ; elle utilise une autre fonction de comparaison pour comparer les deux éléments qui lui sont passés en paramètres (`$a` et `$b`, ici).

- Pour être utilisable par `usort()`, `compare_noms()` doit simplement indiquer l'ordre relatif de `$a` par rapport à `$b` : Si `$a` est inférieur à `$b`, elle doit renvoyer `-1`.
- Si `$a` est égal à `$b`, elle doit renvoyer `0`.
- Si `$a` est supérieur à `$b`, elle doit renvoyer `1`.

Dans cette fonction de comparaison, `$a` et `$b` sont des tableaux dont nous voulons comparer les valeurs des clés `nom` ; nous devons donc comparer `$a["nom"]` et `$b["nom"]`. Nous pourrions aisément le faire en utilisant quelques instructions

if/then mais, ces valeurs étant des chaînes, il est préférable d'utiliser la fonction prédéfinie `strcasecmp()` pour des raisons de performance, de lisibilité et de fiabilité.

NOTE *PHP dispose d'une fonction `array_multisort()` qui se veut être une solution fourre-tout pour le tri des tableaux à plusieurs dimensions. Cependant, elle est vraiment très compliquée ; nous vous déconseillons de l'utiliser.*

Amélioration du script

La modification la plus évidente que vous pouvez apporter à la fonction de comparaison consiste à ajouter des critères lorsque les deux éléments sont égaux. En reprenant notre exemple précédent, voici comment trier d'abord sur les noms, puis sur les prix lorsque deux noms sont égaux :

```
function compare_noms($a, $b) {
    $r = strcasecmp($a["nom"], $b["nom"]);
    if ($r == 0) {
        if ($a["prix"] < $b["prix"]) {
            $r = -1;
        } elseif ($a["prix"] > $b["prix"]) {
            $r = 1;
        } else {
            $r = 0;
            /* Pas vraiment nécessaire puisque $r vaut déjà 0. */
        }
    }
    return($r);
}
```

Recette 7 : Créer des templates pour votre site avec Smarty

L'aspect des pages de la plupart des sites est cohérent. Bien que le contenu dynamique au sein d'une page puisse varier, elles ont généralement toutes un en-tête, une barre de navigation sur le côté et, éventuellement quelques publicités. Il existe des moyens simples d'obtenir ce résultat, allant de fonctions d'affichage personnalisées pour créer les en-têtes à l'inclusion de fichiers. Selon la taille du site, ces solutions peuvent suffire, mais plus le contenu grossit et se complique, plus il devient difficile de modifier l'aspect des pages.

Smarty (<http://smarty.php.net>) est la solution la plus connue pour créer des modèles (*templates*) en PHP. Avec lui, vous pouvez créer des modèles paramétrés : en d'autres termes, vous pouvez créer un unique fichier HTML contenant des balises d'emplacement pour les données produites par PHP. En outre, vous pouvez inclure d'autres templates Smarty dans un template, ce qui permet de mieux organiser et modifier les parties d'un site. Lorsque l'on connaît bien Smarty, on

peut mettre les données en cache afin d'améliorer considérablement les accès à un site, mais il s'agit d'une technique avancée qui dépasse le cadre de ce livre.

Installation de Smarty

Pour installer Smarty sur votre serveur, il suffit de suivre les étapes suivantes :

1. Créez un répertoire *smarty* dans l'arborescence du serveur web, afin d'y stocker les fichiers principaux de Smarty.
2. Téléchargez la dernière version de Smarty à partir de <http://smarty.php.net/>. Décompressez l'archive et extrayez-la dans un répertoire de votre ordinateur.
3. Transférez tous les fichiers Smarty de votre ordinateur vers le répertoire *smarty* du serveur.
4. Sur le serveur, créez un autre répertoire appelé *templates* pour y stocker vos templates Smarty. Créez deux sous-répertoires sous celui-ci : *html* pour les templates bruts et *compile* pour les templates compilés par Smarty.
5. Faites en sorte que le serveur puisse écrire dans le répertoire *compile*. Si vous ne savez pas comment faire, lisez la section "Permissions des fichiers", au début du Chapitre 7.
6. Dans le répertoire *templates*, créez (ou téléchargez) un fichier nommé *smarty_initialize.php*, contenant les lignes suivantes :

```
<?php

define ("SMARTY_DIR", "/chemin/vers/web/racine/smarty/");

require_once (SMARTY_DIR."Smarty.class.php");
$smarty = new Smarty;
$smarty->compile_dir = "/chemin/vers/web/racine/templates/compile";
$smarty->template_dir = "/chemin/vers/web/racine/templates/html";

?>
```

Il y a quatre aspects très importants dans le fichier *smarty_initialize.php* :

- La constante `SMARTY_DIR` pointe vers le répertoire de la bibliothèque Smarty.
- Pour charger la bibliothèque, *smarty_initialize.php* a besoin du fichier *Smarty.class.php*.
- Smarty étant orienté objet, vous devez créer un objet Smarty. C'est ce que fait `$smarty = new Smarty` dans ce script.
- Smarty doit savoir où se trouvent les templates bruts et compilés. Pour cela, il utilise deux attributs d'objets, `compile_dir` et `template_dir`.

Maintenant que Smarty a été configuré, il est temps d'apprendre à s'en servir.

Initiation rapide à Smarty

Comme l'indique le fichier *smarty_initialize.php* présenté dans la section précédente, les templates Smarty sont placés dans le répertoire *templates/html*. Ces templates sont formés de HTML classique, de JavaScript, de CSS et de tout ce qui peut être envoyé à un navigateur web. Supposons que vous ayez créé un template HTML dans un fichier *basic.tpl*. Pour que Smarty l'affiche, vous devez utiliser la méthode `display()` comme dans cet exemple :

```
<?php
require_once("smarty_initialize.php");
$smarty->display("basic.tpl");
?>
```

Pour bénéficier de toute la puissance de Smarty, vous devez placer les données produites par PHP dans un template. Pour créer une variable Smarty, utilisez des accolades avec un dollar (\$), comme dans cet exemple :

```
<HTML>
  <head>
    <title>{$var_titre}</title>
  </head>
  <body>
    Je m'appelle {$var_nom}.
  </body>
</HTML>
```

Ce template utilise deux variables, `$var_titre` et `$var_nom`. Pour les initialiser afin que Smarty affiche leurs valeurs, utilisez la méthode `assign()`:

```
<?php

require_once("smarty_initialize.php");

$titre_page = "Test des templates Smarty";
$nom_page = "William Steinmetz";

$smarty->assign("var_titre", $titre_page);
$smarty->assign("var_nom", $nom_page);

$smarty->display("basic.tpl");
?>
```

Vous pouvez affecter ces variables avec quasiment n'importe quel type de données PHP ; peu importe que ce soit des données MySQL, du code HTML provenant d'une autre source, des données fournies par l'utilisateur, etc. Si tout se passe bien, le code HTML ainsi produit affichera une page semblable à la Figure 1.1.



Figure 1.1 : Code HTML affiché dans un navigateur

Problèmes éventuels

Hormis les erreurs classiques dues à des mauvais chemins, le problème le plus fréquent est celui où le serveur web ne peut pas écrire dans le répertoire *compile* que vous avez défini pour stocker les templates compilés de Smarty. Si le serveur n'a pas les droits d'écriture et d'exécution sur ce répertoire, Smarty échouera lorsqu'il tentera de créer les nouveaux fichiers ; la page s'affichera, mais vous verrez apparaître un affreux message d'avertissement à la place de votre template.

Vous devez également vérifier que toutes les variables ont été affectées avec la méthode `$smarty->assign()` ; dans le cas contraire, Smarty leur affectera par défaut une chaîne vide. Cela ne provoquera pas d'erreur, mais l'aspect de votre page s'en ressentira et certains éléments pourront ne pas fonctionner comme prévu.

Amélioration du script

Pour mieux tirer parti de la puissance de Smarty, vous devez connaître quelques détails supplémentaires. Vous pouvez, par exemple, inclure plusieurs templates dans le même script :

```
<?php

// Initialisation de Smarty.
require_once("smarty_initialize.php");

// Affichage du premier template Smarty
$smarty->display("entete.tpl");
```

```
echo "Contenu de la page<br />";

// Affichage du second template Smarty.
$smarty->display("pied_page.tpl");
?>
```

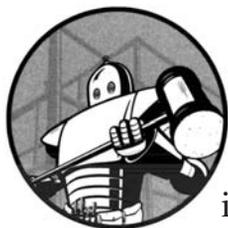
Parfois, il est préférable de décomposer les templates complexes en plusieurs parties. Si, par exemple, vous intégrez une bannière publicitaire dans un fichier *entete.tpl* qui change peu, vous pouvez préférer éditer cette bannière dans un fichier séparé *publicite.tpl* qui sera inclus dans le template *entete.tpl* :

```
<tr><td></td>
<td>
{include file="publicite.tpl"}
</td>
</tr>
```

Smarty est très puissant – vous pouvez, par exemple, construire des tableaux et les formater entièrement en affectant simplement un tableau PHP à une variable Smarty. Le langage dispose en outre de boucles if/then rudimentaires ; il permet de mettre des données en cache et fournit des fonctions de pré et de post-filtrage, etc. Maintenant que vous connaissez les rudiments du fonctionnement de Smarty, le mieux que vous ayez à faire est de consulter sa documentation en ligne pour en savoir plus.

2

CONFIGURATION DE PHP



À l'instar de tout logiciel, PHP utilise de nombreuses options de configuration qui affectent son fonctionnement. La plupart de ces options ne sont pas très significatives, mais un petit nombre d'entre elles sont importantes et tout programmeur doit les connaître.

En outre, de nombreuses extensions à PHP, appelés *bibliothèques*, ajoutent de nouvelles possibilités au langage. L'extension cURL, par exemple, permet à un serveur d'envoyer des données de formulaire à d'autres serveurs et de traiter les données qui lui reviennent. Mcrypt, quant à elle, est une extension qui permet d'effectuer un chiffrement sophistiqué pour stocker des données sensibles en toute sécurité.

Ce chapitre présente les options de configuration que les programmeurs PHP utilisent souvent et indique quand il faut les utiliser.

Les options de configuration et le fichier php.ini

De nombreux programmeurs débutants considèrent les options de configuration par défaut de PHP comme s'ils étaient des locataires emménageant dans un nouvel appartement : ils ont peur de faire des modifications de crainte de perdre leur caution. Vous devez plutôt considérer PHP comme votre propre maison :

si vous comptez y vivre longtemps, pourquoi ne pas réarranger le mobilier et supprimer un mur ou deux ?

NOTE *Selon l'hôte qui héberge votre serveur web, vous ne pourrez peut-être pas modifier vous-même les options de configuration. Cependant, les services d'hébergement sérieux voudront bien effectuer ces modifications pour vous et de nombreuses formules d'hébergement haut de gamme permettent de modifier les directives à l'aide d'un fichier de configuration spécifique à chaque utilisateur.*

Les options de configurations de PHP se trouvent dans un fichier nommé *php.ini*, que vous pouvez consulter et modifier avec n'importe quel éditeur de texte. Ces options sont classées en sections et sont de la forme :

```
max_execution_time = 30 ; Temps maximum d'exécution
max_input_time = 60      ; Temps maximum d'analyse d'une saisie
memory_limit = 8M       ; Mémoire maximum utilisable par un script
```

Pour configurer les options, utilisez le signe égal (=). Un point-virgule (;) indique le début d'un commentaire, mais il y a quelques exceptions car certaines options peuvent contenir des points-virgules. Si vous voulez modifier définitivement la valeur d'une option, sauvegardez d'abord le fichier *php.ini*, modifiez l'original et relancez Apache. Si vous préférez modifier les options script par script, utilisez la fonction `ini_set()`.

Trouver l'emplacement de votre fichier php.ini

Il peut être parfois un peu difficile de trouver l'emplacement du fichier *php.ini* sur un système, surtout s'il y a plusieurs installations de PHP qui cohabitent. Voici quelques moyens de le trouver :

- Sur les systèmes Unix, regardez dans `/usr/lib` ou `/usr/local/lib`. Il devrait se trouver dans le répertoire *lib* de la sous-arborescence où PHP a été installé.
- Avec Windows, essayez `C:\php` ou `c:\Program Files\PHP`.
- Appelez la fonction `phpinfo()` dans un script PHP (la section suivante donnera plus de détails). L'emplacement du fichier apparaît près du début avec l'étiquette *Configuration File (php.ini) Location*.
- Sur la plupart des systèmes Unix, la commande `find / -name php.ini` renvoie tous les noms de fichiers qui correspondent à *php.ini*.

NOTE *De nombreuses options de configuration ne se trouvent pas dans le fichier php.ini par défaut ; PHP utilise des valeurs par défaut lorsque celles-ci ne sont pas précisées. Vous trouverez la liste de ces valeurs par défaut à l'URL <http://www.php.net/manual/fr/ini.php>.*

Recette 8 : Afficher toutes les options de configuration de PHP

PHP possède de nombreuses fonctionnalités, mais elles ne sont pas toutes activées ou intégrées lors de l'installation. Pour connaître celles dont dispose votre installation, vous pouvez utiliser un script très simple qui vous dira tout ; cependant, ce script donne *tellement* d'informations qu'il provoque quasiment les pirates potentiels en leur disant "Hé, voici mes points faibles, utilisez-les pour pénétrer sur mon système". Par conséquent, n'oubliez pas de le supprimer dès que vous n'en avez plus besoin.

```
<?php
phpinfo();
?>
```

La fonction `phpinfo()` affiche tout ce que sait PHP de sa configuration, *absolument tout*. Non seulement elle indique l'état de chaque variable de la configuration, l'emplacement du fichier *php.ini* et la version de PHP, mais elle précise également la version du logiciel utilisé par le serveur web, les extensions qui ont été compilées avec PHP et l'API du serveur. Elle est donc très pratique pour examiner les options de configuration et s'assurer qu'une fonctionnalité a été installée et activée correctement.

Pour exécuter ce script, chargez la page dans votre navigateur web. N'oubliez pas de le supprimer lorsque vous avez terminé.

Recette 9 : Obtenir la valeur d'une option de configuration particulière

Dans certains cas, `phpinfo()` peut être un peu trop lourd par rapport à ce que l'on recherche. Vous voulez peut-être simplement savoir si les "apostrophes magiques" sont activées ou uniquement vérifier un chemin. En outre, `phpinfo()` ne vous aidera pas si vous écrivez un script qui se comporte d'une certaine façon lorsqu'une option est activée et d'une autre si elle ne l'est pas.

La fonction `ini_get()` permet d'obtenir la valeur d'une option de configuration particulière :

```
<?php
echo "register_globals vaut " . ini_get('register_globals');
?>
```

`ini_get()` renvoie la valeur de l'option dont on lui a passé le nom en paramètre. Cette valeur étant une valeur classique, elle peut donc être affichée, affectée à une variable, etc. Vous devez cependant savoir deux choses.

La première est que les valeurs booléennes, comme "false", sont généralement renvoyées sous la forme d'une chaîne vide lorsque vous les affichez. Si `register_globals` vaut "off", vous verrez donc probablement ce message :

```
register_globals vaut
```

Le second point est que les valeurs numériques sont souvent stockées sous forme raccourcie. Si `upload_max_filesize` vaut 8 192 octets, par exemple, la valeur renvoyée sera 8KB. De même, si la taille maximale d'un fichier déposé est de 2 Mo, `upload_max_filesize` ne vaudra pas 2 097 152 octets, mais 2MB.

Cela peut évidemment poser problème si vous effectuez des opérations arithmétiques sur ces nombres. La documentation officielle de PHP propose donc la fonction suivante pour convertir les formes abrégées K et M en véritables nombres :

```
function renvoie_octets($val) {
    $val = trim($val);
    $dernier = $val{strlen($val)-1};
    switch(strtoupper($dernier)) {
        case 'K':
            return (int) $val * 1024;
            break;
        case 'M':
            return (int) $val * 1048576;
            break;
        default:
            return $val;
    }
}
```

Recette 10 : Signaler les erreurs

Lorsque l'on programme, on peut très facilement oublier un nom de variable ou utiliser un code obsolète. Parfois, PHP est si compatissant envers l'utilisateur qu'il contourne de nombreuses erreurs de programmation classiques.

Il permet, par exemple, d'écrire un programme sans déclarer toutes les variables au début, ce qui est très pratique jusqu'à ce que vous orthographiez `$chaîne` comme `$chiane` et que le programme se plaigne d'une valeur nulle. Vous pouvez également passer des paramètres aux fonctions un peu n'importe comment et la plupart du temps cela fonctionnera malgré tout car PHP essaie de supposer ce que vous vouliez faire. Tout ira pour le mieux jusqu'à que ces suppositions soient incorrectes et que vous vous cassiez la tête à découvrir la raison de ce bogue mystérieux qui empêche votre programme de fonctionner.

Pour empêcher PHP de corriger automatiquement ces problèmes, vous pouvez activer le suivi des erreurs, ce qui remplira votre écran de messages à chaque fois que PHP détecte une erreur – quelle que soit sa gravité. Vous pourrez alors utiliser ces messages d’erreur pour corriger les potentiels trous de sécurité et découvrir les mauvaises variables de votre programme. Pour activer le suivi des erreurs, placez le code suivant au début d’un script :

```
<?php
error_reporting(E_ALL);
// Reste du script.
?>
```

L’activation du suivi des erreurs force PHP à afficher les messages d’erreur à l’écran avant de traiter le reste du programme (ce qui rend impossible la mise en place de cookies en cas d’erreur – ce n’est donc pas la peine d’essayer de modifier les valeurs de cookies la première fois que l’on active ce suivi).

Messages d’erreurs classiques

Il faut bien comprendre les trois messages d’erreurs les plus fréquents.

Notice: Undefined variable: var in script.php on line n

Ce message indique que vous utilisez une variable qui n’a pas été définie auparavant dans le script, ce qui peut arriver dans de nombreuses circonstances :

- Vous avez mal orthographié un nom de variable.
- La définition de la variable se trouve dans une structure conditionnelle, comme ici :

```
if ($fred == "Je m'appelle Fred") {
    $son_nom_est_fred = "yes";
}
```

- Vous avez concaténé une variable sans l’avoir déclarée explicitement.

Vous rencontrerez plus probablement le problème suivant lorsque vous tenterez d’utiliser un ancien code PHP dans votre programme :

Notice: Use of undefined constant k - assumed 'k' in script.php on line n

Ce message d’avertissement signifie généralement que vous avez passé une chaîne en paramètre à une fonction sans la mettre entre apostrophes. En d’autres termes, vous avez écrit quelque chose comme `strtolower(chaine)` au lieu de `strtolower("chaine")`.

Enfin, voici un message d'erreur classique qui apparaît lorsque l'on accède à un tableau :

```
Notice: Undefined index: i in script.php on line n
```

Il signifie que vous avez tenté de lire `$tableau[i]` alors qu'il n'y a pas d'indice `i` dans `$tableau`. Cette erreur survient souvent lorsque l'on tente de récupérer une valeur de formulaire à partir de `$_POST` ou `$_GET`, alors qu'aucun de ces tableaux ne comporte de valeur de ce nom. Cela signifie généralement que l'utilisateur n'a pas coché la bonne case ou le bon bouton radio (et ne lui a donc pas donné de valeur) ou que la variable n'a pas été passée à l'URL dans une requête GET.

Sur un site en production, il est préférable de désactiver ce suivi des erreurs afin que les utilisateurs n'en aient pas connaissance et ne vous cassent pas les pieds, mais également parce qu'il interfère avec les cookies, ce qui peut poser problème pour le suivi des sessions.

Recette 11 : Supprimer tous les messages d'erreur

Parfois, un script peut très bien fonctionner alors que PHP insiste pour corriger un problème. Par ailleurs, si quelque chose se passe mal, vous pouvez ne pas vouloir que les utilisateurs voient s'afficher un message d'erreur horrible (qui, en outre, révèle des informations que les pirates adorent connaître).

Heureusement, vous pouvez supprimer tous les messages d'erreur grâce à une option de configuration de *php.ini* :

```
display_errors = Off -
```

Vous devriez utiliser cette configuration dans un environnement de production afin que tout Internet ne puisse prendre connaissance des messages d'erreur de votre script. Si vous devez connaître ces messages pour résoudre des problèmes, utilisez plutôt l'option suivante pour que les messages soient inscrits dans le journal d'Apache :

```
log_errors = On
```

Si vous préférez, vous pouvez même envoyer les messages d'erreur à `syslog` ou dans un fichier en initialisant l'option `error_log` avec `syslog` ou en lui indiquant un nom de fichier.

Dans un environnement de développement, les choses sont différentes car vous souhaitez, au contraire, obtenir le plus possible de messages de diagnostic. Avec l'option `display_errors` activée, vous pouvez configurer l'option `error_reporting` de *php.ini* avec un champ de bits de votre choix (pour plus de détails,

consultez le fichier *php.ini* fourni avec votre distribution de PHP). Si un script pose vraiment problème dans votre environnement de développement et que vous voulez simplement le faire taire, utilisez la fonction suivante dans le script pour censurer ses messages d'erreur :

```
error_reporting(0);
```

Recette 12 : Allonger le temps d'exécution d'un script

Un jour, la société pour laquelle je travaille a changé ses paniers virtuels d'achat en ligne, ce qui m'a obligé à écrire un script pour convertir les 250 Mo de données correspondant aux produits de l'ancien panier au format du nouveau. Le script fonctionnait parfaitement mais il manipulait et transformait beaucoup de données, or PHP coupait son exécution après 30 secondes, bien avant qu'il ait le temps de finir. Ce n'est qu'après avoir découvert une option de configuration bien pratique que j'ai pu accomplir mon travail. La ligne suivante, ajoutée au début d'un script, lui permet de disposer de 240 secondes pour se terminer.

```
ini_set(max_execution_time, "240");
```

L'option de configuration `max_execution_time` précise le temps maximal octroyé à un script pour qu'il s'exécute. Passé ce délai, son exécution est automatiquement stoppée. N'abusez pas de cette option : si votre script ne peut pas s'exécuter en moins de quatre minutes, soit vous manipulez une base de données bien plus grosse que celles qu'utilisent la plupart des utilisateurs, soit votre script est très peu efficace, soit vous vous êtes trompé de langage de programmation.

Problèmes éventuels

Votre serveur tourne peut-être en mode sécurisé, ce qui désactive la modification de `max_execution_time`.

Vérifiez également votre code : vous avez peut-être une boucle infinie quelque part ou vous avez imbriqué une boucle dans une autre boucle qui ne fait rien.

Recette 13 : Empêcher les utilisateurs de déposer de gros fichiers

Pour empêcher vos utilisateurs de déposer les 70 Go du dernier volet de la "Guerre des étoiles" au format MPEG, vous pouvez fixer une taille maximale pour les fichiers que les utilisateurs sont autorisés à déposer sur votre serveur

(pour savoir comment traiter les fichiers déposés, consultez la recette n°54, "Déposer des images dans un répertoire").

```
upload_max_filesize = 500K
```

La taille maximale du fichier peut être indiquée de trois façons différentes :

- sous forme d'entier (qui représente le nombre total d'octets) ;
- sous forme d'un nombre suivi de M pour indiquer une taille en méga-octets, comme 2M (2 Mo) ;
- sous forme d'un nombre suivi de K pour indiquer une taille en kilo-octets, comme 8K (8 Ko).

Quel que soit le format choisi, les utilisateurs ne pourront plus, désormais, déposer un fichier d'une taille supérieure. Par défaut, cette limite est de 2 Mo.

Recette 14 : Désactiver les variables globales automatiques

PHP dispose d'une fonctionnalité historique malheureuse qui facilite l'accès aux paramètres GET et POST. Si, par exemple, il existe un paramètre POST nommé `mon_param`, PHP peut automatiquement l'extraire et le placer dans une variable nommée `$mon_param`. Malheureusement, il s'agit d'un risque énorme pour la sécurité car vous pouvez ainsi créer n'importe quelle variable globale et, si vous oubliez d'en initialiser une, l'utilisateur peut manipuler des parties vulnérables de votre script.

Pour désactiver cette fonctionnalité, il suffit de mettre à "Off" l'option `register_globals` du fichier *php.ini* de votre serveur :

```
register_globals = Off
```

Heureusement, cette fonctionnalité est désactivée à partir de la version 4.2 de PHP. Cependant, il s'agit d'un problème que vous devez toujours vérifier.

Recette 15 : Activer les apostrophes magiques

Les apostrophes magiques sont un outil pratique, utilisé par les serveurs pour se protéger contre les attaques par injection SQL (comme on l'explique dans la recette n°19, "Attaques par injection SQL"). Les apostrophes magiques transforment automatiquement toutes les variables provenant d'un formulaire en préfixant par un anti-slash toutes les apostrophes simples, les apostrophes doubles et les anti-slash qu'elles contiennent : "Il l'a dit" devient donc `"Il l\'a dit"`.

Si vous utilisez MySQL, vous devriez systématiquement désactiver les apostrophes magiques car elles sont la source de nombreux problèmes. Il existe des fonctions de contournement, comme `mysql_real_escape_string()` qui permet de

protéger les données lorsque vous désactivez les apostrophes magiques, mais vous devez encore prendre vos précautions afin d'éviter le doublement des anti-slash d'échappement. Cependant, faute de mieux, elles feront l'affaire. Voici comment les activer dans le fichier *php.ini* :

```
magic_quotes_gpc = 1
```

Problèmes éventuels

Si les apostrophes magiques ne sont pas activées, vous devrez utiliser `mysql_real_escape_string()` pour garantir que les données ont été protégées. Cependant, l'utilisation de cette fonction sur des données de formulaires alors que les apostrophes magiques sont également activées, provoquera le doublement des anti-slash de protection : `"I l 1\'a dit"` deviendra donc `\"I l 1\\'a dit\"`. Il vaut donc mieux être cohérent car, si vous n'y prenez pas garde, les anti-slash risquent de s'accumuler dans les chaînes des tables de votre base de données¹.

Recette 16 : Restreindre l'accès de PHP aux fichiers

Si vous avez peur qu'un script PHP malicieux accède aux fichiers de votre système (le fichier des mots de passe, par exemple), vous pouvez limiter les répertoires auxquels PHP a accès grâce à l'option `open_basedir`. Lorsqu'elle est activée, PHP ne peut ouvrir ou manipuler aucun fichier en dehors de ceux qui se trouvent dans les répertoires indiqués. Voici un exemple qui limite l'accès au répertoire */home/www* :

```
open_basedir = /home/www
```

Vous pouvez indiquer plusieurs répertoires en les séparant par un caractère deux-points (:) sous Unix ou un point-virgule (;) sous Windows.

NOTE *Par défaut, PHP autorise l'accès au répertoire indiqué, ainsi qu'à tous ses sous-répertoires. Si vous souhaitez que seul ce répertoire soit autorisé d'accès, ajoutez une barre de fraction à la fin du chemin, /home/www/ par exemple.*

Problèmes éventuels

Lorsqu'un utilisateur dépose un fichier, celui-ci est stocké dans un répertoire temporaire jusqu'à ce qu'il soit traité par un script. Ce répertoire étant traditionnellement éloigné du reste des fichiers PHP, vous ne devez pas oublier de l'ajouter à la liste de `open_basedir`.

1. NdT : Notez qu'il faut toujours désactiver les apostrophes magiques avec MySQL.

Recette 17 : Supprimer des fonctions précises

Supposons que vous ayez décidé que la fonction `exec()`, qui permet à un script PHP d'exécuter directement des commandes sur le serveur, est trop dangereuse. Afin de minimiser les risques de sécurité, vous pouvez désactiver une fonction PHP précise tout en autorisant les autres à fonctionner correctement. Voici comment désactiver un certain nombre de fonctions potentiellement dangereuses dans le fichier `php.ini` :

```
disable_functions = system, exec, passthru, shell_exec, proc_open
```

Recette 18 : Ajouter des extensions à PHP

Si vous êtes un développeur sérieux, vous finirez peut-être par atteindre les limites d'une installation de base de PHP. Bien que PHP intègre une foule de fonctionnalités, il ne sait pas, nativement, gérer le chiffrement des données, les graphiques, les documents XML ni l'accès à d'autres pages web.

Pour accomplir toutes ces opérations, PHP passe par un certain nombre d'extensions qui utilisent des bibliothèques tierces pour effectuer le véritable travail. Nous présentons ici quelques-unes des extensions les plus pratiques.

cURL

cURL permet à un serveur PHP d'accéder à d'autres sites web en envoyant et recevant des informations au moyen d'un protocole utilisant des URL (vous utiliserez le plus souvent HTTP, qui permet d'accéder à d'autres pages web et FTP, qui permet de déposer et de télécharger des fichiers). En pratique, cela signifie que vous pouvez faire en sorte que votre serveur imite un navigateur web, visite d'autres sites et télécharge leurs pages dans une variable de votre choix.

cURL est un outil essentiel pour tout panier virtuel d'achat en ligne car il permet de valider les versements par carte de crédit et de fournir en direct aux clients les détails de livraison. On utilise cURL pour se connecter et transmettre les données de la transaction au serveur d'une autre société ; celui-ci répond en indiquant si le versement a été accepté, rejeté et pourquoi.

Mcrypt

Pour des raisons de sécurité, toutes les données qui sont placées dans des cookies ou des sessions doivent être chiffrées et, si vous stockez des données sensibles comme des numéros de cartes bancaires ou autres informations personnelles, vous devez à tout prix vous assurer qu'elles ne peuvent pas être lues par une simple lecture de la base de données. Heureusement, la bibliothèque Mcrypt permet d'effectuer un chiffrement sophistiqué sans savoir quoi que ce soit des techniques de chiffrement ! (Nous expliquerons comment l'utiliser à la recette n°23, "Chiffrer les données avec Mcrypt".)

GD

La bibliothèque GD permet de créer des graphiques à la demande ou d'obtenir des informations sur une image. Elle peut créer des fichiers aux formats JPEG ou GIF lorsque vous avez besoin de produire des graphiques ou de manipuler des images existantes à ces formats pour produire des vignettes.

MySQL

Une compilation totalement nue de PHP ne sait pas accéder à une base de données. Heureusement, MySQL et PHP étant unis comme les doigts de la main, la plupart des serveurs web qui proposent PHP disposent également des bibliothèques MySQL : c'est la raison pour laquelle la plupart des programmeurs ignorent généralement que la fonction `mysql_connect()` fait partie d'une extension.

Parmi les autres extensions de PHP, citons SOAP (qui permet d'accéder aux services web), PDF et Verisign Payment Pro. Il pourrait sembler intéressant d'ajouter toutes ses extensions à PHP, mais n'oubliez pas que l'ajout d'une extension peut ralentir l'initialisation et ajouter des failles de sécurité. En outre, les extensions qui ne servent pas souvent ne sont pas aussi bien maintenues que les autres et peuvent donc ne jamais être mises à jour.

Ajouter des extensions PHP

Voyons maintenant comment installer ces extensions. La première étape consiste à savoir si l'extension que vous voulez ajouter est déjà présente dans votre installation.

Vérifier la présence d'une extension

Généralement, les serveurs web installent par défaut les extensions les plus utiles ; avant de vous lancer dans l'installation d'une extension particulière, il est donc préférable de vérifier si elle n'est pas déjà installée et chargée.

La méthode la plus simple pour effectuer ces tests consiste à utiliser la fonction `phpinfo()` décrite à la recette n°8, "Afficher toutes les options de configurations de PHP". Parcourez la page ainsi produite en recherchant votre bibliothèque. Si, par exemple, l'extension MySQL est activée, vous verrez une section comme celle-ci :

```
mysql
```

```
MySQL Support => enabled
```

```
...
```

Si cela ne fonctionne pas ou si vous pensez que c'est un peu lent, vous pouvez utiliser d'autres méthodes.

Une extension ajoute de nouvelles fonctions à PHP : cURL, par exemple, ajoute des fonctions comme `cURL_init()` et `cURL_setopt()`, Mcrypt ajoute `mcrypt_encrypt()` et `mcrypt_decrypt()`, etc. Supposons que Mcrypt ne soit pas installée : en ce cas, PHP ne connaît pas la fonction `mcrypt_decrypt()` – pour lui, ce n'est rien d'autre qu'une fonction non définie.

Vous pouvez en tirer parti à l'aide de la fonction PHP `function_exists()`. Essayez par exemple ce script qui détecte la présence de l'extension MySQL :

```
<?php
if (function_exists(mysql_connect)) {
    print 'MySQL est disponible';
} else {
    print "MySQL n'est pas disponible";
}
?>
```

Demander à son hébergeur de charger des extensions

Si ce n'est pas vous qui hébergez votre serveur web, comme c'est généralement le cas de nombreuses personnes, vous êtes à la merci de l'hébergeur. Comme vous n'avez pas d'accès root, vous ne pouvez pas installer vous-même les bibliothèques. En ce cas, vous devez demander aux administrateurs du serveur de le faire pour vous. Lorsque vous rédigez votre demande, soyez le plus précis possible ; sinon, vous risquez de ne pas avoir la bonne version, ni même l'extension que vous souhaitez.

Certaines sociétés le feront gracieusement alors que d'autres vous demanderont de payer pour être déplacé vers un serveur plus élaboré, avec plus d'extensions. D'autres encore vous répondront "Nous ne proposons pas d'installations personnalisées".

Si vous ne pouvez pas obtenir les extensions dont vous avez besoin, vous devrez faire sans ou migrer vers un autre hébergeur.

NOTE *Même si vous possédez votre propre serveur, il est souvent plus simple de demander au support technique d'installer les extensions. De cette façon, ils seront capables (en théorie, du moins) de réparer les problèmes si quelque chose se passe mal au cours de l'installation.*

Installer des extensions avec un panneau de contrôle web

Les serveurs loués proposent généralement un panneau de contrôle permettant d'effectuer les tâches classiques d'un webmestre, comme relancer Apache ou redémarrer le serveur, à partir d'un navigateur web.

Ces panneaux de contrôle permettent parfois de recompiler automatiquement Apache ou PHP en utilisant des cases à cocher ou des menus déroulants pour choisir les extensions à ajouter à PHP. WHM, par exemple, (un panneau de contrôle assez connu, bien qu'un peu bogué) propose "Update Apache", qui permet de réinstaller Apache et PHP avec les extensions souhaitées.

NOTE Si votre serveur ne dispose pas d'un panneau de contrôle, vous pouvez généralement demander son installation moyennant quelques euros.

Installer manuellement une extension

La recompilation de PHP est la méthode Unix pour réinstaller PHP en lui ajoutant les extensions voulues par la même occasion. Ceux qui ne connaissent pas l'administration d'Unix risquent donc d'être un peu décontenancés par cette approche.

Il est préférable de faire des tests sur un serveur Apache local bien distinct de celui d'un environnement de production car vous pouvez vraiment endommager le serveur. Vous devez donc vous assurer de pouvoir compter sur un support technique en cas de problème. Si vous ne vous sentez pas compétent pour cette opération, demandez de l'aide à quelqu'un capable de vous aider.

Pour qu'une bibliothèque fonctionne avec PHP, il faut franchir deux étapes : la première consiste à installer les bibliothèques pour les extensions, la seconde à faire en sorte que PHP reconnaisse ces extensions.

Installer les bibliothèques

Les étapes d'installation d'une extension dépendent de la bibliothèque utilisée. Nous vous présenterons rapidement ici une méthode pour le faire, mais vous devez consulter tous les didacticiels disponibles sur la page de téléchargement de la bibliothèque et lire tous les fichiers *README* avant de faire quoi que ce soit. Si vous vous voulez savoir comment tout cela se passe, vous trouverez une explication de la compilation des programmes dans le livre de Michael Kofler "*Linux*" (Pearson, 2008).

Les étapes générales sont les suivantes :

1. **Connectez-vous sur votre serveur sous le compte root ou sous le compte d'un utilisateur qui a le droit d'installer de nouveaux programmes.**
2. **Téléchargez le fichier archive de la bibliothèque dans le répertoire racine de votre serveur.** Une recherche rapide avec Google sur le nom de la bibliothèque et PHP (*mcrypt php*, par exemple) vous mènera généralement à la page d'accueil de cette bibliothèque, où vous trouverez les fichiers sources qui sont souvent archivés et compressés avec tar pour économiser l'espace disque. Les noms de ces fichiers archives sont généralement de la forme *nomficbiblio.tar.gz*.
3. **Extrayez le contenu de l'archive.** Une archive tar contient plusieurs fichiers et répertoires. Cette archive est elle-même compressée avec Gzip, d'où son extension *.gz* finale. Vous pouvez considérer un fichier *.tar.gz* comme un fichier *.zip*, sauf qu'il a été créé en deux étapes distinctes, par deux programmes différents.

Cependant, vous n'aurez pas besoin de lancer ces deux programmes puisque GNU tar sait comment lancer l'extracteur. Pour extraire le contenu du fichier archive, faites `tar zxvf nomficbiblio.tar.gz` à l'invite de commande. Vous verrez alors défiler la liste des fichiers et des répertoires en

cours d'extraction. La plupart de ces archives créent une arborescence sous un répertoire racine.

4. **Placez-vous dans le répertoire racine des sources de la bibliothèque en lançant `cd nomrepertoire`.** Si vous ne connaissez pas le nom de ce répertoire ou que vous ne l'avez pas noté à l'étape précédente, sachez qu'il porte généralement le nom de la bibliothèque – *cURL*, par exemple (sous Unix, les noms des fichiers et des répertoires étant sensibles à la casse, *cURL* est différent de *CURL*).

5. **Lancez la commande `configure` pour vérifier que tous les ingrédients nécessaires à l'installation sont présents sur la machine.** En raison du grand nombre de variantes Unix, l'installation d'un logiciel nécessite un certain savoir-faire ; heureusement, la commande `configure` se charge de l'essentiel du travail en examinant la configuration du serveur et en déduisant les valeurs correctes pour l'installation du programme. Tapez `./configure` à l'invite de commande.

Certaines extensions nécessitent de passer des options à la commande `configure` pour pouvoir fonctionner correctement. Mcrypt, par exemple, exige la commande `./configure --disable-nls --disable-posix-threads` pour garantir la compatibilité avec Apache. Ces options étant spécifiques à chaque extension, consultez les didacticiels et les fichiers README pour savoir si la bibliothèque que vous voulez installer a besoin d'une option particulière.

6. **Compilez et installez le paquetage.** Avec Unix, l'utilitaire `make` permet d'effectuer ces deux opérations. Lancez d'abord `make` pour compiler le paquetage : vous verrez défiler la liste des commandes qui construisent le logiciel à mesure qu'elles sont exécutées. Puis, faites `make check` pour lancer les tests fournis avec le paquetage (parfois, il peut ne pas y en avoir). Enfin, faites `make install` pour installer l'extension.

7. **Créez un script `phpinfo()`.** Vous pensiez avoir fini ? Désolé, mais vous venez simplement d'installer l'extension sur votre serveur. Vous devez maintenant réinstaller PHP et lui indiquer où se trouve l'extension et comment l'utiliser.

`phpinfo()` (voir la recette n°8, "Afficher toutes les options de configuration de PHP") vous informe sur la configuration complète du serveur. Aux alentours du premier tiers de la première page produite par le script, vous trouverez une section intitulée *Configure Command*, qui contient une liste assez cryptique, de la forme :

```
'./configure' '--with-apxs=/usr/local/apache/bin/apxs'  
'--with-xml' '--enable-bcmath' '--enable-calendar' '--enable-ftp'  
'--enable-magic-quotes' '--with-mysql' '--enable-discard-path'  
'--with-pear' '--enable-sockets' '--enable-track-vars'  
'--enable-versioning' '--with-zlib'
```

Si vous voulez réinstaller PHP dans la même configuration que celle actuelle, vous devrez lancer cette commande après avoir ôté les apostrophes autour de la commande configure. Vous obtiendrez donc :

```
./configure '--with-apxs=/usr/local/apache/bin/apxs' '--with-xml'  
'--enable-bcmath' '--enable-calendar' '--enable-ftp'  
'--enable-magic-quotes' '--with-mysql' '--enable-discard-path'  
'--with-pear' '--enable-sockets' '--enable-track-vars'  
'--enable-versioning' '--with-zlib'
```

Si vous ajoutez une extension comme GD, vous ne voudrez pas pour autant perdre celles qui sont déjà installées. Pour cela, copiez ces informations de configuration dans un fichier et ajoutez à la fin les options `--with` adéquates. Si, par exemple, vous voulez ajouter Mcrypt à ce serveur, il suffit d'ajouter l'option `--with-mcrypt` à la fin de la commande précédente. Pour connaître l'option `--with` appropriée, consultez la documentation de l'extension.

NOTE *Si vous avez écrasé la structure arborescente de l'archive tar et que vous avez placé la bibliothèque dans un répertoire autre que celui par défaut, vous devez ajouter ce chemin avec une option `--with` afin que PHP puisse la trouver. C'est ce qui s'est passé avec la bibliothèque `apxs` (Apache Extension Tool Synopsis) dans l'exemple précédent : `--with-apxs=/usr/local/apache/bin/apxs` indique à PHP que `apxs` se trouve dans le répertoire `/usr/local/apache/bin/apxs`.*

8. **Récupérez et décompactez une nouvelle distribution source de PHP puis allez dans son répertoire.** Vous pouvez décompacter le code source de PHP exactement comme vous l'avez fait précédemment pour le code source de la bibliothèque. Si vous disposez déjà d'une arborescence source de PHP, vous pouvez l'utiliser à condition de lancer d'abord la commande `make clear`.
9. **Copiez la commande configure que vous aviez stocké plus haut dans un fichier, collez-la sur la ligne de commande et pressez Entrée pour l'exécuter.** Cela reconfigurera PHP avec la nouvelle bibliothèque en conservant les anciennes.
10. **Compilez et installez PHP** en lançant les commandes `make` puis `make install`. Ces commandes peuvent prendre un certain temps puisqu'elles compilent et installent chaque composant de PHP.

NOTE *Si vous avez modifié vos fichiers `.ini` comme on l'a expliqué plus haut, ils seront peut-être écrasés par les valeurs par défaut lors de la recompilation de PHP. Par conséquent, vérifiez que vos réglages ont été préservés.*

11. **Relancez Apache** en lançant la commande `apachectl graceful`.
12. **Testez PHP.** Essayez d'abord avec un bon vieux script "Hello world!" afin de vous assurer que PHP fonctionne sans erreur, puis expérimentez quelques fonctions spécifiques à la bibliothèque que vous venez d'installer pour vérifier qu'elle fonctionne.

Problèmes éventuels

Il y a tant de problèmes qui peuvent survenir au cours de la compilation qu'il est presque impossible de tous les énumérer. Bien que de nombreuses erreurs soient compliquées et que beaucoup soient spécifiques à une bibliothèque, on retrouve systématiquement trois problèmes classiques.

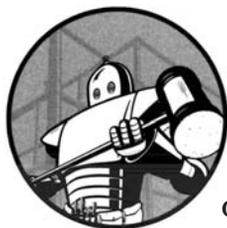
Le premier peut survenir si les outils de développement ne sont pas installés sur votre système. Vous avez besoin du compilateur C et de nombreuses bibliothèques de développement pour pouvoir compiler un programme.

Le second est que vous devrez peut-être configurer PHP avec une option `--with` associée à un chemin précis : `--with-mcrypt=/usr/lib/mcrypt`, par exemple.

Le troisième problème peut provenir du fait que vous avez mal configuré la bibliothèque de l'extension. Comme indiqué plus haut, vous devez configurer Mcrypt avec les options `--disable-nls` et `--disable-posix-threads` sous peine de poser des problèmes à Apache. Les autres bibliothèques peuvent également nécessiter ce type d'ajustement pour fonctionner correctement avec Apache et PHP. Consultez les FAQ, les pages de manuel et les fichiers *README* pour plus de détails.

3

SÉCURITÉ ET PHP



Si vous développez des services web, ne négligez jamais la sécurité en écrivant des programmes. Si la plupart des scripts PHP ne tiennent pas compte des problèmes de sécurité, c'est essentiellement dû au fait que ce langage est utilisé par un grand nombre de programmeurs inexpérimentés.

En ce qui vous concerne, ne prenez pas ce sujet à la légère et mettez en place une politique de sécurité robuste, en examinant minutieusement le rôle de vos scripts. Lorsque vous ajouterez des intérêts financiers sur votre serveur, il est probable que des personnes essaieront de le pirater. Créez un programme de gestion de forums ou de panier virtuel et la probabilité des attaques grandira à coup sûr.

Voici quelques conseils généraux pour la sécurité.

Ne faites pas confiance aux formulaires.

Il est très facile de pirater des formulaires. Bien sûr, en utilisant une simple astuce en JavaScript, vous pouvez faire en sorte qu'un formulaire n'accepte que des nombres compris entre 1 et 5 dans un champ donné. Mais si un visiteur désactive JavaScript dans son navigateur ou envoie des données de formulaire adaptées, votre validation côté client tombe à l'eau.

Les utilisateurs interagissent essentiellement avec vos scripts au moyen de paramètres, qui constituent donc un risque de sécurité majeur. La leçon qu'il faut en tirer est qu'*un script doit toujours vérifier les données qui lui sont transmises*. La section "Stratégies de vérification", plus loin dans ce chapitre, montre comment vérifier des données discrètes. Nous vous montrerons comment détecter et comment vous protéger contre les attaques XSS (*cross-site scripting*), qui peuvent détourner les autorisations de vos utilisateurs à leur profit (voire plus). Nous verrons également comment empêcher les attaques par injection SQL qui peuvent perturber ou supprimer vos données.

Ne faites pas confiance aux utilisateurs.

Supposez que toutes les données reçues par votre site web sont chargées de code malicieux. Nettoyez chaque donnée, même si vous pensez que personne n'attaquera jamais votre site. La paranoïa a parfois du bon.

Désactivez les variables globales automatiques.

Le plus gros trou de sécurité consiste à activer l'option de configuration `register_globals`. Heureusement, elle est désactivée par défaut à partir de la version 4.2 de PHP. Consultez la recette n°14, "Désactiver les variables globales automatiques" pour savoir comment faire.

Les programmeurs débutants considèrent les variables globales automatiques comme un outil très pratique, sans réaliser le danger qu'elles représentent. Un serveur avec cette option activée affecte automatiquement des variables globales avec n'importe quelle valeur de formulaire. Étudions un exemple pour comprendre leur fonctionnement et leur danger.

Supposons que vous utilisiez un script nommé *traitement.php* qui stocke dans votre base de données des utilisateurs les valeurs saisies dans un formulaire. Ce formulaire est de la forme :

```
<input name="nom_utilisateur" type="text" size="15" maxlength="64">
```

Lorsqu'il exécute *traitement.php* et que les variables globales automatiques sont activées, PHP place automatiquement la valeur de ce paramètre dans la variable `$nom_utilisateur`. Cela permet d'économiser un peu de saisie par rapport à un accès explicite *via* `$_POST['nom_utilisateur']` ou `$_GET['nom_utilisateur']`. Malheureusement, cela ouvre également une brèche dans la sécurité puisque PHP crée une variable pour toute valeur envoyée au script par un paramètre GET ou POST. Ceci pose un problème manifeste si vous n'initialisez pas explicitement la variable et que vous ne souhaitez pas que quiconque puisse la manipuler.

Étudiez par exemple le script suivant : si la variable `$autorise` vaut vrai, il montre des données confidentielles à l'utilisateur. Dans des circonstances normales, cette variable ne vaut vrai que si l'utilisateur s'est correctement authentifié *via* une hypothétique fonction `authentification_`

utilisateur(), mais si les variables globales automatiques sont activées, n'importe qui peut envoyer un paramètre GET comme `autorise=1` pour contourner cette protection :

```
<?php
// Définit $autorise = true uniquement si l'utilisateur s'est
// authentifié.
if (authentification_utilisateur()){
    $autorise = true;
}
?>
```

La morale de cette histoire est que vous devez suivre la procédure de la recette n°25, "Récupérer en toute sécurité les valeurs des formulaires" pour accéder aux données des formulaires. Il pourrait sembler que ce type d'attaque n'arrive pas souvent mais l'exploitation des variables globales automatiques vient en second après les attaques XSS. En outre, les variables globales posent des problèmes plus subtils qui peuvent plus tard causer des bogues dans votre code.

Options de configuration recommandées pour la sécurité

Plusieurs options de configuration de PHP affectent la sécurité. Voici celles que j'utilise pour les serveurs en production (le Chapitre 2 explique comment les configurer) :

- `register_globals` à off. Voir la discussion précédente.
- `safe_mode` à off. Cette option ne sécurise absolument rien.
- `error_reporting` à 0. Lorsqu'elle est activée, cette option affiche le rapport d'erreurs dans le navigateur de l'utilisateur. Pour les serveurs en production, il est préférable d'inscrire ces messages dans un journal à travers l'option de configuration `error_log` (voir la recette n°11, "Supprimer tous les messages d'erreur", au Chapitre 2). Les serveurs de développement peuvent activer cette option s'ils se trouvent derrière un pare-feu.
- Désactivez les fonctions `system()`, `exec()`, `passthru()`, `shell_exec()`, `proc_open()` et `popen()`. Voir la recette n°17, "Supprimer des fonctions précises".
- `open_basedir` initialisée avec le répertoire `/tmp` (pour pouvoir stocker les informations de session) et le répertoire racine du serveur web pour que les scripts ne puissent pas accéder à l'extérieur de cette arborescence.
- `expose_php` à off. Lorsqu'elle est activée, cette option ajoute aux en-têtes d'Apache une signature PHP qui contient le numéro de version de l'interpréteur. Pourquoi divulguer cette information ?
- `allow_url_fopen` à off. Cette option n'est pas réellement nécessaire si vous faites attention à la façon dont vous accédez aux fichiers à partir de vos scripts, c'est-à-dire si vous vérifiez toutes les données qui vous sont transmises.

- `allow_url_include` à off. Il n'y a vraiment aucune raison pour accéder à des fichiers inclus *via* HTTP.

En règle générale, vous ne devez pas faire confiance à un code qui veut utiliser ces fonctionnalités. Faites tout particulièrement attention à tout ce qui tente d'utiliser une fonction comme `system()` : un tel code est la plupart du temps une source de problèmes.

Ces options de configuration étant désormais correctement initialisées, études quelques attaques spécifiques, ainsi que les méthodes qui permettront à votre serveur de s'en protéger.

Recette 19 : Attaques par injection SQL

Les requêtes que passe PHP aux bases de données MySQL étant écrites en SQL, un pirate peut utiliser les paramètres de formulaire pour tenter une *attaque par injection SQL*. En insérant des fragments de code SQL malicieux dans ces paramètres, ce pirate tentera de pénétrer sur votre serveur (ou de le rendre indisponible).

Supposons que la valeur d'un paramètre de formulaire soit placée dans une variable nommée `$produit` et que vous avez créé une requête SQL comme celle-ci :

```
$sql = "select * from infos_produits where num_produit = '$produit';"
```

Si ce paramètre vient directement du formulaire, utilisez les protections proposées par la base de données en passant par des fonctions PHP, comme ici :

```
$sql = 'select * from infos_produits
      where num_produit = ''' . mysql_real_escape_string($produit)
      . ''';'
```

Si vous ne le faites pas, un pirate peut glisser ce fragment de code dans le paramètre du formulaire :

```
'39'; DROP infos_produits; SELECT 'TRUC'
```

Le contenu de `$sql` serait alors :

```
select * from infos_produits where num_produit = '39';
DROP infos_produits; SELECT 'TRUC'
```

En SQL, le point-virgule étant un séparateur d'instructions, la base de données exécutera donc ces trois instructions :

```
select * from infos_produits where num_produit = '39'  
DROP infos_produits  
SELECT 'TRUC'
```

Et votre table a disparu...

Notez que cette syntaxe ne fonctionnerait pas avec PHP et MySQL car la fonction `mysql_query()` n'autorise le traitement que d'une seule instruction par requête. Cependant, une sous-requête pourrait marcher.

Pour empêcher les attaques par injection SQL, vous devez prendre deux mesures :

- Vérifiez toujours tous les paramètres. Si vous attendez un nombre, par exemple, assurez-vous qu'il s'agit bien d'un nombre.
- Utilisez toujours la fonction `mysql_real_escape_string()` sur les données afin de désactiver les apostrophes simples et doubles dans celles-ci.

NOTE *Pour protéger automatiquement toutes les données des formulaires, vous pouvez activer les apostrophes magiques, comme on l'a expliqué dans la recette n°15, "Activer les apostrophes magiques". Sachez toutefois que cette option de configuration est fortement déconseillée dans les dernières versions de PHP (http://fr3.php.net/magic_quotes) et que vous devriez privilégier des solutions de contournement, comme nous l'avons vu à la recette n°15.*

Vous pouvez éviter plusieurs problèmes en limitant les privilèges de l'utilisateur MySQL. Tout compte MySQL peut, en effet, être limité à un certain type de requêtes sur les tables sélectionnées. Vous pourriez, par exemple, créer un utilisateur n'ayant que le droit de consulter les lignes de ces tables. Cependant, ce n'est guère utile pour des données dynamiques et, en outre, cela n'empêchera pas l'accès à des données confidentielles. Un utilisateur ayant accès à des données de connexion pourrait, par exemple, injecter du code lui permettant d'accéder à un autre compte que celui qui est affecté à la session courante.

Recette 20 : Empêcher les attaques XSS basiques

XSS signifie *cross-site scripting*. À la différence de la plupart des attaques, celle-ci fonctionne côté client. La forme la plus basique de XSS consiste à placer du code JavaScript dans un contenu que l'on soumet *via* un formulaire afin de voler les données du cookie d'un utilisateur. La plupart des sites utilisant des cookies et des sessions pour identifier leurs visiteurs, ces données volées peuvent servir à usurper l'identité des utilisateurs, ce qui est assez pénible pour un compte classique et carrément désastreux s'il s'agit d'un compte administrateur. Si votre site n'utilise ni cookies, ni identifiants de sessions, vos utilisateurs ne seront pas vulnérables à ce type d'attaque, mais vous devez tout de même savoir comment elle fonctionne.

À la différence des attaques par injection SQL, les attaques XSS sont difficiles à contrer ; d'ailleurs, Yahoo!, eBay, Apple et Microsoft ont tous été les proies de

ces attaques. Bien que XSS ne soit pas lié à PHP, vous pouvez nettoyer les données utilisateurs avec PHP afin d'empêcher les attaques. Pour ce faire, vous devez restreindre et filtrer les données qui sont soumises par les utilisateurs. C'est pour cette raison que la plupart des sites de forums en ligne n'autorisent pas les balises HTML dans les articles et les remplacent par des balises personnalisées comme [b] et [linkto].

Examinons un script simple, qui illustre comment se prémunir contre certaines de ces attaques. Pour une solution plus complète, il est préférable d'utiliser SafeHTML qui sera présenté plus loin dans ce chapitre.

```
function transforme_HTML($chaine, $longueur = null) {
// Aide à empêcher les attaques XSS

// Suppression des espaces inutiles.
$chaine = trim($chaine);

// Empêche des problèmes potentiels avec le codec Unicode.
$chaine = utf8_decode($chaine);

// HTMLise les caractères spécifiques à HTML.
$chaine = htmlentities($chaine, ENT_NOQUOTES);
$chaine = str_replace("#", "&#35;", $chaine);
$chaine = str_replace("%", "&#37;", $chaine);

$longueur = intval($longueur);
if ($longueur > 0) {
    $chaine = substr($chaine, 0, $longueur);
}
return $chaine;
}
```

Cette fonction transforme les caractères spécifiques à HTML par des littéraux HTML. Un navigateur affichera donc le code HTML passé par ce script comme du texte sans balise. Considérons, par exemple, cette chaîne HTML :

```
<STRONG>Texte en gras</STRONG>
```

Normalement, ce code HTML devrait s'afficher de la façon suivante :

Texte en gras

Mais, modifié par `transforme_HTML()`, il s'affichera comme la chaîne initiale car les caractères des balises sont devenues des entités HTML dans la chaîne traitée. En effet, la chaîne renvoyée ici par la fonction sera :

```
&lt;STRONG&gt;Texte en gras&lt;/STRONG&gt;
```

L'essentiel de cette fonction est l'appel à la fonction `htmlentities()` qui transforme les `<`, `>` et `&` en leurs entités équivalentes, `<`, `>` et `&`. Bien que cela règle les attaques les plus classiques, les pirates XSS plus expérimentés ont plus d'un tour dans leur sac : ils encodent leurs scripts malicieux en hexadécimal ou en UTF-8 au lieu de l'ASCII afin de contourner nos filtres. Ils peuvent ainsi envoyer le code sous la forme d'une variable GET de l'URL qui indique "C'est du code hexadécimal, mais pourriez-vous quand même l'exécuter pour moi?". Un exemple hexadécimal a cet aspect :

```
<a href="http://host/a.php?variable=%22%3e %3c%53%43%52%49%50%54%3e%43%6f%64%65%4d%65%63%68%61%6e%74%3c%2f%53%43%52%49%50%54%3e">
```

Mais, lorsque le navigateur affiche cette information, elle se transforme en :

```
<a href="http://host/a.php?variable="> <SCRIPT>CodeMechant</SCRIPT>
```

C'est pour empêcher cette astuce que `transforme_HTML()` ajoute des étapes supplémentaires pour convertir les symboles `#` et `%` en leurs entités correspondantes, ce qui bloque les attaques hexadécimales. Il fait de même pour convertir les données encodées en UTF-8.

Enfin, pour empêcher que quelqu'un essaie de surcharger une chaîne avec une saisie très longue en espérant casser quelque chose, vous pouvez ajouter un paramètre `$longueur` facultatif pour couper la chaîne à une longueur maximale de votre choix¹.

Recette 21 : Utiliser SafeHTML

Le problème du script précédent est sa simplicité et le fait qu'il n'autorise aucun type de balise. Malheureusement, il y a des centaines de façons pour faire passer du JavaScript à travers des filtres et, à moins de supprimer toutes les balises HTML des valeurs qui nous sont transmises, il n'y a aucun moyen de l'empêcher.

Actuellement, aucun script ne peut donc garantir qu'il ne peut pas être corrompu par ce type d'attaque, bien que certains soient mieux armés que d'autres. Comme nous le verrons plus en détail à la section "Stratégies de vérification", au Chapitre 4, il existe deux approches pour la sécurité : les listes blanches et les listes noires, les premières étant moins complexes à mettre en œuvre et plus efficaces.

1. NdR : Méfiez-vous des scripts clés en main qui s'installent en des millions d'exemplaires sur des serveurs web.

L'une des solutions reposant sur les listes blanches s'appelle SafeHTML, un analyseur anti-XSS écrit par PixelApes.

SafeHTML est suffisamment malin pour reconnaître du code HTML correct et peut donc chasser et supprimer toutes les balises dangereuses. Cette analyse est réalisée par un autre paquetage, HTMLSax.

Pour installer et utiliser SafeHTML, suivez les étapes suivantes :

1. Téléchargez la dernière version de SafeHTML.
2. Placez les fichiers dans un répertoire *classes* sur votre serveur. Ce répertoire contiendra tous les fichiers nécessaires au fonctionnement de SafeHTML et HTMLSax.
3. Incluez le fichier classe de SafeHTML (*safehtml.php*) dans votre script.
4. Créez un objet SafeHTML appelé `$safehtml`.
5. Nettoyez vos données avec la méthode `$safehtml->parse()`.

Voici un exemple complet :

```
<?php
/* Si vous avez stocké HTMLSax3.php et safehtml.php dans le répertoire
   /classes, définissez XML_HTMLSAX3 comme une chaîne vide. */
define(XML_HTMLSAX3, '');
// Inclusion du fichier classe.
require_once('classes/safehtml.php');
// Création d'un exemple de code incorrect.
$donnees = "Ce contenu lèvera une alerte <script>alert('Attaque XSS')</script>";
// Création d'un objet safehtml.
$safehtml = new safehtml();
// Analyse et nettoyage des données.
$donnees_sures = $safehtml->parse($donnees);
// Affichage du résultat.
echo 'Les données propres sont <br />' . $donnees_sures;
?>
```

Si vous voulez nettoyer d'autres données de votre script, il n'est pas nécessaire de créer un nouvel objet. Il suffit de réutiliser tout au long du script la méthode `$safehtml->parse()`.

Problèmes éventuels

L'erreur la plus importante que vous puissiez faire est de supposer que cette classe empêche totalement les attaques XSS. SafeHTML est un script assez complexe qui vérifie quasiment tout, mais rien n'est garanti. Vous devrez tout de même effectuer une validation adaptée à vos besoins des données qui vous sont transmises. Cette classe, par exemple, ne teste pas la longueur d'une variable pour vérifier qu'elle tiendra dans le champ d'une table. Elle ne vous protège pas non plus contre les problèmes de débordement de tampon.

Les pirates XSS ont de l'imagination et utilisent un grand nombre d'approches pour tenter d'atteindre leurs objectifs. Il suffit de lire le didacticiel XSS de RSnake (<http://ha.ckers.org/xss.html>) pour se rendre compte du nombre de façons de faire passer du code à travers les filtres. Le projet SafeHTML regroupe de bons programmeurs qui prennent sur leur temps libre pour essayer d'endiguer les attaques XSS et l'approche choisie est fiable, mais on ne peut pas être certain qu'un pirate ne trouvera pas une nouvelle technique pour court-circuiter ses filtres.

NOTE La page <http://namb.la/popular/tech.html> donne un exemple de la puissance des attaques XSS en montrant comment créer pas à pas le ver JavaScript qui a surchargé les serveurs de MySpace.

Recette 22 : Protéger les données avec un hachage non réversible

Ce script effectue une transformation des données dans un seul sens – en d'autres termes, il peut créer un hachage d'un mot de passe mais il est impossible de revenir au mot de passe initial à partir du hachage. On comprend tout l'intérêt de cette technique pour le stockage des mots de passe : l'administrateur n'a pas besoin de connaître les mots de passe des utilisateurs – il est préférable que seul l'utilisateur connaisse le sien. Le système (et uniquement lui) devrait pouvoir identifier un mot de passe correct : c'est d'ailleurs comme cela que le modèle des mots de passe Unix fonctionne depuis des années. La sécurité des mots de passe non réversibles fonctionne de la façon suivante :

1. Lorsqu'un utilisateur ou un administrateur crée ou modifie un mot de passe, le système crée un hachage de ce mot de passe et stocke le résultat. Le mot de passe en clair est supprimé du système.
2. Lorsque l'utilisateur se connecte au système, le mot de passe qu'il saisit est de nouveau haché.
3. Le système hôte supprime le mot de passe en clair qui a été entré.
4. Le nouveau mot de passe haché est comparé au hachage stocké.
5. Si les deux hachages concordent, le système autorise l'accès.

Le système hôte réalise toutes ces opérations sans même connaître le mot de passe original ; en réalité, sa valeur lui est totalement inutile. Un effet de bord de cette technique est qu'un pirate ayant réussi à pénétrer le système peut voler la base de données des mots de passe : il disposera alors d'une liste de hachages qu'il ne pourra pas utiliser pour retrouver les mots de passe originaux mais, avec suffisamment de temps, de puissance de calcul et si les mots de passe ont été mal choisis, il pourra utiliser une attaque par dictionnaire pour les découvrir. Il faut donc faire en sorte d'empêcher l'accès à la base des mots de passe et, si quelqu'un y parvient, tous les utilisateurs doivent immédiatement changer leurs mots de passe.

Chiffrement et hachage

D'un point de vue technique, ce traitement n'est pas un chiffrement mais un hachage, ce qui est différent pour deux raisons :

- Contrairement au chiffrement, les données ne peuvent pas être déchiffrées.
- Il est possible (mais très peu probable) que deux chaînes différentes produisent le même hachage. Il n'y a, en effet, aucune garantie qu'un hachage soit unique : vous ne devez donc pas utiliser un hachage pour vous en servir de clé dans une base de données.

```
function hash_ish($chaîne) {  
    return md5($chaîne);  
}
```

La fonction `md5()` renvoie une chaîne hexadécimale de 32 caractères formée à partir de l'algorithme Message-Digest de RSA Data Security Inc. (également connu sous le nom de MD5). Vous pouvez insérer cette chaîne de 32 caractères dans une base de données, la comparer à d'autres chaînes de 32 caractères ou simplement admirer sa perfection.

Amélioration du script

Il est virtuellement impossible de déchiffrer des données MD5. En tous cas, c'est très difficile. Cependant, vous devez quand même utiliser de bons mots de passe car il est quand même assez simple de créer une base de données de hachage pour tout le dictionnaire. Il existe d'ailleurs des dictionnaires MD5 en ligne dans lesquels vous pouvez entrer la chaîne **90a8db953336c8dabbcf48b1592a8c06** et obtenir "chien".

Par conséquent, bien que, *techniquement*, l'on ne puisse pas déchiffrer les chaînes MD5, elles sont quand même vulnérables – si quelqu'un réussit à récupérer votre base de données des mots de passe, vous pouvez être sûr qu'il utilisera un dictionnaire MD5. Lorsque vous créez des systèmes avec authentification par mot de passe, vous avez donc intérêt à utiliser des mots de passe suffisamment longs (au moins six caractères, mais huit est préférable) et contenant à la fois des lettres et des chiffres. Vérifiez également que ces mots de passe ne figurent pas dans un dictionnaire.

Recette 23 : Chiffrer les données avec Mcrypt

Les hachages MD5 sont parfaits si vous savez que vous n'aurez jamais besoin des données sous forme lisible. Malheureusement, ce n'est pas toujours le cas : si vous stockez des informations bancaires sous forme chiffrée, vous devrez les déchiffrer à un moment ou à un autre.

L'une des solutions les plus simples à ce problème consiste à utiliser le module Mcrypt, qui est une extension PHP permettant d'effectuer un chiffrement

sophistiqué de vos données. La bibliothèque Mcrypt offre plus de 30 chiffrements possibles et permet d'utiliser une phrase secrète garantissant que vous seul (ou, éventuellement, vos utilisateurs) pourrez déchiffrer les données. Pour utiliser ce module, vous devez recompiler PHP pour qu'il le prenne en compte, comme on l'a expliqué à la recette n°18, "Ajouter des extensions à PHP".

Le script suivant contient des fonctions qui se servent de Mcrypt pour chiffrer et déchiffrer les données :

```
<?php

$donnees = "Données à chiffrer";
$cle = "Phrase secrète utilisée par chiffrer pour chiffrer les données";
$code = "MCRYPT_SERPENT_256";
$mode = "MCRYPT_MODE_CBC";

function chiffrer($donnees, $cle, $code, $mode) {
// Chiffre les données

return (string)
    base64_encode
    (
        mcrypt_encrypt
        (
            $code,
            substr(md5($cle),0,mcrypt_get_key_size($code, $mode)),
            $donnees,
            $mode,
            substr(md5($cle),0,mcrypt_get_block_size($code, $mode))
        )
    );
}

function dechiffre($donnees, $cle, $code, $mode) {
// Déchiffre les données

return (string)
    mcrypt_decrypt
    (
        $code,
        substr(md5($cle),0,mcrypt_get_key_size($code, $mode)),
        base64_decode($donnees),
        $mode,
        substr(md5($cle),0,mcrypt_get_block_size($code, $mode))
    );
}

?>
```

La fonction `mcrypt()` a besoin de plusieurs informations :

- Les données à chiffrer.
- La phrase secrète, également appelée *clé*, pour chiffrer et déchiffrer les données.

- Le code utilisé pour chiffrer les données, qui indique l'algorithme de chiffrement. Ce script utilise `MCRYPT_SERPENT_256`, mais vous pouvez en choisir un autre, notamment `MCRYPT_TWOFISH192`, `MCRYPT_RC2`, ainsi que `MCRYPT_DES` ou `MCRYPT_LOKI97`.

NOTE *Pour connaître les chiffrements disponibles sur votre serveur, utilisez la recette n°8, "Afficher toutes les options de configuration de PHP" et recherchez la section "Mcrypt" sur la page produite par `phpinfo()`. Si Mcrypt est disponible, vous verrez deux sections : "Supported Cipher" et "Supported Modes". Vous pouvez chiffrer vos données en les utilisant exactement comme ils sont écrits.*

- Le mode utilisé pour chiffrer les données. Il existe plusieurs modes, dont *Electronic Codebook* et *Cipher Feedback*. Ce script utilise `MCRYPT_MODE_CBC` (*Cipher Block Chaining*).
- Un vecteur d'initialisation, également appelé IV (*Initialization Vector*) ou *graine*, qui est un ensemble de données binaires supplémentaires utilisées pour initialiser l'algorithme de chiffrement. C'est une mesure supplémentaire pour compliquer un peu plus la découverte de la valeur chiffrée.
- Les longueurs des chaînes de la clé et de IV qui dépendent, respectivement, du code et du bloc. Ces longueurs sont obtenues grâce aux fonctions `mcrypt_get_key_size()` et `mcrypt_get_block_size()`. Coupez ensuite la valeur de la clé à la bonne longueur avec un appel à `substr()` (si la clé est plus petite que la valeur indiquée, Mcrypt la complètera avec des zéros).

Si quelqu'un vole vos données et votre phrase secrète, il peut se contenter de tester tous les codes de chiffrement jusqu'à trouver celui qui fonctionne. C'est la raison pour laquelle nous chiffrons également la phrase avec la fonction `md5()` avant de l'utiliser : la possession des données et de la phrase ne permettra plus au pirate d'obtenir les informations qu'il souhaite puisqu'il devra à la fois connaître la fonction, les données et la phrase. Si c'est le cas, c'est qu'il a sûrement déjà un accès complet à votre serveur et que vous êtes de toutes façons mal parti.

Il faut également régler un petit problème lié au format de stockage des données. Mcrypt renvoie les données chiffrées sous un format binaire qui provoque de terribles erreurs lorsque l'on tente de les stocker dans certains champs MySQL. C'est la raison pour laquelle on utilise les fonctions `base64encode()` et `base64decode()` pour transformer les données sous un format compatible avec SQL.

Amélioration du script

En plus de tester les différentes méthodes de chiffrement, vous pouvez faciliter l'utilisation de ce script en plaçant, par exemple, la clé et le mode dans des constantes globales que vous stockerez dans un fichier inclus (voir la recette n°1, "Inclure un fichier extérieur dans un script") : cela vous évitera de devoir les répéter à chaque fois.

Recette 24 : Produire des mots de passe aléatoires

Les chaînes aléatoires (mais difficiles à deviner) jouent un rôle important dans la sécurité des utilisateurs. Si, par exemple, quelqu'un perd son mot de passe alors que l'on utilise des hachages MD5, vous ne pourrez pas le retrouver : en ce cas, vous devrez produire un mot de passe aléatoire suffisamment sûr et l'envoyer à cet utilisateur. Une autre application des chaînes aléatoires consiste à créer des liens d'activation pour l'accès aux services de votre site. Voici une fonction qui crée un mot de passe :

```
<?php
function cree_mdp($nb_cars) {
    if ((is_numeric($nb_cars) &&
        ($nb_cars > 0) &&
        (! is_null($nb_cars))) {

        $mdp = '';
        $cars_ok = 'abcdefghijklmnopqrstuvwxyz1234567890';

        // Initialise le générateur si nécessaire.
        srand(((int)((double)microtime()*1000003)) );

        for ($i = 0; $i <= $nb_cars; $i++) {
            $nb_aleatoire = rand(0, (strlen($cars_ok) -1));
            $mdp .= $cars_ok[$nb_aleatoire] ;
        }

        return $mdp;
    }
}
?>
```

Utilisation du script

La fonction `cree_mdp()` renvoie une chaîne ; tout ce qui vous reste à faire consiste à lui fournir la longueur de cette chaîne en paramètre :

```
<?php
$mdp_de_quinze_caracteres = cree_mdp(15);
?>
```

Elle fonctionne de la façon suivante :

1. Elle s'assure que `$nb_cars` est un nombre entier positif non nul.
2. Elle initialise la variable `$mdp` avec une chaîne vide.
3. Elle initialise la variable `$cars_ok` avec la liste des caractères admis dans le mot de passe. Ce script utilise toutes les minuscules non accentuées et les

chiffres de 0 à 9, mais vous pourriez choisir le jeu de caractères de votre choix.

4. Le générateur de nombres aléatoires ayant besoin d'une graine, on lui fournit un ensemble des valeurs pseudo-aléatoires (ce n'est pas vraiment nécessaire à partir de PHP 4.2).
5. La fonction effectue `$nb_cars` itérations, une par caractère du mot de passe.
6. Pour chaque nouveau caractère, le script utilise la longueur de `$cars_ok` pour produire un nombre aléatoire supérieur ou égal à 0 et strictement inférieur à cette longueur, puis ajoute à `$mdp` le caractère situé à cet indice dans `$cars_ok`.
7. À la fin de la boucle, la fonction renvoie `$mdp`.

4

TRAITEMENT DES FORMULAIRES



Les formulaires sont les moyens par lesquels les utilisateurs peuvent communiquer avec vos scripts ; pour tirer parti de la puissance de PHP, vous devez donc maîtriser leur fonctionnement. La première chose que vous devez comprendre est que, bien que PHP facilite l'accès aux données provenant des formulaires, il faut faire attention à la façon dont vous traitez ces données.

Mesures de sécurité : ne faites pas confiance aux formulaires

Les débutants commettent souvent l'erreur de faire confiance aux données provenant d'un formulaire HTML. Si un menu déroulant n'autorise l'utilisateur qu'à choisir une seule parmi trois valeurs, vous devez quand même la vérifier. Comme on l'a expliqué au Chapitre 3, vous ne pouvez pas non plus vous fier à JavaScript pour empêcher l'envoi de données quelconques à votre serveur.

Les visiteurs de votre site peuvent écrire leurs propres formulaires HTML et l'utiliser avec votre serveur ; ils peuvent également se passer totalement d'un navigateur et utiliser des outils automatisés pour interagir avec vos scripts. Lorsque vous mettez un script à disposition sur le Web, vous devez supposer que des personnes essaieront de jouer avec les paramètres pour tenter de découvrir un moyen plus simple d'utiliser votre site (bien qu'ils puissent également essayer quelque chose de bien moins innocent).

Pour garantir la sécurité de votre serveur, vous devez donc vérifier toutes les données reçues par vos scripts.

Stratégies de vérification

Il y a deux approches pour vérifier les données des formulaires : les listes noires et les listes blanches.

Les *listes noires* consistent à tenter de supprimer toutes les données incorrectes en supposant que les données provenant des formulaires sont correctes puis en éliminant explicitement les mauvaises. En général, cette technique n'est pas efficace. Supposons, par exemple, que vous vouliez éliminer tous les "mauvais" caractères d'une chaîne, comme les apostrophes. Vous pourriez rechercher et remplacer ces apostrophes, mais le problème est qu'il y aura *toujours* des mauvais caractères auxquels vous n'avez pas pensé. En général, les listes noires supposent que les données que vous recevez ne sont pas malicieuses.

En réalité, il est préférable de considérer systématiquement que ces données sont suspectes ; vous pourrez alors les filtrer pour n'accepter que celles qui sont correctes : c'est ce qu'on appelle les *listes blanches*. Si, par exemple, une chaîne ne doit contenir que des caractères alphanumériques, vous pouvez la comparer avec une expression régulière qui correspond à une chaîne formée uniquement des caractères A-Za-z0-9.

Le filtrage par liste blanche peut également forcer les données à appartenir un intervalle connu et modifier le type d'une valeur. Voici un résumé de quelques tactiques spécifiques :

- Si la valeur doit être numérique, utilisez la fonction `is_numeric()` pour le vérifier. Vous pouvez convertir une valeur en entier avec la fonction `intval()`.
- Si la valeur doit être un tableau, testez-la avec la fonction `is_array()`.
- Si la valeur doit être une chaîne, testez-la avec la fonction `is_string()`. Pour la convertir en chaîne, utilisez la fonction `strval()`.
- Si la valeur doit être null, testez-la avec `is_null()`.
- Si la valeur doit être définie, testez-la avec `isset()`.

Listes blanches et valeurs entières

Voici un exemple classique d'utilisation d'un filtrage par liste blanche pour une valeur numérique. Si la donnée n'est pas numérique, on utilise une valeur par défaut de zéro (cela suppose évidemment que zéro soit une valeur acceptable) :

```
if (! is_numeric($donnee)) {  
    // Utilise une valeur par défaut de 0  
    $donnee = 0;  
}
```

Pour les entiers, si vous savez que toutes les valeurs entières sont admises, vous pouvez utiliser l'instruction `$donnee = intval($donnee)`; pour transtyper `$donnee` en valeur entière.

Utiliser `$_POST`, `$_GET`, `$_REQUEST` et `$_FILES` pour accéder aux données des formulaires

La recette n°14 du Chapitre 2, "Désactiver les variables globales automatiques", a montré comment désactiver l'option `register_globals`, qui crée automatiquement des variables globales à partir des données de formulaires.

Nous allons maintenant expliquer comment utiliser les variables `$_POST`, `$_FILES` et `$_GET` pour récupérer les données de formulaires.

Recette 25 : Récupérer les données des formulaires en toute sécurité

Vous devriez toujours extraire les données des formulaires à partir des variables prédéfinies du serveur. Toutes les données passées à votre page par un formulaire utilisant la méthode POST sont automatiquement stockées dans un tableau nommé `$_POST`; de même les données des formulaires qui utilisent la méthode GET sont stockées dans un tableau nommé `$_GET`. Les informations sur les dépôts de fichiers sont stockées dans un tableau spécial `$_FILES` (voir la recette n°54, "Déposer des images dans un répertoire"). En outre, il existe également une variable combinée, appelée `$_REQUEST`.

Pour accéder à la valeur du champ `nom_utilisateur` d'un formulaire utilisant la méthode POST, il suffit d'écrire `$_POST['nom_utilisateur']`. Pour faire de même avec un formulaire utilisant la méthode GET, on utilisera `$_GET['nom_utilisateur']`. Si la méthode utilisée par le formulaire ne vous intéresse pas, vous pouvez accéder à la valeur de ce champ avec `$_REQUEST['nom_utilisateur']`.

```
<?php

$valeur_post = $_POST['champ_post'];
$valeur_get = $_GET['champ_get'];
$sune_valeur = $_REQUEST['un_champ'];

?>
```

`$_REQUEST` est l'union des tableaux `$_GET`, `$_POST` et `$_COOKIE`. Si vous avez plusieurs paramètres de même nom il faut savoir que, par défaut, PHP renvoie d'abord le cookie, puis le paramètre POST et enfin le paramètre GET.

La sécurité de `$_REQUEST` a donné lieu à des débats qui n'ont pas lieu d'être : toutes ses sources venant du monde extérieur (le navigateur de l'utilisateur), vous devez de toutes façons vérifier toutes les données que vous comptez utiliser dans ce tableau, exactement comme vous le feriez avec les autres tableaux prédéfinis. Les seuls problèmes que vous pourriez rencontrer sont des bogues ennuyeux susceptibles d'apparaître à cause des cookies qu'il contient.

Recette 26 : Supprimer les espaces inutiles

Les espaces inutiles sont toujours un problème lorsque l'on manipule les données des formulaires. Généralement, la fonction `trim()` est le premier outil vers lequel se tourne le programmeur car elle permet de supprimer les espaces inutiles au début ou à la fin d'une chaîne. " Le langage PHP " devient ainsi "Le langage PHP". En fait, cette fonction est si pratique que vous l'utiliserez sur quasiment toutes les données saisies par l'utilisateur, sauf les tableaux :

```
$saisie = trim($saisie);
```

Parfois, les espaces inutiles peuvent se trouver *au milieu* de la chaîne – l'utilisateur a pu copier et coller un contenu de courrier électronique, par exemple. En ce cas, vous pouvez remplacer les suites d'espaces et les autres caractères d'espacement par une espace unique à l'aide de la fonction `preg_replace()`. Le terme *reg* indique que cette fonction utilise les *expressions régulières*, une forme très puissante de recherche par motif que vous rencontrerez plusieurs fois dans ce chapitre.

```
<?php

function suppr_espaces($chaîne) {
    $chaîne = preg_replace('/\s+/', ' ', $chaîne);
    $chaîne = trim($chaîne);
    return $chaîne;
}

?>
```

Ce script vous servira en de maintes occasions, même en dehors de la vérification des formulaires, car il permet de nettoyer les données provenant d'autres sources externes.

Recette 27 : Importer des données de formulaire dans un tableau

L'une des astuces les plus pratiques que vous pouvez utiliser en PHP n'est en réalité pas une astuce PHP mais une astuce HTML. Lorsqu'un utilisateur remplit un formulaire, on vérifie souvent les valeurs de plusieurs cases à cocher. Supposons, par exemple, que vous fassiez une enquête sur les types de films que regardent les visiteurs de votre site et que vous voulez insérer automatiquement ces valeurs dans une base de données appelée *preferences_clients*. La méthode brutale consiste à donner un nom distinct à chaque case du formulaire, comme ici :

```
<p>Quel genre de film regardez-vous ?</p>
<input type="checkbox" name="action" value="oui"> Action
<input type="checkbox" name="drame" value="oui"> Drame
<input type="checkbox" name="comedie" value="oui"> Comédie
<input type="checkbox" name="sfiction" value="oui"> Science-fiction
```

Malheureusement, lorsque vous traiterez ce formulaire sur la page suivante, vous devrez utiliser une suite de tests `if/then` pour vérifier les données – un test pour la valeur de `$action`, un autre pour la valeur de `$drame`, etc. L'ajout d'une nouvelle case dans le formulaire implique d'ajouter un nouveau test `if/then` à la page qui le traite.

Le meilleur moyen de simplifier ce traitement consiste à stocker toutes les valeurs des cases dans un même tableau en ajoutant `[]` après le nom, comme ici :

```
<form action="traitement.php" method="post">
<p>Comment vous appelez-vous ?</p>
<p><input type="text" name="nom_client"></p>

<p>Quel genre de film regardez-vous ?</p>
<p><input type="checkbox" name="genre_film[]" value="action"> Action
<input type="checkbox" name="genre_film[]" value="drame"> Drame
<input type="checkbox" name="genre_film[]" value="comedie"> Comédie
<input type="checkbox" name="genre_film[]" value="sfiction"> Science-fiction</p>
<input type="submit">
</form>
```

Lorsque PHP récupère les données d'un formulaire comme celui-ci, il stocke les valeurs des cases à cocher dans un unique tableau que vous pouvez ensuite parcourir de la façon suivante :

```

<?php
$genre_film = $_POST["genre_film"];
$nom_client = strval($_POST["nom_client"]);

if (is_array($genre_film)) {
    foreach ($genre_film as $genre) {
        print "$nom_client regarde des films du genre $genre.<br>";
    }
}
}

?>

```

Cette technique ne fonctionne pas qu'avec les cases à cocher ; elle est extrêmement pratique à chaque fois qu'il faut traiter un nombre quelconque de lignes. Supposons, par exemple, un menu où nous voulons montrer tous les articles d'une catégorie donnée. Bien que l'on ne sache pas le nombre d'articles d'une catégorie, le client devrait pouvoir entrer une quantité dans un champ de saisie pour chaque article qu'il veut acheter et ajouter tous les articles d'un simple clic. Ce menu ressemblerait à celui de la Figure 4.1.

Comment vous appelez-vous ?

Quel genre de film regardez-vous ?

Action Drame Comédie Science-fiction

Figure 4.1 : Un formulaire avec un tableau de cases à cocher

Pour construire ce formulaire, nous avons besoin d'accéder aux noms et aux numéros de produit dans la table MySQL `infos produits` décrite en annexe :

```

<?php
/* Insérer ici le code pour se connecter à $db. */

$categorie = "chaussures";
/* Obtention des produits à partir de la base. */
$sql = "SELECT nom_produit, num_produit
        FROM infos_produits
        WHERE categorie = '$categorie'";

$resultat = @mysql_query($sql, $db) or die;

```

```

/* Initialise les variables. */
$commande = ""; /* Contendra les données du formulaire */
$i = 1;

print '<form action="ajoutpanier.php" method="post">';

while ($ligne = mysql_fetch_array($resultat)) {
    // Parcourt le résultat de la requête SQL.
    $nom_produit = stripslashes($row['nom_produit']);
    $num_produit = $row['num_produit'];

    // Ajoute la ligne au formulaire.
    print "<input type=\"hidden\" name=\"num_produit[$i]\"
        value=\"$num_produit\">";
    print "<input type=\"text\" name=\"quantite[$i]\"
        size=\"2\" value=\"0\"> $nom_produit<br />";

    $i++;
}

print '<input type="submit" value="Ajouter au panier"></form>';

?>

```

Pour traiter ce formulaire, vous devez examiner les deux tableaux passés au script de traitement – l'un contient tous les numéros de produit (`$num_produit`) et l'autre les valeurs des champs de saisie pour les quantités respectives (`$quantite`). Peu importe le nombre d'articles qui sont affichés sur la page : `$num_produit[123]` contient le numéro de produit du 123^e article affiché et `$quantite[123]` le nombre que le client a entré dans le champ de saisie correspondant.

Le script de traitement *ajoutpanier.php* est le suivant :

```

<?php

$num_produit = $_POST["num_produit"];
$quantite = $_POST["quantite"];

if (is_array($quantite)) {
    foreach ($quantite as $cle => $qte_article) {
        $qte_article = intval($qte_article);
        if ($qte_article > 0) {
            $num = $num_produit[$cle];
            print "Ajout de $qte_article articles du produit $num.<br>";
        }
    }
}

?>

```

Comme vous pouvez le constater, ce script dépend entièrement de l'utilisation de l'indice du tableau \$quantite (la variable \$cle) pour le tableau \$num_produit.

Recette 28 : S'assurer qu'une réponse fait partie d'un ensemble de valeurs

Comme on l'a déjà précisé, il ne faut *jamais* supposer que les données transmises par un formulaire sont sûres. Étudiez par exemple ce fragment de formulaire :

```
<SELECT NAME="type_carte">
  <OPTION value="visa">Visa</OPTION>
  <OPTION value="amex">American Express</OPTION>
  <OPTION value="mastercard">MasterCard</OPTION>
</SELECT>
```

Comment être sûr que la donnée que vous recevrez sera vraiment *visa*, *amex* ou *mastercard* ? C'est assez simple : il suffit d'utiliser les valeurs possibles comme clés d'un tableau et vérifier que la valeur reçue est bien une clé de ce tableau. Voici un exemple :

```
<?php

$cartes_credit = array(
    "amex" => true,
    "visa" => true,
    "mastercard" => true,
);
$type_carte = $_POST["type_carte"];
if ($cartes_credit[$type_carte]) {
    print "$type_carte est une carte de crédit autorisée.";
} else {
    print "$type_carte n'est pas une carte de crédit autorisée.";
}
?>
```

Amélioration du script

L'avantage de cette méthode de stockage est que vous pouvez temporairement désactiver un choix en mettant à *false* la valeur qui correspond à sa clé. Vous pouvez aussi légèrement modifier le script pour qu'il fournisse à la fois les valeurs et les textes qui leur sont associés : il suffit de placer ces textes dans les valeurs du tableau afin, par exemple, d'afficher *American Express* quand l'utilisateur a choisi *amex*.

Voici un exemple qui utilise cette technique :

```
<?php

$cartes_credit = array(
    "amex" => "American Express",
    "visa" => "Visa",
    "mastercard" => "MasterCard",
);
$type_carte = $_POST["type_carte"];
if (count($cartes_credit[$type_carte]) > 0) {
    print "Type de paiement choisi : $cartes_credit[$type_carte].";
} else {
    print "Type de carte non autorisé.";
}
?>
```

NOTE *L'exemple précédent est une information extrêmement utile qui mérite d'être stockée dans un fichier de configuration central.*

Recette 29 : Utiliser plusieurs boutons de validation

Vous pouvez parfois avoir besoin d'un formulaire qui effectue deux traitements distincts en fonction du bouton sur lequel l'utilisateur a cliqué – un bouton peut permettre de modifier un article posté dans un forum alors qu'un autre permettra de le supprimer, par exemple. Vous pourriez placer deux formulaires dans la même page afin de renvoyer l'utilisateur vers deux pages distinctes en fonction de son choix, mais vous devrez alors mettre des informations redondantes dans ces deux formulaires et ce serait insupportable pour l'utilisateur.

En HTML, les boutons aussi ont des valeurs que vous pouvez consulter. Il faut donc construire le formulaire de cette façon :

```
<form action="traitement.php" method="post">
  <input name="num_article" type="hidden" value="1234">
  <input name="action" type="submit" value="Modifier">
  <input name="action" type="submit" value="Supprimer">
</form>
```

Dans *traitement.php*, il suffit ensuite de lire `$_POST['action']` pour savoir quel est le bouton qui a été cliqué par l'utilisateur et agir en conséquence.

Recette 30 : Vérifier la validité d'une carte de crédit

Voici un bref résumé du fonctionnement du paiement en ligne par carte de crédit. Vous devez d'abord trouver un fournisseur de services en ligne (*Authorize.net* ou *Secpay.com*, par exemple) pour vous créer un compte marchand.

Ce compte ressemble à un compte bancaire, sauf qu'il vous permet d'effectuer des prélèvements sur les cartes de crédit. Généralement, le fournisseur prend un pourcentage sur chaque transaction effectuée.

Si vous avez une boutique réelle qui accepte le paiement par carte de crédit, vous utilisez sûrement déjà une solution marchande, mais toutes ne proposent pas les transactions en ligne. Celles qui le font vous donnent accès à une passerelle de paiement, c'est-à-dire un serveur sécurisé prenant en charge le traitement des paiements par carte de crédit. Généralement, ces transactions ont lieu *via* un flux XML : vous pouvez donc utiliser cURL pour échanger ces données XML avec la passerelle de paiement (voir le Chapitre 11 pour plus de détails).

Cependant, vous pouvez effectuer quelques étapes de vérification de formulaire avant de vous connecter à la passerelle de paiement : si l'utilisateur s'est trompé dans son numéro de carte, cela permettra d'économiser une transaction, donc des frais et cela accélèrera également le traitement. En fait, vous pouvez éliminer les mauvais numéros de carte à l'aide d'un algorithme assez simple ; il est même possible de trouver le type d'une carte de crédit à partir de son numéro. N'oubliez pas, cependant, que la réussite de ces tests ne garantit pas que la carte n'a pas été volée, annulée ou qu'elle n'appartient pas à une autre personne.

```
<?php

function verifie_num_cc($num_cc) {
/* Renvoie le type de la carte si son numéro est correct */
    $faux = false;
    $type_carte = "";
    $regex_cartes = array(
        "/^4\d{12}(\d\d){0,1}$/" => "visa",
        "/^5[12345]\d{14}$/" => "mastercard",
        "/^3[47]\d{13}$/" => "amex",
        "/^6011\d{12}$/" => "discover",
        "/^30[012345]\d{11}$/" => "diners",
        "/^3[68]\d{12}$/" => "diners",
    );

    foreach ($regex_cartes as $regex => $type) {
        if (preg_match($regex, $num_cc)) {
            $type_carte = $type;
            break;
        }
    }

    if (!$type_carte) {
        return $faux;
    }

    /* Algorithme de somme de contrôle modulo 10 */
    $code_inverse = strrev($num_cc);
```

```

$checksum = 0;

for ($i = 0; $i < strlen($code_inverse); $i++) {
    $num_courant = intval($revcode[$i]);
    if ($i & 1) { /* Position impaire */
        $num_courant *= 2;
    }
    /* Sépare les chiffres et les additionne. */
    $checksum += $num_courant % 10;
    if ($num_courant > 9) {
        $checksum += 1;
    }
}

if ($checksum % 10 == 0) {
    return $type_carte;
} else {
    return $faux;
}
}
?>

```

Cette fonction se décompose en deux parties. La première trouve le type de la carte et la seconde détermine si sa somme de contrôle est correcte. Si la carte réussit ces deux tests, la fonction renvoie son numéro sous forme de chaîne. Dans le cas contraire, elle renvoie `false` (vous pouvez changer cette valeur en modifiant le contenu de la variable `$faux`).

Dans la première partie, on utilise une astuce permettant de déterminer le type de la carte et de confirmer son préfixe en une seule étape. En effet, les numéros des cartes bancaires respectent un certain format : tous les numéros de cartes Visa, par exemple, commencent par le chiffre 4 et sont formés de 13 ou de 16 chiffres ; les numéros MasterCard commencent par les nombres 51 à 55 et ont 16 chiffres et les numéros American Express commencent par 34 ou 37 et ont 15 chiffres. Ces règles s'expriment aisément par quelques expressions régulières ; ces règles étant disjointes, nous pouvons faire correspondre ces expressions à leurs types de cartes respectifs dans le tableau `$regex_cartes`. Pour vérifier un format de numéro, il suffit de parcourir les expressions régulières jusqu'à en trouver une qui correspond. En ce cas, on initialise `$type_carte` et on passe à l'étape suivante. Si aucune expression ne convient, on sort de la fonction en renvoyant une valeur indiquant cet échec.

Le test de la somme de contrôle d'un numéro de carte utilise un algorithme modulo 10, qui est relativement simple à écrire et qui effectue le traitement suivant :

- Il commence avec une somme de contrôle égale à 0.
- Il parcourt de droite à gauche tous les chiffres du numéro de carte.

- Si le chiffre courant est à un indice impair (l'indice du chiffre le plus à droite est 0), ce chiffre est multiplié par 2. Si le résultat est supérieur à 9, on additionne les deux chiffres qui le composent et on ajoute cette somme à la somme de contrôle (si un 8 devient 16, par exemple, on additionne 1 et 6 et l'on obtient 7). Sinon, le chiffre courant (doublé s'il est à un indice impair) est simplement ajouté à la somme de contrôle.
- Après le parcours de tous les chiffres, la somme de contrôle finale doit être divisible par 10 ; sinon, le numéro était incorrect.

Cet algorithme peut être codé de plusieurs façons ; nous avons privilégié ici la compacité et la lisibilité.

Utilisation du script

Il suffit de fournir à la fonction `verifie_num_cc()` une chaîne contenant un numéro de code et de tester sa valeur de retour. La seule précaution à prendre consiste à vérifier que cette chaîne ne contient que des chiffres ; pour cela, vous pouvez utiliser `preg_replace()` avant d'appeler la fonction. Voici un fragment de code qui appelle la fonction pour tester plusieurs numéros de carte :

```
$nums = array(
    "3721 0000 0000 000",
    "3400000000000009",
    "5500 0000 0000 0004",
    "4111 1111 1111 1111",
    "4222 2222 2222",
    "4007000000027",
    "30000000000004",
    "601100000000004",
);
foreach ($nums as $num) {
    /* Supprime tous les caractères non numériques du numéro de carte */
    $num = preg_replace('/[^0-9]/', "", $num);

    $t = verifie_num_cc($num);
    if ($t) {
        print "$num est correct (type : $t).\n";
    } else {
        print "$num est incorrect.\n";
    }
}
}
```

Amélioration du script

Si vous connaissez le format de leurs numéros, vous pouvez ajouter les autres cartes de crédits connues. La page <http://www.sitepoint.com/print/card-validation-class-php> contient une foule d'informations utiles sur les cartes de crédit, de même que le site <http://www.phpsources.org/scripts407-PHP.htm> qui propose un script permettant de vérifier les numéros SIRET.

Recette 31: Vérifier la date d'expiration d'une carte de crédit

Lorsque vous acceptez une carte de crédit, vous devez savoir si elle a expiré. Pour cela, le plus simple consiste à ajouter à votre code HTML un menu déroulant permettant aux utilisateurs de choisir la date d'expiration, afin d'éviter les ambiguïtés dans les formats des dates :

```
<select name="mois_cc">
  <option value="01" >01 : Janvier</option>
  <option value="02" >02 : Février</option>
  <option value="03" >03 : Mars</option>
  <option value="04" >04 : Avril</option>
  <option value="05" >05 : Mai</option>
  <option value="06" >06 : Juin</option>
  <option value="07" >07 : Juillet</option>
  <option value="08" >08 : Août</option>
  <option value="09" >09 : Septembre</option>
  <option value="10" >10 : Octobre</option>
  <option value="11" >11 : Novembre</option>
  <option value="12" >12 : Décembre</option>
</select>

<select name="annee_cc">
<?php
/* Crée les options pour les dix années à partir de l'année en cours */
$a = intval(date("Y"));
for ($i = $a; $i <= $a + 10; $i++) {
  print "<option value=\"$i\">$i</option>\n";
}
?>

</select>
```

Vous disposez maintenant d'un formulaire permettant d'indiquer la date d'expiration ; il vous reste à vérifier les données qu'il envoie :

```
<?php

function verif_date_exp($mois, $annee) {
  /* Valeur de minuit pour le jour suivant le mois d'expiration */
  $expiration = mktime(0, 0, 0, $mois + 1, 1, $annee);

  $maintenant = time();
  /* On ne tient pas compte des dates dans plus de 10 ans. */
  $max = $maintenant + (10 * 365 * 24 * 60 * 60);

  if ($expiration > $maintenant && $expiration < $max) {
```

```
    return true;
} else {
    return false;
}
?>
```

Pour vérifier la date d'expiration d'une carte de crédit, il suffit de s'assurer que la date est située entre la date courante et une certaine date future (cette fonction utilise 10 ans). Les meilleurs outils pour ce traitement sont décrits au Chapitre 6 ; nous ne ferons donc que passer rapidement ce traitement en revue.

La seule astuce à connaître est que la carte cesse d'être utilisable après le dernier jour du mois d'expiration : si une carte expire en 06/2009, elle cessera en fait de fonctionner le 1^{er} juillet 2009. Nous devons donc ajouter un mois à la date indiquée. Cette opération peut être délicate car elle peut également faire passer à l'année suivante ; comme vous le verrez au Chapitre 6, la fonction `mktime()` que nous utilisons ici pour calculer automatiquement la date d'expiration sait gérer les numéros de mois qui sont supérieurs à 12. Après avoir calculé cette date, vous avez simplement besoin de la date courante et de la date limite. Ensuite, la vérification de la date d'expiration se ramène à deux comparaisons simples :

Utilisation du script

```
if (verif_date_exp($mois_cc, $annee_cc)) {
    // Accepte la carte.

} else {
    // La carte n'est plus valable.
}
}
```

Recette 32 : Vérifier la validité des adresses de courrier électronique

Les clients entrent toutes sortes d'adresses électroniques fantaisistes dans les formulaires. Le script de cette section vérifie qu'une adresse e-mail respecte le plus possible les règles énoncées dans la RFC 2822. Il n'empêchera personne d'entrer une adresse fausse (mais conforme) comme *bidon@inexistant.com*, mais il pourra au moins détecter quelques erreurs de saisie.

NOTE *Si le fait d'avoir une adresse e-mail valide est essentiel, vous devez faire en sorte que les comptes utilisateurs ne soient activés que par des liens envoyés par courrier électronique, comme on l'explique dans la recette n°65, "Vérifier les comptes utilisateurs avec le courrier électronique". C'est une mesure plutôt extrême ; si vous souhaitez que plus de personnes partagent leurs adresses avec vous, indiquez-leur simplement que vous ne les spammez pas (et respectez cette promesse).*

```

<?php

function adresse_ok($mel) {
// Vérifie le format de l'adresse mel

    if (! preg_match( '/^[A-Za-z0-9!#$%&\'*+,-/=^_`{|}~]+@[A-Za-z0-9-]+(\.[A-Za-z0-9-]+)+[A-Za-z]$/', $mel)) {
        return false;
    } else {
        return true;
    }
}
?>

```

Ce script utilise une expression régulière pour tester si l'adresse e-mail indiquée utilise des caractères autorisés (lettres, points, tirets, barres de fraction, etc.) avec un symbole @ au milieu et au moins un point avant la fin. La recette n°39, "Expressions régulières" vous en apprendra plus sur le sujet.

Recette 33 : Tester la validité des numéros de téléphone

Comme pour les adresses e-mail, il n'existe aucun moyen de s'assurer qu'un numéro de téléphone est correct, à moins d'appeler ce numéro. Vous pouvez néanmoins tester le nombre de chiffres et les mettre dans un format standard. La fonction suivante renvoie un numéro de téléphone de 10 chiffres si la chaîne qui lui est passée contient des caractères numériques et commence par '0' ou si elle commence par '+33' suivi de 9 caractères numériques. Si le nombre fourni ne correspond pas, la fonction renvoie false.

```

<?php
function tel_ok($num_tel) {
    $inter = ($num_tel[0] == '+');
    $num_tel = preg_replace('/[^\d]+/', '', $num_tel);
    $nb_chiffres = strlen($num_tel);
    if ($inter && $nb_chiffres == 11 && substr($num_tel, 0, 2) == "33") {
        return "0" . substr($num_tel, 2);
    } else if ($nb_chiffres == 10 && $num_tel[0] == "0") {
        return $num_tel;
    } else {
        return false;
    }
}
?>

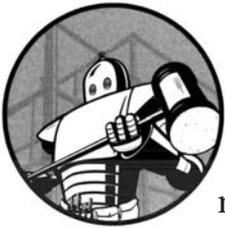
```

Ce script montre la puissance des expressions régulières combinées aux fonctions standard sur les chaînes. La clé consiste à supprimer tous les caractères qui

ne sont pas des chiffres – une opération faite pour `preg_replace()`. Lorsque vous êtes sûr qu’il ne reste plus que des chiffres dans la chaîne, il suffit simplement d’examiner sa longueur pour connaître le nombre de chiffres et le reste vient quasiment tout seul.

5

TRAITEMENT DU TEXTE ET DE HTML



Savoir comment retrouver, transformer et supprimer des mots est essentiel pour tout webmestre. Certaines fonctions de ce chapitre sont assez simples, mais nous les mentionnerons malgré tout pour mémoire. Les fonctions plus complexes utilisent des expressions régulières, une partie puissante de PHP que chaque webmestre doit savoir manipuler. Commençons par quelques opérations élémentaires sur les chaînes.

Recette 34 : Extraire une partie d'une chaîne

Imaginez que vous soyez à la tête d'une boutique en ligne vendant des cartes de collection. Lorsqu'ils achètent un lot de cartes, vos clients sont susceptibles d'effectuer des recherches au pluriel — *Châteaux* au lieu de *Château*, par exemple. Cela posait des problèmes à notre système de recherche car une requête utilisant *Châteaux* ne renvoyait aucun résultat. Pour résoudre ce problème, j'ai utilisé quelques fonctions de manipulation des chaînes pour analyser la fin de la requête de l'utilisateur et supprimer les occurrences de la lettre *x*. Passons ces fonctions en revue.

La fonction `substr()` permet d'extraire la partie de la chaîne que vous utiliserez lors des comparaisons et des autres opérations. Si, par exemple, la dernière lettre de la chaîne est le caractère *x*, vous pouvez le supprimer et réessayer si la requête initiale n'a renvoyé aucun résultat

L'appel de `substr()` est de la forme :

```
substr(chaine, debut, nbre)
```

chaine est la chaîne initiale, *debut* est l'indice de début de l'extraction et *nbre* est le nombre de caractères à extraire. Le premier indice d'une chaîne vaut 0. Le code suivant, par exemple, affiche *cde*, les trois caractères à partir de l'indice 2 :

```
echo substr('abcdef', 2, 3);
```

NOTE Pour calculer un indice pour `substr()`, vous aurez peut-être besoin de connaître la longueur de la chaîne. Pour cela, utilisez la fonction `strlen(chaine)`.

Si vous omettez le dernier paramètre de `substr()`, l'appel renverra tous les caractères de la chaîne à partir de l'indice de départ (*debut*) jusqu'à la fin de la chaîne. L'exemple suivant affiche tous les caractères d'une chaîne à partir de l'indice 2 :

```
echo substr('abcdef', 2); // cdef
```

En outre, si *debut* est négatif, `substr()` (et de nombreuses autres fonctions sur les chaînes) commencera à compter à partir de la fin de la chaîne, comme dans cet exemple qui affiche les deux avant-derniers caractères d'une chaîne :

```
echo substr('abcdef', -3, 2); // de
```

Lorsque vous connaissez la sous-chaîne qui vous intéresse, vous pouvez manipuler la chaîne de différentes façons :

- **Réaffecter une partie donnée de la chaîne** en utilisant `substr()` pour supprimer les caractères inutiles. `$chaine = substr($chaine, 0, 10)`; initialise, par exemple, `$chaine` avec les 10 premiers caractères de sa valeur initiale.
- **Supprimer les N derniers caractères** en recourant conjointement aux fonctions `substr()` et `strlen()`. `$chaine = substr($chaine, 0, strlen($chaine) - 3)`; initialise, par exemple, `$chaine` avec sa valeur initiale, sauf les trois derniers.
- **Remplacer les caractères** avec la fonction `substr_replace()`, qui permet de remplacer une sous-chaîne par une autre de votre choix. `substr_replace('abcdef', 'bbb', 1, 2)`, par exemple, renvoie *abbbdef*.

Pour comprendre comment utiliser des sous-chaînes dans votre travail quotidien, revenons à l'exemple des châteaux. Rappelez-vous que les utilisateurs

peuvent rechercher *Châteaux*, mais qu'un article peut être décrit par *Château*. Ce fragment de code montre un moyen de gérer cette situation :

```
$sql = "SELECT * FROM infos_produits WHERE nom_produit = '$saisie'"
$resultat = @mysql_query($sql, $connexion);

if (mysql_num_rows($resultat) == 0) {
    // Aucune ligne dans l'ensemble résultat (pas de produit trouvé).
    if (substr($saisie, -1) == 'x') {
        // Le dernier caractère de la chaîne est un "x".
        // Supprime le dernier caractère de la chaîne (le x).
        $saisie = substr_replace($saisie, '', -1);
        // Crée une autre requête SQL avec la nouvelle chaîne.
        $sql = "SELECT * FROM infos_produits WHERE nom_produit = '$saisie'";
        $resultat = @mysql_query($sql, $connexion);
    }
}
```

Cet algorithme recherche d'abord la chaîne `$saisie` dans l'inventaire des articles. Si aucun article n'est trouvé *et* que le dernier caractère de `$saisie` est un `x`, il réessaie la requête sans le `x`. Quel que soit le dernier caractère, vous pouvez examiner le résultat de la requête dans `$resultat` après l'exécution de l'algorithme.

Amélioration du script

Ce fragment de code s'avère problématique. Il peut y avoir un article *Châteaux* et un article *Château*. Tel qu'il est écrit, le script ne renvoie un résultat que pour l'un de ces deux cas et il est incohérent car il dépend de la présence de la forme plurielle. Si, par exemple, vous avez un article *Châteaux* et deux articles *Château*, vous n'obtiendrez que le premier article mais, si l'article au pluriel n'existe pas, vous obtiendrez les deux autres.

Vous pouvez ajouter cette fonctionnalité en utilisant le test `substr()` pour qu'il corrige votre requête SQL au lieu de la remplacer entièrement :

```
$sql = "SELECT * FROM infos_produits WHERE nom_produit = '$saisie'";
if (substr($saisie, -1) == 'x') {
    // Le dernier caractère de la chaîne est un "x".
    // On ajoute une autre possibilité à la clause WHERE.
    $sql .= " OR nom_produit = '" . substr_replace($saisie, '', -1) . "'";
}
$resultat = @mysql_query($sql, $connexion);
```

Cette approche est très différente car elle n'utilise qu'une seule requête SQL pour tout faire. Au lieu de tenter une seconde requête lorsque la première

échoue, le principe consiste à ajouter une partie `OR` à la clause `WHERE` si le nom de produit saisi se termine par un `x`.

Recette 35 : Mettre une chaîne en majuscules, en minuscules ou en capitales

Un problème qui se pose parfois avec PHP est que MySQL sait gérer les champs textuels sans tenir compte de la casse alors que les chaînes de PHP sont sensibles à la casse. Dans une requête, MySQL ne fait pas de différence entre *Ferrett*, *FERRETT* et *FerReTt*. Pourtant, en tant que chaînes PHP, elles n'ont rien en commun. Par conséquent, vous devez modifier la casse des caractères d'une chaîne PHP avant de les comparer ou de les afficher.

PHP dispose de trois fonctions essentielles pour modifier la casse des chaînes `strtolower()`, `strtoupper()` et `ucwords()`. Voici un exemple de leur utilisation :

```
<?
$chaine = "salUt, cOmMenT vAs-tU ?";
echo strtoupper($chaine);
// Affiche "SALUT, COMMENT VAS-TU ?"

echo strtolower($chaine);
// Affiche "salut, comment vas-tu ?"

echo ucwords(strtolower($chaine));
// Affiche "Salut, Comment Vas-tu ?"
?>
```

Ces appels fonctionnent de la façon suivante :

- `strtoupper()` met tous les caractères d'une chaîne en majuscules.
- `strtolower()` passe tous les caractères d'une chaîne en minuscules.
- `ucwords()` met la première lettre de chaque mot de la chaîne en majuscule.

NOTE Vous remarquerez que ce script recourt à une petite astuce : nous avons utilisé `strtolower()` avant `ucwords()`. Sinon, l'affichage serait *SalUt, cOmMenT vAs-tU ?*

Problèmes éventuels

Il y a quelques petits problèmes avec `ucwords()`. Le premier est qu'elle ne capitalise pas les lettres situées après un caractère non alphabétique : une chaîne comme *m. dupont-durand* deviendrait donc *M. Dupont-durand* et *ac/dc* donnerait *Ac/dc*. Si cela vous pose problème, vous pouvez créer une fonction utilisant les expressions régulières pour décomposer la chaîne en un tableau de mots, puis appeler `ucwords()` pour capitaliser chaque mot et, enfin, rejoindre les mots de ce tableau en une seule chaîne.

Un second problème est que `ucwords()` capitalise certains mots qui ne devraient pas l'être, comme *et*, *ou* et *non*. Si vous voulez utiliser un style typographique correct, vous pouvez écrire une simple fonction qui utilise `str_replace()` pour prendre ces mots en charge :

```
<?
function capitalise_mieux($chaine) {
    // Met les mots en majuscules, sauf certains.
    $mots_majuscules = array("De ", "Des", "Le ", "La ", "Les ", "Et ",
                            "Un ", "Une ", "Ou ", "Non ");
    $mots_minuscules = array("de ", "des ", "le ", "la", "les ", "et ",
                             "un ", "une ", "ou ", "non ");

    $chaine = ucwords(strtolower($chaine));
    $chaine = str_replace($mots_majuscules, $mots_minuscules, $chaine);
    // Met la première lettre en majuscule.
    return ucfirst($chaine);
}
?>
```

Enfin, si vous manipulez des noms, `ucwords(strtolower())` supprime les capitales existantes : un nom comme *McMurdo* deviendra donc *McMurdo*. Si ces capitales sont importantes et que vous devez les préserver, mais que vous devez comparer ce genre de chaînes en PHP, utilisez `strcasecmp(ch1, ch2)`, qui ne tient pas compte de la casse lors de la comparaison de *ch1* et *ch2*.

Recette 36 : Rechercher des sous-chaînes

PHP dispose de plusieurs fonctions permettant de rechercher une sous-chaîne dans une chaîne et votre choix dépendra de ce que vous comptez faire du résultat. Voici les trois fonctions de base :

- `strpos()` trouve la position de la première occurrence de la sous-chaîne.
- `strrpos()` trouve la position de la dernière occurrence de la sous-chaîne. Utilisez cette fonction avec `substr()` pour extraire tout ce qui se trouve après dans la chaîne.
- `strstr()` renvoie tout ce qui se trouve après la première occurrence de la sous-chaîne.

Ces trois fonctions renvoient `False` si la sous-chaîne n'existe pas. Une position et une chaîne pouvant être évaluées à `False` dans une conditionnelle, il est très important de vérifier le type de la valeur de retour, comme dans le script suivant :

```
<?php
$chaine = "J'approuve ce qu'a fait le sénateur Foghorn dans la guerre sur
Buttermilk. Je m'appelle Ferrett Steinmetz, et j'approuve ce message.";
```

```

$terme = "approuve";
// Apparaît-il dans la chaîne ?
$pos = strpos($chaîne, $terme);
if ($pos === False) {
    print "$terme n'est pas dans la chaîne\n";
} else {
    print "position : $pos\n";
    print "dernière position : " . strval(strrpos($chaîne, $terme)) . "\n";
    print strstr($chaîne, $terme) . "\n";
    // Affiche "approuve ce qu'a fait le sénateur..."
    print substr($chaîne, strpos($chaîne, $terme)) . "\n";
    // Affiche "approuve ce message."
}
?>

```

Problèmes éventuels

Trois problèmes classiques peuvent survenir. Le premier est qu'il faut utiliser l'opérateur triple égal (===) au lieu du double (==) dans une comparaison car l'opérateur triple égal garantit que les valeurs et les types des termes comparés sont les mêmes. Ceci est important parce `strpos()` peut renvoyer 0, alors que `strstr()` et `strrpos()` peuvent renvoyer une chaîne vide ; or ces deux valeurs sont considérées comme `False` avec `==`.

Le second problème est que ces fonctions sont sensibles à la casse ; l'appel `strstr($chaîne, 'Approuve')` dans l'exemple précédent renverrait donc `False`. Les versions insensibles à la casse de `strpos()`, `strrpos()` et `strstr()` s'appellent, respectivement `stripos()`, `strripos()` et `stristr()`.

Enfin, n'oubliez pas que ces fonctions permettent de rechercher de simples sous-chaînes, pas des *mots*. Si vous devez effectuer un traitement plus élaboré, comme rechercher des mots qui commencent par *appro*, comme *approbation* ou *approuve* mais pas *désapprouve*, vous avez besoin d'un mécanisme plus puissant : les expressions régulières. Nous verrons comment les utiliser dans la recette n°39, "Expressions régulières".

Recette 37: Remplacer des sous-chaînes

Utilisez la fonction `str_replace()` pour effectuer un simple remplacement de chaîne. Voici comment remplacer *lapin* par *canard* dans une chaîne tirée d'une bande dessinée :

```

$chaîne = "c'est la saison des lapins !";
print(str_replace("lapin", "canard", $chaîne));

```

Vous remarquerez que `str_replace()` ne remplace pas la sous-chaîne dans la chaîne initiale. Si vous avez besoin de le faire, réassignez la chaîne modifiée à la chaîne initiale :

```
Schaine = str_replace("lapin", "canard", $Schaine);
```

`str_replace()` dispose de plusieurs fonctionnalités supplémentaires. Pour ne remplacer que les n premières occurrences d'une sous-chaîne, ajoutez ce nombre comme quatrième paramètre (c'est particulièrement pratique pour ne remplacer que la première occurrence) :

```
str_replace("lapin", "canard", $Schaine, n);
```

Pour supprimer toutes les occurrences d'une sous-chaîne, il suffit d'utiliser une chaîne vide comme chaîne de remplacement :

```
Schaine = str_replace("lapin", "", $Schaine);
```

Vous pouvez demander à `str_replace()` de remplacer plusieurs sous-chaînes en les plaçant dans un tableau :

```
$mots_affreux = array("hamburger", "bacon", "pot de crème");  
Schaine = str_replace($mots_affreux, "Chez Paulette", $Schaine);
```

Cet exemple remplace toutes les occurrences de *hamburger*, *bacon* et *pot de crème* présentes dans `$Schaine` par *Chez Paulette*.

Vous pouvez aussi fournir deux tableaux de même taille en paramètre, afin que `str_replace()` remplace chaque élément du premier tableau par l'élément au même indice dans le deuxième :

```
$mots_affreux = array("hamburger", "bacon", "pot de crème");  
$remplacements = array("carotte", "brocoli", "crème allégée");  
Schaine = str_replace($mots_affreux, $remplacements, $Schaine);
```

Toutes les occurrences de *hamburger* seront alors remplacées par *carotte*, toutes celles de *bacon* par *brocoli* et toutes celles de *pot de crème* par *crème allégée*.

Problèmes éventuels

Le problème de `str_replace()` est qu'elle remplace tout ce qui correspond au motif que vous lui avez indiqué, où qu'il se trouve dans la chaîne. Si vous remplacez *oui* par *non*, par exemple, vous constaterez que *souillé* s'est transformé en *snollé* et *Louis* en *Lnon*.

Par conséquent, bien que `str_replace()` soit d'une aide inestimable pour supprimer certains mots dans les petites chaînes, vous devrez utiliser des expressions régulières pour les opérations plus complexes (voir la recette n°39, "Expressions régulières").

Recette 38 : Trouver et corriger les fautes d'orthographe avec *pspell*

Vous avez forcément, un jour, fait une faute d'orthographe en saisissant les mots-clés d'une recherche sur Google : *musique alternative*, par exemple. En ce cas, vous avez pu constater que Google essayait de vous aider en affichant *Essayez avec cette orthographe : musique alternative*.

Si votre site propose une fonction de recherche, pouvoir indiquer les fautes d'orthographe lorsqu'aucun résultat (ou trop peu) n'a été trouvé est une fonctionnalité très pratique, surtout si le mauvais Français d'un visiteur peut vous faire rater une vente. Heureusement, le module PHP *pspell* permet de vérifier l'orthographe d'un mot et de suggérer un remplacement à partir de son dictionnaire par défaut (mais vous pouvez aussi créer un dictionnaire personnalisé). Pour commencer, vous devez vous assurer que la bibliothèque *pspell* est installée :

```
<?php
$config_dico = pspell_config_create('fr');
?>
```

Si vous obtenez une erreur, c'est que la bibliothèque n'est pas installée. Revenez à la recette n°18 : "Ajouter des extensions à PHP" pour savoir comment corriger ce problème.

Utiliser le dictionnaire par défaut

Voici une petite fonction pour vous aider à comprendre le fonctionnement de *pspell* :

```
<?php

function suggere_orthographe($chaine) {
    // Suggère les mots possibles en cas de faute d'orthographe
    $config_dico = pspell_config_create('fr');
    pspell_config_ignore($config_dico, 3);
    pspell_config_mode($config_dico, PSpell_FAST);
    $dico = pspell_new_config($config_dico);

    // Pour savoir si l'on a suggéré un remplacement
    $remplacement_suggere = false;
```

```

// pspell est configuré... On découpe la chaîne en mots.
$chaine = explode(' ', $chaine);
foreach ($chaine as $cle => $valeur) {
    $valeur = trim(str_replace(' ', '', $valeur));
    if ( (strlen($valeur) > 3) && (! pspell_check($dico, $valeur)) ) {
        // Si l'on ne trouve pas une suggestion
        $suggestion = pspell_suggest($dico, $valeur);
        // Les suggestions sont sensibles à la casse...
        if (strtolower($suggestion[0]) != strtolower($valeur)) {
            $chaine[$cle] = $suggestion[0];
            $remplacement_suggere = true;
        }
    }
}

if ($remplacement_suggere) {
    // On a une suggestion, donc on revient aux données.
    return implode(' ', $chaine);
} else {
    return null;
}
}
?>

```

Pour utiliser cette fonction, il suffit de lui passer une chaîne en paramètre :

```

<?
// recherche vient du formulaire précédent
$recherche = $_POST['saisie'];

$suggestion_spell = suggere_orthographe($recherche);
if ($suggestion_spell) {
    // pspell a trouvé une suggestion.
    echo "Essayez avec cette orthographe : <i>$suggestion_spell</i>.";
}
?>

```

Si la chaîne de caractères que vous soumettez à pspell est "voici ma phrase mal orthographiée ", le script précédent retournera bien : "Essayez avec cette orthographe : voici ma phrase mal orthographiée ". En revanche, les résultats ne sont pas miraculeux avec des orthographes ou des coquilles extrêmement approximatives, en particulier lorsqu'on n'exploite que la première suggestion délivrée par pspell ! Pour obtenir de meilleurs résultats, vous pouvez utiliser l'ensemble des suggestions offertes par pspell. Le très modeste script suivant renvoie une vingtaine de propositions autour du mot "lappin". N'hésitez pas à l'adapter pour, par exemple, créer une véritable fonction qui accepte un terme en guise de paramètre :

```
<?php
$dico = pspell_new("fr");

if (!pspell_check($dico, "lappin")) {
    $suggestions = pspell_suggest($dico, "lappin");

    foreach ($suggestions as $suggestion) {
        echo "Voulez-vous dire : $suggestion ?<br />";
    }
}
?>
```

Cet exemple renvoie "lapin", "alpin", "lopin", "latin" et toutes les autres orthographes s'approchant du terme mal saisi !

Vous devez configurer un dictionnaire pour initialiser pspell. Pour ce faire, il faut créer un descripteur vers un fichier de configuration de dictionnaire, modifier quelques options de ce descripteur, puis utiliser la configuration de dictionnaire pour créer un deuxième descripteur pour le véritable dictionnaire.

Si cela vous semble un peu compliqué, ne vous inquiétez pas : le code change rarement et vous pouvez généralement vous contenter de le copier à partir d'un autre script. Cependant, nous allons ici l'étudier étape par étape. Voici le code qui configure le dictionnaire :

```
$config_dico = pspell_config_create('fr');
pspell_config_ignore($config_dico, 3);
pspell_config_mode($config_dico, PSPELL_FAST);
```

`$config_dico` est le descripteur initial, qui contrôle les options de votre dictionnaire. Vous devez charger toutes les options dans `$config_dico`, puis vous en servir pour créer le dictionnaire. `pspell_config_create()` crée un dictionnaire français (fr). Pour utiliser la langue anglaise et préciser que vous préférez l'orthographe américaine, passez 'en' en premier paramètre et 'american' en second.

`pspell_config_ignore()` indique à votre dictionnaire qu'il devra ignorer tous les mots de 3 lettres ou moins. En effet, la vérification orthographique de chaque *un* ou *le* serait coûteuse en temps de calcul.

Enfin, `pspell_config_mode()` indique à pspell le mode de fonctionnement choisi :

- `PSPELL_FAST` est une méthode rapide qui renverra le minimum de suggestions.
- `PSPELL_NORMAL` renvoie un nombre moyen de suggestions, à une vitesse normale.

- PSpell_SLOW permet d'obtenir toutes les suggestions possibles, bien que cette méthode demande un certain temps pour effectuer la vérification orthographique.

Nous pourrions utiliser encore d'autres options de configuration (pour ajouter, par exemple, un dictionnaire personnalisé, ainsi que nous le verrons plus loin) mais, comme il s'agit ici d'une vérification rapide, nous nous contenterons de créer le dictionnaire lui-même avec cette ligne :

```
$dico = pspell_new_config($config_dico);
```

À partir de cet instant, vous pouvez utiliser le dictionnaire de deux façons :

1. `pspell_check($dico, mot)` renvoie True si *mot* est dans le dictionnaire.
2. `pspell_suggest($dico, mot)` renvoie un tableau des mots suggérés si *mot* n'est pas dans le dictionnaire (le premier élément de ce tableau est le candidat le plus probable). Si *mot* est dans le dictionnaire, ou si aucune suggestion n'a été trouvée, cet appel ne renvoie rien. Le nombre de mots obtenu varie, mais vous en obtiendrez plus avec PSpell_SLOW et moins avec PSpell_FAST.

NOTE *Ces fonctions ne vérifient pas l'orthographe selon le contexte : Marie a pour du voir sera donc considéré comme correct, bien que vous vouliez écrire Marie a peur du noir. En outre, pspell renvoie toujours True si la longueur du mot est inférieure à celle indiquée par `pspell_config_ignore()` : ici, un mot comme jlz sera donc considéré comme correct.*

Revenons au script initial. Maintenant que le dictionnaire est prêt, nous découpons la chaîne qui a été passée en paramètre pour obtenir un tableau de mots : *voici ma phrase* devient donc un tableau de trois éléments, *voici*, *ma* et *phrase*.

Puis, nous vérifions l'orthographe de chaque mot en utilisant le dictionnaire de pspell. Ce dernier n'aimant pas les virgules, on commence par les supprimer du mot avant de lancer la vérification. Si le mot compte plus de 3 caractères, la vérification a lieu et en cas de faute d'orthographe, nous effectuons les opérations suivantes :

1. Nous demandons à pspell de nous fournir un tableau contenant ses suggestions de correction.
2. Nous prenons la suggestion la plus probable (le premier élément du tableau `$suggestion`) et nous remplaçons le mot mal orthographié par celle-ci.
3. Nous mettons l'indicateur `$remplacement_suggere` à vrai pour qu'à la fin de la boucle de traitement, l'on sache que l'on a trouvé une faute d'orthographe quelque part dans `$chaîne`.

À la fin de la boucle, s'il y a eu des corrections orthographiques, nous reformons une chaîne à partir des éléments du tableau corrigé et nous renvoyons cette chaîne. Sinon, on renvoie null pour indiquer que l'on n'a pas détecté de faute d'orthographe.

Ajouter un dictionnaire personnalisé à pspell

Si un mot ne se trouve pas dans le dictionnaire par défaut, vous pouvez aisément l'y ajouter. Cependant, vous pouvez aussi créer un dictionnaire personnalisé qui sera utilisé avec celui par défaut.

Créez un répertoire sur votre site où PHP a le droit d'écrire et initialisez le nouveau dictionnaire dans celui-ci. Pour créer un nouveau fichier dictionnaire *perso.pws* dans le répertoire chemin de votre serveur, utilisez le script suivant :

```
<?
$config_dico = pspell_config_create('fr');
pspell_config_personal($config_dico, 'chemin/perso.pws');
pspell_config_ignore($config_dico, 2);
pspell_config_mode($config_dico, PSPELL_FAST);
$dico = pspell_new_config($config_dico);
?>
```

C'est le même script que celui de la section précédente, mais avec un ajout essentiel : l'appel à `pspell_config_personal()` initialise un fichier dictionnaire personnel. Si ce fichier n'existe pas déjà, pspell en crée un vide pour vous.

Vous pouvez ajouter à ce dictionnaire autant de mots que vous le souhaitez en utilisant la fonction suivante :

```
pspell_add_to_personal($dico, $mot);
```

Tant que vous n'avez pas sauvegardé le dictionnaire, les mots ne lui sont ajoutés que temporairement. Par conséquent, après avoir inséré les mots souhaités, ajoutez cette ligne à la fin du script :

```
pspell_save_wordlist($dicto);
```

Puis, appelez `pspell_config_personal()` comme ci-dessus dans un autre script et votre nouveau dictionnaire sera prêt à collaborer avec le dictionnaire par défaut. C'est une méthode très intéressante si votre script, ou plus généralement votre site web, manipule des données très spécifiques (un jargon de spécialistes, par exemple) qui n'apparaissent pas dans le dictionnaire français usuel.

Problèmes éventuels

N'oubliez pas que pspell supporte mal les caractères de ponctuation qui, normalement, ne se trouvent pas dans les mots – virgules, points-virgules, deux-points, etc. Vous devez les supprimer des mots avant de les ajouter à votre dictionnaire personnalisé.

Recette 39 : Expressions régulières

Tôt ou tard, vous devrez remonter vos manches et apprendre à utiliser les expressions régulières, car ce sont les outils les plus puissants pour manipuler le texte. Vous recherchez tous les mots entre < et > pour supprimer le code HTML d'une chaîne ? Vous avez besoin des expressions régulières. Vous voulez rechercher toutes les adresses IP contenues dans une chaîne ? Vous voulez vérifier que le mot de passe choisi par un utilisateur n'est pas simplement une suite de chiffres (comme *123456*) ou qu'elle contient un mélange de minuscules et de majuscules ? Vous voulez trier des données dans une base de données or les utilisateurs ont parfois saisi *porteclé*, *porte-clé* ou *porte-clef* ? Pour toutes ces opérations, vous aurez besoin des expressions régulières.

Cela dit, les expressions régulières peuvent être difficiles à lire. L'expression suivante, par exemple, est assez cryptique si vous ne la décortiquez pas caractère par caractère :

```
/^0[1-68]([-. ]?[0-9]{2}){4}$/
```

Bien qu'au premier abord, cela ne ressemble à rien, cette expression régulière permet de trouver un numéro de téléphone dans une chaîne. C'est donc un compromis : les expressions régulières permettent d'effectuer des traitements très puissants mais vous devez les construire caractère par caractère, comme un immense château de cartes ; si vous ratez un seul caractère, tout le motif s'effondre.

Introduction aux expressions régulières

PHP utilise deux types d'expressions régulières : les expressions POSIX étendues et les expressions compatibles Perl. Les noms des fonctions permettant de manipuler les premières commencent généralement par *ereg*, tandis que celles qui manipulent les secondes débutent par *preg*. Ces deux types d'expressions ont des différences de syntaxe mineures ; en outre, leurs performances ne sont pas toujours identiques. La syntaxe compatible Perl étant très connue, ce sont ces fonctions que nous utiliserons ici. L'une de leurs caractéristiques est qu'il faut placer l'expression entre des délimiteurs ; la plupart des développeurs utilisent la barre de fraction car c'est avec elle que l'on recherche une expression régulière dans l'éditeur *vi*.

Supposons que vous recherchiez le mot *fred* dans une chaîne. En ce cas, l'expression régulière est simplement *fred* : il n'y a pas besoin de caractères spéciaux, ni de modificateurs. Voici comment l'utiliser avec la fonction `preg_match()` :

```
if (preg_match('/fred/', "J'ai vu Alfred passer ")) {  
    print "J'ai trouvé fred !";  
}
```

Vous remarquerez que l'expression régulière est une chaîne et que les délimiteurs (les barres de fraction) sont eux-mêmes dans la chaîne. En outre, il est préférable d'utiliser des apostrophes simples afin que PHP n'effectue pas d'expansion des variables et n'interprète pas les séquences d'échappement. En effet, la syntaxe des expressions régulières entre souvent en conflit avec les séquences d'échappement dans les chaînes entre apostrophes doubles.

NOTE *Il est plus pratique d'utiliser un délimiteur différent, la barre droite (|) par exemple, lorsque l'expression contient beaucoup de barres de fraction comme dans les URL ou les chemins.*

Les délimiteurs des expressions régulières compatibles Perl existent essentiellement pour ne pas dépayser ceux qui connaissent déjà Perl et pour troubler les débutants, mais ils autorisent des fonctionnalités supplémentaires, les *modificateurs*, qui sont des caractères placés après le délimiteur fermant. Le plus courant est le modificateur *i*, qui rend l'expression insensible à la casse. Si vous voulez rechercher *fred*, *Fred*, *FRED*, *fRed*, etc., l'expression régulière sera donc `/fred/i`.

Caractères spéciaux

Pour que les expressions régulières soient plus compactes et plus pratiques, vous pouvez utiliser un certain nombre de caractères et de séquences correspondant à des circonstances particulières. Le caractère le plus simple (mais extrêmement utile) est le point (`.`). Il correspond à n'importe quel caractère, sauf le retour à la ligne. L'expression régulière `/f.ed/`, par exemple, capturera *fled*, *fred*, *f!ed*, etc. Pour capturer un vrai point utilisez un anti-slash (cette méthode de désactivation fonctionne pour tous les caractères spéciaux) ; `/f\.red/` ne capturera donc que *f.red*.

Voici d'autres caractères spéciaux utiles :

- `^` correspond au début de la chaîne. `/^fred/` capturera donc *fred* et *frederic*, mais pas *alfred*.
- `$` correspond à la fin de la chaîne. `/fred$/` capturera donc *fred* et *alfred*, mais pas *frederic*.
- `\s` correspond à un caractère d'espacement comme un espace ou une tabulation.
- `\S` correspond à tout caractère qui n'est pas d'espacement.
- `\d` correspond à un chiffre décimal (0–9).

Vous en trouverez bien d'autres dans le manuel de PHP, mais ceux que nous venons de citer sont essentiels.

Itérateurs de motifs

Il est temps de passer à la véritable puissance des expressions régulières en expliquant comment répéter des motifs. Commençons par le caractère astérisque (`*`) qui signifie *capture zéro ou un un nombre quelconque d'occurrences*

du caractère ou de la sous-expression qui me précède. L'expression `/fr*ed/`, par exemple, permet de capturer *fed*, *fred*, *frred*, *frrred*, etc. Combiné avec le caractère point, cet itérateur se comporte donc comme un joker DOS ou shell ; `/fr.*ed/` capturera toute chaîne contenant *fr* suivi d'un nombre quelconque (éventuellement nul) de caractères quelconques, suivis de *ed* : par exemple *fred*, *fried*, *fritterpated*, etc. etc.

Le signe plus (+) est similaire, mais exige *au moins une occurrence* : à part *fred*, `/fr.+ed/` capture donc la même chose que `/fr.*ed/`.

Le point d'interrogation (?) signifie *zéro ou une occurrence*. `/porte-?clef/` capturera donc *porteclef* et *porte-clef*.

Enfin, vous pouvez indiquer un *nombre donné de répétitions* entre accolades ({}). Ainsi, `/fr.{3}ed/` capture tout ce qui contient *fr* suivi de trois caractères quelconques, suivis de *ed*. Vous pouvez aussi donner *un nombre minimum et un nombre maximum d'occurrences* : {3,5}, par exemple, signifie trois à cinq occurrences.

NOTE *N'oubliez pas que vous devez utiliser un anti-slash si vous voulez capturer l'un de ces caractères spéciaux ! Pour capturer *fred?*, par exemple, il faut écrire `/fred\?/`, pas `/fred?/`.*

Groupements

L'astuce suivante consiste à grouper un ensemble de caractères entre parenthèses. Ce groupement permet d'utiliser un itérateur sur une sous-expression au lieu d'un simple caractère. Dans l'expression `/f(re)+d/`, par exemple, `(re)+` signifie qu'il faut capturer *re* au moins une fois ; le motif complet capture donc *fred*, *frered*, *frerered*, etc.

Une autre fonctionnalité que vous trouverez souvent dans les groupements est la barre droite (|) qui demande à PHP de capturer, soit la partie gauche, soit la partie droite de la barre. L'expression `/lait (et|ou) viande/` capturera à la fois *lait et viande* et *lait ou viande*. De même, `/fred(ing|ed)?/` capturera *freding*, *freded* et *fred*.

Classes de caractères

La dernière syntaxe que nous présenterons consiste à utiliser des crochets ([]) pour créer une classe de caractères, c'est-à-dire plusieurs caractères qui seront considérés comme un seul lors de la capture. Un tiret permet de créer un intervalle : [0-9], par exemple, capture n'importe quel chiffre décimal, tandis que [0-9a-fA-F] capture n'importe quel chiffre hexadécimal.

Si le premier caractère après le crochet ouvrant est un accent circonflexe (^), il s'agit d'une classe de caractères inversée, qui correspondra à n'importe quel caractère qui *n'est pas* entre les crochets. `[^0-9]`, par exemple, capturera n'importe quel caractère qui n'est pas un chiffre décimal.

Construction d'une expression régulière

Vous connaissez maintenant les briques essentielles permettant de construire des expressions régulières. Par elles-mêmes, elles ne font pas grand-chose mais, une fois assemblées, elles permettent de construire des expressions très puissantes. Rappelez-vous cependant deux points essentiels : tout d'abord, la création d'expressions complexes demande de la pratique ; ensuite, vous devriez toujours construire les expressions les plus compliquées morceau par morceau. À titre d'exemple, décortiquons l'expression régulière suivante :

```
/<a\s+[^>]*href="[^\"]+"/i
```

Cette expression permet de capturer une balise de lien en HTML. Ses différentes parties sont les suivantes :

1. Toutes les balises de liens commencent par <a, c'est donc le premier composant.
2. Il doit ensuite y avoir au moins un espace ; sinon, on pourrait avoir <arbre>, ce qui n'aurait aucun sens. C'est la raison du \s+.
3. Maintenant, nous voulons capturer tous les caractères de la balise jusqu'au lien, car il pourrait y avoir d'autres attributs comme target="blank". Les balises se terminant par >, nous utilisons [^>]* pour capturer zéro ou plusieurs caractères qui ne sont pas >. Si cette étape ne semble pas évidente, c'est normal : généralement, vous constaterez que vous en avez besoin après avoir écrit le reste de l'expression.
4. Les attributs pour les liens commencent par href=", c'est donc ce qui vient après (pour des raisons de simplicité, nous ignorons pour l'instant le fait que certains ne mettent pas de guillemet).
5. Nous sommes maintenant sur le premier caractère du lien et nous voulons le capturer en entier ; c'est la raison pour laquelle on utilise la même astuce que dans l'étape 3 : [^\"]+ capture tous les caractères du lien (qui en comporte au moins un).
6. Pour capturer le guillemet qui ferme le lien, nous ajoutons un guillemet ("). Ceci pourrait sembler inutile puisque nous avons déjà capturé le lien, mais il est généralement préférable de continuer un peu le motif pour ne pas capturer accidentellement une valeur entièrement fautive.
7. Les noms des balises et des attributs HTML étant insensibles à la casse, nous ajoutons le modificateur i après le délimiteur.

Comme vous pouvez le constater, la recherche par motif avec les expressions régulières est assez simple. Il est maintenant temps de voir comment les utiliser avec PHP.

Recherches et extractions avec les expressions régulières

PHP dispose de plusieurs fonctions pour manipuler les expressions régulières. Les plus simples sont celles qui vous indiquent si elles ont trouvé des correspondances, ce que sont ces correspondances et leurs emplacements. Commençons par la fonction `preg_match()` qui recherche une seule correspondance :

```
<?php
$s = 'blah<a href="a.php">blah</a><a href="b.php">blah</a>';
if (preg_match('/<a\s+[^>]*href="[^"]+"/i', $s, $corresp)) {
    print "correspondance : $corresp[0]";
}
?>
```

Ici, nous recherchons une correspondance avec l'expression régulière de la section précédente. `preg_match()` attend au moins deux paramètres : l'expression et la chaîne à analyser. Le troisième paramètre facultatif `$corresp` est un tableau où `preg_match()` devra placer la première sous-chaîne qui correspond. Ici, le tableau n'ayant qu'un seul élément, on y accède par `$corresp[0]`.

Poursuivons cet exemple en ne capturant que le lien et non pas les caractères qui le précèdent. Pour cela, on peut utiliser un groupement pour délimiter la partie qui nous intéresse :

```
preg_match('/<a\s+[^>]*href="(("[^"]+)"|"/i', $s, $corresp);
```

Le groupement est signalé par les parenthèses autour de `[^"]+`. Si une correspondance est trouvée, `$corresp[0]` sera toujours la totalité de la chaîne capturée, mais `$corresp[1]` contiendra aussi ce qui a été capturé par le premier groupe (*a.php*, ici). S'il y avait eu des groupes supplémentaires dans l'expression, ils auraient été désignés par `$corresp[2]`, `$corresp[3]`, etc.

À ce stade, il est peut être sage de se rappeler de la fonction `print_r()`, qui affiche le contenu complet d'un tableau. Elle peut être vraiment utile afin de savoir comment fonctionne le tableau qui contient les correspondances, notamment si vous souhaitez également connaître l'indice de ces correspondances car le format de ce tableau change lorsque vous avez besoin de cette information. En effet, vous devez alors ajouter le paramètre `PREG_OFFSET_CAPTURE` à l'appel de la fonction :

```
preg_match('/<a\s+[^>]*href="(("[^"]+)"|"/i', $s, $corresp, PREG_OFFSET_CAPTURE);
```

S'il y a une correspondance, `$corresp[0]` et `$corresp[1]` contiennent toujours des informations sur la capture complète et celle du premier groupe mais ce sont maintenant des tableaux et non plus des chaînes. Le premier élément de

chaque tableau est la chaîne capturée et le second est l'indice de début de la capture. Voici le résultat légèrement compacté de `print_r($corresp)` pour notre exemple :

```
Array (
  [0] => Array (
    [0] => <a href="a.php"
    [1] => 4
  )
  [1] => Array (
    [0] => a.php
    [1] => 13
  )
)
```

Extraction de toutes les correspondances

La fonction `preg_match_all()` permet d'extraire toutes les correspondances d'une expression régulière. Elle fonctionne exactement comme `preg_match()`, mais le tableau des correspondances a une structure différente et la valeur de retour est le nombre de correspondances trouvées. Supposons que vous utilisez à nouveau `$corresp` comme tableau résultat avec le même exemple de groupement que précédemment. Désormais, `$corresp[0]` correspond au premier ensemble de correspondances : `$corresp[0][0]` est la sous-chaîne complète qui a été capturée et `$corresp[0][1]` est la chaîne capturée par le premier groupe (*a.php*). `$corresp[1]` est un tableau de même structure correspondant à la correspondance suivante : dans notre exemple, `$corresp[1][1]` contient donc *b.php*.

Tout cela peut vraiment devenir illisible, surtout si vous ajoutez le paramètre `PREG_OFFSET_CAPTURE` puisqu'il remplacera toutes les chaînes du tableau des captures par une autre couche de tableaux, exactement comme avec `preg_match()`. Là encore, `print_r()` peut vous être d'une aide inestimable.

Remplacement de sous-chaînes avec les expressions régulières

La fonction `preg_replace()` se comporte comme `preg_match()`, mais elle remplace également les sous-chaînes et renvoie le résultat. Commençons par un exemple simple, où nous remplaçons toutes les occurrences correspondant à `/fre+d/` par *deb* dans `$s` :

```
print preg_replace('/fre+d/', 'deb', $s);
```

Tout fonctionne bien, mais est perfectible. Supposons que nous voulions utiliser une partie de la chaîne initiale – au lieu de tout remplacer par *deb*, comme précédemment, on veut conserver les *e* de la chaîne. Ici, *fred* deviendra *deb*, *freed* deviendra *deeb*, etc. Pour ce faire, il suffit de grouper la partie concernée

dans l'expression régulière, puis d'utiliser une *référence arrière* dans la chaîne de remplacement. Voici comment faire :

```
print preg_replace('/fr(e+)d/', 'd$1b', $s);
```

Comme précédemment, le groupement est réalisé par les parenthèses et la référence arrière est le \$1 qui signifie *premier groupe* dans la chaîne de remplacement. S'il y a plusieurs groupes, vous pouvez utiliser \$2, \$3, etc. pour les désigner. La capture complète est représentée par \$0.

NOTE *Vous rencontrerez parfois des références arrières avec des anti-slash (\0, \1, \2, etc.) : il s'agit d'une syntaxe ancienne qui a la même signification.*

Recette 40 : Réarranger un tableau

Étudions quelques outils qui utilisent les expressions régulières. Supposons que vous ayez un tableau HTML contenant beaucoup d'entrées de la forme :

```
<tr><td>nom, prénom</td>
<td>adresse</td>
<td>téléphone</td>
</tr>
```

Supposons maintenant que vous deviez modifier son format en celui-ci :

```
<tr><td>prénom</td>
<td>nom</td>
<td>adresse</td>
<td>téléphone</td>
</tr>
```

Grâce aux références arrières, cette transformation peut se faire en une seule étape :

```
$table = preg_replace('/<td>([<]*),\s*([<]*)</td>/',
    '<td>$1</td>' . "\n" . '<td>$2</td>',
    $table);
```

Tout cela mérite quelques explications. J'ai séparé les paramètres sur trois lignes afin de rendre cet appel plus lisible. Vous noterez qu'il y a deux groupes et, qu'avant le second, \s* capture tous les espaces en trop. Enfin, vous remarquerez que la chaîne de remplacement est produite par concaténation car on a besoin d'un retour à la ligne, or ce caractère doit être dans une chaîne entre apostrophes doubles pour pouvoir être interprété.

Vous pouvez alors annoncer à votre chef que la transformation vous prendra un temps fou et vous avez maintenant du temps pour vous amuser avec votre console de jeu.

Recette 41 : Extraire des données des pages

Un "screen scraper" est un programme qui accède à une page web et parcourt son code HTML pour en extraire les données intéressantes. En voici un très simple, permettant d'extraire les liens hypertextes d'une page et de les classer en catégories. Cet extracteur utilise de nombreuses expressions régulières que nous étudierons une à une. Nous vérifions d'abord que l'entrée (dans `$_REQUEST["page"]`) est bien un lien et non une tentative d'accéder aux fichiers du système local :

```
<?php

$page = $_REQUEST["page"];

if (!preg_match('^https{0,1}://|', $page)) {
    print "L'URL $page est incorrecte ou non reconnue.";
    exit;
}
```

Lorsque ce test a été effectué, nous pouvons passer à la lecture des données et à l'extraction des liens placés dans les balises (voir l'exemple de la section "Construction d'une expression régulière", plus haut dans ce chapitre). Vous remarquerez que nous utilisons simplement la fonction `file_get_contents()` au lieu de `cURL` car nous n'avons pas besoin des fonctionnalités avancées de ce dernier, comme l'authentification HTTP et la gestion des cookies.

```
$donnees = file_get_contents($page);
preg_match_all('<a\s[>]*href="(^[^"]+)"|i', $donnees, $corresp);
```

Tous les liens sont maintenant dans `$corresp[1]` (rappelez-vous que `$corresp[0]` contient tout ce qui a été capturé). Initialisons quelques tableaux qui serviront plus tard à stocker et à classer les liens :

```
$tous_les_liens = array();
$liens_js = array();
$liens_complets = array();
$liens_locaux = array();
```

Nous pouvons maintenant parcourir tous les liens pour effectuer le classement. On teste d'abord que le lien n'a pas déjà été rencontré et, si ce n'est pas le cas, on utilise plusieurs expressions régulières pour déterminer sa catégorie :

```

foreach ($corresp[1] as $lien) {
    if ($tous_les_liens[$lien]) {
        continue;
    }
    $tous_les_liens[$lien] = true;

    if (preg_match('/^JavaScript:/', $lien)) {
        $liens_js[] = $lien;
    } elseif (preg_match('/^https{0,1}:/i', $lien)) {
        $liens_complets[] = $lien;
    } else {
        $liens_locaux[] = $lien;
    }
}

```

On peut alors afficher le résultat de l'analyse (voir la Figure 5.1) :

```

print '<table border="0">';
print "<tr><td>Nombre de liens :</td><td>";
print strval(count($corresp[1])) . "</td></tr>";
print "<tr><td>Liens uniques :</td><td>";
print strval(count($tous_les_liens)) . "</td></tr>";
print "<tr><td>Liens locaux :</td><td>";
print strval(count($liens_locaux)) . "</td></tr>";
print "<tr><td>Liens complets :</td><td>";
print strval(count($liens_complets)) . "</td></tr>";
print "<tr><td>Liens JavaScript :</td><td>";
print strval(count($liens_js)) . "</td></tr>";
print '</table>';
?>

```

Nombre de liens : 210
Liens uniques : 141
Liens locaux : 89
Liens complets : 36
Liens Javascript : 16

Figure 5.1 : Résumé des liens

Amélioration du script

Un script comme celui-ci peut s'étendre à l'infini : vous pouvez classer les liens selon vos besoins, en suivre certains, etc. Un exercice très utile consiste à décomposer l'URL initiale en ses différents composants afin de transformer les liens locaux (relatifs) en URL complètes (absolues).

Recette 42 : Convertir du texte normal en document HTML

L'un des ennuis de la programmation web est que les documents en texte pur ne sont pas compatibles avec ceux en HTML, bien qu'ils soient souvent utilisés de concert. Ce que les utilisateurs tapent dans les champs des formulaires, par exemple, est du texte pur mais il y a de fortes chances qu'ils veuillent l'afficher en HTML. Voici un exemple de texte qui pourrait avoir été saisi dans un formulaire :

Bonjour à tous,

Suite à ma thérapie, une partie de mon cerveau a été supprimée. Je vais bien et je peux maintenant regarder les films d'Éric et Ramzy avec mes amis.

Sincèrement,

Fred

Comme il n'y a pas de balises `
` ou `<P>` pour créer des coupures de lignes, ce texte s'affichera de la façon suivante s'il est simplement injecté dans un navigateur web :

Bonjour à tous, Suite à ma thérapie, une partie de mon cerveau a été supprimée.

Je vais bien et je peux maintenant regarder les films d'Éric et Ramzy avec mes amis.

Sincèrement, Fred

Bien qu'il soit dangereux de recopier aveuglément tout ce qu'a saisi l'utilisateur puisque cela vous expose à des attaques XSS, ignorons ce problème pour le moment et supposons que vous ayez une entière confiance en cet utilisateur.

Le moyen le plus simple de convertir les retours à la ligne en HTML consiste à utiliser la fonction PHP `n2br()`. Malheureusement, cette méthode n'est pas très souple car elle traduit chaque retour à la ligne en balise `
` : si votre texte brut contient également du code HTML, vous risquez donc d'obtenir un résultat assez illisible. Considérons, par exemple, cet extrait HTML :

```
<table>
<tr><td>Je suis une ligne du tableau
</td></tr>
</table>
```

`n2br()` le transforme pour obtenir :

```
<table><br>
<tr><td>Je suis une ligne du tableau<br>
</td></tr><br>
</table><br>
```

Ce code aura donc un rendu différent de celui de l'original. Au cours de mes recherches d'une meilleure solution, j'ai découvert la fonction `autop()` écrite par Matthew Mullenweg, le développeur-fondateur de WordPress :

```

<?
function autop($pee, $br = 1) {
    // Convertit du texte brut en HTML

    $pee = $pee . "\n"; // On marque la fin pour faciliter le traitement.
    $pee = preg_replace('|<br />\s*<br />|', "\n\n", $pee);
    $pee = preg_replace('!(<?(?:table|ul|ol|li|pre|form|blockquote|
        h[16])[^>]*>)!', "\n$1", $pee); // Espace un peu...
    $pee = preg_replace('!(</?(?:table|ul|ol|li|pre|form|blockquote|
        h[16])>)!', "$1\n", $pee); // Espace un peu...
    $pee = preg_replace("/(\r\n|\r)/", "\n", $pee); // Retours à la ligne
        // portables.
    $pee = preg_replace("/\n\n+/", "\n\n", $pee); // Gère les doublons.
    $pee = preg_replace('/\n?(.+?)(?:\n\s*\n|\z)/s', "\t<p>$1</p>\n",
        $pee);
    // Crée les paragraphes, dont un à la fin.
    $pee = preg_replace('<p>\s*</p>|', '', $pee); // Dans certaines
    // conditions, cela pourrait créer un P ne contenant que des espaces.
    $pee = preg_replace("<p>( <li.+?</p>|", "$1", $pee); // Problème avec
    // les listes imbriquées.
    $pee = preg_replace('<p><blockquote([>]*)>|i', "<blockquote$1><p>",
        $pee);
    $pee = str_replace('</blockquote></p>', '</p></blockquote>', $pee);
    $pee = preg_replace('<p>\s*(</?(?:table|tr|td|th|div|ul|ol|
        li|pre|select|form|blockquote|p|h[1-6])[^>]*>|',
        "$1", $pee);
    $pee = preg_replace('!(</?(?:table|tr|td|th|div|ul|ol|li|pre|select|
        form|blockquote|p|h[1-6])[^>]*>)\s*</p>!', "$1",
        $pee);
    // Crée éventuellement des retours à la ligne
    if ($br) $pee = preg_replace('|(?<!<br />)\s*\n|', "<br />\n", $pee);
    $pee = preg_replace('!(</?(?:table|tr|td|th|div|dl|dd|dt|ul|ol|li|pre|
        select|form|blockquote|p|h[1-6])[^>]*>)\s*<br />!',
        "$1", $pee);
    $pee = preg_replace('<br />(\s*</?(?:p|li|div|th|pre|td|ul|ol)>|',
        '$1', $pee);
    $pee = preg_replace('/&([^\#])(?![a-z]{1,8};)/', '&#038;$1', $pee);

    return $pee;
}
?>

```

autop() attend un unique paramètre, la chaîne à filtrer, et elle renvoie la chaîne filtrée. Ce script est suffisamment malin pour conserver les éléments des blocs HTML (les tableaux et les listes formatées n'auront pas de
 insérés aléatoirement lorsqu'il y a des espaces) tout en ignorant les <P> et les
. Appliquée à ce texte, par exemple :

```
print autop('Un haïku est formé de cinq  
<br />
```

```
Syllabes placées verticalement  
Sept au milieu');
```

vous obtiendrez le résultat suivant :

```
<p>Un haïku est formé de cinq</p>  
<p>Syllabes placées verticalement<br />  
Sept au milieu  
</p>
```

La fonction `autop()` est découpée en trois phases : nettoyage, remplacement puis suppression des balises. Avant de détailler chacune d'elles, nous devons examiner les modificateurs employés dans cette fonction.

La syntaxe `(?:truc)`, notamment, représente un groupement identique à `(truc)`, mais `?:` demande à l'analyseur d'expressions régulières de ne pas créer de référence arrière pour ce groupe ; vous voulez connaître la valeur qui a été capturée par ce groupe, mais vous souhaitez également utiliser les fonctionnalités des groupes – répétitions du groupe, options, etc. Cette syntaxe laisse une plus grande marge à la personnalisation, notamment lorsque vous devez insérer un groupe après avoir écrit beaucoup de code dépendant d'une référence arrière. Étudiez par exemple cette expression extraite du code de `autop()` :

```
!(/(?:table|ul|ol|li|pre|form|blockquote|h[1-6])>)!)
```

Elle signifie *capture toute balise table, ul, ol, li, pre, form, blockquote ou h1 à h6*. Mais, ici, tout ce qui intéresse Matthew Mullenweg est le groupement, car il lui permet de préciser plusieurs balises dans la même expression régulière.

Sans faire une analyse ligne par ligne de cette fonction, nous pouvons décrire le déroulement de `autop()` :

1. **Nettoyage** : `autop()` remplace toutes les instances répétées de `
` par des retours à la ligne et corrige tous les retours à la ligne pour qu'ils fonctionnent avec Unix (ce dernier utilise un unique caractère `\n`, alors que Windows utilise `\r\n`) ; elle réduit également les longues chaînes de retours à la ligne en deux retours à la ligne.
2. **Remplacement** : `autop()` recherche tous les retours à la ligne suivis de texte, lui-même suivi de deux retours à la ligne et remplace les retours à la ligne par des balises de paragraphes, comme dans `<p>truc</p>` .
3. **Suppression des balises** : `autop()` recherche les balises de blocs HTML (comme les listes numérotées et les tableaux) qui seraient perturbées par `<p>` ou `
` et les supprime.

Recette 43 : Créer des liens automatiques vers les URL

La plupart des logiciels de forums de discussion ou de blog convertissent automatiquement en liens hypertextes les URL postées dans les articles et les commentaires. Vous pourriez penser qu'il suffit de capturer `http://` puis d'utiliser une référence arrière pour encadrer l'URL par une balise de lien, mais si cette URL comporte déjà une balise, cela créera une belle confusion !

Vous devez donc vous assurer que l'URL n'est pas déjà dans une balise. Vous pourriez vouloir utiliser le modificateur de groupement `?!`, qui permet de *rejeter tout ce qui correspond au groupe* mais cela ne fonctionne que si le groupe non désiré suit celui que vous voulez capturer – les expressions régulières ne traitent qu'un caractère à la fois et ne reviennent jamais en arrière. Vous avez donc besoin de la fonctionnalité appelée *assertion de recherche vers l'arrière*, qui signifie essentiellement *vérifie cette condition lorsque l'on trouve une correspondance plus loin dans l'expression*. Pour représenter une assertion de recherche vers l'arrière négative, il faut utiliser le modificateur de groupement `?<!`.

Ceci étant dit, voici le code permettant de créer automatiquement des liens pour les URL, lorsque vous ne voulez pas perturber tout ce qui est préfixé par `href="` dans une balise de lien HTML :

```
preg_replace('|(?<!href=")(https?://[A-Za-z0-9+\=._/*()@\$;:&!%]+)|i',  
            '<a href="$1">$1</a>',  
            $chaine);
```

L'essentiel de cette expression régulière est une classe de caractères contenant ceux admis dans une URL. Différentes variations sur ce thème sont évidemment possibles : vérification de la validité du nom de domaine, recherche d'un point ou d'une virgule finale, etc. Mais ne vous laissez pas trop emporter.

Recette 44 : Supprimer les balises HTML contenues dans une chaîne

Bien que les expressions régulières soient utiles, elles ne constituent pas la solution universelle à tous les problèmes. Supposons, par exemple, que vous comptiez afficher du texte éventuellement codé en HTML dans une autre page HTML et que vous devez donc supprimer toutes les balises qui risquent de poser problème. La fonction `strip_tags()` permet d'effectuer ce traitement :

```
<?php  
print strip_tags($chaine);  
?>
```

Pour indiquer les balises que vous voulez malgré tout autoriser, passez à la fonction un paramètre supplémentaire, une chaîne contenant les balises autorisées (n'indiquez que les balises ouvrantes) :

```
print strip_tags($chaine, "<i><b><p>");
```

Cette fonction n'est qu'un exemple des nombreuses fonctionnalités qu'offre PHP pour le traitement du texte, celles-ci constituant une alternative aux expressions régulières ou aux autres solutions maison. Il est toujours intéressant de parcourir le manuel de PHP pour voir s'il existe une réponse toute prête à vos besoins, mais n'ayez pas peur de mettre les mains dans le cambouis si nécessaire.

6

TRAITEMENT DES DATES



Ce chapitre explique comment traiter les dates et les temps avec PHP. Les programmes web impliquent souvent un nombre important d'extractions, de manipulations, de comparaisons et d'affichages des dates et cela devient encore plus évident lorsqu'il s'agit d'ajouter MySQL à l'ensemble. Mais commençons d'abord par présenter le format natif utilisé par votre serveur pour représenter les dates et les temps.

Représentation du temps avec Unix

Avant d'entrer dans les détails de la magie des dates, nous devons présenter la façon dont les serveurs Unix/Linux (et donc PHP) conformes à POSIX stockent les valeurs temporelles. Heureusement, ce système est assez simple.

Sur les ordinateurs Unix, le temps "commence" le premier janvier 1970 à minuit. En termes Unix, cet instant est l'instant 0 et s'appelle *l'epoch*.

Toute date depuis cet instant précis est représentée par le nombre de secondes écoulées depuis l'epoch. Si, par exemple, ce texte est écrit le 11 janvier 2008 à 12 h 17 mn et 25 sec, il se sera écoulé 1 200 082 645 secondes et c'est cette valeur qui représentera cet instant.

Ce principe simplifie la manipulation des dates en PHP tant qu'il ne s'agit pas de les afficher. Vous pouvez, par exemple, passer au jour suivant en ajoutant ($60 * 60 * 24$) secondes au temps courant ou revenir une heure en arrière en soustrayant 3600 secondes. Le Tableau 6.1 résume les valeurs classiques utilisées avec les temps.

Tableau 6.1: Incréments de temps classiques mesurés en secondes

Secondes	Temps
60	Une minute
3600	Une heure
28800	Huit heures
86400	Un jour
604800	Une semaine

La valeur stockée pour représenter le temps n'étant qu'un nombre, vous pouvez aisément comparer deux instants : la valeur la plus grande des deux correspond alors à l'instant le plus proche de nous. Ceci est particulièrement utile pour comparer des instants par rapport au moment présent.

Outre l'extraction, la manipulation et le stockage des temps, ce chapitre explique comment les convertir dans un format lisible et comment obtenir d'autres informations, comme le jour de la semaine. Nous étudierons également la manipulation des formats des dates avec MySQL.

Recette 45 : Connaître l'instant courant

Pour connaître l'instant courant, il suffit d'appeler la fonction `time()` sans paramètre. En ce cas, cet appel renvoie la valeur de l'instant courant, telle qu'elle est stockée sur le serveur. Ce code, par exemple, affiche l'heure courante :

```
echo "L'instant courant est " . time();
```

Vous pouvez également stocker ce résultat de cette fonction dans une variable :

```
$temps = time();
```

De nombreuses fonctions PHP sur les dates attendent un instant en paramètre et la plupart utilisent l'instant courant si vous ne leur fournissez pas cette information. La fonction `date()`, par exemple, renvoie une chaîne formatée selon vos envies à l'aide de la syntaxe `date("format", instant)` mais, si vous ne fournissez pas *instant*, elle renverra l'instant courant formaté selon *format*.

Cela signifie que, dans la plupart des cas, vous n'avez même pas besoin d'employer `time()` si vous voulez travailler sur le moment présent.

NOTE *Un instant est toujours exprimé en UTC (Coordinated Universal Time, un standard international très précis). Cependant, les fonctions d'affichage des dates et des temps utilisent la zone horaire du serveur pour en déduire l'heure locale à afficher. L'instant 1136116800, par exemple, représentera une heure du matin sur un serveur utilisant la zone MET (Middle European Time), mais sept heures du soir le jour précédent sur un serveur de la zone EST (Eastern Standard Time).*

Recette 46 : Obtenir l'instant correspondant à une date du passé ou du futur

Voyons maintenant comment obtenir les instants de dates proches de l'instant courant : celle d'hier ou de vendredi prochain, par exemple. Il existe deux méthodes générales pour y parvenir :

- utiliser une chaîne ;
- utiliser les valeurs d'une date (ce qui peut être un peu plus compliqué).

Création d'instants à partir d'une chaîne

La fonction `strtotime()` est parfois la meilleure amie du programmeur PHP. Comme son nom l'indique, elle produit une valeur temporelle à partir d'une chaîne contenant une date exprimée en anglais, comme *April 1* ou *Friday*. Ces chaînes peuvent être relatives à l'instant présent ou être des dates absolues. Le Tableau 6.2 présente quelques chaînes possibles.

Tableau 6.2 : Exemples d'utilisation de `strtotime()`

Appel	Résultat (sous forme d'instant)
<code>strtotime("Friday")</code>	Vendredi à minuit
<code>strtotime("+1 week Friday")</code>	Vendredi de la semaine prochaine à minuit
<code>strtotime("+1 week")</code>	Dans une semaine à partir de maintenant
<code>strtotime("-2 months")</code>	Il y a deux mois
<code>strtotime("October 1, 2008")</code>	Le 1 ^{er} octobre 2008 à minuit
<code>strtotime("2008-10-01")</code>	Le 1 ^{er} octobre 2008 à minuit
<code>strtotime("Friday 12:01 p.m.")</code>	Vendredi à 12h 01mn
<code>strtotime("+7 days 12:01 p.m.")</code>	Dans sept jours à 12h 01mn

Bien que la plupart des formats de date soient reconnus par `strtotime()`, certains peuvent poser problème et il n'est pas évident de distinguer ceux qui fonctionnent de ceux qui ne fonctionnent pas ; `strtotime("2008-10-01")` se comporte

bien, par exemple, alors que `strtotime("10-01-2008")` produit un instant incorrect qui correspond au 28 juin 2015. Pour obtenir des résultats cohérents, utilisez toujours des dates au format ISO 8601 complet (voir la page <http://www.cl.cam.ac.uk/~mgk25/iso-time.html>), c'est-à-dire de la forme `AAAA-MM-JJ`. Ce format est particulièrement intéressant parce que c'est l'un de ceux que MySQL comprend, comme nous le verrons dans la section "Formats des dates MySQL", à la fin de ce chapitre.

Si `strtotime()` est incapable de traduire votre requête, elle renverra soit `false` (à partir de PHP 5.1), soit `-1` (avec les versions plus anciennes). Ce `-1`, notamment, peut induire en erreur car il risque d'être interprété comme le 31 décembre 1969, *une seconde avant l'époque*.

Vérification des dates avec `strtotime()`

`strtotime()` permet également de vérifier qu'une date appartient à un intervalle correct. Si, par exemple, vous autorisez les utilisateurs à saisir des dates de la forme `2008-01-01`, vous pouvez vérifier que leurs saisies sont correctes en appelant `strtotime()` mais cela ne vous indiquera pas si la date est autorisée : un utilisateur pourrait saisir `yesterday` et l'appel renverrait `true`. Si vous placez ces dates dans une base de données, par exemple, vous devriez donc ajouter une couche de vérification supplémentaire pour garantir que le contenu du champ est valide. Le script suivant contrôle une date fournie par l'utilisateur pour vérifier qu'elle appartient bien à un intervalle correct, puis la met dans un format reconnu par la base :

```
<?php
$date_utilisateur = $_GET['date'];
$instant_utilisateur = strtotime($date_utilisateur);

// Début/fin de l'intervalle de date autorisé.
$date_deb = strtotime('2008-01-01');
$date_fin = strtotime('2009-01-01');

if ($instant_utilisateur < $date_deb ||
    $instant_utilisateur >= $date_fin) {
    die("La date n'appartient pas à l'intervalle autorisé");
}

$date_base = date('Y-m-d', $instant_utilisateur);
// On peut alors utiliser $date_base dans une requête SQL
?>
```

NOTE *La fonction `date()` sera présentée plus loin dans ce chapitre.*

Création d'instants à partir de dates

Si vous connaissez déjà la date et l'heure précises dont vous avez besoin, vous pouvez créer l'instant correspondant avec la fonction `mktime()` en lui fournissant l'heure, les minutes, les secondes, le numéro du mois, le jour et l'année. Voici un exemple :

```
$date_future = mktime($heure, $minutes, $secondes, $mois, $jour, $annee);
```

Vous pouvez omettre des paramètres en partant de la droite ; dans ce cas, `mktime()` utilisera les valeurs de l'instant courant. `mktime(12, 00, 0)`, par exemple, crée l'instant qui représente midi pour aujourd'hui. Les paramètres doivent être de vrais nombres et non des chaînes de caractères. Ainsi `mktime('12', '05', '2008')` ne correspond pas au 5 décembre 2008 ni au 12 mai 2008 mais au jour présent, à 12h05 et 2008 secondes (soit 12h38min28sec).

NOTE *Si vous créez une date avec `mktime()`, il est généralement préférable d'utiliser une heure valant 12 par défaut, juste au cas où le serveur aurait des problèmes avec les zones horaires. En effet, certains sont configurés pour utiliser UTC et, si PHP utilise la zone EST, la création d'un instant pour une date débutant à minuit peut ne pas donner cette date pendant quatre heures.*

Une astuce classique consiste à utiliser les fonctions `date()` (voir la recette n°47 : "Formater les dates et les heures") et `strtotime()` pour obtenir le jour et le mois courants à utiliser avec `mktime()`.

Cette ligne de code, par exemple, utilise `date()` pour obtenir le mois courant au format 01-12, puis l'insère comme valeur du paramètre mois à l'appel de la fonction `mktime()` :

```
$premier_jour_du_mois = mktime(0, 0, 0, intval(date("m")), 1);
```

Cet extrait emploie `strtotime()` pour obtenir le mois suivant sous forme d'instant, puis convertit cet instant à la fois en numéro de mois sur deux chiffres et en année sur quatre chiffres pour les utiliser avec `mktime()` (vous remarquerez que l'on utilise ici explicitement l'année pour prendre en compte le cas où l'on passerait de décembre à janvier, puisque le mois suivant serait alors également l'année suivante).

```
$mois_suivant = strtotime("+1 month");
$premier_jour_mois_suiv = mktime(0, 0, 0,
                                intval(date("m", $mois_suivant)), 1,
                                intval(date("y", $mois_suivant)));
```

Parfois, la fonction `mktime()` ne suffit pas car on ne travaille pas seulement avec des jours et des mois, mais également avec des semaines : c'est dans ces moments-là que l'on a besoin de `strtotime()`.

Enfin, la fonction `checkdate()` est souvent utilisée avec `mktime()`, car elle vérifie qu'un jour, un mois et une année forment bien une date du calendrier. Ainsi, `checkdate(12, 31, 2008)` renvoie `true`, alors que `checkdate(2, 31, 2008)` renvoie `false`.

Maintenant que nous savons créer et manipuler des dates en PHP, voyons comment les afficher.

Recette 47 : Formater les dates et les heures

Vu que nous sommes peu habitués à exprimer les dates en secondes, il est préférable d'utiliser un format comme *15 Oct 2008* lorsque l'on affiche les dates.

Nous allons donc présenter ici le fonctionnement de `date()` que nous avons déjà rencontrée plus haut dans ce chapitre. Cette fonction renvoie une chaîne à partir de ses deux paramètres qui sont, respectivement, un format de date (comme `'j M Y'`) et un instant. L'appel `date('j M Y', 1151884800)`, par exemple, produit la chaîne *2 Jul 2006* dans la zone PST. Si vous omettez le deuxième paramètre, `date()` utilise l'instant courant.

La chaîne de format peut contenir d'autres caractères, comme des deux-points et des virgules mais, à la différence des autres fonctions attendant un format, comme `printf`, vous devez faire très attention à ne pas utiliser des caractères réservés au formatage car il n'y a pas de préfixe de format. Le Tableau 6.3 provient directement de la documentation officielle de PHP et présente les différentes chaînes de format.

Tableau 6.3 : Chaînes de format de `date()`

Caractère de format	Description	Exemple de résultat
Jour		
d	Jour du mois sur deux chiffres	01 à 31
D	Représentation textuelle du jour sur trois lettres	Mon à Sun
J	Jour du mois sans zéro de tête	1 à 31
l (L minuscule)	Représentation textuelle complète du jour de la semaine	Sunday à Saturday
S	Suffixe ordinal anglais du jour du mois, sur deux caractères	'st', 'nd', 'rd' ou 'th' ; marche bien avec j
W	Représentation numérique du jour de la semaine	0 (Dimanche) à 6 (Samedi)

Tableau 6.3 : Chaînes de format de date()

Z	Jour de l'année	0 à 365
Semaine		
W	Numéro de semaine ISO-8601 de l'année. Les semaines commencent le Lundi (à partir de PHP 4.1.0)	42 (42 ^e semaine de l'année)
Mois		
F	Représentation textuelle complète du mois, comme <i>January</i> ou <i>March</i> .	January à December
m	Représentation numérique du mois, avec des zéros en tête.	01 à 12
M	Représentation textuelle du mois abrégée en trois lettres	Jan à Dec
N	Représentation numérique du mois, sans zéro de tête	1 à 12
T	Nombre de jours dans le mois	28 à 31
Année		
L	Année bissextile	1 si l'année est bissextile, 0 sinon
Y	Représentation sur quatre chiffres	1999 ou 2003
Y	Représentation sur deux chiffres	99 ou 03
Heure		
A	Représentation minuscule de AM et PM	am ou pm
A	Représentation majuscule de AM et PM	AM ou PM
B	Heure Internet Swatch	000 à 999
G	Format 12 heures sans zéro de tête	1 à 12
G	Format 24 heures sans zéro de tête	0 à 23
H	Format 12 avec zéros de tête	01 à 12
H	Format 24 heures avec zéros de tête	00 à 23
I	Minutes, avec zéros de tête	00 à 59
S	Secondes, avec zéros de tête	00 à 59

Tableau 6.3 : Chaînes de format de date()

Zone horaire		
I	Heure d'été	1 si heure d'été, 0 sinon
O	Différence en heures par rapport à UTC	+0200
T	Zone horaire de cette machine	EST, MDT, etc.
Z	Décalage de la zone en secondes. Ce décalage est toujours négatif pour les zones à l'ouest de UTC et toujours positif pour les zones à l'EST de UTC	-43200 à 43200
Date/heure complète		
C	Date au format ISO 8601 (à partir de PHP 5)	2008-12-18T16:01:07 +02:00
R	Date au format RFC 2822	Thu, 18 Dec 2008 16:01:07 +0200

Les caractères non reconnus dans la chaîne de format sont affichés tels quels : si vous voulez utiliser littéralement l'un des caractères de format, vous devez donc le protéger par un anti-slash ("`\e j \d \e M`"). Cependant, on a tôt fait de se perdre dans les anti-slash puisqu'ils sont également utilisés par les séquences d'échappement.

Le Tableau 6.4 présente quelques exemples de formats de dates.

Tableau 6.4 : Exemples de chaînes produites par date()

Chaîne de format	Exemple de résultat
l (L minuscule)	Saturday
M	Oct
H:m	1:36
G:i:s A	5:26:01 PM
d-m-Y	04-10-2008
j M y	1 Jun 08
d M Y h:m:s a	16 Aug 2008 12:08:00 am

Formater les dates en français

Dans la plupart des scripts que vous développerez, vous devrez vraisemblablement manipuler des dates en notation française. Il existe plusieurs méthodes pour aboutir rapidement à ce formatage, mais vous avez tout intérêt à utiliser la fonction `strftime()`. En guise de paramètres, elle accepte le formage spécial et

éventuellement un instant spécifique (sinon, c'est l'heure courante qui sera utilisée). En parallèle, la fonction `setlocale()` modifie les informations de localisation renvoyées par le serveur : en personnalisant la constante `LC_TIME`, on change le format d'heure. Ainsi, l'exemple suivant :

```
setlocale(LC_TIME, 'fr', 'fr_FR', 'fr_FR.ISO8859-1');
echo strftime("%A %d %B %Y.");
```

affichera "lundi 18 août 2008". Nous avons tout d'abord modifié le format d'heure (les paramètres "fr", "fr_FR" et "fr_FR.ISO8859-1" correspondent à tous les types de valeurs attendues par les serveurs). Puis nous appelons la fonction `strftime()` sur la date courante. Le formatage est spécifique à cette fonction : ici, `%A` est le nom complet du jour, `%d` sa valeur numérique, `%B` le nom complet du mois et `%Y` l'année. Vous retrouverez la liste complète des caractères de formatage à l'adresse <http://tinyurl.com/5usynv>.

Si vous n'avez pas la possibilité d'exécuter `setlocale()` sur votre serveur, vous pouvez vous rabattre sur une solution plus simple, qui exploite la fonction `date()`. En voici un exemple :

```
$liste_jours = array("dimanche", "lundi", "mardi", "mercredi", "jeudi",
"vendredi", "samedi");
$liste_mois = array("", "janvier", "février", "mars", "avril", "mai", "juin",
"juillet", "août", "septembre", "octobre", "novembre", "décembre");
list($nom_jour, $jour, $mois, $annee) = explode('/', date("w/d/n/Y"));
echo $liste_jours[$nom_jour].' '.$jour.' '.$liste_mois[$mois].' '.$annee;
```

Ici, nous créons deux tableaux contenant le nom de tous les jours et de tous les mois. Nous récupérons la date du jour et nous isolons chaque information (jour, mois, année) dans une variable. Il ne nous reste plus qu'à parcourir les deux tableaux afin d'afficher une date complète, "mardi 28 juillet 2009" par exemple.

Recette 48 : Calculer le jour de la semaine d'une date

Cette recette est la suite logique de la recette précédente ; elle explique comment obtenir des informations très spécifiques à partir d'une date.

Là encore, la clé consiste à obtenir l'instant correspondant à une date :

```
<?php
$instant = strtotime("2008-07-03");
$jour_de_la_semaine = date('l', $instant);
echo 'Le jour de la semaine est ' . $jour_de_la_semaine;
?>
```

Ce script comporte trois étapes très simples :

1. Il stocke dans \$instant l'instant correspondant au 3 juillet.
2. Il utilise la fonction date() pour extraire le jour de la semaine de \$instant.
3. Il affiche le jour de la semaine.

Recette 49 : Calculer la différence entre deux dates

Si vous devez trouver le temps qui s'est écoulé entre deux dates, voici ce dont vous avez besoin :

```
<?php

function calcule_diff_temps($instant1, $instant2, $unite_temps) {
// Calcule la différence entre deux instants
    $instant1 = intval($instant1);
    $instant2 = intval($instant2);
    if ($instant1 && $instant2) {
        $ecart_temps = $instant2 - $instant1;
        $secondes_en_unites = array(
            'secondes'=> 1,
            'minutes' => 60,
            'heures' => 3600,
            'jours' => 86400,
            'semaines' => 604800,
        );

        if ($secondes_en_unites[$unite_temps]) {
            return floor($ecart_temps/$secondes_en_unites[$unite_temps]);
        }
    }

    return false;
}
?>
```

La fonction calcule_diff_temps attend trois paramètres : le premier et le deuxième instants, ainsi que l'unité dans laquelle exprimer cette différence, en secondes, heures, jours ou semaines.

Elle commence par transtyper les instants en valeurs numériques, puis soustrait le premier instant du second afin de déterminer le nombre de secondes entre les deux (n'oubliez pas que les instants étant simplement des nombres de secondes écoulées depuis l'époque, c'est-à-dire le premier janvier 1970 à minuit ;

leur différence est un nombre de secondes). Puis, elle utilise un tableau pour trouver l'unité dans laquelle l'utilisateur souhaite obtenir son résultat. Ces unités sont exprimées en secondes. Si l'unité indiquée est correcte, la fonction divise le nombre total de secondes par le nombre de secondes de cette unité. Si, par exemple, l'utilisateur a choisi d'obtenir un résultat en minutes et que la différence entre les deux instants est de 60 secondes, la valeur renvoyée sera donc 1 (60 divisé par 60).

Si l'utilisateur a choisi une unité non reconnue (ou a fourni des instants incorrects), la fonction renvoie `false` par défaut.

Utilisation du script

Cet exemple traduit une différence de sept jours dans toutes les unités :

```
<?php
// Prend comme exemples l'instant courant et sept jours plus tard
$instant1 = time();
$instant2 = strtotime('+7 days');

$unites = array("secondes", "minutes", "heures", "jours", "semaines");
foreach ($unites as $u) {
    $nunités = calcule_diff_temps($instant1, $instant2, $u);
    echo $nunités . " $u se sont écoulés entre " . date("d-m-Y", $instant1)
        . ' et ' . date("d-m-Y", $instant2);
    print "\n";
}
?>
```

Amélioration du script

Cette fonction peut renvoyer des valeurs négatives si le deuxième instant est avant le premier. Si, par exemple, `$instant1` est le 7 juillet 2008 et `$instant2` le 1 juillet 2008, la différence renvoyée est de -6 jours. Si seule la différence vous importe, remplacez la ligne `$cart_temps = $instant2 - $instant1` par ce fragment de code :

```
if ($instant2 > $instant1) {
    $cart_temps = abs($instant2 - $instant1);
}
```

On obtiendra ainsi une valeur toujours positive ou nulle, quel que soit l'ordre des dates.

Maintenant que nous avons étudié en détail les dates et les temps de PHP, finissons par une courte présentation des dates et des temps en MySQL.

Format des dates MySQL

Tout comme PHP, MySQL 5 utilise des étiquettes temporelles mais, sous leur forme native, celles-ci ne sont pas compatibles avec celles de PHP. MySQL reconnaît trois types de temps/date pour les champs de ses tables : DATE (une date), TIME (une heure) et DATETIME (une date et une heure). Il dispose également d'un type de date spécial, TIMESTAMP, qui fonctionne comme DATETIME sauf que les champs de ce type sont automatiquement initialisés avec l'instant courant à chaque insertion ou mise à jour et qu'il utilise un mécanisme de stockage différent.

Bien qu'il existe plusieurs moyens de représenter les données de ces types dans les requêtes, le plus simple consiste à utiliser une chaîne SQL. Vous pouvez, par exemple, utiliser '2008-09-26' comme une date, '13:23:56' comme une heure et '2008-09-26 13:23:56' comme une date et une heure. Pour convertir un instant PHP stocké dans la variable \$instant sous une forme adaptée à MySQL, utilisez l'appel suivant :

```
date('Y-m-d H:i:s', $instant).
```

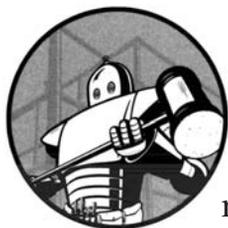
Bien que vous puissiez stocker les étiquettes temporelles PHP/Unix dans des champs de type INT(10), l'utilisation des formats natifs de MySQL est bien plus pratique car vous pourrez ensuite employer ces données indépendamment de PHP. Pour obtenir une étiquette temporelle PHP à partir d'une requête SQL, utilisez la fonction SQL UNIX_TIMESTAMP(), comme dans cet exemple :

```
SELECT UNIX_TIMESTAMP(ma_date) FROM table;
```

MySQL 5 dispose de nombreuses fonctions sur les dates, comme DATE_FORMAT et DATE_ADD. Pour en connaître la liste complète, consultez la documentation en ligne sur <http://www.mysql.com/>.

7

TRAITEMENT DES FICHIERS



Le traitement des fichiers joue un grand rôle dans la programmation en PHP. Pour accomplir votre travail, vous pouvez avoir besoin de créer des fichiers textes à la volée, de lire des fichiers délimités par des tabulations pour importer d'importants volumes de données ou même de créer des fichiers cache pour accélérer votre serveur et réduire les coûts en terme d'utilisation du processeur.

Permissions des fichiers

PHP a besoin de permissions pour pouvoir manipuler les fichiers. Sur la plupart des serveurs web, PHP peut lire assez facilement les fichiers, mais il n'a pas les permissions nécessaires pour en créer ni pour les modifier. C'est une bonne chose car un accès en écriture pour tout le monde donne généralement carte blanche aux pirates pour faire ce qu'ils veulent sur un serveur.

Les serveurs Unix définissent trois jeux de permissions pour le propriétaire, le groupe et tous les autres utilisateurs, qui représentent le monde. L'utilisateur qui possède le fichier est le *propriétaire* et tout autre utilisateur du système est alors

considéré comme faisant partie du *monde* (nous ne présenterons pas les *groupes* dans ce livre ; mettez les mêmes permissions au groupe qu'au monde).

Les serveurs sécurisés traitent PHP comme un utilisateur véritablement non privilégié, qui ne peut écrire nulle part sur le système. On ne souhaite pas que PHP possède des droits sur la machine car les utilisateurs externes influencent au moins en partie son comportement. Si un pirate trouve un moyen de compromettre PHP, il ne faudrait pas que cela se répande à tout le reste du système.

Il y a trois façons d'accéder à un fichier et il y a donc trois types de permissions, lecture, écriture et exécution, qui sont indépendantes : vous pouvez, par exemple, donner les droits de lecture et d'exécution sans pour autant donner le droit d'écriture.

- Le droit de *lecture* autorise PHP à lire un fichier, c'est-à-dire à examiner son contenu. Pour les répertoires, ce droit permet de lire le contenu du répertoire (mais pas nécessairement d'y accéder, comme nous le verrons avec le droit d'*exécution*).
- Le droit d'*écriture* autorise PHP à modifier le contenu du fichier, de supprimer le fichier et, dans le cas d'un répertoire, d'y créer un fichier ou un sous-répertoire.
- Le droit d'*exécution* autorise PHP à exécuter des programmes, ce qui n'est généralement pas conseillé car le serveur peut alors lancer des programmes malicieux. Si votre serveur a été correctement configuré pour le Web, les scripts PHP devraient s'exécuter correctement sans avoir besoin du droit d'exécution. Pour les répertoires, cette permission a une autre signification puisqu'elle autorise l'accès aux fichiers contenus dans un répertoire (en supposant que vous ayez le droit de lecture sur ceux-ci). Pour les répertoires, vous devez donc souvent donner le droit d'exécution en même temps que le droit de lecture.

Par défaut, la plupart des fichiers donnent le droit de *lecture* au propriétaire, au groupe et au monde. En outre, le propriétaire a généralement le droit d'*écriture*. Il en va de même pour les répertoires, sauf qu'ils ont quasiment toujours le droit d'*exécution* positionné en même temps que le droit de *lecture*.

Si vous souhaitez que PHP puisse créer des fichiers (plusieurs scripts de ce livre en ont besoin), vous devez créer un répertoire où PHP sera autorisé à écrire.

La méthode la plus classique pour octroyer des permissions consiste à le faire de façon absolue, en indiquant en une seule fois les droits du propriétaire, du groupe et du monde à l'aide d'une valeur numérique. La valeur pour donner les droits de lecture/écriture au propriétaire, ainsi que le droit de lecture au groupe et au monde, par exemple, est 644 où 6 concerne le propriétaire, le premier 4 le groupe et le deuxième 4 le monde. Chaque chiffre est, en fait, un champ de bits représenté en octal. Les valeurs les plus courantes sont :

0 : aucun droit

4 : droit de lecture

5 : droits de lecture/exécution

6 : droits de lecture/écriture

7 : tous les droits¹

Ces permissions peuvent être modifiées par un programme FTP ou en ligne de commande.

Permissions avec un client FTP

La plupart des clients FTP permettent de définir les permissions pour les fichiers et les répertoires. En général, il suffit de cliquer avec le bouton droit sur le nom d'un répertoire et de rechercher une option nommée *CHMOD*, *Permissions* ou *Propriétés*. Une boîte de dialogue devrait alors apparaître et vous permettre de définir les permissions. Si ce n'est pas le cas, lisez la documentation de votre client.

Pour un répertoire, vous devriez utiliser la valeur 755 qui, sous Unix, correspond à tous les droits pour le propriétaire et aux droits de lecture/écriture pour le groupe et le monde. Pour les fichiers, cette valeur devrait être 644, ce qui correspond aux mêmes droits, moins l'exécution.

La ligne de commande

Si vous avez accès à un shell Unix sur le serveur, vous pouvez donner les mêmes permissions que ci-dessus en utilisant une méthode bien plus directe puisqu'il suffit de taper la commande suivante, qui fonctionne pour tout type de fichier et de répertoire :

```
chmod 755 repertoire
```

Problèmes éventuels

Parmi les nombreux problèmes possibles, votre hébergeur peut faire tourner PHP sous un compte spécifique, différent du vôtre. Dans ce cas, vous devrez peut-être donner le droit d'écriture à tout le monde, ce qui risque d'être interdit par votre hébergeur. Il faudra alors vous contenter des accès en lecture, ce qui ne vous empêchera pas de faire beaucoup de choses.

1. NdR : Vous êtes susceptible de rencontrer une autre notation, composée des caractères "rwx". Retenez que "r" signifie "read" et vaut 4 (droit de lecture), "w" signifie "write" et vaut 2 (droit d'écriture) et enfin "x" signifie "eXecute" et vaut 1 (droit d'exécution). On retrouve les valeurs précédentes : les droits de lecture/exécution valent bien 5 (4+1), les droits de lecture/écriture 6 (4+2) et l'ensemble des droits correspond à 7 (4+2+1). Vous trouverez souvent la notation "-rwx" à travers des clients FTP.

N'oubliez pas que donner le droit d'écriture sur un répertoire vous expose potentiellement à un certain nombre de problèmes de sécurité. Vous ne devriez jamais permettre à PHP d'exécuter des fichiers dans un répertoire où il a le droit d'écrire – en fait, vous devriez interdire ce répertoire au serveur web. Si vous ne faites pas attention à la façon dont vos scripts nomment et accèdent aux fichiers, vous allez au devant de gros ennuis.

Recette 50 : Mettre le contenu d'un fichier dans une variable

Supposons que vous vouliez placer tout le contenu d'un fichier texte dans une variable pour y accéder plus tard. C'est une bonne introduction aux accès fichiers car elle montre toutes les étapes de base. Voici comment recopier le contenu de *fichier.txt* dans la variable `$donnees_fichier` :

```
<?php

$donnees_fichier = '';

$fd = fopen('fichier.txt', 'r');
if (!$fd) {
    echo "Erreur ! Impossible d'ouvrir le fichier.";
    die;
}

while (! feof($fd)) {
    $donnees_fichier .= fgets($fd, 5000);
}
fclose($fd);

?>
```

La fonction `fopen()` est une étape essentielle de la manipulation des fichiers ; elle agit comme une passerelle entre le système et PHP. Lorsque l'on ouvre un fichier, on précise la façon dont on souhaite y accéder : ici, on l'ouvre en lecture, mais on pourrait également l'ouvrir en écriture.

`fopen()` renvoie un identifiant de ressource qui servira aux autres fonctions pour effectuer leurs opérations sur le fichier. Ici, ces fonctions s'appellent `fgets()` et `feof()`.

`fopen()` attend deux paramètres : le chemin d'accès au fichier et le mode d'ouverture. Voici les modes les plus utilisés (n'oubliez pas que chacun d'eux peut échouer si vous n'avez pas les permissions correspondantes) :

- r Ouverture en lecture seule ; la lecture commence au début du fichier.
- w Ouverture en écriture seule (voir la section suivante) ; l'écriture commence au début du fichier et écrase donc le contenu de celui-ci. Si le fichier n'existe pas, `fopen()` tente de le créer.

- x Création et ouverture en écriture seule ; l'écriture commence au début du fichier. Si le fichier existe déjà, l'appel renvoie `false`. Ce mode n'est disponible qu'à partir de PHP 4.3.2.
- a Ouverture en écriture seule ; l'écriture commence à la fin du fichier. Si le fichier n'existe pas, `fopen()` tente de le créer.

Les modes suivants ouvrent le fichier à la fois en lecture et en écriture. Ne les utilisez que si vous savez vraiment ce que vous faites :

- w+ Ouverture en lecture et en écriture ; l'accès commence au début du fichier et supprime son contenu éventuel. Si le fichier n'existe pas, `fopen()` tente de le créer.
- r+ Ouverture en lecture et en écriture ; l'accès commence au début du fichier.
- a+ Ouverture en lecture et en écriture ; l'accès commence à la fin du fichier. Si le fichier n'existe pas, `fopen()` tente de le créer.
- x+ Création et ouverture en lecture et en écriture ; l'accès commence au début du fichier. Si le fichier existe déjà, l'appel renvoie `false`. Ce mode n'est disponible qu'à partir de PHP 4.3.2.

Si l'on revient au script, la ligne `$fd = fopen('fichier.txt', 'r')` signifie donc *Ouvre le fichier en lecture seule et affecte l'identifiant de ressource à \$fd*. Pour savoir si l'ouverture s'est bien passée, il suffit de tester que `$fd` a une valeur.

Nous sommes maintenant prêt à effectuer le véritable traitement dans une boucle. La fonction `feof()` indiquant si l'on a atteint la fin du fichier, on l'utilise comme condition de sortie de la boucle. L'appel à `fgets()` récupère la ligne suivante du fichier jusqu'à 5 000 octets à la fois. Ces données sont ajoutées à la fin de `$donnees_fichier`. Lorsque la lecture est finie, on appelle `fclose($fd)` pour libérer les ressources du système et lui indiquer qu'on a terminé d'accéder au fichier.

Amélioration du script

De nombreux scripts traitent les fichiers ligne par ligne au lieu de les stocker dans une seule variable énorme. Cela arrive assez souvent, notamment lorsque l'on examine les listes d'éléments produits par d'autres programmes (la recette n°55 : "Lire un fichier CSV", montre un exemple où l'on a besoin de lire ligne par ligne le contenu d'un fichier). Pour cela, il suffit de modifier le code à l'intérieur de la boucle, comme dans cet exemple qui affiche toutes les lignes qui contiennent *chapitre*.

```
while (! feof($fd)) {
    $donnees_fichier = fgets($fd, 5000);
    if (strstr($donnees_fichier, 'chapitre') !== FALSE) {
        print $donnees_fichier;
    }
}
```

Vous remarquerez que l'opérateur de concaténation `.` du script initial a été remplacé par l'opérateur d'affectation. Cette modification subtile est très importante.

Selon la configuration de votre serveur, `fopen()` peut lire des données à partir d'une URL avec un appel comme celui-ci :

```
$fr = fopen('http://www.yahoo.fr', 'r');
```

Cependant, si vous utilisez votre propre serveur, vous devez être très prudent lorsque vous autorisez `fopen()` à accéder des fichiers situés à l'extérieur de votre site : certains vers PHP ont utilisé cette fonctionnalité à leur profit. Vous devez notamment faire très attention aux noms de fichiers –; assurez-vous que l'utilisateur n'ait pas son mot à dire sur les noms des fichiers ouverts par `fopen()` ! Pour désactiver cette fonctionnalité, initialisez l'option `allow_url_fopen` à `false`, comme on l'a vu à la section "Options de configuration et le fichier *php.ini*". Pour disposer de fonctionnalités plus puissantes, utilisez plutôt `cURL` pour accéder aux sites web, comme expliqué au Chapitre 11.

Problèmes éventuels

L'erreur la plus classique est due au fait que PHP n'a pas les permissions de lire le fichier que vous tentez d'ouvrir. Certains fichiers ne devraient pas pouvoir être lus par PHP (ceux qui contiennent des mots de passe, par exemple) et vous pouvez recevoir un message d'erreur si vous tentez d'ouvrir l'un d'eux. En ce cas, vérifiez les permissions comme on l'a expliqué dans la section "Permissions des fichiers".

Cela nous conduit à un problème plus important : n'autorisez jamais, en aucun cas, un utilisateur à ouvrir un fichier avant de vérifier qu'il a de bonnes raisons de le faire. N'oubliez pas que vous ne pouvez pas avoir confiance dans ce qu'un utilisateur vous envoie. Si les noms de fichiers reposent de trop près sur les données fournies par l'utilisateur, celui-ci peut très bien parvenir à accéder à n'importe quel fichier de votre site. Vous devez donc appliquer des règles qui restreignent les accès aux répertoires des fichiers sur votre serveur.

Vous devriez également vérifier les données contenues dans les fichiers que déposent les utilisateurs. La recette n°54, "Déposer des images dans un répertoire", montre comment vérifier le type, l'emplacement et la taille d'un fichier déposé et bien d'autres choses encore.

Recette 51 : Écrire dans un fichier

Voici comment écrire une chaîne dans un fichier :

```
<?  
$donnees_fichier = "Bonjour fichier.\nDeuxième ligne."  
$fd = fopen('fichier.txt', 'w');
```

```
if (!$fd) {
    echo "Erreur ! Impossible d'ouvrir/créer le fichier.";
    die;
}
fwrite($fd, $donnees_fichier);
fclose($fd);
?>
```

Vous remarquerez que la chaîne contient un retour à la ligne explicite. Affiché sur une machine Unix, le contenu du fichier aura donc cet aspect :

```
Bonjour fichier.
Deuxième ligne.
```

Le séparateur entre les deux lignes est un retour à la ligne. Sur Unix et Mac, il s'agit d'un simple caractère représenté en PHP par `\n` entre des apostrophes doubles.

Pendant, avec Windows, ce retour à la ligne est représenté par la séquence `\r\n` ("retour chariot, puis nouvelle ligne").

Par conséquent, vous devez faire un peu attention si vous vous souciez de la portabilité des fichiers textes ; sinon, un fichier Unix apparaîtra sur Windows comme une seule longue ligne et, inversement, Unix verra un retour chariot à la fin de chaque ligne d'un fichier Windows.

Si vous le souhaitez, vous pouvez explicitement décider que le retour à la ligne sera `\r\n` mais, pour des raisons de portabilité, ajoutez plutôt un `t` à la fin du mode fourni à `fopen()`. Avec le mode `wt`, les caractères `\n` seront automatiquement traduits en séquences `\r\n` sous Windows.

Recette 52 : Tester l'existence d'un fichier

Si vous tentez d'ouvrir un fichier qui n'existe pas, `fopen()` produit des messages d'erreur, tout comme `unlink()` lorsque vous essayez de supprimer un fichier inexistant. Pour tester l'existence d'un fichier avant d'effectuer des opérations sur celui-ci, utilisez la fonction `file_exists()` :

```
<?
if (file_exists('fichier.txt')) {
    print 'OK, fichier.txt existe.';
}
```

Cette fonction renvoie `true` si le fichier existe, `false` sinon.

Recette 53 : Supprimer des fichiers

Pour supprimer un fichier sous Unix, utilisez la fonction `unlink()` :

```
<?
if (unlink("fichier.txt")) {
    echo "fichier.txt supprimé.";
} else {
    echo "fichier.txt : échec de la suppression.";
}
?>
```

Vous devez, bien sûr, avoir les permissions adéquates pour supprimer un fichier. Cependant, le plus grand danger, ici, est que vous pouvez supprimer un fichier par inadvertance : la suppression d'un fichier sous Unix est définitive car il n'y a ni poubelle ni commande *undelete*. La seule façon de récupérer des données consiste à utiliser les sauvegardes de l'administrateur.

Si vous comptez autoriser les utilisateurs à supprimer des fichiers, vous devez donc être très prudent.

Recette 54 : Déposer des images dans un répertoire

Déposer périodiquement 5 ou 10 photos par semaine sur un serveur web est une opération pénible. Lorsque j'ai mis à jour mon serveur, j'ai dû sauvegarder les photos sur mon disque dur, lancer un client FTP, les mettre sur le site, puis diffuser l'URL à tous ceux qui étaient intéressés. Il existe de nombreux programmes de galeries photos qui effectuent ces tâches, mais la plupart sont très compliqués et je voulais faire quelque chose d'un peu plus simple.

Comme tout bon programmeur fainéant, j'ai donc décidé d'automatiser ce processus pour disposer d'un système assez complet permettant aux utilisateurs de déposer des images (et uniquement des images) dans un répertoire. Je voulais également bloquer les images trop grosses et produire un code HTML complet, avec des attributs `height` et `width`. La première partie du système est un formulaire nommé *depot.html*, qui permet de saisir les informations sur les images :

```
<table border="0" cellpadding="10">
<form action="traite_image.php" enctype="multipart/form-data"
      method="post">
<tr>
  <td valign=top><strong>Fichier image :</strong></td>
  <td><input name="fichier" type="file"><br>
    Les fichiers images doivent être aux formats JPEG, GIF, ou PNG.
  </td>
</tr>
<tr>
  <td valign="top"><strong>Répertoire cible :</strong></td>
```

```

<td>
  <select name="emplacement">
    <option value="articles" selected>Images d'articles</option>
    <option value="bannieres">Bannières/Pubs</option>
  </select>
</td>
</tr>
<tr>
  <td valign="top"><strong>Nom du fichier (Facultatif) :</strong></td>
  <td><input name="nouv_nom" type="text" size="64" maxlength="64"></td>
</tr>
<tr>
  <td colspan="2">
    <div align="center"><input type="submit" value="Déposer"></div>
  </td>
</tr>
</form>
</table>

```

Le script qui traite ce formulaire s'appelle *traite_image.php*. Ses premières lignes consistent à initialiser quelques variables de configuration :

```

<?php
/* Configuration */
$racine = "/home/www/wcphp/images"; /* Répertoire racine des images */
$racine_url = "http://www.exemple.com/images"; /* Racine de l'URL */
$largeur_max = 420; /* Largeur max d'une image */
$hauteur_max = 600; /* Hauteur max d'une image */
$ecrase_images = false; /* Autorise l'écrasement */
/* Sous-répertoires autorisés */
$rep_cibles = array("articles", "bannieres");
/* Fin de la configuration */

```

La variable `$racine` doit contenir le répertoire sous lequel vous voulez placer les images, tandis `$racine_url` contient le nom qui sera utilisé pour les visualiser à partir d'un navigateur. Les éléments de `$rep_cibles` sont des sous-répertoires de `$racine` : ce sont les seuls emplacements où les utilisateurs seront autorisés à déposer leurs images.

Après cette configuration, vérifions d'abord que le script fonctionne – tout dépend de la disponibilité de la fonction `getimagesize()` :

```

if (!function_exists(getimagesize)) {
  die("La fonction getimagesize() est requise.");
}

```

L'extraction des informations à partir du formulaire suit la procédure classique :

```
/* Récupère les informations du dépôt. */
$emplacement = strval($_POST['emplacement']);
$nouv_nom = strval($_POST['nouv_nom']);
$fichier = $_FILES['fichier']['tmp_name'];
$nom_fichier = $_FILES['fichier']['name'];
```

Nous devons déterminer le nom que nous voulons donner au fichier sur le serveur. Ce faisant, nous voulons nous assurer que le nom ne comporte pas de caractères bizarres et qu'il sera impossible de quitter le répertoire cible en utilisant des barres de fraction (/). Une expression régulière permet de remplacer en une seule étape tous les caractères non admis. Après ce traitement, \$nouv_nom contient le nouveau nom du fichier déposé :

```
/* Supprime les caractères non admis dans le nom cible */
if ($nouv_nom) {
    $nouv_nom = preg_replace('/[^A-Za-z0-9_-]/', '', $nouv_nom);
} else {
    $nouv_nom = preg_replace('/[^A-Za-z0-9_-]/', '', $nom_fichier);
}
```

L'étape suivante consiste à valider les paramètres. On commence par vérifier que le répertoire cible fait partie de la liste des répertoires autorisés :

```
/* Validation des paramètres. */
if (!in_array($emplacement, $rep_cibles)) {
    /* Emplacement incorrect */
    die("Répertoire cible non autorisé.");
} else {
    $racine_url .= "/" . $emplacement;
}
```

Voici une vérification qui s'assure que l'on a bien indiqué le nom du fichier à déposer :

```
if (!$fichier) {
    /* Aucun fichier */
    die("Aucun fichier à déposer.");
}
```

Il est temps maintenant de valider les données elles-mêmes. Pour cela, on initialise les types de fichiers autorisés, puis on appelle `getimagesize()` pour obtenir

les dimensions de l'image déposée et son type (\$attr sera utilisée plus tard dans le script).

```
/* Vérification du type du fichier. */
$types_fichiers = array(
    "image/jpeg" => "jpg",
    "image/pjpeg" => "jpg",
    "image/gif" => "gif",
    "image/png" => "png",
);
$largeur = null;
$hauteur = null;

/* Extrait le type MIME et la taille de l'image. */
$infos_image = getimagesize($fichier);
$type_fichier = $infos_image["mime"];
list($largeur, $hauteur, $t, $attr) = $infos_image;
```

À partir des paramètres que l'on vient d'extraire, nous nous assurons que l'image est dans un format autorisé, nous en déduisons le suffixe du nom du fichier et nous vérifions qu'il n'est pas trop gros :

```
/* Vérification du type. */
if (!$types_fichiers[$type_fichier]) {
    die("L'image doit être au format JPEG, GIF ou PNG.");
} else {
    $suffixe_fichier = $types_fichiers[$type_fichier];
}

/* Vérification de la taille. */
if ($largeur > $largeur_max || $hauteur > $hauteur_max) {
    die("$largeur x $hauteur excède $largeur_max x $hauteur_max.");
}
```

De temps en temps, quelqu'un dépose un fichier avec un suffixe incorrect (non reconnu ou mal orthographié). Ce n'est pas un problème crucial mais cela perturbe le type MIME lorsque le serveur l'envoie. Comme on connaît le suffixe correct, nous pouvons l'utiliser pour corriger un éventuel suffixe incorrect : *truc.jpog* pourrait ainsi devenir *truc.jpog.jpg*.

Après avoir trouvé le nom final pour le fichier déposé, nous stockons son chemin complet dans \$nouv_chemin :

```
/* Force le suffixe du fichier. */
$nouv_nom = preg_replace('/\.(jpe?g|gif|png)$/i', "");
$nouv_nom .= $suffixe_fichier;
$nouv_chemin = "$racine/$emplacement/$nouv_nom";
```

Maintenant que nous avons le nom final, nous pouvons vérifier qu'il n'existe pas déjà un fichier de ce nom si l'on n'a pas activé l'écrasement des fichiers :

```
if ((!$ecrase_images) && file_exists($nouveau_chemin)) {
    die("Le fichier existe déjà ; il ne sera pas écrasé.");
}
```

On peut alors copier le fichier vers sa destination finale et s'assurer que cette copie a bien fonctionné :

```
/* Copie le fichier vers son emplacement final. */
if (!copy($fichier, $nouv_chemin)) {
    die("Échec de la copie.");
}
```

Si l'on est arrivé ici, c'est que le dépôt s'est bien passé et il nous reste simplement à produire un peu de HTML pour fournir à l'utilisateur un lien vers ce fichier :

```
$url_image = "$racine_url/$nouv_nom";

/* Affiche l'état. */
print "HTML pour l'image :</strong><br>
      <textarea cols=\"80\" rows=\"4\">";
print "<img src=\"$url_image\" $attr alt=\"$nom_fichier\"
      border=\"0\"/>";
print "</textarea><br>";
print '<a href="depot.html">Déposer une autre image ?</a>';?>
```

Utilisation du script

Ce script exige que PHP ait la permission d'écrire dans tous les répertoires cibles de \$rep_cibles. En outre, il a besoin du module GD, une extension PHP qui permet d'analyser et de créer des fichiers images. La plupart des serveurs l'installent par défaut ; si ce n'est pas le cas du vôtre, reportez-vous à la recette n°28, "Ajouter des extensions à PHP".

Il vous reste seulement à faire pointer votre navigateur vers *depot.html* et PHP fera le reste.

Problèmes éventuels

Mis à part les classiques problèmes de permissions, le plus gros problème est la sécurité. Tel qu'il est écrit, tout idiot qui a accès à l'URL du script peut déposer autant d'images qu'il le souhaite et éventuellement écraser les vôtres. Si le script n'est pas derrière un pare-feu, vous pouvez lui ajouter une fonction de connexion

pour empêcher les accès non autorisés, comme on l'explique dans la recette n°63, "Système de connexion simple".

Amélioration du script

Vous pouvez imposer une limite à la taille d'un fichier en utilisant la variable `$_FILES['fichier']['size']`. Il suffit alors d'écrire un test comme celui-ci :

```
if ($_FILES['fichier']['size'] > $taille_max) {  
    $erreur_fatale = "La taille de ce fichier dépasse $taille_max octets."  
}
```

NOTE *Étudiez également la recette n°13, "Empêcher les utilisateurs de déposer de gros fichiers", car elle explique comment imposer une limite globale sur la taille des fichiers déposés.*

Recette 55 : Lire un fichier CSV

Une tâche de programmation classique consiste à transformer des données provenant de feuilles de calcul Excel pour les mettre sous un format comme HTML ou comme des lignes d'une table MySQL. Si l'on devait tous travailler avec le format Excel (XLS), ce serait un véritable problème. Heureusement, Excel et OpenOffice.org permettent d'exporter un fichier Excel au format CSV (*Comma-Separated Value*), dans lequel les rangées de données sont organisées en lignes où chaque champ est séparé du suivant par une virgule. Voici un exemple :

```
"Aéroport", "Ville", "Activité"  
"LON", "Londres", "Musées"  
"PAR", "Paris", "Restaurants"  
"SLC", "Salt Lake City", "Ski"
```

Bien que vous pourriez penser qu'il ne s'agit que de lire les lignes et de les diviser en utilisant une virgule comme délimiteur, vous devez également vous occuper des apostrophes, des anti-slash et d'autres détails mineurs de ce format. Cependant, la plupart des langages disposent de fonctionnalités permettant de gérer ceci et PHP n'y fait pas exception. Grâce à la fonction prédéfinie `fgetcsv()`, le traitement des fichiers CSV est très simple. `fgetcsv()` fonctionne exactement comme `fgets()`, mis à part qu'elle renvoie un tableau contenant les valeurs de la ligne courante au lieu de renvoyer une chaîne. Voici un script très simple qui traite un fichier CSV déposé *via* le champ `fic_csv` d'un formulaire :

```
<table>  
<tr>  
    <th>Champ 1</th>  
    <th>Champ 2</th>
```

```
<th>Champ 3</th>
</tr>

<?php
$fn = $_FILES["fic_csv"]["tmp_name"];
$fd = fopen($fn, "r");
while (!feof($fd)) {
    $champs = fgetcsv($fd);
    print "<tr>";
    print "<td>$champs[0]</td><td>$champs[1]</td><td>$champs[2]</td>";
    print "</tr>";
}
fclose($fd);
?>
</table>
```

Comme vous pouvez le constater, ce script affiche les trois premières colonnes du fichier CSV sous la forme d'un tableau HTML.

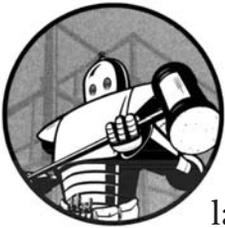
Parfois, le fichier utilise des tabulations à la place des virgules pour délimiter les champs. Pour lire ce format, il suffit de remplacer l'appel précédent à `fgetcsv()` par une ligne comme celle-ci :

```
$champs = fgetcsv($fd, 0, "\t");
```

Le troisième paramètre indique le délimiteur ; assurez-vous d'utiliser des apostrophes doubles autour de `\t` pour que cette séquence soit correctement interprétée. Le second paramètre est la longueur maximale de la ligne du fichier CSV, 0 indique que l'on n'impose pas de limite à cette longueur.

8

GESTION DES UTILISATEURS ET DES SESSIONS



Le concept initial du World Wide Web allait un peu plus loin qu'une simple suite de pages et de médias statiques : il avait pour but de faciliter la publication des pages et la navigation entre elles, mais le Web n'est devenu réellement utile que lorsque les sites ont commencé à offrir du contenu dynamique.

La plupart de ces contenus sont spécifiques à une session – un panier virtuel, par exemple, est lié à une session : ses informations disparaissent lorsqu'on ferme le navigateur ou qu'un autre utilisateur se connecte.

Les outils et les techniques pour suivre les sessions web ont été conçus après coup et sont des solutions *ad hoc* – rien qui ne ressemble à un travail raisonnable. Parfois tout se brouille et vous devez faire attention à la sécurité (comme toujours), mais PHP peut vous aider à résoudre les difficultés.

Suivi des données des utilisateurs avec des cookies et des sessions

Pour savoir ce que fait un utilisateur précis sur votre site, vous devez stocker des informations sur cet utilisateur, comme son nom, son mot de passe, le temps

écoulé entre ses visites, ses préférences, etc. En programmation web, on utilise pour cela deux techniques connues sous les noms de cookies et de sessions.

Les cookies

Les cookies sont des fragments de données stockés sur la machine de l'utilisateur. Lorsque celui-ci accède à plusieurs pages de votre site, son navigateur renvoie tous les cookies valides à votre serveur lors de ces accès. Le stockage de données sur l'ordinateur d'un utilisateur sans son consentement étant un risque potentiel pour sa sécurité, vous n'avez pas de contrôle direct sur la façon dont ce stockage est effectué. Si vous respectez certaines règles que le navigateur (et l'utilisateur) ont mises en place, le navigateur acceptera vos cookies et les renverra dans les circonstances appropriées. Un cookie qui a été accepté est dit configuré.

Avantages

- Les cookies peuvent stocker des informations pendant des années.
- Les cookies fonctionnent bien avec les serveurs distribués dont la charge est équilibrée (cas des sites à fort trafic) puisque toutes les données sont sur la machine de l'utilisateur.

Inconvénients

- Vous devez soigneusement suivre les règles ; sinon, de nombreux navigateurs n'accepteront pas vos cookies et ne vous informeront pas de ce refus.
- La taille des cookies est limitée (il est généralement préférable de ne pas dépasser 512 Ko).
- S'ils le souhaitent, les utilisateurs peuvent aisément supprimer les cookies.
- Les cookies sont généralement spécifiques à un utilisateur sur un ordinateur. Si cet utilisateur passe sur une autre machine, il n'utilisera pas les anciens cookies.

Les sessions

Les sessions sont formées d'un identifiant de session unique et d'un mécanisme de stockage spécial sur votre serveur. Les données étant sur le serveur, l'utilisateur n'a aucun moyen de les manipuler.

Avantages

- Bien que la plupart des sessions fonctionnent en plaçant leur identifiant dans un cookie, PHP et les autres systèmes de programmation web savent créer des sessions sans utiliser de cookies. Cette méthode fonctionne donc quasiment toujours.
- Vous pouvez stocker autant d'informations de session que vous le souhaitez.

- Les utilisateurs ne peuvent généralement ni lire ni modifier les données de session ; dans le cas contraire, vous avez le contrôle sur ce processus de modification.

Inconvénients

- Les sessions sont généralement spécifiques à une fenêtre de navigateur ou à un seul processus navigateur. Lorsque vous fermez cette fenêtre, à moins d'avoir stocké l'identifiant de session dans un cookie persistant, l'utilisateur ne peut plus récupérer les anciennes valeurs. En utilisant un système de connexion, vous pouvez cependant associer un identifiant de session à un nom d'utilisateur.
- Avec une installation de PHP de base, les sessions sont spécifiques à un serveur. Si vous avez un serveur pour les ventes et un autre pour le contenu, le premier ne verra aucune donnée de session du deuxième. Vous pouvez cependant adapter le système de stockage des sessions pour résoudre ce problème.
- L'identifiant de session étant une donnée lue et envoyée par l'utilisateur, des espions peuvent accéder aux sessions si vous ne prenez pas quelques précautions.

En termes abstraits, vous pouvez considérer les sessions comme des cookies améliorés. Bien que certaines données (comme l'identifiant de session) soient encore stockées sur la machine de l'utilisateur, les véritables données sont toujours sur le serveur. Il existe de nombreuses implémentations possibles des sessions, mais l'idée de base est toujours la même.

N'oubliez pas, cependant, que si vous ne voulez pas que les utilisateurs puissent lire ou modifier certaines données, vous devez les placer dans une session, pas dans un cookie. Vous devez vérifier les cookies exactement de la même façon que les données provenant des formulaires car elles viennent du client et sont donc facile à falsifier.

Recette 56 : Créer un message "Heureux de vous revoir NomUtilisateur !" avec les cookies

Une astuce à peu de frais qu'utilisent de nombreux sites consiste à afficher un message "Heureux de vous revoir" aux utilisateurs qui reviennent sur le site. Si vous donnez votre nom au site, il peut l'utiliser pour vous souhaiter la bienvenue.

Pour illustrer un moyen de le faire, voici un script qui stocke les informations sur l'utilisateur dans un cookie et affiche ce cookie s'il est disponible :

```
<?php
if (isset($_REQUEST["nom_utilisateur"])) {
    setcookie("nom_utilisateur_stocke",
        $_REQUEST["nom_utilisateur"], time() + 604800, "/");
    $_COOKIE["nom_utilisateur_stocke"] = $_REQUEST["nom_utilisateur"];
}
```

```

}
if (isset($_COOKIE["nom_utilisateur_stocke"])) {
    $utilisateur = $_COOKIE["nom_utilisateur_stocke"];
    print "Heureux de vous revoir <b>$utilisateur</b> !";
} else {
?>
    <form method="post">
    Nom Utilisateur : <input type="text" name="nom_utilisateur" />
    </form>
    <?php
}??>

```

Avec PHP, l'accès et le stockage des cookies impliquent deux mécanismes différents. Le tableau `$_COOKIE` contient les cookies que vous envoie le client ; il fonctionne comme les tableaux `$_POST` et `$_GET`.

Pour mettre en place des cookies, utilisez la fonction `setcookie()` en lui passant trois paramètres :

- **le nom du cookie**, `nom_utilisateur_stocke` dans le script ;
- **la valeur du cookie** ;
- **la date d'expiration du cookie sur la machine de l'utilisateur, exprimée sous la forme d'une étiquette temporelle Unix**. Le script utilise `time() + 604800`, soit sept jours à partir de la date courante (voir le Chapitre 6 pour plus de détails).

Vous pouvez également lui passer trois autres paramètres facultatifs :

- **Un chemin pour limiter les emplacements de votre site pour lesquels le cookie est valide**. Cela fonctionne comme un répertoire. Si vous voulez, par exemple, que le cookie ne soit valide que pour ce qui est placé sous `/contenu/` sur votre site, utilisez `/contenu/` pour ce paramètre. Si vous voulez qu'il soit valide partout, préférez `/`.
- **Un domaine de validité du cookie**. Si vos hôtes s'appellent `www.exemple.com` et `ventes.exemple.com` et que vous souhaitez que le cookie soit valide pour les deux, utilisez `.exemple.com` comme paramètre de domaine. Si vous n'avez qu'un seul serveur web dans votre domaine, ce paramètre ne vous concerne pas.
- **Un indicateur de connexion sécurisée**. Si ce paramètre vaut 1, le navigateur ne devra envoyer le cookie que si la connexion est sécurisée.

Problèmes éventuels

Plusieurs problèmes peuvent survenir lors de la mise en place et la récupération des cookies :

Vous avez envoyé des données au navigateur de l'utilisateur avant d'appeler `setcookie()`.

Les informations sur les demandes de cookie se trouvent dans l'en-tête HTTP d'une réponse du serveur. Vous ne pouvez donc pas envoyer au

client des données faisant partie du document avant d'appeler `setcookie()` ; en d'autres termes, vous ne devez pas afficher quoi que ce soit, ni appeler une opération qui provoquerait un affichage. Si les avertissements sont activés, PHP vous indiquera cette erreur.

Ce problème est souvent dû à une gestion laxiste des espaces dans vos fichiers PHP. Si des lignes blanches ou des espaces précèdent la balise `<?>` qui débute la section de code PHP, ce problème surviendra forcément. Cette remarque s'applique également aux fichiers inclus avant l'appel à `setcookie()` ; ces fichiers ne doivent pas non plus avoir d'espace après la balise fermante `?>`.

Le navigateur de l'utilisateur a rejeté le cookie.

Si le navigateur n'accepte pas le cookie, vous n'aurez aucun retour. Pour vérifier la présence d'un cookie, appelez la fonction `isset()` pour le rechercher dans le tableau `$_COOKIE`. La cause la plus classique de rejet d'un cookie est un domaine incorrect (les navigateurs n'acceptent généralement pas les cookies pour les domaines qui ne correspondent pas à celui du serveur qui fait la demande).

Quelqu'un a mis un mauvais paramètre.

Les cookies, comme tout ce qu'envoie un client, peuvent aisément être fabriqués de toute pièce. Ne leur faites pas confiance et vérifiez leurs valeurs comme celles de n'importe quelle donnée provenant d'un formulaire.

Vous tentez de stocker un tableau dans une variable cookie.

Ce n'est pas possible, mais vous pouvez stocker un tableau sérialisé (voir la recette n°5 : "Transformer un tableau en variable scalaire qui pourra être restaurée ultérieurement").

Recette 57 : Utiliser les sessions pour stocker temporairement des données

Les interfaces graphiques traditionnelles nécessitent que l'utilisateur saisisse un certain nombre d'informations réparties sur plusieurs formulaires. Vous pouvez également avoir à stocker un ensemble de données tant que le navigateur de l'utilisateur est ouvert (pour un panier virtuel, par exemple). Bien qu'il soit techniquement possible d'utiliser pour cela des champs cachés, ce n'est pas souhaitable dans la plupart des cas du fait de la complexité de mise en œuvre et les problèmes de gestion des états du navigateur.

En revanche, vous pouvez utiliser le système intégré de gestion des sessions de PHP pour stocker et accéder aux données pour une session de navigateur donnée. Les sessions PHP s'occupent quasiment de tout le travail de mise en place des cookies (ou d'un identifiant de session) et du stockage des données sur votre serveur. La seule chose qui vous reste à faire est de lancer le gestionnaire de

sessions dans vos scripts à l'aide de la fonction `session_start()`, puis d'accéder aux données *via* le tableau `$_SESSION`.

Voici un formulaire qui utilise les sessions pour capturer, résumer et modifier les données. Commençons par un script de formulaire simple. La première ligne lance la session et les deux lignes suivantes extraient les données de session existantes :

```
<?
session_start();

$nom = $_SESSION["nom"];
$couleur = $_SESSION["couleur"];
```

Pour les nouveaux visiteurs, ces deux variables ne seront pas initialisées mais, s'il reviennent à ce formulaire à partir de n'importe où, cette partie du script capturera les anciennes valeurs. Pour afficher le formulaire, on utilise ces anciennes valeurs comme valeurs par défaut :

```
print '<form action="vue_session.php" method="post">';
print 'Comment vous appelez-vous ? ';
print '<input name="nom" type="text" value="' . $nom . '" /><br/>';
print 'Quelle est votre couleur préférée ? ';
print '<input name="couleur" type="text" value="' . $couleur . '" /><br/>';
print '<input type="submit" />';
print '<input type="submit" name="raz" value="Réinitialisation" />';
?>
```

Vous remarquerez qu'on a ajouté un bouton de validation supplémentaire afin de remettre à zéro les valeurs de la session (nous verrons plus loin comment faire). Voici maintenant le script *vue_session.php* qui stocke les données du formulaire et autorise l'utilisateur à revenir en arrière pour modifier les valeurs qu'il a saisi. On lance d'abord la session, puis on vérifie que le nom et/ou la couleur ont été envoyés comme données de formulaire, auquel cas on place également ces valeurs dans des variables de session :

```
<?
session_start();

if ($_REQUEST["nom"]) {
    $_SESSION["nom"] = $_REQUEST["nom"];
}
if ($_REQUEST["couleur"]) {
    $_SESSION["couleur"] = $_REQUEST["couleur"];
}
```

La partie suivante consiste à tester si l'utilisateur a cliqué sur le bouton *Réinitialisation* du formulaire. Si c'est le cas, on utilise la fonction `unset()` pour supprimer les valeurs de session :

```
if ($_REQUEST["raz"]) {
    unset($_SESSION["nom"]);
    unset($_SESSION["couleur"]);
}
```

Nous savons maintenant que le tableau `$_SESSION` contient toutes les valeurs correctes ; nous pouvons alors les utiliser pour afficher les informations à destination de l'utilisateur :

```
$nom = $_SESSION["nom"];
$couleur = $_SESSION["couleur"];

if ($nom) {
    print "Vous vous appelez <b>$nom</b>.<br />";
}
if ($couleur) {
    print "Votre couleur préférée est le <b>$couleur</b>.<br />";
}
```

Enfin, on autorise l'utilisateur à revenir en arrière pour modifier ou supprimer les valeurs. Tout ayant déjà été fait dans les deux scripts précédents, il suffit de placer des liens vers les bons endroits :

```
print '<a href="formulaire_session.php">Modifier les détails</a>';
print ' | <a href="vue_session.php?clear=1">Réinitialiser</a>';
?>
```

Bien que ce soit un exemple très simple, il permet de montrer que les sessions nous facilitent beaucoup la vie lorsque l'on a besoin de créer des formulaires en plusieurs parties ou de suivre à la trace d'autres informations.

Problèmes éventuels

Tout comme pour les cookies, vous devez appeler `session_start()` au début du script, avant d'envoyer la moindre donnée, afin que le serveur puisse configurer l'identifiant de session sur le client. Dans le cas contraire, le client n'acceptera pas cet identifiant et le serveur ne pourra pas associer le client à une session.

Recette 58 : Vérifier qu'un navigateur accepte les cookies

Pour savoir si un navigateur accepte les cookies, vous devez effectuer une vérification en deux étapes, dans deux requêtes web différentes. Le navigateur du client doit faire deux requêtes car il ne configure un cookie que lorsqu'il obtient une réponse à la première requête.

Ces deux requêtes peuvent être envoyées par le même script, mais vous devez faire attention à ne pas placer le navigateur dans une boucle infinie. Le principe consiste à vérifier la présence du cookie et, s'il n'existe pas, essayer de le créer et recharger la page. Cependant, pour ne recharger la page qu'une seule fois, vous devez indiquer au script qu'il effectue un rechargement, afin de l'empêcher de recharger une nouvelle fois la page si le cookie n'existe pas.

C'est donc un petit script, mais vous devez bien l'écrire pour ne pas créer une boucle de rechargements infinis. Le premier traitement consiste à vérifier si le cookie testé existe déjà. En ce cas, puisque le navigateur reconnaît les cookies, il n'y a plus rien à faire :

```
<?php
if (isset($_COOKIE["test"])) {
    print "Cookies activés.";
```

Si ce cookie n'existe pas, cela peut être dû à deux raisons : la première est que le navigateur n'accepte peut-être pas les cookies. Cependant, nous ne pouvons pas en être sûr avant de recharger la page et nous devons savoir que le navigateur a rechargé la page. Pour cela, au deuxième accès on initialise un paramètre GET appelé `testing`. Si ce paramètre existe mais pas le cookie, on sait que le navigateur n'envoie pas de cookie :

```
} else {
    if (isset($_REQUEST["testing"])) {
        print "Cookies désactivés.";
```

Si ni le cookie ni le paramètre `testing` n'existent, c'est parce que c'est la première fois que le navigateur accède à la page ; nous initialisons alors le cookie puis nous rechargeons la page en initialisant `testing` pour signaler qu'il s'agit du second accès. Le fonctionnement de l'en-tête `Location` sera décrit dans la section suivante.

```
} else {
    setcookie("test", "1", 0, "/");
    header("Location: $_SERVER[PHP_SELF]?testing=1");
}
?>
```

La présence ici du paramètre `testing` est fondamentale : si vous l'oubliez et que le navigateur ne reconnaît pas les cookies, la page se rechargera indéfiniment.

Ce script n'est pas très facile à lire car les premières lignes de code ne correspondent pas à ce que voit l'utilisateur au début. Cependant, il est si court que vous pouvez aisément le comprendre dans sa totalité.

Recette 59 : Rediriger les utilisateurs vers des pages différentes

Rediriger les utilisateurs vers de nouvelles pages fait partie intégrante de la programmation des sites web dynamiques. La raison essentielle est qu'il faut souvent rediriger un utilisateur après avoir modifié l'état d'une session. Lorsque, par exemple, on ajoute un article à un panier virtuel, on renvoie l'utilisateur vers une page décrivant l'état courant du panier sans pour autant ajouter le même article au panier si l'on recharge cette page. Pour ce faire, la plupart des sites utilisent un script qui traite le changement d'état puis redirige les utilisateurs vers la page qu'ils souhaitent voir.

Il y a deux moyens d'y parvenir. Le premier, et le plus apprécié, consiste à utiliser l'en-tête HTTP `Location` :

```
<?
header("Location: nouvelle_page.php");
?>
```

La fonction `header()` permet d'envoyer des en-têtes HTTP bruts au navigateur de l'utilisateur. Il faut donc l'utiliser avant d'envoyer quoi que ce soit au navigateur, comme pour les cookies.

Cette méthode a cependant deux inconvénients. Le premier est qu'elle est instantanée et que le script intermédiaire n'apparaîtra pas dans l'historique du navigateur. Le second est qu'elle repose sur HTTP et ne nécessite donc pas un navigateur pour être traitée ; les aspirateurs web comme `wget` sauront donc la gérer.

Si vous voulez montrer une page intermédiaire à l'utilisateur avec un certain délai, vous devez donc utiliser une autre méthode, qui utilise la balise HTML `<meta>`.

Cette méthode est assez simple : pour envoyer l'utilisateur sur une autre page après avoir affiché la page courante pendant cinq secondes, par exemple, il suffit de placer cette ligne dans l'en-tête de la page HTML :

```
<meta http-equiv="Refresh" content="5;URL= nouvelle_page.php" />
```

Tous les navigateurs savent reconnaître ces balises, ce qui n'est pas le cas de tous les aspirateurs web. En outre, cette page intermédiaire apparaîtra dans l'historique du navigateur de l'utilisateur.

Recette 60 : Imposer l'utilisation de pages chiffrées par SSL

Lorsque l'on manipule des cartes de crédit, il faut garantir que toutes les informations liées aux cartes passent toujours par une connexion SSL (*Secure Socket Layer*).

Quand un utilisateur fait pointer son navigateur vers `www.exemple.com`, l'URL est considérée comme étant `http://www.exemple.com/` et non `https://www.exemple.com/`. Ce n'est pas un problème si tous vos formulaires désignent spécifiquement des pages accessibles *via* `https://www.exemple.com/`, mais c'est un point assez difficile à vérifier, sans compter les problèmes de mise à jour si le nom de votre machine vient à être modifié.

Voici une fonction simple qui teste si un utilisateur se connecte *via* SSL ou non :

```
function test_SSL() {
    /* Vérifie que la page est en mode sécurisé */
    if ($_SERVER['SERVER_PORT'] == "443") {
        return true;
    } else {
        return false;
    }
}
```

Ce code fonctionne en testant le port du serveur sur lequel se connecte le client (SSL utilise le port 443). Si l'accès n'est pas sécurisé alors qu'il devrait l'être, vous pouvez utiliser `$_SERVER['PHP_SELF']` et la fonction `header()` décrite dans la section précédente pour rediriger l'utilisateur vers une version sécurisée de la page.

Recette 61 : Obtenir des informations sur le client

Les serveurs web peuvent extraire des informations sur les clients qui se connectent en examinant l'état TCP/IP et les en-têtes HTTP. Vous pouvez connaître très simplement l'adresse IP et la version du navigateur du client en utilisant les variables PHP. Cependant, comme quasiment tout ce que vous envoie le client, ces informations sont totalement inutiles pour le suivi des sessions : à cause des proxy et des passerelles NAT (*Network Address Translation*), les adresses IP ne sont pas uniques à un client et les versions des navigateurs peuvent être totalement fausses.

Ces informations permettent cependant de disposer de statistiques sur vos utilisateurs. Quelques en-têtes faux ne sont pas réellement un problème lorsque l'on étudie 100 000 accès.

La fonction suivante extrait une adresse IP. Sa première partie est relativement simple puisqu'elle examine ce que le serveur pense savoir du client :

```
function get_ip() {
    /* Recherche d'abord une adresse IP dans les données du serveur */
    if (!empty($_SERVER["REMOTE_ADDR"])) {
        $ip_client = $_SERVER["REMOTE_ADDR"];
    }
}
```

Selon votre intérêt pour cette information, ce code peut suffire. Toutefois, si vous voulez savoir si un utilisateur est passé par un serveur mandataire (proxy), le code suivant permettra de rechercher ce proxy et de trouver l'adresse IP sous-jacente du client :

```
/* Recherche du serveur mandataire. */
if ($_SERVER["HTTP_CLIENT_IP"]) {
    $ip_proxy = $_SERVER["HTTP_CLIENT_IP"];
} else if ($_SERVER["HTTP_X_FORWARDED_FOR"]) {
    $ip_proxy = $_SERVER["HTTP_X_FORWARDED_FOR"];
}
}
```

La recherche de clients derrière des serveurs mandataires est gênée par le fait que de nombreux clients utilisent un mandataire parce qu'ils sont sur un réseau privé. Or, une adresse IP sur un réseau privé est inutile puisqu'elle n'est pas unique et ne fournit aucune information géographique. En ce cas, il est préférable d'utiliser l'adresse IP originale. Le code suivant recherche une adresse IP valide et, s'il la trouve, teste si elle appartient à un réseau privé, auquel cas on utilise l'adresse IP originale :

```
/* Recherche la véritable adresse IP sous un mandataire */
if ($ip_proxy) {
    if (preg_match("/^([0-9]+\.[0-9]+\.[0-9]+\.[0-9]+)/",
        $ip_proxy, $liste_ip)) {
        $ip_privees = array(
            '/^0\./',
            '/^127\.\0\.\0\.\1/',
            '/^192\.\168\.\.\./',
            '/^172\.\16\.\.\./',
            '/^10\.\.\./',
            '/^224\.\.\./',
            '/^240\.\.\./',
        );
        $ip_client = preg_replace($ip_privee, $ip_client, $liste_ip[1]);
    }
}
}
```

Enfin, on renvoie l'adresse IP que l'on pense avoir trouvée :

```
return $ip_client;
}
```

PHP stocke les informations sur l'agent du client dans la variable `$_SERVER['HTTP_USER_AGENT']`. Malheureusement, cette chaîne est complexe et ne respecte pas de standard bien établi. Voici, par exemple, ce que l'on obtiendrait avec Internet Explorer sous Windows :

```
Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1)
```

Avec Opéra, elle aurait cette forme :

```
Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1) Opera 7.54 [en]
```

Et voici ce que l'on obtiendrait avec Firefox sous Linux :

```
Mozilla/5.0 (X11; U; Linux i686; en-US; rv:1.8.1.4) Gecko/20070515 Firefox/2.0.0.4
```

La raison pour laquelle les chaînes des agents utilisateurs semblent être des mensonges éhontés est que la plupart des navigateurs essaient de tromper le serveur en se faisant passer pour ce qu'ils ne sont pas.

Ici, Internet Explorer prétend être Mozilla (le nom de code de Netscape Navigator) ; Opéra se nomme aussi Mozilla, puis indique qu'il est Explorer (MSIE 6.0) et, à la fin, annonce qu'il s'appelle Opéra. Ce comportement complètement idiot est dû à une pratique, deux fois plus stupide, consistant à optimiser les pages pour des navigateurs particuliers.

Les seules parties de cette chaîne susceptibles de vous intéresser sont le nom du navigateur, sa version et le nom du système d'exploitation car elles vous donneront des informations sur vos utilisateurs. Si, par exemple, vous constatez que vous avez une large proportion d'utilisateurs Mac sur votre boutique en ligne, vous avez tout intérêt à leur proposer des produits spécifiques. Voici une fonction qui extrait ces informations sous forme lisible. Le code est assez horrible mais, ici, nous devons combattre l'horreur par l'horreur.

Nous initialisons d'abord le tableau résultat et nous plaçons la chaîne d'identification du navigateur dans la variable `$agent` :

```
function trouve_navigateur() {
    // Détermine le SE, la version et le type du navigateur du client.
    $infos_navigateur = array(
        "nom" => "Unknown",
        "version" => "Unknown",
        "SE" => "Unknown",
    );
}
```

```
// Lit la chaîne de l'agent utilisateur.
if (!empty($_SERVER["HTTP_USER_AGENT"])) {
    $agent = $_SERVER["HTTP_USER_AGENT"];
}
}
```

Trouvons maintenant le système d'exploitation de l'utilisateur. C'est relativement simple puisque ces chaînes sont uniques :

```
// Trouve le système d'exploitation.
if (preg_match('/win/i', $agent)) {
    $infos_navigateur["SE"] = "Windows";
} else if (preg_match('/mac/i', $agent)) {
    $infos_navigateur["SE"] = "Macintosh";
} else if (preg_match('/linux/i', $agent)) {
    $infos_navigateur["SE"] = "Linux";
}
}
```

La partie épineuse pour extraire les détails du navigateur peut alors commencer. L'ordre dans lequel les navigateurs sont testés est important, car certaines chaînes prétendent être plusieurs navigateurs différents. Le plus gros menteur étant Opéra, on doit commencer par lui. Avec ce dernier, le pire est qu'il peut indiquer sa version de deux façons, avec ou sans barre de fraction ; il faut donc essayer les deux :

```
if (preg_match('/opera/i', $agent)) {
    // On commence par Opera, puisqu'il correspond aussi à IE
    $infos_navigateur["nom"] = "Opera";
    $agent = strstr($agent, "Opera");
    if (strpos("/", $agent)) {
        $agent = explode("/", $agent);
        $infos_navigateur["version"] = $agent[1];
    } else {
        $agent = explode(" ", $agent);
        $infos_navigateur["version"] = $agent[1];
    }
}
```

Le suivant sur la liste des suspects est Internet Explorer, parce qu'il prétend s'appeler Mozilla. L'obtention de sa version est plus simple puisqu'il suffit de supprimer le point-virgule final :

```
} else if (preg_match('/msie/i', $agent)) {
    $infos_navigateur["nom"] = "Internet Explorer";
    $agent = strstr($agent, "msie");
    $agent = explode(" ", $agent);
    $infos_navigateur["nom"] = str_replace(";", "", $agent[1]);
}
```

Pour l'instant, à part lui, aucun autre navigateur ne prétend s'appeler Firefox, mais Firefox est lui-même une version de Mozilla, c'est donc le moment de tester ce navigateur. Vous remarquerez que l'extraction de la version est bien plus simple :

```
} else if (preg_match('/firefox/i', $agent)) {
    $infos_navigateur["nom"] = "Firefox";
    $agent = stristr($agent, "Firefox");
    $agent = explode("/", $agent);
    $infos_navigateur["nom"] = $agent[1];
}
```

Safari prétend utiliser KHTML, "like Gecko", le moteur de Mozilla. Comme nous utilisons aussi Gecko pour trouver les différentes versions de Mozilla, nous devons d'abord tester Safari :

```
} else if (preg_match('/safari/i', $agent)) {
    $infos_navigateur["nom"] = "Safari";
    $agent = stristr($agent, "Safari");
    $agent = explode("/", $agent);
    $infos_navigateur["version"] = $agent[1];
}
```

Netscape Navigator est, évidemment, une version de Mozilla :

```
} else if (preg_match('/netscape/i', $agent)) {
    $infos_navigateur["nom"] = "Netscape Navigator";
    $agent = stristr($agent, "Netscape");
    $agent = explode("/", $agent);
    $infos_navigateur["version"] = $agent[1];
}
```

Enfin, si l'on a affaire à un navigateur utilisant le moteur Gecko, on sait qu'il s'agit sûrement de Mozilla ou de l'une de ses variantes :

```
} else if (preg_match('/Gecko/i', $agent)){
    $infos_navigateur["nom"] = 'Mozilla';
    $agent = stristr($agent, "rv");
    $agent = explode(":", $agent);
    $agent = explode(")", $agent[1]);
    $infos_navigateur["version"] = $agent[1];
}
return $infos_navigateur;
}
```

Comme on l'a indiqué plus haut, cette fonction est assez horrible car elle se contente d'examiner au hasard la chaîne de l'agent jusqu'à trouver une information semblant intelligible. Vous devrez stocker le résultat de cette fonction pour

pouvoir l'exploiter ensuite. Si vous avez accès aux fichiers journaux de votre serveur, il est sans doute préférable d'utiliser un analyseur comme awstats (<http://awstats.sourceforge.net/>), qui peut extraire un grand nombre d'informations, dont les adresses IP et les navigateurs utilisés par vos visiteurs.

Recette 62 : Délais d'expiration des sessions

Sur les sites très sécurisés, les utilisateurs ne devraient pas pouvoir rester connectés trop longtemps. Si un visiteur s'absente de son poste pendant une session sur un site comme PayPal, quelqu'un d'autre pourrait profiter de son ordinateur pour détourner de l'argent de son compte. Pour éviter cela, ces sites utilisent les délais d'expiration des sessions, qui déconnectent automatiquement les utilisateurs qui n'ont rien fait pendant une certaine période de temps assez courte (10 minutes, par exemple). Il faut bien noter qu'il ne faut pas le faire sur les sites qui n'exigent pas une sécurité aussi poussée, car cela ennuie les utilisateurs.

Voici deux fonctions qui implémentent des délais d'expiration pour les sessions. Vous remarquerez que les variables qui contiennent les délais sont des variables de session – les informations provenant du navigateur n'étant pas dignes de confiance, vous devez stocker ces valeurs sur votre serveur. La première fonction valide la session de connexion :

```
function valide_login () {
    /* Mise en place d'un délai pour une session de connexion. */
    /* L'expiration est de 10 minutes par défaut (600 secondes). */
    @session_start();
    $delai = 600;
    $_SESSION["expires_by"] = time() + $delai;
}
```

NOTE *Si vous êtes sûr que la session a déjà débuté au moment où vous appelez cette fonction, vous pouvez supprimer l'appel @session_start().*

La seconde fonction vérifie la connexion courante pour savoir si elle a expiré. Si la session est valide, elle réinitialise son délai d'expiration :

```
function verif_login () {
    @session_start();
    /* Vérifie le délai d'expiration de la session */
    $expiration = intval($_SESSION["expires_by"]);
    if (time() < $expiration) {
        /* La session est toujours en cours ; on réinitialise son délai.*/
        valide_login();
        return true;
    } else {
```

```
/* la session a expiré ; on supprime la variable de session. */
unset($_SESSION["expires_by"]);
return false;
}
}
```

La raison de la présence de deux fonctions distinctes et qu'il faut configurer le délai d'expiration pour la première fois avec `valide_login()` lorsque l'utilisateur se connecte pour la première fois. Bien que cette mise en place du délai soit très simple, il faut qu'elle soit cohérente sous peine de compliquer les choses plus tard.

L'utilisation de `verif_login()` est très simple ; voici un exemple permettant de protéger les pages nécessitant une connexion à partir de la page *login.php* :

```
<?
if (!verif_login()) {
    header("Location: login.php");
    exit(0);
}
?>
```

Comme pour le suivi des utilisateurs, n'oubliez pas que `session_start()` doit être appelée au début du script, avant la production de tout en-tête. Ces fonctions ignorent les erreurs de `session_start()` car elles pourraient provenir d'appels précédents.

Recette 63 : Système de connexion simple

Certains sites web ont besoin d'un système d'authentification simple pour les tâches administratives. Les petits sites, qui n'ont que deux administrateurs, ne nécessitent pas un système de connexion complet avec des noms d'utilisateurs individuels : il suffit simplement d'écarter les pirates. Un exemple de ce type de site serait celui de la recette n°54, "Déposer des images dans un répertoire".

Nous présenterons ici le code permettant d'effectuer une authentification pour un seul utilisateur. L'idée générale consiste à mettre en place une variable de session `$_SESSION["auth"]` à remplir lorsque l'on est connecté.

On définit d'abord le mot de passe. Comme d'habitude, on ne le stocke pas en clair. Celui-ci est un hachage MD5 – vous devrez modifier cette chaîne par celle que vous aurez produite (nous verrons bientôt comment procéder).

```
<?
$mdp_enc = "206bfaa5da7422d2f497239dcf8b96f3";
```

Commençons par définir ce qu'il faut faire quand quelqu'un se déconnecte (ce qui est signalé par le paramètre `logout`). On initialise d'abord la variable de

session avec `incomplet`, on envoie une page générique à l'utilisateur (*index.php*, ici), puis on sort :

```
session_start();
if ($_REQUEST["logout"]) {
    $_SESSION["auth"] = "incomplet";
    header("Location: index.php");
    exit(0);
}
```

L'utilisateur se connecte au moyen d'un paramètre `mdp`. Pour savoir si le mot de passe est correct, on calcule le hachage MD5 de ce paramètre et on le compare à celui du mot de passe stocké. S'ils sont égaux, on valide la variable de session en la déclarant terminée et on redirige l'utilisateur vers une page d'accueil. Selon la façon dont votre site envoie les paramètres des formulaires, vous n'aurez pas besoin de cette redirection, mais vous courez alors le risque de perdre vos paramètres d'origine et de ne plus être authentifié. Il est donc plus sûr de rediriger les nouvelles connexions vers une sorte de page d'accueil.

```
if ($_REQUEST["mdp"]) {
    if (md5($_REQUEST["mdp"]) == $mdp_enc) {
        $_SESSION["auth"] = "terminee";
        header("Location: index.php");
        exit(0);
    }
}
```

Si l'on est ici, nous savons que l'on n'est pas en train de se connecter ou de se déconnecter ; la seule chose à faire consiste donc à vérifier si l'utilisateur est déjà connecté. N'oubliez pas que cette information se trouve dans la variable `$_SESSION["auth"]` et qu'il faut donc la vérifier. Si l'utilisateur n'est pas connecté, on lui donne la possibilité de le faire en affichant un formulaire de connexion, puis on sort. On pourrait également rediriger l'utilisateur vers une page de connexion spéciale, mais il est important de toujours sortir après cette redirection ou l'affichage du formulaire car, après cette étape, PHP ne doit plus exécuter la moindre tâche accessible à un utilisateur authentifié !

```
$auth_ok = $_SESSION["auth"];
if ($auth_ok != "termine") {
    ?><html><head></head><body>
        <form method="post">
            Entrez un mot de passe : <input type="password" name="mdp"/>
        </form>
    </body></html><?php
    exit(0);
}
?>
```

Pour utiliser ce script, nommez-le *login.php* et incluez-le au début de tout script ayant besoin d'une authentification. Lorsqu'un client rencontre la page pour la première fois, le script affiche un formulaire de connexion et se termine avant de laisser à un autre code le temps de s'exécuter.

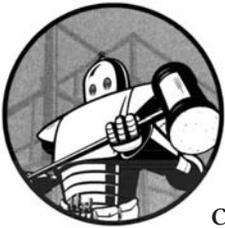
Pour mettre en place un nouveau mot de passe, lancez le script suivant pour produire un nouvel hachage, puis copiez la chaîne obtenue dans la variable `$mdp_enc` :

```
<?
print md5("nouveau mot de passe");
?>
```

Ne perdez pas de vue qu'il s'agit d'un système d'authentification très simple. Vous pouvez l'améliorer en ajoutant le code d'expiration des sessions de la section précédente mais, si vous avez besoin de fonctionnalités supplémentaires, il est sûrement préférable de partir d'un système d'authentification parfaitement testé. Vous trouverez plusieurs systèmes gratuits sur l'Internet qui méritent d'être essayés.

9

TRAITEMENT DU COURRIER ÉLECTRONIQUE



En général, vous ne traiterez pas beaucoup de courrier électronique avec PHP, mais vous devez au moins savoir comment envoyer des messages de confirmation aux utilisateurs et aux administrateurs pour leur confirmer l'activation de leurs comptes, la prise en compte de leurs commandes, etc.

Le moyen le plus simple d'envoyer du courrier électronique avec PHP consiste à utiliser la fonction `mail()`. Si votre serveur de courrier est correctement configuré et que vous n'avez besoin de n'envoyer des messages qu'à vous-même, c'est sûrement la seule fonction dont vous aurez besoin.

Voici un script simple qui illustre le fonctionnement de `mail()`. Il suffit de remplacer `toto@exemple.com` par une adresse de courrier valide :

```
if (mail('toto@exemple.com', 'Test courrier PHP, 'Ça marche !')) {  
    echo "Courrier envoyé."  
} else {  
    echo "Échec de l'envoi du courrier."  
}
```

Si ce script affiche *Courrier envoyé*, vérifiez la boîte de réception du destinataire afin de vous assurer que tout a bien fonctionné. Le sujet devrait être *Test courrier PHP* et le corps du message *Ça marche !*

La fonction `mail()`, comme tous les autres systèmes qui expédient du courrier électronique, peut poser des problèmes au système de délivrance du courrier : il sera donc peut-être nécessaire de modifier la configuration de votre serveur de courrier pour que cela fonctionne. Ceci dit, il y a tellement de spam dans les courriers actuels qu'un en-tête de courrier mal formé ou anormal peut provoquer le rejet de votre courrier par un serveur distant sans même qu'il vous prévienne. En outre, l'envoi de pièces attachées ou l'ajout de texte HTML demande beaucoup plus de travail. La section suivante montre comment résoudre ces problèmes avec PHPMailer.

Recette 64 : Envoyer du courrier avec PHPMailer

PHPMailer est un paquetage Open Source de gestion du courrier électronique qui reconnaît les fichiers attachés, les destinataires multiples, l'authentification SMTP et qui dispose d'un grand nombre d'autres fonctionnalités. Il a été abondamment testé et il est relativement simple à utiliser et à mettre à jour. Il suffit d'inclure le fichier PHPMailer principal dans votre script et vous êtes prêt à envoyer du courrier.

Installation de PHPMailer

L'installation de PHPMailer s'effectue en suivant ces étapes :

1. Téléchargez les fichiers de PHPMailer à partir de <http://phpmailer.sourceforge.net/>.
2. Créez un répertoire *phpmailer* sur votre serveur afin d'y stocker les fichiers de PHPMailer.
3. Extrayez les fichiers de PHPMailer dans le répertoire *phpmailer*.
4. Choisissez votre méthode de transport du courrier. PHPMailer propose trois méthodes différentes : `mail`, `sendmail` et `smtp`. `mail` est la méthode par défaut ; elle utilise la fonction `mail()` de PHP, décrite dans la section précédente. C'est la plus simple à configurer si le courrier du serveur web est correctement paramétré. Si cette méthode ne fonctionne pas, il vous reste deux possibilités :
 - a. Vous pouvez indiquer à PHPMailer un serveur SMTP auquel il pourra s'adresser. SMTP signifie *Simple Mail Transfer Protocol*, c'est le protocole de transport du courrier le plus utilisé. Pour que PHPMailer puisse utiliser SMTP, vous devez connaître le nom d'hôte d'un serveur SMTP. Si celui-ci exige une authentification (ce qui est souvent le cas), vous devrez fournir un nom d'utilisateur et un mot de passe pour ce serveur. Votre FAI vous fournira tous les détails nécessaires sur la configuration de son serveur SMTP.

- b. Si votre serveur web utilise `sendmail` ou un logiciel compatible (comme `Postfix`), vous pouvez configurer `PHPMailer` afin qu'il l'utilise pour envoyer le courrier. Vous devrez alors indiquer l'emplacement de l'exécutable `sendmail`, qui est généralement `/usr/sbin/sendmail` ou `/usr/lib/sendmail`.
5. Ajustez la configuration par défaut de `PHPMailer` en modifiant le contenu du fichier `class.phpmailer.php`. Les variables à modifier se trouvent dans la section *Public Variables* située au début du fichier. Les réglages les plus importants sont :

`var $Mailer = "mail";` La méthode utilisée par `PHPMailer` pour envoyer le courrier. Utilisez la valeur `mail`, `sendmail` ou `smtp`, comme on l'a expliqué à l'étape 4.

`var $From = "root@localhost";` L'adresse de l'expéditeur par défaut.

`var $FromName = "Root User";` Le nom par défaut associé à l'adresse de courrier par défaut.

`var $Host = "";` Le serveur SMTP utilisé avec la méthode `smtp`. Vérifiez les informations que vous a communiquées votre FAI. Vous pouvez indiquer plusieurs serveurs SMTP en les séparant par des points-virgules au cas où le premier serveur serait hors service ou rejetterait vos courriers.

`var $SMTPAuth = false;` Si votre serveur SMTP exige une authentification pour envoyer du courrier, mettez cette variable à `true`. En ce cas, vous devrez également initialiser les deux variables suivantes.

`var $Username = "";` Le nom d'utilisateur sur le serveur SMTP (uniquement lorsqu'on utilise l'authentification SMTP).

`var $Password = "";` Le mot de passe associé à `$Username`, si nécessaire.

`var $Helo = "";` Le nom de votre serveur web : `www.exemple.com`, par exemple.

Après avoir modifié les variables de configuration nécessaires, vous êtes prêt à étudier un script simple permettant d'envoyer un message de test.

Utilisation du script

Assurez-vous que le fichier `class.phpmailer.php` se trouve dans un des chemins où PHP recherche les fichiers inclus, puis essayez ce script :

```
<?php
include_once("class.phpmailer.php");
$mail = new PHPMailer;
$mail->ClearAddresses();
$mail->AddAddress('toto@adresse.com', 'toto');
$mail->From = 'toi@exemple.com';
$mail->FromName = 'Ton nom';
```

```
$mail->Subject = 'Sujet du message de test';
$mail->Body = 'Voici le corps du message de test.';
if ($mail->Send()) {
    echo "Message envoyé.";
} else {
    echo $mail->ErrorInfo;
}
?>
```

L'interface de PHPMailer étant orientée objet, vous devez créer un objet (`$mail`, ici) puis configurer quelques attributs et appeler des méthodes pour construire le message. Utilisez ensuite la méthode `Send()` de l'objet pour envoyer le message. Cette méthode renvoie `true` si PHPMailer a pu transmettre le message à l'agent de distribution. En cas de problème, vous trouverez tous les détails dans la variable `ErrorInfo`.

Voici un résumé des méthodes et attributs utilisés dans cet exemple :

```
$mail->AddAddress(adresse_mel, nom_destinataire)
```

Ajoute un destinataire pour le message courant. *adresse_mel* est l'adresse du destinataire et *nom_destinataire* est son nom réel (ou, au moins, le nom que vous donnez au destinataire).

```
$mail->ClearAddresses()
```

Vide la liste courante des destinataires. La méthode `AddAddress()` ne supprimant pas les adresses précédentes, vous risquez d'envoyer plusieurs fois le même message au même destinataire si vous envoyez les courriers dans une boucle avec le même objet PHPMailer. Il est donc conseillé de prendre l'habitude de vider la liste des destinataires avant de traiter un message ou après avoir appelé la méthode `Send()`.

```
$mail->isHTML = true|false
```

Si cet attribut vaut `true`, PHPMailer utilise HTML au lieu du texte pur. Avec HTML, vous pouvez intégrer des belles images, mais tous les clients de courrier n'acceptent pas le HTML. En cas de problème, vérifiez la valeur de l'attribut suivant.

```
$mail->AltBody = texte
```

Si l'attribut `isHTML` vaut `true`, vous pouvez configurer l'attribut `AltBody` avec une version texte du corps du message.

Ajout de fichiers attachés

Pour attacher des fichiers aux messages, utilisez cette méthode :

```
$mail->AddAttachment(chemin, nom, encodage, type)
```

Ses paramètres sont les suivants :

chemin : Le chemin complet du fichier que l'on veut attacher.

nom : Le nouveau nom pour le fichier attaché. Si, par exemple, le fichier s'appelle *011100.jpg* sur votre système mais que vous voulez qu'il s'appelle *exemple_produit.jpg* sur la machine destinataire, passez cette chaîne au paramètre. Celui-ci est facultatif, mais vous devriez toujours l'utiliser pour vous assurer que le destinataire sauvegardera correctement le fichier attaché.

encodage : L'encodage du fichier attaché. Par défaut, il s'agit de base64, qui convient parfaitement aux attachements binaires.

type : Le type MIME du fichier attaché. Le type par défaut, *application/octet-stream*, convient à la plupart des fichiers mais, dans certains cas, vous pouvez avoir envie de le modifier (*image/jpeg*, par exemple, correspond aux images JPEG). Cependant, à moins de savoir ce que vous faites, il est préférable de ne pas s'en occuper et de laisser le client courrier du destinataire le deviner.

Pour envoyer le message avec la pièce jointe, il suffit d'utiliser normalement la méthode `Send()` de PHPMailer. Si vous devez supprimer toutes les pièces jointes d'un objet (parce vous bouclez sur des destinataires ayant chacun un fichier attaché unique), choisissez cette méthode :

```
$mail->ClearAttachments()
```

Enfin, si vous avez stocké un fichier attaché dans une variable PHP, vous pouvez utiliser la méthode `AddStringAttachment()` pour l'attacher. Elle fonctionne exactement comme `AddAttachment()`, mais elle se sert d'une chaîne PHP ou d'une variable à la place du paramètre `chemin`.

Problèmes éventuels

Si vous utilisez la fonction `mail()` pour envoyer le courrier, vous devez vérifier que PHP puisse le faire. Dans le cas contraire, utilisez plutôt `smtp` ou `sendmail`.

Si vous envoyez un courrier avec SMTP, vous pouvez recevoir ce message d'erreur :

```
SMTP Error: The following recipients failed [email@exemple.com]
```

Dans la plupart des cas, cela signifie qu'il y a eu un problème lors de la connexion au serveur SMTP. Vous vous êtes peut-être trompé en tapant le nom

du serveur SMTP ou le serveur est indisponible. Cependant, la raison la plus fréquente est que le serveur exige une authentification SMTP : en ce cas, vérifiez que la variable de configuration SMTPAuth vaut true, mettez le nom d'utilisateur et le mot de passe pour ce serveur puis réessayez.

Il est important de se rappeler que la gestion du courrier est un traitement complexe. Il existe des milliers de raisons pour lesquelles un courrier pourrait ne pas parvenir à une adresse donnée : cette adresse peut être incorrecte, votre serveur peut être considéré comme un serveur de spam et donc être dans la liste noire de nombreux autres serveurs, ou un serveur peut ne pas aimer vos fichiers attachés – et ce n'est que le début.

Le meilleur moyen de déboguer un problème consiste à commencer par un message en texte pur expédié à une adresse dont on est sûr. À partir de là, vous pouvez vous plonger dans les fichiers journaux du serveur pour repérer les problèmes particuliers.

Recette 65 : Vérifier les comptes utilisateurs avec le courrier électronique

Sur les sites qui demandent une inscription, certaines personnes créent des comptes uniquement pour semer la zizanie. Certains en profitent pour poster une tonne de commentaires ridicules sur votre forum ou pour tenter de saturer une autre partie de votre système. Avec les boutiques en ligne, un problème classique est qu'un client peut faire une commande en utilisant une fausse adresse de courrier électronique afin de ne pas être spammé, ce qui vous empêche de le joindre si vous avez besoin de lui poser une question sur sa commande.

Un moyen assez efficace de vérifier l'identité réelle des utilisateurs consiste à les obliger à valider leurs adresses électroniques. Lorsque l'on crée le compte d'un utilisateur, celui-ci ne sera donc pas activé tant que cet utilisateur n'aura pas cliqué sur un lien qui lui aura été transmis par courrier électronique.

Dans cette section, nous allons présenter un système qui prend en charge les nouveaux utilisateurs qui n'ont pas encore activé leur compte. Lorsqu'un utilisateur tente de se connecter à votre système, vous pouvez d'abord vérifier si ce compte a été activé ou non. Pour cela, vous avez besoin des composants suivants :

- une base de données MySQL ;
- une table `attentes_inscription` formée de deux colonnes : un nom d'utilisateur (*login*) et une clé. Pour la créer, vous pouvez utiliser la requête SQL suivante :

```
CREATE TABLE attentes_inscription (  
    'login' varchar(32), 'cle' varchar(32),  
    PRIMARY KEY ('login'),  
    INDEX (cle));
```

- PHPMailer, installé dans le répertoire *phpmailer*.

On utilisera trois fonctions qui joueront le rôle de générateur, d'activateur et de vérificateur. Toutes ces fonctions supposent que le nom d'utilisateur a déjà été validé et que c'est une chaîne MySQL correcte. Étudions d'abord la fonction générateur. Vous devrez l'appeler dans le code de création de compte pour produire une clé permettant de débloquent le compte et envoyer cette clé à l'adresse électronique de l'utilisateur. Cette fonction commence par produire une chaîne de 32 caractères aléatoires qui serviront de clé pour débloquent le compte :

```
function verification($login, $email, $bd) {
    /* Crée un lien de vérification et l'envoi à l'utilisateur */
    /* Crée une clé */
    $cle = ""; $i = 0;
    while ($i < 32) {
        $cle .= chr(rand(97, 122));
        $i++;
    }
}
```

Puis, nous plaçons la clé dans la table *attentes_inscription*. La clé primaire étant *login*, il ne peut pas y avoir plusieurs noms de comptes identiques ; nous vérifions donc d'abord qu'il n'existe pas déjà une ligne pour ce compte, puis nous insérons les données dans la table :

```
/* Place la clé dans la table
   On supprime d'abord un éventuel compte identique */
$requete = "DELETE FROM attentes_inscription WHERE login = '$login'";
mysql_query($requete, $bd);
$requete = "INSERT INTO attentes_inscription (login, cle)
           VALUES ('$login','$cle')";
mysql_query($requete, $bd);
if (mysql_error($bd)) {
    print "Erreur de génération de la clé.";
    return false;
}
```

Nous devons maintenant produire l'URL sur laquelle l'utilisateur devra se rendre pour activer son compte. Vous devrez évidemment modifier celle-ci en fonction du nom de votre serveur et du script d'activation, mais vous devez surtout vous assurer d'envoyer la clé en paramètre :

```
/* URL d'activation */
$url = "http://comptes.exemple.com/activation.php?k=$cle";
```

Il reste simplement à envoyer le courrier à l'utilisateur. Là encore, il est probable que vous personnalisiez cette partie.

```

include_once("phpmailer/class.phpmailer.php");
$mail = new PHPMailer;
$mail->ClearAddresses();
$mail->AddAddress($mail, $login);
$mail->From = 'generateur@exemple.com';
$mail->FromName = 'Générateur de compte';
$mail->Subject = 'Vérification du compte';
$mail->Body = "Pour activer votre compte, cliquez sur l'URL suivante :

$url

";
if ($mail->Send()) {
    print "Message de vérification envoyé.";
} else {
    print $mail->ErrorInfo;
    return false;
}
return true;
}

```

Pour utiliser cette fonction, appelez-la de la façon suivante (bd est un descripteur de base de données MySQL qui a été ouvert au préalable). Elle renverra true si le compte a été placé dans la table `attentes_inscription` et si PHPMailer a pu envoyer le message d'activation :

```

verification(login, adresse_mel, bd)

```

Cette vérification ayant été faite, la partie suivante est une fonction qui active un compte lorsque l'utilisateur clique sur le lien. La première chose à faire consiste à nettoyer la clé qui nous a été envoyée au cas où l'utilisateur l'ait abîmé ou qu'un pirate essaie de pénétrer sur le système. La fonction précédente qui a produit la clé n'utilisant que des minuscules, nous supprimons tout ce qui ne correspond pas :

```

function activer_compte($cle, $bd) {
    /* Active un compte à partir d'une clé. */

    /* Nettoie la clé si nécessaire. */
    $cle = preg_replace("/[^a-z]/", "", $cle);

```

On examine ensuite la validité de la clé. Si elle n'est même pas dans la table `attentes_inscription`, il n'y a rien à faire et on renvoie false pour l'indiquer :

```

$requete = "SELECT login FROM attentes_inscription WHERE cle = '$cle';
$c = mysql_query($requete, $bd);
if (mysql_num_rows($c) != 1) {
    return false;
}

```

Si l'on est arrivé ici, nous savons que la clé est dans la table : il reste donc à supprimer la ligne correspondante pour activer le compte. Vous remarquerez que l'on n'a même pas besoin de savoir quel est le nom du compte.

```
$requete = "DELETE FROM attentes_inscription WHERE cle = '$cle";
mysql_query($requete, $bd);
if (mysql_error($bd)) {
    return false;
}
return true;
}
```

Pour utiliser cette fonction, appelez-la de la façon suivante :

```
activer_compte($_REQUEST["k"], db)
```

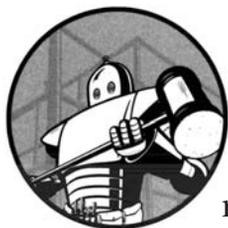
La dernière fonction permet de savoir si un compte est actif ou non. Si un compte n'a pas été activé, une ligne lui correspond dans la table `attentes_inscription` ; il suffit donc de rechercher ce compte dans cette table :

```
function est_actif($login, $bd) {
    /* Teste si un compte a été activé. */
    $requete = "SELECT count(*) AS c FROM attentes_inscription
                WHERE login = '$login'";
    $c = mysql_query($requete, $bd);
    if (mysql_error($bd)) {
        return false;
    }
    $r = mysql_fetch_array($c);
    if (intval($r["c"]) > 0) {
        return false;
    }
    return true;
}
```

Pour adapter ce système à votre site, vous pouvez lui ajouter un certain nombre de choses : ajouter un champ `date` à la table pour supprimer les comptes inactifs qui n'ont jamais été vérifiés, par exemple. Il peut y avoir beaucoup d'autres raisons de désactiver un compte ; en ce cas, vous aurez besoin de stocker dans la table la raison de cette désactivation. Vous pouvez même inclure la clé d'activation dans la table principale des comptes. Le mécanisme d'activation décrit ici est spécifiquement conçu pour être greffé à moindre frais sur d'autres systèmes de connexion.

10

TRAITEMENT DES IMAGES



Ce chapitre explique comment créer et manipuler des images GIF et JPEG. Nous ne vous expliquerons pas comment développer des scripts capables d'effectuer de lourdes manipulations d'images en ligne ou de reconnaître des caractères à la volée (votre serveur limite de toutes manières le nombre de calculs que peut réaliser un script), mais vous découvrirez en détail de nombreuses opérations sur les images.

Vous serez notamment en mesure de générer aléatoirement des images qui agiront comme des codes de vérification ou de décliner vos clichés en de multiples vignettes. Pratique pour tous vos projets de sites web !

Recette 66 : Créer une image CAPTCHA pour améliorer la sécurité

De nombreux exemples de ce livre utilisant cURL pour se connecter et interagir avec les sites web, vous savez donc qu'il est facile d'automatiser ces interactions. Vous savez également que si votre site dispose d'une fonctionnalité que vous ne voulez offrir qu'aux visiteurs et non à des robots spammeurs, vous devez

la protéger. Sinon, votre site risquerait d'être totalement submergé par ces derniers. Un moyen de contourner ce problème consiste à produire dynamiquement une image contenant un texte que l'utilisateur devra reproduire avant de continuer. Si le texte saisi par le visiteur correspond au texte de l'image, c'est qu'il s'agit vraisemblablement bien d'une personne et non d'un robot.

Ce type de test s'appelle CAPTCHA (*Completely Automated Public Turing Test to Tell Computers and Humans Apart*, ce qui peut se traduire par "Test de Turing automatique pour distinguer les ordinateurs des humains"). Cette méthode présente pourtant deux inconvénients. Le premier concerne évidemment les malvoyants qui ne peuvent pas lire le texte de l'image CAPTCHA. De nos jours, les visiteurs malvoyants peuvent utiliser Internet grâce à des logiciels vocaux qui lisent les pages, mais ces logiciels ne savent pas interpréter les images dépourvues d'attribut alt. Un CAPTCHA ne tient donc pas compte de ces utilisateurs.

En outre, un CAPTCHA ne fonctionne pas toujours. Les spammeurs sont toujours à la recherche de solutions pour contourner vos défenses et l'un des moyens les plus ingénieux qu'ils ont trouvé consiste à remplacer le lien hypertexte de votre image CAPTCHA par l'adresse d'un site pornographique qu'ils administrent. Ils n'hésitent pas à ajouter la mention "Entrez cette phrase et admirez de superbes créatures nues !" pour attirer certains visiteurs. Bien que vous ne souhaitez pas que votre site web soit piraté de la sorte et qu'il s'abaisse malgré lui à de telles pratiques, sachez que ce genre de techniques est encore relativement rare.

Connaissant ces deux faiblesses, vous devez utiliser les CAPTCHA parcimonieusement et les comparer avec d'autres méthodes comme le scanner Akismet¹.

Le CAPTCHA que nous présentons ici est assez simple (la Figure 10.1 en montre une copie d'écran) mais il est efficace. Pour l'utiliser, vous devez avoir installé la bibliothèque graphique GD avec la reconnaissance de Freetype. Comme on l'a expliqué à la recette n°8, "Afficher toutes les options de configuration de PHP", utilisez `phpinfo()` pour savoir si GD est installé sur votre serveur : si vous voyez une section GD avec des informations sur Freetype tout va bien. Sinon, recompilez PHP comme indiqué à la recette n°18, "Ajouter des extensions à PHP".

Vous aurez également besoin d'une ou plusieurs polices dans le même répertoire que le script. Il est préférable d'utiliser des polices qui sont à peu près lisibles.

1. Ndr : Vous avez également la possibilité, à l'image du projet ReCaptcha inventé par Luis von Ahn (le créateur du concept d'origine), d'enchaîner deux CAPTCHA d'affilée afin de compliquer la reconnaissance des caractères par des robots. Mais vos visiteurs devront redoubler d'effort pour valider leur saisie !

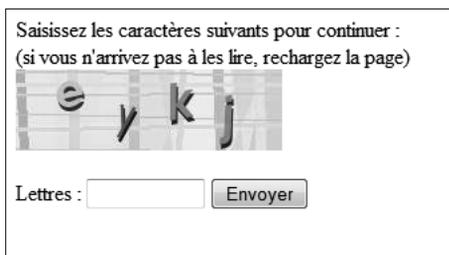


Figure 10.1 : Une image CAPTCHA

Ce système est formé de deux parties :

1. Le composant principal est un script qui crée des images CAPTCHA qui auront des fonds irréguliers avec d'étranges treillis colorés de lignes horizontales et verticales. Puis, le script demande une phrase secrète, la stocke dans la variable `$_SESSION['tt_pass']` et dessine chaque lettre de cette phrase de façon différente. Enfin, il envoie l'image au client.
2. Le second composant est un script qui compare ce qui a été saisi dans un formulaire CAPTCHA avec le contenu de la variable `$_SESSION['tt_pass']`. Le script que nous présentons ici n'est qu'un simple squelette, mais vous pourrez l'adapter aisément à vos besoins car le script de production d'image fait l'essentiel du travail.

Commençons par ce dernier. La première étape consiste à mettre en place plusieurs variables de configuration pour stocker la longueur de la phrase secrète, la taille de l'image et l'emplacement des polices :

```
<?php
/* image_captcha.php */

/* Longueur de la phrase secrète */
$lg_phrase = 4;

/* Dimensions de l'image*/
$largeur = 200;
$hauteur = 60;

/* Chemin des polices TTF */
$chemin_polices = dirname(__FILE__);
```

Créons maintenant la phrase secrète. À la différence de la plupart des mots de passe, cette phrase n'a pas besoin d'être particulièrement compliquée ; il n'est pas nécessaire d'ennuyer inutilement vos visiteurs.

```
session_start();
```

```
/* Création de la phrase secrète. */
$mdp = "";
$i = 0;
while ($i < $lg_phrase) {
    $mdp .= chr(rand(97, 122));
    $i++;
}
```

Nous placerons la phrase secrète dans une variable de session : elle n'apparaîtra donc jamais sur les pages, ce qui évite de devoir l'encoder.

```
/* Stockage de la phrase secrète. */
$_SESSION["tt_pass"] = $mdp;
```

Avant de dessiner quoi que ce soit, nous voulons nous assurer que nous disposons de certaines polices. Ce code crée donc une liste des polices du répertoire courant. Si vous voulez accélérer cette étape, vous pouvez initialiser le tableau `$polices` avec un ensemble de noms de fichiers de polices TTF :

```
/* Récupère la liste des polices disponibles. */
$polices = array();
if ($desc = opendir($chemin_polices)) {
    while (false !== ($fichier = readdir($desc))) {
        /* Recherche des polices TTF. */
        if (substr(strtolower($fichier), -4, 4) == '.ttf') {
            $polices[] = $chemin_polices . '/' . $fichier;
        }
    }
}

if (count($polices) < 1) {
    die("Aucune police n'a été trouvée !");
}
```

Le client doit savoir le type d'image que vous envoyez. Ce script créant une image JPEG, c'est ce type MIME qu'il enverra. En outre, il envoie également plusieurs en-têtes pour désactiver le cache et empêcher ainsi le navigateur de mettre l'image en cache :

```
/* En-tête de l'image */
header("Content-Type: image/jpeg");
/* Désactivation du cache. */
header("Expires: Mon, 01 Jul 1998 05:00:00 GMT");
```

```
header("Last-Modified: " . gmdate("D, d M Y H:i:s") . " GMT");
header("Cache-Control: no-store, no-cache, must-revalidate");
header("Cache-Control: post-check=0, pre-check=0", false);
header("Pragma: no-cache");
```

Après tout ce travail préparatoire, nous pouvons commencer à créer l'image. Les fonctions GD commencent toutes par le préfixe `image`. Pour initialiser la couleur du canevas de l'image, on appelle `imagecreatetruecolor()`.

```
/* Création de l'image. */
$img = imagecreatetruecolor($largeur, $hauteur);
```

Nous commencerons par tracer un rectangle qui recouvre tout le fond. Pour cela, il faut d'abord allouer une couleur dans l'image avec `imagecolorallocate()` dont le second, troisième et quatrième paramètres sont, respectivement, les valeurs rouge, verte et bleue. Ces valeurs sont des entiers compris entre 0 et 255, 0 étant la couleur la plus sombre et 255 la plus claire. L'instruction suivante crée une couleur pastel aléatoire :

```
/* Remplit le fond avec un pastel aléatoire */
$fond = imagecolorallocate($img, rand(210,255), rand(210,255),
                           rand(210,255));
```

On trace ensuite le rectangle avec la fonction `imagefilledrectangle()`. Le rectangle étant parallèle aux axes de l'image, on peut le définir par deux points, (0, 0) et (\$largeur, \$hauteur) :

```
imagefilledrectangle($img, 0, 0, $largeur, $hauteur, $fond);
```

Pour compliquer un peu plus le fond, on trace plusieurs polygones orientés verticalement sur le canevas. La boucle suivante crée les polygones avec la fonction `imagefilledpolygon()`. Le tracé est différent de celui d'un rectangle car un polygone peut avoir plus de trois côtés orienté différemment : on doit donc fournir un tableau de points en paramètre (`$points_poly`, ici) au lieu de deux coordonnées.

Ne vous perdez pas dans les détails de cette boucle car ce n'est pas une partie très importante du script.

```
/* Complique le fond en le recouvrant de polygones de couleurs
   différentes ayant chacun quatre sommets. */

/* Crée des treillis de 10 à 30 pixels de largeur sur l'image. */
$d droite = rand(10, 30);
$d gauche = 0;
while ($gauche < $largeur) {
```

```

$points_poly = array(
    $gauche, 0, /* Coin supérieur gauche */
    $droite, 0, /* Coin supérieur droit */
    rand($droite-25, $droite+25), $hauteur, /* Coin inférieur droit */
    rand($gauche-15, $gauche+15), $hauteur); /* Coin inférieur gauche */

/* Création du polygone à partir des quatre points du tableau */
$c = imagecolorallocate($img, rand(210,255), rand(210,255),
    rand(210,255));
imagefilledpolygon($img, $points_poly, 4, $c);

/* Avance vers le côté droit. */
$offset_aleatoire = rand(10, 30);
$gauche += $offset_aleatoire;
$droite += $offset_aleatoire;
}

```

Pour compliquer encore la tâche de ceux qui utiliseraient un logiciel de reconnaissance de caractères pour tenter de contourner ce système, nous tracerons quelques lignes verticales et horizontales aléatoires en travers de l'image. Pour mélanger tout cela un peu plus, nous définirons un intervalle de couleur différent pour les lignes de chaque image. En choisissant des limites aléatoires inférieure et supérieure pour les couleurs, les lignes peuvent ou non varier en intensité et changeront d'épaisseur d'une image à l'autre.

```

/* Choisit un intervalle de base pour les lignes
verticales et horizontales. */
$c_min = rand(120, 185);
$c_max = rand(195, 280);

```

Pour tracer les lignes verticales, on part du côté gauche et on choisit une épaisseur et un léger décalage pour déplacer la ligne. Puis, on choisit une couleur (entre les limites que l'on vient de définir), on trace la ligne comme un polygone et on avance vers la droite avec une progression aléatoire :

```

/* Trace des lignes verticales aléatoire dans la largeur. */
$gauche = 0;
while ($gauche < $largeur) {
    $droite = $gauche + rand(3, 7);
    $offset = rand(-3, 3); /* Offset de l'angle */

    $points_ligne = array(
        $gauche, 0, /* Coin supérieur gauche */
        $droite, 0, /* Coin supérieur droit */
        $droite + $offset, $hauteur, /* Coin inférieur droit */
        $gauche + $offset, $hauteur); /* Coin inférieur gauche */
}

```

```

$pc = imagecolorallocate($img, rand($c_min, $c_max),
                        rand($c_min, $c_max),
                        rand($c_min, $c_max));
imagefilledpolygon($img, $points_ligne, 4, $pc);

/* Avance vers la droite. */
$gauche += rand(20, 60);
}

```

On utilise la même procédure pour créer les lignes horizontales :

```

/* Crée des lignes horizontales aléatoires dans la hauteur. */
$haut = 0;
while ($haut < $hauteur) {
    $bas = $haut + rand(1, 4);
    $offset = rand(-6, 6); /* Offset de l'angle */

    $points_ligne = array(
        0, $haut, /* Coin supérieur gauche */
        0, $bas, /* Coin inférieur gauche */
        $largeur, $bas + $offset, /* Coin inférieur droit */
        $largeur, $haut + $offset); /* Coin supérieur droit */
    $pc = imagecolorallocate($img, rand($c_min, $c_max),
                            rand($c_min, $c_max),
                            rand($c_min, $c_max));
    imagefilledpolygon($img, $points_ligne, 4, $pc);
    $haut += rand(8, 15);
}

```

Nous sommes enfin prêts à produire les caractères de la phrase secrète. Avant de les dessiner, nous devons déterminer grossièrement l'espacement des lettres, puis initialiser une variable qui contiendra la position du caractère le plus à gauche :

```

/* Espacement des caractères. */
$espacement = $largeur / (strlen($mdp)+2);

/* Coordonnée x initiale */
$x = $espacement;

```

On parcourt ensuite tous les caractères en faisant légèrement varier à chaque fois le décalage, l'angle, la taille et la police :

```

/* Dessine chaque caractère. */
for ($i = 0; $i < strlen($mdp); $i++) {
    $lettre = $mdp[$i];
}

```

```

$taille = rand($hauteur/3, $hauteur/2);
$rotation = rand(-30, 30);
/* Position y aléatoire en laissant de la place pour les pattes
des caractères */
$y = rand($hauteur * .90, $hauteur - $taille - 4);
/* Choix d'une police au hasard. */
$police = $polices[array_rand($polices)];

```

Ces lettres seront très difficiles à lire sans une certaine mise en valeur. Vous pouvez créer une ombre colorée en divisant les valeurs de couleurs initiales par 3 :

```

/* Choix d'une couleur pour la lettre. */
$r = rand(100, 255); $g = rand(100, 255); $b = rand(100, 255);
/* Création de la lettre et de l'ombre colorée */
$couleur = imagecolorallocate($img, $r, $g, $b);
$ombre = imagecolorallocate($img, $r/3, $g/3, $b/3);

```

Pour dessiner l'ombre puis la lettre, on utilise `imagefttext()` puis on passe au caractère suivant :

```

/* Dessine l'ombre, puis la lettre. */
imagefttext($img, $taille, $rotation, $x, $y, $ombre, $police,
    $lettre);
imagefttext($img, $taille, $rotation, $x-1, $y-3, $couleur, $police,
    $lettre);

/* Avance sur le canevas. */
$x += rand($espacement, $espacement * 1.5);
}

```

Lorsque cette boucle s'est terminée, vous êtes prêt à envoyer l'image au client en appelant la fonction `imagejpeg()`, puis on libère la mémoire qu'elle occupait avec `imagedestroy()` :

```

imagejpeg($img); /* Envoie l'image. */
imagedestroy($img); /* Libère la mémoire de l'image. */
?>

```

L'autre partie du système CAPTCHA est un petit fragment de code pour afficher le formulaire, intégrer l'image produite par le script précédent et vérifier que la phrase saisie dans le formulaire est celle créée par le générateur d'image. Ce dernier ayant fait tout le travail, le reste n'appelle pas de commentaires.

```

<?php

session_start();

/* Recherche un mot de passe soumis. */
if ($_REQUEST["tt_pass"]) {
    if ($_REQUEST["tt_pass"] == $_SESSION["tt_pass"]) {
        echo "Phrase secrète correcte.";
    } else {
        echo "Phrase secrète incorrecte.";
    }
    exit(0);
}

/* Par défaut, on envoie le formulaire. */

print '<form action="' . $_SERVER['PHP_SELF'] . '" method="post">';
?>
Saisissez les caractères suivants pour continuer :<br />
(si vous n'arrivez pas à les lire, réactualisez la page)<br />
<br /><br />
Lettres : <input name="tt_pass" type="text" size="10" maxlength="10">
<input type="submit">
</form>

```

Évidemment, vous devrez incorporer ce script un peu plus soigneusement dans votre code, mais l'un des points les plus positifs de ce système est qu'il est relativement autonome. Vous pouvez faire en sorte de le renforcer un peu plus, en supprimant la phrase secrète une fois qu'elle a été vérifiée, par exemple (afin qu'elle ne puisse servir qu'une fois). Cependant, si vous pensez que vous avez besoin de beaucoup plus, votre problème est plus important et vous devrez le résoudre avec un système d'authentification.

Recette 67 : Créer des vignettes

Si vous autorisez les utilisateurs à déposer des images sur votre site (voir la recette n°54, "Déposer des images dans un répertoire"), vous aurez probablement besoin de petites images (appelées encore imagettes ou "thumbnails" en anglais) pour créer des pages de prévisualisation et faciliter la navigation dans les galeries. Cette section présente la fonction `mkthumb()` permettant de créer ces vignettes avec GD. Pour l'utiliser, vous aurez besoin des éléments suivants :

- Un répertoire accessible en écriture pour sauvegarder les vignettes (si vous ne savez pas créer ce répertoire, reportez-vous à la section "Permissions des fichiers").
- La bibliothèque GD.

La fonction `mkthumb()` prend deux paramètres : le nom du fichier image et le nom de la vignette. Vous devrez vérifier séparément que le fichier image existe et que son nom porte une extension `.jpg`, `.gif` ou `.png`. Si ce fichier n'existe pas encore, consultez la recette n°54, "Déposer des images dans un répertoire").

La fonction commence par définir des valeurs par défaut pour la largeur et la hauteur maximales des vignettes (ici, elles doivent être égales) :

```
<?php

function mkthumb($nom_fic, $nom_vignette) {
    /* Création d'une vignette. */
    $largeur_vignette = 125;
    $hauteur_vignette = $largeur_vignette;
```

On charge ensuite l'image en mémoire d'après son extension en utilisant les fonctions `imagecreatefromformat()` de GD qui renvoient un descripteur d'image. Il faut tout de suite noter, qu'ici, il n'y a aucune vérification d'erreur car `mkthumb()` suppose que vous avez rempli tous les prérequis. Si vous voulez ajouter ce contrôle des erreurs, il suffit de vérifier que `$image_src` existe après l'appel.

```
if (preg_match('/\.(gif$/i', $nom_fic)) {
    $image_src = imagecreatefromgif($nom_fic);
} else if (preg_match('/\.(png$/i', $nom_fic)) {
    $image_src = imagecreatefrompng($nom_fic);
} else {
    /* Part du principe qu'il s'agit d'une image JPEG par défaut */
    $image_src = imagecreatefromjpeg($nom_fic);
}
```

Puis, on extrait la largeur et la hauteur de l'image avec les fonctions `imagesx()` et `imagesy()`:

```
$largeur = imagesx($image_src);
$hauteur = imagesy($image_src);
```

On ne modifie la taille de l'image que si les dimensions de l'image originale sont supérieures aux dimensions maximales des vignettes :

```
if (($hauteur > $hauteur_vignette) || ($largeur > $largeur_vignette)) {
```

Pour modifier la taille d'une image, on a besoin de connaître ses dimensions finales. On ne peut pas utiliser la largeur et la hauteur maximale des vignettes car

l'image initiale n'est peut-être pas carrée : si l'on tentait d'imposer de telles dimensions à une image rectangulaire, elle apparaîtrait déformée . Pour résoudre ce problème, on recherche donc le plus long côté de l'image initiale et on utilise un rapport d'échelle égal au rapport entre la longueur du plus grand côté et celle du côté de la vignette :

```
/* Création d'une vignette. */
if ($largeur > $hauteur) {
    $ratio = $largeur_vignette / $largeur;
} else {
    $ratio = $hauteur_vignette / $hauteur;
}
```

ATTENTION *La raison pour laquelle cette fonction utilise des dimensions de vignettes carrées est que le raisonnement précédent ne marche pas pour des rectangles génériques. Si vous essayez de transformer une image de 200 par 400 pixels en une vignette de 100 par 400, \$ratio vaudra 1 et vous obtiendrez une vignette de 200 par 400 (voir le code qui suit). Cette taille est donc supérieure à la taille maximale d'une vignette. Corriger ce problème n'est pas compliqué, mais nous le laissons en exercice au lecteur.*

On utilise le ratio d'échelle pour trouver les dimensions exactes de la vignette en pixels, puis on crée un nouveau descripteur pour une image avec cette nouvelle taille :

```
$nouv_largeur = round($largeur * $ratio);
$nouv_hauteur = round($hauteur * $ratio);
$image_dest = ImageCreateTrueColor($nouv_largeur, $nouv_hauteur);
```

On peut alors créer la nouvelle image avec cette nouvelle taille grâce à la fonction `imagecopyresampled()` :

```
imagecopyresampled($image_dest, $image_src, 0, 0, 0, 0,
                  $nouv_largeur, $nouv_hauteur, $largeur, $hauteur);
```

L'image initiale en mémoire ne servant plus à rien, on libère l'espace qu'elle occupe :

```
imagedestroy($image_src);
```

`$image_dest` contient maintenant les données de l'image redimensionnée, prête à être affichée. Si l'on n'a pas eu besoin de modifier la taille, on échoue dans la clause `else` suivante, qui indique que l'on peut affecter `$image_dest` avec le descripteur de l'image initiale :

```
} else {
  /* L'image est déjà suffisamment petite... On la renvoie simplement. */
  $image_dest = $image_src;
}
```

On sait maintenant que l'on peut afficher `$image_dest` ; il reste à l'écrire dans le fichier passé comme deuxième paramètre et à libérer la mémoire qu'elle occupe :

```
imagejpeg($image_dest, $nom_vignette);
imagedestroy($image_dest);
}
?>
```

Voici un script très simple qui montre comment utiliser la fonction `mkthumb()` :

```
<?php
include("mkthumb.inc");
$fichier = $_FILES["fichier"]["tmp_name"];
$fnn = $_FILES["fichier"]["name"];
$nom_vignette = "vignettes/$fnn";

if ($fichier) {
  mkthumb($fichier, $nom_vignette);
  print "<img src=\"\$nom_vignette\" />";
} else { ?>
  <form action="thumb.php" enctype="multipart/form-data" method="post">
  Déposer une image :<br />
  <input name="fichier" type="file" /><br />
  <input name="Submit" type="submit" value="Déposer" />
  <?
}
?>
```

Ce script fait évidemment *un très grand nombre* de suppositions ; ne l'utilisez jamais en production ! La recette n°54, "Déposer des images dans un répertoire" présente, par contre, un script de dépôt d'images complet.

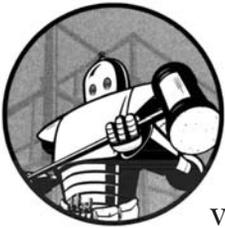
La fonction `mkthumb()` peut être optimisée de différentes façons pour s'adapter à vos besoins particuliers. JPEG est un format avec perte, par exemple : si vous créez une vignette d'un fichier GIF ou PNG, celle-ci sera de mauvaise qualité. Pour corriger ce problème, vous pouvez choisir de créer la vignette au format PNG en appelant la fonction `imagepng()` au lieu de `imagejpeg()`. Vous pourriez également essayer de ruser en modifiant la taille de l'image initiale. Malheureusement, cela risque de se retourner contre vous car les images GIF ont un ensemble

de couleurs limité et, lorsque vous modifiez la taille d'une image, les couleurs ont tendance à changer également.

De toutes façons, évitez les astuces de manipulation des images en PHP : les scripts sont généralement utilisés à la demande et vous avez donc peu de temps pour effectuer des opérations amusantes mais coûteuses en temps d'exécution. Non seulement vos utilisateurs s'impatienteraient, mais le serveur web ne laisserait pas non plus votre script s'exécuter assez longtemps.

11

UTILISATION DE cURL POUR LES SERVICES WEB



Internet regorge d'informations très intéressantes : UPS peut vous dire le prix exact du transport d'un paquet de 3 kg de Bâton Rouge en Louisiane vers Toulouse en France. Authorize.net peut vous dire s'il reste suffisamment d'argent sur le compte d'un client pour qu'il puisse acheter un livre qui coûte 50 € sur votre site. Cependant, vous devez savoir comment demander ces informations.

Pour les trouver, vous devriez toujours penser à la bibliothèque cURL de PHP pour gérer la connexion entre votre serveur web et les autres. Le principe consiste à considérer que vos scripts sont des clients, un peu comme des navigateurs web. Vous pouvez demander à cURL tout ce qui va de la récupération du code HTML d'une page web à l'accès à un service web reposant sur XML. Il y a trois façons d'accéder aux données :

- On télécharge la page web du site pour décortiquer son code HTML, comme on l'a expliqué au Chapitre 5, à la recette n°41, "Extraire des données des pages".
- On poste des paramètres de requêtes et on fait le tri dans le résultat.

- On utilise l'API d'un service web pour accéder aux données et on analyse le résultat avec un parseur XML. Vous rencontrerez différents protocoles comme SOAP (*Simple Object Access Protocol*) ou REST (*REpresentational State Transfer*; c'est généralement une autre façon d'appeler l'envoi de paramètres POST ou GET). N'attendez pas trop de cohérence : même si deux sites fournissent le même type de données, il est fort probable que leurs formats de sortie et leurs méthodes d'accès soient différents.

Selon ce que vous comptez faire, vous aurez également besoin d'une partie des bibliothèques suivantes :

1. cURL (la bibliothèque et l'extension PHP). Voyez la recette n°28, "Ajouter des extensions à PHP", si cette extension n'est pas déjà installée.
2. OpenSSL pour accéder aux sites sécurisés.
3. XML pour analyser les données provenant des services web. Les extensions XML (telles libXML) sont installées par défaut sur la plupart des serveurs PHP.

Recette 68 : Se connecter à d'autres sites web

Pour illustrer l'utilisation des fonctions cURL de PHP, voyons comment nous connecter à une page web pour récupérer ses données. On crée d'abord un descripteur de connexion cURL en appelant la fonction `curl_init()` :

```
$c = curl_init();
```

On utilise ensuite la fonction `curl_setopt()` pour préciser les options de connexion, dont la plus importante est l'URL cible. L'appel est de la forme suivante (`CURLOPT_URL` est le nom de l'option et le dernier paramètre est l'URL cible) :

```
curl_setopt($c, CURLOPT_URL, "http://www.google.fr/");
```

Par défaut, cURL active certaines fonctionnalités qui, soit sont inutiles, soit gênent un traitement de page classique. L'une d'elles consiste à inclure l'en-tête HTTP dans le résultat ; vous pouvez la désactiver de la façon suivante :

```
curl_setopt($c, CURLOPT_HEADER, false);
```

De même, cURL affiche automatiquement la page accédée au lieu de la renvoyer sous forme de chaîne. Comme vous avez besoin d'une chaîne pour analyser le contenu de la page, utilisez l'option `CURLOPT_RETURNTRANSFER` pour indiquer que vous voulez obtenir le résultat sous cette forme :

```
curl_setopt($c, CURLOPT_RETURNTRANSFER, true);
```

Nous sommes maintenant prêt à accéder à la page grâce à la fonction `curl_exec()` :

```
$donnees_page = curl_exec($c);
```

Avec cet appel, cURL accède à la page et renvoie les données qu'elle contient ; celles-ci sont alors affectées à la variable `$donnees_page`. Enfin, il ne reste plus qu'à fermer la connexion avec `curl_close()` pour libérer la ressource :

```
curl_close($c);
```

C'est un bon début car nous pouvons maintenant accéder à des données web *via* la méthode GET. Par contre, pour soumettre des données avec la méthode POST, nous devons configurer des options supplémentaires avec `curl_setopt()`. Nous allons donc écrire une fonction `recup_page()` permettant d'accéder aux pages avec la méthode GET ou POST.

Cette fonction attend deux paramètres : l'URL cible et un tableau facultatif de paramètres POST (les clés seront les noms des paramètres et seront associées à leurs valeurs). Si ce second paramètre n'est pas fourni à l'appel, la fonction utilise la méthode GET. La première partie de son code consiste à construire une chaîne de requête à partir du tableau des paramètres, de la forme `param1=valeur1¶m2=valeur2[...]`, exactement comme pour une requête GET sauf qu'elle ne commence pas par un point d'interrogation. La fonction prédéfinie `http_build_query()` permettant de créer ce type de chaîne à partir d'un tableau, il ne nous reste plus qu'à vérifier le paramètre tableau :

```
function recup_page($url, $params_post = null) {
    /* Connexion à un site avec POST ou GET et récupération des données */
    $chaine_requete = null;
    if (!is_null($params_post)) {
        if (!is_array($params_post)) {
            die("Les paramètres POST ne sont pas sous forme de tableau.");
        }
        /* Construction de la chaîne de requête. */
        $chaine_requete = http_build_query($params_post);
    }
}
```

On peut maintenant configurer le descripteur de connexion cURL. S'il existe une chaîne de requête à ce stade, on sait que c'est parce qu'on utilise la méthode POST ; on met donc en place la connexion en utilisant cette chaîne comme données POST :

```
$ch = curl_init();
if ($chaine_requete) {
```

```
curl_setopt($ch, CURLOPT_POST, true);
curl_setopt($ch, CURLOPT_POSTFIELDS, $chaine_requete);
}
```

À partir de maintenant, on configure la connexion comme on l'a fait plus haut, on exécute la requête, puis l'on renvoie les données :

```
curl_setopt($ch, CURLOPT_URL, $url);
curl_setopt($ch, CURLOPT_HEADER, false);
curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);
$donnees_renvoyees = curl_exec($ch);
curl_close($ch);
return $donnees_renvoyees;
}
```

Voici comme utiliser cette fonction pour faire une recherche sur Yahoo! :

```
print recup_page("http://search.yahoo.com/search",
                array("p" => "vache"));
```

Cette fonction convient à la plupart des besoins classiques des accès clients. Nous allons maintenant montrer comment satisfaire les exigences de certains sites et services.

Recette 69 : Utiliser les cookies

Si vous devez vous connecter à un serveur qui utilise une authentification par cookie, il faut mettre en place dans cURL un système de récupération des cookies. Le principe consiste à faire en sorte que cURL stocke dans un fichier (un "pot à cookies") les cookies qu'il reçoit lors de l'accès à une page et, lors des accès suivants, qu'il recherche dans ce fichier les cookies qu'il doit envoyer au serveur.

Pour cela, vous devez typiquement ajouter deux lignes de configuration : la première pour définir l'emplacement où écrire, la seconde pour préciser l'emplacement où lire :

```
curl_setopt(c, CURLOPT_COOKIEJAR, 'pot_cookies');
curl_setopt(c, CURLOPT_COOKIEFILE, 'pot_cookies');
```

Ici, *c* est un descripteur de connexion cURL et *pot_cookies* est un fichier accessible en écriture.

Le plus grand inconvénient de cette approche est qu'elle ne fonctionne pas avec les connexions en parallèle ; si plusieurs processus utilisent le même pot à cookies, ils auront la même session et se marcheront dessus les uns et les autres, voire pire. Vous pouvez contourner ce défaut en insérant le PID du processus

PHP (avec `getmypid()`) dans le nom du fichier du pot à cookies. Cependant, vous devez alors vous assurer de supprimer le fichier en question lorsque vous avez terminé, ou vous finirez par avoir un grand nombre de pots à cookies qui risquent de perturber les processus suivants.

Recette 70 : Transformer du XML sous une forme utilisable

XML (*eXtensible Markup Language*) est l'un des langages à balises les plus connus. Il est conçu pour fournir des formats d'échange standard pour tout type de données. Bien que XML soit lourd et terriblement inefficace, la plupart des services web l'utilisent en format de sortie et certains l'exigent même comme format d'entrée. Vous devrez donc sûrement le traiter à un moment ou à un autre.

Le meilleur moyen de commencer avec XML consiste à analyser un fragment de document, c'est-à-dire à vérifier que les données sont du XML valide, puis à les examiner. Vous pensiez qu'avec tout le bruit qui a accompagné l'introduction de XML quelqu'un aurait déjà trouvé un moyen de le traiter simplement ? Malheureusement, depuis des années, l'accès aux données XML est redoutablement compliqué ; des milliers de systèmes d'analyse ont vu le jour, comme DOM et SAX. Rien d'étonnant à ce que vous soyez perdu parmi les 27 000 extensions PHP consacrées à XML.

La bonne nouvelle est que l'on s'est désormais rendu compte que, la plupart du temps, on se contentait d'analyser les données XML comme un arbre formé d'un grand nombre de tableaux imbriqués afin que les programmeurs puissent utiliser des outils d'accès aux données normaux, comme les itérateurs et les indices. Une nouvelle génération d'analyseurs est donc apparue dans ce but et PHP dispose d'une telle extension : SimpleXML.

Voici un exemple de document XML permettant d'introduire SimpleXML :

```
<?xml version="1.0" encoding="utf-8"?>
<sermon>
  <peches>
    <peche type="mortel">gourmandise</peche>
    <peche type="mineur">mauvais jeu de mots</peche>
    <peche type="nécessaire">flatulence</peche>
  </peches>
</sermon>
```

Ce fragment contenant des nœuds imbriqués, des données et des attributs, il couvre à peu près tout ce que vous rencontrerez. Pour l'analyser avec SimpleXML, il suffit de placer les données dans une chaîne et de s'en servir pour créer un objet SimpleXMLElement :

```
$xs = file_get_contents("test.xml");
$donnees = new SimpleXMLElement($xs);
```

NOTE Au lieu de passer par ces deux étapes, vous pouvez aussi créer l'objet à partir d'un fichier en appelant `simplexml_load_file()`.

Si aucun message d'erreur n'est produit, le document XML se trouve maintenant dans l'objet `$donnees`. Pour examiner le premier péché du nœud `<peches>`, on utilise une combinaison de syntaxe orientée objet et d'accès aux tableaux :

```
print $donnees->peches->peche[0];
```

Cela affichera *gourmandise* ; si vous voulez accéder au nœud *mauvais jeu de mots*, il faut examiner `$donnees->peches->peche[1]`. Si vous n'utilisez pas d'indice (`$donnees->peches->peche`), vous obtiendrez le premier élément du tableau, ce qui est pratique si vous savez qu'il n'y a qu'un seul élément.

Vous pouvez également examiner les attributs d'un nœud en utilisant des indices, sauf que ces derniers sont désormais des chaînes et non des nombres. Pour, par exemple, connaître le type de péché correspondant à *mauvais jeu de mots*, il suffit d'écrire :

```
print $donnees->peches->peche[1]["type"];
```

Le plus grand intérêt de la syntaxe d'accès aux tableaux est qu'elle permet de parcourir tous les nœuds. Voici, par exemple, comment afficher tous les péchés de notre document :

```
foreach ($donnees->peches->peche as $peche) {
    print $peche . " : " . $peche["type"];
    print "<br />";
}
```

Comme vous l'avez sûrement remarqué, il est possible de confondre les attributs et les nœuds fils avec cette syntaxe mais, en général, ce n'est pas trop grave. Vous pouvez même utiliser `print_r()` pour tout afficher, auquel cas vous obtiendrez le résultat suivant :

```
SimpleXMLElement Object
(
    [peches] => SimpleXMLElement Object
    (
        [peche] => Array
        (
            [0] => gourmandise
            [1] => mauvais jeu de mots
            [2] => flatulence
        )
    )
)
```

SimpleXML peut faire beaucoup plus, notamment créer et modifier un document XML, mais ce que nous avons présenté ici est suffisant pour commencer à travailler avec les services web. Dans les sections suivantes, nous allons étudier quelques applications réelles.

Recette 71 : Utiliser des services web de localisation géographique

Maintenant que vous savez accéder à une URL et analyser un document XML, il est temps de les combiner. Nous allons prendre le service de géolocalisation de Yahoo! comme exemple. L'API est un service REST que l'on peut accéder via des paramètres GET. À l'aide de la fonction `recup_page()` de la recette n°68, "Se connecter à d'autres sites web", voici comment obtenir des informations sur l'adresse "47bis rue des Vinaigriers 75010 Paris".

```
$requete = http_build_query(array(
    "appid" => "YahooDemo",
    "street" => "47bis rue des Vinaigriers",
    "city" => "Paris",
));
$page = recup_page("http://local.yahooapis.com/MapsService/V1/geocode?$requete");
```

NOTE *Dans l'idéal, vous remplacerez YahooDemo par votre identifiant Yahoo! pour cette application, bien que vous pouvez la laisser telle quelle si vous ne comptez pas utiliser cette application en production ni trop souvent. Utilisez éventuellement les paramètres "state" et "zip" si vous recherchez une adresse américaine. Vous avez également la possibilité d'utiliser le paramètre "country" afin de préciser qu'il s'agit d'une adresse en France, mais la requête fonctionne en l'état dans notre exemple précédent – il n'existe pas d'homonyme. Comme vous le verrez par la suite, le document XML renvoyé se complète automatiquement et mentionne "France" dans la balise <state> et "FR" dans <country>. Vous avez donc parfaitement le droit d'ajouter "country" => "FR" dans la requête précédente, afin d'éliminer le moindre doute !*

Après l'exécution de ce code, `$page` contient le document XML suivant :

```
<?xml version="1.0"?>
<ResultSet xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="urn:yahoo:maps" xsi:schemaLocation="urn:yahoo:maps http://
api.local.yahoo.com/MapsService/V1/GeocodeResponse.xsd">
  <Result precision="address">
    <Latitude>48.873032</Latitude>
```

```
<Longitude>2.360605</Longitude>
<Address>47bis, rue des Vinaigriers</Address>
<City>75010 Paris</City>
<State>France</State>
<Zip/>
<Country>FR</Country>
</Result>
</ResultSet>
```

Pour extraire la latitude et la longitude d'une adresse, il suffit donc d'utiliser ce code PHP :

```
$donnees = new SimpleXMLElement($page);
$lat1 = $donnees->Result->Latitude[0];
$lon1 = $donnees->Result->Longitude[0];
```

Si vous ne vouliez connaître que ces deux informations, vous avez terminé l'utilisation du service web. Faisons maintenant quelque chose d'un peu plus amusant. Supposons d'abord que le script s'appelle *demo_carte.php* et qu'il est exécuté à la réception de ce formulaire :

```
<form action="demo_carte.php">
Rue : <input type="text" name="rue" /><br />
Ville : <input type="text" name="ville" /><br />
État/Pays : <input type="text" name="etat" /><br />
<input type="submit" /><br >
</form>
```

Ajoutez les lignes suivantes à *carte_demo.php* afin d'extraire un second emplacement géographique correspondant à l'adresse qui sera saisie dans ce formulaire :

```
$requete = http_build_query(array(
    "appid" => "YahooDemo",
    "street" => $_REQUEST["rue"],
    "city" => $_REQUEST["ville"],
    "state" => $_REQUEST["etat"],
));
$page = recup_page("http://local.yahooapis.com/MapsService/V1/geocode?$requete");
$donnees = new SimpleXMLElement($page);
$lat2 = $donnees->Result->Latitude[0];
$lon2 = $donnees->Result->Longitude[0];
```

Représentons maintenant un point sur une carte à l'aide de l'API de Google Maps, mais en utilisant les données que nous venons de récupérer du service Yahoo!. On commence par fermer le bloc PHP et on met en place une carte Google avec ce code HTML et JavaScript (remplacez *votre_cle* par votre identifiant Google Maps). Si vous ne connaissez pas JavaScript, ne vous inquiétez pas car ce code est très classique :

```
?>
<!DOCTYPE html "-//W3C//DTD XHTML 1.0 Strict//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <meta http-equiv="content-type" content="text/html; charset=utf-8"/>
    <title>Exemple de carte</title>
    <script src="http://maps.google.com/
maps?file=api&v=2&key=votre_cle"
        type="text/JavaScript"></script>

    <script type="text/JavaScript">
      function initialize() {
        if (GBrowserIsCompatible()) {
          var map = new GMap2(document.getElementById("canevas_carte"));
```

Puis, créez les points sur la carte Google correspondant aux deux emplacements extraits précédemment et centrez la carte sur le premier (ceci avant toute opération) :

```
var latlon1 = new GLatLng(<?php print "$lat1, $lon1"; ?>);
var latlon2 = new GLatLng(<?php print "$lat2, $lon2"; ?>);
map.setCenter(latlon1);
```

Placez des marqueurs sur les emplacements :

```
map.addOverlay(new GMarker(latlon1));
map.addOverlay(new GMarker(latlon2));
```

Tracez une ligne entre les points :

```
var ligne = new GPolyline([latlon1, latlon2], "#3333aa", 5);
map.addOverlay(ligne);
```

Puis recentrez la carte et zoomez pour que la ligne occupe toute la vue :

```
var limites = line.getBounds();
niveau = map.getBoundsZoomLevel(limites);
map.setCenter(limites.getCenter(), niveau);
```

Enfin, ajoutez un panneau de contrôle (pour le zoom et le défilement) et enveloppez le tout dans du code HTML pour afficher la carte :

```
map.addControl(new GLargeMapControl());
}
}
</script>
</head>
<body onload="initialize()" onunload="GUnload()">
  <div id="canevas_carte" style="width: 500px; height: 300px"></div>
</body>
</html>
```

Vous disposez donc désormais d'une application simple permettant de tracer une ligne sur Google Maps entre les bureaux de Pearson Education France et l'adresse de votre choix. En lui-même, ce script n'est qu'un jouet (notamment parce que vous pouvez obtenir les informations de géolocalisation avec Google au lieu de Yahoo!), mais c'est un point de départ qui n'attend que votre imagination pour produire de véritables applications.

Nous pourrions continuer à vous présenter des services web et vous montrer comment autoriser des cartes de crédits et se faire payer les frais d'expédition, mais tout cela revient toujours au même : vous récupérez des données auprès du client, vous les empaquetez pour le serveur, vous envoyez la requête et vous obtenez le résultat ; d'ailleurs, dans la plupart des cas, vous trouverez probablement du code PHP qui fait le travail pour vous. Il nous reste donc à présenter SOAP, un protocole très important car il est utilisé par de nombreux services web.

Recette 72 : Interroger Amazon avec PHP et SOAP

SOAP (*Simple Object Access Protocol*) est un service web standard et complet qui, comme tout ce qui comporte *simple* dans son nom est tout sauf simple. Le principe consiste à placer tous les détails sur l'entrée et la sortie d'un service web dans un document WSDL (*Web Services Description Language*) afin de pouvoir produire automatiquement une interface de programmation et appeler un service web exactement comme une méthode ou une fonction. Vous n'avez pas besoin de vous occuper de cURL, de construire des requêtes, etc. Il suffit de créer un objet avec votre requête, de demander à PHP de créer l'interface, d'appeler la méthode et vous obtiendrez un objet contenant plein de bonnes choses.

En fait, quand tout fonctionne correctement, c'est aussi simple que cela et le script de cette section vous mettra le pied à l'étrier.

PHP 5 contient un ensemble de classes SOAP prédéfinies, dont `SoapClient` que nous présenterons bientôt. Cependant, ces classes ne sont généralement pas construites par défaut : il faut utiliser l'option `--enable--soap` lors de la configuration et de la compilation de PHP (voyez le Chapitre 2 pour plus de précisions sur cette étape).

Notre exemple repose sur le service web "Associates" offert par Amazon pour vous aider à vendre des articles sur son site. Il faut seulement demander un identifiant et vous avez ensuite un accès gratuit au service (à raison d'une requête par seconde). On commence par créer une instance de `SoapClient` en indiquant l'emplacement du document WSDL :

```
<?php
$client = new SoapClient("http://webservices.amazon.com/AWSECommerceService/
AWSECommerceService.wsdl");
```

Il faut ensuite configurer un objet pour la requête que l'on veut envoyer. Ici, on paramètre un objet `$recherche` contenant des nœuds pour l'identifiant d'accès, la catégorie et les mots-clés d'une recherche :

```
$recherche-->AWSAccessKeyId = "votre_cle";
$recherche-->Request-->SearchIndex = "Music";
$recherche-->Request-->Keywords = "James Blunt";
```

Pour lancer cette recherche, il suffit d'appeler la méthode `itemSearch()` du client :

```
$r = $client->itemSearch($recherche);
```

Si tout s'est bien passé, `$r` contiendra le résultat XML que vous pourrez consulter comme une instance `SimpleXML` :

```
foreach ($r->Items->Item as $elt) {
    $attributs = $elt->ItemAttributes;
    if (is_array($attributs->Artist)) {
        $artiste = implode($attributs->Artist, ", ");
    } else {
        $artiste = $attributs->Artist;
    }
    print "Artiste : $artiste ; titre : $attributs->Title<br />";
}
?>
```

C'est donc très simple si vous savez comment tout cela fonctionne. Malheureusement, la plupart du temps ce ne sera pas le cas. Les fichiers WSDL sont souvent fournis sans aucune documentation et, en supposant que vous lisez du WDSL, vous pourrez retrouver toutes les méthodes disponibles et les paramètres qu'elles attendent, mais vous ne saurez pas à quoi servent ces paramètres. En outre, même si vous savez à quoi ressembleront les réponses aux méthodes, vous ne saurez même pas ce que sont censées faire ces méthodes exactement. Cela est dû en partie au fait qu'il est difficile de tirer parti d'une documentation lorsqu'on ne connaît pas le langage de programmation ou l'implémentation SOAP utilisée. De plus, personne ne *veut* les documenter.

Un autre problème sérieux concerne les performances. Dans l'exemple précédent, PHP doit trouver le fichier WSDL, analyser son contenu XML puis créer une toute nouvelle classe et une instance à partir de ce contenu. Tout ceci est très inefficace, surtout si l'on considère que l'on a simplement besoin d'envoyer quelques petits paramètres à un service. PHP met en cache les fichiers WSDL qu'il rencontre (d'autres langages peuvent exiger que vous produisiez un code compilé à partir de WSDL, ce qui n'est pas beaucoup plus efficace). En conséquence, si vous comptez lancer immédiatement un certain nombre de requêtes pour des services web reposant sur WSDL, vous aurez besoin soit d'un matériel puissant, soit d'un moyen de contourner une bonne partie de ce processus en construisant manuellement les requêtes SOAP.

Pour toutes ces raisons, les services web SOAP ne sont pas aussi répandus qu'on pourrait le penser. Celui d'Amazon.com, par exemple, n'a été ajouté que récemment et cohabite avec un service REST traditionnel. Amazon.com documente les paramètres REST mais vous renvoie à la lecture du fichier WSDL pour savoir comment faire en SOAP. Une autre société du nom de Google a décidé d'abandonner SOAP pour son service de recherche web. Cependant, SOAP n'est pas mort : vous le rencontrerez sûrement lorsque vous aurez à faire à des services utilisant l'architecture .NET de Microsoft.

Recette 73 : Construire un service web

Pour conclure ce chapitre, voyons rapidement comment construire un service web REST. Le principe ressemble beaucoup à celui des autres types de pages dynamiques mais, au lieu de se soucier de l'aspect de la page, on ne s'occupe que de créer correctement un objet, de le transformer en XML et d'afficher le résultat. Le reste n'est plus notre problème (du moins, en théorie).

Nous utiliserons les données de la table SQL fournie en annexe. Le service web prend un paramètre (GET ou POST) représentant une catégorie de produit et renvoie une liste d'articles sous la forme suivante :

```
<?xml version="1.0" encoding="utf-8"?>
<Articles>
  <Article>
    <Nom>Bottes Western</Nom>
```

```
<ID>12</ID>
<Prix>19.99</Prix>
</Article>
<Article>
  <Nom>Pantoufles</Nom>
  <ID>17</ID>
  <Prix>9.99</Prix>
</Article>
</Articles>
```

Il existe de nombreuses façons de créer du XML en PHP, allant de DOM aux fonctions `xmlwriter`. Ici, nous utiliserons la classe `SimpleXML` que l'on a déjà présentée dans ce chapitre. La première étape consiste à initialiser la connexion avec la base de données, à indiquer au client qu'il va recevoir du XML et à vérifier le paramètre catégorie. C'est un traitement classique et vous avez maintenant l'habitude de ce genre de code (ne vous occupez pas de l'appel à `affiche_erreur()`, nous y reviendrons plus tard) :

```
<?php
$dbd = mysql_connect("localhost", "nom_utilisateur", "secret");
mysql_select_db("nom_base");

header("Content-type: text/xml");
$categorie = $_REQUEST["categorie"];

if ($categorie) {
    $resultat = mysql_query("SELECT DISTINCT categorie
                            FROM infos_produits", $dbd);
    $categories = array();
    while ($ligne = mysql_fetch_array($resultat)) {
        $categories[] = $ligne["categorie"];
    }
    if (!in_array($categorie, $categories)) {
        affiche_erreur("Catégorie non reconnue. ");
        exit;
    }
}
```

NOTE *Techniquement, il n'est pas nécessaire de faire tout ce traitement pour trouver les noms de catégories possibles car vous pourriez les mettre en cache (et vous devriez le faire si vous les utilisez souvent).*

Récupérons maintenant les informations sur les produits à partir de la base de données :

```
$resultat = mysql_query("SELECT nom_produit, num_produit, prix
                        FROM infos_produits
                        WHERE categorie = '$categorie'", $bd);
```

Les choses vont maintenant commencer à devenir intéressantes. Pour initialiser le document XML, on crée une instance de `SimpleXMLElement` à partir d'un document existant : libre à vous de le personnaliser et de le rendre éventuellement plus complexe. Notre structure XML de base étant très simple, nous utilisons une chaîne en ligne :

```
$doc = new SimpleXMLElement('<?xml version="1.0" encoding="utf-8"?>
                            <Articles></Articles>
                            ');
```

Le document est prêt à être rempli et les données se trouvent dans `$resultat` : il reste donc à parcourir les lignes. La première opération consiste à ajouter un nouveau nœud fils `Article` au nœud `Articles` avec la méthode `$doc->addChild()` :

```
while ($produit = mysql_fetch_array($resultat)) {
    $fils = $doc->addChild("Article");
```

Le premier fils est maintenant accessible par `$doc->Article[0]`, mais il est plus simple d'utiliser le descripteur renvoyé par la méthode `addChild()`. Ce nouveau fils a maintenant besoin de ses propres nœuds fils `Nom`, `ID` et `Prix` que nous créerons avec la méthode `addChild()` appliquée cette fois-ci à `$fils` au lieu de `$doc`. Ces nouveaux nœuds ayant des valeurs, nous les fournissons en deuxième paramètre :

```
$fils->addChild("Nom", $produit["nom_produit"]);
$fils->addChild("ID", $produit["num_produit"]);
$fils->addChild("Prix", $produit["prix"]);
}
```

Après avoir traité tous les articles, il reste à produire le document XML en appelant la méthode `$doc->asXML()` :

```
print $doc->asXML();
```

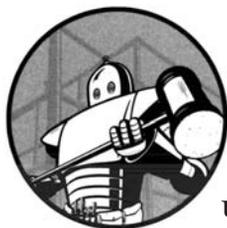
En fait, vous n'avez pas tout à fait encore terminé car il faut vous occuper de la fonction `affiche_erreur()` que nous avons mentionnée plus haut. Comme nous supposons que le message d'erreur est très simple, nous n'avons pas à nous occuper des objets :

```
function affiche_erreur($message) {
    $message = htmlentities($message);
    print "<?xml version=\"1.0\" encoding=\"utf-8\"?>
        <Erreur>$message</Erreur>
        ";
}
?>
```

Nous en avons terminé avec les services web. Il y a bien sûr des milliers de façons de les compliquer – en utilisant SOAP et WSDL notamment – mais, en définitive, tout cela revient à rassembler des données et à les placer dans les bonnes cases.

12

MISE EN APPLICATION



Ce dernier chapitre contient trois ensembles de scripts qui implémentent des fonctionnalités classiques que l'on trouve sur de nombreux sites web diffuseurs de contenus : un système de sondage, un service de carte postale électronique et un blog.

Ce ne sont que des points de départ : bien qu'ils fonctionnent parfaitement, vous devrez les adapter à vos besoins et à votre politique de sécurité avant de les utiliser en production.

Tous ces projets utilisent MySQL pour stocker leurs données. La structure des tables et les requêtes utilisées ici sont un peu plus compliquées que celles que vous avez déjà rencontrées dans ce livre, mais ce n'est pas infranchissable. Chaque système utilise ses propres tables et nous vous expliquerons comment les créer.

Recette 74 : Un système de sondage

Les sondages en ligne ne sont pas très utiles pour justifier quoi que ce soit, mais ce sont des outils pratiques pour savoir comment orienter votre contenu afin que vos visiteurs soient toujours intéressés, d'optimiser les aspects des pages ou d'irriter le plus de personnes possible. L'aspect le plus positif de ces sondages

est qu'ils sont très simples à mettre en œuvre. Celui que nous présentons ici devant permettre d'effectuer des sondages multiples, les questions et les réponses ne sont pas codées en dur.

Pour un sondage en ligne classique, vous n'avez normalement pas à trop vous soucier de tout ce qui concerne les votes car cela a rarement de l'importance. Pour empêcher que l'on puisse voter plusieurs fois, nous utiliserons des cookies. Bien que ce système puisse facilement être contourné, les enjeux sont généralement si peu importants que personne ne s'en soucie et, si vous avez besoin d'un système plus fiable, vous pouvez utiliser un système d'authentification pour enregistrer les votes.

Ce système de sondage comporte quatre scripts :

form_vote.php : Affichage d'un bulletin de vote pour l'utilisateur.

traitement_vote.php : Traitement d'un vote.

affichage_vote.php : Affichage des résultats du sondage.

config_vote.php : Connexion à la base de données.

Il utilise trois tables. La table des sondages contient les questions :

```
CREATE TABLE sondages (  
    ID INT NOT NULL AUTO_INCREMENT ,  
    question MEDIUMTEXT NOT NULL ,  
    PRIMARY KEY ( ID )  
) TYPE = MYISAM ;
```

La table des réponses contient les réponses aux questions de la table des sondages. Le champ ID permet de faire des jointures avec cette dernière :

```
CREATE TABLE reponses (  
    ID_reponse INT NOT NULL AUTO_INCREMENT ,  
    ID INT NOT NULL ,  
    reponse MEDIUMTEXT NOT NULL ,  
    PRIMARY KEY ( ID_reponse )  
) TYPE = MYISAM ;
```

Les champs ID et ID_reponse de la table des votes sont des répliques des champs correspondants dans les tables des sondages et des réponses :

```
CREATE TABLE votes (  
    ID INT NOT NULL ,  
    ID_reponse INT NOT NULL ,  
    INDEX ( ID )  
) TYPE = MYISAM ;
```

Enfin, voici un exemple de question et les lignes des réponses possibles :

```
INSERT INTO sondages (question) VALUES ("Aimez-vous les sandwiches ?");
INSERT INTO reponses (ID, reponse) VALUES (1, "Oui.");
INSERT INTO reponses (ID, reponse) VALUES (1, "Non.");
INSERT INTO reponses (ID, reponse) VALUES (1, "Je ne sais pas.");
```

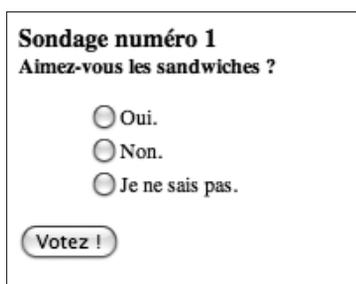
Tous les scripts se connectant à la base de données, il est préférable de placer les détails de connexion dans un fichier *config_vote.php* qui sera inclus par tous les autres :

```
<?php
$dbd = @mysql_connect("localhost", "login_sql", "mdp_sql") or
    die("Échec de la connexion.");
@mysql_select_db("nom_base", $dbd) or
    die("Impossible de se connecter à la base de données.");
?>
```

Étudions maintenant les différents scripts.

Création d'un formulaire pour les bulletins de vote

Le script *formulaire_vote.php* est assez évident : on lui passe un identifiant de sondage dans le paramètre *sondage* et il affiche le bulletin, comme dans la Figure 12.1.



Sondage numéro 1
Aimez-vous les sandwiches ?

Oui.

Non.

Je ne sais pas.

Figure 12.1 : Un bulletin de vote pour un sondage

Il commence par charger la configuration de la base de données et vérifie que l'identifiant de sondage est un entier :

```
<?php
/* Affiche un formulaire pour voter. */
require_once("config_vote.php");
```

```

$sondage = $_GET['sondage'];
if (!is_numeric($sondage)) {
    die("Sondage incorrect");
}

```

Nous pouvons vérifier en une seule requête que l'identifiant du sondage est correct et rechercher les choix à afficher. En effet, si aucun sondage ne correspond à cet identifiant, la requête ne renverra aucune ligne.

```

/* Recherche du sondage dans la base de données. */
$sql = "SELECT S.question, R.reponse, R.ID_reponse
        FROM sondages S, reponses R
        WHERE S.ID = $sondage
        AND R.ID = S.ID";

$resultat = mysql_query($sql, $bd) or
    die ("Erreur mysql : " . mysql_error());
if (mysql_num_rows($resultat) == 0) {
    die('Sondage inconnu.');
```

Si l'identifiant de sondage est reconnu, on doit s'assurer que l'utilisateur n'a pas déjà voté. Comme on l'a expliqué précédemment, nous le vérifierons avec des cookies. On suppose que si le cookie *id_vote* (ou *id* est l'identifiant du sondage) existe, c'est que l'utilisateur a voté, auquel cas on lui envoie le résultat du sondage :

```

/* Si l'utilisateur a déjà voté, on affiche le résultat. */
if ($_COOKIE["${sondage}_vote"]) {
    header("Location: affichage_vote.php?sondage=$sondage");
    exit;
}

```

Si l'on est arrivé ici, c'est que l'utilisateur n'a pas encore voté et il faut donc parcourir la liste des choix pour construire le formulaire. Cette boucle place une suite de boutons radios dans la variable *\$liste_questions* :

```

/* Formulaire de vote */
$liste_questions = "";
while($ligne = mysql_fetch_array($resultat)) {
    $question = $row['question'];
    $liste_questions .= '<li><input name="reponse" type="radio" value="' .
        $ligne['ID_reponse'] . '"> ' . $ligne['reponse'] .
        '</li>';
}

```

Il ne reste plus qu'à afficher le HTML en utilisant au maximum le mode littéral :

```
?>
<html>
<head></head>
<body>
<span style="font-size: 12px;">
  <span style="font-weight: bold; font-size: 14px;">
    Sondage numéro <?php print $sondage; ?>
  </span><br />
  <span style="font-weight: bold"><?php print $question; ?></span>
  <form action="traitement_vote.php" method="post">
    <ul style="list-style-type: none;">
      <?php print $liste_questions; ?>
    </ul>
    <input name="sondage" type="hidden" value="<?php print $sondage; ?>">
    <input name="" type="submit" value="Votez !">
  </form>
</span>
</body></html>
```

Vous remarquerez que l'action du formulaire est *traitement_vote.php*, script que nous allons maintenant étudier.

Traitement des votes

Le but de *traitement_vote.php* est d'ajouter un vote à la base de données s'il est valide. Il commence par charger la configuration de la base et par s'assurer que les paramètres sondage et réponse sont bien des nombres :

```
<?php
require_once("config_vote.php");
$sondage = $_POST['sondage'];
$reponse = $_POST['reponse'];
if (!is_numeric($sondage) || !is_numeric($reponse)) {
    die("Sondage ou réponse incorrects.");
}
```

Nous pouvons vérifier que les identifiants du sondage et de la réponse existent en recherchant leurs lignes dans la base de données. Si c'est le cas, une jointure entre la table des sondages et des réponses sur ces champs doit donner exactement une ligne ; on teste donc si cette requête a renvoyé quelque chose :

```

/* Recherche du sondage et de la réponse. */
$sql = "SELECT R.ID_reponse
        FROM sondages S, reponses R
        WHERE S.ID = R.ID
              AND S.ID = $sondage
              AND R.ID_reponse = $reponse";

$resultat = @mysql_query($sql, $bd) or die (mysql_error());
if (mysql_num_rows($resultat) == 0) {
    die('Sondage ou réponse inexistant. ');
}

```

Si nous sommes arrivés ici, nous pouvons vérifier que l'utilisateur n'a pas déjà voté et, si c'est le cas, insérer une ligne de vote dans la table des votes :

```

/* Vérifie la présence d'un vote précédent. */
if (!$_COOKIE["${sondage}_vote"]) {
    /* On ajoute le vote dans la table. */
    $sql = "INSERT INTO votes ( ID_reponse , ID)
            VALUES ($reponse, $sondage)";
    $resultat = @mysql_query($sql, $bd) or
        die ("Ajout impossible : " . mysql_error());
}

```

Si l'insertion du vote a réussi, nous pouvons configurer le cookie indiquant que l'utilisateur a déjà voté. Ce cookie expirera dans 30 jours.

```

/* Marque que l'utilisateur a voté pour ce sondage. */
setcookie("${sondage}_vote", "1", time() + (60*60*24 * 30));
}

```

Enfin, qu'il ait précédemment voté ou non, on lui envoie le résultat du sondage :

```

/* Redirection vers le résultat du sondage. */
header("Location: affichage_vote.php?sondage=$sondage");
?>

```

Examinons maintenant les résultats.

Récupération du résultat d'un sondage

Tout participant d'un sondage veut, évidemment, en connaître le résultat. Nous utiliserons quelques petites astuces HTML pour afficher ce résultat comme dans la Figure 12.2.

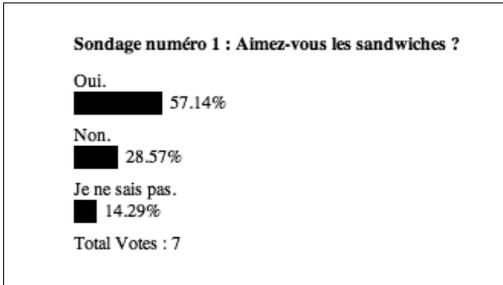


Figure 12.2 : Résultat du sondage

Le script *affichage_vote.php* commence comme les deux autres, en chargeant la configuration de la base de données et en vérifiant que le paramètre *sondage* contient bien un identifiant de sondage valide :

```
<?php
/* Affiche le résultat d'un sondage. */
require_once("config_vote.php");

$sondage = $_REQUEST['sondage'];
if (!is_numeric($sondage)) {
    die("Sondage incorrect.");
}
}
```

Lorsque l'on vérifie qu'un identifiant de sondage existe, nous pouvons aussi rechercher en même temps la question de ce sondage car on aura éventuellement besoin de l'afficher :

```
/* Recherche de la question. */
$sql = "SELECT question
      FROM sondages
      WHERE ID = $sondage";
$resultat = @mysql_query($sql, $bd) or
    die ("Erreur MySQL : " . mysql_error());
if (mysql_num_rows($resultat) != 1) {
    die('Sondage inexistant.');
```

Trouvons le nombre total de votes car nous en aurons besoin plus tard pour donner les pourcentages des différents votes :

```

$requete = "SELECT count(*) AS nb_votes_total
           FROM votes V
           WHERE V.ID = $sondage";

$resultat = @mysql_query($requete, $bd) or
    die ("Erreur MySQL : " . mysql_error());
$ligne = mysql_fetch_array($resultat);
$nb_votes_total = $ligne["nb_votes_total"];

```

Il est temps de passer à la grosse requête qui récupère les nombres de chaque vote. C'est l'endroit idéal pour utiliser la clause `LEFT JOIN` avec un groupement SQL pour classer tous les votes. Bien que cette requête soit un peu plus compliquée que toutes celles que nous avons déjà rencontrées dans ce livre, il est facile de la décortiquer pour mieux la comprendre :

```

$req = "SELECT R.reponse, R.ID_reponse, count(V.ID_reponse) as nb_votes
       FROM reponses R
       LEFT JOIN votes V
           ON V.ID = R.ID
           AND V.ID_reponse = R.ID_reponse
       WHERE R.ID = $sondage
       GROUP BY R.reponse
       ORDER BY nb_votes DESC , R.reponse ASC
";

$resultat = @mysql_query($req, $bd) or
    die ("Erreur MySQL : " . mysql_error());

```

Avec les résultats de cette requête sous la main, nous préparons l'en-tête HTML et la première partie de la page :

```

print "<html><head><title>Sondage : $question</title></head><body>";
print '<ul style="list-style-type: none; font-size: 12px;">';
print '<li style="font-weight: bold; padding-bottom: 10px;">';
print "Sondage numéro $sondage : $question";
print '</li>';

```

Puis, nous parcourons chaque choix et nous affichons le résultat pour chacun d'eux :

```

while ($ligne = mysql_fetch_array($resultat)) {
    if ($nb_votes_total != 0) {
        $pct = sprintf("%.2f", 100.0 * $ligne["nb_votes"] / $nb_votes_total);
    } else {
        $pct = "0";
    }
}

```

```

}
$largeur_boîte = strval(1 + intval($pct)) . "px";
print '<li style="clear: left;">';
print "$ligne[reponse]";
print "</li>";
print '<li style="clear: left; padding-bottom: 7px;">';
print '<div style="width: ' . $largeur_boîte . ' ; height: 15px;' .
      ' ; background: black; margin-right: 5px; float: left;">' .
      "</div>$pct%";
print '</li>';
}

```

Enfin, nous terminons le code HTML avec le nombre total de votes et les balises fermantes :

```

print '<li style="clear: left;">';
print "Total Votes : $nb_votes_total";
print '</li>';
print '</ul>';
print '</body></html>';
?>

```

Il reste bien sûr de la place pour des améliorations.

Amélioration du script

Vous pouvez adapter ce système à vos besoins de différentes façons. Vous pouvez d'abord ajouter une interface graphique pour l'administration du sondage d'où vous pourrez non seulement créer de nouveaux sondages mais également activer ou désactiver des sondages existants.

Vous pouvez rendre le sondage intégrable ; au lieu qu'il apparaisse comme un script sur sa propre page, vous pouvez le transformer en un ensemble de fonctions. Lorsque vous affichez une page, vous pouvez alors placer le bulletin de vote avec une balise `<div>`. L'aspect le plus intéressant d'un sondage intégré est que vous pouvez utiliser AJAX avec les résultats. Lorsque l'utilisateur clique sur le bouton *Votez !*, le navigateur peut lancer un code JavaScript qui prend en compte le vote et remplace le bulletin par le résultat du sondage.

Enfin, vous pourriez réfléchir à d'autres moyens de vous assurer que les utilisateurs ne votent pas deux fois. Pour ce faire, vous devez coupler la table des votes avec un système d'authentification. Ajoutez un champ indexé contenant les identifiants de connexion et vérifiez dans cette table si l'utilisateur a déjà voté au lieu d'utiliser un cookie.

Recette 75 : Cartes postales électroniques

Les cartes électroniques existent depuis que le Web a pénétré dans les foyers. Elles n'ont rien de bien compliqué puisqu'il suffit de fournir un contenu, d'ajouter un peu de code pour afficher ce contenu, d'ajouter certaines fonctionnalités comme un accusé de réception et vous avez un service de cartes électroniques. Le système décrit ici est formé de quatre scripts, comme le précédent. Chacun d'eux contient un minimum de code et vous pouvez les personnaliser en fonction de vos envies.

choisir_carte.php : Affiche les cartes disponibles.

envoi_carte.php : Présente un formulaire pour choisir et envoyer une carte particulière

affiche_carte.php : Affiche la carte au destinataire et prévient l'expéditeur.

config_carte.php : Configure la connexion à la base et fournit une fonction auxiliaire.

Ce système utilise deux tables. La première s'appelle `cartes` et possède la structure suivante :

```
CREATE TABLE cartes (  
  ID INT NOT NULL AUTO_INCREMENT ,  
  description MEDIUMTEXT NOT NULL ,  
  contenu VARCHAR( 500 ) NOT NULL ,  
  categorie VARCHAR(20) NOT NULL ,  
  largeur INT NOT NULL ,  
  hauteur INT NOT NULL ,  
  apercu VARCHAR(120),  
  PRIMARY KEY (ID)  
) TYPE = MYISAM ;
```

Chaque ligne de cette table décrit une carte. Le champ `contenu` est du HTML : il peut s'agir d'une balise `img`, d'un fichier Flash intégré ou même de texte brut. Le champ `apercu` permet de prévisualiser la carte dans la galerie ; il n'est pas obligatoire. Voici un exemple de ligne de cette table :

```
INSERT INTO cartes  
  (description, contenu, categorie, largeur, hauteur, apercu)  
VALUES  
  ("Carte d'anniversaire 1", "<b>Joyeux anniversaire !</b> (1)",  
  "Anniversaire", 600, 300, NULL );
```

La table `cartes_envoyees` mémorise les cartes qui ont été envoyées. Au début, cette table est donc vide :

```
CREATE TABLE cartes_envoyees (  
  ID_envoi INT NOT NULL AUTO_INCREMENT ,  
  me1_exp VARCHAR( 50 ) NOT NULL ,
```

```

nom_exp VARCHAR( 50 ) NOT NULL ,
me1_dest VARCHAR( 50 ) NOT NULL ,
nom_dest VARCHAR( 50 ) NOT NULL ,
message MEDIUMTEXT NOT NULL ,
jeton VARCHAR (32) NOT NULL ,
ID INT NOT NULL ,
reception TINYINT NULL ,
PRIMARY KEY ( ID_envoi )
) TYPE = MYISAM ;

```

Le script *config_carte.php* qui configure la connexion avec MySQL et qui est inclus par tous les autres contient également une fonction `afficher_carte()` :

```

<?php
$connexion = @mysql_connect("localhost", "login_bd", "mdp_bd") or
    die(mysql_error());
$db = @mysql_select_db("nom_base", $connexion) or die(mysql_error());

function afficher_carte($carte) {
    print '<div style="height: ' . $carte["hauteur"] . ';" .
        ' width: ' . $carte["largeur"] . ';" .
        ' border: 1px solid; ' .
        ' text-align: center;">';
    print $carte["contenu"];
    print '</div>';
}
?>

```

Le paramètre `$carte` de cette fonction est un tableau reflétant les champs de la table *cartes*. Vous pouvez donc directement lui passer une ligne obtenue par un appel à `mysql_fetch_array()`.

Choix d'une carte

La première étape pour envoyer une carte électronique consiste à en choisir une. Le script *choisir_carte.php* est simplement une boucle qui affiche un menu contenant toutes les cartes disponibles. Il commence donc par configurer l'entête HTML et par créer quelques classes CSS :

```

<html><head>
    <title>Choix d'une carte</title>
    <style>
        table.choix { font-family: sans-serif; font-size: 12px; }
        table.choix th { text-align: left; }
    </style>
</head>
<body>

```

Le code PHP commence par inclure la configuration de la base de données, puis recherche les cartes disponibles :

```
<?php
require_once("config_carte.php");

/* Recherche des cartes. */
$sql = "SELECT ID, apercu, description, categorie
      FROM cartes
      ORDER BY categorie, description";

$resultat = @mysql_query($sql, $connexion) or die (mysql_error());
```

Si l'on n'a trouvé aucune carte, on indique le problème. Sinon, on affiche le nombre de cartes disponibles et l'on ouvre un tableau HTML (chaque carte sera représentée comme une ligne de ce tableau) :

```
$nb_cartes = mysql_num_rows($resultat);
if ($nb_cartes == 0) {
    die("Aucune carte n'a été trouvée.");
}
$pluriel = ($nb_cartes == 1) ? "carte disponible" : "cartes disponibles";
print "Il y a $nb_cartes $pluriel :<br />";
print "(cliquez sur une carte pour l'envoyer)<br />";
print '<table class="choix">';
print "<tr><th>Catégorie</th><th>Nom</th><th>Aperçu</th></tr>";
```

Nous sommes prêts à afficher toutes les cartes comme des lignes du tableau. C'est une boucle relativement simple qui ne nécessite pas de formatage car nous l'avons déjà précisé dans l'en-tête du tableau. Vous remarquerez qu'on utilise un lien vers le script suivant, *envoi_carte.php*.

```
while($ligne = mysql_fetch_array($resultat)) {
    $lien = "envoi_carte.php?ID=$ligne[ID]";
    print '<tr>';
    print "<td>$ligne[categorie]</td>";
    print "<td><a href=\"\$lien\">$ligne[description]</a></td>";
    if ($ligne["apercu"]) {
        print "<td><a href=\"\$lien\">
                <img src=\"\$ligne['apercu']\" /></a></td>";
    } else {
        print "<td>(pas d'aperçu)</td>";
    }
    print "</tr>";
}
```

On termine le script en ajoutant les balises fermantes. Techniquement, la balise `</table>` pourrait être placée dans la section HTML littérale mais, comme la balise ouvrante provient d'une instruction `print`, il est préférable d'être cohérent car nous pourrions éventuellement recopier ce code dans un autre script.

```
print "</table>";
?>
</body>
</html>
```

L'exécution de ce script produit un affichage comme celui de la Figure 12.3.

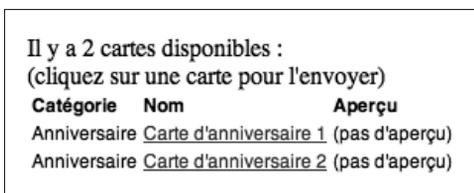


Figure 12.3 : Menu pour choisir une carte

Envoi d'une carte

Après avoir choisi une carte, l'utilisateur doit remplir un formulaire pour indiquer le destinataire, l'expéditeur, le message et envoyer la carte. Comme de nombreux scripts de ce type, *envoi_carte.php* joue deux rôles et il faut donc bien faire attention à son déroulement. Lorsqu'il présente le formulaire, le script attend un paramètre ID, contenant l'identifiant de la carte. Lorsqu'il envoie le formulaire, plusieurs paramètres se sont ajoutés : `me1_exp`, `nom_exp`, `me1_dest`, `nom_dest` et `message`.

Le script commence par valider l'identifiant de carte qui lui a été transmis et recherche cette carte dans la base de données :

```
<?php
require_once("config_carte.php");

/* Validation de l'identifiant de carte. */
$ID = $_REQUEST['ID'];
if ((!is_numeric($ID)) || ($ID == '') || ($ID < 1) ) {
    die("Identifiant de carte incorrect.");
}
$sql = "SELECT ID, categorie, contenu, largeur, hauteur, description
        FROM cartes
        WHERE ID = $ID";
```

```

$resultat = @mysql_query($sql, $connexion) or die (mysql_error());
if (mysql_num_rows($resultat) == 0) {
    die('Identifiant de carte inconnu.');
```

En vérifiant la présence d'un paramètre, on peut maintenant tester si l'on est en train d'afficher le formulaire ou d'envoyer la carte. Nous commencerons par traiter le cas où l'on envoie la carte. La première chose consiste à examiner les paramètres d'entrée. Les méthodes que nous utilisons sont minimales : vous pouvez notamment ajouter un CAPTCHA (voir la recette n°66, "Créer une image CAPTCHA pour améliorer la sécurité ") ou vérifier que les adresses de courrier sont au bon format.

```

/* Détermine le mode - Affichage du formulaire ou envoi d'une carte ? */
if (isset($_POST['mel_dest'])) {
    /* Envoi d'une carte */
    /* Vérification et nettoyage des données. */
    $mel_exp = substr($_POST['mel_exp'], 0, 50);
    $nom_exp = substr($_POST['nom_exp'], 0, 50);
    $mel_dest = substr($_POST['mel_dest'], 0, 50);
    $nom_dest = substr($_POST['nom_dest'], 0, 50);
    $message = substr($_POST['message'], 0, 600);
    $message = strip_tags($message);
    $nom_dest = strip_tags($nom_dest);
    $nom_exp = strip_tags($nom_exp);

    if ($_POST['message'] == '') {
        die("Vous devez fournir un message !");
    }
}
```

Le script a simplement supprimé les balises HTML du message, mais on peut aussi vouloir le formater un peu ; c'est la raison pour laquelle on utilise la fonction `autop()` présentée dans la recette n°42, "Convertir du texte normal en HTML".

```

/* Transformation du message texte en HTML. */
require("autop.php");
$message = autop($message);
```

On fournit maintenant au destinataire un jeton unique qui lui permettra de visualiser le message personnalisé. Bien que, techniquement, MD5 ne garantisse pas l'unicité, la probabilité d'avoir deux hachages MD5 identiques avec les paramètres utilisés est proche de zéro ; on choisira donc cette méthode pour sa simplicité :

```
/* Création d'un jeton de visualisation. */
$jeton = md5(strval(time()) . $mel_exp . $mel_dest . $ID);
```

Nous n'utilisons pas d'identifiant auto-incrémenté pour ce jeton car il serait trop facile à deviner. On insère maintenant les informations sur cette carte dans la table `cartes_envoyees` :

```
/* Insère la ligne dans la table des cartes envoyées. */
$sql = 'INSERT INTO cartes_envoyees
      (mel_exp, nom_exp, message,
       mel_dest, nom_dest, jeton, ID)
      VALUES
      (" . $mel_exp . "', "' .
       mysql_escape_string($nom_exp) . "', "' .
       mysql_escape_string($message) . "', "' .
       $mel_dest . "', "' .
       mysql_escape_string($nom_dest) . "', "' .
       $jeton . "', '
       $ID . ')';

$resultat = @mysql_query($sql, $connexion) or die (mysql_error());
```

Puis on utilise PHPMailer (voir la recette n°64, "Envoyer du courrier avec PHPMailer") pour expédier le message à son destinataire :

```
/* Classe PHPMailer pour envoyer du courrier */
include_once("phpmailer/class.phpmailer.php");
$mail = new PHPMailer;
$mail->ClearAddresses();
$mail->AddAddress($mel_dest, $nom_dest);
print "$mel_dest, $nom_dest<br />";
$mail->From = 'cartes@exemple.com';
$mail->FromName = $nom_exp;
$mail->Subject = "Vous avez reçu une carte postée par $nom_exp !";
$mail->Body = "Vous avez reçu une carte électronique !\n";
$mail->Body .= "Rendez-vous sur
http://www.exemple.com/affiche_carte.php?jeton=$jeton pour la voir.\n\n";
$mail->Body .= "Cordialement,\nL'équipe e-cartes.";

if ($mail->Send()) {
    print 'Votre carte a été envoyée !';
} else {
    print "Problème d'envoi : " . $mail->ErrorInfo;
}
}
```

C'est tout ce qu'il y a à faire pour envoyer les données du formulaire. Vous remarquerez le lien vers le script final *affiche_carte.php* et son paramètre jeton dans le message envoyé. Vous devrez modifier certaines parties de ce code, notamment l'URL, que vous pouvez placer dans *config_carte.php*. En outre, ce script devrait rediriger l'utilisateur vers une autre page un peu plus jolie.

Le traitement pour afficher le formulaire est des plus classiques :

```
} else {
    /* Afficher la carte et envoyer le formulaire. */

    print '<span style="font-family: sans-serif; font-size: 12px;">';
    print '<form action="envoi_carte.php" method="post">';

    print '<input name="ID" type="hidden" value="' . $carte['ID'] . '">';
    affiche_carte($carte);
    print '<br />';

?>

Mél du destinataire :<br />
<input name="mel_dest" type="text" size="30" maxlength="50">
<br /><br />
Nom du destinataire :<br />
<input name="nom_dest" type="text" size="30" maxlength="50">
<br /><br />
Message (Pas de HTML, 600 caractères max) :<br />
<textarea name="message" cols="40" rows="6"></textarea>
<br /><br />
Mél de l'expéditeur :<br />
<input name="mel_exp" type="text" size="30" maxlength="50">
<br /><br />
Nom de l'expéditeur :<br />
<input name="nom_exp" type="text" size="30" maxlength="50"></td></tr>
<br /><br />
<input name="" type="submit" value="Envoyez votre carte !"></td></tr>
</form>
</span>
<?php
}
?>
```

Ce formulaire est représenté à la Figure 12.4.

Joyeux Anniversaire ! (1)

Mél du destinataire :

Nom du destinataire :

Message (Pas de HTML, 600 caractères max) :

Mél de l'expéditeur :

Nom de l'expéditeur :

Figure 12.4 : Envoi d'une carte

Visualisation d'une carte

Nous avons presque terminé : il reste à écrire le script *affiche_carte.php* qui affichera la carte à son destinataire et qui préviendra l'expéditeur que sa carte a été lue. Le début du script ressemble à celui des autres : il charge le fichier *config_carte.php* et nettoie les paramètres d'entrée. Ici, nous supprimons tous les caractères non alphanumériques du paramètre *jeton* :

```
<?php
require_once("config_carte.php");
$jeton = preg_replace('/[^a-z0-9]/', '', $_REQUEST['jeton']);
```

Nous avons besoin de connaître le contenu de la carte et les détails de ce message particulier. Pour cela, il suffit de joindre les tables *cartes* et *cartes_envoyees* sur leur champ ID et de rechercher le jeton.

```
$sql = "SELECT E.nom_exp, E.mel_exp, E.message,
          E.nom_dest, E.mel_dest, E.reception,
          C.contenu, C.largeur, C.hauteur
```

```

FROM cartes_envoyees E, cartes C
WHERE C.ID = S.ID
      AND E.jeton = '$jeton';

$resultat = @mysql_query($sql, $connexion) or die (mysql_error());
if (mysql_num_rows($resultat) == 0) {
    die('Carte incorrecte.');
```

```

}

$ligne = mysql_fetch_array($resultat);
```

Le champ `reception` de la table `cartes_envoyees` indique si la carte a déjà été lue ou non. Il devrait toujours valoir 0 juste après l'envoi d'une carte.

L'affichage de la carte et du message est un traitement ennuyeux. Heureusement, la fonction `affiche_carte()` de `config_carte.php` s'en charge et nous n'avons donc plus à nous en préoccuper.

```

print '<span style="font-family: sans-serif; font-size: 12px">';
print '<p>Vous avez reçu une carte !</p>';

affiche_carte($ligne);
print '<br />';

print '<strong>' . stripslashes($ligne["nom_exp"]) .
      '</strong> a écrit :';
print '<br />';
print stripslashes($ligne["message"]);
```

Passons maintenant à l'accusé de réception. Nous devons tester si la carte a déjà été consultée, car nous ne voulons évidemment pas envoyer un message à l'expéditeur à chaque fois qu'elle est lue. Comme nous l'avons fait dans `envoi_carte.php`, nous utiliserons PHPMailer pour envoyer l'accusé de réception :

```

if (!$ligne['reception']) {
    /* Prévient l'expéditeur que son message a été lu. */
    include_once("phpmailer/class.phpmailer.php");
    $mail = new PHPMailer;
    $mail->ClearAddresses();
    $mail->AddAddress($ligne['mel_exp'], $ligne['nom_exp']);
    $mail->From = 'exemple@exemple.com';
    $mail->FromName = 'Équipe E-cartes';
    $mail->Subject = 'Votre carte a été lue';
    $mail->Body = "Votre carte a été lue par $ligne[nom_dest].

        Cordialement,
        L'équipe E-cartes.";
    $mail->Send();
```

Nous devons maintenant mettre à jour le champ `reception` de cette carte dans la table `cartes_envoyees` pour que ce soit la seule fois où cet accusé de réception soit envoyé :

```
$sql = "UPDATE cartes_envoyees
      SET reception = 1
      WHERE jeton = '$jeton'";
@mysql_query($sql, $connexion);
}
?>
```

La Figure 12.5 montre ce que `affiche_carte.php` présentera à l'utilisateur. Ce n'est évidemment pas très joli mais, en tant que maître ès HTML, vous n'aurez aucune difficulté à améliorer tout cela, n'est-ce pas ?



Figure 12.5 : La carte, telle qu'elle est vue par le destinataire

Amélioration du script

Ce système peut être amélioré de plusieurs façons. Un des ajouts les plus importants consiste à installer un CAPTCHA ou un système similaire pour compliquer l'envoi de cartes de spam (voir la recette n°66, "Créer une image CAPTCHA pour améliorer la sécurité").

Une autre amélioration importante serait d'ajouter un outil d'administration pour ajouter, modifier et désactiver des cartes. En outre, à partir d'un certain nombre de cartes, il devient difficile de toutes les afficher et les gérer sur une seule page : vous pourriez donc ajouter une fonction de recherche.

Recette 76 : Un système de blog

Les blogs sont nombreux parce qu'ils sont très faciles à écrire – il suffit d'un système capable de mémoriser des dates et du contenu. Le blog que nous présenterons ici permet d'ajouter des billets, des commentaires et de visualiser les billets. Le système stocke les billets et les commentaires dans une base de données MySQL et utilise Smarty pour afficher des templates.

La table qui contient les billets s'appelle `billets_blog` :

```
CREATE TABLE billets_blog (
  ID INT NOT NULL AUTO_INCREMENT ,
  titre VARCHAR( 120 ) NOT NULL ,
  contenu TEXT NOT NULL ,
  annonce TINYTEXT NOT NULL ,
  date_billet DATETIME NOT NULL ,
  categorie VARCHAR( 12 ) NOT NULL ,
  PRIMARY KEY ( ID )
) TYPE = MYISAM ;
```

La signification de plupart de ces champs est évidente. `annonce` est un extrait du billet, ne contenant aucune balise. Les commentaires sont stockés dans la table `commentaires_blog` :

```
CREATE TABLE commentaires_blog (
  ID_comment INT AUTO_INCREMENT ,
  nom VARCHAR( 50 ) NOT NULL ,
  comment TEXT NOT NULL ,
  date_comment timestamp,
  ID INT NOT NULL ,
  PRIMARY KEY ( ID_comment )
) TYPE = MYISAM ;
```

La raison pour laquelle le champ `date_comment` est de type `timestamp` est que nous n'avons pas l'intention de permettre la modification des commentaires : une fois posté, il ne changera jamais et il n'y a donc pas besoin de configurer manuellement sa date.

Comme pour les systèmes précédents, nous utiliserons un fichier de configuration `config_blog.php` pour mettre en place la connexion MySQL et créer un objet Smarty. Avec cette configuration par défaut, les templates Smarty se trouveront dans le répertoire `templates` :

```
<?php
session_start();
$connexion = @mysql_connect("localhost", "login", "secret") or
    die("Échec de la connexion.");
$db = @mysql_select_db("nom_base", $connexion) or die(mysql_error());
require_once ("smarty/Smarty.class.php");
$smarty = new Smarty();
?>
```

Voici un résumé des quatre scripts du système :

editer_blog.php : Ajoute et modifie les billets du blog

index_blog.php : Affiche la liste des billets du blog

afficher_blog.php : Affiche un billet individuel en intégralité

commenter_blog.php : Ajoute un commentaire à un billet du blog

Créations de billets

Avant de faire quoi que ce soit d'autre, il faut pouvoir ajouter du contenu. Nous allons donc commencer par l'éditeur de billet. Avec ce script, nous pourrions réellement nous rendre compte que l'utilisation de Smarty permet de séparer les templates HTML de PHP et que cela produit un code bien plus propre. Commençons par le template *templates/edition_blog.tpl* :

```
<html><head>
<title>{$titre}</title>
{literal}
<style>
h1 {
    font-family: sans-serif;
    font-size: 20px;
}
table.champs_saisie {
    font-family: sans-serif;
    font-size: 12px;
}
table.champs_saisie td {
    vertical-align: top;
}
</style>
{/literal}
```

```

</head>
<body>
<h1>Nouveau billet</h1>
<form method="post" action="editer_blog.php">
<table class="champs_saisie">
<tr> <td>Titre :</td><td><input name="titre" type="text" /></td> </tr>
<tr> <td>Contenu :</td>
    <td><textarea name="contenu" rows="15" cols="40"></textarea></td>
</tr>
<tr> <td>Catégorie :</td><td><input name="categorie" type="text" /> </tr>
<tr> <td /><td><input name="submit" type="submit" value="Poster" /></td> </tr>
</table>
</form>
</body>
</html>

```

Ce template n'est rien de plus qu'un formulaire avec des champs titre, contenu et categorie envoyés à *editer_blog.php* via la méthode POST. La Figure 12.6 montre ce formulaire affiché dans un navigateur.

The image shows a web browser window displaying a form titled "Nouveau billet". The form contains three input fields and a submit button. The "Titre" field contains the text "Essai". The "Contenu" field is a large text area containing the text "Ceci est mon premier billet de test". The "Catégorie" field contains the text "test". Below the "Catégorie" field is a button labeled "Poster".

Figure 12.6 : Poster un billet dans le blog

Le script *editer_blog.php* fonctionne en deux modes. S'il prend ses entrées à partir du formulaire précédent, il nettoie ces entrées, ajoute le nouveau billet dans la base de données puis redirige l'utilisateur vers une page d'affichage de ce billet :

```

<?php
require_once("config_blog.php");

if ($_REQUEST["submit"]) {
    $contenu = mysql_escape_string(strip_tags($_REQUEST["contenu"],
                                                "<a><i><b><img>"));
    $annonce = substr(strip_tags($contenu), 0, 80);
    $titre = mysql_escape_string(strip_tags($_REQUEST["titre"]));
    $categorie = mysql_escape_string(strip_tags($_REQUEST["categorie"]));
    $q = "INSERT INTO billets_blog
          (titre, contenu, categorie, annonce, date_billet)
          VALUES ('$titre', '$contenu', '$categorie', '$annonce', now())";
    mysql_query($q, $connexion) or die(mysql_error());
    $id = mysql_insert_id($connexion);
    header("Location: affiche_blog.php?ID=$id");
}

```

Notez l'utilisation de la fonction `mysql_insert_id()` qui renvoie la valeur du dernier champ `AUTO_INCREMENT` inséré. Ici, il s'agit donc du champ `ID` de la nouvelle ligne de `billets_blog` et nous pouvons donc l'utiliser pour rediriger l'utilisateur vers la page qui affiche ce nouveau billet.

Si l'on doit juste afficher le formulaire au lieu d'insérer un billet, il suffit de demander à Smarty de le faire :

```

} else {
    $smarty->assign("titre", "Blog : poster un billet");
    $smarty->display("editer_blog.tpl");
}
?>

```

Techniquement, vous pourriez coder en dur le titre dans le template mais, avec une variable, vous êtes prêt à créer un éditeur plus joli si besoin est.

Affichage d'un billet

Le script *afficher_blog.php* doit afficher trois éléments : un billet, les commentaires sur ce billet et un formulaire permettant de saisir un nouveau commentaire.

Ces trois composants sont décrits dans *templates/afficher_blog.tpl*. La première partie contient des informations d'en-tête et une fonction JavaScript qui nous servira plus tard à afficher le formulaire pour les commentaires :

```
<html><head>
<title>{$titre}</title>
{literal}
<style>
  h1 {
    font-family: sans-serif;
    font-size: 20px;
  }
  h4 {
    font-family: sans-serif;
    font-size: 12px;
  }
  .contenu {
    font-family: sans-serif;
    font-size: 12px;
  }
</style>
<script>
function affiche_form_comment() {
  o = document.getElementById("form_comment");
  if (o) { o.style.display = ""; }
  o = document.getElementById("lien_comment");
  if (o) { o.style.display = "none"; }
}
</script>
{/literal}
</head>
```

Passons maintenant à la section pour le contenu du billet. Comme sa structure est statique, elle est relativement simple. Notez le lien vers *index_blog.php*, qui sera notre dernier script :

```
<body>
<span class="contenu">
<a href="index_blog.php">Mon blog</a>
<h1>{$titre}</h1>
{$date}

<p>
{$contenu}
</p>
Catégorie : {$categorie}<br />
<br />
```

Pour les commentaires sur les billets, nous utiliserons la fonctionnalité {section} de Smarty car elle nous permet d'afficher un nombre quelconque de commentaires en parcourant un tableau. Ici, la variable \$commentaires est un tableau de commentaires, chacun étant lui-même un tableau ayant pour clé nom, date et comment pour accéder aux différents contenus des commentaires. Smarty parcourt les commentaires et affiche le contenu de cette section pour chacun d'eux, ce qui permet une représentation très compacte de ce traitement.

Si vous êtes perdu, regardez plus bas comment *affiche_blog.php* affecte la variable \$commentaires.

```
<h4>Commentaires</h4>
{section name=i loop=$commentaires}
<b>{$commentaires[i].nom}</b> ({$commentaires[i].date})<br />
{$commentaires[i].comment}
<br /><br />
{/section}
```

Enfin, nous devons afficher le formulaire pour permettre aux utilisateurs d'ajouter leurs commentaires. Pour cela, on utilise une petite astuce : au lieu d'afficher directement le formulaire, on le cache jusqu'à ce que l'utilisateur clique sur le lien JavaScript "Ajouter un commentaire". L'action de ce formulaire est *commenter_blog.php*.

```
<div id="lien_comment">
  <a href="JavaScript:affiche_form_comment();">Ajouter un commentaire</a>
</div>
<div id="form_comment" style="display: none;">
  <form method="post" action="commenter_blog.php">
    <input type="hidden" name="ID" value="{ $id }">
    Votre nom :<br />
    <input name="nom" type="text" />
    <br /><br />
    Commentaire :<br />
    <textarea name="commentaire" rows="8" cols="40"></textarea>
    <br />
    <input type="submit" value="Poster le commentaire">
  </form>
</div>
</span>
</body>
</html>
```

La Figure 12.7 montre un exemple de billet avec un formulaire de commentaire caché.



Figure 12.7 : Un billet de blog (avec des commentaires)

Maintenant que nous nous sommes occupé du HTML, *affiche_blog.php* est très simple à écrire. Nous suivons le scénario classique consistant à vérifier le paramètre d'entrée ID et à rechercher le billet dans la base de données :

```
<?php
require_once("config_blog.php");
$ID = intval($_REQUEST['ID']);

$requete = "SELECT titre, categorie, contenu,
            UNIX_TIMESTAMP(date_billet) AS date_billet
            FROM billets_blog
            WHERE ID = $ID";

$resultat = @mysql_query($requete, $connexion) or die(mysql_error());

if (mysql_num_rows($resultat) == 0) {
    die('Identifiant incorrect.');
```

Si nous sommes arrivés jusqu'ici, c'est que l'identifiant du billet est correct et que l'on peut donc affecter ses données à l'objet Smarty :

```
$ligne = mysql_fetch_array($resultat);
$smarty->assign("titre", $ligne["titre"]);
$smarty->assign("contenu", $ligne["contenu"]);
$smarty->assign("categorie", $row["categorie"]);
$smarty->assign("date", date("j M Y, G:m", $ligne["date_billet"]));
$smarty->assign("id", $ID);
```

On retrouve les commentaires avec une requête SQL très simple (il n'y a pas besoin de jointure) :

```
/* Recherche des commentaires. */
$requete = "SELECT comment, nom,
            UNIX_TIMESTAMP(date_comment) AS date_comment
            FROM commentaires_blog
            WHERE ID = $ID
            ORDER BY date_comment ASC";

$resultat = @mysql_query($requete, $connexion) or die (mysql_error());
```

Il est temps de placer les commentaires dans un tableau de tableaux nommé `$commentaires` et d'affecter ce tableau à la variable `$commentaires` de Smarty. Habituellement, ce type de code est une succession d'instructions `print` pour ouvrir et fermer des balises mais, comme on l'a expliqué plus haut, la fonctionnalité `{section}` de Smarty rend tout cela trivial :

```
$commentaires = array();
while ($ligne = mysql_fetch_array($resultat)) {
    $commentaires[] = array(
        "nom" => $ligne["nom"],
        "comment" => $ligne["comment"],
        "date" => date("j M Y, G:m", $ligne["date_comment"]),
    );
}
$smarty->assign("commentaires", $commentaires);

    Il ne reste plus qu'à traiter le template :

/* Affichage de la page. */
$smarty->display("affiche_blog.tpl");
?>
```

Voyons maintenant comment traiter l'ajout d'un commentaire.

Ajout de commentaires

Le script qui ajoute des commentaires aux billets, *commenter_blog.php*, est le seul du système qui n'utilise pas de template car il n'affiche rien. Dans la section précédente, nous avons vu qu'il prenait trois paramètres : ID, commentaire et nom contenant, respectivement, l'identifiant du billet, le contenu du commentaire et le nom de celui qui a posté le commentaire. La première étape, comme d'habitude, consiste à vérifier que l'identifiant est correct et correspond à un billet existant :

```

<?php
require_once("config_blog.php");

$ID = intval($_REQUEST["ID"]);
/* Recherche le billet pour vérifier qu'il existe bien. */
$requete = "SELECT titre FROM billets_blog WHERE ID = $ID";
$resultat = mysql_query($requete, $connexion) or die(mysql_error());
if (mysql_num_rows($resultat) != 1) {
    header("Location: index_blog.php");
    exit;
}

```

Puis, nous nettoyons le texte du commentaire et le nom en supprimant tout le code HTML qu'ils pourraient contenir. S'il reste encore quelque chose après ce traitement, nous l'insérons dans la table *commentaires_blog* :

```

$nom = mysql_escape_string(strip_tags($_REQUEST["nom"]));
$commentaire = mysql_escape_string(strip_tags($_REQUEST["commentaire"]));
$commentaire = nl2br($commentaire);

if (!empty($nom) && !empty($commentaire)) {

    $requete = "INSERT INTO commentaires_blog
                (ID, comment, nom)
                VALUES ($ID, '$commentaire', '$nom')";
    mysql_query($requete, $connexion) or die(mysql_error());
}

```

Nous terminons en redirigeant l'utilisateur vers la page d'affichage du billet qui contient désormais (en théorie) son commentaire :

```

header("Location: afficher_blog.php?ID=$ID");
?>

```

Désormais, la seule chose qui manque à notre système de blog est une page d'index.

Création d'un index des billets

La page d'accueil qui présente les billets les plus récents ressemble à la page d'affichage d'un billet car elle utilise la fonctionnalité {section} de Smarty pour réduire le code de l'itération.

Le fichier *templates/index_blog.tpl* commence par cette information d'en-tête classique :

```

<html><head>
  <title>{$titre}</title>
{literal}
<style>
  table.billets {
    font-size: 12px;
  }
  table.billets td {
    padding-bottom: 7px;
  }
  .entete {
    font-size: 14px;
    font-weight: bold;
  }
</style>
{/literal}
</head>
<body>
<a href="index_blog.php">Mon blog</a>

```

L'index du blog pourra afficher les billets classés par catégorie si besoin est. La partie suivante affiche la catégorie courante, s'il y en a une :

```

{if $categorie}
  <br />
  Catégorie : {$categorie}
{/if}

```

On utilise ensuite {section} pour parcourir la variable \$billets, qui est un tableau de tableaux contenant, chacun, des informations sur un billet du blog :

```

<br />
<table class="billets">
{section name=i loop=$billets}
<tr><td>
  <span class="entete">
    <a href="affiche_blog.php?ID={$billets[i].ID}">{$billets[i].titre}</a>
  </span>
  <br />
  {$billets[i].date}<br />
  {$billets[i].annonce}<br />
  Catégorie : <a href="index_blog.php?categorie={$billets[i].param_cat}">
    {$billets[i].categorie}
  </a>

```

```
</td></tr>
{/section}
</table>
</body>
</html>
```

Le script *index_blog.php* commence par vérifier l'existence d'un paramètre *categorie* ; s'il existe, l'index n'affichera que les billets de cette catégorie. Pour ce faire, on nettoie d'abord ce paramètre et on ajoute une clause *WHERE* pour restreindre la requête que nous verrons bientôt. S'il n'existe pas, on initialise la clause *WHERE* et la variable *Smarty* avec des chaînes vides.

```
<?php
require_once("config_blog.php");

if ($_REQUEST["categorie"]) {
    $categorie = mysql_escape_string($_REQUEST["categorie"]);
    $clause_where = "WHERE categorie = '$categorie'";
    $smarty->assign("categorie", $categorie);
} else {
    $clause_where = "";
    $smarty->assign("categorie", "");
}

    La requête SQL s'exprime donc de la façon suivante :
    $sql = "SELECT titre, categorie, annonce,
            UNIX_TIMESTAMP(date_billet) AS date_billet, ID
            FROM billets_blog
            $clause_where
            ORDER BY date_billet DESC
            LIMIT 0, 20";
    $resultat = @mysql_query($sql, $connexion) or die (mysql_error());
    if (mysql_num_rows($resultat) == 0) {
        die("Aucun billet n'a été trouvé.");
    }
```

Si cette requête a réussi, nous pouvons passer directement à la construction du tableau que l'on affectera à la variable *\$billets* de *Smarty* :

```
$elts = array();
while ($ligne = mysql_fetch_array($resultat)) {
    $elts[] = array(
        "ID" => $ligne["ID"],
        "date" => date("j M Y, G:m", $ligne['date_billet']),
        "titre" => $ligne["titre"],
        "annonce" => $ligne["annonce"],
```

```

    "categorie" => $ligne["categorie"],
    "param_cat" => urlencode($ligne["categorie"]),
);
}
$smarty->assign("billets", $elts);

```

On termine en affectant le titre de la page et en affichant le template (voir la Figure 12.8 pour le résultat final) :

```

$smarty->assign("titre", "Blog : index");
$smarty->display("index_blog.tpl");
?>

```

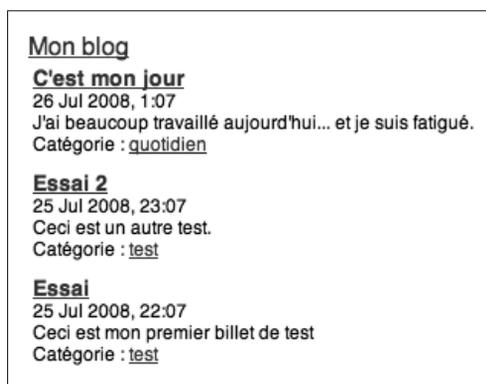


Figure 12.8 : Index des billets

Amélioration du script

Les blogs ont été conçus pour être bricolés. Vous pouvez donc ajouter un grand nombre de fonctionnalités qui utiliseront vos connaissances en PHP ou qui les étendront. Voici quelques idées simples :

Archivage : Lorsque vous commencerez à avoir un certain nombre de billets, vous devrez diviser l'index en plusieurs pages. La recette n°3, "Créer des liens Précédent/Suivant" au Chapitre 1 pourra vous y aider.

Flux RSS : Appliquez votre connaissance des services web (recette n°73, "Construire un service web ") afin de syndiquer ce blog.

Authentification et administration : Tel qu'il est conçu, n'importe qui peut ajouter des billets au blog. Vous pouvez empêcher cela en demandant une authentification avant d'ajouter un billet. En outre, vous pouvez gérer simultanément plusieurs blogs ou plusieurs auteurs, en ajoutant des noms d'utilisateurs dans les tables SQL du système.

CAPTCHA pour les commentaires : Si vous avez un blog, vous recevrez inmanquablement des commentaires de spam. Vous pouvez les éliminer avec le script de la recette n° 66, "Créer une image CAPTCHA pour améliorer la sécurité" ou avec Akismet.

Quoi que vous fassiez, amusez-vous !

Annexe



Plusieurs scripts de ce livre utilisent une table `infos_produits` contenant les détails d'un inventaire d'une hypothétique boutique. Voici la définition de cette table :

```
CREATE TABLE infos_produits (  
  nom_produit varchar(50) default NULL,  
  num_produit int(10) default NULL,  
  categorie enum('chaussures', 'gants', 'chapeaux') default NULL,  
  prix double default NULL,  
  PRIMARY KEY (num_produit)  
);
```

Après avoir créé cette table, vous voudrez lui ajouter quelques données :

```
INSERT INTO infos_produits VALUES ('Bottes Western',12,'chaussures',19.99);  
INSERT INTO infos_produits VALUES ('Pantoufles',17,'chaussures',9.99);  
INSERT INTO infos_produits VALUES ('Bottes de Snowboard',15,'chaussures',89.99);  
INSERT INTO infos_produits VALUES ('Tongs',19,'chaussures',2.99);  
INSERT INTO infos_produits VALUES ('Casquette de Baseball',20,'chapeaux',12.79);
```

Bien que le champ `num_produit` soit un peu arbitraire, il doit être unique pour chaque ligne de la table.

INDEX

A

- Accès aux fichiers [31](#)
- AddAddress(), méthode de PHPMailer [142](#)
- AddAttachment(), méthode de PHPMailer [143](#)
- AddStringAttachment(), méthode de PHPMailer [143](#)
- Afficher un tableau [14](#)
- AltBody, attribut de PHPMailer [142](#)
- Apostrophes magiques [30](#)
- Applications exemples [179](#)
- array_multisort(), fonction [17](#)
- assign(), méthode Smarty [19](#)
- Attaques
 - par injection SQL
 - protection [30](#)
 - XSS [43](#)
- autop(), fonction [90](#)

B

- Balises HTML, supprimer [93](#)
- base64decode(), fonction [50](#)
- base64encode(), fonction [50](#)
- Blog, créer [198](#)
- Boutons de validation [61](#)

C

- Carte de crédit
 - connexion SSL [130](#)
 - expiration [65](#)
 - validité [62](#)

- Cartes électroniques [188](#)
- Cascading Style Sheet (CSS)
 - Voir* Feuilles de style
- Chaînes
 - comparer [16](#)
 - extraire une partie [69](#)
 - modifier la casse [72](#)
 - sous-chaîne
 - rechercher [73](#)
 - remplacer [74](#)
- checkdate(), fonction [100](#)
- Chemins
 - des fichiers [5](#)
 - droits d'accès [5](#)
- Chiffrement
 - données [49](#)
 - Mcrypt [32](#)
- Classes
 - constructeurs [8](#)
 - destructeurs [8](#)
- ClearAddresses(), méthode de PHPMailer [142](#)
- ClearAttachments(), méthode de PHPMailer [143](#)
- Clients, extraire des informations sur [130](#)
- Comparaisons de chaînes [16](#)
- Configuration
 - ini_get(), fonction [25](#)
 - ini_set(), fonction [24](#)
 - options [25](#)
 - php.ini, fichier [24](#)

Connexion simple 136
\$_COOKIE, tableau 124
Cookies 122, 166
 créer un message 123
 navigateurs et 128
Courrier électronique 139
 vérifier les comptes utilisateurs 144
CSS, *Voir* Feuilles de style
CSV, format de fichier 119
cURL 163
 curl_close(), fonction 165
 curl_exec(), fonction 165
 curl_init(), fonction 164
 curl_setopt(), fonction 164
 extension 32
curl_close(), fonction cURL 165
curl_exec(), fonction cURL 165
curl_init(), fonction cURL 164
curl_setopt(), fonction cURL 164

D

Date
 formater 100
 instant courant 96
 jour de la semaine 103
 MySQL, format 106
 passé et futur 97
date(), fonction 96, 100
DATE, type MySQL 106
DATETIME, type MySQL 106
Désactiver une fonction PHP 32
deserialize(), fonction 15
disable_functions, option de configuration 32
display(), méthode Smarty 19
display_errors, option de configuration 28

E

Epoch, représentation du temps Unix 95
Erreurs
 activer le suivi 27
 supprimer les messages 28

error_log, option de configuration 28
error_reporting(), fonction 27, 29
Espaces inutiles 56
Expressions régulières 81
 extraire les correspondances 85
 fonction
 preg_match() 63, 67
 preg_replace() 56, 64, 68
 remplacement 86
 syntaxe 82
 tableau HTML 87
Extensions PHP, ajouter 32

F

fclose(), fonction 111
feof(), fonction 110
Feuilles de style 7, 8
fgetcsv(), fonction 119
fgets(), fonction 110, 119
Fichiers
 chemin 5
 CSV, lire 119
 écrire dans 112
 inclure 4
 mettre le contenu dans une variable 110
 permissions 107
 supprimer 114
 tester l'existence 113
file_exists(), fonction 113
file_get_contents(), fonction 88
\$_FILES, variable 119
Filtrage des données par listes
 blanches 54
 noires 54
Fonctions
 array_multisort() 17
 autop() 90
 base64decode() 50
 base64encode() 50
 checkdate() 100
 cURL 164
 date() 100
 désactiver 32

- deserialize() 15
- error_reporting() 29
- fclose() 111
- feof() 110
- fgetcsv() 119
- fgets() 110
- file_exists() 113
- fopen() 110
- function_exists() 34
- GD (graphiques) 153
- getimagesize() 115
- header() 129, 130
- htmlentities() 45
- http_build_query() 165
- include 6
- include_once() 5
- ini_get() 25
- ini_set() 24
- is_array() 54
- is_null() 54
- is_numeric() 54
- isset() 54
- is_string() 54
- mail() 139
- mcrypt() 49
- md5() 48, 50
- mktime() 66, 99
- mysql_insert_id() 201
- mysql_query() 43
- mysql_real_escape_string() 30
- nl2br() 90
- phpinfo() 25, 33, 36
- preg_match() 85
- preg_match_all() 86
- preg_replace() 56, 64, 86
- print_r() 14, 85
- pspell_config_create() 78
- pspell_config_ignore () 78
- pspell_config_mode() 78
- pspell_config_personal() 80
- pspell_save_wordlist() 80
- rand() 51
- require_once 4
- serialize() 15
- session_start() 126

- setcookie() 124
- srand() 51
- strcasecmp() 17, 73
- strip_tags () 93
- strlen() 70
- strpos() 73
- str_replace() 73, 74
- strstr() 73
- strtolower() 72
- strtotime() 97
- strtoupper() 72
- strval() 54
- substr() 70
- time() 96
- trim() 56
- ucwords() 72
- unlink() 113
- unset() 127
- usort() 16
- fopen(), fonction 110
- Formulaire 53
 - recupérer les données 55
 - sécurité 39
- function_exists(), fonction 34

G

GD

- extension 33
- fonctions
 - imagecolorallocate () 153
 - imagecopyresampled() 159
 - imagecreatefrom...() 158
 - imagecreatetruecolor() 153
 - imagedestroy() 156, 159
 - imagefilledpolygon() 153
 - imagefilledrectangle() 153
 - imagejpeg() 160
 - imagepng() 160
 - imagesx() 158
 - imagesy() 158
 - imagettftext() 156
- Géolocalisation 169
- getimagesize(), fonction 115
- getmypid(), fonction 167

H

Hachage de mot de passe [47](#)
header(), fonction [129](#), [130](#)
htmlentities(), fonction [45](#)
 protection contre les attaques [45](#)
HTMLSax, analyseur syntaxique [46](#)
http_build_query(), fonction [165](#)

I

imagecolorallocate(), fonction GD [153](#)
imagecopyresampled(), fonction GD [159](#)
imagecreatefrom(), fonction GD [158](#)
imagecreatetruecolor(), fonction GD [153](#)
imagedestroy(), fonction GD [156](#), [159](#)
imagefilledpolygon(), fonction GD [153](#)
imagefilledrectangle(), fonction GD [153](#)
imagejpeg(), fonction GD [160](#)
imagepng(), fonction GD [160](#)
Images [149](#)
 CAPTCHA pour la sécurité [149](#)
 créer des vignettes [157](#)
 déposer dans un répertoire [114](#)
 GD [33](#)
imagesx(), fonction GD [158](#)
imagesy(), fonction GD [158](#)
imagettftext(), fonction GD [156](#)
include, fonction [6](#)
include_once(), fonction [5](#)
Inclure des fichiers [4](#)
ini_get(), fonction [25](#)
ini_set(), fonction [24](#)
intval(), fonction [54](#)
is_array(), fonction [54](#)
isHTML, attribut de PHPMailer [142](#)
is_null(), fonction [54](#)

is_numeric(), fonction [54](#)
isset(), fonction [54](#), [125](#)
is_string(), fonction [54](#)

L

Liens
 automatiques, créer [93](#)
 Précédent/Suivant [9](#)
LIMIT, clause SQL [11](#)
Listes
 blanches [54](#)
 noires [54](#)
log_errors, option de configuration [28](#)

M

magic_quotes_gpc, option de configuration [31](#)
mail(), fonction [139](#)
max_execution_time, option de configuration [29](#)
Mcrypt [48](#)
 mcrypt(), fonction [49](#)
 Mcrypt, extension [32](#)
 mcrypt_get_block_size(), fonction [50](#)
 mcrypt_get_key_size(), fonction [50](#)
 md5(), fonction [48](#), [50](#)
 mktime(), fonction [66](#), [99](#)
Modèles, *Voir* Templates
Mot de passe aléatoire [51](#)
MySQL
 extension [33](#)
 format des dates [106](#)
 mysql_insert_id(), fonction [201](#)
 mysql_query(), fonction [43](#)
 mysql_real_escape_string(), fonction [30](#), [42](#)

N

nl2br(), fonction [90](#)
Numéro de téléphone, vérifier [67](#)

O

- open_basedir
 - option de configuration 31
 - variable 5
- Orthographe, corriger 76

P

- php.ini, fichier 24
- phpinfo(), fonction 24, 25, 33, 36
- PHPMailer 140, 193, 196
- Postfix, programme 141
- preg_match(), fonction 81, 85
- preg_match_all(), fonction 86
- preg_replace(), fonction 56, 64 68, 86
- print_r(), fonction 14, 85, 168
- pspell 76
- pspell_check(), fonction 79
- pspell_config_create(), fonction 78
- pspell_config_ignore(), fonction 78
- pspell_config_mode(), fonction 78
- pspell_config_personal(), fonction 80
- pspell_save_wordlist(), fonction 80
- pspell_suggest(), fonction 79

R

- rand(), fonction 51
- Rediriger vers une page web 129
- Référence arrière 87
- register_globals, option de configuration 30, 40
- require_once, fonction 4
- REST, protocole 164, 174
 - construire un service web 174
- Risques de sécurité
 - attaques XSS 43
 - formulaire 39
 - utilisateurs 40
 - variables globales automatiques 40

S

- SafeHTML 46
- Screen scraper 88

- Scripts d'exemples 179
- Sécurité 39
 - formulaire 54
 - Voir aussi* Risques de sécurité
- Send(), méthode de PHPMailer 142
- sendmail, programme 141
- Sérialiser un tableau 15
- serialize(), fonction 15
- \$_SERVER, variable 12, 130
- Services web
 - construire 174
 - cURL, utiliser 163
 - géolocalisation 169
 - interroger Amazon 172
- \$_SESSION, variable 126
- Sessions 122
 - délais d'expiration 135
 - pour stocker des données 125
- session_start(), fonction 126
- setcookie(), fonction 124
- SimpleXML, extension 167
 - méthode simplexml_load_file() 168
 - objet SimpleXMLElement 167
- simplexml_load_file(), méthode de SimpleXML 168
- Sites web
 - connexion simple 136
 - extraire des données 88
 - se connecter à d'autres sites web 164
- Smarty 17
 - initiation 19
 - installation 18
 - variables 19
- smarty_initialize.php, fichier de configuration 18
- SMTP (Simple Mail Transfer Protocol) 140
- SOAP, protocole 164, 172
 - interroger Amazon 172
- Sondages en ligne 179
- SQL, clause LIMIT 11
- srand(), fonction 51
- SSL (Secure Socket Layer) 130
- strcasecmp(), fonction 17, 73

strip_tags(), fonction 93
strlen(), fonction 70
strpos(), fonction 73
str_replace(), fonction 73, 74
strrpos(), fonction 73
strstr(), fonction 73
strtolower(), fonction 72
strtotime(), fonction 97
strtoupper(), fonction 72
strval(), fonction 54
substr(), fonction 50, 70, 73
substr_replace(), fonction 70

T

Table infos_produits 211
Tableaux
 afficher 14
 couleur des lignes 7
 sérialisation 15
 tri 16
Templates 17
Temps d'exécution d'un script 29
Texte, convertir en HTML 90
time(), fonction 96
TIME, type MySQL 106

TIMESTAMP, type MySQL 106
Tri de tableaux 16
trim(), fonction 56

U

ucwords(), fonction 72
unlink(), fonction 113, 114
unset(), fonction 127
upload_max_filesize, option de configuration 29
usort(), fonction 16

V

Variable globale automatique,
désactiver 30

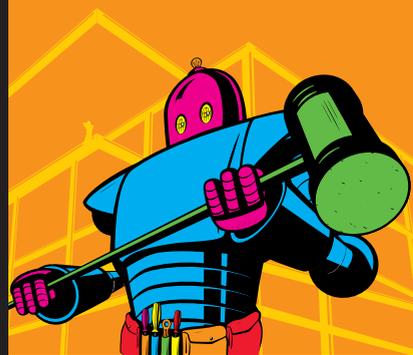
W

WSDL, langage 172

X

XLS, format de fichier 119
XML, transformer 167
XSS, attaques 43

76 SCRIPTS PHP POUR GAGNER DU TEMPS ET RÉSOLVRE VOS PROBLÈMES !



Vous découvrez le Web dynamique et PHP et vous vous demandez comment l'utiliser dans vos applications ? Si vous souhaitez apprendre tout en obtenant rapidement des résultats, ce livre est fait pour vous.

Les auteurs, programmeurs expérimentés, vous livrent des solutions «clés en main» en PHP pour résoudre les problèmes couramment rencontrés dans la création de site web. Les 76 scripts de cet ouvrage vous permettront bien sûr d'installer et de configurer PHP ou de sécuriser vos scripts, mais aussi de gérer des sessions et de manipuler fichiers, e-mails et images.

Grâce à des exemples simples et concrets et à l'explication de chaque extrait de code, vous pourrez appliquer ces 76 «recettes» pour :

- envoyer et recevoir du courrier électronique ;
- mémoriser le comportement des visiteurs à l'aide des cookies et des sessions ;

À propos des auteurs :

Programmeur et auteur, **William Steinmetz** est également webmestre éditeur de StarCityGames.com, site web dont le trafic a quadruplé depuis les améliorations qu'il lui a apportées en utilisant PHP.

Brian Ward est l'auteur d'ouvrages sur Linux et sur la virtualisation, parus chez No Starch Press.

- utiliser au mieux les options de configuration de PHP ;
- manipuler des dates, des images et du texte à la volée ;
- valider des cartes de crédit ;
- comprendre SOAP et les autres web services ;
- utiliser des modèles HTML ;
- créer un sondage en ligne, un système d'envoi de cartes électroniques et un blog en utilisant, notamment, le système de base de données MySQL ;
- chiffrer vos données confidentielles ;
- empêcher les attaques XSS...

Enfin, vous découvrirez pour chaque script des améliorations possibles, adaptées à vos besoins.

Niveau : Intermédiaire

Catégorie : Développement Web

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



ISBN : 978-2-7440-4030-6

