

Un livre simple, clair et drôle pour maîtriser PHP et MySQL !

PHP et MySQL

POUR LES NULS

1^{re} édition

A mettre
entre toutes
les mains !

*Avec les Nuls,
simplifiez-vous
la micro !*



anet Valade

PHP & MySQL
POUR
LES NULS
2^e édition

Janet Valade

FIRST

PHP & MySQL POUR LES NULS *2^e édition*

Janet Valade

FIRST
Interactive

PHP & MySQL pour les Nuls

Publié par Wiley Publishing, Inc.
111 River Street
Hoboken, NJ 07030-5774
USA

Copyright © 2007 Wiley Publishing, Inc.

Pour les Nuls est une marque déposée de Wiley Publishing, Inc.
For Dummies est une marque déposée de Wiley Publishing, Inc.

Edition française publiée en accord avec Wiley Publishing, Inc.

© 2007 Éditions First Interactive

27, rue Cassette

75006 Paris - France

Tél. 01 45 49 60 00

Fax 01 45 49 60 01

E-mail : firstinfo@efirst.com

Web : www.efirst.com

ISBN : 2-75680-0411-4

Dépôt légal : 2^e trimestre 2007

Collection dirigée par Jean-Pierre Cano

Edition : Pierre Chauvot

Maquette et mise en page : Edouard Chauvot

Traduction : Olivier Engler, François Saluden

Imprimé en Italie

Tous droits réservés. Toute reproduction, même partielle, du contenu, de la couverture ou des icônes, par quelque procédé que ce soit (électronique, photocopie, bande magnétique ou autre) est interdite sans autorisation par écrit des Editions First.

Limites de responsabilité et de garantie. L'auteur et l'éditeur de cet ouvrage ont consacré tous leurs efforts à préparer ce livre. Les Editions First, Wiley Publishing, Inc. et l'auteur déclinent toute responsabilité concernant la fiabilité ou l'exhaustivité du contenu de cet ouvrage. Ils n'assument pas de responsabilités pour ses qualités d'adaptation à quelque objectif que ce soit, et ne pourront être en aucun cas tenus responsables pour quelque perte, profit ou autre dommage commercial que ce soit, notamment mais pas exclusivement particulier, accessoire, conséquent, ou autres.

Marques déposées. Toutes les informations connues ont été communiquées sur les marques déposées pour les produits, services et sociétés mentionnés dans cet ouvrage. Wiley Publishing, Inc. et les Editions First déclinent toute responsabilité quant à l'exhaustivité et à l'interprétation des informations. Tous les autres noms de marque et de produits utilisés dans cet ouvrage sont des marques déposées ou des appellations commerciales de leur propriétaire respectif.

Sommaire

Introduction	XV
<i>Première partie : Développement d'une application de base de données sur le Web avec PHP et MySQL.....</i>	1
Chapitre 1 : Introduction à PHP et MySQL.....	3
Qu'est-ce qu'une application de base de données sur le Web ?	3
La base de données	5
L'application : transférer les informations d'une base de données dans les deux sens	6
MySQL, ma base de données	6
Avantages de MySQL	7
Comment fonctionne MySQL	8
Communications avec le serveur MySQL	9
PHP, véhicule de données	9
Avantages de PHP	10
Comment fonctionne PHP	11
Chapitre 2 : Configuration de votre environnement de travail	13
Les outils nécessaires	13
Quelle solution adopter ?	14
Le site Web d'une entreprise	14
Un site Web installé chez un hébergeur	17
Créer vous-même votre propre site Web	20
Une alternative intéressante : EasyPHP	26
Vingt fois sur le métier, remettez votre ouvrage... ..	27
Versions 4 et 5 de MySQL	28
Test de PHP	28
Test de MySQL	30
Chapitre 3 : Développement d'une application de base de données sur le Web	33
Planification de votre application de base de données sur le Web	34
Identification des objectifs de l'application	34
Se préoccuper du client	37
Faciliter l'utilisation de votre site Web	38
Prévoir de la place pour les extensions	39

Ecrivez	39
Conception de la base de données	40
Choix des données	40
Organisation des données	42
Conception des deux bases de données d'exemple	48
Type de données	55
Ecrivez-le !	57
Revenons à nos deux exemples	57
Développement de l'application	61
Construction de la base de données	61
Ecriture des programmes	61

Deuxième partie : Bases de données MySQL 63

Chapitre 4 : Construction de la base de données 65

Communications avec MySQL	65
Construction de requêtes SQL	66
Envoi de requêtes SQL	67
Construction d'une base de données	75
Création d'une nouvelle base	75
Suppression d'une base de données	76
Ajout de tables dans une base de données	76
Modification de la structure d'une base de données	79
Manipulation des données d'une base	80
Ajouter des informations	81
Recherche d'informations	86
Combinaison de tables	92
Mise à jour des informations	98
Suppression d'informations	98

Chapitre 5 : Protection de vos données 101

Contrôle des accès à vos données	101
Comptes et noms d'hôtes	102
Quelques mots sur les mots de passe	103
Les droits d'accès	104
Création de comptes MySQL	105
Identification des comptes existants	107
Ajout de nouveaux comptes et modification des droits d'accès	108
Ajout et modification des mots de passe	109
Modification des droits d'accès	109
Suppression d'un compte ou de droits d'accès	111
Sauvegarde de données	112
Restauration de vos données	115
Mise à jour de MySQL	119

Troisième partie : PHP 121

Chapitre 6 : A la découverte de PHP	123
Comment ajouter des sections écrites en PHP dans un document HTML	123
Ecriture des instructions PHP	126
Les variables PHP	130
Ecriture des noms de variables	130
Affectation d'une valeur à une variable	131
Comment maîtriser les avertissements	133
Les constantes PHP	134
Les nombres	135
Les chaînes de caractères	137
Guillemets et apostrophes	138
Concaténation de chaînes de caractères	140
Dates et heures	141
Définition de l'heure locale (local time)	141
Mise en forme d'une date	142
Rangement d'un timestamp dans une variable	144
Les dates et MySQL	145
Comparaison de valeurs	147
Comparaisons simples	148
Chaînes de caractères et motifs de recherche	150
Association de comparaisons	154
Mettez des commentaires dans vos programmes	156
Commentaires longs	157
Commentaires courts	158
Chapitre 7 : Briques de base pour l'écriture de programmes en PHP	159
Instructions simples mais utiles	159
echo	160
Instructions d'affectation	163
Incrémementation et décrémementation	164
exit	165
Appels de fonctions	166
Les tableaux	167
Création d'un tableau	167
Afficher le contenu d'un tableau	168
Suppression d'éléments dans un tableau	169
Tri de tableaux	170
Extraction de valeurs à partir d'un tableau	172
Parcours des éléments d'un tableau	174
Tableaux à plusieurs dimensions	177
Instructions conditionnelles usuelles	181
if	181
switch	184
Les boucles	185

for	186
while	189
do ... while	192
Boucles infinies	193
Rupture d'une boucle	194
Les fonctions	196
Les variables et les fonctions	198
Passage de valeurs entre une fonction et le programme environnant	200
Chapitre 8 : Mouvements de données	205
Fonctions PHP/MySQL	205
Etablissement de la connexion	207
Connexion au serveur MySQL	208
Sélection de la base de données	212
Envoi de requêtes MySQL	213
Extraction d'informations d'une base de données	214
Envoi d'une requête SELECT	215
Transfert et utilisation des informations	215
Extraction d'informations à l'aide de fonctions	222
Recueil d'informations auprès de l'utilisateur	227
PHP et les formulaires HTML	228
Créer des formulaires dynamiques	234
Traitement des informations provenant d'un formulaire	248
Contrôle des informations	252
Plusieurs boutons submit	261
Insertion d'informations dans une base de données	263
Préparation des informations	263
Ajout de nouvelles informations	268
Mise à jour des informations existantes	273
Transférer des informations par fichier	277
Utiliser un formulaire pour charger un fichier	277
Traiter le fichier téléchargé	278
Le transfert en action	279
Chapitre 9 : Transfert d'informations d'une page Web à l'autre	283
Parcours des pages Web par l'utilisateur	283
Transfert d'informations d'une page à l'autre	285
Ajout d'informations à la suite d'une URL	286
Utilisation d'un cookie pour conserver des informations	290
Passage d'informations via un formulaire HTML	292
Le mécanisme de session PHP	293
Détails de fonctionnement	293
Ouverture d'une session	295
Utilisation des variables de session	295
Les sessions sans les cookies	298
Sessions privées	301
Fermeture d'une session PHP	301

Quatrième partie : Introduction au développement d'application 303

Chapitre 10 : Construction d'une application	305
Exploitation du code source fourni	305
Choix d'un emplacement	305
Comprendre le code PHP	307
Programmation procédurale et programmation orientée objet	307
Modification du code source	308
Éditeurs de code	309
Environnement de développement intégré (IDE)	311
Planifier le développement de l'application	312
Planification du logiciel	313
Organisation supplémentaire	314
 Chapitre 11 : Intégration de la sécurité dans une application	 317
Comprendre les risques de sécurité	318
Ajout de sécurité aux scripts PHP	319
Ne faire confiance à aucune information de source externe	319
Stockage des données	325
Emploi d'appels système	326
Gestion des erreurs	328
Sécurité MySQL	329
Configuration de comptes et mots de passe	329
Accès à MySQL à partir de scripts PHP	334
Attaques par injection de commandes SQL	335
Sauvegarde des bases de données	338
Emploi d'un serveur Web sécurisé	339

Cinquième partie : Création d'une application d'authentification d'utilisateur 341

Chapitre 12 : Authentification d'utilisateur avec HTTP	343
Authentification HTTP	344
Fonctionnement du Web	345
Requête d'un fichier protégé par mot de passe	346
Autoriser l'accès	347
Authentification HTTP avec Apache	347
Configuration d'Apache	348
Création du fichier .htaccess	349
Création du fichier des mots de passe	350
L'authentification HTTP Apache en action	351
Conception d'une application d'authentification HTTP en PHP	352
Création d'une base de données d'utilisateurs	354
Conception de la base de données d'utilisateurs	354
Création de la base de données d'utilisateurs	355

Accès à la base de données d'utilisateurs	355
Construction de l'application d'authentification en PHP : démarche procédurale ...	356
Construction de l'application d'authentification en PHP : approche orientée objet	360
Développement des objets	361
Écriture de la classe PasswordPrompter	362
Écriture de la classe Database	364
Écriture de la classe Account	368
Écriture de la classe WebPage	374
Écriture du script Auth-OO	377

Chapitre 13 : Application d'identification d'utilisateur 383

Conception de l'application d'identification	384
Création de la base de données d'utilisateurs	385
Conception de la base de données	385
Création de la base de données	386
Accès à la base de données	387
Ajout de données à la base de données	388
Création de la page Web d'identification	388
Conception de la page Web d'identification	389
Écriture du code de la page d'identification	390
Affichage de la page Web d'identification	400
Création de l'application d'identification : démarche procédurale	400
Écriture du script de l'application	401
Protection de vos pages Web	411
Création de l'application d'identification : approche orientée objet	412
Développement des objets	412
Écriture de la classe WebForm	413
Écriture de la classe Database	424
Écriture de la classe Account	424
Écriture de la classe Session	428
Écriture de la classe Email	432
Écriture du script de l'application d'identification	435
Protection de vos pages Web	442
Ajout de caractéristiques à l'application	443

Sixième partie : Création d'applications de vente en ligne 445

Chapitre 14 : Application de catalogue en ligne 447

Conception de l'application de catalogue en ligne	447
Création de la base de données du catalogue	448
Conception de la base de données du catalogue	449
Création de la base de données du catalogue	451
Accès à la base de données d'alimentation	451
Ajout de données à la base de données	452
Création des pages Web du catalogue	452

Conception des pages Web du catalogue	453
Écriture du code de la page d'index	455
Écriture du code de la page produits	459
Affichage des pages Web du catalogue	464
Création de l'application de catalogue en ligne : démarche procédurale	465
Création de l'application de catalogue en ligne : approche orientée objet	469
Développement des objets	469
Écriture de la classe Catalog	470
Écriture du script de l'application catalogue	476
Extension de la classe Catalog	479

Chapitre 15 : Application de panier d'achat 481

Conception de l'application de panier d'achat	481
Décisions élémentaires de conception d'application	482
Conception de la fonctionnalité de l'application	484
Création de la base de données pour le panier d'achat	485
Conception de la base de données pour le panier d'achat	485
Création de la base de données pour le panier d'achat	487
Accès à la base de données du panier d'achat	489
Ajout de données à la base de données du panier d'achat	489
Création des pages Web du panier d'achat	490
Conception des pages Web du panier d'achat	490
Écriture du code de la page des informations produits	495
Écriture du code de la page du panier d'achat	500
Écriture du code du formulaire d'expédition	508
Écriture du code pour la page de récapitulatif	515
Création de l'application de panier d'achat : démarche procédurale	520
Écriture de ShopCatalog.php	521
Écriture de ShoppingCart.php	526
Écriture de ProcessOrder.php	530
Création de l'application de panier d'achat : approche orientée objet	538
Développement des objets	538
Écriture de la classe Catalog	539
Écriture de la classe Item	542
Écriture de la classe ShoppingCart	544
Écriture de la classe Database	548
Écriture de la classe Order	549
Écriture de la classe WebForm	555
Écriture de la classe WebPage	556
Écriture de la classe Email	557
Écriture du script de l'application de panier d'achat	557
Ajout de caractéristiques à l'application	566

Septième partie : Création d'autres applications utiles 569

Chapitre 16 : Création d'un système de gestion de contenu	571
Conception de l'application CMS	571
Création de la base de données CMS	572
Conception de la base de données CMS	573
Création de la base de données CMS	577
Accès à la base de données CMS	579
Conception des pages Web du CMS	579
Création de l'application CMS : démarche procédurale	583
Écriture du code d'identification	583
Écriture de CompanyHome.php, fichier d'accès aux données	590
Écriture de company.inc, fichier d'affichage HTML	601
Écriture du code pour le détail de contenu	604
Écriture de Admin.php (manipulation des données)	608
Création de l'application CMS : approche orientée objet	616
Écriture du modèle objet	616
Écriture d'une classe de données élémentaire	619
Écriture de la classe Department	622
Écriture de la classe ContentType	624
Écriture de la classe ContentItem	626
Écriture de la classe ContentDownload	633
Écriture de la classe Database	636
Écriture de la classe WebForm	636
Écriture du code de la page d'identification	637
Écriture de fields_content.inc et content_form.inc	638
Écriture du code d'affichage	639
Écriture de company-OO.inc (code d'affichage principal)	646
Écriture de fields_content.inc et de content_form.inc	649
Écriture de Admin-OO.php (code de manipulation des données)	649
Amélioration du système de gestion de contenu	653

Huitième partie : Les dix commandements 655

Chapitre 17 : Dix conseils pour le développement d'application	657
Première chose, planifiez	657
Soyez homogène	658
Testez le code de manière incrémentale	658
Pensez à ceux qui prendront la relève	659
Utilisez des constantes	659
Écrivez du code réutilisable	659
Séparez la mise en page et la fonction	659
Ne réinventez pas la roue	660
Utilisez fréquemment les listes de discussion, mais intelligemment	660
Documentez tout	661

Chapitre 18 : Dix sources de code PHP	663
SourceForge.net	663
WeberDev	664
PHP Classes	664
Codewalkers	665
PHP Builder	665
HotScripts.com	666
Zend	666
PHP Freaks	666
PX : PHP Code Exchange	667
Free PHP et MySQL Hosting Directory	667

Neuvième partie : Annexes **669**

Annexe A : Installation de MySQL	671
MySQL sous Windows	671
Téléchargement et installation de MySQL	672
Exécution de l'assistant de configuration MySQL	674
Lancement et arrêt du serveur MySQL	676
Sous Linux/UNIX	678
Installation avec les fichiers RPM (Linux uniquement)	679
Installation avec les fichiers source	681
Sous MacOS	683
Contrôle d'intégrité d'un fichier téléchargé	685
Configuration de MySQL	686

Annexe B : Installation de PHP	689
Sous Linux/UNIX/Mac avec Apache	689
Sous Linux/UNIX	689
Sous Mac OS X	693
Options d'installation	697
Configurer Apache pour PHP	699
Installation de PHP sous Windows	700
Configurer votre serveur Web pour PHP	702
Configurer Apache pour PHP	702
Configurer IIS pour PHP	704
Configurer PHP pour MySQL	705

Annexe C : EasyPHP et PHPMyAdmin	709
Pourquoi installer PHP et MySQL ?	710
EasyPHP, qu'est-ce ?	710
Téléchargement de EasyPHP	712
Installation de EasyPHP	712
PHPMyAdmin	716

Index	721
--------------------	------------

Introduction

Bienvenue dans le monde passionnant des applications de bases de données sur le Web ! Bien que ce livre vous expose les techniques vous permettant de construire n'importe quelle application exploitant une base de données, je vous recommande de commencer par une application simple. Vous en trouverez deux exemples dans cet ouvrage, choisis en vue de représenter deux types d'applications que l'on rencontre fréquemment sur le Web : les catalogues de produits et les sites réservés à des membres ou à des clients particuliers qui doivent s'enregistrer et fournir un mot de passe pour y avoir accès. Ces exemples sont suffisamment élaborés pour que leur réalisation fasse appel à plusieurs programmes. Ils mettent en œuvre des données et des techniques de manipulation diversifiées, tout en restant faciles à comprendre. Vous pourrez aisément les adapter à la construction de toute une variété de sites Web, et en développer la structure pour y inclure toutes les fonctionnalités qui vous sembleront nécessaires.

Qu'y a-t-il dans ce livre ?

Ce livre est un guide chargé de vous faciliter la construction d'applications de bases de données. Il a été conçu comme une référence et non comme un outil pédagogique, aussi n'avez-vous nul besoin de le parcourir d'un bout à l'autre, page après page, sauf, bien sûr, si tel est votre désir. J'ai divisé la tâche que représente la réalisation d'une application de base de données sur le Web en fragments d'informations faciles à digérer. Aussi une simple consultation du sommaire vous permettra-t-elle de localiser le sujet qui vous intéresse et de vous y reporter immédiatement. Si vous avez besoin d'informations supplémentaires contenues dans un autre chapitre, vous trouverez des références à ce chapitre.

Voici un échantillon de quelques-uns des sujets traités :

- ✓ Construction et utilisation d'une base de données MySQL.
- ✓ Inclusion d'instructions PHP dans un fichier HTML.
- ✓ Mise en œuvre des fonctionnalités de PHP.
- ✓ Emploi de formulaires HTML pour collecter des informations en provenance des utilisateurs.
- ✓ Présentation dans une page Web d'informations extraites d'une base de données.
- ✓ Enregistrement d'informations dans une base de données.

Conventions utilisées dans ce livre

Ce livre renferme plusieurs exemples dans lesquels interviennent des instructions PHP allant d'une ou deux lignes de code à un programme complet. Pour ces instructions, j'ai utilisé une typographie particulière comme celle de la ligne suivante :

Ceci est une instruction PHP

J'ai également utilisé cette convention lorsque, à l'intérieur d'un paragraphe normal, j'ai eu besoin de citer une instruction ou un mot clé PHP ou MySQL. Ainsi, ce texte est un exemple d'information PHP présentant ce texte dans le cours d'un paragraphe normal.

Dans les exemples, vous verrez souvent des mots écrits en italique. Ils représentent des types généraux qui doivent être remplacés par un nom ou un mot spécifique en fonction de l'endroit où ils figurent. Par exemple, lorsque vous verrez une ligne comme celle-ci :

```
SELECT champ1, champ2 FROM table
```

vous saurez que *champ1*, *champ2* et *table* doivent être remplacés par des noms véritables parce qu'ils sont imprimés en italique. Aussi, dans votre programme, cette ligne pourra-t-elle se présenter ainsi :

```
SELECT nom, age FROM Client
```

En outre, vous pourrez trouver occasionnellement dans un exemple des points de suspension (...) à la suite d'une liste. Il ne faut pas les taper, car ils indiquent simplement que vous pouvez avoir autant d'articles que vous le voulez dans cette liste. Par exemple, si vous lisez la ligne suivante :


```
SELECT champ1,champ2,... FROM table
```

vous ne devez pas reproduire les trois points dans votre instruction ; ils signifient simplement que votre énumération peut comprendre deux articles ou davantage. Vous pourriez donc les remplacer par champ3, champ4, et ainsi de suite, ce qui donnerait concrètement :

```
SELECT nom,age,taille,pointure... FROM Client
```

Lire ou ne pas lire ?

Certaines informations figurant dans ce livre sont flanquées d'un pictogramme marqué *Note technique*. Il signifie qu'il s'agit d'une information qu'il n'est pas indispensable de lire pour être en mesure de créer une application de base de données sur le Web. Cela peut aller d'une explication complémentaire sur un sujet donné à la description d'une astuce demandant davantage de connaissances techniques pour être mise en œuvre. Certains lecteurs pourront s'y intéresser, mais vous devez savoir que vous êtes tout à fait libre d'ignorer ces notes si vous ne les trouvez ni intéressantes ni utiles.

Stupidés suppositions

Lorsqu'on écrit un livre sur un sujet précis et non une encyclopédie, on est obligé de faire quelques suppositions sur l'état des connaissances du lecteur. Ainsi, j'ai supposé que vous connaissiez HTML et aviez déjà créé des sites Web avec ce "langage". En conséquence, bien que j'aie utilisé fréquemment HTML, vous ne trouverez ici aucune explication le concernant. Si vous ne savez pas ce qu'est HTML, je vous suggère d'acquérir un livre sur le sujet (par exemple, *HTML 4 pour les Nuls*, de Ed Tittel, Natanya Pitts et Chelsea Valentine, Editions First) et de réaliser quelques pages Web avant de poursuivre votre lecture. En particulier, il est indispensable de bien connaître tout ce qui concerne les formulaires et les tableaux. Toutefois, si vous êtes du genre impatient, sachez qu'il n'est pas totalement impossible de profiter de ce livre sans bien connaître HTML. Vous aurez en effet l'occasion d'y glaner çà et là suffisamment de HTML pour être en mesure de construire votre propre site Web. Si vous décidez de continuer sans vous intéresser spécialement à HTML, je vous suggère néanmoins d'avoir à portée de main un ouvrage de référence sur le sujet. Vous pourrez certainement y trouver les explications nécessaires sur ce qui n'est pas détaillé dans mon livre.

Si, en dépit de votre peu d'expérience pratique de HTML, vous avez choisi de poursuivre la lecture de ce livre, vous risquez de manquer des connaissances de base nécessaires. Vous devez dans tous les cas savoir comment créer et sauvegarder des textes non formatés avec un éditeur de texte tel que le Bloc-notes de Windows ou votre traitement de texte habituel. Dans ce dernier cas, attention à bien spécifier que votre fichier doit être sauvegardé en texte pur et non en texte mis en forme. Vous devez aussi savoir où placer les fichiers texte contenant le code (HTML et/ou PHP) de vos pages Web, afin qu'elles soient accessibles à tous les utilisateurs qui visiteront votre site. Enfin, il faut que vous sachiez comment transférer vos fichiers sur un serveur Web afin que le monde entier puisse contempler vos pages.

Inutile d'être un spécialiste émérite des bases de données. Inutile également de savoir programmer. Toutes les informations nécessaires pour y parvenir se trouvent dans ce livre.

Comment est organisé ce livre

Ce livre est divisé en neuf parties, chacune d'elles contenant plusieurs chapitres. Cela va de l'introduction à PHP et MySQL jusqu'à l'installation, la création et l'exploitation de bases de données en passant par l'écriture d'applications complètes en PHP.

Première partie : Développement d'une application de base de données sur le Web avec PHP et MySQL

Cette partie constitue un bref tour d'horizon de l'utilisation de PHP et MySQL pour créer une application de base de données sur le Web. PHP et MySQL y sont décrits en mettant en lumière les avantages de leur utilisation conjointe. Vous y apprendrez comment démarrer, de quoi vous allez avoir besoin, comment accéder à PHP et à MySQL, et comment tester votre logiciel. Je vous montrerai comment se présente le processus de développement de ce type d'application.

Deuxième partie : Bases de données MySQL

Cette partie vous donne des détails concernant la mise en œuvre de bases de données avec MySQL : comment créer une base de données, comment la modifier et comment y placer ou en extraire des informations.

Troisième partie : PHP

Vous trouverez dans cette partie des détails sur l'écriture de programmes en PHP, programmes qui permettront à votre application Web d'insérer de nouvelles informations, de mettre à jour les informations existantes ou de supprimer des informations dans une base de données MySQL. Vous découvrirez de quelle façon utiliser les fonctionnalités de PHP pour dialoguer avec une base de données et traiter des formulaires HTML.

Quatrième partie : Introduction au développement d'applications

Cette partie décrit de manière introductive comment créer une application Web considérée comme un tout. Vous y verrez comment organiser un programme PHP, intégrer de la sécurité, ajouter des scripts et comment employer un serveur Web sécurisé.

Cinquième partie : Création d'une application d'authentification utilisateur

Dans cette partie, vous apprendrez dans un premier temps à créer une application d'authentification utilisateur avec HTTP, puis dans un second temps à créer une application d'identification utilisateur.

Sixième partie : Création d'une application de vente en ligne

Dans cette partie, vous allez développer une application complète de vente en ligne en partant de la création de la base de données du catalogue jusqu'à la création des pages Web du panier d'achat.

Septième partie : Création d'autres applications utiles

Dans cette partie, vous apprendrez à créer un système de gestion de contenu. Vous commencerez par la conception de l'application CMS, puis vous créerez la base de données CMS et ensuite, vous vous attaquerez au pages Web à la fois du point de vue d'une démarche procédurale et également et d'une approche orientée objet.

Huitième partie : Les dix commandements

C'est ici que vous trouverez les listes habituelles de ce qu'il faut faire et ne pas faire lorsqu'on développe une application de base de données sur le Web.

Neuvième partie : Annexes

Dans les annexes A et B, vous trouverez des instructions pour installer MySQL, PHP et Apache. L'Annexe C, quant à elle, est consacrée à EasyPHP et PHPMyAdmin, utilitaire de manipulation des données contenues dans une base MySQL.

Pictogrammes utilisés dans ce livre



Cette icône indique que vous allez trouver ici des informations supplémentaires sur un point particulier. Cela peut vous faire gagner du temps et vous épargner des efforts. Aussi est-il important de lire ce ou ces paragraphes.



Ces "avertissements" sont loin d'être inutiles. Ils mettent l'accent sur tel ou tel point et vous expliquent ce que vous devez faire pour éviter de tomber dans un piège.



Ce pictogramme signale des informations qu'il n'est pas indispensable de connaître en raison de leur nature plus technique. Elles peuvent néanmoins vous être utiles, bien qu'elles ne soient pas essentielles à la compréhension du sujet qui nous occupe.



Ce pictogramme est une sorte de pense-bête qui signale des informations qu'il est utile d'avoir présentes en mémoire.

Pour aller plus loin

Ce livre est organisé en suivant l'ordre à respecter lorsque l'on doit aborder un nouveau projet. Si vous êtes un débutant complet, il vous sera probablement utile de commencer par la première partie, dans laquelle vous apprendrez comment concevoir les différentes composantes de votre application et comment elles vont dialoguer. Pour réaliser votre application, il est nécessaire que vous sachiez d'abord comment créer une application MySQL. C'est pourquoi je vous présente celui-ci avant PHP. Ensuite, lorsque vous saurez ce qu'est MySQL, vous devrez le faire dialoguer avec PHP. Je vous dirai comment dans la quatrième partie. Si vous avez déjà des connaissances suffisantes sur certains des sujets abordés, passez votre chemin. Par exemple, si vous êtes familier du concept de bases de données, rendez-vous directement à la deuxième partie, dans laquelle est décrite la façon d'implémenter ce concept avec MySQL. Si vous connaissez bien MySQL, alors rendez-vous directement à la troisième partie dans laquelle je vous présente PHP.

Exploitation du code source fourni

Le code des applications de ce livre est téléchargeable sur le site des Editions First (www.efirst.com/EFI_90download.html). Chaque application se trouve dans son propre répertoire. Après copie de tous les fichiers d'un de ces répertoires vers votre espace Web, l'application est exécutable dans votre navigateur.

Première partie

Développement d'une application de base de données sur le Web avec PHP et MySQL



"Regarde, ça fait déjà trois fois que j'ai lancé
une recherche sur 'cadavres de filles ressuscités'
et je n'ai toujours rien trouvé !"

Dans cette partie...

Cette partie constitue un bref tour d'horizon de PHP et de MySQL. Vous y apprendrez comment chacun d'eux fonctionne et comment ils coopèrent dans la réalisation d'une application de base de données sur le Web. Une fois décrits ces outils, je vous montrerai comment créer votre environnement de travail, et vous présenterai les options d'accès à PHP et MySQL en vous indiquant ce qu'il faut rechercher dans chaque environnement.

Je vous présenterai ensuite une vue d'ensemble du processus de développement, du planning, de la structuration et de la construction de votre application.

Chapitre 1

Introduction à PHP et MySQL

Dans ce chapitre :

- Découvrons ce qu'est une application de base de données sur le Web.
- Jetons un regard sur PHP.
- Comment fonctionne MySQL.

PHP et MySQL dialoguent très facilement. Dans ce chapitre, vous allez voir comment s'établit ce partenariat dynamique, quels sont les avantages de chacun d'eux, comment ils travaillent et comment, de leur conjugaison, résulte une application de base de données dynamique sur le Web.

Qu'est-ce qu'une application de base de données sur le Web ?

Une *application* est un programme ou un groupe de programmes conçus pour être exploités par un utilisateur final quel qu'il soit (client, membre, acrobate...). Lorsque l'utilisateur final dialogue avec l'application au moyen d'un navigateur, on dit qu'il s'agit d'une *application de base de données sur le Web* ou, plus simplement, d'une *application Web*. Dans ce livre, vous allez trouver les informations dont vous aurez besoin pour développer une application de base de données sur le Web, capable d'accéder à une base de données au moyen d'un navigateur tel que Internet Explorer ou Firefox.

Ce type d'application est conçu pour permettre à un utilisateur d'accomplir une certaine tâche. Il peut s'agir d'une simple application affichant des informations dans la fenêtre d'un navigateur ou

d'un programme plus élaboré permettant, par exemple, de passer une commande comme les applications qu'utilisent les sites de vente en ligne Amazon ou eBay.

Vous ne serez pas étonné d'apprendre qu'une application de base de données consiste en deux composants : une application et une base de données :

- ✓ **La base de données est la mémoire à long terme de votre application Web.** Cette dernière ne peut remplir son office sans la base de données. Mais, seule, la base de données est inopérante.
- ✓ **L'application proprement dite consiste en un ou plusieurs programmes destinés à accomplir une certaine tâche.** Les programmes créent l'affichage que voit l'utilisateur dans la fenêtre de son navigateur. C'est ce qui rend votre application interactive en lui permettant d'accepter des informations de l'utilisateur, en traitant ces informations et en renvoyant une réponse composée d'après les informations extraites de la base de données. La base de données seule est inutile si vous n'êtes pas en mesure de faire circuler des informations entre l'utilisateur et elle.

Les pages Web que vous avez précédemment créées avec HTML seul sont dites *statiques*, parce que l'utilisateur ne peut dialoguer avec elles. Tous les utilisateurs voient la même page. Les pages *dynamiques*, au contraire, permettent à l'utilisateur de dialoguer avec la page Web qu'il a sous les yeux. Différents utilisateurs voient différentes pages Web. Par exemple, l'un d'eux peut consulter en ligne le catalogue d'un marchand de meubles et choisir la page des sofas, alors qu'un autre désirera consulter la page des tables de salon. Pour créer de telles pages dynamiques, vous devez ajouter quelque chose à HTML.

Il existe un langage largement utilisé pour dynamiser des pages Web : JavaScript. Il convient à plusieurs objectifs comme la gestion des déplacements de la souris (permettant, par exemple, d'afficher en surbrillance la zone sur laquelle se promène le pointeur de la souris) ou la validation des données saisies par l'utilisateur avant qu'elles ne soient envoyées au serveur. Cependant, JavaScript ne permet pas d'accéder à des bases de données. On ne peut pas s'en servir pour sauvegarder les informations saisies par l'utilisateur dans une base de données. Pour cela, c'est PHP qu'il faut utiliser. Il est capable de valider les données de l'utilisateur avant de les

sauvegarder dans la base de données. Tous les programmes de ce livre sont écrits en PHP.

La base de données

Le cœur d'une application Web de base de données est la *base de données* proprement dite, celle qui constitue la mémoire à long terme des informations utilisées par l'application. Une base de données n'est rien d'autre qu'un classeur électronique qui renferme des informations structurées de telle façon qu'il soit facile de s'y reporter. Car, après tout, conserver des informations ne sert pas à grand-chose si on est incapable de les retrouver plus tard. Une base de données peut être quelque chose de simple, possédant une structure rudimentaire comme les titres et les noms des auteurs des livres de votre bibliothèque. Ou bien quelque chose d'énorme, doté d'une structure complexe, comme celle qu'Amazon utilise pour contenir toutes les informations utiles sur son gigantesque catalogue et ses très nombreux clients.

Il existe de nombreuses variétés d'informations pouvant être conservées dans une base de données. Celle qui constitue le catalogue en ligne d'une entreprise doit contenir toutes les informations concernant chaque produit. La base de données utilisée par le site Web d'une association doit renfermer tout ce qui concerne chacun de ses membres. La direction d'une entreprise utilise une ou plusieurs bases de données recelant non seulement les renseignements d'identité des membres de son personnel, mais aussi des informations concernant leur C.V. et leur carrière. Les informations que vous projetez de conserver peuvent être du même type que celles d'autres sites Web de l'Internet ou être de nature très spécifique. Tout dépend de votre application.

Techniquement, l'expression *base de données* désigne un fichier ou un groupe de fichiers contenant des données réelles. Ces informations sont accessibles au moyen d'un ensemble de programmes appelé SGBD (système de gestion de bases de données). Presque tous les SGBD sont de type *relationnel* ; ce sont en réalité des SGBDR dans lesquels les informations sont conservées dans des tables en relation les unes avec les autres.

Dans ce livre, MySQL est le SGBDR qui sera utilisé parce qu'il convient tout particulièrement bien à des sites Web. Nous parlerons en détail de ses avantages dans la section "MySQL, ma base de données", plus

loin dans ce chapitre. Vous apprendrez comment organiser et concevoir une base de données MySQL au Chapitre 3.

L'application : transférer les informations d'une base de données dans les deux sens

Pour qu'une base de données serve à quelque chose, vous devez être en mesure de transférer des informations dans les deux sens entre elle-même et une application. Pour cela, il faut écrire un ou plusieurs programmes qui seront chargés de lancer les requêtes nécessaires. Ces requêtes sont de la forme : "Prends ces informations et stocke-les à un endroit particulier." Ou encore : "Trouve les données spécifiées et envoie-les-moi." Ces programmes sont lancés lorsque l'utilisateur dialogue avec une page Web. Par exemple, lorsque l'utilisateur clique sur le bouton Soumettre (ou Envoyer) d'un formulaire HTML, il peut provoquer l'appel d'un programme qui va traiter les informations du formulaire et les placer dans une base de données.

MySQL, ma base de données

MySQL est développé et commercialisé par MySQL AB, un éditeur suédois qui en assure également le support. Il existe deux types de licences :

- ✓ **Licence de type "open source" :** Il s'agit de la licence GPL (General Public License) du GNU qui est gratuite. Toute personne remplissant les conditions du GPL peut utiliser ce type de logiciel gratuitement. Si vous utilisez MySQL pour réaliser une base de données sur un site Web (comme ce que nous allons faire dans ce livre), vous êtes dans le bon cas de figure, même si vous gagnez de l'argent avec votre site.
- ✓ **Licence commerciale :** Pour ceux qui préfèrent cette solution à la licence GPL, il existe une licence commerciale. C'est le cas, par exemple, d'un développeur qui compte utiliser MySQL à l'intérieur d'un nouveau produit logiciel qu'il a l'intention de commercialiser. Il faut alors qu'il acquière la licence commerciale, car il n'est plus dans le cadre du GPL.

Obtenir des renseignements techniques et une assistance sur MySQL ne présente pas de difficulté. Vous pouvez vous abonner à

plusieurs listes de diffusion telles que celles proposées sur le site Web de MySQL à l'URL <http://www.mysql.com>. Vous pouvez également consulter les archives de ces listes dans lesquelles vous trouverez de nombreuses informations sous forme de questions/réponses. Si vous le souhaitez, vous pouvez opter pour une solution plus confortable et souscrire un contrat de support technique avec MySQL AB. Il en existe de cinq sortes, allant d'un simple échange d'e-mails à un contact téléphonique. Les prix sont naturellement en rapport avec la solution choisie.

Avantages de MySQL

MySQL est un SGBD très populaire parmi les développeurs. Sa rapidité et sa petite taille en font un outil idéal pour un site Web. Qu'il s'agisse d'un logiciel en open source (gratuit) entre pour une bonne part dans sa popularité. Voici un résumé de ses principaux avantages :

- ✔ **Il est rapide.** L'objectif principal des développeurs qui l'ont créé était la rapidité. En conséquence, cette préoccupation était présente dès le début de sa réalisation.
- ✔ **Il n'est pas cher.** MySQL est gratuit dans le cadre de la licence GPL et le coût d'une licence commerciale reste très raisonnable.
- ✔ **Il est facile à utiliser.** Vous pouvez réaliser et utiliser une base de données MySQL avec quelques instructions simples écrites dans le langage d'interrogation SQL qui est celui qu'utilisent habituellement tous les SGBDR. Voyez le Chapitre 4 pour en savoir davantage sur ce sujet.
- ✔ **Il fonctionne sur de nombreux systèmes d'exploitation.** Il est supporté par Windows, Linux, Mac OS, de nombreux avatars d'UNIX (Solaris, AIX et DEC UNIX, en particulier), FreeBSD, OS/2, Irix...
- ✔ **Il existe une assistance technique importante.** Le grand nombre de développeurs utilisant MySQL garantit une assistance efficace par le biais des listes de diffusion spécialisées. Les développeurs de MySQL eux-mêmes sont abonnés à ces listes. Vous pouvez même, moyennant finances, vous abonner à une assistance technique fournie par MySQL AB.

- ✓ **Il est sûr.** Il dispose d'un système d'autorisations très souple qui permet des accès à différents niveaux de privilèges. Par exemple, la création ou la suppression d'une base de données peut être limitée à certains utilisateurs ou groupes d'utilisateurs. Les mots de passe qui circulent sur l'Internet sont cryptés.
- ✓ **Il permet la création et la manipulation de bases de données de grande taille.** Le nombre de lignes de ces bases de données peut atteindre cinquante millions. Par défaut, la taille d'une table est limitée à 4 Go, mais vous pouvez aller jusqu'à 8 To pour peu que votre système d'exploitation (ou, plus précisément, celui de l'installation sur laquelle se trouve votre base de données) le permette.
- ✓ **Il est configurable.** La licence open source GPL autorise les programmeurs à modifier MySQL pour qu'il s'adapte au mieux à des besoins spécifiques.

Comment fonctionne MySQL

Le logiciel MySQL consiste en un serveur MySQL, en plusieurs programmes utilitaires destinés à faciliter l'administration des bases de données, et en quelques logiciels de support dont le serveur MySQL a besoin mais qu'il est inutile que vous connaissiez. Le cœur du système est le serveur MySQL.

Le serveur MySQL est le gestionnaire du système de bases de données. C'est lui qui manipule toutes les instructions adressées à la base de données. Par exemple, si vous voulez créer une nouvelle base de données, vous envoyez un message au serveur MySQL disant : "Crée une nouvelle base de données que tu appelleras *nouvellebase*." Le serveur MySQL crée alors un sous-répertoire dans son dossier de données, lui donne le nom *nouvellebase* et crée les fichiers nécessaires au format requis dans ce nouveau sous-répertoire. De la même façon, pour ajouter des données à cette base de données, vous envoyez un message au serveur MySQL en lui fournissant les données et en lui disant à quel endroit vous voulez qu'elles soient rangées. Vous apprendrez comment réaliser ces opérations dans la deuxième partie du livre.

Avant que vous puissiez envoyer des instructions au serveur MySQL, il doit naturellement être en service. Il est souvent configuré de façon à démarrer automatiquement en même temps que le

système d'exploitation et continue à tourner sans interruption. C'est là la configuration habituelle d'un site Web. Cependant, il est possible d'adopter une configuration différente et de lancer le serveur à la demande. Lorsqu'il est actif, le serveur MySQL est constamment à l'écoute des messages qui peuvent le concerner.

Communications avec le serveur MySQL

La totalité de votre dialogue avec une base de données s'effectue en passant des messages au serveur MySQL. Ces messages peuvent être envoyés de plusieurs façons ; dans ce livre, nous mettrons l'accent sur la communication via PHP. Il existe des instructions spéciales dans ce langage pour adresser des messages au serveur MySQL.

Le serveur MySQL doit pouvoir comprendre les instructions que vous lui envoyez et qui sont formulées dans le langage *SQL* (*Structured Query Language*). PHP, lui, ne comprend pas ce langage, mais ce n'est pas nécessaire, car il n'est là que pour passer de façon transparente à MySQL les requêtes écrites en SQL. Recevant ces requêtes, le serveur les interprète et les exécute, puis renvoie en retour un message contenant le résultat de cette exécution ou un diagnostic d'erreur si la requête n'était pas correcte. Vous trouverez des informations sur ces requêtes dans la deuxième partie du livre.

PHP, véhicule de données

PHP est un *langage de script* conçu spécialement pour être utilisé sur le Web. C'est l'outil que vous allez employer pour écrire vos pages Web dynamiques. Comme c'est un langage spécialisé, il ne contient pas toutes les fonctionnalités des langages de programmation, généraux. En conséquence, il est bien plus simple que d'autres langages comme C ou Java. Cela ne l'empêche pas d'être employé par des millions et des millions de sites, et sa popularité toujours croissante indique combien il donne satisfaction.



PHP signifie *PHP Hypertext Processor*. Il a été créé par Rasmus Lerdorf qui lui avait, à l'origine, donné le nom de *Personal Home Page* (page d'accueil personnelle). Au cours de sa croissance, son nom a été changé pour mieux refléter l'ensemble de ses possibilités.



La syntaxe de PHP ressemble à celle du C. Aussi, si vous connaissez déjà un peu ce langage, vous ne serez pas (trop) dépaycé par PHP.

En réalité, il est beaucoup plus simple que C dont il n'a pas repris l'ensemble des concepts. Il ne possède pas les possibilités de programmation de bas niveau du C qui sont inutiles dans le cas d'une programmation Web.

La force de PHP se révèle dans son habileté à communiquer avec des bases de données. Il peut dialoguer avec presque n'importe quel SGBD. Dès lors, il est inutile de connaître les subtilités de la connexion et des échanges de messages avec telle ou telle base de données. Il suffit de lui indiquer le nom de la base de données et son emplacement et il se chargera de tous les détails : connexion, transmission de vos instructions, récupération de la réponse.

Comme pour MySQL, PHP dispose d'une assistance technique. Vous pouvez vous abonner à des listes de diffusion dont vous trouverez la référence sur le site Web de PHP, à l'URL <http://www.php.net>. Il existe aussi une interface Web expérimentale pour les listes de discussion (<http://news.php.net>) à partir de laquelle vous pourrez consulter les messages échangés.

Avantages de PHP

La popularité de PHP va croissant en raison de ses nombreux avantages. En voici quelques-uns :

- ✓ **Il est rapide.** Comme il est inclus dans HTML, ses temps de réponse sont courts.
- ✓ **Il n'est pas cher. En fait, il est gratuit.** Vous en aurez pour votre argent, voire plus !
- ✓ **Il est facile à utiliser.** Il ne contient que les éléments de langage de programmation nécessaires pour créer des pages Web dynamiques. Il a été conçu pour être facilement inclus dans un fichier HTML.
- ✓ **Il fonctionne sur de nombreux systèmes d'exploitation.** On le trouve sous Windows, Linux, Mac OS, et la plupart des avatars d'UNIX.
- ✓ **Il existe une large assistance technique.** L'importante base installée des développeurs assure une assistance efficace par le moyen des listes de diffusion.
- ✓ **Il est sûr.** L'utilisateur final ne peut pas voir le code PHP.

- ✓ **Il a été conçu pour supporter les bases de données.** PHP contient des fonctionnalités qui lui permettent de dialoguer avec des bases de données spécifiques, ce qui vous évite d'avoir à apprendre les détails techniques de ces communications.
- ✓ **Il est configurable.** La licence open source permet aux programmeurs de modifier l'interpréteur en ajoutant ou en supprimant certaines fonctionnalités selon tel ou tel besoin particulier.

Comment fonctionne PHP

PHP est un *langage de script inclus*, c'est-à-dire que le code PHP est contenu dans le code HTML. Des balises HTML particulières séparent ce qui est PHP de ce qui est HTML. On crée et on édite des pages Web contenant du PHP de la même façon que s'il n'y avait que du HTML pur.

Le logiciel PHP travaille en coopération avec le serveur Web qui est chargé d'envoyer les pages Web au monde entier. Lorsque vous saisissez une URL dans votre navigateur, vous adressez un message au serveur Web qui se trouve à cette URL, message qui lui demande de vous envoyer un fichier HTML. Lorsqu'il reçoit ce fichier, le navigateur interprète et affiche son contenu. La même chose a lieu lorsque vous cliquez sur un lien d'une page Web. De la même façon, c'est le serveur Web qui va recevoir les données que vous envoyez lorsque vous cliquez sur le bouton Soumettre (ou Envoyer, ou encore Submit) d'une page.

Lorsque PHP est installé, le serveur Web est configuré pour reconnaître certaines extensions de fichiers contenant des instructions PHP. La plus courante de ces extensions est `.php`, mais il en existe d'autres (`.php3` ou `.phtml`, par exemple, quoique ces formes soient aujourd'hui périmées). Lorsque le serveur Web reçoit une requête concernant un tel fichier, il envoie la partie HTML telle quelle, mais les instructions PHP sont traitées par l'interpréteur PHP avant de les transmettre au demandeur.

Le traitement des instructions PHP produit du code HTML qui est substitué dans le fichier original aux instructions PHP. De cette façon, on est certain que l'utilisateur ne verra jamais une instruction PHP, ce qui est un gage de sécurité et de transparence. Par exemple, considérez cette simple instruction PHP :

```
<?php echo "<p>Bonjour le Monde"; ?>
```

<?php est la balise initiale (ouvrante) signalant que ce qui suit est du code PHP, et ?> la balise terminale (fermante) qui indique la fin du code. echo est une instruction PHP qui demande à PHP d'afficher ce qui suit sous forme de code HTML ordinaire. Cette instruction est donc transformée en :

```
<P>Bonjour le Monde
```

qui est une ligne de code HTML standard. C'est cette ligne qui est envoyée au navigateur de l'utilisateur. L'instruction PHP elle-même n'est pas transmise.

PHP et le serveur Web doivent coopérer étroitement. Tous les serveurs Web ne contiennent pas un interpréteur PHP, mais la plupart d'entre eux acceptent très bien sa présence. PHP a été développé en tant que projet du groupe de logiciels Apache, aussi est-il très à l'aise avec les serveurs Apache. Mais il est également accepté par les serveurs IIS/PWS (Microsoft), iPlanet (anciennement Netscape Enterprise Server) et d'autres encore.



Bien que PHP puisse coopérer avec plusieurs types de serveurs Web, c'est avec le serveur Apache qu'il s'entend le mieux. Si vous pouvez choisir ou avoir voix au chapitre dans la sélection du serveur Web par votre entreprise, n'hésitez pas, choisissez Apache ! Il est gratuit, fourni en open source, stable et très répandu. Il équipe actuellement plus de 60 % des sites Web du monde, si l'on en croit le recensement qui figure à l'URL <http://www.netcraft.com>. Il tourne sous Windows, Linux, Mac OS et la plupart des variantes d'UNIX.

Chapitre 2

Configuration de votre environnement de travail

Dans ce chapitre :

- Comment accéder à PHP et à MySQL.
- Construction de votre propre site Web à partir de zéro.
- Tester PHP et MySQL.

Une fois prise la décision d'utiliser PHP et MySQL, votre première tâche va consister à accéder à ces deux logiciels. Peut-être disposez-vous déjà d'un environnement de travail adapté à l'écriture d'applications Web dans lequel tous les outils nécessaires sont déjà installés ? Si ce n'est pas le cas, vous allez devoir vous préoccuper de ces détails. Dans ce chapitre, nous verrons comment procéder.

Les outils nécessaires

Pour être à même de réaliser des sites Web dynamiques, vous devez avoir accès aux trois outils logiciels suivants :

- ✓ **Un serveur Web.** C'est lui qui enverra vos pages Web à vos utilisateurs.
- ✓ **MySQL.** C'est le SGBDR (système de gestion de bases de données relationnelles) qui va manipuler les données de la base.

- ✓ **PHP.** C'est le langage de script que vous allez utiliser pour écrire les programmes qui rendront votre site Web dynamique.

Chacun de ces outils a été décrit au Chapitre 1.

Quelle solution adopter ?

Pour créer vos pages dynamiques, vous devez avoir accès à un site Web sur lequel ces trois outils logiciels sont déjà installés. Si tous les sites Web possèdent un serveur, en revanche, tous ne proposent pas PHP et MySQL. Voici quels sont les environnements les plus répandus dans lesquels vous trouverez tout ce qui est nécessaire pour réaliser vos pages :

- ✓ **Un site Web installé par une entreprise sur son propre ordinateur.** C'est le département informatique de l'entreprise qui a en charge les moyens informatiques et administre les logiciels installés. Dans le cadre de ce livre, votre travail consistera à programmer un site Web soit en tant que collaborateur de l'entreprise, soit en tant que consultant.
- ✓ **Un site Web installé chez un hébergeur.** Un "hébergeur" est une entreprise qui offre d'héberger sur ses machines les sites Web de tout un chacun gratuitement, comme le font la plupart des fournisseurs d'accès, ou bien moyennant finances, comme c'est le cas pour les sites Web à usage commercial. C'est l'hébergeur qui gère le logiciel installé chez lui, et alloue la place sur ses disques aux différents fichiers nécessaires à l'élaboration d'un site Web complet.
- ✓ **Un site Web qui n'existe pas encore.** Vous pouvez décider d'installer vous-même les trois outils énumérés plus haut. En général, ce sera sur votre propre machine, mais ce peut être aussi sur une autre machine si vous travaillez en tant que consultant pour une entreprise.

Le site Web d'une entreprise

Lorsque le site Web est géré par l'entreprise, vous n'avez nul besoin de posséder des compétences d'administrateur ou de savoir installer des logiciels, car c'est le personnel spécialisé de l'entreprise qui s'en charge. Presque toujours, ce site Web existe déjà et votre tâche se limite à demander l'ajout de logiciels ou la modification de la

configuration actuelle pour actualiser ce site. Quel que soit le cas, votre responsabilité se limite à l'écriture et à l'installation des fichiers HTML. Vous n'intervenez pas dans l'exploitation du site Web.

Vous accédez aux ressources informatiques de l'entreprise au travers de son département informatique. Le nom de ce département peut varier, mais son rôle est toujours le même : c'est lui le responsable des moyens d'informations de l'entreprise et de leur utilisation.

Si PHP et/ou MySQL ne sont pas déjà installés, c'est ce département qui les installera et les mettra à votre disposition. Vous pourrez intervenir pour choisir ensemble les paramètres de configuration optimaux. En général, vous ne serez pas autorisé à effectuer ces modifications vous-même. Par exemple, PHP doit être configuré avec le support de MySQL activé. Si ce n'est pas le cas, le département informatique de l'entreprise devra le réinstaller.



Pour que le monde entier puisse consulter vos pages Web, vos fichiers HTML doivent être placés dans un répertoire particulier de l'ordinateur. Le serveur Web qui les transmet s'attend à les trouver à un endroit précis. Le département informatique devra donc vous permettre d'accéder à ce répertoire. Généralement, vous testerez vos pages dans un répertoire de travail, et ne les transférerez dans le répertoire spécialisé qu'une fois qu'elles seront bien au point. Ce transfert pourra s'effectuer soit par des commandes du système d'exploitation, soit par *FTP (File Transfert Protocol)*, selon les privilèges d'accès qui vous auront été accordés par le département informatique de l'entreprise. Parfois, pour des raisons de sécurité, vous n'aurez pas directement accès au répertoire des pages Web ; dans ce cas, ce sera le département informatique qui effectuera le transfert.

Pour pouvoir utiliser les outils logiciels nécessaires à la construction de votre site Web, vous aurez besoin des informations suivantes :

- ✓ **L'adresse des pages Web.** Vous devrez connaître le chemin d'accès complet du répertoire dans lequel les fichiers devront être installés. Vous devrez également posséder un *login* et un mot de passe vous permettant d'accéder à l'ordinateur ; enfin, vous devrez connaître les commandes du système d'exploitation vous permettant d'effectuer vos transferts de fichiers.



Noms de domaines

Chaque site Web possède une adresse unique sur l'Internet : son *adresse IP* qui consiste en une suite de quatre nombres compris entre 0 et 255, séparés les uns des autres par un point. Exemples : 127.17.204.2 ou 192.163.2.33.

Ce type d'adresse n'est pas facile à mémoriser. Aussi lui préfère-t-on une adresse "parlée", formée de mots dont l'association est plus aisée à retenir. Ces deux adresses sont associées dans les deux sens. Exemples : www.amazon.fr, www.cnrs.fr, www.php.net... Un nom qui représente l'adresse d'un site Web est appelé *nom de domaine*. Il peut représenter une machine isolée ou un groupe de machines interconnectées. Dans ce dernier cas, chaque ordinateur du groupe peut avoir son propre nom. Un nom dans lequel est inclus le nom d'une machine individuelle, comme thor.monserveur.com, identifie un *sous-domaine*.

Pour être utilisable comme adresse, un nom "parlé" doit être unique. C'est pour cette raison que toutes ces adresses doivent être enregistrées par un organisme centralisateur particulier, ce qui permet d'éviter toute duplication. En principe, n'importe qui peut faire enregistrer n'importe quel nom de domaine à la seule condition que ce nom ne soit pas déjà attribué. Aux Etats-Unis, vous pouvez effectuer cet enregistrement directement sur le Web. Vous commencez par indiquer le nom que vous souhaitez enregistrer. S'il n'est pas encore affecté, vous le déclarez à votre nom personnel ou à celui de votre entreprise et réglez les droits d'enregistrement. Dès lors, ce nom est votre bien et nul autre que vous n'a le droit de l'utiliser. La somme à payer est généralement de 30 euros par an.

En France, pays où l'Administration impose ses règles tatillonnes, cela n'est pas vrai. Il faut passer sous les fourches caudines de l'AFNIC (Association française pour le nommage Internet en coopération) qui oblige à respecter des règles contraignantes. Ou alors faire enregistrer son nom aux Etats-Unis, ce qui est parfaitement légal. (N.d.T.)

Il existe de nombreux sites Web sur lesquels vous pouvez enregistrer un nom de domaine. Une recherche par Google (<http://www.google.fr>) sur les mots clés *domain name register* renvoie quelque trois millions de réponses. Vous pouvez y naviguer pour trouver le moins cher. Certains sites Web vous permettent de savoir si tel nom de domaine est ou non déjà attribué. Pour cela, ils utilisent un logiciel appelé *whois*. Une recherche par Google sur les mots clés *domain name whois* renvoie plus de un million de réponses. Vous pouvez consulter à ce sujet les sites Web de Allwhois.com (<http://www.allwhois.com>) et de BetterWhois.com (<http://www.betterwhois.com>).

- ✓ **Les noms de fichiers par défaut.** Lorsqu'un utilisateur indique une URL dans laquelle aucune page HTML n'est spécifiée, il reçoit néanmoins quelque chose. Le serveur Web lui envoie un fichier dont le nom par défaut est fixé à l'avance. Presque toujours, ce nom sera `index.htm` ou `index.html`, mais ce pourrait être `default.htm` ou `default.html`. A vous de vous renseigner.
- ✓ **Un compte MySQL.** L'accès aux bases de données MySQL est contrôlé au moyen d'un système de comptes et de mots de passe. C'est le département informatique qui crée ces entités. (Nous reverrons cette notion de compte MySQL au Chapitre 5.)
- ✓ **L'emplacement des bases de données MySQL.** Les bases de données MySQL ne sont pas nécessairement situées sur la même machine que les pages Web qui les exploitent. Lorsqu'elles figurent sur une autre machine, il faut savoir quel est le nom de cette machine (par exemple : `thor.monserveur.com`).
- ✓ **L'extension des fichiers PHP.** Lors de l'installation de PHP, une extension par défaut des fichiers PHP est définie pour que le site Web puisse les reconnaître. La plupart du temps, cette extension sera `.php` ou `.phtml`, mais il existe d'autres variantes. Les fichiers PHP qui n'auraient pas la bonne extension seraient tout simplement ignorés. Ici encore, renseignez-vous auprès du département informatique de l'entreprise.

Un site Web installé chez un hébergeur

Un *hébergeur* est une entreprise qui vous propose tout ce dont vous pouvez avoir besoin pour créer un site Web : de la place sur ses disques, des logiciels, et même, en cas de besoin, une assistance technique. Dès lors, il ne vous reste plus qu'à créer les fichiers de votre application et à les installer à l'emplacement qui vous a été alloué par l'hébergeur.

Il existe, tant en France qu'aux Etats-Unis, des quantités d'hébergeurs. La plupart vous facturent mensuellement l'utilisation de leurs moyens informatiques. De leur côté, les fournisseurs d'accès (gratuits ou payants) vous offrent souvent un accès gratuit pour vos pages Web, mais (tout au moins en France) cette gratuité est parfois limitée à des sites Web non commerciaux. Le prix à payer dépend des ressources utilisées. Par exemple, un site Web qui



occupe 2 Mo d'espace disque vous coûtera moins cher qu'un site Web qui en requiert 10.

Pour des raisons de sécurité, les fournisseurs d'accès qui vous proposent PHP ne vous permettent pas toujours d'utiliser l'ensemble de ses instructions. Par exemple, l'instruction `mail`, qui autorise l'envoi d'un e-mail au moyen d'un script, est souvent désactivée afin de lutter contre le *spam*.

Lors de la recherche d'un hébergeur, assurez-vous que celui-ci offre les fonctionnalités suivantes :

- ✓ **PHP et MySQL.** Tous les hébergeurs n'offrent pas ces outils. Par exemple, en France, ni Club-Internet ni Wanadoo ne les proposent (sauf en tant qu'option payante). Chez les hébergeurs spécialisés, certains vous feront payer un supplément pour les utiliser.
- ✓ **Accès à PHP 4.** Il peut arriver que l'interpréteur PHP installé ne soit pas le plus récent. Actuellement, on en est à PHP 5, mais de nombreux hébergeurs proposent encore PHP 4. La version 6 va se répandre dans les prochains mois.

Considérez aussi les points suivants :

- ✓ **Fiabilité.** Il est indispensable que vous puissiez compter sur votre hébergeur. Aussi doit-il avoir pignon sur rue et bonne réputation pour ne pas risquer de disparaître d'un jour à l'autre. Et son matériel doit être récent. Méfiez-vous des machines assemblées avec du fil de fer et des bouts de ficelle, et dont le temps d'indisponibilité est supérieur au temps d'activité.
- ✓ **Rapidité.** Les pages Web qui se chargent avec une lenteur majestueuse ont vite fait de lasser les utilisateurs, toujours prompts à s'impatienter. Cette lenteur peut être causée par un équipement insuffisant, incapable de supporter dans de bonnes conditions la charge qui lui est imposée. Parfois, la configuration et la mise à jour de cet équipement n'ont pas suivi l'accroissement du nombre de ses utilisateurs. Quoi qu'il en soit, il faut éviter ce type d'hébergeur.
- ✓ **Assistance technique.** Certains hébergeurs ne vous proposent pas d'assistance, et vous n'aurez pas d'interlocuteur valable pour répondre aux questions techniques que vous vous posez ou pour résoudre les difficultés que vous risquez

de rencontrer. Parfois, cette assistance n'est fournie que par e-mail, ce qui n'est acceptable que si la réponse parvient dans un délai suffisamment court. Vous pouvez parfois tester la qualité de cette assistance en l'appelant au téléphone ou en lui envoyant un e-mail.

- ✓ **Le nom de domaine.** Certains hébergeurs vous permettent de disposer de votre propre nom de domaine : celui que vous avez enregistré selon la procédure qui a été expliquée plus haut. Certains iront même jusqu'à vous aider à enregistrer ce nom. En revanche, d'autres vous imposeront un nom dans lequel figurera leur propre nom de domaine. Par exemple, si vous vous appelez Jules Dupont, le fournisseur d'accès Free vous imposera le nom `http://jules.dupont.free.fr`. Chez Wanadoo, ce serait `http://perso.wanadoo.fr/julesdupont`. Notez que ce n'est généralement pas le cas pour les hébergeurs payants.
- ✓ **Sauvegardes.** Pour des raisons de sécurité, il est bon que vous ayez des copies de vos fichiers HTML/PHP et de vos bases de données, pour le cas où ils se trouveraient endommagés, quelle que soit la raison de ce désastre. Vérifiez donc que votre hébergeur effectue régulièrement des copies de sauvegarde des fichiers de ses clients. Et demandez-lui aussi combien de temps il lui faut pour procéder à la restauration des fichiers endommagés en utilisant cette sauvegarde.
- ✓ **Fonctionnalités diverses.** Selon l'objet de votre site Web, certaines de ces fonctionnalités peuvent être importantes et d'autres de second ordre. Presque toujours, plus les fonctionnalités offertes seront nombreuses et plus votre facture mensuelle sera salée. Voici un échantillon de ce qui peut vous être proposé :

Espace disque. Quelle place sur le disque votre application demande-t-elle ? N'oubliez pas que certains fichiers comme les fichiers audio ou vidéo réclament beaucoup de place.

Volume de trafic de votre site. Certains hébergeurs vous font payer un prix qui augmente avec le nombre de consultations de vos pages. Si votre site devient très populaire, votre facture s'en ressentira.

Adresses e-mail. La plupart des hébergeurs vous proposent un lot d'adresses e-mail pour votre site Web. A vous de voir comment se présentent ces adresses et combien vous sont offertes.

Logiciels. Les hébergeurs vous proposent généralement un certain nombre d'outils logiciels pour créer vos pages. Nous avons déjà parlé de PHP et MySQL. D'autres SGBD peuvent vous être proposés. On peut même vous proposer des outils de support de pages Web comme FrontPage (qui nécessite l'installation, côté serveur, d'extensions particulières), des logiciels de gestion de caddie virtuel, de validation de carte de crédit, etc.

Statistiques. Vous pourrez souvent obtenir des statistiques mensuelles d'utilisation de votre site Web : nombre de consultations des pages, durée des accès, etc.

Créer vous-même votre propre site Web

Voici quelles sont les étapes à parcourir pour mettre sur pied un site Web personnel (vous trouverez des détails dans les sections qui suivent) :

1. **Configuration et mise en service d'un ordinateur.**
2. **Installation du logiciel serveur.**
3. **Installation de MySQL.**
4. **Installation de PHP.**

Si vous n'avez ni matériel ni logiciel, commencez à l'étape 1. Si vous possédez un ordinateur mais sans logiciel pour le Web, rendez-vous à l'étape 2. Si vous avez déjà installé un serveur personnel sans PHP et MySQL, l'étape 3 est pour vous.

Installation d'un ordinateur

Votre première décision à prendre concerne le choix du matériel ou, plus précisément, de la plate-forme (ensemble matériel + logiciel) à acquérir puis à installer. Dans la plupart des cas, vous allez choisir un PC et votre système d'exploitation sera Windows ou Linux. Voici

quelques-uns des avantages et inconvénients de ces deux systèmes d'exploitation :

- ✓ **Linux.** C'est un système d'exploitation écrit en open source, donc gratuit. Sa stabilité est justement renommée : il peut tourner des journées entières sans qu'il soit nécessaire de réamorcer le système. Le logiciel serveur Apache est actuellement plus à son aise dans cet environnement que sous Windows, mais la situation évolue sensiblement de ce côté. C'est la solution dont le coût apparent est le plus faible. Du côté des inconvénients, il faut bien reconnaître que la majorité de ceux qui s'y sont essayés reconnaissent que Linux est plus difficile à installer et à gérer que Windows, et que l'installation comme l'administration de nouveaux logiciels ne sont pas toujours simples.



Théoriquement, oui. Toutefois, dans la pratique, vous devrez **acquérir** une "distribution Linux", c'est-à-dire un ensemble de logiciels gratuits regroupés sur plusieurs CD, accompagnés du programme d'installation et de la notice indispensables. Parmi les plus connues, citons *Red Hat*, *Mandrake*, *Slackware*, *Caldera*, *Suse*... (N.d.T.)

- ✓ **Windows.** Windows est payant et, bien qu'il en existe plusieurs moutures, seule la dernière (Windows XP) est maintenant commercialisée. Presque tout le monde s'accorde à dire que Windows est plus facile à utiliser. De plus, comme ce système d'exploitation domine de très loin ses concurrents en ce qui concerne le nombre de systèmes installés, vous êtes toujours certain de trouver quelqu'un pour vous aider en cas de problème.

Si'il est facile d'acquérir un ordinateur sur lequel Windows soit déjà installé, c'est moins courant en ce qui concerne Linux. Actuellement, Dell, IBM et HP offrent de telles configurations.

Si vous avez décidé d'assembler vous-même votre machine en achetant les composants (carte mère, mémoire, disques, écran, clavier...), vous risquez de rencontrer des difficultés d'un autre ordre. Il est préférable d'avoir de petites compétences en électronique pour être certain de ne pas faire d'erreurs en interconnectant ces composants. Mais vous êtes alors libre d'adopter le système d'exploitation que vous préférez. Si vous en connaissez déjà un pour l'avoir pratiqué peu ou prou, choisissez-le sans hésiter, car c'est de cette façon que vous rencontrerez le moins de problèmes.



En ce qui concerne PHP et MySQL, vous devriez envisager sérieusement le passage à Linux, car PHP est un projet de l'Apache Software Foundation, ce qui fait que ces deux outils logiciels sont plus à l'aise sous Apache. Donc, toutes choses égales par ailleurs, si votre ordinateur doit principalement être utilisé comme serveur pour des applications de bases de données sur le Web, Linux est probablement le meilleur choix possible.

Il existe d'autres solutions qu'un PC tournant sous Windows ou Linux, mais elles sont moins courantes :

- ✓ Il existe des systèmes UNIX gratuits pour PC : FreeBSD (que certains préfèrent à Linux) ou une version de Solaris proposée gratuitement par Sun en téléchargement.
- ✓ Vous pouvez aussi utiliser un Macintosh en tant que serveur Web. Les nouveaux Macintosh peuvent être livrés avec PHP préinstallé. Mais le nombre limité d'utilisateurs constitue un handicap. Un bon conseil : n'hésitez pas à visiter le site <http://www.phpmac.com> pour plus d'informations.

Installation du serveur Web

Une fois l'ordinateur installé et testé, vous devez choisir le logiciel serveur à installer. Presque toujours, ce sera Apache, car il offre les avantages suivants :

- ✓ **Il est gratuit.** Que dire d'autre ?
- ✓ **Il tourne sur de nombreuses plates-formes.** Nous en avons déjà fait la remarque.
- ✓ **Il est très répandu.** Environ 60 % des sites Web de l'Internet utilisent Apache, selon le recensement effectué par Netcraft (<http://www.netcraft.com>). Vous pouvez aussi consulter la page Web à l'URL http://www.securityspace.com/s_survey/data/. Ce ne serait pas le cas si ce logiciel marchait mal. Et, de ce fait, de nombreux utilisateurs peuvent vous apporter leur aide.
- ✓ **Il est fiable.** Une fois lancé, Apache ne connaît pas de problèmes et peut tourner longtemps sans connaître d'incident.

- ✓ **Il est configurable.** La licence open source permet à tout un chacun de modifier le logiciel Apache, d'y ajouter des modules ou de modifier ceux qui existent pour coller au plus près à tout environnement particulier.
- ✓ **Il est sûr.** Il existe des adjonctions gratuites qui permettent de faire tourner Apache dans un environnement sécurisé (*SSL*, pour *Secure Sockets Layer*). Si vous avez l'intention d'utiliser votre serveur pour faire du e-commerce, c'est un impératif.



Apache est automatiquement présent lorsque vous installez la plupart des distributions Linux. On le trouve aussi sur les Macintosh récents. Pour les différentes moutures d'UNIX, vous devez télécharger le code source d'Apache et le compiler vous-même. Il existe cependant quelques *binaires* (programmes déjà compilés pour des systèmes d'exploitation particuliers). Pour Windows, vous devez installer un de ces binaires, de préférence sur Windows NT/2000/XP, bien qu'Apache puisse aussi tourner sous Windows 95/98.ME. Au moment où ces lignes sont écrites, il en existe deux versions : la 1.3.36 et la 2.0.59, qui sont toutes deux sûres. Pour plus d'informations sur ce point, consultez l'Annexe C et la page située à l'URL <http://httpd.apache.org/>. Vous y trouverez également une abondante documentation, des recommandations pour l'installation sur différentes plates-formes ainsi que des logiciels complémentaires à charger.

Sous Windows, n'oubliez pas EasyPHP (cocorico!) qui, comme son nom l'indique, vous facilitera l'utilisation de PHP (ainsi que de MySQL). Son installation commence par celle d'un serveur personnel Apache. Pour plus de détails, consultez l'Annexe C.

D'autres serveurs Web existent, parmi lesquels on peut citer celui de Microsoft : IIS/PWS (Internet Information Server), qui vient au second rang parmi les serveurs les plus utilisés avec environ 27 % de parts de marché. C'est probablement la solution à préférer si vous vous attendez à un trafic important sur votre site Web. Il est réputé pour fonctionner plus efficacement qu'Apache dans un environnement Windows. De son côté, Sun propose IPlanet (connu auparavant sous le nom de Netscape Enterprise Server), qui n'est actuellement utilisé que sur environ 5 % des sites Web exploités. Il en existe aussi d'autres, mais leur base installée est encore plus réduite.

Installation de MySQL

Une fois installés l'ordinateur et Apache, vous êtes prêt pour installer MySQL. Il faut l'installer avant PHP, parce que vous devrez connaître le chemin d'accès de MySQL lorsque vous installerez PHP.

Toutefois, avant d'installer MySQL, vérifiez qu'il ne se trouve pas déjà sur votre machine. Ou peut-être s'y trouve-t-il tout en n'étant pas activé ? Ainsi, plusieurs distributions de Linux installent automatiquement MySQL. Voici comment vérifier que MySQL est actif :

✓ Sous Linux/UNIX. Tapez :

```
ps -ax
```

Vous devriez voir s'afficher une série de programmes. Selon la variante du système d'exploitation, la commande `ps` peut prendre différentes options. Si la syntaxe ci-dessus ne donne pas le résultat escompté, consultez la liste de ces options en tapant `man ps`.

Dans la liste de programmes qui s'affiche, cherchez-en un qui s'appelle `mysqld`.

✓ Sous Windows. Si MySQL est actif, vous devez voir sa petite icône dans le coin inférieur droit de l'écran, en même temps que l'horloge temps réel et quelques autres utilitaires. Vous pouvez aussi aller vérifier dans l'outil système Services du Panneau de configuration. La ligne MySQL doit indiquer *Démarré*.

Même si MySQL n'est pas actif, il peut avoir été installé mais pas lancé. Voici comment vérifier s'il est installé sur votre machine :

✓ Sous Linux/UNIX. Tapez :

```
find / -name "mysql*"
```

Si un répertoire nommé `mysql` est affiché, c'est que MySQL a été installé.

✓ Sous Windows. Si vous n'avez pas installé EasyPHP (voir l'Annexe C), cherchez la présence d'un répertoire MySQL, dans `C:\Program Files` ou directement dans la racine de `C:`.

Si vous avez l'intention d'utiliser EasyPHP, tout ce qu'il faut sera installé sans que vous ayez besoin d'y songer.



Si MySQL est installé mais non lancé, voici comment le démarrer :

✓ **Sous Linux/UNIX :**

1. Placez-vous dans le répertoire `mysql/bin`.

C'est le répertoire que vous devriez avoir trouvé lorsque vous avez vérifié que MySQL était bien installé.

2. Tapez :

```
safe_mysqld &
```

Lorsque la commande se termine, l'invite est affichée.

3. Vérifiez que MySQL est bien actif en tapant :

```
ps -ax
```

Dans la liste de programmes qui s'affiche, cherchez-en un qui s'appelle `mysqld`.

✓ **Sous Windows (hors EasyPHP) :**

1. Ouvrez une fenêtre de ligne de commande.

Cherchez dans le sous-menu des *Accessoires* le programme **Invite de commandes**.

2. Passez dans le sous-répertoire principal de MySQL.

Voyez par défaut dans *C:\Program Files\MySQL*.

3. Descendez dans le sous-répertoire *bin*.

4. Démarrez le serveur MySQL par la commande suivante :

```
mysqld -install
```

Vous démarrez ainsi le serveur MySQL en tant que service Windows, ce que vous pouvez vérifier en allant dans l'outil de gestion des services système mentionné plus haut.

Si MySQL n'est pas installé sur votre ordinateur, vous devez le télécharger et l'installer. Pour cela, consultez le site Web dont l'URL est <http://www.mysql.com>. Vous y trouverez toutes les informations



et tous les logiciels dont vous allez avoir besoin (sinon, reportez-vous à l'Annexe A).

Installation de PHP

Une fois installé MySQL, vous êtes prêt pour installer PHP. J'ai expliqué plus haut les raisons pour lesquelles PHP devait être installé en dernier.

Cependant, avant de procéder à l'installation de PHP, vérifiez si, par hasard, il ne serait pas déjà présent. Ainsi, certaines distributions de Linux l'installent automatiquement. Pour vérifier ce point, recherchez sur votre disque dur les fichiers PHP. Pour cela :

✓ **Sous Linux/UNIX.** Tapez :

```
find : -name "php*"
```

✓ **Sous Windows (hors EasyPHP).** Utilisez la fonction de recherche (voir ce qui a été dit pour MySQL) en indiquant comme critère de recherche : `php*`.

Si vous trouvez les fichiers PHP, c'est que PHP est déjà installé ; il est donc inutile de le réinstaller. Par exemple, même si vous avez installé MySQL vous-même après PHP, peut-être l'avez-vous copié à l'endroit où PHP s'attend à le trouver. Mieux vaut vous en assurer maintenant que plus tard. Effectuez le test décrit dans la section suivante pour vérifier que MySQL et PHP sont tous deux installés et actifs.

Si vous ne trouvez aucun fichier PHP, c'est que PHP n'est pas là ! Pour l'installer, vous devez pouvoir accéder au serveur Web de votre site. Par exemple, lorsque vous installez PHP sous Apache, il vous faut éditer le fichier de configuration d'Apache. Tout ce qui est nécessaire (documentation et logiciels) se trouve sur le site de PHP, à l'URL <http://www.php.net>.

Une alternative intéressante : EasyPHP

Installer un serveur personnel, un serveur MySQL et un interpréteur PHP n'est pas à la portée de tout le monde. C'est ce qu'ont compris trois développeurs français, enthousiastes de PHP, et désireux de faire du prosélytisme. Ils ont créé un "package" contenant ces trois logiciels (package qui s'installe sous Windows avec autant de facilité

que n'importe quel outil bureautique) et l'ont appelé **EasyPHP**, autrement dit "PHP facile".

Pour apprendre PHP et MySQL et tester ses bases de données et ses scripts PHP avant de les installer sur un serveur externe, EasyPHP est l'outil idéal, car il permet de se concentrer sur les seuls problèmes concernant PHP et MySQL, sans se perdre dans les intrications que posent aux non-spécialistes l'installation et surtout la configuration de logiciels aussi élaborés qu'un serveur personnel, un SGBD et un interpréteur appelés à coopérer. Avantage supplémentaire : EasyPHP est gratuit. C'est ce système qui nous servira à illustrer les exemples du livre dans tout ce qui suit.

Comme nous l'avons signalé plus haut, l'utilisation d'un serveur personnel n'est pas réaliste pour l'exploitation d'un site Web. Il faut pratiquement recourir aux bons offices d'un hébergeur. Mais la mise au point des bases de données et des scripts peut s'effectuer à domicile, sur son propre serveur personnel, avec beaucoup plus de souplesse, une plus grande disponibilité et une moindre dépense que sur un serveur externe, propriété d'un fournisseur d'accès ou d'un hébergeur. Nous incitons donc fortement le lecteur à opérer de cette façon.

Vous trouverez tous les détails concernant le téléchargement, l'installation et l'utilisation de EasyPHP à l'Annexe C.

Vingt fois sur le métier, remettez votre ouvrage...

Autrement dit : testez, testez et testez encore. Il y a au moins trois raisons qui peuvent vous conduire à supposer que PHP et MySQL sont utilisables sur votre site Web :

- ✓ Le département informatique de votre entreprise ou celui de l'entreprise qui est votre cliente vous a dit qu'il en était ainsi.
- ✓ Votre hébergeur vous a fourni toutes les informations que vous lui avez demandées et vous a assuré que tout était prêt.
- ✓ Vous avez suivi les instructions précédentes et tout installé vous-même.

Mais cela n'est pas suffisant. Encore faut-il vérifier que PHP et MySQL sont bien en ordre de marche.

Versions 4 et 5 de MySQL

PHP sait exploiter toutes les versions de MySQL, mais son installation diffère d'une version à l'autre. PHP définit une famille de fonctions qui sont les fonctions **mysql** pour communiquer avec une base MySQL 4.0 ou antérieure, et une nouvelle famille de fonctions (la famille **mysqli**, notez le i final) pour MySQL 4.1 et suivants. Vous pouvez utiliser l'ancienne famille **mysql** avec les versions récentes de MySQL, mais vous n'aurez dans ce cas pas accès aux nouveautés fonctionnelles qu'elles apportent. La famille **mysqli** n'est disponible qu'à partir de la version 5 de PHP.

Dans ce livre, y compris pour les tests de ce chapitre, nous utilisons MySQL 5.0 et les nouvelles fonctions **mysqli**. Si vous utilisez encore PHP 4, vous devrez modifier les exemples pour appeler les fonctions **mysql** dont la syntaxe varie un peu. Un tableau du Chapitre 8 présente les différences. Des versions des exemples pour PHP 4 sont fournies avec un chiffre 4 dans le nom de fichier.

Voici le genre de message d'erreur que vous risquez de rencontrer :

```
Fatal error: Call to undefined function mysql_connect()
```

Ce message peut signifier que vous tentez d'utiliser une fonction **mysql** (et non **mysqli**) dans le programme alors que le support de ces fonctions n'est pas activé. Nous expliquons comment procéder dans l'Annexe B.

Nous présentons ces fonctions dans la suite du livre. Celles qui permettent à PHP de communiquer avec MySQL sont abordées dans le Chapitre 8. Ceux qui utilisent PHP 4 ne pourront pas lancer les exemples directement, car ils ont été modifiés pour utiliser les nouvelles fonctions **mysqli**.

Test de PHP

Pour vérifier que PHP est installé et actif, voici les étapes à parcourir :

1. **Trouvez le répertoire dans lequel vos fichiers PHP doivent être sauvegardés.**

Ce dossier et ses sous-répertoires constituent votre *espace Web*. Dans le cas d'Apache, il s'agit du répertoire dans lequel celui-ci est installé (normalement `htdocs`, également appelé

Document Root). Sous IIS, ce sera par défaut `Inetpub\wwwroot`. Sous Linux, il devrait s'agir de `/var/www/html`. Tout cela peut bien sûr être personnalisé en reconfigurant le serveur Web (voir l'Annexe C). Si votre site est hébergé, n'oubliez pas de demander le nom du répertoire qui vous est alloué pour vos scripts PHP.

2. Créez le fichier ci-après dans votre espace Web et donnez-lui le nom `test.php`.

```
<html>
<head>
<title>Test de PHP</title>
</head>
<body>
<p> Ceci est une ligne HTML
<p>
<?php
    echo "Ceci est une ligne PHP";
    phpinfo();
?>
</body>
</html>
```

3. Pointez votre navigateur sur le fichier `test.php` que vous venez de créer (en tapant le nom de votre site Web, ou encore une adresse du type `localhost/test.php` si vous effectuez un test sur votre propre système – n'utilisez pas la commande Ouvrir du menu Fichier !).

Vous devriez voir s'afficher les deux lignes suivantes :

```
Ceci est une ligne HTML
Ceci est une ligne PHP
```

Ces deux lignes sont suivies par un grand tableau montrant toutes les informations associées à PHP dans votre système (il est produit par la fonction `phpinfo()`). Vous pourrez y repérer le chemin d'accès et le nom de divers fichiers, des noms de variables et l'état de diverses options.

4. Vérifiez les valeurs des options de PHP que vous comptez utiliser.

Par exemple, le support de MySQL doit être activé. Recherchez la rubrique MySQL dans ce tableau et assurez-vous que le support de MySQL a bien la valeur "on" (ou *Enabled*).

5. Modifiez éventuellement des valeurs.

La configuration générale de PHP est stockée dans le fichier *php.ini*. Si vous avez installé PHP localement, vous avez accès à ce fichier en écriture. En revanche, si vous n'avez pas le privilège d'administrateur sur PHP, vous devrez demander au responsable du département informatique qui en dispose d'effectuer ces modifications pour votre compte. Si c'est vous qui avez installé PHP ou que vous avez ces privilèges, pas de problème, faites vous-même les modifications. (Les modifications des options de PHP sont traitées à l'Annexe B.)

Test de MySQL

Lorsque vous aurez vérifié que PHP tournait correctement, il vous restera à tester MySQL au moyen de PHP. Pour cela :

1. Créez le fichier ci-après dans votre espace Web et donnez-lui le nom `test-mysql.php` :

```
<?php
/* Programme: mysql_up.php
 * Desc:      Se connecte au serveur MySQL et
 *            affiche les paramètres.
 */
echo "<html>
        <head><title>Test de MySQL5 avec PHP5</title></head>
        <body>";
$host="localhost";
$user="";
$password="";

$cxn = mysqli_connect($host,$user,$password);
$sql="SHOW STATUS";
$result = mysqli_query($cxn,$sql);
if($result == false)
{
    echo "<h4>Erreur: ".mysqli_error($cxn)."</h4>";
}
else
{
    /* Tableau affichant les résultats */
    echo "<table border='1'>
        <tr><th>Nom_Variable</th>
        <th>Valeur</th></tr>";
    for($i = 0; $i < mysqli_num_rows($result); $i++)
    {
```



```
echo "<tr>";
$row_array = mysqli_fetch_row($result);
for($j = 0;$j < mysqli_num_fields($result);$j++)
{
    echo "<td>".$row_array[$j]."</td>\n";
}
echo "</table>";
}
?>
</body>
</html>
```

2. Les lignes imprimées en gras peuvent devoir être modifiées. Ce sont :

```
$host="localhost";
$user="mysqlaccount";
$password="mysqlpassword";
```

Remplacez les valeurs situées à droite du signe "=" par les véritables valeurs en fonction de votre environnement. A la place de localhost, indiquez le nom de l'ordinateur sur lequel MySQL est installé. Si vous utilisez un serveur local, sur lequel se trouve aussi MySQL, vous pouvez laisser le nom localhost (ou ne rien mettre).

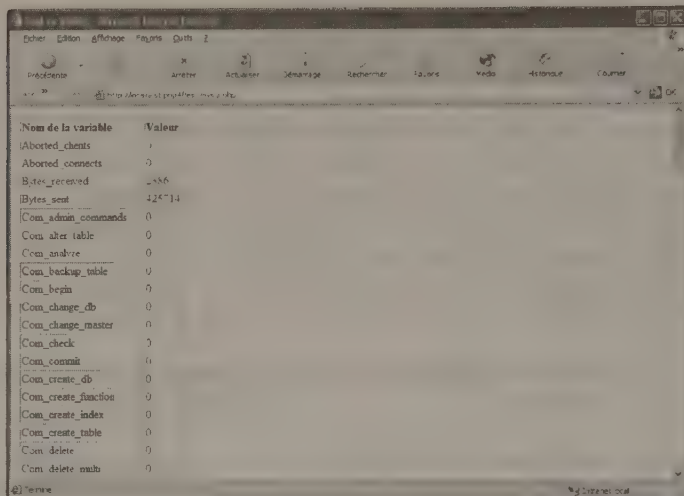
Pour la variable \$user, indiquez votre nom de login, et pour la variable \$password, votre véritable mot de passe. Si votre compte ne demande pas de mot de passe, indiquez tout simplement une chaîne de caractères vide : "".

N'oubliez pas les points-virgules à la fin de chacune de ces trois lignes.

3. Pointez votre navigateur sur le fichier test-mysql.php que vous avez créé à l'étape 1 et éventuellement modifié à l'étape 2.

Vous devriez voir s'afficher le tableau dont la Figure 2.1 vous montre le début. Il contient une longue liste de noms de variables accompagnées de leurs valeurs. Vous ne devriez voir s'afficher aucun message d'erreur. Ne vous préoccupez pas du contenu de ce tableau. La seule chose qui compte est qu'il soit affiché, ce qui vous permettra d'être certain que MySQL fonctionne correctement et qu'il peut dialoguer avec PHP.





Nom de la variable	Valeur
Aborted_clients	0
Aborted_connects	0
Bytes_received	2556
Bytes_sent	424714
Com_admin_commands	0
Com_alter_table	0
Com_analyze	0
Com_backup_table	0
Com_begin	0
Com_change_db	0
Com_change_master	0
Com_check	0
Com_commit	0
Com_create_db	0
Com_create_function	0
Com_create_index	0
Com_create_table	0
Com_delete	0
Com_delete_multi	0

Figure 2.1 :
Ce qui est
affiché par
l'exécution
du script
d'essai test-
mysql.php.

Si un message d'erreur s'affiche, il y a quelque chose qui cloche dans votre installation de MySQL et/ou de PHP.

Les messages d'erreur sont généralement assez clairs. Selon les options que vous aurez choisies, ils seront affichés en anglais ou en français, voire dans un mélange des deux langues. En voici un exemple :

```
MySQL Connection Failed: Access denied for user: 'tartempion'
(Using password: YES)
```

Ce message signifie que votre nom d'utilisateur et/ou votre mot de passe est ou sont incorrects. Vous noterez que, pour des raisons de sécurité, le mot de passe réel n'est pas affiché mais remplacé par YES. Si vous aviez indiqué une chaîne vide en guise de mot de passe, YES serait remplacé par NO.

En cas de message d'erreur, vérifiez soigneusement votre numéro de compte et votre mot de passe. Souvenez-vous qu'il s'agit ici de votre compte MySQL, différent de votre compte de login. Si vous ne trouvez aucune erreur, contactez le département informatique, votre hébergeur ou votre fournisseur d'accès pour leur soumettre le problème. (Les mots de passe et comptes MySQL seront traités au Chapitre 5.)

Chapitre 3

Développement d'une application de base de données sur le Web

Dans ce chapitre :

- Planification de votre application.
- Choix et organisation des données.
- Conception de la base de données.
- Vue d'ensemble de la construction de votre base de données.
- Vue d'ensemble de l'écriture des programmes de votre application.

Développer une application de base de données sur le Web demande bien plus qu'un simple rangement de données dans une base de données MySQL suivi de l'écriture de quelques programmes PHP. La construction de l'application doit être précédée d'une sérieuse planification. Les étapes à suivre sont les suivantes :

- 1. Développer un plan listant les tâches que votre application doit accomplir.**
- 2. Concevoir la base de données nécessaire pour supporter les tâches précédentes.**
- 3. Construire la base de données MySQL en respectant la façon dont elle a été conçue.**

4. Ecrire les programmes PHP qui accomplissent les diverses tâches de l'application.

Ce sont ces quatre étapes que nous allons détailler.

Planification de votre application de base de données sur le Web

Avant même de poser le doigt sur votre clavier pour écrire un programme PHP, vous devez planifier votre application de base de données sur le Web. C'est probablement l'étape la plus importante dans le processus de développement de l'application. Il serait douloureux de découvrir, une fois terminée l'écriture du dernier programme de votre application, que vous avez oublié un détail et qu'il vous faut tout reprendre de zéro. Ce serait également pénible pour votre ordinateur si, de rage, vous le projetiez violemment sur le sol.

Identification des objectifs de l'application

La première étape de la phase de planification est d'identifier avec exactitude les raisons qui justifient le développement de l'application et ce que vous en attendez. Par exemple, votre but principal pourrait être :

- ✓ La collecte de noms et d'adresses auprès d'utilisateurs pour créer une liste de clients.
- ✓ La fourniture d'informations concernant vos produits à des clients potentiels, sous forme d'un catalogue.
- ✓ La vente en ligne de vos produits.
- ✓ L'assistance technique aux gens qui sont déjà en possession de vos produits.

Lorsque vous aurez clairement identifié l'objectif principal de votre application, faites une liste exacte de ce que vous en attendez. Par exemple, si votre objectif est de développer une base de données des noms et adresses de vos clients à des fins de marketing, la liste des tâches est plutôt brève :

- ✓ Proposer un formulaire grâce auquel vos clients potentiels pourront s'identifier.
- ✓ Placer les informations ainsi recueillies dans une base de données.

Si votre objectif est la vente en ligne de vos produits, la liste s'allonge un peu :

- ✓ Proposer à vos clients potentiels des informations sur ce que vous vendez.
- ✓ Les inciter à acheter tel ou tel produit.
- ✓ Proposer un formulaire grâce auquel ils pourront concrétiser leur achat.
- ✓ Proposer une ou plusieurs méthodes de paiement.
- ✓ Valider le paiement pour être certain d'être réellement payé.
- ✓ Envoyer la commande pour exécution au service chargé des expéditions.

A ce niveau, la description des tâches reste dans le domaine des généralités. Vous pouvez accomplir chacune de ces tâches de diverses façons. Aussi faut-il les examiner de près et entrer plus avant dans les détails. Par exemple, si votre objectif est la vente en ligne, la liste précédente doit être ainsi complétée :

- ✓ **Proposer à vos clients potentiels des informations sur ce que vous vendez.**

Afficher une liste par catégorie de produits, chacune de ces catégories étant un lien.

Lorsque le client clique sur un des liens, afficher la liste des produits de cette catégorie. Chaque nom de produit est à son tour un lien.

Lorsqu'un client clique sur le nom d'un produit, afficher la description de ce produit.

- ✓ **Les inciter à acheter tel ou tel produit.**

Proposer des descriptions soigneusement rédigées de chaque produit mettant en lumière ses qualités intrinsèques.

Montrer des images flatteuses de chaque produit.

Proposer une documentation imprimée, disponible en ligne, de ces produits.

Offrir des remises par quantité.

✓ **Proposer un formulaire grâce auquel le client pourra concrétiser son achat.**

Afficher un bouton sur lequel le client potentiel puisse cliquer pour matérialiser son intention d'acquérir le produit.

Proposer un formulaire destiné à collecter toutes les informations nécessaires sur le produit : taille, couleur, etc.

Calculer et afficher le coût total de la commande.

Calculer et afficher le montant des frais d'expédition.

Calculer et afficher le montant de la TVA.

Proposer un formulaire dans lequel le client pourra indiquer son identité et ses adresses de facturation et de livraison.

✓ **Proposer une ou plusieurs méthodes de paiement.**

Afficher un bouton correspondant au mode de paiement choisi.

S'il s'agit d'une carte de crédit, afficher un formulaire dans lequel le client indiquera les renseignements nécessaires.

✓ **Valider le paiement pour être certain d'être réellement payé.**

La méthode usuelle de validation consiste à recourir à un service en ligne spécialisé dans ce contrôle.

✓ **Envoyer la commande au service chargé des expéditions.**

Pour cela, l'envoi d'un e-mail à ce service devrait suffire.

Se préoccuper du client

L'identification de ce que vous voulez que fasse votre application n'est qu'un des aspects de la planification. Vous devez également prendre en considération ce que vos utilisateurs en attendent. Par exemple, si votre objectif est de constituer une liste de clients potentiels à des fins de marketing, vous devez vous demander s'ils ne risquent pas d'être réticents à fournir ces informations.

Votre application doit correspondre à un but réel, tant pour vous que pour vos clients, faute de quoi ils l'ignoreront tout simplement. Pour les inciter à fournir leurs coordonnées, vous devez les motiver et leur faire comprendre le bénéfice qu'ils pourront en retirer. Voici quelques exemples de raisons pouvant inciter un utilisateur à fournir ses nom et adresse :

- ✓ **Recevoir une lettre d'information.** Pour paraître intéressante, cette lettre devra montrer des applications réelles de vos produits et apporter des informations d'actualité. Il ne faut pas qu'elle se contente d'être un instrument de marketing.
- ✓ **Participer à une loterie dotée d'un premier prix attractif.** Qui peut laisser passer une chance de gagner une semaine de vacances au soleil ou une Ferrari ?
- ✓ **Se voir accorder une remise intéressante.** Vous pouvez, par exemple, proposer périodiquement des offres promotionnelles.
- ✓ **Etre informé de la sortie de nouveaux produits ou de mises à jour des anciens dès qu'elles sont disponibles.** S'il s'agit de logiciels, les clients seront intéressés par l'existence d'une nouvelle version ou de correctifs qu'ils pourront télécharger.
- ✓ **Avoir accès à des informations réellement utiles.** Vous pouvez, par exemple, leur proposer un abonnement gratuit à un journal spécialisé.

Vous pouvez maintenant compléter votre liste de tâches initiale qui était, rappelons-le :

- ✓ Proposer un formulaire grâce auquel vos clients pourront s'identifier.

- ✓ Placer les informations ainsi recueillies dans une base de données.

En prenant en compte le point de vue du client, cette liste devient :

- ✓ Présenter une description des avantages dont bénéficieront les clients qui accepteront de donner leur identité.
- ✓ Proposer un formulaire grâce auquel vos clients pourront s'identifier.
- ✓ Ajouter les coordonnées du client à la liste des abonnés à votre lettre d'information.
- ✓ Placer les informations ainsi recueillies dans une base de données.

De tout cela découle l'ébauche d'un plan pour votre application de base de données sur le Web. Vous-même et vos clients pourrez retirer un réel bénéfice de cette application.

Faciliter l'utilisation de votre site Web

Faciliter l'utilisation de votre application fait appel à plusieurs techniques comme :

- ✓ **Navigation.** Tout client doit pouvoir accéder facilement à telle ou telle partie de votre site Web.
- ✓ **Graphisme.** "Un court croquis vaut un long discours." Les images et les schémas accroissent le caractère attractif d'une page Web. N'oubliez pas, cependant, que le temps de chargement d'une image est assez important.
- ✓ **Accès.** Certains choix dans la présentation d'une page peuvent contribuer à en faciliter la perception par les visiteurs souffrant de troubles visuels.
- ✓ **Navigateurs.** Différents navigateurs (et même différentes versions d'un navigateur donné) peuvent afficher la même page d'une façon différente.

Ces considérations sont importantes, mais en dire davantage sur ce sujet dépasse les objectifs de ce livre. Si vous lisez l'anglais technique, vous tirerez bénéfice de la consultation des deux sites Web suivants conçus par des experts de la question :

✓ Jakob Nielsen : <http://www.useit.com>.

✓ Jarod Spool : <http://world.std.com/~uiweb>.

Enfin, reportez-vous au livre de Rhonda Crowder *Concevoir et créer un site Web*, paru dans la même collection, chez le même éditeur.

Prévoir de la place pour les extensions

Vous pouvez être certain que votre application évoluera au cours du temps. Vous pouvez, par exemple, imaginer de nouvelles fonctionnalités ou simplement modifier quelques éléments dans sa composition actuelle. Ou bien les logiciels généraux évolueront, vous offrant de nouvelles perspectives. Aussi, lorsque vous planifiez votre application, prévoyez ces possibles évolutions.

Vous pouvez concevoir votre application comme une succession d'étapes qui tiendraient compte de cette évolution. Vous pouvez établir un plan correspondant à vos préoccupations actuelles en conservant la possibilité d'en faire davantage plus tard, lorsque cette première étape sera réalisée de façon satisfaisante. Vous pouvez aussi la compléter au fur et à mesure que vous pourrez développer les programmes nécessaires. Par exemple, vous pouvez publier le catalogue de vos produits sur le Web dès qu'il est prêt. Puis vous pourrez commencer à travailler sur une fonction de prise de commande en ligne que vous incorporerez à l'application lorsqu'elle sera écrite et testée.



Vous n'êtes pas obligé de tout prévoir au départ. Si vous vous occupez d'une agence de voyages, vous pouvez vous limiter actuellement aux destinations sur la Terre et reporter à plus tard l'ouverture des pages concernant des vacances sur Mars ou Alpha du Centaure. En d'autres termes, plus votre application sera souple, et plus il sera facile de la faire évoluer dans le futur.

Ecrivez

Ecrivez votre plan sur papier. Je vous le répéterai souvent, car j'ai fait la douloureuse expérience d'écarter ce "détail". Lorsque vous développez un plan, il est très présent dans votre esprit et parfaitement clair. Mais au bout de quelques semaines, vous serez surpris de découvrir que certains de ces aspects sont devenus flous, d'autres préoccupations ayant mobilisé votre attention. Ou bien

vous souhaiterez, au bout d'un temps plus ou moins long, apporter certains changements au plan initial. Mieux vaudra alors savoir au plus juste comment s'articule l'application. Autre hypothèse : vous travaillez avec un partenaire pour développer cette application et découvrez que ce dernier a compris de travers vos explications depuis le début et développé des fonctionnalités qui n'entrent pas dans vos objectifs. Fixer votre plan par écrit sur le papier vous évitera ces écueils.

Conception de la base de données

Lorsque vous avez déterminé exactement ce que doit faire votre application de base de données sur le Web, vous êtes prêt pour étudier la conception de la base de données qui contiendra les informations utilisées par l'application. A ce stade, vous devez identifier les informations dont vous avez besoin et les organiser de la façon imposée par le logiciel de cette base de données.

Choix des données

Vous devez commencer par identifier les informations à conserver dans la base de données. En regardant la liste des tâches à accomplir, vous allez pouvoir les identifier.

Voici quelques exemples :

- ✓ Un catalogue de produits demande une base de données contenant des informations sur les produits.
- ✓ Une application de prise de commande en ligne demande une base de données pouvant contenir des informations sur le client et sur sa commande.
- ✓ Une agence de voyages a besoin d'une base de données contenant des informations sur toutes les destinations qu'elle propose, les moyens de transport, les réservations, les tarifs, les horaires...

Dans beaucoup de cas, votre application pourra contenir une tâche d'identification du client. Vous devrez trouver un juste équilibre entre votre souci de collecter le plus d'informations possible et la réticence que peut éprouver un client à s'identifier avec trop de précision. Evitez les formulaires qui contiennent trop de rubriques

et demandent donc trop de temps pour les renseigner. Un bon compromis consiste à considérer certaines informations comme facultatives et à signaler (par exemple au moyen d'une couleur différente ou d'un astérisque) celles qui doivent obligatoirement être fournies. Ce sera au client de décider quels sont les champs qu'il accepte de renseigner. Vous pouvez aussi imaginer un moyen d'incitation poussant le client à renseigner plus de rubriques qu'il ne le ferait spontanément. Offrir par exemple une loterie dont les prix ont une valeur intéressante constitue à l'évidence un motif supplémentaire de motivation.

Dans le premier exemple d'application, vos clients vont consulter le catalogue en ligne des animaux pour choisir celui qu'ils envisagent d'acheter. Vous voulez qu'ils aient sous les yeux des informations qui les incitent à acheter. Pour cela, voici ce qu'ils devraient pouvoir trouver :

- ✓ Le type d'animal : caniche, licorne, etc.
- ✓ Une description de l'animal.
- ✓ Une image de l'animal.
- ✓ Le prix de l'animal.

Dans le second exemple (la section à accès réservé), vous voulez enregistrer des informations concernant les membres. Par exemple :

- ✓ Nom.
- ✓ Adresse.
- ✓ Numéro de téléphone.
- ✓ Numéro de fax.
- ✓ Adresse e-mail.



Prenez le temps de réfléchir à toutes les informations dont vous estimez avoir besoin pour garnir votre base de données. Bien que vous puissiez modifier plus tard le nombre et le type des informations contenues dans la base de données et ajouter d'autres données, mieux vaut y penser dès le départ. Faute de quoi, il y aurait des informations manquantes pour les premiers membres enregistrés. Si, par exemple, vous décidez, à un certain moment, de demander en plus l'âge de vos clients, vous ne connaîtrez pas celui des premiers à s'être enregistrés.

Organisation des données

MySQL est un *SGBD relationnel*, ce qui signifie que les données qu'il conserve y figurent sous forme de tables. Il existe des relations entre les tables qui sont dans la base de données.



Tables ou tableaux ? Le mot anglais "table" peut être traduit par l'un ou par l'autre. Alors que, dans un livre ou un document HTML, on parle presque toujours de "tableau", dans le domaine des bases de données, "table" s'est imposé. C'est ce qu'on peut constater dans les ouvrages publiés, écrits directement dans notre langue. C'est le choix que nous avons fait pour cette traduction. (N.d.T.)

Organisation des données sous forme de tables

Les tables de bases de données sont organisées comme des tableaux : en lignes et en colonnes, ainsi que vous pouvez le voir sur la Figure 3.1. La cellule (la donnée) située à l'intersection d'une ligne et d'une colonne est appelée *champ*.

	Colonne 1	Colonne 2	Colonne 3	Colonne 4
Ligne 1				
Ligne 2				
Ligne 3				
Ligne 4				
Ligne 5				

Figure 3.1 : Les données utilisées par MySQL sont organisées sous forme de tables.

Le rôle d'une table est de renfermer des informations sur un objet particulier, quel qu'il soit. Voici une liste de quelques objets pouvant être concernés par ce concept de *table* :

- ✓ Clients
- ✓ Produits

- ✓ Entreprises
- ✓ Animaux
- ✓ Villes
- ✓ Pièces d'habitation
- ✓ Livres
- ✓ Ordinateurs
- ✓ Profils
- ✓ Documents
- ✓ Projets
- ✓ Semaines

Vous devez créer une table pour chaque objet et le nom de cette table doit désigner clairement et sans ambiguïté l'objet qu'elle représente. Ce nom doit être écrit en un seul mot, sans espaces intercalés. En général, on adopte le singulier pour ce nom. En conséquence, une table contenant des informations sur les clients pourrait s'appeler **Client**. Une table contenant des informations sur les commandes d'un client s'appellerait alors **CommandeClient**.



Linux et Unix font la différence entre majuscules et minuscules, ce qui n'est pas le cas de Windows. L'échec ou la réussite peuvent tenir à ce genre de détail !

Dans le jargon des bases de données, un objet est appelé *entité*. Une entité possède des *attributs*. Dans la table, chaque ligne représente une entité, et chaque colonne un attribut. Ainsi, dans une table de clients, chaque ligne contient des informations concernant un client particulier, alors que chaque colonne contient le même type d'informations pour tous les clients. Exemple d'attributs : nom, prénom, âge, numéro de téléphone...

L'organisation de vos données en tables s'effectue en parcourant les étapes suivantes :

1. Donnez un nom à votre base de données.

Ce nom, comme je l'ai dit plus haut, doit être significatif. Par exemple, la base de données utilisée par un libraire contenant des informations sur les livres qu'il vend s'appellera tout

simplement **Livre**. Evitez les noms trop longs dans l'écriture desquels vous pourriez faire des fautes de frappe.

2. Identifiez les objets.

Considérez la liste des informations que vous voulez conserver dans la base de données et analysez-la pour identifier les objets. Dans l'exemple de la table **Livre**, ces objets pourraient être :

- Auteur
- Titre
- Date de publication
- Prix
- Editeur
- Quantité en stock

Ce faisant, vous pourrez généralement constater que vous conservez des informations sur plusieurs objets : un auteur particulier et un de ses ouvrages, un éditeur, par exemple. Ou encore : tous les membres d'une famille possèdent la même adresse, mais ils n'ont pas la même chambre...

3. Définissez et nommez une table pour chaque objet.

Pour reprendre l'exemple précédent, vous pourriez avoir une table **Auteur** et une table **Editeur**.

4. Identifiez les attributs pour chaque objet.

Analysez votre liste d'informations et identifiez les attributs dont vous aurez besoin pour chaque objet. Eclatez les informations à conserver en morceaux raisonnablement petits. Par exemple, lorsque vous voulez conserver le nom d'une personne dans une table, vous pouvez le décomposer en deux attributs : nom propre et prénom. De cette façon, vous pourrez ensuite faire facilement un tri sur le nom propre, ce qui serait plus difficile si l'attribut contenait, dans cet ordre, le prénom suivi du nom propre. Les Américains, qui utilisent couramment jusqu'à trois prénoms, peuvent pousser la décomposition plus loin. Chez nous, il est rare de s'identifier par tous ses prénoms. Tout au plus peut-on avoir un prénom

composé (Jean-Claude, Paul-Emile, Anne-Aymone...) qu'il faut alors traiter comme une entité unique.

5. Définissez les colonnes et donnez-leur un nom.

Vous devez le faire pour chaque article d'information identifié à l'étape précédente. Ici encore, choisissez un nom de colonne significatif et suffisamment bref. La syntaxe est toujours la même : un seul mot et pas d'espace. Exemples : Nom, Nom_propre, Prénom...

Certains mots sont réservés par SQL (et donc aussi par MySQL) pour son propre usage (ou pour des extensions futures). Ils ne peuvent pas être utilisés comme noms de colonnes. En voici quelques exemples : ADD, ALL, AND, CREATE, DROP, GROUP, ORDER, RETURN, SELECT, SET, TABLE, USE, WHERE. Et il y en a bien d'autres ! Vous trouverez une liste complète des mots réservés dans le manuel en ligne de MySQL, à l'adresse http://www.mysql.com/doc/en/Reserved_words.html.

6. Identifiez la clé primaire.

Chaque ligne d'une table doit avoir un identificateur unique, ce qui empêche deux lignes ou plus d'avoir le même nom. Lors de la conception de la table, vous devez choisir parmi les colonnes celle ou celles qui contiendront l'identificateur unique qui va devenir la *clé primaire*. Dans la plupart des cas, les attributs d'un objet ne peuvent avoir un identificateur unique, parce que, par exemple, deux clients peuvent avoir le même nom. Mais si vous associez le prénom au nom, vous diminuez ce risque de collision. Dans la réalité, on utilise plutôt comme clé primaire une numérotation des colonnes. Dans notre exemple, cet identificateur pourrait être constitué par le "code client", numéro unique attribué par le logiciel au moment de l'enregistrement du client. De cette façon, on ne risque pas d'avoir deux codes identiques.

7. Définissez des valeurs par défaut.

Vous pouvez définir une valeur par défaut qui sera assignée à un champ lorsque aucune valeur ne lui aura été spécifiquement attribuée. Ce n'est pas indispensable, mais souvent utile. Par exemple, si votre application inclut le pays parmi les champs qui définissent une adresse, vous pouvez définir le champ "pays" comme ayant la valeur "France" par défaut.



Si rien n'a été spécifié pour ce champ, il prendra automatiquement cette dernière valeur.

8. Identifiez les colonnes qui doivent obligatoirement contenir des informations.

Vous pouvez spécifier que certaines colonnes doivent obligatoirement être renseignées, ce qui implique qu'elles ne peuvent pas recevoir de valeur vide (ou NULL). C'est le cas, par exemple, pour la colonne désignée comme clé primaire. Tout champ de ce type laissé vide produirait un message d'erreur au moment de son enregistrement.

Les bases de données bien conçues ne conservent chaque élément d'information qu'en un seul exemplaire, à un endroit unique. Agir autrement est inefficace et peut créer des problèmes lorsque cette information doit être déplacée, car si vous ne la déplacez que d'un seul des endroits qu'elle occupe, laissant l'autre ou les autres intacts, votre base de données risque fort de connaître de sérieuses difficultés.



Si vous constatez que vous rangez la même donnée dans plusieurs lignes, vous avez probablement besoin de réorganiser vos tables. Par exemple, supposons que vous traitiez des informations concernant des livres, informations parmi lesquelles figure l'adresse de l'éditeur. Lorsque vous placez les données dans la base, vous vous rendez compte que vous allez ranger cette adresse à plusieurs endroits, pour chaque livre publié par cet éditeur. Il serait bien plus efficace de ranger tout ce qui concerne le livre proprement dit dans une table *Livre* (titre, auteur, date de publication, prix) et de constituer une seconde table, *Editeur*, qui ne contiendrait que des éléments concernant les éditeurs (nom, adresse, numéro de téléphone...). Un pointeur établirait alors une relation entre une entrée de la table *Livre* et la table *Editeur*, comme nous allons le montrer dans la section suivante.

Création de relations entre les tables

Certaines tables d'une base de données ont des relations avec d'autres tables. Le plus souvent, il existe une relation entre une ligne d'une certaine table et plusieurs lignes d'une autre. Il est nécessaire de consacrer une colonne à l'établissement de ces relations entre les lignes de ces différentes tables. Dans la plupart des cas, une colonne

d'une table contiendra une information (un pointeur) correspondant à la clé primaire d'une autre table.

Pour illustrer ce type de relation, considérons une application dans laquelle une base de données renferme des clients et leurs commandes. Une des tables contiendra les informations "Client" (nom, adresse...). Chaque client peut avoir commandé un nombre quelconque d'articles. Dans la seconde table, celle des commandes, on trouvera donc, pour un client donné, de 0 à n entrées (une par commande). Une colonne de cette table "Commande" contiendra une information d'identification pour un client donné (normalement, la clé primaire de celui-ci), ce qui permettra d'établir un lien entre les deux tables. La Figure 3.2 illustre l'agencement de la table des clients.

ID_client	Prénom	Nom	Téléphone
27895	Jean	Dupont	0123456789
44555	Arthur	Martin	0455667788
23695	Georgette	Durand	02999888777
27822	Kevin	Dupond	0666666668
29844	Emma	Bovary	0233445566

Figure 3.2 :
Extrait de la
table des
clients.

La Figure 3.3 montre un extrait de la table des commandes correspondant à l'extrait de la table des clients illustré par la Figure 3.2. Vous remarquerez la correspondance entre les colonnes ID_client des deux tables, établissant ainsi le lien nécessaire entre les commandes de chaque client et le client concerné.

Dans cet exemple, les colonnes qui créent la relation entre les deux tables ont le même nom, mais ce n'est pas un impératif. Ce qui compte, c'est que leur contenu soit identique.

No_cde	ID_Client	No_article	Prix
87-222	27895	chat_3	200
87-223	44555	chat-4	225
87-224	23695	cheval_1	550
87-225	27822	chien_27	210
87-226	29844	oiseau_1	50

Figure 3.3 :
Extrait de la
table des
commandes.

Conception des deux bases de données d'exemple

Dans les deux sections qui suivent, nous verrons comment sont conçues les deux bases de données pour les deux exemples proposés.

Catalogue d'animaux (AniCata)

Vous voulez afficher la liste des informations suivantes lorsque le client consulte le catalogue des animaux :

- ✓ Le type d'animal : caniche, licorne, etc.
- ✓ Une description de l'animal.
- ✓ Une image de l'animal.
- ✓ Le prix de l'animal.

Chaque animal doit être classé dans sa catégorie, et celle-ci comporter autant d'entrées qu'il y a d'animaux. Il en découle l'exécution des étapes suivantes, d'après le schéma général que nous avons expliqué plus haut :

1. Donnez un nom à votre base de données.

Ce sera **AniCata** (contraction de **AnimalCatalogue**).

2. Identifiez les objets.

Voici quelle est la liste des informations à entrer pour chaque animal :

- Le type d'animal : caniche, licorne, etc.
- Une description de l'animal.
- Une image de l'animal.
- Le prix de l'animal.
- La catégorie de l'animal.

Toutes ces informations concernent un animal. Aussi, le seul objet ici présent est-il "animal".

3. Définissez et nommez une table pour chaque objet.

Ce sera tout simplement `Animal`.

4. Identifiez les attributs pour chaque objet.

Voyons de plus près la nature des informations à saisir :

- **Type de l'animal.** Un seul attribut : caniche, licorne... Cependant, il semble évident que plusieurs caniches sont à vendre dans votre boutique en même temps. Il vous faut donc un identificateur unique comme clé primaire.
- **Numéro d'identification de l'animal.** Ce numéro unique est affecté de façon séquentielle lorsqu'un nouvel animal est ajouté à la table. Ce sera notre clé primaire.
- **Description de l'animal.** Ici, nous avons affaire à deux attributs : la description écrite de l'animal et sa couleur.
- **Image de l'animal.** Ce sera un pointeur vers une image de l'animal.
- **Prix de l'animal.** Prix de vente exprimé en euros.

- **Catégorie de l'animal.** Ici encore, deux attributs : le nom de la catégorie (chien, cheval, dragon...) et une description générale de cette catégorie.

Il serait peu efficace d'inclure deux types d'informations dans cette table :

- Les informations concernant la catégorie contiennent une description de celle-ci. Comme chaque catégorie contient plusieurs animaux, inclure la description de la catégorie dans la table `Animal` entraînerait sa duplication dans plusieurs lignes. Il est bien plus efficace de définir une catégorie d'animal comme un objet ayant sa propre table.
- S'il existe plusieurs couleurs pour un animal, toutes les informations seront répétées dans une ligne séparée pour chaque couleur. Ici encore, il est plus efficace de créer une autre table : celle des couleurs d'animaux.

Ces deux nouvelles tables s'appelleront `Type` et `Couleur`.

5. Définissez les colonnes et donnez-leur un nom.

La table `Animal` contient une ligne pour chaque animal. Les noms des colonnes sont :

- **`animalID`.** L'identificateur de chaque animal doit être unique.
- **`animalNom`.** Définit le nom de l'animal.
- **`animalType`.** Le nom de la catégorie de l'animal. C'est la colonne qui va établir une relation avec l'entrée appropriée de la table `Type`.
- **`animalDesc`.** La description de l'animal.
- **`animalPrix`.** Le prix de l'animal.
- **`animalImage`.** Le nom du fichier d'image contenant une photo ou un dessin de l'animal.

La table `Type` possède une ligne pour chaque catégorie d'animal et chaque ligne contient les deux colonnes suivantes :

- **animalType**. Le nom de la catégorie de l'animal. C'est la colonne qui va établir une relation avec l'entrée appropriée de la table `Animal` vue ci-dessus.
- **typeDesc**. La description du type.

La table `Couleur` a une entrée pour chaque couleur d'animal et chaque ligne contient les trois colonnes suivantes :

- **animalNom**. Le nom de l'animal. C'est cette colonne qui établira la relation entre la ligne de la couleur et la ligne appropriée de la table `Animal`.
- **animalCouleur**. La couleur de l'animal.
- **couleurImage**. Le nom du fichier d'image contenant une photo ou un dessin de l'animal dans la couleur considérée.

6. Identifiez la clé primaire.

- La clé primaire de la table `Animal` est `animalID`.
- La clé primaire de la table `Type` est `animalType`.
- La clé primaire de la table `Couleur` est formée de l'association de `animalNom` et `animalCouleur`.

7. Définissez des valeurs par défaut.

Aucune valeur par défaut n'est définie pour ces tables.

8. Identifiez les colonnes qui doivent obligatoirement contenir des informations.

Les colonnes suivantes doivent toujours être renseignées :

- `animalID`
- `animalNom`
- `animalCouleur`
- `animalType`

Ces colonnes sont celles qui sont utilisées pour les clés primaires. Une ligne qui ne possède pas de valeurs pour ces attributs ne doit pas se trouver dans une table.

Base à accès réservé (MembresSeuls)

Vous avez établi la liste d'informations ci-après qui représente les informations à exploiter lorsque les clients veulent accéder à la section à accès réservé de votre site Web :

- ✓ Nom.
- ✓ Adresse.
- ✓ Numéro de téléphone.
- ✓ Numéro de fax.
- ✓ Adresse e-mail.

En outre, vous aimeriez mémoriser la date à laquelle chaque membre s'est enregistré, ainsi que la date de chacune de ses consultations de la section à accès réservé.

La base de données utilisée dans cette section va être créée en suivant les étapes énumérées ci-après, d'après les règles générales que nous avons vues plus haut.

1. Donnez un nom à votre base de données.

Ce sera **MembresSeuls**.

2. Identifiez les objets.

Voici quelle est la liste des informations à entrer pour chaque membre :

- Nom
- Adresse
- Numéro de téléphone
- Numéro de fax
- Adresse e-mail
- Date d'enregistrement
- Enregistrement des accès à cette section

Toutes ces informations concernent un membre déterminé. Aussi le seul objet de cette liste est-il "membre".

3. Définissez et nommez une table pour chaque objet.

Ce sera tout simplement *Membre*.

4. Identifiez les attributs pour chaque objet.

Voyons de plus près la nature des informations à saisir :

- **Nom.** Deux attributs : nom propre et prénom.
- **Adresse.** Quatre attributs : adresse proprement dite (nom de la rue et numéro), ville, département et code postal. Comme actuellement tous vos clients sont domiciliés en France, nous supposons que ces informations sont exprimées conformément aux règles en usage dans notre pays.
- **Numéro de téléphone.** Un seul attribut (un nombre de dix chiffres).
- **Numéro de fax.** Un seul attribut (un nombre de dix chiffres).
- **Adresse e-mail.** Un seul attribut.
- **Date d'enregistrement.** Un seul attribut.

Ici, plusieurs informations sont en relation avec chaque accès :

- L'accès à la section réservée requiert un nom de login et un mot de passe qui doivent être conservés dans la base.
- La façon la plus simple de conserver trace des accès est d'enregistrer la date et l'heure de chaque accès dans la base.

Comme chaque membre peut visiter autant de fois qu'il le souhaite la section à accès réservé, plusieurs groupes date/heure devront pouvoir être enregistrés. En conséquence, plutôt que de définir ce groupe date/heure comme étant un attribut de l'objet membre, mieux vaut le considérer comme un objet à part entière, relié au membre mais disposant de sa propre table.

Cette nouvelle table prendra le nom *Login*. L'attribut de l'objet login est le groupe date/heure du login.

5. Définissez les colonnes et donnez-leur un nom.

La table `Membre` contient une ligne pour chaque membre. Les noms des colonnes sont :

- **login.** Nom de login.



Chaque nom de login doit être unique. Les programmes de l'application doivent vérifier cette unicité.

- **mPasse.** Mot de passe.
- **initial.** Date de l'enregistrement du membre.
- **prénom.**
- **nom.**
- **rue.** Nom de la rue et numéro.
- **ville.**
- **département.**
- **codePostal.**
- **email.**
- **tph.** Numéro de téléphone.
- **fax.** Numéro de fax.

La table `Login` possède une entrée pour chaque login. Elle contient les trois colonnes suivantes :

- **login.** C'est le nom de login du membre. Cette colonne est celle qui établira un lien avec la colonne de même nom de la table `Membre`. Alors que cette valeur est unique pour cette table, elle peut figurer plusieurs fois dans la table `Login`.
- **date.** C'est le groupe date/heure du login considéré.

6. Identifiez la clé primaire.

- La clé primaire de la table `Membre` est `login`.
- La clé primaire de la table `Login` est formée de l'association de `login` et de `date`.

7. Définissez des valeurs par défaut.

Aucune valeur par défaut n'est définie pour ces tables.

8. Identifiez les colonnes qui doivent obligatoirement contenir des informations.

Les colonnes suivantes doivent toujours être renseignées :

- login
- mPasse
- date

Ces colonnes sont celles qui sont utilisées pour les clés primaires. Une ligne qui ne posséderait pas de valeurs pour ces attributs ne doit pas se trouver dans une table.

Type de données

MySQL conserve les informations selon différents formats dépendant du type de ces informations tel que vous le déclarez dans les tables. Les principaux types sont : chaîne de caractères, numérique et date/heure.

Chaîne de caractères

C'est le type le plus courant : noms, adresses, numéros de téléphone ou de fax, descriptions... Il est représenté par une suite de caractères quelconques et ne peut être manipulé que sous cette forme. Une chaîne de caractères peut être déplacée, comparée, affichée, imprimée et concaténée (ou accolée) à une ou plusieurs autres. Il est possible de définir une sous-chaîne formée d'une suite de caractères consécutifs extraits d'une chaîne donnée. Une chaîne peut être remplacée par une autre.

Une chaîne de caractères peut avoir une longueur fixe ou variable. Dans le premier cas, MySQL réserve un emplacement de longueur fixe pour la chaîne. Si celle-ci s'avère plus longue, seuls les premiers caractères en seront conservés, le reste étant perdu. Si elle est plus courte, elle sera complétée à droite par des espaces.

Dans le cas d'un format de longueur variable, la chaîne est rangée dans un champ qui a la même longueur. Vous continuez à spécifier

une taille, mais aucun espace n'est ajouté si la chaîne est plus courte. Par contre, les caractères supplémentaires sont toujours perdus si la chaîne est plus longue que ce qui est attendu.

Lorsque la longueur d'une chaîne varie peu, le format de longueur fixe est préférable. Dans le cas contraire, il est évident qu'il faut choisir une longueur variable. Ce serait le cas, par exemple, pour le champ Titre d'un livre.

Numérique

C'est un type de données également courant. On peut avoir affaire à des nombres décimaux *réels* (10.5, 2.34567, -45.91) ou à des *entiers* (-5, 234, 0). Une valeur stockée sous forme numérique peut intervenir dans des calculs classiques. Si une information ne doit pas être utilisée dans des expressions arithmétiques ou mathématiques, il est préférable de l'enregistrer sous forme de chaîne de caractères. C'est le cas, par exemple, pour un code postal, un numéro de téléphone ou de fax.



Rappelez-vous qu'en notation anglo-saxonne (celle qu'utilise MySQL) la partie fractionnaire et la partie entière d'un nombre décimal sont séparées par un point (.) et non par une virgule (,).

Les nombres peuvent être positifs, négatifs ou nuls, mais vous pouvez spécifier que tel ou tel champ ne doit contenir que des nombres positifs. On a alors affaire à des nombres sans signe (*unsigned*). Un bon exemple en est le nombre d'habitants d'une ville ou le nombre de pages d'un livre.

MySQL définit un type numérique spécial à auto-incrémentation. Une colonne de ce type contient d'office une valeur numérique entière croissante si aucune valeur n'est fournie. Si vous insérez un enregistrement en indiquant la valeur 5 dans cette colonne, le prochain enregistrement créé recevra a priori le numéro 6 (sauf si vous en fournissez un, mais ce n'est pas le but). Les colonnes à auto-incrémentation sont très utiles par exemple pour générer des numéros de référence de produits ou de commandes sans risquer de doublon.

Date/heure

C'est un autre type de données, toutefois un peu moins utilisé. Une information conservée sous forme de groupe date/heure peut être

affichée sous différents formats. Elle peut également être utilisée pour déterminer l'intervalle entre deux dates ou deux heures.

Énumération

Il peut arriver qu'un champ ne puisse recevoir qu'un nombre limité de valeurs. Par exemple : "oui" ou "non". Pour cela, MySQL propose le type *énumération*. Il faut indiquer dans la déclaration du champ la liste des valeurs acceptables et MySQL vérifiera qu'aucune autre valeur n'est placée dans cette colonne.

Nom des types de données reconnus par MySQL

Lorsqu'on crée une base de données, on déclare quels seront les types de valeurs convenant aux diverses colonnes. Le Tableau 3.1 montre la liste des types de variables les plus utilisés dans les applications de bases de données sur le Web.

Il existe d'assez nombreux autres types de données dont l'emploi est moins fréquent. Pour une description complète de tous les types de données possibles, faites une recherche sur le Web.

Ecrivez-le !

A nouveau ma petite ritournelle : écrivez tout ça ! Vous avez sans doute passé pas mal de temps à déterminer la structure et le contenu de votre base de données. Alors, maintenant que tout cela est clair et (provisoirement) définitif dans votre esprit, même si vous croyez que vous ne pourrez jamais l'oublier, notez tout cela sur des feuilles de papier. Rédigez tout ce dont vous aurez besoin pour n'oublier aucun détail.

Documentez l'organisation des tables, les noms des colonnes et tous vos choix. Une bonne façon de le faire est de décrire chaque table dans un document séparé, sous la forme d'un tableau avec une ligne pour chaque colonne et une colonne pour chacun des choix effectués. Les colonnes pourraient, par exemple, s'appeler : *nom de colonne*, *type de données* et *description*.

Revenons à nos deux exemples

Dans cette section, nous allons revoir les choix effectués pour nos deux exemples et dresser l'inventaire du contenu des tables.

Tableau 3.1 : Types de données reconnus par MySQL.

Type de données	Description
CHAR(longueur)	Chaîne de caractères de longueur fixe.
VARCHAR(longueur)	Chaîne de caractères de longueur variable (255 caractères au plus).
TEXT	Chaîne de caractères de longueur variable (65 535 caractères au plus).
INT(longueur)	Entier compris entre -2147483648 et +2147483647. La longueur du nombre pouvant être affiché est déterminée par longueur. Ainsi, si longueur vaut 4, seuls les nombres compris entre -999 et +999 pourront être affichés, même si la valeur conservée est plus grande.
INT(longueur) UNSIGNED	Nombre entier compris entre 0 et 4294967295. longueur représente la taille maximale du nombre pouvant être affiché. Ainsi, si longueur vaut 4, seuls les nombres compris entre 0 et 9999 pourront être affichés, même si la valeur conservée est plus grande.
BIGINT	Grand nombre entier. S'il est déclaré signé, la plage va de -9223372036854775808 à +9223372036854775807. S'il est déclaré non signé, la plage va de 0 à 18446744073709551615.
DECIMAL(longueur, dec)	Nombre décimal. longueur représente le nombre de caractères pouvant être utilisés pour l'affichage, y compris le signe, l'exposant et le point décimal. dec représente le nombre de chiffres de la partie fractionnaire. Par exemple, pour 12.34, longueur vaut 5 et dec vaut 2.
DATE	Date (année, mois, jour) selon le découpage AAAA-MM-JJ. Par exemple : 2002-05-17.
TIME	Heure (heures, minutes, secondes) selon le découpage hh:mm:ss.
DATETIME	Groupe date/heure selon le découpage AAAA-MM-JJ hh:mm:ss.
ENUM("val1", "val2",...)	Liste des seules valeurs que peut prendre le champ 65 535.
SERIAL	Abréviation remplaçant la suite de mots réservés très utilisée BIGINT UNSIGNED NOT NULL AUTO_INCREMENT.

Base AniCata (vente en ligne)

La base de données prévue pour le catalogue de l'application de vente en ligne d'animaux contient trois tables : *Animal*, *Type* et *Couleur*. Les Tableaux 3.2 à 3.4 montrent comment sont organisées ces tables. Ces structures ne sont pas ainsi figées pour l'éternité. MySQL est suffisamment souple pour autoriser des modifications de structure. Si, initialement, vous avez fixé la longueur d'une chaîne de caractères à 20 caractères et que vous découvrez un peu plus tard que c'est trop court, vous pourrez facilement revoir cette définition.

Structure de la base AniCata

Tableau 3.2 : Contenu de la table *Animal*.

Nom de la variable	Type	Description
animalID	SERIAL	Numéro unique de l'animal. Clé primaire.
animalNom	CHAR(25)	Nom de l'animal.
animalType	CHAR(15)	Catégorie de l'animal.
animalDesc	VARCHAR(255)	Description de l'animal.
animalPrix	DECIMAL(9,2)	Prix de l'animal.
animalImage	CHAR(15)	Pointeur vers l'image de l'animal.

Tableau 3.3 : Contenu de la table *Type*.

Nom de la variable	Type	Description
animalType	CHAR(15)	Catégorie de l'animal. Clé primaire.
typeDesc	VARCHAR(255)	Description de la catégorie.

Tableau 3.4 : Contenu de la table *Couleur*.

Nom de la variable	Type	Description
animalNom	CHAR(25)	Nom de l'animal (clé primaire 1).
animalCouleur	CHAR(15)	Nom de la couleur (clé primaire 2).
couleurImage	CHAR(15)	Nom du fichier d'image de l'animal dans la couleur considérée.

Base MembresSeuls (accès réservé)

La base de données prévue pour la section à accès réservé contient deux tables appelées *Membre* et *Login*. Les Tableaux 3.5 et 3.6 en présentent l'organisation. Ces structures ne sont pas figées pour l'éternité. MySQL est suffisamment souple pour autoriser des modifications de structure. Si, initialement, vous avez fixé la longueur d'une chaîne de caractères à 20 caractères et que vous découvrez un peu plus tard que c'est trop court, vous pourrez facilement revoir cette définition.

Structure de la base MembresSeuls

Tableau 3.5 : Contenu de la table Membre.

Nom de la variable	Type	Description
login	VARCHAR(20)	Nom de login (clé primaire).
mPasse	CHAR(255)	Mot de passe.
initial	DATE	Date enregistrement initial.
nom	VARCHAR(50)	Nom propre.
prénom	VARCHAR(40)	Prénom.
rue	VARCHAR(50)	Nom de la rue et numéro dans la rue.
ville	VARCHAR(50)	Ville de résidence.
département	CHAR(2)	Code du département.
codePostal	CHAR(10)	Code postal.
mail	VARCHAR(50)	Adresse e-mail.
tph	CHAR(15)	Numéro de téléphone.
fax	CHAR(15)	Numéro de fax.

Tableau 3.6 : Contenu de la table Login.

Nom de la variable	Type	Description
login	CHAR(20)	Nom de login (clé primaire 1).
date	DATETIME	Groupe date/heure des logins (clé primaire 2).

Développement de l'application

Maintenant que sont définies les structures de chacune des tables des deux bases de données et que votre plan contient la liste des tâches que devra accomplir l'application, vous êtes prêt pour créer l'application. Vous allez d'abord construire la base, puis écrire vos programmes PHP. Mais vous êtes encore loin d'une application pleinement fonctionnelle ! Allons, j'exagère : vos progrès sont déjà sensibles.

Construction de la base de données

Cette tâche consiste à transformer la description sur papier de la base en une suite de déclarations que MySQL sera à même de comprendre. Cette phase est indépendante de la programmation PHP, car la base de données qui va en résulter pourrait tout aussi bien être utilisée par des programmes écrits dans d'autres langages : C, Perl, Java... La base de données forme un tout.



Il faut créer la base avant d'entamer la programmation PHP, car les scripts écrits dans ce langage sont destinés à manipuler les informations qu'elle contient. Vous ne pourrez donc pas les tester tant que la base n'existera pas.

Les déclarations qui vont matérialiser la base de données doivent être écrites en SQL pour être comprises par MySQL. Vous allez lui expliquer comment créer la base, et comment y ajouter les tables. Vous lui indiquerez la façon dont les données sont organisées et quel est leur format. Nous étudierons en détail ces déclarations au Chapitre 4.

Ecriture des programmes

Ce sont ces programmes qui vont accomplir les tâches que vous avez définies pour votre application. Ils créeront, entre autres, les affichages que verra l'utilisateur dans la fenêtre de son navigateur et ils assureront le dialogue avec lui. D'un autre côté, ils géreront les informations contenues dans la base (en lecture comme en écriture). En l'absence de tout programme, une base de données ne sert à rien.

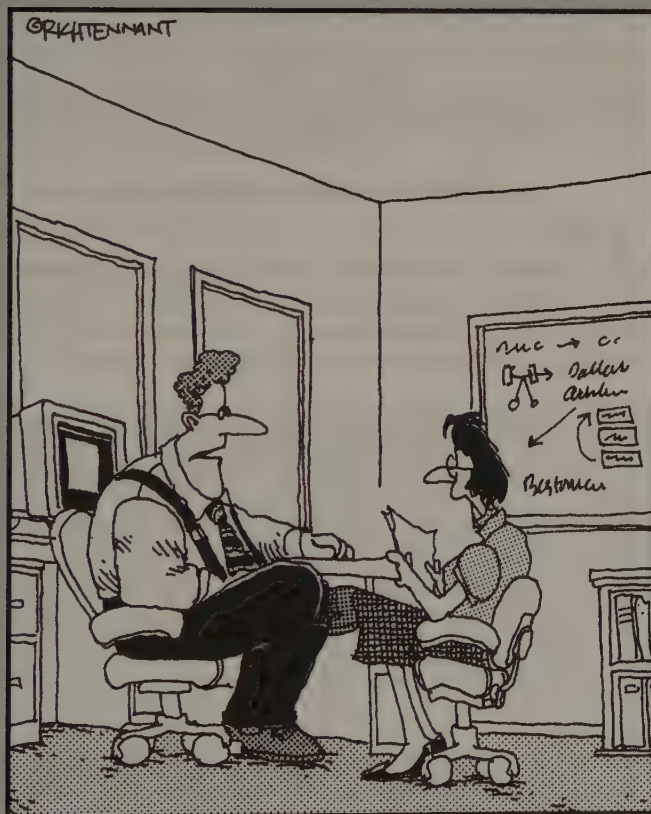
Le plan qui a été développé dans les sections précédentes a tracé les grandes lignes des programmes que vous allez devoir écrire.

En général, chaque tâche de votre projet appelle un programme distinct. Par exemple, si votre plan implique l'affichage d'un formulaire, vous devez écrire un programme à cet effet. Et le principe est le même pour lire le contenu du formulaire et l'enregistrer dans la base de données.

Le langage PHP a été développé spécifiquement pour écrire des applications interactives de bases de données sur le Web ; il contient des fonctionnalités natives à cet effet. En particulier, vous disposez de méthodes spécifiques pour gérer des formulaires et leur contenu. Nous étudierons l'écriture des programmes dans la troisième partie du livre.

Deuxième partie

Bases de données MySQL



"Notre processus de réponse automatisée à un gros crash informatique consiste à en informer la direction, à sauvegarder les données existantes et à vendre au plus vite 90 % de mes actions de l'entreprise."

Dans cette partie...

Vous allez trouver dans cette partie des détails sur la façon d'exploiter une base de données MySQL grâce au langage SQL. Vous découvrirez en outre comment créer une base de données et en manipuler les données.

Chapitre 4

Construction de la base de données

Dans ce chapitre :

Envoi de requêtes SQL à MySQL.

Création d'une nouvelle base de données.

Ajout d'informations à une base de données existante.

Recherche d'informations dans une base de données existante.

Suppression d'informations dans une base de données existante.

Lorsque vous avez achevé le plan de votre base de données ainsi que nous venons de le voir au Chapitre 3, vous êtes prêt à créer, puis à utiliser, la base proprement dite.

Cette phase de création consiste à donner un nom à la base et à définir les tables qui vont contenir les données. Pour cela, vous devez communiquer un certain nombre d'informations à MySQL sur cette base. Plus tard, vous communiquerez avec lui pour en exploiter le contenu. Dans ce chapitre, nous allons voir comment créer des requêtes SQL, le langage que comprend MySQL.

Communications avec MySQL

Le serveur MySQL est le gestionnaire de vos bases de données. C'est lui qui :

- ✓ Crée les nouvelles bases de données.
- ✓ Sait à quel endroit sont conservées les bases de données.

- ✓ Range les données et les retrouve selon les *requêtes* qu'il reçoit.

Pour cette communication, vous devez rédiger une requête SQL puis l'envoyer au serveur MySQL (qui a été décrit dans le Chapitre 1). Les deux sections qui suivent vous donnent des détails sur la façon de procéder.

Construction de requêtes SQL

Le premier mot d'une requête est son nom. C'est un vocable d'action, mot ou verbe, qui indique à MySQL ce qu'il doit faire. Les requêtes dont nous parlerons dans ce chapitre sont : CREATE, DROP, ALTER, SHOW, INSERT, LOAD, SELECT, UPDATE et DELETE. (En français : créer, lâcher, modifier, montrer, insérer, charger, sélectionner, actualiser et supprimer.) Ce vocabulaire de base est suffisant pour créer et manipuler des bases de données sur un site Web.

Le nom de la requête doit être suivi par des mots ou des phrases (certains obligatoires, d'autres facultatifs) qui indiquent à MySQL comment accomplir l'action spécifiée. Par exemple, vous devez toujours indiquer à MySQL ce qu'il doit créer et dans quelle table insérer ou extraire des données.

La requête suivante vous donne un exemple de ce langage :

```
SELECT nom FROM Membre
```

Cette requête demande à MySQL de rechercher tous les champs appelés "nom" dans la table appelée Membre. Il existe des formes de requête plus complexes, comme la suivante :

```
SELECT nom, prénom FROM Membre WHERE département="95"  
AND ville="Montmorency" ORDER BY nom
```

Cette requête demande à MySQL de retrouver tous les noms et prénoms des membres qui habitent à Montmorency, dans le Val-d'Oise (code 95) et de les présenter triés en séquence croissante sur le nom.



Voici quelques points importants dont vous devez vous souvenir lorsque vous construisez une requête en SQL du genre de celles que nous venons de voir.

- ✓ **Majuscules et minuscules.** Dans ce livre, j'écrirai les mots du langage SQL en capitales (majuscules) et ceux des champs (variables) de la base de données en bas de casse (minuscules), ce qui facilitera la lecture et la compréhension des requêtes. Mais, comme pour le langage HTML, ce n'est pas un impératif exigé par MySQL : `select` et `SELECT` seront compris de la même façon. En revanche, les noms des tables, des colonnes comme de toute information variable doivent respecter la casse utilisée pour les déclarer si vous œuvrez sous Linux ou UNIX. Dans ce cas, `motPasse` et `motpasse` ne sont pas identiques. Si vous travaillez sous Windows, cela n'a pas d'importance, car ce système d'exploitation ne fait pas de distinction entre majuscules et minuscules. Avec Windows, `motPasse` et `motpasse` sont identiques.
- ✓ **Espaces.** Les mots du vocabulaire SQL doivent être séparés les uns des autres par au moins un espace. Peu importe qu'il y en ait un ou vingt. En outre, une requête peut s'étaler sur plusieurs lignes sans qu'il soit nécessaire de l'indiquer par des "caractères de continuation".
- ✓ **Guillemets.** Dans la dernière commande que j'ai montrée, vous aurez sans doute noté que `95` et `Montmorency` étaient placés entre guillemets. Ce sont en effet des chaînes de caractères (nous reviendrons un peu plus loin, dans ce même chapitre, sur cette notion) et, comme telles, elles exigent cette forme particulière d'écriture. En ce qui concerne les autres types de variables (numériques, par exemple), **il ne faut pas** utiliser de guillemets. Les principaux types de données nous ont été présentés dans le Chapitre 3.
- ✓ **Caractères accentués.** De même que PHP (comme nous le verrons dans la troisième partie), MySQL accepte les caractères accentués dans les noms de champs, de tables et de bases de données.

Envoi de requêtes SQL

Comme l'indique le titre de ce livre, je ne parlerai ici que de l'envoi de requêtes SQL au moyen de PHP. La vie de MySQL ne s'arrête pas là, il s'en faut de très loin, mais ce n'est pas notre propos. L'exemple du programme `mysql_envoi.php` (voir le Listing 4.1) montre comment envoyer une requête à MySQL. Recopiez les instructions de ce

programme dans le répertoire de votre application Web, en modifiant les informations des lignes imprimées en gras pour qu'elles soient conformes à votre compte de base de données, et chargez ce programme dans votre navigateur.



Si saisir les programmes du livre vous ennuie profondément, sachez que vous pouvez les charger sur le site de l'auteur, à l'adresse <http://janet.valade.com>. Bien entendu, vous disposerez alors de sources originaux, donc non traduits. Les noms des bases, des tables ou encore des champs ne correspondront donc pas à ce qui est proposé dans ce livre. Mais faire vous-même ce travail d'adaptation peut être un excellent exercice !

Listing 4.1 Exemple de script PHP envoyant une requête SQL et en récupérant le résultat pour l'afficher.

```
<?php
/*Programme :   mysql_envoi.php
 *Description : Envoi d'une requête SQL au serveur SQL
 *              et affichage des résultats.
 */
echo "<html>
      <head><title>Envoi de requêtes SQL (Edition 2007)</title></head>
      <body>";
if(ini_get("magic_quotes_gpc") == "1")
{
    $_POST['query'] = stripslashes($_POST['query']);
}
$host="localhost";
$user="";
$password="";

/* Section d'exécution de la requête et affichage des résultats */
if(!empty($_POST['form']))
{
    $cxn = mysqli_connect($host,$user,$password,
                        $_POST['database']);
    $result = mysqli_query($cxn,$_POST['query']);
    echo "Base de données sélectionnée : <b>{$_POST['database']}</b><br>
          Requête : <b>{$_POST['query']}</b>
          <h3>Résultats</h3><hr>";
    if($result == false)
    {
        echo "<h4>Erreur SQL : ".mysqli_error($cxn)."</h4>";
    }
    elseif(@mysqli_num_rows($result) == 0)
```

```

[
    echo "<h4>Requête exécutée.
        Aucun résultat envoyé.</h4>";
]
else
[
/* Affiche les résultats */
echo "<table border='1'><thead><tr>";
$finfo = mysqli_fetch_fields($result);
foreach($finfo as $field)
{
    echo "<th>".$field->name."</th>";
}
echo "</tr></thead>
    <tbody>";
for ($i=0;$i < mysqli_num_rows($result);$i++)
[
    echo "<tr>";
    $row = mysqli_fetch_row($result);
    foreach($row as $value)
    [
        echo "<td>".$value."</td>";
    ]
    echo "</tr>";
]
echo "</tbody></table>";
]
/* Affiche le formulaire avec des boutons après les résultats */
$query = str_replace("'", "%'", $_POST['query']);
echo "<hr><br>
    <form action='{$_SERVER['PHP_SELF']}' method='POST'>
    <input type='hidden' name='query' value='$query'>
    <input type='hidden' name='database'
        value='{$_POST['database']}'>
    <input type='submit' name='queryButton'
        value='Nouvelle requête'>
    <input type='submit' name='queryButton'
        value='Modifier la requête'>
    </form>";
exit();
]

/* Affiche le formulaire de saisie de requête */
if (@$_POST['queryButton'] != "Modifier la requête")
{
    $query = " ";
}
else

```

```

{
    $query = str_replace("%&%", "", $_POST['query']);
}
?>
<form action="<?php echo $_SERVER['PHP_SELF'] ?>"
    method="POST">
<table>
<tr><td style='text-align: right; font-weight: bold'>
    Saisir le nom de la base de données</td>
<td><input type="text" name="database"
    value=<?php echo @$_POST['database'] ?> ></td>
</tr>
<tr><td style='text-align: right; font-weight: bold'
    valign="top">Saisir la requête SQL</td>
<td><textarea name="query" cols="60"
    rows="10"><?php echo $query ?></textarea></td>
</tr>
<tr><td colspan="2" style='text-align: center'>
<input type="submit" value="Valider la requête"></td>
</tr>
</table>
<input type="hidden" name="form" value="yes">
</form>
</body></html>

```



Le programme ci-dessus ne fonctionnera pas si vous utilisez des fonctions de la famille `mysql` en remplacement des nouvelles fonctions `mysqli` qui y sont utilisées (nous avons présenté ce sujet dans le Chapitre 2). La syntaxe n'est pas exactement la même. La nouvelle fonction de connexion `mysqli_connect()` attend un quatrième paramètre pour le nom de la base (voyez aussi le début du Chapitre 8). Il y a même une nouvelle fonction sans équivalent antérieur : `mysqli_fetch_fields`. Voici comment reformuler les lignes suivantes pour les anciennes fonctions `mysql` :

```

$finfo = mysqli_fetch_fields($result);
foreach($finfo as $field)
{
    echo "<th>".$field->name."</th>";
}

```

Voici la formulation à adopter :

```

for($i = 0; $i < mysql_num_fields($result); $i++)
{
    echo "<th>".mysql_field_name($result, $i).

```



```
"</th>";  
}
```

Dans les deux approches, vous aurez peut-être à modifier les lignes imprimées en gras dans le code source. Elles indiquent à MySQL les informations qui lui permettent de retrouver la base de données et d'y accéder. Ce sont :

```
$host="localhost";  
$user="";  
$password="";
```

Modifiez ces lignes pour qu'elles reflètent respectivement le nom de l'ordinateur, votre login et votre mot de passe de base de données. Si vous utilisez les bons offices d'un hébergeur, c'est lui qui vous aura indiqué ces valeurs. Si vous avez installé votre configuration vous-même, vous avez déjà défini ces valeurs. En particulier, le nom d'hôte `localhost` devrait convenir si MySQL est installé sur le même ordinateur que votre site. Elles peuvent même être (très éventuellement) constituées par des chaînes de caractères vides :

```
$host="";  
$user="";  
$password="";
```



Sous Linux/UNIX, n'utilisez pas `root` sans mot de passe comme valeur pour `user`, car vous iriez au-devant de problèmes de sécurité en permettant à tout le monde d'accéder à votre base de données. Il vaut mieux supprimer ce compte s'il existe. Reportez-vous au Chapitre 5 pour plus de détails sur les comptes et les mots de passe sous MySQL.

Une fois que vous aurez effectué les corrections indiquées pour ces trois lignes, exécutez les étapes suivantes :

1. **Chargez le programme `mysql_envoi.php` dans votre navigateur.**

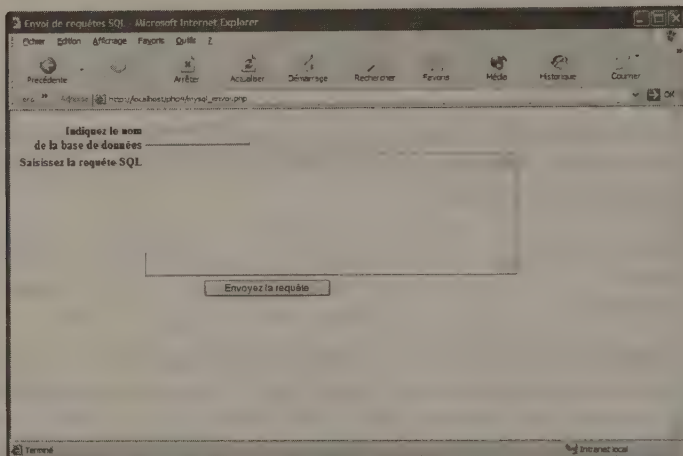
La Figure 4.1 vous montre ce que vous devriez voir.

2. **Indiquez un nom de base de données dans la boîte de saisie plus petite, si la requête SQL le nécessite.**

Les détails d'écriture d'une requête SQL vous seront expliqués dans les sections suivantes de ce même chapitre.

3. **Tapez la requête SQL dans la grande zone de texte.**

Figure 4.1 :
Ce qu'affiche
l'exécution
du pro-
gramme
mysql_envoi.php.



4. Cliquez sur le bouton Valider la requête.

La requête est exécutée et une nouvelle page affichée qui en montre les résultats. Si votre requête contenait des erreurs, ces résultats sont remplacés par un message d'erreur.

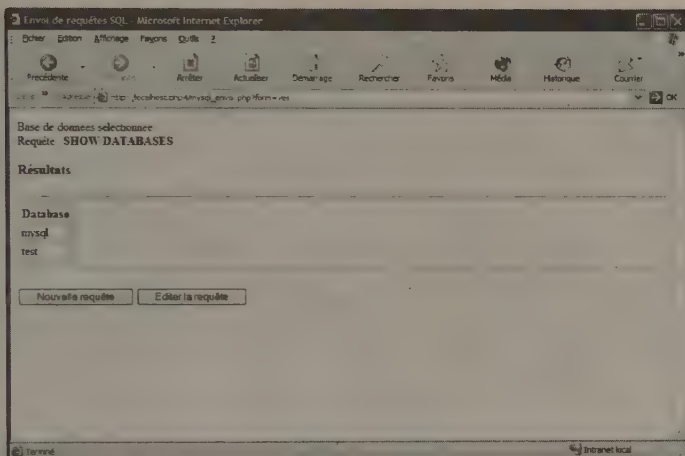
Vous pouvez tester facilement ce programme en tapant :

```
SHOW DATABASES
```

Normalement, vous n'avez pas besoin d'indiquer un nom de base de données, aussi pouvez-vous ignorer l'étape 3. Lorsque vous cliquez sur le bouton Valider la requête, la liste des bases de données présentes sur le serveur est affichée, comme on peut le voir sur la Figure 4.2.

Presque toujours parmi cette liste figureront les bases **test** et **MySQL**, installées automatiquement lors de l'installation de MySQL. (La base de données **MySQL** sert aux besoins propres de MySQL.) Même si aucune base n'est détectée, la requête doit s'exécuter correctement. Un message d'erreur s'affichera en cas de problème. De ce point de vue, les messages MySQL sont généralement bien conçus et ils facilitent la localisation de l'erreur.

Figure 4.2 :
Résultat
d'une
interrogation
affichant la
liste des
bases de
données déjà
présentes
sous MySQL



Une façon plus rapide d'envoyer des requêtes au serveur MySQL

Lorsque vous installez MySQL, vous installez automatiquement un autre programme appelé `mysql`. (Avec certains systèmes d'exploitation, il peut s'appeler *terminal monitor* ou *monitor*.) Les programmes qui communiquent avec le serveur MySQL sont appelés *logiciels clients*, parce que ce qui s'adresse à un serveur est tout naturellement un client. Lorsque vous envoyez une requête au serveur MySQL au moyen d'un "client `mysql`", le serveur renvoie une réponse à ce client, réponse qui est affichée sur l'écran. Ce programme de monitoring peut envoyer des requêtes sur un réseau, car il n'est pas nécessaire qu'il tourne sur la même machine que celle où est installée la base de données.

Pour envoyer des requêtes à MySQL au moyen du "client `mysql`", suivez les étapes énumérées ci-dessous :

1. Localisez le client `mysql`.

Par défaut, il se trouve dans le sous-répertoire `bin` du répertoire dans lequel MySQL a été installé. Avec Linux/UNIX, il figure dans le sous-répertoire `/usr/local/mysql/bin` ou encore sous `/usr/local/bin`. Sous Windows, il s'agit par défaut du sous-répertoire `c:\Program Files\mysql\bin`, et avec EasyPHP de `C:\Program Files\EasyPHP\mysql\bin`. Mais selon les options choisies par l'administrateur de la machine que vous utilisez, il peut se trouver à un autre endroit. Si vous n'êtes pas vous-même l'administrateur de MySQL, vous pouvez fort bien ne pas pouvoir accéder à ce programme.

Dans ce cas, demandez à ce personnage tout-puissant de vous faire une copie du client `mysql` à un endroit d'où vous puissiez l'utiliser.

2. Lancez le client `mysql`.

- Sous Linux/UNIX, tapez le nom du client précédé de son chemin d'accès (par exemple : `/usr/local/mysql/bin/mysql`).
- Sous Windows, ouvrez une fenêtre MS-DOS. Avec Windows 95/98, cliquez sur Démarrer/Exécuter puis tapez `dosprmt` dans la boîte de saisie qui s'affiche et, enfin, cliquez sur OK. Avec Windows 2000/XP, il vous suffit d'ouvrir la fenêtre Invite de commandes à partir du groupe des Accessoires. Dans cette fenêtre, allez dans le sous-répertoire des programmes de MySQL et tapez `mysql`.
- Si vous êtes sur un réseau local, ajoutez les paramètres suivants après la commande :
 - h *hôte* où *hôte* est le nom de la machine sur laquelle MySQL est installé.
 - u *utilisateur* où *utilisateur* est le nom de votre compte MySQL.
 - p *mpasse* où *mpasse* est le mot de passe attaché à votre compte MySQL.
- Avec EasyPHP, tapez `C:\Program Files\EasyPHP\mysql\bin\mysql.exe`.

La commande finale pourrait ressembler à ceci :

```
mysql -h mysqlhost.macompanie.com -u root -p
```

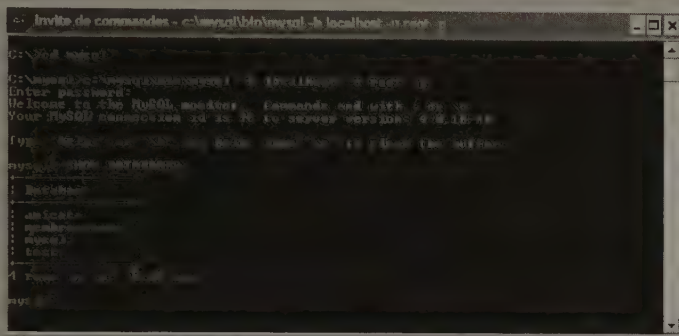
Quel que soit le cas, terminez par un appui sur la touche <Entrée>.

3. Si un mot de passe vous est demandé, tapez-le.
4. Lorsque le client `mysql` démarre, il affiche les trois premières lignes que vous pouvez voir sur la Figure 4.3, suivies de l'invite `mysql>`.
5. Sélectionnez la base de données que vous voulez utiliser en saisissant la commande (remplacez `basededonnées` par le nom voulu, puis validez) :

```
use basededonnées
```

6. Vous pouvez alors taper votre commande SQL que vous terminerez par un point-virgule (;). La réponse est affichée à l'écran. Continuez si nécessaire à saisir vos requêtes de la même façon.
7. Pour quitter le client `mysql`, tapez simplement `quit` suivi d'un appui sur la touche <Entrée>.

Figure 4.3 :
Invite de commandes
dans laquelle
est utilisé le
client mysql.



Construction d'une base de données

Une base de données se compose de deux parties : une structure, qui contiendra les données, et les données proprement dites. Dans les sections qui suivent, nous verrons comment créer la structure de la base de données. Pour commencer, vous allez créer une base de données vide, c'est-à-dire dépourvue de tout contenu.

Pour cela, vous allez utiliser la commande SQL `CREATE` suivie du nom de la base à créer. Nous verrons également d'autres commandes : `ALTER`, `DROP` et `SHOW`. Pour l'instant, vous devrez utiliser le client `mysql`, dont nous venons de voir le fonctionnement, puisque nous ne savons pas encore le faire avec un script PHP. En dehors de EasyPHP, vous devrez également avoir les privilèges nécessaires pour conduire à bien cette opération. Des précisions sur ce dernier point vous seront données au Chapitre 5.

Création d'une nouvelle base

La commande à utiliser pour créer une nouvelle base de données est la suivante :

```
CREATE DATABASE nomBase
```


où *nomBase* est le nom que vous allez donner à cette base de données. En voici deux exemples en rapport avec les deux exemples que nous avons commencé à étudier dans le chapitre précédent :

```
CREATE DATABASE AniCata  
CREATE DATABASE MembresSeuls
```



Certains fournisseurs d'accès, parmi ceux qui vous proposent PHP et MySQL, ne vous autorisent à créer qu'une seule base de données. C'est le cas de Free, en particulier. En fait, ni PHP ni MySQL ne vous empêchent de placer toutes vos tables dans une seule base de données au lieu de créer autant de bases que vous avez de projets différents. Mais il est évident que ce n'est pas une méthode bien efficace.

Ensuite, vous pourrez passer à la création des tables qui contiendront les données. Pour vérifier que vos bases ont bien été créées, lancez la commande SQL que nous avons déjà rencontrée :

```
SHOW DATABASES
```

Suppression d'une base de données

Pour supprimer une base de données, tapez la commande :

```
DROP DATABASE nomBase
```



Utilisez la commande `DROP DATABASE` avec prudence, car son effet est irréversible. Les données que contenait la base qui vient d'être détruite sont irrécupérables.

Ajout de tables dans une base de données

Vous pouvez ajouter des tables à n'importe quelle base de données (pour peu que vous ayez les privilèges nécessaires pour y accéder), qu'elle soit vide ou qu'elle contienne déjà quelque chose. Pour cela, vous devez utiliser une autre forme de la requête `CREATE`.

Repartons des deux exemples du Chapitre 3. La base de données AniCata possède trois tables : Animal, Type et Couleur. La seconde base, MembresSeuls, possède deux tables : Membre et Login. Comme une table appartient toujours à une base de données, il faut commencer par sélectionner cette base. Faute de quoi, vous verriez

s'afficher le message `No Database Selected` (aucune base de données n'est sélectionnée). Si vous utilisez le formulaire de la Figure 4.1, vous devrez remplir le champ de saisie qui se trouve en haut de la fenêtre.

La requête qui permet d'ajouter une table à une base de données commence par :

```
CREATE TABLE nomtable
```

Vient ensuite une liste de noms de colonnes accompagnés de leurs définitions. Les informations relatives à une colonne sont séparées de celles de la suivante par une virgule, et l'ensemble de cette liste est placé entre parenthèses. Les définitions sont constituées par le type de données (voir à ce sujet le Chapitre 3), la longueur et/ou d'autres mots clés dont la liste qui suit donne un échantillon :

- ✓ **NOT NULL** Cette colonne doit toujours avoir une valeur. Elle ne peut pas rester vide.
- ✓ **DEFAULT valeur** Valeur à placer par défaut dans la colonne si aucune valeur n'est spécifiée.
- ✓ **AUTO_INCREMENT** Création d'une suite de valeurs numériques entières consécutives : 1, 2, 3... L'accroissement est d'une unité par ligne ajoutée. Cet attribut peut être annulé si on spécifie une valeur particulière pour la colonne. C'est un moyen pratique pour créer des clés primaires.
- ✓ **UNSIGNED** La valeur qui sera placée dans cette colonne devra toujours être positive ou nulle.

Le dernier article d'une requête de création de table indique quelle colonne ou combinaison de colonnes sera l'identificateur unique pour la ligne : c'est sa *clé primaire*. Chaque ligne d'une table doit avoir un champ ou une combinaison de champs qui sera (seront) unique(s) dans la table. Si vous tentez d'ajouter une nouvelle ligne ayant la même clé primaire qu'une ligne qui existe déjà, vous obtiendrez un message d'erreur et la ligne ne sera pas ajoutée. La clause qui spécifie la clé primaire s'écrit :

```
PRIMARY KEY (nom_de_colonne)
```

Le nom de la colonne (*nom_de_colonne*) doit être écrit entre parenthèses. S'il s'agit d'une combinaison de colonnes, il faut séparer les noms des colonnes par autant de virgules que nécessaire.

Par exemple, pour définir la clé primaire de la table `Login` de la base `MembresSeuls`, il faut écrire :

```
PRIMARY KEY (login,date)
```

Le Listing 4.2 présente la requête utilisée pour créer la table `Membre` de la base de données `MembresSeuls`. Cette requête peut s'écrire sur une seule ligne, mais il est recommandé de l'écrire de la façon indiquée pour en faciliter la relecture (et éventuellement la correction). (Notez qu'il faudrait dans la pratique émettre une requête `USE MembresSeuls` avant de créer cette table.)

Listing 4.2 : Requête de création de la table `Membre` pour la base de données `MembresSeuls`.

```
CREATE TABLE Membre
( login      VARCHAR(20) NOT NULL,
  initial    DATE        NOT NULL,
  mPasse     CHAR(255)   NOT NULL,
  nom        VARCHAR(50),
  prénom     VARCHAR(40),
  rue        VARCHAR(50),
  ville      VARCHAR(50),
  département CHAR(2),
  codePostal CHAR(10),
  email      VARCHAR(50),
  tph        CHAR(15),
  fax        CHAR(15),
  PRIMARY KEY(login)
);
```

Comme vous pouvez le voir, l'ensemble des paramètres de la requête est placé entre parenthèses, chaque entrée étant séparée de la suivante par une virgule. L'instruction se termine par un point-virgule.



Rappelez-vous qu'il est interdit d'utiliser un des mots réservés de MySQL pour appeler une colonne. Faute de quoi, vous récolteriez un message d'erreur du genre de celui-ci :

```
You have an error in your SQL syntax near
'create VARCHAR(50), prénom VARCHAR(40)' at line 5
```

Comme vous le voyez, ce message vous indique la définition d'une colonne qui n'est pas bonne et la ligne où elle a été trouvée. L'erreur provient ici de l'utilisation du mot réservé "create" en guise de nom de colonne.

- ✓ Pour voir les tables que vous venez d'ajouter à une base de données, tapez la requête :

```
SHOW TABLES
```

- ✓ Pour voir la structure d'une table, utilisez la requête :

```
DESCRIBE nomTable
```

- ✓ Pour supprimer une table, utilisez la requête :

```
DROP TABLE nomTable
```



Utilisez la commande `DROP TABLE` avec prudence, car son effet est irréversible. Les données que contenait la table qui vient d'être détruite sont irrécupérables.

Modification de la structure d'une base de données

La base de données que vous venez de créer n'est pas figée pour l'éternité. Vous pouvez en modifier la structure au moyen de la requête `ALTER`, ajouter, supprimer ou renommer une colonne ou modifier le type de données qu'elle contient ou tout autre de ses attributs.

Le format de base de cette commande est :

```
ALTER TABLE nomTable
```

A sa suite, vous indiquez les changements que vous voulez opérer. Le Tableau 4.1 donne la liste des modifications possibles.

Modifier une base de données n'est pas quelque chose de rare. Vous pouvez avoir besoin de le faire pour de nombreuses raisons. Par exemple, supposez que vous ayez défini le type de données de la colonne `nom` de la table `Membre` (base de données `MembresSeuls`) comme `VARCHAR(20)`, et qu'ultérieurement survienne un nouveau membre du nom de Marc Schwartzheimer-Losterman. Pour ne pas amputer ce magnifique patronyme en le réduisant à ses vingt

Tableau 4.1 : Modifications possibles au moyen de la requête ALTER.

Changement	Description
ADD nomCol définition	Ajouter une colonne. <i>définition</i> renferme le type de données et d'autres paramètres facultatifs.
ALTER nomCol SET DEFAULT valeur	Modifier la valeur par défaut d'une colonne.
ALTER nomCol DROP DEFAULT valeur	Supprimer la valeur par défaut d'une colonne.
CHANGE nomCol nouvNomCol définition	Modifier la définition d'une colonne et changer son nom. <i>définition</i> renferme le type de données et d'autres paramètres facultatifs.
DROP nomCol	Supprimer une colonne ainsi que les données contenues dans cette colonne, lesquelles seront définitivement perdues.
MODIFY nomCol définition	Modifier la définition d'une colonne. <i>définition</i> renferme le type de données et d'autres paramètres facultatifs.
RENAME nouvNomTable	Changer le nom d'une table.

premières lettres, vous allez redéfinir par prudence cette colonne ainsi :

```
ALTER TABLE Membre MODIFY nom VARCHAR(50)
```

Manipulation des données d'une base

Une base de données vide est pareille à une bonbonnière vide : elle ne présente aucun intérêt. Elle ne peut être utile à quelque chose qu'à partir du moment où elle contient des informations. Le rôle d'une base de données est de recevoir des informations et de pouvoir les compléter, les restituer ou les modifier à la demande.

Voici les quatre types de requêtes qui peuvent être adressées à une base de données :

- ✓ **Recevoir de nouvelles informations.** Ajouter une ligne à une table.

- ✓ **Mettre à jour des informations.** Modifier les données contenues dans une ligne qui existe déjà. Par exemple, ajouter une donnée dans un champ jusque-là vide.
- ✓ **Retrouver des informations.** Rechercher une valeur de donnée. Cette requête ne modifie pas le champ où se trouve la valeur.
- ✓ **Supprimer des informations.** Supprimer une valeur dans une base de données.

Une requête peut parfois concerner plusieurs tables. Par exemple, la question : "Combien coûte un dragon vert ?" demande des informations provenant de la table *Animal* et de la table *Couleur*. Vous pouvez poser cette question au moyen d'une seule requête `SELECT` associant les deux tables.

Dans les sections qui suivent, nous allons voir comment accepter et retrouver des informations et comment associer des tables.

Ajouter des informations

Toute base de données doit contenir des données. L'exemple de la base *AniCata* montre comment le contenu de cette table permet aux visiteurs de se documenter sur les animaux commercialisés. À l'inverse, ces données peuvent provenir des utilisateurs eux-mêmes, comme c'est le cas pour la base *MembresSeuls*. Dans les deux cas, des données sont ajoutées à une base.

Si vos données sont encore sur papier, vous pouvez les saisir dans une base MySQL, ligne par ligne, au moyen d'une requête MySQL. Cependant, si vous avez beaucoup de données, cette méthode se révèle vite fastidieuse. Supposez, par exemple, que vous ayez des données sur un millier de produits à ajouter à votre base. En admettant que vous soyez un virtuose du clavier et que vous soyez capable de taper une ligne par minute, cela représente une quinzaine d'heures de dactylographie forcenée. Pas très drôle, n'est-ce pas ? Mais néanmoins faisable. D'un autre côté, supposez que vous deviez entrer les informations concernant les 5 000 membres d'une organisation et qu'il faille cinq minutes pour saisir tout ce qui concerne un des membres. Maintenant, c'est environ 400 heures de travail continu que cela implique. Qui va s'en charger ?

Lorsque vous avez beaucoup d'informations à saisir, il faut trouver une solution appropriée. Scanner ces informations est une des méthodes possibles, à condition que vous ayez ensuite un bon logiciel d'OCR (*Optical Character Recognition* - reconnaissance optique de caractères). Vous pourrez aussi sous-traiter cette saisie sous forme de texte au kilomètre plutôt que de considérer la saisie individuelle sous forme de requête SQL.

Il existe une requête SQL (LOAD) qui permet de lire des données contenues dans un fichier texte. Si vos données sont déjà sous une forme lisible par l'ordinateur, ce fichier est utilisable. Et si ce n'est pas du texte pur mais un fichier Excel, Access ou Oracle, par exemple, la conversion est généralement réalisable, et vous pouvez ensuite lire le fichier dans votre base de données MySQL. Dans le cas où ces données ne sont pas encore enregistrées sur l'ordinateur, il sera plus rapide de les saisir dans un fichier de texte avant de les transférer dans MySQL.

Presque tous les fichiers texte peuvent être lus par MySQL, mais certaines formes sont plus faciles à lire que d'autres. Si vous avez l'intention de saisir des données sous cette forme, lisez la section "Ajout d'une grande quantité de données" pour voir quel est le meilleur format pour votre fichier texte. Mais, naturellement, si ces données sont déjà dans un fichier informatique, vous les récupérez généralement telles qu'elles sont. Encore qu'il existe des logiciels capables de modifier de façon simple la structure d'un fichier texte pour l'adapter à tel ou tel usage.

Ajout d'une ligne à la fois

C'est la requête INSERT qui va s'en charger. Elle indique à MySQL dans quelle table doit être ajoutée la ligne et quelles sont les valeurs à ajouter dans la ligne. Sa forme générale est la suivante :

```
INSERT INTO nomTable (nomCol1, nomCol2... nomColx)
VALUES (val1, val2..., valx)
```

Voici quelles sont les règles qui s'appliquent à la requête INSERT :

- ✓ **Les valeurs doivent apparaître dans le même ordre que les noms des colonnes correspondantes.** La première valeur sera placée dans la première des colonnes énumérées, et ainsi de suite.

- ✓ **Une liste partielle des noms de colonnes dans la ligne est possible.** Il n'est pas indispensable que toutes les colonnes de la ligne reçoivent un contenu. Elles prendront alors la valeur définie par défaut, ou sinon resteront vides.
- ✓ **Une liste des noms de colonnes n'est pas obligatoire.** Lorsque toutes les colonnes d'une ligne doivent recevoir des valeurs, il n'est pas nécessaire que vous indiquiez les noms de toutes ces colonnes dans la requête. L'absence de noms de colonnes est interprétée par MySQL comme l'énumération implicite de l'intégralité des colonnes. Il faudra alors autant de valeurs qu'il existe de colonnes dans la table.
- ✓ **La liste de valeurs et la liste de noms de colonnes doivent être de même longueur.** Faute de quoi, vous recevrez un message de ce genre :

```
Column count doesn't match value count
```

La requête INSERT ci-après ajoute une ligne dans la table Membre :

```
INSERT INTO Membre (login,initial,mPasse,nom,
                    rue,ville,département,codePostal,
                    email,tph,fax)
VALUES ("PetitMalin", "2002-Jan-30","a234wyz","Dupondt",
        "23 rue de la Poupée-qui-Tousse", "Richeville",
        "45", "45100", "dupondt@monserveur.com", "0233445566","")
```

Notez que le prénom ne figure pas dans la liste des noms de colonnes. Aucune valeur ne sera entrée dans ce champ. S'il avait l'attribut NOT NULL, un message d'erreur serait envoyé par MySQL. Si une valeur par défaut (attribut DEFAULT) existait pour ce champ, c'est elle qui serait retenue. Notez aussi que le numéro de fax est une chaîne de caractères vide : "", ce que MySQL considère comme parfaitement valable.

Pour voir quelles valeurs ont les données qui viennent d'être entrées dans une table et être à même de les vérifier, envoyez une requête SQL d'interrogation. Nous étudierons cette forme de requête plus loin dans ce même chapitre, dans la section "Recherche d'informations". Sans plus attendre, sachez que vous pouvez récupérer toutes les valeurs d'une ligne au moyen de la requête :

```
SELECT * FROM nomTable;
```

Ajout d'une grande quantité de données

Lorsque vous avez de nombreuses données à saisir dans une table, et qu'elles existent déjà sous forme lisible par un ordinateur, vous pouvez en effectuer le transfert au moyen d'une requête `LOAD` dans laquelle vous spécifierez le nom de la base de données concernée par cette entrée de données.



Par défaut, MySQL considère que le fichier contenant vos données se trouve dans le même répertoire que la base de données elle-même.

Du fait de la façon dont sont organisées les tables de MySQL, le fichier texte source doit indiquer où commence et où finit chaque suite de valeurs à entrer dans la base. Pour cela, on utilise un caractère séparateur qui ne doit pas se trouver dans les données elles-mêmes. Par défaut, c'est une tabulation, mais vous pouvez choisir un autre caractère à condition de l'indiquer à MySQL. Toujours par défaut, la fin d'une ligne du texte est considérée comme la fin de la ligne correspondante de la base de données. Ici encore, vous pouvez spécifier un autre caractère. Un fichier texte destiné à décrire des animaux pour la base de données AniCata pourrait se présenter ainsi :

Licorne	cheval	corne spiralée	5000	/images/licorne.jpg
Pégase	cheval	animal ailé	8000	/images/pegas.jpg
Lion	chat	grande taille	2000	/images/lion.jpg

Un tel fichier est appelé *fichier délimité par des tabulations* (*tab delimited file*). Il en existe une variante : le fichier délimité par des virgules dans lequel les tabulations sont remplacées par des virgules. Si votre fichier est dans un autre format, vous devez commencer par le convertir dans un format "délimité".



Pour convertir un fichier d'un type dans un autre, consultez le manuel du logiciel qui a créé ce fichier ou posez la question à votre gourou informatique habituel. Excel, Access, Oracle, pour ne citer qu'eux, ont des options qui permettent d'effectuer cette conversion. Pour un simple fichier texte, une opération du type *recherche et remplacement* devrait permettre d'en venir à bout. Si cela ne suffit pas, appelez à l'aide un expert ou un développeur.

La forme générale de la commande `LOAD` est la suivante :

```
LOAD DATA INFILE "fichierTexte" INTO TABLE nomTable
```


La requête injecte des données lues dans un fichier texte situé sur le serveur. Si le nom du fichier n'est pas précédé d'un chemin d'accès, MySQL le cherche dans le répertoire (dossier) contenant le fichier de définition des tables (fichier *tablename.frm*). Par défaut, le nom de ce dossier est déduit du nom de la base, par exemple *anicata*, et c'est un sous-dossier du dossier *data* de votre implantation de MySQL. Par exemple, si le fichier se nomme *data.dat*, la commande **LOAD** doit désigner le fichier dans ce chemin : `C:\Program Files\MySQL\data\anicata\data.dat`.

Des paramètres facultatifs permettent de spécifier les séparateurs. Ce sont :

```
FIELDS TERMINATED BY "caractère"
FIELDS ENCLOSED BY "caractère"
LINES TERMINATED BY "caractère"
```

ce qu'on peut traduire respectivement par :

```
champs terminés par...
champs encadrés par ...
lignes terminées par ...
```

Supposons que vous disposiez d'un fichier texte, *animaux.txt*, convenant au catalogue d'animaux que nous étudions à titre d'exemple, fichier dont chaque champ est séparé du suivant par une virgule et non par une tabulation. Vous devrez adresser la requête suivante à MySQL :

```
LOAD DATA INFILE "animaux.txt" INTO TABLE Animal
FIELDS TERMINATED BY ",";
```

Pour pouvoir utiliser la requête **LOAD DATA INFILE**, le compte MySQL doit disposer du privilège **FILE** sur le serveur. Ce sujet sera abordé dans le Chapitre 5.

Vous pouvez également charger des données depuis un fichier au format texte local à votre machine au moyen du mot réservé **LOCAL** :

```
LOAD DATA LOCAL INFILE "chemin/nomfichier"
INTO TABLE nomtable
```

Il faut fournir le chemin d'accès complet au fichier en utilisant des barres obliques normales, même sous Windows, dans le style `"C:/base/monfichier1.txt"`. Si vous obtenez une erreur avec cette



requête, c'est sans doute que LOCAL n'est pas activée. (Nous montrons comment l'activer dans le chapitre suivant.)

Pour vérifier les données injectées dans la table, vous utilisez une requête SQL de lecture de données. (Nous voyons ces requêtes dans la section suivante.) N'oubliez pas de contrôler le résultat en examinant le contenu de la table avec la requête suivante :

```
SELECT * FROM Animal;
```

Recherche d'informations

Le seul intérêt de stocker des informations dans une base de données est de pouvoir les récupérer plus tard. Une base de données n'existe que pour répondre à des questions. Quels sont les animaux en vente ? Qui sont les membres ? Combien y a-t-il de membres dans le Lot-et-Garonne ? Avez-vous un alligator à vendre ? Quel est le prix d'un dragon ? etc. C'est la requête SELECT qui permet de rechercher la réponse à toutes ces questions.

Sa forme générale est la suivante :

```
SELECT * FROM nomTable
```

Cette requête récupère toutes les informations contenues dans la table. L'astérisque est un caractère joker qui signifie *toutes les colonnes*.

On peut accroître la sélectivité de la requête SELECT en y ajoutant des mots du vocabulaire SQL permettant de situer exactement l'emplacement de l'information recherchée ou les critères à laquelle elle doit répondre. Vous pouvez spécifier quelles informations vous souhaitez, comment elles doivent être présentées et quelle en est leur source :

- ✓ **Vous pouvez limiter la recherche aux colonnes qui répondent exactement à votre question.** Par exemple, vous pouvez demander les seuls noms et prénoms d'une liste de membres.
- ✓ **Vous pouvez spécifier dans quel ordre les informations doivent vous être présentées.** Par exemple, vous pouvez demander qu'elles soient triées en ordre alphabétique inverse.

- ✓ **Vous pouvez spécifier les objets (les lignes) dans lesquels doivent être recherchées les informations.** Par exemple, vous pouvez demander le nom et le prénom des seuls membres qui résident dans les Bouches-du-Rhône.



La version 4.1 de MySQL a ajouté la possibilité d'imbriquer une requête SELECT dans une autre pour constituer des sous-requêtes. De telles sous-requêtes sont utilisables avec SELECT, INSERT, UPDATE, DELETE et dans les clauses SET. Les données renvoyées par la sous-requête (une valeur, un enregistrement complet ou toute une table) sont injectées en entrée de la requête de niveau supérieur. Toutes les options de SELECT sont utilisables dans les sous-requêtes. Voyez la documentation de référence de MySQL (<http://dev.mysql.com/doc/refman/5.0/fr/subqueries.html>, par exemple) pour d'autres détails.

Recherche d'informations spécifiques

Pour rechercher des informations spécifiques, listez les colonnes dans lesquelles doivent se trouver ces informations :

```
SELECT col1,col2,colx... FROM nomTable
```

Cette requête vous fournira les valeurs de toutes les lignes pour les seules colonnes dont les noms sont énumérés dans la requête. Par exemple, la requête suivante retrouve tous les noms et prénoms contenus dans la table Membre :

```
SELECT nom,prénom FROM Membre
```

Il est parfaitement possible d'effectuer des opérations mathématiques sur les colonnes sélectionnées. Ainsi, la requête SELECT suivante va ajouter les valeurs provenant de deux colonnes :

```
SELECT col1+col2 FROM nomTable
```

Autre exemple :

```
SELECT animalPrix,animalPrix*1.196 FROM Animal
```

N'oubliez pas le point-virgule final dans vos requêtes lorsque vous les émettez dans le moniteur MySQL.



Le résultat obtenu est un prix hors taxes et un prix TTC. Vous pouvez aussi changer le nom d'une colonne lorsque vous la sélectionnez, comme dans :

```
SELECT animalPrix, animalPrix*1.196 AS PrixTTC FROM Animal
```

La clause `AS` demande à MySQL de donner comme nom `PrixTTC` à la seconde colonne retrouvée. La requête renvoie donc deux colonnes : `animalPrix` et `PrixTTC`.

Dans certains cas, ce ne sont pas les valeurs elles-mêmes qui vous intéressent, mais quelque chose qui est en rapport avec la ou les colonnes retrouvées. Par exemple, vous pouvez vous intéresser à la plus petite ou à la plus grande valeur située dans une certaine colonne d'une table. Le Tableau 4.2 donne la liste de quelques-unes des informations que vous pouvez connaître sur le contenu des colonnes.

Tableau 4.2 : Type d'informations pouvant être obtenues par une requête SELECT.

Forme SQL	Description de l'information
<code>AVG(nomCol)</code>	Valeur moyenne de toutes les colonnes de nom <i>nomCol</i> .
<code>COUNT(nomCol)</code>	Nombre de lignes dans lequel le champ <i>nomCol</i> n'est pas vide.
<code>MAX(nomCol)</code>	Plus grande valeur située dans la colonne <i>nomCol</i> .
<code>MIN(nomCol)</code>	Plus petite valeur située dans la colonne <i>nomCol</i> .
<code>SUM(nomCol)</code>	Somme des valeurs situées dans la colonne <i>nomCol</i> .

La requête ci-dessous permettra de savoir quel est l'animal qui coûte le plus cher :

```
SELECT MAX(animalPrix) FROM Animal
```



Les mots du vocabulaire SQL tels que `MAX()` ou `MIN()` sont appelés *fonctions*. Il en existe beaucoup d'autres (une bonne centaine). Certaines, comme celles du Tableau 4.2, donnent des informations au sujet d'une certaine colonne. D'autres modifient chaque valeur sélectionnée. Par exemple, `SQRT()` renvoie la racine carrée de chacune des valeurs d'une colonne. `DAYNAME()` renvoie (malheureusement en anglais) le nom du jour de la semaine de chaque champ d'une colonne au format date. Vous trouverez la description de ces

fonctions dans la page Web située à l'URL <http://dev.mysql.com/doc/refman/5.0/fr/functions.html>.

Présentation d'informations dans un certain ordre

La recherche d'informations peut être automatiquement suivie d'un tri. Par exemple, dans la table *Membre*, vous pouvez demander que les noms des membres extraits de la table soient présentés en ordre alphabétique croissant (ou décroissant). Dans la table *Animal*, vous pouvez demander que la liste des animaux soit triée sur le nom de l'animal, son type ou sa couleur.



Dans une requête `SELECT`, ce sont les mots clés `ORDER BY` ou `GROUP BY` qui effectuent ce tri :

✓ `ORDER BY`

L'information trouvée est triée dans un certain ordre. Par défaut, c'est l'ordre alphabétique ascendant :

```
ORDER BY nomCol
```

Pour spécifier l'ordre descendant (inverse), ajoutez le mot `DESC` avant le nom de la colonne :

```
SELECT * FROM Membre ORDER BY DESC nom
```

✓ `GROUP BY`

Les lignes dans lesquelles les valeurs de la colonne *nomCol* sont identiques sont regroupées. Exemple :

```
SELECT * FROM Animal GROUP BY type
```

Vous pouvez utiliser `ORDER BY` et `GROUP BY` dans la même requête.

Limiter une recherche à une source particulière

Très souvent, vous ne vous intéresserez pas à toutes les informations contenues dans une table, mais seulement à celles de certaines *lignes* répondant à un critère particulier. Pour cela, il existe trois mots dans le vocabulaire SQL :

✓ `WHERE` Permet de spécifier une condition d'existence. Par exemple, vous pouvez limiter une recherche aux seuls



membres qui résident dans le Calvados (département 14) en écrivant :

```
SELECT * FROM Membre WHERE département="14"
```

- ✓ **LIMIT** Vous pouvez limiter la recherche aux seules *n* premières lignes spécifiées par la clause **LIMIT**.
- ✓ **DISTINCT** Lorsque les colonnes de plusieurs lignes contiennent la même information, vous limitez ainsi la recherche à la première ligne dont la colonne contient cette valeur, ce qui permet d'éliminer les doublons. De cette façon, vous pourriez répondre à la question : "Le membre Untel a-t-il déjà consulté la section à accès réservé ?", au lieu de demander : "Combien de fois ce membre s'est-il connecté ?"

La clause **WHERE** permet d'effectuer des recherches obéissant à des critères complexes. Supposez que vous souhaitiez obtenir la liste de tous les membres dont les noms commencent par un "B", qui habitent à Rochefort et dont le numéro de téléphone ou de fax contient au moins un "8". (Il est certain qu'une telle question présente un très grand intérêt !) Vous utiliseriez pour cela une condition de sélection de la forme :

```
WHERE expression1 AND|OR expression2 AND|OR expressionx...
```

expression spécifie une comparaison entre deux valeurs dont l'une, au moins, est celle d'un champ de la base. De cette façon, seules les lignes de la table qui répondent à cette association de conditions seront sélectionnées. Vous pouvez ainsi grouper autant d'expressions de sélection que vous le souhaitez, en les reliant par les conditions **AND** (ET) ou **OR** (OU). Avec **AND**, les deux expressions doivent être vraies, alors qu'avec **OR** il suffit qu'une seule le soit.

Le Tableau 4.3 présente quelques conditions fréquemment utilisées.



Remarquez que, pour les deux dernières expressions, ce qui figure à droite du mot clé **LIKE** peut contenir des caractères jokers. "%" signifie "n'importe quelle chaîne de caractères" et "_" veut dire "n'importe quel caractère".

Vous pouvez combiner n'importe laquelle des expressions du Tableau 4.3 avec **AND** et/ou **OR**. Lorsque ces associations deviennent trop complexes, regroupez les conditions dans des paires de parenthèses pour être certain que les comparaisons et associations se feront bien dans l'ordre que vous souhaitez. Par exemple, pour

Tableau 4.3 : Exemple d'expressions utilisables dans une clause WHERE.

Expression	Exemple	Résultat
colonne = valeur	codePostal = "78000"	Sélectionne les lignes où le champ codePostal a pour valeur 78000.
colonne > valeur	codePostal > "78000"	Sélectionne les lignes où le champ codePostal a une valeur supérieure à 78000.
colonne >= valeur	codePostal >= "78000"	Sélectionne les lignes où le champ codePostal a une valeur supérieure ou égale à 78000.
colonne < valeur	codePostal < "78000"	Sélectionne les lignes où le champ codePostal a une valeur inférieure à 78000.
colonne <= valeur	codePostal <= "78000"	Sélectionne les lignes où le champ codePostal a une valeur inférieure ou égale à 78000.
colonne BETWEEN	codePostal BETWEEN	Sélectionne les lignes où le valeur1 AND "75000" AND "78000" champ codePostal a une valeur2 valeur supérieure à 74999 et inférieure à 78001.
colonne IN (valeur1,valeur2...)	codePostal IN ("78541", "87260")	Sélectionne les lignes où le champ codePostal a une valeur égale à 78541 ou à 87260.
colonne NOT IN (valeur1,valeur2...)	codePostal NOT IN ("78541", "87260")	Sélectionne les lignes où le champ codePostal a une valeur différente de 78541 et de 87260.
colonne LIKE valeur	codePostal LIKE "7%"	Sélectionne les lignes dont le champ codePostal commence par un 7. Les caractères de substitution possibles sont % (qui remplace une chaîne quelconque) et _ (qui figure un caractère quelconque).
colonne NOT LIKE valeur	codePostal NOT LIKE "7%"	Sélectionne les lignes dont le champ codePostal ne commence pas par un 7.

rechercher dans la table *Membre* les gens dont les noms commencent par un "B", qui habitent à Rochefort et dont le numéro de téléphone ou de fax contient au moins un "8", vous écririez :

```
SELECT nom,prénom FROM Membre
      WHERE nom LIKE "B%"
      AND ville="Rochefort"
      AND (tph LIKE "8%" OR fax LIKE "8%")
```

En l'absence de parenthèses, les expressions et associations s'effectuent de gauche à droite (ou encore du début vers la fin). Si vous supprimiez les parenthèses de la dernière ligne, vous obtiendriez la liste des gens dont les noms commencent par un "B", qui habitent à Rochefort et dont le numéro de téléphone **et** de tous les membres dont le numéro de fax contient au moins un "8", quelle que soit l'initiale de leur nom et leur lieu de résidence. Lorsque le **OR** final est évalué, les membres retenus sont ceux dont les caractéristiques vérifient la condition précédant cet opérateur, *ou* qui vérifient l'expression qui le suit. En d'autres termes, tous les **AND** ont déjà été traités à ce stade et forment donc un résultat global. Par contre, la dernière expression est évaluée à part.

LIMIT (comme on s'en douterait) limite le nombre de réponses fournies. Cette clause s'écrit :

```
LIMIT début,nbRéponses
```

début représente la première ligne à explorer (1, par défaut) et *nbRéponses* le nombre maximal de réponses à fournir. Pour obtenir la liste des trois premiers membres de la table *Membre* qui résident dans les Yvelines, vous pouvez écrire :

```
SELECT * FROM Membre WHERE département="78" LIMIT 3
```



Certaines requêtes **SELECT** peuvent renvoyer des enregistrements identiques, mais lorsque vous ne voulez en voir qu'un seul, vous devez ajouter la clause **DISTINCT** pour n'obtenir que le premier. Exemple :

```
SELECT DISTINCT * FROM Membre WHERE département="78"
```

Combinaison de tables

Dans les premières sections du Chapitre 1, j'ai supposé que toutes les informations sur lesquelles vous travailliez étaient situées dans

une seule table. Cependant, vous pouvez avoir besoin d'associer des informations provenant de différentes tables. Cela peut se faire de façon simple dans une seule requête.

Deux mots servent à combiner des informations provenant de diverses tables dans une requête `SELECT` :

- ✓ **UNION** : Les lignes sont extraites d'une ou plusieurs tables, et sont renvoyées ensemble, l'une après l'autre, dans un même résultat. Si votre requête sélectionne 6 lignes dans une table et 5 dans une autre, la réponse en contiendra 11.
- ✓ **JOIN** : Les tables sont combinées côte à côte et les informations récupérées dans les deux tables.

UNION

`UNION` sert à combiner les résultats produits par deux requêtes `SELECT` (ou davantage). Les réponses fournies par chaque requête sont ajoutées à celles de la précédente pour former le résultat final. La syntaxe de `UNION` se présente ainsi :

```
SELECT requête UNION ALL SELECT requête ...
```

Il est possible de combiner un nombre quelconque de requêtes `SELECT`. Chacune est susceptible de posséder son propre format, et y compris d'inclure de clauses `WHERE`, `LIMIT`, etc. Les règles à respecter sont les suivantes :

- ✓ Toutes les requêtes `SELECT` doivent fournir le même nombre de colonnes.
- ✓ Les colonnes sélectionnées dans les requêtes doivent posséder le même type de donnée.

Les *jeux* résultant contiendront toutes les lignes de la première requête, suivies de toutes les lignes de la seconde requête, et ainsi de suite. Les noms de colonnes utilisés dans ces jeux sont ceux qui sont fournis par la première requête `SELECT`.

Les requêtes `SELECT` successives peuvent sélectionner différentes colonnes d'une même table. Cependant, il est rare de trouver une situation dans laquelle vous avez besoin d'une certaine colonne d'une table, suivie d'une autre colonne de cette table. Le but est généralement de combiner des colonnes provenant de différentes

tables. Supposons par exemple que votre base de données contienne d'un côté la liste des personnes qui ont quitté le club, et de l'autre celle des membres actuels de celui-ci. Vous pouvez facilement obtenir la liste des anciens comme des modernes à l'aide d'une instruction comme celle-ci :

```
SELECT nom,prénom FROM Membre UNION ALL  
SELECT nom,prénom FROM AncienMembre
```

Le résultat produit par cette requête donne la liste des noms et prénoms de tous les membres actuels, suivie des noms et prénoms de tous les anciens membres.

Selon la manière dont vos données sont organisées, il se peut que des noms soient dupliqués. Rien n'interdit par exemple qu'une personne après avoir démissionné du club (elle peut alors figurer dans la table *AncienMembre*) reprenne son adhésion (et voit donc son nom ajouté à nouveau dans la table *Membre*). Pour éviter une telle situation, n'employez pas le mot *ALL*. En son absence, les lignes dupliquées n'apparaîtront pas dans le résultat.

La clause *ORDER BY* décrite dans la section précédente peut être utilisée dans chacune des requêtes *SELECT*, mais aussi dans la requête *UNION* elle-même afin de trier toutes les lignes du jeu résultant. Dans ce cas, utilisez des parenthèses de regroupement comme dans l'exemple qui suit :

```
(SELECT nom FROM Membre UNION ALL  
SELECT nom FROM AncienMembre) ORDER BY nom
```



L'instruction *UNION* a été introduite dans la version 4.0 de MySQL. Elle n'est pas disponible dans MySQL 3.

JOIN

Associer (combinaison) deux tables est appelé une *jointure*. Les tables sont combinées en associant les données correspondantes d'une colonne (celle qui est commune aux deux). Les résultats obtenus à partir d'une jointure contiennent les données provenant des colonnes des deux tables. Par exemple, si une table a deux colonnes, *ID* et *hauteur*, et que la seconde table a deux colonnes, *ID* et *poids*, la jointure résultera en une table à quatre colonnes : *ID* de la première table, *hauteur*, *ID* de la seconde table et *poids*.

Il existe deux types de jointures : les jointures *internes* (*inner*) et *externes* (*outer*). La différence entre ces deux types réside dans le nombre de lignes incluses dans la table qui en résulte. Dans le cas d'une jointure interne, on n'obtiendra que les lignes qui existaient dans les deux tables. Avec une jointure externe, en revanche, on trouve toutes les lignes d'une table et des vides dans les colonnes pour les lignes qui n'existent pas dans la seconde table. Par exemple, si *table1* contient une ligne pour Joseph et une pour Suzanne, et que *table2* ne contient qu'une ligne pour Suzanne, une jointure interne ne produira que la ligne de Suzanne, alors qu'une jointure externe donnera deux lignes : une pour Joseph et une pour Suzanne. Mais la colonne *poids* (pour reprendre le modèle du paragraphe précédent) de la ligne de Joseph sera vide.

Le jeu produit par une jointure externe contient toutes les lignes d'une table. Si une des lignes de cette table n'existe pas dans la seconde, les colonnes de cette dernière sont vides. En d'autres termes, le contenu renvoyé par la requête est déterminé par la table dont toutes les lignes contribuent au résultat, et la seconde table doit se conformer à cette exigence. Comment définir le maître et le valet dans une jointure externe ? En spécifiant la source principale à l'aide des mots `LEFT JOIN` et `RIGHT JOIN`.

C'est dans la requête `SELECT` qu'on spécifie la jointure. Pour une jointure interne, ce sera quelque chose comme :

```
SELECT listeCol FROM table1,table2
      WHERE table1.col2 = table2.col2
```

alors que, pour une jointure externe, on écrirait au choix :

```
SELECT listeCol FROM table1 LEFT JOIN table2
      ON table1.col1 = table2.col2
```

```
SELECT listeCol FROM table1 RIGHT JOIN table2
      ON table1.col1 = table2.col2
```

Dans ces trois requêtes, *table1* et *table2* sont les tables composant la jointure. (Il est possible de joindre plus de deux tables.) *col1* et *col2* sont les noms des colonnes à faire correspondre pour joindre les tables. Cette correspondance s'établit d'après les données présentes dans ces deux colonnes. Elles peuvent ou non avoir le même nom. Mais elles doivent contenir le même type de données.

Pour donner un exemple de jointure, nous allons prendre un court extrait de notre catalogue d'animaux. L'une des tables est *Animal*,

dans laquelle nous retiendrons les colonnes `animalNom` et `animalType` contenant les informations suivantes :

nom	type
Licorne	Cheval
Pégase	Cheval
Lion	Chat

L'autre table est `Couleur`, avec deux colonnes, `animalNom` et `animalCouleur`, qui contiennent les données suivantes :

nom	couleur
Licorne	blanc
Licorne	argent
Poisson	Doré

Vous voulez trouver la réponse à une question nécessitant la jointure de ces deux tables. Si vous faites une jointure interne avec la requête suivante :

```
SELECT * FROM Animal,Couleur WHERE Animal.animalNom = Couleur.animalNom
```

vous allez obtenir la table de résultats suivante contenant quatre colonnes : `nom` (provenant de `Animal`), `type`, `nom` (provenant de `Couleur`) et `couleur` :

nom	type	nom	couleur
Licorne	Cheval	Licorne	blanc
Licorne	Cheval	Licorne	argent

Remarquez que `Licorne` apparaît dans la table de résultat, parce que seule `Licorne` se trouvait dans les deux tables originales, avant la jointure.

D'un autre côté, supposons que vous fassiez une jointure externe à l'aide de la requête suivante :

```
SELECT * FROM Animal LEFT JOIN Couleur
ON Animal.animalNom = Couleur.animalNom
```

Vous obtiendrez alors les résultats suivants avec les quatre mêmes colonnes : `nom` (provenant de `Animal`), `type`, `nom` (provenant de `Couleur`) et `couleur` :

nom	type	nom	couleur
Licorne	Cheval	Licorne	blanc
Licorne	Cheval	Licorne	argent

Pégase	Cheval	<NULL>	<NULL>
Lion	Chat	<NULL>	<NULL>

Cette table a quatre lignes. Les deux premières sont identiques à celles que nous avons obtenues avec une jointure interne. Dans les deux suivantes, les champs `nom` (provenant de `Couleur`) et `couleur` restent vides.

Essayons maintenant de voir ce que donnerait une jointure externe droite. La requête doit être légèrement modifiée :

```
SELECT * FROM Animal RIGHT JOIN Couleur
      ON Animal.animalNom = Couleur.animalNom
```

Vous obtenez alors une table qui affiche les mêmes colonnes, mais avec des lignes différentes :

nom	type	nom	couleur
Licorne	Cheval	Licorne	blanc
Licorne	Cheval	Licorne	argent
<NULL>	<NULL>	Poisson	Doré

Vous remarquerez que l'on trouve toutes les lignes de la table `Couleur` (celle de droite dans la requête), mais pas de la table `Animal`. Cette dernière ne possédant pas de ligne *Poisson*, les colonnes correspondantes sont vides.



Le type de jointure à utiliser dépend de la question à laquelle vous voulez répondre. Si vous voulez obtenir une liste des animaux qui sont de couleurs différentes, utilisez une jointure interne. Si vous voulez une liste de tous les animaux, utilisez une jointure externe.

Les jointures abordées ci-dessus servent à rechercher des entrées qui se correspondent dans des tables. Mais on peut tout aussi bien se poser la question inverse. Supposons que vous vouliez savoir qui ne s'est *jamais* connecté dans votre liste de membres privés. Vous disposez d'une table pour les noms des utilisateurs, et d'une autre pour les dates et heures de connexion. Réaliser une jointure pour afficher toutes les correspondances et retrouver les absents serait extrêmement fastidieux, si ce n'est impossible avec une base de données importante. Mais une simple astuce vous aidera à contourner la difficulté. Il suffit en effet de sélectionner uniquement les noms *vides* dans la jointure, comme ceci :

```
SELECT login from Membre LEFT JOIN Login
      ON Membre.login=Login.login
      WHERE Login.login IS NULL
```

Cette requête vous donne une liste de tous les noms d'utilisateurs de la table `Membre` qui ne figurent pas dans la table `Login`.

Mise à jour des informations

Mettre à jour des informations consiste à modifier les données contenues dans une ligne. Par exemple, vous pouvez souhaiter modifier l'adresse d'un membre parce qu'il a déménagé, ou ajouter un numéro de fax pour un membre qui n'était pas équipé de cet appareil auparavant.

Pour effectuer une mise à jour, on utilise une requête `UPDATE` dont la forme générale est :

```
UPDATE nomTable SET nomCol1=valeur1,nomCol2=val2... WHERE clause
```

Dans la clause `SET`, on fait figurer des couples de la forme `nom=valeur` pour les colonnes dont on veut modifier la valeur. Sans clause `WHERE`, toutes les lignes de la table seront modifiées. Cette clause permet de limiter la portée de la mise à jour aux seules lignes à modifier. Considérez, par exemple, la mise à jour d'une adresse de la table `Membre` :

```
UPDATE Membre SET rue="45 rue de la Cloche",  
                 téléphone="0456789012"  
WHERE login="monToto"
```

Suppression d'informations

Les informations périmées doivent être supprimées si vous voulez que votre base de données soit le reflet de l'existant. Pour effectuer une suppression, vous pouvez utiliser la requête `DELETE` :

```
DELETE FROM nomTable WHERE clause
```



N'usez de cette requête qu'avec la plus grande prudence, car les informations ainsi supprimées sont irrécupérables. Pis, si vous omettez la clause `WHERE`, vous allez supprimer définitivement **toutes** les informations contenues dans la table. Aucune récupération n'est possible. C'est le genre de commande qui se trouve au premier rang de la liste à-ne-jamais-essayer-juste-pour-voir-avant-de-filer-au-lit.

Vous pouvez supprimer une colonne dans une table au moyen de la requête `ALTER` :

```
ALTER TABLE nomTable DROP nomCol
```

Pour supprimer la totalité des éléments d'une table, vous devez utiliser la requête DROP :

```
DROP TABLE nomTable
```

ou, plus globalement :

```
DROP DATABASE nomBase
```


Chapitre 5

Protection de vos données

Dans ce chapitre :

Comment fonctionne la sécurité avec MySQL.

Ajout de nouveaux comptes MySQL.

Modification des comptes existants.

Modification des mots de passe.

L'art de la sauvegarde.

Réparation des données.

Restauration des données.

Vos données sont essentielles au bon fonctionnement de votre application de base de données sur le Web. Vous avez passé beaucoup de temps à développer votre base de données, et maintenant elle contient d'importantes informations que vous ou vos utilisateurs y avez placées. Il est temps d'assurer leur protection. Dans ce chapitre, vous allez apprendre comment procéder.

Contrôle des accès à vos données

MySQL propose un système de sécurisation pour protéger vos données. Personne ne peut y accéder sans être titulaire d'un compte. Chaque compte MySQL a les attributs suivants :

- ✓ Un nom.
- ✓ Un nom d'hôte (celui de la machine à partir de laquelle le compte permet d'accéder à vos données sur le serveur MySQL).

- ✓ Un mot de passe.
- ✓ Un jeu de droits d'accès (permissions ou privilèges).

Celui ou celle qui veut accéder à vos données doit montrer patte blanche, c'est-à-dire utiliser un compte MySQL associé à un mot de passe, tous deux en état de validité. En outre, cette personne doit se connecter à partir d'un ordinateur ayant également la permission de se connecter à votre base de données avec ce compte et ce mot de passe.

Une fois qu'un utilisateur s'est vu accorder l'accès à la base de données, ce qu'il va être autorisé à faire dépend des droits d'accès qui ont été définis pour ce compte. Chaque compte a ou non la permission d'exécuter sur votre base de données des opérations telles que `SELECT`, `DELETE`, `INSERT`, `CREATE`, `DROP`, etc. Vous pouvez accorder tous les droits (également appelés *privilèges* ou *permissions*) à n'importe quel compte, ne lui en accorder aucun ou seulement quelques-uns. Par exemple, pour protéger votre catalogue en ligne, s'il est normal qu'un utilisateur puisse le consulter, mais il vaut mieux qu'il ne soit pas autorisé à le modifier.

Comptes et noms d'hôtes

Le couple "*nom de compte/nom d'hôte*" (le nom de l'ordinateur autorisé à se connecter à la base de données) identifie un compte unique. Deux comptes de même nom peuvent coexister pour peu que leurs noms d'hôtes soient différents. Ils peuvent, dans ce cas, avoir des mots de passe et des droits d'accès différents. En revanche, il est impossible d'avoir deux comptes ayant le même nom et le même hôte.

Lorsque vous exécutez une requête `GRANT` ou `REVOKE` (voir plus loin dans ce chapitre), vous devez être identifié par le couple compte/nom d'hôte sous la forme suivante : *compte@nom_d'hôte* (par exemple : *root@localhost*).

Les noms de comptes MySQL sont ainsi définis :

- ✓ **Un nom de compte peut comporter 16 caractères au plus.** Vous pouvez utiliser des caractères spéciaux tels qu'un tiret (-) ou un blanc souligné (_). Vous ne pouvez pas utiliser de caractère joker.

- ✓ **Un nom de compte peut être vide.** Dans ce cas, tout nom de compte non vide sera reconnu comme valable ; ainsi, n'importe qui pourra se connecter sous son propre nom, pour peu que son nom d'hôte et son mot de passe (si nécessaire) soient reconnus comme valides. Un nom de compte vide permet à des utilisateurs anonymes de se connecter à votre base de données.
- ✓ **Le nom d'hôte peut être soit un nom, soit une adresse IP.** Par exemple : `thor.monserveur.com` ou `192.163.2.33`. La machine sur laquelle est installé MySQL a pour nom `localhost`, ce qui correspond à l'adresse IP `127.0.0.1`.
- ✓ **Des caractères jokers peuvent être utilisés dans le nom d'hôte.** Le caractère pour-cent (%) correspond à n'importe quel nom d'hôte. Si le nom de compte `georges@%` existe, le nom `georges` sera reconnu comme valide, quel que soit l'ordinateur à partir duquel la connexion est établie.
- ✓ **Le nom d'hôte peut être vide.** Un nom d'hôte vide est l'équivalent de %.

Un compte dont le nom d'utilisateur et le nom d'hôte associé sont tous deux vides peut exister. N'importe qui peut alors se connecter à la base de données, sous n'importe quel nom et à partir de n'importe quel ordinateur. C'est la même chose si le nom d'hôte est %. En dehors de l'utilisation **locale** d'un serveur personnel pour la mise au point d'une base de données et de scripts PHP (comme c'est le cas avec EasyPHP), cette situation est extrêmement dangereuse. Un tel compte est parfois installé automatiquement avec MySQL, mais il ne possède aucun droit d'accès et ne peut donc rien faire.



Avec certains systèmes d'exploitation, d'autres comptes, de la famille de `root`, sont automatiquement installés. Par exemple, sous Windows, le compte `root@%` peut avoir été installé sans protection par mot de passe. Le compte `root` ayant tous les droits d'accès peut être utilisé par n'importe qui depuis n'importe quelle machine. Il est donc prudent de supprimer immédiatement ce compte ou, à tout le moins, de le compléter par un mot de passe.

Quelques mots sur les mots de passe

Pour augmenter la sécurité, les mots de passe sont conservés sous forme cryptée. Malheureusement, des petits malins parviennent à

les reconstituer. Pour cela, ils utilisent des moyens plus ou moins évolués dont le plus simple consiste à utiliser un logiciel qui essaie à grande vitesse une multitude d'expressions. On les appelle *hackers* ou *crackers*. Pour déjouer (dans la limite du possible) leurs tentatives, voici quelques recommandations à suivre dans le choix d'un mot de passe :

- ✓ Adoptez un mot de passe de 6 à 8 caractères.
- ✓ Utilisez un mélange de caractères : majuscules, minuscules, chiffres, caractères spéciaux...
- ✓ N'utilisez ni votre nom de login, ni votre nom de compte MySQL ni aucune variante de ces noms.
- ✓ Ne prenez pas un mot qui se trouve dans le dictionnaire.
- ✓ Ne choisissez pas un nom propre (celui de l'élue) de votre cœur ou de votre animal favori, par exemple).
- ✓ N'utilisez ni numéro de téléphone ni date.

Les droits d'accès

A chaque base de données, table ou même colonne est attaché un jeu de droits d'accès particulier. Par exemple, un compte MySQL peut être défini de façon que son titulaire puisse lire des données à partir des tables de la base, mais n'être autorisé à modifier qu'une seule colonne d'une certaine table, nommément désignée, et à insérer des données dans une autre, également spécifiée.

Les droits d'accès sont accordés par la requête `GRANT` (*accorder*) et supprimés par la requête `REVOKE` (*révoquer*). Ces requêtes ne peuvent être lancées qu'à partir d'un compte ayant le droit de les utiliser. Toute tentative non autorisée entraînera l'affichage d'un message d'erreur. Par exemple, si vous essayez de vous accorder le droit d'utiliser une requête qui vous est normalement "interdite" à partir d'un compte qui n'est pas autorisé à utiliser ces commandes, vous obtiendrez comme seul résultat le message :

```
grant command denied
```

Les droits d'accès peuvent être accordés ou révoqués un par un ou globalement. Le Tableau 5.1 donne une vue des principaux droits que vous pouvez accorder ou révoquer.

Tableau 5.1 : Droits d'accès attachés aux comptes MySQL.

Droit d'accès	Description
ALL	Tous les droits.
ALTER	Modifier la structure des tables.
CREATE	Créer de nouvelles bases et/ou de nouvelles tables.
DELETE	Supprimer des lignes dans une table.
DROP	Supprimer des bases et/ou des tables.
FILE	Lire et écrire des fichiers sur le serveur.
GRANT	Modifier les droits d'accès d'un compte MySQL.
INSERT	Insérer de nouvelles lignes dans une table.
SELECT	Lire les données d'une table.
SHUTDOWN	Fermer le serveur MySQL.
UPDATE	Modifier les données d'une table.
USAGE	Aucun droit d'accès.



Il n'est pas recommandé d'accorder le droit d'accès ALL qui est bien trop permissif. En particulier, cela autorise l'utilisateur à fermer le serveur MySQL. Un tel privilège ne doit échoir qu'à vous-même, divin administrateur de la base de données.

Création de comptes MySQL

Pour créer un nouveau compte, vous devez spécifier le mot de passe, le nom de l'ordinateur ou des ordinateurs autorisés à accéder à la base de données et les droits d'accès accordés (en sachant que vous pourrez toujours les modifier ultérieurement).

Toutes ces informations d'identification sont placées dans la base de données `mysql`, automatiquement créée lors de l'installation de MySQL. Pour faire ces opérations, vous devez vous-même avoir les droits d'accès nécessaires sur la base de données `mysql`.

Vous avez besoin d'au moins un compte MySQL pour accéder au serveur. Lorsque MySQL est installé, il crée automatiquement quelques comptes, y compris le compte appelé `root` qui possède tous les droits d'accès. Si vous utilisez les bons offices du département informatique de votre entreprise ou si vous accédez à MySQL

via un hébergeur ou un fournisseur d'accès, c'est l'administrateur du système hôte qui devra vous ouvrir ce compte. Dans ces conditions, ce dernier a peu de chances de s'appeler `root` et ses droits d'accès seront limités.



La base de données de sécurité de MySQL

Lors de l'installation de MySQL, une base de données particulière, `mysql`, est automatiquement créée. Elle servira à conserver les informations chargées de protéger vos données : noms des comptes, mots de passe, noms des hôtes et droits d'accès.

Les permissions sont sauvegardées en colonnes. Le format de chaque colonne est `permission_priv` où `permission` est l'un des droits d'accès énumérés au Tableau 5.1. Par exemple, la colonne contenant le droit d'accès `ALTER` a pour nom `alter_priv`. La valeur de chaque droit est Y (yes) ou N (no), selon que le droit est accordé ou non. Ainsi, par exemple, dans la table `user` (décrite un peu plus loin), il y a une ligne par compte et, dans cette ligne, une colonne pour `alter_priv`. Si cette colonne contient un Y, le compte sera autorisé à exécuter la requête `ALTER`. Si elle contient N, le compte ne pourra pas exécuter cette requête.

La base de données `mysql` contient cinq tables :

- ✔ **user.** Cette table contient les permissions qui s'appliquent à toutes les bases de données et à leurs tables. Elle contient une ligne pour chaque compte valide. On y trouve le nom d'utilisateur, le nom d'hôte et le mot de passe. Le serveur MySQL rejettera toute tentative de connexion émise par un compte qui ne figure pas dans cette table.
- ✔ **db.** Dans cette table se trouvent les droits d'accès s'appliquant à une base de données particulière. Elle contient une ligne pour la base de données et, dans cette ligne, on trouve les droits d'accès pour chaque couple compte/mot de passe existant. Les droits d'accès figurant dans la table `user` ont priorité sur ceux qui se trouvent dans cette table. Par exemple, si, dans la table `user`, figure une entrée pour le compte `projet`, qui accorde le droit d'accès `INSERT`, ce compte pourra insérer de nouvelles données dans toutes les bases de données. En revanche, si la ligne de la table `db` relative au compte `projet` contient un N pour le droit d'accès `INSERT` relatif à la table `AniCata`, il ne sera pas pris en compte et `projet` sera autorisé à insérer de nouvelles données.
- ✔ **host.** Cette table contrôle les accès à une base de données selon l'hôte qui émet la requête. Elle travaille en conjonction avec la table `db`. Si une ligne de cette table contient un champ vide pour l'hôte, MySQL recherche dans la

table une ligne relative à db. De cette façon, il est possible de permettre l'accès à une base de données à partir de certains hôtes et de l'interdire aux autres. Par exemple, supposons que vous ayez deux bases de données : bd1 et bd2. La première contient des informations "sensibles", ce qui vous incite à en limiter l'accès à une frange étroite d'utilisateurs. En revanche, bd2 contient des informations que tout le monde peut consulter. S'il y a une ligne dans la table db pour bd1 dont le champ hôte est vide, vous pouvez avoir deux lignes dans host pour bd1. L'une d'elles peut donner tous les droits aux utilisateurs connectés à partir d'un certain hôte, tandis que l'autre peut refuser tous les privilèges aux utilisateurs qui se connectent à partir de n'importe quel autre hôte.

- ✓ tables_priv. Cette table contient les droits d'accès qui s'appliquent à des tables spécifiques.
- ✓ columns_priv. Cette table contient les droits d'accès qui s'appliquent à certaines colonnes particulières.

Vous pouvez directement afficher le contenu des tables de mysql et les modifier si vous opérez à partir d'un compte ayant les droits d'accès nécessaires. Vous pouvez lancer des requêtes SELECT, INSERT, UPDATE et autres. Dans le cas général (utilisation du MySQL d'un hébergeur, d'un fournisseur d'accès ou du département informatique d'une entreprise), il y a gros à parier que ces droits vous seront refusés.

Identification des comptes existants

Pour savoir quels sont les comptes qui existent déjà, vous devez disposer vous-même d'un compte ayant les permissions nécessaires. Essayez d'exécuter la requête suivante sur la base de données mysql :

```
SELECT * FROM user
```

Vous devriez voir s'afficher la liste de tous les comptes. Cependant, si vous accédez à MySQL via le département informatique de l'entreprise, un hébergeur ou un fournisseur d'accès, vous n'aurez probablement pas les droits d'accès suffisants pour afficher cette liste. Vous obtiendrez alors le message suivant :

```
No Database Selected
```

(ou un autre avertissement vous apprenant que vous ne disposez pas du privilège d'employer la commande `SELECT`). Ce message signifie que vous n'êtes pas autorisé à afficher la liste des comptes MySQL existants. Il est rassurant dans la mesure où c'est une preuve de sécurité : n'importe qui ne peut pas accéder à n'importe quel compte. Par contre, cette situation est assez désagréable car vous ne pouvez même pas savoir quelles droits d'accès vous sont accordés. Vous devez pour cela vous renseigner auprès de votre administrateur MySQL ou tenter d'envoyer diverses requêtes afin de trouver celles dont l'issue est positive.

Ajout de nouveaux comptes et modification des droits d'accès

La meilleure façon d'accéder à MySQL à partir de PHP est de définir un compte créé spécifiquement dans ce but et pourvu du minimum de droits d'accès nécessaires. Dans cette section, nous allons voir ce qu'il faut faire pour y parvenir. Si vous accédez à MySQL autrement que par l'intermédiaire d'un serveur personnel, vous n'aurez probablement pas les droits d'accès suffisants pour créer un nouveau compte et modifier vos droits d'accès. Dans ce cas, vous ne pourrez pas exécuter la requête `GRANT` pour ajouter vous-même un compte. Il vous faudra alors un second point d'entrée pour vos scripts PHP.



Si vous avez besoin de demander un nouveau compte, choisissez-le pourvu de droits d'accès restreints (si c'est possible). De cette façon, votre application sera plus sécurisée.



Certains fournisseurs d'accès ne vous autorisent à créer qu'une seule base de données. Et encore ne le font-ils qu'à la suite d'une demande que vous devez formuler sur une page spécialisée de leur site Web. (N.d.T.)

La requête `CREATE USER` permet de créer un ou plusieurs comptes d'utilisateurs. Elle est apparue dans la version 5.0.2 de MySQL. Voici sa syntaxe générique :

```
CREATE USER nomcompte@nomhôte IDENTIFIED BY 'motpasse',
nomcompte@nomhôte IDENTIFIED BY 'motpasse',...
```

Cette requête crée le compte utilisateur mentionné avec un mot de passe, mais sans droits d'accès particuliers. Le mot de passe est facultatif : si vous ne mentionnez pas `IDENTIFIED BY 'motpasse'`, le compte sera créé avec un accès libre. Vous pouvez définir un mot

de passe ou en changer plus tard (nous expliquons comment plus loin).

Si vous utilisez une version de MySQL antérieure à la 5.0.2, vous devez utiliser la requête GRANT pour créer les comptes. Elle est décrite dans la section sur les changements des droits d'accès.

Ajout et modification des mots de passe

Pour intervenir sur le mot de passe d'un compte utilisateur, vous disposez de la requête SET PASSWORD :

```
SET PASSWORD FOR nomutilisateur@nomhôte = PASSWORD('motpasse')
```

Le mot de passe d'accès au compte *nomutilisateur@nomhôte* devient *motpasse*. La clause FOR est facultative ; si vous l'omettez, c'est le mot de passe du compte courant qui est modifié ou défini.

Pour supprimer le mot de passe d'un compte, vous définissez un mot de passe vide :

```
SET PASSWORD FOR nomutilisateur@nomhôte = PASSWORD('')
```

Modification des droits d'accès

La requête suivante permet de connaître les droits d'accès actuellement en vigueur pour un compte :

```
SHOW GRANTS FOR nomcompte@nomhôte
```

Le résultat affiché est une formulation exacte de la requête GRANT qui permettrait de créer ce compte avec ce jeu de droits d'accès. Si vous omettez la clause FOR, vous affichez les droits d'accès du compte courant (celui à partir duquel la requête est émise).

Voici le format générique de la requête GRANT qui sert à modifier les droits d'accès d'un compte :

```
GRANT permission (colonnes) ON nomtable  
TO nomcompte@nomhôte IDENTIFIED BY 'motpasse'
```

Voici la signification des paramètres de GRANT :

- ✓ **permission (colonnes)** : Vous devez lister ici au moins un droit d'accès. Vous pouvez le restreindre aux seules colonnes

énumérées à la suite, entre parenthèses. Si cette liste est vide, le droit d'accès s'applique à toutes les colonnes de la table. Vous pouvez faire figurer autant de couples permission/colonne(s) que vous le souhaitez, en les séparant par des virgules. La liste des droits d'accès a été donnée dans le Tableau 5.1. Voici un exemple montrant le début d'une telle requête :

```
GRAND SELECT (prénom,nom), UPDATE, INSERT(date_naissance) ...
```

- ✓ **nomTable** : Liste des tables concernées par cette requête. Il en faut au moins une, évidemment. Si vous énumérez plusieurs tables dans cette liste, séparez-les par des virgules. Vous pouvez faire porter la requête GRANT sur toutes les tables de la base de données couramment sélectionnée en indiquant un astérisque (*) comme valeur pour ce paramètre. Si aucune base n'est à ce moment sélectionnée, la requête concernera toutes les tables de toutes les bases de données existantes.

Pour faire porter cette requête sur une certaine table d'une base de données particulière, *nombase*, donnez à ce paramètre la valeur : *nombase.nomTable*. L'utilisation d'un astérisque donne la valeur *tout* soit à la base, soit à la table. Par conséquent, la syntaxe *. * concernera toutes les tables de toutes les bases de données existantes.

- ✓ **compte@nom_d'hôte** : Le nom du compte intéressé. S'il existe déjà, c'est lui qui sera destinataire des droits d'accès spécifiés. S'il n'existe pas, il sera créé. L'identification exacte et complète du compte nécessite la présence des deux champs et de l'arobase qui les relie. S'il existe un compte portant le nom indiqué, mais pour un autre hôte, il n'est pas changé. Par contre, un nouveau compte est créé.
- ✓ **mot de passe** : Représente le mot de passe que vous ajoutez ou modifiez. Si vous ne voulez rien changer au mot de passe, omettez la clause IDENTIFIED BY 'motpasse'.

La requête GRANT permettant de créer un nouveau compte pour la base de données AniCata pourrait être :

```
GRANT SELECT ON AniCata.* TO jules@localhost
IDENTIFIED BY '5Feza37'
```


Suppression d'un compte ou de droits d'accès

Pour supprimer un compte, vous disposez de la requête `DROP USER` qui est apparue dans la version 4.1.1 de MySQL :

```
DROP USER nomcompte@nomhôte, nomcompte@nomhôte, ...
```

Le compte à partir duquel vous lancez la requête doit disposer de privilèges `DELETE` sur la base concernée pour que la requête `DROP USER` soit acceptée.

Le comportement exact de `DROP USER` a évolué entre les versions de MySQL. Dans la version 5.0.2, la requête supprime le compte et tous les enregistrements descriptifs qui concernent ce compte, y compris ceux qui désignent les droits d'accès aux bases et aux tables. Avant la version 5.0.2, `DROP USER` ne pouvait supprimer un compte qu'une fois que ce compte n'avait plus de droits d'accès. Il fallait donc d'abord supprimer tous les droits d'accès du compte avant de le supprimer.

C'est la requête `REVOKE` qui sert à supprimer des droits d'accès sur un compte existant. Voici sa syntaxe générique :

```
REVOKE permission (colonnes) ON nomTable FROM compte@nom_d'hôte
```

et la signification de ses paramètres :

- ✓ **permission (colonnes) :** Vous devez lister ici au moins un droit d'accès. Vous pouvez le supprimer pour une ou plusieurs colonnes, en énumérant ces colonnes entre parenthèses, séparées par des virgules. Si cette liste est vide, le droit d'accès est supprimé pour toutes les colonnes de la table. Vous pouvez faire figurer autant de couples `permission/colonne(s)` que vous le souhaitez, en les séparant par des virgules. La liste des droits d'accès figure dans le Tableau 5.1. Exemple de cette requête :

```
REVOKE SELECT (prénom,nom), UPDATE, INSERT(date_naissance) ...
```

- ✓ **nomTable :** Liste des tables concernées par cette requête. Il en faut au moins une, évidemment. Si vous énumérez plusieurs tables dans cette liste, séparez-les par des virgules. Vous pouvez faire porter la requête `REVOKE` sur toutes les tables de la base de données couramment sélectionnée, en indiquant un astérisque (*) comme valeur pour ce paramètre. Si aucune

base n'est à ce moment sélectionnée, la requête concernera toutes les bases de données existantes.

Pour faire porter cette requête sur toutes les tables d'une base de données particulière, `nombase`, donnez à ce paramètre la valeur : `nombase.nomTable`. L'utilisation d'un astérisque donne la valeur *tout* soit à la base, soit à la table. Par conséquent, la syntaxe `*.*` concernera toutes les tables de toutes les bases de données existantes.

- ✓ `compte@nom_d'hôte` : Le nom du compte intéressé. Il doit évidemment exister déjà. L'identification exacte et complète du compte nécessite la présence des deux champs et de l'arobase qui les relie. S'il existe un compte de même nom, mais pour un hôte différent, la requête `REVOKE` échouera et vous recevrez un message d'erreur.

Si vous ne voulez plus utiliser un compte, supprimez tous les droits d'accès attachés à ce compte. Pour cela, vous pouvez écrire :

```
REVOKE ALL ON *.* FROM compte@nom_d'hôte
```

Sauvegarde de données

Pour des raisons de sécurité évidentes, vous devez posséder au moins une copie de sauvegarde de vos données. Les désastres sont rares mais pas inexistantes. L'ordinateur sur lequel se trouve votre base de données peut tomber en panne et perdre vos données, les fichiers peuvent s'abîmer, l'immeuble prendre feu, etc. Une bonne copie de sauvegarde vous prémunira contre ces désastres.

Pour plus de sécurité, conservez cette sauvegarde en un lieu autre que celui où est installée votre base de données. Si vous êtes un peu parano, n'hésitez pas à faire deux, voire trois copies.

- ✓ Placez une copie à un endroit où vous pourrez facilement mettre la main dessus, éventuellement sur le site même de votre base de données, afin de minimiser le temps de restauration.
- ✓ Placez la deuxième copie sur un autre ordinateur, de façon à ne pas mettre tous vos œufs dans le même panier.
- ✓ Enfin, pour éliminer tout risque lié à des phénomènes tels que tremblements de terre, incendies, inondations, cyclones

(heureusement très rares en France), placez la troisième copie en un endroit éloigné de celui où réside votre base de données.

Si votre base de données est située chez un hébergeur, celui-ci pourra généralement prendre en charge (éventuellement moyennant un surcoût) la réalisation de copies de sauvegarde périodiques. Chez un fournisseur d'accès, n'y comptez pas. Sur le site informatique de votre entreprise, cela devrait pouvoir se négocier facilement avec le département informatique. Enfin, sur votre serveur personnel, vous êtes libre d'adopter la tactique qui vous semblera la meilleure.

Dans ce cas, je vous signale l'existence d'un utilitaire, `mysqldump`, fourni en même temps que MySQL, qui peut créer un fichier texte contenant toutes les commandes MySQL nécessaires à la restauration de votre base, de toutes ses tables et de chaque ligne de donnée (grâce aux requêtes `CREATE` et `INSERT`). Vous pouvez ainsi restaurer la base de données sur le site même où elle se trouvait ou sur un autre ordinateur (pour peu que ce dernier ait une configuration MySQL compatible avec celle de l'ancien site).

Voici comment procéder pour effectuer une copie de sauvegarde sous Linux/UNIX/Mac :

1. Placez-vous dans le sous-répertoire `bin` du répertoire dans lequel est installé MySQL.

Par exemple, tapez la commande :

```
cd /usr/local/mysql/bin
```

2. Tapez ensuite :

```
mysqldump --user=compte --password=mpasse  
nomBase > chemin_d'accès/fichier_sauvegarde
```

avec :

- `compte` : Nom du compte MySQL que vous utilisez pour sauvegarder votre base de données.
- `mpasse` : Mot de passe de ce compte.
- `nomBase` : Nom de la base de données que vous voulez sauvegarder.

- `chemin_d'accès/fichier_sauvegarde` : Chemin d'accès et nom du fichier destinataire de la sauvegarde qui sera produite par la sortie SQL.



Le compte que vous utilisez doit avoir le droit d'accès `SELECT`. S'il ne demande pas de mot de passe, ignorez le paramètre `--password`.

Une fois la commande tapée sur une seule ligne, appuyez sur <Entrée>. Si vous avez besoin de plusieurs lignes pour saisir la totalité de la commande, terminez chaque ligne intermédiaire par un antislash (\) suivi de <Entrée>.

Voyons comme exemple la commande permettant de faire une copie de sauvegarde du fichier `AniCata` :

```
mysqldump --user=root --password=bigsecret AniCata \
>/usr/local/mysql/backups/AniCataCopie
```

Le compte Linux/UNIX sous lequel vous êtes logé doit disposer de la permission d'écrire un fichier dans le répertoire de sauvegarde.

Voici comment procéder pour effectuer une copie de sauvegarde sous Windows :

1. Ouvrez une fenêtre d'invite de commandes.

Sous Windows XP/2000, il suffit d'ouvrir cette fenêtre à partir du groupe de programmes Accessoires.

2. Dans la fenêtre Invite de commandes, placez-vous dans le répertoire où est installé MySQL.

Par exemple, tapez la commande :

```
CD C:\MYSQL\BIN
```

3. Tapez ensuite :

```
mysqldump --user=compte --password=mpasse
nomBase > chemin_d'accès/fichier_sauvegarde
```

avec :

- `compte` : Nom du compte MySQL que vous utilisez pour sauvegarder votre base de données.

- `mpasse` : Mot de passe de ce compte.
- `nomBase` : Nom de la base de données que vous voulez sauvegarder.
- `chemin_d'accès/fichier_sauvegarde` : Chemin d'accès et nom du fichier destinataire de la sauvegarde qui sera produite par la sortie SQL.

Le compte que vous utilisez doit avoir le droit d'accès `SELECT`. S'il ne demande pas de mot de passe, ignorez le paramètre `-password`.



Vous devez saisir la commande sur une seule ligne avant de frapper la touche <Entrée>.

Voyons comme exemple la commande permettant de faire une copie de sauvegarde du fichier `AniCata` :

```
mysqldump --user=root AniCata>/AniCataCopie
```

Pour bien faire, il est nécessaire d'effectuer périodiquement des copies de sauvegarde : au moins une fois par jour si votre site Web est fréquemment consulté ; plus souvent, s'il l'est davantage. A la limite, faites une copie toutes les heures et une copie sur un autre ordinateur une fois par jour.

Restauration de vos données

Si, à un moment quelconque et pour un motif imprévisible (problème lié au matériel, arrêt inopiné de l'ordinateur), les données d'une de vos bases de données sont endommagées, il peut arriver que le serveur SQL se bloque.

Voici le genre de message qui indique qu'une table est corrompue :

```
Incorrect key file for table: 'XXXXX'
```

Dans certains cas, il est possible de réparer ce dommage au moyen d'un utilitaire fourni en même temps que MySQL. Si cela ne suffit pas, rien n'est perdu : vous pouvez toujours remplacer les tables endommagées par une copie de sauvegarde récente. La situation est différente si c'est toute la base qui est perdue (suite à une panne grave du disque dur, à l'effondrement du bureau sur lequel se trouvait l'ordinateur, ou encore à un lâche attentat). Vous devrez

alors remplacer la machine avant de restaurer votre base à partir d'une sauvegarde.

Vous devez restaurer la ou les tables corrompues à partir de votre copie de sauvegarde. Bien évidemment, les modifications intervenues entre le moment où cette sauvegarde a été effectuée et l'instant où est survenu l'incident seront perdues et vous devrez —si possible— les recréer manuellement.

Comme toujours, en dehors de l'utilisation d'un serveur personnel, vous devrez faire appel aux bons offices de votre hébergeur ou de votre fournisseur d'accès, selon l'endroit où est installée votre base de données. Lorsque vous êtes le propre administrateur de MySQL, vous pouvez procéder à la restauration vous-même.

Ainsi que nous l'avons vu à propos de l'utilitaire `mysqldump` (Chapitre 4), la copie de sauvegarde est un fichier texte qui se compose de requêtes `CREATE` et `INSERT` recréant la base à partir de zéro.

Votre base n'existe peut-être plus ou bien certaines de ses tables sont corrompues. Voici comment restaurer une table :

1. **Si la table existe encore, supprimez-la en lançant la requête MySQL :**

```
DROP TABLE nomTable
```

où *nomTable* est le nom de la table à supprimer.

2. **Chargez dans votre navigateur le document HTML de maintenance `mysql_envoi.php`.**

Il a été décrit dans le Chapitre 4.

3. **Copiez la commande `CREATE` de la table à partir du fichier de sauvegarde dans la grande boîte de saisie affichée dans le navigateur.**

Par exemple, par Edition/Copie de l'éditeur de texte puis (sous Windows : <Ctrl>+<V>) dans le navigateur.

4. **Tapez dans la petite boîte de saisie le nom de la base de données que vous allez créer pour placer la restauration de la base de données.**
5. **Cliquez sur le bouton Envoyer la requête.**

Une nouvelle page s'affiche dans le navigateur montrant le résultat de la requête et proposant deux boutons.

6. Cliquez sur Nouvelle requête.
7. Copiez une requête INSERT (pour recréer une table) depuis l'éditeur de texte vers la grande boîte de saisie du navigateur.

Par exemple, par Edition/Copie de l'éditeur de texte puis (sous Windows : <Ctrl>+<V>) dans le navigateur.

8. Tapez dans la petite boîte de saisie le nom de la base de données que vous allez créer pour restaurer votre table.
9. Cliquez sur le bouton Envoyer la requête.

Une nouvelle page s'affiche dans le navigateur montrant le résultat de la requête et proposant deux boutons.

10. Cliquez sur Nouvelle requête.
11. Répétez les étapes 7 à 10 jusqu'à ce que toutes les requêtes INSERT aient été envoyées.

S'il y a vraiment trop de requêtes INSERT et CREATE pour que cette procédure soit réellement applicable, vous pouvez envoyer toutes les requêtes d'un seul coup en suivant les étapes ci-après. Vous devrez sans doute d'abord retoucher le fichier de sauvegarde ainsi :

1. Ouvrez le fichier de sauvegarde dans un éditeur de texte.
2. Cherchez la ligne qui indique le numéro de version du serveur (Server Versions).
3. Pour reconstruire toute une base, ajoutez l'instruction suivante après la ligne repérée à l'étape 2 :

```
CREATE DATABASE IF NOT EXISTS nombase
```

4. Après cette ligne, ajoutez une instruction pour désigner la base de données qui doit recevoir les tables à restaurer :

```
USE nombase
```

5. Etudiez les instructions de restauration des tables. Pour chaque table que vous ne voulez pas restaurer, neutralisez

la ligne en la transformant en commentaires. Ajoutez deux tirets en début de ligne (-).

6. Vérifiez les instructions `INSERT` de chaque table et neutralisez celles des tables que vous ne voulez pas modifier.

7. Enregistrez le fichier de sauvegarde modifié.

Une fois que ce fichier est conforme à vos désirs, il reste à appliquer les instructions ou votre base ou vos tables.

Voici comment procéder sous Linux/UNIX :

1. Placez-vous dans le sous-répertoire `bin` du répertoire où est installé MySQL.

```
cd /usr/local/mysql/bin
```

2. Tapez la commande suivante qui va transférer le contenu du fichier de sauvegarde dans la base de données maintenant vide :

```
mysql -u compte -p nomBase < chemin_d'accès/nomSauve
```

où `compte` est le nom du compte MySQL, `nomBase` celui de la base à restaurer, et `chemin_d'accès/nomSauve` le nom du fichier de sauvegarde précédé de son chemin d'accès complet. L'option `-p` est facultative. Si elle figure, le mot de passe associé au compte vous sera demandé. Par exemple, pour restaurer la base `AniCata`, la commande devrait se présenter ainsi :

```
mysql -u root -p AniCata < /usr/backupfiles/AniCata.bak
```

Voici comment procéder sous Windows :

1. Ouvrez une fenêtre d'invite de commandes.

Sous Windows XP/2000, il suffit d'ouvrir cette fenêtre à partir du groupe de programmes Accessoires.

2. Dans la fenêtre Invite de commandes, placez-vous dans le répertoire où est installé MySQL.

Par exemple, tapez la commande :

```
CD C:\Program Files\MYSQL\BIN
```

3. Tapez ensuite la commande suivante pour exécuter les requêtes de restauration :

```
mysql -u compte -p nomBase < chemin_d'accès\nomSauve
```

où *compte* est le nom du compte MySQL, *nomBase* celui de la base à restaurer, et *chemin_d'accès\nomSauve* le nom du fichier de sauvegarde précédé de son chemin d'accès. L'option *-p* est facultative. Si elle figure, le mot de passe associé au compte vous sera demandé. Par exemple, pour restaurer la base *AniCata*, la commande devrait se présenter ainsi :

```
mysql -u root -p AniCata < c:\mysql\back\AniCata.bak
```



Attention au sens du caractère de redirection ! Pour une *sauvegarde* avec *mysqldump*, c'est ">" ; alors qu'ici, pour une *restauration*, c'est le sens contraire : "<".

La restauration des tables peut demander un certain temps. En cas de problème, un message d'erreur sera affiché. En revanche, si tout se passe bien, aucun message ne vous en avisera. En fin de commande, l'invite réapparaît.

Bien entendu, toutes les modifications qui ont pu être opérées depuis la dernière sauvegarde seront perdues ! Vous devrez donc éventuellement, et si cela est possible, terminer le travail de récupération manuellement.

Mise à jour de MySQL

De temps à autre, une nouvelle version de MySQL est rendue disponible. Les informations concernant les mises à jour sont fournies dans le manuel Web de MySQL à l'adresse suivante :

<http://dev.mysql.com/doc/refman/5.0/fr/upgrade.html>

Avant toute mise à jour de MySQL, prenez bien soin d'effectuer une sauvegarde complète de vos bases de données, sans oublier les tables techniques comme celles contenant les droits d'accès.

MySQL déconseille de sauter des versions. Pour passer de la version MySQL 4.0 à MySQL 5.0, mieux vaut progresser d'une sous-version à la suivante, en testant le bon fonctionnement avant de passer à l'étape suivante. Autrement dit, vous passez de 4.0 à 4.1, vous testez et vous passez de 4.1 à 5.0.

Parfois, des incompatibilités surgissent dans une nouvelle version de MySQL. Certaines mises à jour modifient la structure des tables de droits d'accès (*grant tables*). MySQL 4.1 a ainsi modifié la méthode de cryptage des mots de passe en obligeant à créer des mots de passe plus sûrs dans les tables de droits.

Juste après une mise à jour, vous devez lancer le script `mysql_upgrade` qui va vérifier les fichiers, les réparer si nécessaire et mettre à jour les tables système en conséquence. Avant la version MySQL 5.0.19, ce script ne fonctionnait pas sous Windows, mais seulement sous Unix. Sous Windows, vous deviez utiliser le script `mysql_fix_privileges_tables` pour actualiser les tables système, mais il ne procédait pas au test et à la réparation de toutes les tables, comme le fait `mysql_upgrade`.

Troisième partie

PHP



"Votre base de données est en cours de réparation. Cependant, avant que je vous fasse quelques recommandations concernant les sauvegardes, permettez-moi de vous poser une question. Combien de fiches cartonnées pensez-vous pouvoir ranger dans votre salle machine ?"

Dans cette partie...

Dans cette partie, vous allez apprendre comment utiliser PHP pour réaliser votre application de base de données sur le Web. Voici quelques-uns des sujets qui seront abordés :

- ✓ Comment incorporer du PHP dans un document HTML.
- ✓ Quelles sont les fonctionnalités de PHP qui peuvent servir à construire une application de base de données sur le Web ?
- ✓ Comment tirer le meilleur parti des spécificités de PHP.
- ✓ Comment utiliser des formulaires pour collecter des informations provenant des utilisateurs.
- ✓ Comment afficher des informations provenant de bases de données dans une page Web.
- ✓ Comment enregistrer des informations dans une base de données.
- ✓ Comment passer des informations d'une page Web à une autre.

Vous allez trouver ici tout ce que vous devez savoir pour écrire les programmes PHP dont vous avez besoin.

Chapitre 6

A la découverte de PHP

Dans ce chapitre :

- Comment ajouter des sections écrites en PHP dans un document HTML.
- Ecriture des instructions PHP.
- Utiliser des variables PHP.
- Comparaison de valeurs contenues dans des variables PHP.
- Mettez des commentaires dans vos programmes.

PHP est le langage que vous allez utiliser pour écrire vos programmes. C'est un langage de script conçu spécialement pour être utilisé sur le Web. Dans ce chapitre, nous verrons les règles générales qui s'appliquent à la rédaction des instructions PHP. Comme pour l'écriture des phrases du langage, il existe des règles de grammaire et de ponctuation pour les instructions. Dans les chapitres suivants, je vous parlerai des instructions PHP proprement dites et vous montrerai comment les utiliser pour accomplir des tâches spécifiques.

Comment ajouter des sections écrites en PHP dans un document HTML

PHP est un partenaire de HTML (*Hypertext Markup Language*) dont il accroît les possibilités. Il permet à une page Web de faire plus de choses qu'avec le seul recours à HTML. Par exemple, les programmes HTML sont capables d'afficher des pages Web formatées. Ces pages peuvent contenir des images, et même jouer de la musique. Mais ils ne vous servent à rien pour interagir avec les personnes qui visualisent ces pages.

L'interactivité de HTML est donc faible. Les formulaires permettent bien aux utilisateurs de taper des informations qui seront collectées par la page Web. Cependant, vous ne pouvez pas accéder à ces informations sans recourir à un autre langage que HTML. De son côté, PHP est capable de traiter les informations contenues dans un formulaire sans avoir besoin du secours d'un autre programme.

Certaines balises HTML servent à séparer ce qui est PHP de ce qui est HTML. Le fichier qui contient l'ensemble possède une extension spécifique, généralement `.php` (du moins, normalement). Les instructions PHP sont encadrées par deux balises particulières : `<?php` (balise initiale ou ouvrante) et `?>` (balise terminale ou fermante).



Comment le serveur Web traite le code PHP

Lorsqu'un navigateur demande un fichier HTML dont l'extension est `.htm` ou `.html`, le serveur Web lui envoie le fichier tel quel. Le navigateur traite le fichier et affiche la page Web qu'il décrit sur son écran. Lorsqu'il demande un fichier PHP dont l'extension est `.php`, au lieu d'envoyer directement le fichier au navigateur, le serveur Web commence par rechercher les sections PHP parmi le code HTML. Voyons comment il procède :

1. Il explore le fichier et envoie toutes les lignes contenant du HTML pur au navigateur.
2. Lorsqu'il rencontre une balise PHP initiale (`<?php`), il passe en "mode PHP" et envoie toutes les lignes qu'il rencontre à partir de ce moment à l'interpréteur PHP.
3. Celui-ci lui renvoie éventuellement des sorties que le serveur va alors faire parvenir au navigateur.
4. Lorsqu'il rencontre une balise PHP terminale (`?>`), le serveur Web cesse d'envoyer les lignes du fichier HTML à l'interpréteur PHP. Il repasse alors en "mode HTML".
5. Les étapes 1 à 4 sont alors réexécutées, car il peut y avoir plusieurs sections PHP dans un même fichier HTML.

C'est l'interpréteur PHP qui va traiter tout ce qui se trouve entre ces deux balises, puis la section PHP disparaît, remplacée par le résultat de ce traitement. Lorsque ce traitement produit des sorties destinées à être affichées, ce sont elles qui remplacent le code PHP

initial. De la sorte, le navigateur ne voit jamais la section PHP, mais seulement le résultat obtenu par son traitement.

A titre d'exemple, considérons le petit document HTML du Listing 6.1 qui, traditionnellement, affiche "Hello World!" sur l'écran du navigateur.

Listing 6.1 : Document HTML affichant *Hello World!*

```
<html>
<head>
<title>Affichons Hello World</title>
</head>
<body>
<p>Hello World!
</body>
</html>
```

Si vous chargez ce document HTML dans votre navigateur, vous affichez dans sa fenêtre les deux mots :

Hello World!

Le Listing 6.2 vous présente un programme PHP qui fait exactement la même chose.

Listing 6.2 : Document HTML affichant *Hello World!* avec l'aide de PHP.

```
<html>
<head>
<title>Affichons Hello World</title>
</head>
<body>
<?php
    echo "<p>Hello World!";
?>
</body>
</html>
```

Ne chargez pas directement ce fichier dans votre navigateur à l'aide de la commande Ouvrir du menu Fichier. Vous devez entrer son



nom et son emplacement exacts dans la barre d'adresses, comme dans `http://localhost/hello.php`. Si la fenêtre affiche du code PHP, vous avez choisi la mauvaise méthode !

La section PHP de ce document HTML est constituée par :

```
<?php
    echo "<p>Hello World!"
?>
```

Les balises PHP encadrent une seule instruction PHP : `echo`. Celle-ci demande à PHP d'afficher ce qui suit le mot "echo" dans la fenêtre du navigateur. Ici, il s'agit de ce qui se trouve entre les guillemets.

Il n'existe aucune règle vous obligeant à écrire les balises PHP sur une ligne séparée. Vous pouvez fort bien réécrire les trois lignes ci-dessus de la façon suivante :

```
<?php echo "<p>Hello World!"?>
```

L'interpréteur PHP remplace l'instruction `echo` par le texte placé entre guillemets et renvoie ce texte (privé de ses guillemets) au serveur Web, lequel, à son tour, l'envoie au navigateur. Pour vous en convaincre, regardez le code du document HTML renvoyé au navigateur en cliquant sur la commande Source dans le menu Affichage.

Écriture des instructions PHP

Dans un fichier HTML, une section PHP contient une série d'instructions écrites dans le langage de PHP. A chaque instruction correspond une certaine action. C'est ce que nous venons de voir sur un exemple très simple.



Toute instruction PHP doit être terminée par un point-virgule (;). En dehors des chaînes de caractères (placées entre guillemets ou entre apostrophes, comme nous le verrons un peu plus loin), PHP ignore tout ce qui est "espace blanc", c'est-à-dire les espaces vrais, les tabulations, les alinéas et les retours chariot. Il continue à lire ce qui lui est envoyé jusqu'à ce qu'il rencontre un point-virgule ou la balise PHP terminale. L'oubli du point-virgule à la fin d'une instruction est une erreur très fréquente chez les débutants. Il en résulte un message du genre de celui-ci :

```
Parse error: parse error in c:\program files\easyphp\www\hello.php on line 7
```

Remarquez le numéro de ligne indiqué dans le message pour vous aider à localiser l'erreur. En général, il désigne la ligne qui suit l'instruction à laquelle manque son point-virgule final.

Des instructions peuvent être regroupées pour former un *bloc*. Elles sont alors placées entre une paire d'accolades ({ }). Un bloc d'instructions est une sorte de "superinstruction" qu'on doit considérer comme un tout. L'un des usages fréquents d'un bloc est le *bloc conditionnel* dont les instructions ne sont exécutées que si une certaine condition est satisfaite. En voici un exemple symbolique :

```
si (le ciel est bleu)
{ passez la laisse au dragon;
  emmenez le dragon en promenade dans le parc;
}
```

Si "le ciel est bleu", la condition est vérifiée et le bloc qui suit est alors exécuté. Ce bloc comprend les deux "instructions" : "passez sa laisse au dragon;" et "emmenez le dragon en promenade dans le parc;" qui sont toutes deux exécutées, l'une après l'autre. Si le ciel était gris, la condition ne serait pas vérifiée et le dragon resterait dans sa niche.

Les instructions PHP qui font appel à des blocs sont dites *complexes* (nous en verrons d'autres dans le Chapitre 7). PHP lit le bloc en entier, sans s'arrêter au premier point-virgule qu'il rencontre. Il sait reconnaître la présence d'un bloc (ou de plusieurs blocs imbriqués), et il recherche donc l'accolade terminale. Remarquez aussi la présence d'un point-virgule juste avant cette accolade. Elle est indispensable. Par contre, le symbole de fin de bloc n'a pas à être suivi de ce caractère.

Il n'est pas impossible d'écrire la totalité d'une section PHP sous la forme d'une seule longue ligne dans laquelle chaque instruction est séparée de la suivante par un point-virgule. Mais il faut être réaliste : un tel programme serait quasiment impossible à relire. C'est la raison pour laquelle on prend en général la sage habitude de n'écrire qu'une instruction par ligne. Toutefois, des "dérogations peuvent être accordées" lorsqu'il s'agit de très courtes instructions.



Prenez l'habitude d'*indenter* vos lignes d'instructions. C'est-à-dire d'aligner les blocs logiques en ajoutant des espaces en début de ligne. Cette méthode n'influe pas sur le comportement de PHP. Uniquement sur la lecture des listings par des êtres humains. Qui en ont bien besoin pour comprendre de quoi il s'agit.



En général, PHP ne se soucie pas de la différence de casse (majuscules/minuscules) des instructions : `echo`, `Echo`, `ECHO`, `ecHo`... sont pour lui identiques. Cela est vrai pour les instructions et les noms de fonctions, mais pas pour les variables : `$Toto` et `$toto` représentent deux variables distinctes, comme vous allez le voir dans la section suivante.

Messages d'erreur et avertissements

Afin de vous faciliter la mise au point des programmes, PHP distingue plusieurs classes d'erreurs qu'on peut diviser en plusieurs groupes parmi lesquels on distingue les messages d'erreur et les avertissements.

- ✔ Erreurs d'analyse (parse errors). Ce sont des erreurs de syntaxe qui empêchent PHP de générer le code exécutable. Ce sont donc des erreurs fatales. Voici le message correspondant :

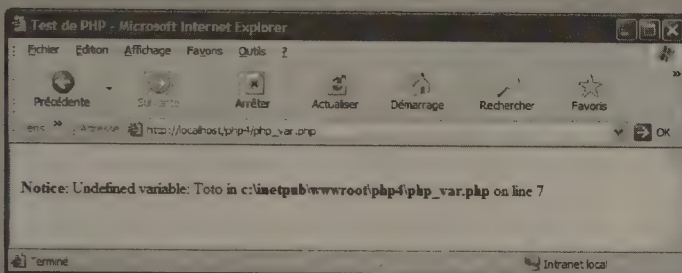
Parse error: erreur, in c:\test\test.php on line 6

En général, ce genre d'erreur est le résultat de l'oubli d'un délimiteur signifiant (point-virgule, parenthèse, accolade, etc.). Le message d'erreur tente d'en donner une explication. Par exemple, la mention `error might be unexpected T_ECHO, expecting ',' or ';'`` signifie que PHP a détecté une instruction `echo` sans virgule ou point-virgule en fin de ligne.

- ✔ Messages d'erreur. Ce message annonce la découverte d'une erreur suffisamment grave pour empêcher la poursuite de l'exécution du programme. Il contient des éléments permettant de localiser la source de l'erreur.
- ✔ Avertissement. C'est un message qui est affiché lorsque l'interpréteur PHP détecte une erreur "bénigne" qui ne risque pas, en principe, d'empêcher l'exécution du programme. Il signifie qu'il y a quelque chose d'anormal dans l'écriture d'une instruction, et que vous devriez y regarder de plus près pour vous assurer qu'il s'agit là de bien ce que vous vouliez écrire.
- ✔ Remarque. Avertissement bénin qui ne bloque pas le script, et qui indique que PHP est incapable de savoir si l'instruction est correcte ou non. Cela signifie simplement que vous avez dû faire quelque chose d'inhabituel et que vous devriez vérifier.

C'est ce qui se passe, par exemple, lorsque vous tentez d'afficher le contenu d'une variable qui n'existe pas. Vous verrez alors s'afficher le message reproduit sur la Figure 6.1.

Figure 6.1 :
Exemple de
message
d'avertisse-
ment.



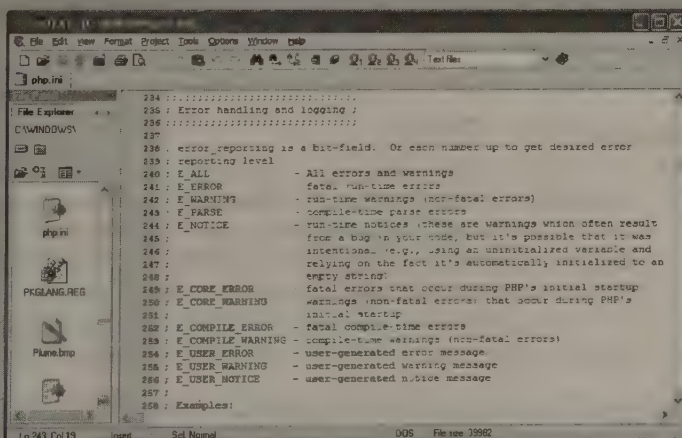
- **Strict.** Cette nouvelle catégorie de messages est apparue dans PHP 5 pour avertir d'une formulation malhabile ou dépassée dans le code source.

Quel que soit le type de l'erreur, le message contient toujours le nom du fichier et le numéro de la ligne qui a provoqué l'erreur.

Il existe plusieurs niveaux de gravité dans les erreurs. On peut spécifier à l'interpréteur qu'il ne doit afficher que les messages dont la gravité se situe à partir d'un certain niveau (par exemple, les remarques sont utiles en phase de développement, mais elles peuvent devenir gênantes lorsque le site est mis en ligne).

Pour cela, il faut disposer de privilèges d'administration sur PHP et éditer le fichier `php.ini`. Dans une de ses sections (`error_reporting`), vous trouverez une explication des niveaux d'erreur et apprendrez comment modifier le degré de signalisation. La Figure 6.2 montre comment se présente le début de cette section.

Figure 6.2 :
Début de la
section de
`php.ini`
concernant
les messages
d'erreur.



Voici trois exemples de définition de niveaux d'erreur :


```
error_reporting = E_ALL | E_STRICT
error_reporting = 0
error_reporting = E_ALL & ~ E_NOTICE
```

La première ligne fait s'afficher toutes les erreurs sauf les strictes par `E_ALL`, puis également ces dernières par `E_STRICT`. La deuxième ligne supprime tout affichage de messages. La dernière n'affiche que les erreurs et avertissements, mais pas les remarques ni les messages d'écriture stricte.

Effectuez la modification que vous souhaitez et sauvegardez le fichier. En général, il faudra relancer le serveur Web pour que cette modification soit prise en compte. Autrement dit, il n'y a que dans le cas d'un serveur personnel que vous pourrez effectuer cette modification.

Si vous n'êtes pas administrateur PHP pour votre système et que vous n'avez pas accès à `php.ini`, vous pouvez ajouter une instruction particulière afin de définir le niveau des rapports d'erreur dans le programme où elle est insérée. Placez simplement la ligne suivante au début de votre fichier source :

```
error_reporting(options)
```

options désigne un code définissant le niveau de signalisation des erreurs. Les valeurs possibles sont celles qui apparaissent sur la Figure 6.2. Pour voir toutes les erreurs sauf celles de la catégorie Strict, utilisez le format ci-dessous :

```
error_reporting(E_ALL);
```

Les variables PHP

Une variable est un conteneur utilisé pour renfermer une certaine information, pas nécessairement numérique. Elle a un nom. Par exemple, la variable `$age` peut contenir le nombre 12. Une fois placée à l'intérieur d'une variable, une information peut être utilisée dans tout le programme. L'une des utilisations les plus fréquentes d'une variable est de recevoir les informations saisies par l'utilisateur.

Ecriture des noms de variables

Pour donner un nom à une variable, vous devez respecter les règles suivantes :

- ✓ Toutes les variables doivent avoir un caractère dollar (\$) comme initiale.
- ✓ Un nom de variable peut comporter n'importe quel nombre de caractères.
- ✓ Un nom de variable peut comporter des lettres, des chiffres ou des traits de soulignement (_).
- ✓ A la suite du \$ initial, on ne peut trouver qu'une lettre ou un soulignement. Jamais un chiffre.
- ✓ Dans un nom de variable, PHP tient compte de la casse. \$Toto et \$toto représentent deux variables distinctes.

Pour appeler une variable, essayez de trouver un nom évocateur qui vous donne des indications sur ce qu'elle contient. A ce titre, évitez des noms comme \$var1, \$var2, \$A, \$b. La relecture et la maintenance de vos programmes en seront grandement facilitées. Préférez donc des noms comme \$age, \$prix, \$totalHT...

Affectation d'une valeur à une variable

Une variable peut contenir un nombre ou une chaîne de caractères. On range de l'information dans une variable au moyen de l'opérateur d'affectation : le signe égal (=). En voici quelques exemples :

```
$age = 12;
$prix = 2.55;
$nombre=-2;
$nom ="Goliath Smith"
```

Remarquez que les chaînes sont écrites entre guillemets. Pas les nombres. Nous y reviendrons dans la suite de ce chapitre.

Ces variables peuvent figurer dans le corps d'une instruction echo si on veut en afficher le contenu. Ainsi, l'instruction suivante :

```
echo $age;
```

affiche le nombre 12. Si vous la placez dans un document HTML en écrivant :

```
<p>Vous avez <?php echo $age ?> ans.
```

vous affichez :



Vous avez 12 ans.

Lorsque vous placez de l'information dans une variable qui n'existait pas encore, cette variable est créée. Considérons l'instruction suivante :

```
$nom = "Dupont";
```

Si c'est la première fois que la variable `$nom` apparaît dans le programme, l'exécution de cette instruction va créer cette variable et lui affecter la chaîne de caractères "Dupont". Si, précédemment, il existait une instruction :

```
$nom = "Marie";
```

l'ancienne valeur, "Marie", de la variable va être écrasée par la nouvelle valeur, "Dupont".

Vous pouvez supprimer le contenu dans une variable en le remplaçant par une autre valeur, éventuellement d'un autre type, car PHP n'est pas sectaire. Si, par exemple, vous écrivez :

```
$age = "";
```

vous ne faites qu'affecter à la variable `$age` une chaîne de caractères vide. Ce n'est pas l'équivalent de zéro (qui est une valeur). Cela signifie simplement que `$age` ne contient plus rien du tout.

Pour vous débarrasser définitivement d'une variable, il faut utiliser une fonction spécialisée de PHP et écrire, par exemple :

```
unset($age);
```

Une fois cette instruction exécutée, la variable `$age` n'existe plus.

Une variable conserve sa valeur pour tout un programme et pas seulement pour l'une des sections PHP d'un document HTML. Si, au début d'un fichier, une variable prend la valeur "oui", à la fin du fichier, elle aura toujours cette valeur. Supposons que, dans un document HTML, vous ayez les instructions suivantes :

```
<p>Hello World!  
<?php  
    $age = 15;  
    $nom = "Jules";  
?>  
<br>Encore une fois : Hello World!<br>
```

```
<?php
    echo $nom;
?>
```

Voici ce qui va être affiché :

```
Hello World!
Encore une fois : Hello World!
Jules
```

Comment maîtriser les avertissements

Si vous tentez d'exécuter une instruction dans laquelle figure une variable qui n'existe pas, vous obtiendrez ou non un avertissement selon le niveau de signalisation des erreurs fixé dans `php.ini`, ce qui n'empêchera pas votre programme de tourner (voir plus haut l'encadré "Messages d'erreur et avertissements").

Si, par exemple, vous écrivez :

```
unset($age);
echo $age;
$age2 = $age;
```

les deux messages d'avertissement reproduits sur la Figure 6.3 s'affichent.

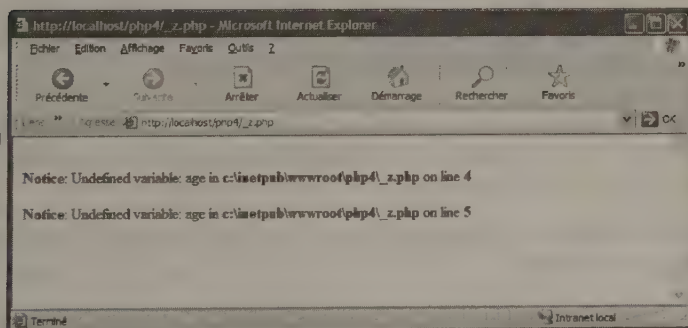


Figure 6.3 : Une variable non définie produit un avertissement.

Supposons maintenant que vous vouliez malgré tout continuer à utiliser ces instructions parce que le programme fait exactement ce que vous souhaitez. Il existe un moyen de vous débarrasser de ces ennuyeux affichages : préfixer le nom de la variable fautive par un arobase (@), ce qui donne :

```
unset($age);
echo @$age;
$age2 = @$age;
```



Si vous avez les droits d'accès d'administrateur PHP, vous pouvez modifier le niveau de signalisation des erreurs dans `php.ini` de la façon indiquée dans l'encadré "Messages d'erreur et avertissements", plus haut dans ce chapitre.

Les constantes PHP

Une *constante* est assez semblable à une variable. Elle possède un nom, mais se contente d'avoir une valeur fixée. Mais il est évident que cette valeur ne peut être modifiée : la valeur d'une constante est... *constante* ! Une fois définie, elle ne peut donc plus être modifiée par le programme. Vous n'avez jamais rêvé d'avoir 20 ans toute votre vie ?

On utilise les constantes lorsque la valeur qu'elles représentent ne risque pas de changer. Si, par exemple, dans une page Web, vous parlez de la date de la bataille de Marignan, vous pouvez coder directement 1515. Mais, comme nous allons le voir, on peut aussi donner un nom à une constante. De cette façon, on identifiera plus facilement ce qu'elle représente. En outre, la définition de cette constante sera généralement placée en tête du programme et on pourra s'y référer sous ce nom symbolique dans tout le programme. En cas de besoin, vous pourrez la retrouver très facilement pour modifier son affectation dans le code source.

Supposons que vous ayez deux constantes pour votre société : son nom et son adresse. Si vous déménagez, il suffira d'ajuster la seconde constante au début du programme. Facile, non ?

Pour définir une constante, on utilise l'instruction `define` sous la forme suivante :

```
define ("nom", "valeur");
```

Pour prendre un exemple simple, on pourrait écrire :

```
define ("MARIGNAN", "1515");
```

Mais si vous souhaitez effectuer des calculs de dates, la variante suivante est parfaitement licite :

```
define ("MARIGNAN", 1515);
```

Si, maintenant, on entre :

```
echo MARIGNAN;
```

on affichera bien 1515.

Pour afficher le contenu d'une constante, ne la placer pas entre des guillemets. Sinon, vous ne verrez que son *nom* (MARIGNAN) et pas sa *valeur* (1515).

L'usage veut qu'on écrive les noms de constantes en lettres capitales (ce qui permet de les identifier plus facilement dans le programme), mais ce n'est pas une obligation.

Une constante peut prendre n'importe quel nom, comme une variable, à cet important détail près que ce nom **ne doit pas** commencer par un caractère dollar (\$).



Les nombres

Avec PHP, vous pouvez *aussi* faire des calculs, grâce aux opérateurs arithmétiques usuels auxquels vient se joindre un petit nouveau : l'opérateur *modulo* qui se note %. Les opérations arithmétiques associent deux valeurs numériques au moyen d'un opérateur, comme dans :

```
1 + 2
```

Ces opérations peuvent être réalisées à partir de variables contenant des valeurs numériques. Par exemple :

```
$n1 = 1;
$n2 = 2;
$somme = $n1 + $n2;
```

Le Tableau 6.1 vous présente les cinq opérateurs reconnus par PHP.

Vous pouvez enchaîner plusieurs opérations arithmétiques. Ainsi, l'instruction suivante effectue trois opérations :

```
$resultat = 1 + 2 * 4 + 3;
```

Les opérations s'effectuent dans un ordre précis qui dépend de leur ordre d'écriture (par défaut : de gauche à droite) et de leurs priorités



Tableau 6.1 : Les cinq opérateurs arithmétiques reconnus par PHP.

Opérateur	Description
+	Addition
-	Soustraction
*	Multiplication
/	Division
%	Modulo : reste de la division entière du premier terme par le second. 13 % 5 a pour valeur 3.

respectives. Multiplication et division ont la même priorité et celle-ci est supérieure à celle de l'addition et de la soustraction, toutes deux de même priorité. Dans notre exemple, voici comment vont être exécutées les opérations :

```
2 * 4           multiplication (8)
1 + résultat partiel précédent addition (9)
3 + résultat partiel précédent addition (12)
```

A la fin de ces opérations, `$resultat` contient donc la valeur 12.

Vous pouvez modifier l'ordre d'exécution des opérations au moyen de parenthèses. Les opérations placées entre parenthèses ont une plus forte priorité que celles qui sont à l'extérieur. Transformons notre précédente instruction en :

```
$resultat = (1 + 2) * 4 + 3;
```

Les opérations vont maintenant s'effectuer dans cet ordre :

```
1 + 2           addition (3)
4 * résultat partiel précédent multiplication (12)
3 + résultat partiel précédent addition (15)
```



En cas de doute, n'hésitez pas ! Mettez des parenthèses : abondance de bien ne nuit pas ! (Mais ça peut alourdir la relecture du programme.)

Lorsque vous manipulez des valeurs réelles, des prix, par exemple, les nombres ont une partie entière et une partie fractionnaire. Pour que l'utilisateur puisse plus facilement comprendre ce que représente un prix, il faut l'afficher avec deux décimales. Sans précaution spéciale, vous risqueriez d'en afficher beaucoup trop. Ce serait le

cas, par exemple, si ce prix résultait d'un calcul. D'un autre côté, si le prix était un nombre dépourvu de partie fractionnaire, il serait affiché sous forme d'un entier. Pour unifier ces affichages, on peut utiliser la *fonction* `sprintf()` qui transforme un nombre en une chaîne de caractères toute prête pour l'affichage. En voici un exemple :

```
$prixHT = 27;
$prixTTC = $prixHT * 1.196;
$affichage = sprintf ("%01.2f", $prixTTC);
echo "$affichage<br>";
```

Vous verrez alors s'afficher :

32.39

Vous pouvez aussi séparer les grands nombres en tranches de trois chiffres. Pour cela, les Anglo-Saxons utilisent le point (.) et nous l'espace. La fonction qui fait ce travail s'appelle `number_format()` et demande quatre arguments. En voici un exemple d'application :

```
$prix = 7123450.67891;
$affichage = number_format($prix, 2, ".", " ");
echo "$affichage<p>";
```

1. Le premier argument représente le nombre à traiter.
2. Le deuxième argument indique le nombre de chiffres à conserver pour la partie fractionnaire. Un arrondi est effectué sur la première décimale perdue.
3. Le troisième argument représente le séparateur à insérer entre la partie entière et la partie fractionnaire.
4. Le quatrième argument représente le séparateur à insérer entre chaque tranche de trois chiffres de la partie entière.

Dans l'exemple précédent, on affichera :

7 123 450,68

Les chaînes de caractères

Une *chaîne de caractères* est constituée d'une suite de caractères : lettres (accentuées ou non), chiffres, ponctuation, caractères spéciaux. Lorsqu'un nombre est conservé sous forme de chaîne de

caractères, il ne peut pas être utilisé pour faire des calculs. Si on a l'habitude de conserver les numéros de téléphone sous forme de chaîne de caractères, c'est précisément parce qu'on ne fait jamais de calcul avec.

Une chaîne de caractères est encadrée par deux caractères particuliers, appairés : guillemets ou apostrophes. Si l'un de ces caractères se trouve déjà dans la chaîne, on choisit l'autre comme délimiteur. Exemples :

```
"Il m'a dit qu'il m'aimait."  
'"Le Misanthrope" est une pièce de Molière.'
```

Lorsque la chaîne comporte une apostrophe **et** des guillemets, on choisit l'un d'eux comme délimiteur et on *l'échappe* avec un antislash à l'intérieur de la chaîne. Exemples :

```
"Il m'a dit : \"Je t'aime.\""  
'Il m'a dit : "Je t'aime"'
```

Une chaîne de caractères peut être affectée à une variable, tout comme un nombre. Exemple :

```
$message = "Caïn, ne dormant pas, songeait au pied des monts.";
echo "$message<br>";
```

affichera :

```
Caïn, ne dormant pas, songeait au pied des monts.
```

Guillemets et apostrophes

Bien qu'en tant que délimiteurs de chaîne de caractères guillemets et apostrophes semblent se comporter de façon identique, dans certains cas, ils sont interprétés différemment. Voici quelques cas à retenir :

- ✓ **Manipulation de variables.** Si vous placez le nom d'une variable entre guillemets, c'est sa valeur qui va être utilisée par PHP. Si vous la placez entre apostrophes, c'est son nom. Les quatre instructions suivantes :

```
$age = 12;  
$result1 = "$age";  
$result2 = 'age';  
echo $result1;
```

```
echo "<br>";
echo $result2;
```

vont afficher ces deux lignes :

```
12
$age
```

- ✓ **Retour à la ligne.** L'association des deux caractères `\n` est appelée *caractère d'alinéa*, parce que c'est un *caractère de contrôle* indiquant à PHP de commencer ce qui suit sur une nouvelle ligne. Si une chaîne de caractères délimitée par une paire de guillemets comporte ce doublet, PHP l'affichera sur deux lignes. Si elle est délimitée par un couple d'apostrophes, une seule ligne sera affichée, car le doublet `\n` ne sera pas interprété comme un caractère d'alinéa mais conservé tel quel. Considérez, par exemple, les deux chaînes suivantes :

```
$chaîne1 = "Chaîne entre \nguillemets";
$chaîne2 = 'Chaîne entre \napostrophes';
```

Pour PHP, la première ligne se présentera ainsi :

```
Chaîne entre
guillemets
```

et la seconde ainsi :

```
Chaîne entre \napostrophes
```

- ✓ **Insertion d'une tabulation.** Le caractère spécial `\t` dit à PHP d'insérer une tabulation. Quand vous utilisez des guillemets, PHP insère réellement une tabulation à cet endroit, alors qu'avec des apostrophes il considérera `\n` comme un littéral de type chaîne de caractères. Par exemple, avec les instructions suivantes :

```
$chaîne1 = "Chaîne entre \tguillemets";
$chaîne2 = 'Chaîne entre \tapostrophes';
```

`$chaîne1` est affichée sous la forme :

```
Chaîne entre      guillemets
```

et `^chaîne2` sous la forme :

```
Chaîne entre \tapostrophes
```

La délimitation des chaînes détermine également le traitement des variables et des caractères spéciaux, y compris si un autre jeu de délimiteurs est présent. Regardons par exemple les instructions suivantes :

```
$nombre = 10;
$chaîne1 = "Il y a '$nombre ' personnes en ligne.";
$chaîne2 = "Il y a \"$nombre \" personnes en attente.";
echo $chaîne1;
echo "<br>";
echo $chaîne2;
```

Vous allez obtenir l'affichage suivant :

```
Il y a '10' personnes en ligne.
Il y a "$nombre" personnes en attente.
```

Concaténation de chaînes de caractères

On peut former une chaîne à partir d'autres chaînes en les plaçant bout à bout, les unes à la suite des autres. Il faut spécifier ce "chaînage" au moyen de l'opérateur de concaténation qui est le point (.). Exemple :

```
$s1 = "Hello";
$s2 = "World!";
$s = $s1.$s2;
```

affichera :

```
HelloWorld!
```

Aucune des deux chaînes ne comporte d'espace, ni à son début ni à sa fin. Elles seront donc collées l'une derrière l'autre. Il est facile d'insérer un espace entre les deux mots en écrivant :

```
$s = $s1." ".$s2;
```

On réalise ainsi la concaténation de trois chaînes : deux sont représentées par une variable et celle du milieu par une constante.

Vous pouvez utiliser les caractères .= pour compléter une chaîne existante. La séquence précédente pourrait donc être remplacée par celle-ci :


```
$s1 = "Hello";
$s1 .= " World!";
echo $s1;
```

Bien d'autres manipulations de chaînes sont possibles : les partager, chercher une sous-chaîne, etc. Nous en étudierons quelques-unes au Chapitre 7.

Dates et heures

Les dates et les heures constituent un élément important dans une application de base de données sur le Web. Comme nous l'avons vu à propos de MySQL, ces entités forment une classe de variables à part. Un groupe date/heure est désigné par PHP au moyen d'un terme emprunté au vocabulaire UNIX : *timestamp*. Sous cette forme, nous ne pouvons pas l'interpréter, mais PHP sait le reconnaître et le convertir sous diverses formes, et le rendre ainsi aisément compréhensible.



Un *timestamp* représente le nombre de secondes écoulées depuis le 1^{er} janvier 1970 à 0 h GMT, an zéro de l'ère UNIX. Pas pratique pour représenter la date de la bataille de Marignan ou celle de la mort de Louis XVI ! En revanche, pour manipuler des temps ou des différences de temps ne faisant pas intervenir des événements de plus d'une trentaine d'années, c'est tout à fait utilisable.

Définition de l'heure locale (*local time*)

Dans sa version 5.1, PHP ajoute dans `php.ini` un paramètre pour le fuseau horaire local. Si vous ne définissez aucun fuseau horaire (time zone) par défaut, PHP va essayer de le deviner, en optant au pire pour l'heure GMT. PHP affiche d'ailleurs un message pour vous rappeler de définir votre fuseau horaire.

Voici comment définir un fuseau horaire local :

1. Ouvrez le fichier `php.ini` dans votre éditeur de texte.
2. Descendez jusqu'à la section intitulée [Date].
3. Trouvez le paramètre `date.timezone =`.
4. Si la ligne commence par un signe point-virgule (;), supprimez-le.

5. Indiquez le nom réservé d'un fuseau horaire après le signe égal.

La liste des codes de fuseaux (*time zone codes*) se trouve dans l'Annexe H du manuel de référence en ligne de PHP (www.php.net/manual/en/timezones.php). Voici comment régler l'heure locale pour la France métropolitaine :

```
date.timezone = ('Europe/Paris');
```

Si vous n'êtes pas en mesure de modifier le fichier `php.ini`, vous pouvez définir le fuseau dans chaque programme source comme ceci :

```
date_default_timezone_set("timezonecode");
```

Pour savoir quel est le fuseau en vigueur, écrivez ceci :

```
$fuseau = date_default_timezone_get()
echo $fuseau;
```

Mise en forme d'une date

La fonction la plus utilisée pour mettre en forme une date est... `date()`. Elle convertit un timestamp selon les directives que vous lui communiquez. Sa forme générale est :

```
$unedate = date ("format", $timestamp);
```

"format" est une chaîne de caractères spécifiant quelles sont les conversions à effectuer et `$timestamp` une date sous forme de timestamp. Si ce second argument est omis, c'est la date de l'instant où cette instruction est exécutée qui sera prise par défaut (plus exactement, celle de l'horloge de l'ordinateur). Avant d'être à même de donner un exemple, nous devons voir quels sont les principaux caractères de conversion qui peuvent intervenir dans le format. Le Tableau 6.2 vous en donne un aperçu. Pour en avoir la liste complète, consultez la documentation officielle de PHP à l'URL <http://www.php.net>.

Tous les autres caractères seront conservés tels quels, sans être interprétés. Pour que certains caractères du format spécifié ne soient pas interprétés comme un format de date, il suffit de les "échapper" avec un antislash, ce qui donnera quelque chose de cette forme :

Tableau 6.2 : Symboles des formats de date.

Symbole	Signification
a	<i>am</i> ou <i>pm</i>
A	<i>AM</i> ou <i>PM</i>
d	Jour du mois sur deux chiffres (01 à 31)
D	Jour de la semaine sur trois lettres (abréviations anglaises)
F	Mois en toutes lettres (noms de mois anglais)
g	Heure au format 12 heures sans zéro en tête
G	Heure au format 24 heures sans zéro en tête
h	Heure au format 12 heures avec zéro en tête
H	Heure au format 24 heures avec zéro en tête
i	Minutes (00 à 59)
j	Jour du mois (1 à 31)
m	Numéro du mois (01 à 12)
M	Nom du mois sur trois lettres (noms de mois anglais)
n	Numéro du mois (1 à 12)
s	Secondes (00 à 59)
y	Année sur deux chiffres (70 à 99)
Y	Année sur quatre chiffres (1970 à 9999)
w	Jour de la semaine (0 à 6 en partant du dimanche)

```
$auj = date("j m Y \e\\t \\i\\l \\e\\s\\t H \\h i \\m\\n.");
echo "Nous sommes le $auj<br>;
```

On affichera ainsi, par exemple :

Nous sommes le 24 02 2004 et il est 12 h 04 mn.

Et pour prendre une formule beaucoup plus simple, l'instruction :

```
echo date("d/m/Y");
```

renverra la date du jour selon les spécifications françaises, par exemple :

24/02/2004

Rangement d'un timestamp dans une variable

Rien n'empêche de ranger la valeur d'un timestamp dans une variable comme le fait l'instruction suivante :

```
$aujourd'hui = time();
```

Une autre façon de spécifier un timestamp fait appel à la fonction `strtotime`. Celle-ci peut prendre comme paramètres divers mots clés et abréviations anglo-saxonnes. Pour mémoriser par exemple la date du 24 février 2004, vous pourriez par exemple écrire :

```
$DateImportante = strtotime("February 24 2004");
```

La fonction `strtotime` reconnaît les formats suivants :

- ✓ **Noms des mois** : Il peut s'agir des noms entiers ou abrégés (en anglais).
- ✓ **Jours de la semaine** : Les sept jours (toujours en anglais) et certaines abréviations.
- ✓ **Unités de temps** : Un certain nombre de termes classiques dans ce domaine : year, month, fortnight, week, date, hour, minute, second, am, pm.
- ✓ **Certains mots utiles** : ago, now, last next, this, tomorrow, yesterday.
- ✓ **Les signes plus et moins.**
- ✓ **Tous les nombres.**
- ✓ **Des fuseaux horaires.** Exemple type : gmt (Greenwich Mean Time).

Voilà de quoi réviser son anglais ! Ces mots et abréviations peuvent être combinés de différentes manières. Toutes les instructions qui suivent sont licites :

```
$DateImportante = strtotime("tomorrow");           #dans 24 heures
$DateImportante = strtotime("now + 24 hours");      #idem
$DateImportante = strtotime("last saturday");      #samedi dernier
$DateImportante = strtotime("8pm + 3 days");       #A 8h du matin
                                                    #dans 3 jours
$DateImportante = strtotime("2 weeks ago");        #il y a 2 semaines
```

```
$DateImportante = strtotime("next year gmt"); #dans 1 an exactement
$DateImportante = strtotime("this 4am");      #à 4h de l'après midi
```

Toutes ces dates peuvent ensuite être affichées ou manipulées à l'aide d'opérateurs arithmétiques. Si vous voulez savoir combien de secondes se sont écoulées depuis \$bonnedate jusqu'à aujourd'hui, écrivez :

```
$nb_secondes = $bonnedate - $aujourd'hui;
```

Si vous préférez avoir cette valeur en heures et fractions d'heure, ajoutez une division par 3600 (nombre de secondes dans une heure) :

```
$nb_heures = ($bonnedate - $aujourd'hui)/3600; •
```



Attention ! La date est représentée "à l'américaine", les mois précédant les jours.

Les dates et MySQL

Vous aurez souvent besoin de ranger une date dans votre base de données MySQL. Ce sera, par exemple, celle où un client aura passé une commande ou bien le moment où un membre d'une association s'est enregistré. Comme nous l'avons vu dans la deuxième partie (Chapitre 4), MySQL, de même que PHP, considère les dates comme étant des entités particulières. Cependant, ils les manipulent différemment. Vous devez donc savoir comment sont manipulées les dates et heures dans PHP (nous venons de le voir) et dans MySQL.

Nous avons présenté les types de données DATE et DATETIME de MySQL dans le Chapitre 3. Voyons rapidement comment se manifestent ces différences.

- ✓ **DATE** : Sous MySQL, l'ordre des éléments d'une date est le suivant : année, mois, jour. L'année peut être exprimée par `aaaa` ou `aa`, le mois par `mm` ou `m` et le jour par `jj` ou `j`. Chaque partie peut être séparée de la suivante par un tiret (-), un slash (/), un point (.) ou un espace.
- ✓ **DATETIME** : Les champs ainsi définis dans MySQL doivent contenir la date et l'heure. La date est formatée comme dans le paragraphe ci-dessus et suivie par l'heure sous la forme `hh:mm:ss`.

Pour pouvoir être placés dans une base de données MySQL, ces éléments doivent avoir le format correct. Pour effectuer la transformation nécessaire, on peut faire usage des fonctions PHP. En voici un premier exemple :

```
$jourdui = date("Y-m-d");
```

La date est bien ici dans l'ordre année, mois, jour. Autre exemple :

```
$DateImportante = date("Y-m-d", strtotime("March 04 2004"));
```

A la suite de la date, on ajoute l'heure. Il ne reste plus qu'à ranger cette valeur dans la base de données au moyen d'une requête de ce genre :

```
UPDATE Membre SET date="$jourdui"
```

Il est parfois plus efficace d'effectuer des traitements sur les dates et heures avec les fonctions MySQL et non celles de PHP. Ainsi, MySQL propose la fonction DATEDIFF pour calculer le nombre de jours entre deux dates :

```
DATEDIFF(date1,date2)
```

La fonction renvoie un nombre de jours. Voici comment savoir le nombre de jours entre une date lue dans une table et la date courante :

```
SELECT DATEDIFF(NOW(),dnaissance) FROM Clients
```

NOW() est une fonction MySQL qui renvoie la date courante et dnaissance est le nom d'une colonne de données de la table Clients table.

La même fonction permet de récupérer le nombre de jours écoulés entre deux dates spécifiées :

```
SELECT DATEDIFF('2008-1-15','1997-06-23')
```

MySQL comporte de nombreuses fonctions très utiles au niveau des dates et des heures. Voyez cette page du manuel de référence de MySQL :

<http://dev.mysql.com/doc/refman/5.0/fr/date-and-time-functions.html>

Comparaison de valeurs

Dans un programme, on trouve de nombreuses *instructions conditionnelles* qui permettent de choisir, à la suite d'une comparaison, entre deux instructions ou entre deux blocs d'instructions. Selon que la comparaison donne comme résultat VRAI ou FAUX, on exécutera l'un ou l'autre des blocs. Voici deux exemples d'instructions conditionnelles :

```
si l'utilisateur est un enfant
    afficher le catalogue des jouets
si l'utilisateur n'est pas un enfant
    afficher le catalogue des produits électroniques
```

Pour choisir la voie à suivre, le programme doit (se) poser des questions en fonction des éléments dont il dispose ou qui lui ont été fournis. Voici quelques questions possibles :

- ✓ L'utilisateur est-il un enfant ? Si oui, afficher le catalogue des jouets.
- ✓ Quels sont les produits les plus vendus ? Afficher les résultats, le plus vendu en tête.
- ✓ Est-ce que l'utilisateur a donné un bon mot de passe ? Si oui, afficher la page dont l'accès est réservé aux membres de l'association.
- ✓ Est-ce que le client réside dans le Doubs ? Si c'est le cas, affichez la liste des vendeurs de ce département.

Pour poser une question dans un programme, vous devez écrire une instruction qui compare deux valeurs. Le programme teste le résultat pour voir s'il est vrai ou faux. En reprenant la première question, on pourrait la détailler ainsi :

- ✓ Le client a moins de 13 ans. Vrai ou faux ? Si c'est vrai, afficher le catalogue des jouets.
- ✓ Les ventes du produit 1 sont supérieures à celles du produit 2. Vrai ou faux ? Si c'est vrai, afficher le produit 1 avant le produit 2. Si c'est faux, afficher le produit 2 avant le produit 1.
- ✓ Le client a donné comme mot de passe : *secret*. Vrai ou faux ? Si c'est vrai, afficher la page réservée aux membres de l'association.

- ✓ Le client réside dans le Doubs. Vrai ou faux ? Si c'est vrai, afficher la liste des vendeurs situés dans ce département.

Une comparaison peut être très simple, comme vous venez de le voir. Plus grand ? Plus petit ? Egal ? Rien de sorcier là-dedans. Mais elle peut aussi être plus élaborée. Par exemple, on peut avoir à rechercher dans une chaîne de caractères si certain(s) caractère(s) s'y trouve(nt) plutôt que de regarder une valeur globale. Est-ce que le nom du client commence par un "S" ? Est-ce que le préfixe téléphonique est celui de la région parisienne ?

Comparaisons simples

Une comparaison simple évalue une valeur par rapport à une autre. PHP propose plusieurs moyens pour effectuer ce genre de comparaisons. Le Tableau 6.3 vous en présente le catalogue.

Tableau 6.3 : Comparaison de deux valeurs.

Comparaison	Signification
==	Les deux valeurs sont-elles égales ?
>	La première valeur est-elle supérieure à la seconde ?
>=	La première valeur est-elle supérieure ou égale à la seconde ?
<	La première valeur est-elle inférieure à la seconde ?
<=	La première valeur est-elle inférieure ou égale à la seconde ?
!=	Les deux valeurs sont-elles différentes ?
<>	Les deux valeurs sont-elles différentes ?

On peut comparer des nombres ou des chaînes de caractères. Ces dernières sont comparées alphabétiquement, les majuscules étant supérieures aux minuscules : la chaîne "SUzanne" est supérieure à la chaîne "Suzanne". Les caractères de ponctuation ont, eux aussi, une place dans l'ordre "alphabétique". Cependant, il faut bien reconnaître que comparer un point à une virgule n'a guère d'intérêt.



Les chaînes de caractères sont comparées selon leur code ASCII. Dans cette codification, tous les caractères standard ont une valeur numérique comprise entre 0 et 127. Les codes allant de 128 à 255 représentent des caractères spéciaux, dont notamment les lettres

accentuées. Lors de la comparaison de deux chaînes de caractères, ce sont ces valeurs qui déterminent le résultat de la comparaison. Par exemple, le code qui représente la virgule a pour valeur décimale 44, alors que celui qui représente le point vaut 46. On pourrait en déduire (si cela avait un sens) que le point est supérieur à la virgule !

En règle générale, les comparaisons servent à exécuter des instructions (ou des blocs d'instructions) sous certaines conditions. Voyons un petit exemple (sachant qu'en anglais, *si* se dit *if*) :

```
if ( $meteo == "pluie" )
{
    prendre le parapluie;
    annuler le pique-nique;
}
```

PHP compare la valeur contenue dans la variable `$meteo` à la valeur *pluie*. Il pleut ? Il va exécuter les deux instructions du bloc qui suit. Le soleil brille ? Ces deux lignes ne seront pas traitées.



Attention au comparateur d'égalité ! Il s'écrit avec deux signes "égal" rigoureusement consécutifs (`==`), et il ne faut surtout pas le confondre avec l'opérateur d'affectation qui, lui, ne s'écrit qu'avec un seul (`=`). Si vous écrivez `if ( $meteo = "pluie" )`, vous donnez à `$meteo` la valeur "pluie" au lieu de la comparer à cette valeur.

Revenons à l'exemple donné en tête de cette section. Il se présente ainsi :

```
si l'utilisateur est un enfant
    afficher le catalogue des jouets
si l'utilisateur n'est pas un enfant
    afficher le catalogue des produits électroniques
```

Pour déterminer si l'utilisateur est un adulte, vous devez choisir l'âge à compter duquel un enfant est supposé ne plus s'intéresser aux jouets et préférer les catalogues de produits électroniques grand public. Prenons 13 ans comme âge de référence. Dans ce cas, notre comparaison peut se formaliser ainsi :

```
$age < 13      (est-ce que la personne a moins de 13 ans ?)
$age >= 13     (est-ce que la personne a au moins 13 ans ?)
```

Une façon de programmer ces conditions consiste à utiliser les instructions ci-dessous :

```

if ($age < 13)
    $etat = "enfant";
if ($age >= 13)
    $etat = "adulte";

```

PHP compare alors le contenu de la variable `$age` à la constante `13` au moyen du comparateur "`<`". Si la réponse est VRAI, la proposition est vérifiée, l'utilisateur est un enfant et on enregistre la valeur *enfant* dans la variable `$etat`. La seconde condition réalise un travail semblable avec l'opérateur de comparaison "`>=`". Si la réponse est VRAI, l'utilisateur est un adulte, ce que l'on enregistre également dans la variable `$etat`. Il ne reste plus qu'à repartir de cette variable pour prendre la bonne décision. Certes, ces instructions pourraient être écrites plus efficacement, mais cela marche. Nous reviendrons plus en détail sur cette question dans le Chapitre 7.

Chaînes de caractères et motifs de recherche

Il peut arriver que vous ayez besoin de rechercher dans une chaîne de caractères la présence de certaines caractéristiques et non de la comparer globalement à une autre. Par exemple, vous pouvez vouloir identifier toutes les chaînes commençant par un "S" ou toutes celles contenant des chiffres. Pour cela, on explore la chaîne avec un *motif de recherche* appelé *expression rationnelle* ou encore *profil de recherche* (en anglais : *regular expression*).

Les *caractères de substitution* (ou *jokers*) comme `*` ou `?` qu'on rencontre dans certains systèmes d'exploitation (MS-DOS ou UNIX, par exemple) constituent une forme élémentaire de motif de recherche. Par exemple, `*.txt` représente tous les fichiers, quel que soit leur nom, dont l'extension est `.txt`. Plus restrictif, `c*.txt` représente tous les fichiers dont l'extension est `.txt` et dont le nom commence par un "c", comme `charles.txt` ou `copie.txt`. Les véritables expressions rationnelles permettent de raffiner considérablement cette approximation.

Dans une page Web, l'usage le plus courant d'une expression rationnelle est la vérification des saisies effectuées par un utilisateur à l'aide d'un formulaire. Cela évite de ranger dans une base de données des informations dépourvues de sens. Dans le cas d'un nom propre, par exemple, vous pourrez ainsi déceler des anomalies telles que la présence de chiffres ou de caractères de ponctuation inhabituels. Un nom consiste, en effet, en une suite de lettres parmi lesquelles on peut trouver (sauf aux extrémités) un tiret, une apostrophe ou un

espace comme c'est le cas dans "Goujon-Duval", "d'Hauteville" ou "De La Rue". Si une recherche par expression rationnelle décèle la présence de caractères autres que a à z, A à Z, lettres accentuées, -, ' ou espace, il y a sûrement une erreur.

Un motif de recherche se compose de caractères *littéraux* et de caractères *spéciaux*. Les premiers sont des caractères ordinaires n'ayant pas de signification particulière. Les caractères spéciaux sont des caractères ordinaires qui, dans une expression rationnelle, prennent une signification particulière. Le Tableau 6.4 vous présente un échantillon des principaux caractères spéciaux utilisés dans les motifs de recherche.

Tableau 6.4 : Caractères spéciaux les plus fréquemment utilisés dans un motif de recherche.

Caractère	Signification	Exemple	Correspond à	Ne correspond pas à
^	Début d'une ligne	^c	chat	mon chat
\$	Fin d'une ligne	t\$	chat	chatte
.	N'importe quel caractère	f.i.re	faire, foire	foirer, défaire
?	Le précédent caractère est facultatif	lapine?s	lapins	lapons
()	Groupement de littéraux dans une chaîne qui doit correspondre exactement	lap(ine)s	lapines	lapins
[]	Définit un ensemble de caractères facultatifs	lapin[es]	lapin, lapine, lapines	lapons
-	Tous les caractères entre deux caractères	lap[i-u]n	lapin, lapon	lapen
+	Un ou plusieurs exemplaires des caractères précédents	beta[1-3]+	beta2, beta22, beta132	beta, beta45
*	Aucun ou plusieurs exemplaires des caractères précédents	beta[1-3]*	beta2, beta, beta132	betax, beta45
&{ , }	Nombre de répétitions possibles	[0-9]{2,5}	123, 145	1, xx3
\	Le caractère suivant est un littéral	m*n	m*n	mon, man
()	Plusieurs groupes possibles	(Marcel Michel)	Marcel, Michel	Manuel

Littéraux et caractères spéciaux sont associés pour créer un motif de recherche dont la complexité peut parfois être déroutante.

La chaîne de caractères examinée est alors explorée selon les spécifications de ce motif de recherche. Si la recherche est couronnée de succès, le résultat est VRAI. On peut dès lors choisir entre deux chemins dans la suite du programme. Voici quelques exemples simples de motifs de recherche :

✓ **[^A-Z].* — Chaîne de caractères commençant par une majuscule**

- **[^A-Z]** — La chaîne doit commencer par une majuscule
- **.*** — Elle doit continuer par un nombre quelconque de caractères

Correspondance :

- Une fois encore
- A

Pas de correspondance :

- une fois encore
- a

✓ **Chère (Julie|Sandra) — Choix entre deux chaînes de caractères**

- **Chère** — Caractères littéraux
- **(Julie|Sandra)** — "Julie" ou "Sandra"

Correspondance :

- Chère Julie
- Chère Sandra

Pas de correspondance :

- Chère Madeleine
- Sandra

✓ **^[0-9]{5,5}(\-[0-9]{4,4})?\$ — N'importe quel code zip (code postal américain)**

- **^[0-9]{5,5}** — N'importe quelle chaîne de 5 chiffres

- \- — un littéral
- [0-9]{4,4} — Une chaîne composée de 4 chiffres
- ()? — Regroupe les deux dernières parties du motif et le rend optionnel

Correspondance :

- 90001
- 90002-4232

Pas de correspondance :

- 9001
- 12-4321

✓ **^.+@.+\.com\$ — Une chaîne de caractères se terminant par ".com" et contenant un arobase (@)**

- ^.+ — N'importe quelle suite de un ou plusieurs caractères
- @ — Un arobase (@) littéral
- .+ — N'importe quelle chaîne de un ou plusieurs caractères
- \. — Un point littéral (.)
- com\$ — La chaîne ".com" à la fin

Correspondance :

- marie@monserveur.com

Pas de correspondance :

- marie@monserveur.net
- marie@com
- @marie.com

Pour faire une recherche dans une chaîne de caractères au moyen d'un motif de recherche, vous pouvez utiliser la fonction `ereg()` dont la forme générale est :

```
ereg("motif", chaîne à explorer);
```

Par exemple, pour vérifier le nom saisi par un utilisateur dans la boîte de saisie d'un formulaire, vous pouvez écrire :

```
ereg("^ [A-Za-z' -]+$", $nom)
```

- ✓ La présence des caractères ^ et \$ délimite le début et la fin de la chaîne. Cela signifie que tous les caractères de celle-ci doivent vérifier le motif.
- ✓ Les crochets délimitent les caractères autorisés dans la chaîne. Tous les autres sont refusés. Il s'agit donc ici des lettres minuscules et majuscules, de l'apostrophe, de l'espace et du tiret. Remarquez que ce dernier sert aussi à définir une plage, comme dans A-Z. Pour éviter toute confusion, le tiret ne doit pas être placé entre deux autres caractères. En l'insérant tout à la fin, il sera normalement intégré au motif.
- ✓ Enfin, le signe + indique que la chaîne doit contenir au moins un caractère pour pouvoir vérifier le motif. Une chaîne vide sera donc automatiquement rejetée.



Insistons un peu. Lorsque vous utilisez un tiret littéral (-) dans un groupe placé entre crochets, ce tiret doit être le premier ou le dernier caractère du groupe. Faute de quoi, s'il se trouve entre deux caractères, il sera interprété comme un caractère spécial spécifiant la suite de caractères se trouvant entre ces deux caractères.

Association de comparaisons

Souvent, une simple comparaison n'est pas suffisante pour vérifier une condition et il faut en associer plusieurs. Supposez, par exemple, que votre entreprise propose plusieurs catalogues pour différents produits et en différentes langues. Vous devez savoir quel produit intéresse l'utilisateur et dans quelle langue il veut qu'il soit affiché. Voici comment se présente une association de comparaisons :

```
comparaison1 and|or|xor comparaison2
               and|or|xor comparaison2
               and|or|xor ...
```

avec :

- ✓ **and** : Les deux comparaisons doivent répondre VRAI.

- ✓ **or** : L'une ou l'autre des deux comparaisons doit répondre VRAI.
- ✓ **xor** : Une seule des deux comparaisons doit répondre VRAI (et donc l'autre FAUX).

La liste suivante vous montre quelques exemples d'associations de comparaisons :

- ✓ `$client == "Dupont" or $client == "Dupond"` — Le nom du client doit être Dupont ou Dupond.
- ✓ `$client == "Dupont" and $ville == "Blois"` — Le nom du client doit être Dupont et il doit habiter Blois.
- ✓ `$client == "Dupont" or $ville == "Blois"` — Le nom du client doit être Dupont ou bien il doit habiter Blois.
- ✓ `$client == "Dupont" xor $ville == "Blois"` — Le nom du client doit être Dupont ou bien il doit habiter Blois, mais pas les deux à la fois.
- ✓ `$client == "Dupont" and $age < 13` — Le nom du client doit être Dupont et il doit avoir moins de 13 ans.

Vous pouvez ainsi associer autant de comparaisons que vous le souhaitez. Le groupe contenant `and` est testé en premier. Vient ensuite celui contenant `xor`, puis celui contenant `or`. Dans l'exemple qui suit, il y a trois comparaisons associées :

`$age == 20 or $age == 30 and $nom == "Jules"`

Si le nom du client est Jules et qu'il est âgé de 30 ans, le résultat est VRAI. Le résultat est également VRAI si, quel que soit son nom, le client a 20 ans. Le résultat est FAUX si le client a 30 ans et qu'il ne s'appelle pas Jules. Ces résultats découlent de l'ordre d'exploration que nous venons d'énoncer :

- ✓ **Association par `and`**. Le programme teste `$age` pour voir s'il contient la valeur 30, puis il teste `$nom` pour voir s'il est égal à "Jules". Si les deux conditions sont remplies, il est inutile de comparer `$age` à 20, puisqu'il suffit qu'une des deux conditions associées par un `or` soit vraie pour que la proposition soit vraie.
- ✓ **Association par `or`**. Lorsque la condition `and` a répondu FAUX, le programme teste `$age` pour voir s'il contient la valeur 20.

Si c'est vrai, le résultat vaut VRAI. Dans le cas contraire, le résultat est FAUX.

Vous pouvez modifier l'ordre de l'évaluation par l'usage de parenthèses qui forcent l'évaluation prioritaire des comparaisons ainsi encadrées. Dans l'exemple précédent, si on écrit :

```
($age == 20 or $age == 30) and $nom == "Jules"
```

C'est la condition **or** qui va être testée en premier et la condition **and** en dernier. Voici ce que cela va donner :

- ✓ **Association par or.** Le programme compare la valeur de `$age` à 20 puis à 30. Si la réponse à l'une des questions ou aux deux est VRAI, le résultat est VRAI. Si le résultat vaut VRAI, il est nécessaire de faire le test suivant. Si ce n'est pas le cas, il n'est pas nécessaire de tester l'association **par and**.
- ✓ **Association par and.** Le programme teste `$age` pour voir s'il contient la valeur 20. Si c'est vrai, le résultat est VRAI.



N'hésitez pas à entourer vos comparaisons de parenthèses, même si vous pensez être certain de l'ordre dans lequel elles vont s'effectuer. Abondance de biens ne nuit pas.



Si vous connaissez déjà d'autres langages de programmation comme le C, par exemple, vous avez peut-être déjà utilisé `||` pour représenter **or** et `&&` pour représenter **and**. Ces deux associations de caractères sont aussi utilisables en PHP. Les deux expressions suivantes donneront le même résultat :

```
$a < $b and $c  
$a < $b && $c
```

Mettez des commentaires dans vos programmes

Les *commentaires* sont des notes incluses à l'intérieur d'un programme pour documenter telle ou telle section ou même telle ou telle instruction. Ils facilitent grandement la relecture et la maintenance des programmes, surtout si cette dernière doit être effectuée par quelqu'un qui n'est pas celui ou celle qui a écrit le programme.

Utilisez avec libéralité des commentaires. PHP les ignore car ils n'intéressent que les humains. Mais, bien entendu, vous devez dire à PHP que ce sont des commentaires. Pour cela, il y a deux façons de procéder selon l'endroit où vous allez les placer et le nombre de lignes qu'ils contiennent.

Commentaires longs

Au début du commentaire, vous devez placer le couple `/*` et à la fin le couple `*/` (dans les deux cas, sans espace entre les deux caractères). Tout ce qui se trouve entre ces deux délimiteurs sera considéré comme du commentaire par PHP. Exemples :

```
/* Le calcul suivant
   détermine le montant
   des frais de port */
```

```
$prixTTC = $prixHT * 1.196; /* Calcul du prix TTC */
```

En tête d'un programme, il est bon de placer un assez long commentaire indiquant le nom et l'objet du programme, le nom de son auteur, la date de son écriture et les dates de ses modifications ultérieures, comme dans l'exemple ci-dessous :

```
/* nom :          catalogue.php
   description :   Ce programme affiche les descriptions des produits.
                   Celles-ci sont conservées dans une base de données.
                   La description à afficher est choisie par le client
                   dans la base de données d'après la catégorie que
                   l'utilisateur a saisie dans un formulaire.

   écrit par :    Jules Martin
   créé le :      23 mars 2004
   modifié le :   15 avril 2004
                   4 juin 2004
*/
```

Ajoutez des commentaires partout où il est nécessaire de préciser ce que fait le programme. C'est un conseil gratuit et particulièrement important lorsque les séquences d'instructions sont compliquées. Par exemple :

```
/* Retrouve l'information dans la base de données */
/* Vérifie que le client a plus de 18 ans */
/* et ajoute les frais de port au montant de la commande */
```

Commentaires courts

Lorsque le commentaire a moins d'une ligne, par exemple à la suite d'une instruction, vous pouvez utiliser une forme particulière qui ne demande qu'un délimiteur de début (`//` ou `#` en l'occurrence). Dans ce cas, la remarque se termine en même temps que la ligne sur laquelle elle figure. Exemple :

```
// Nous allons calculer le prix TTC
$prixTTC = $prixHT * 1.196; // Calcul du prix TTC
```

Pour mettre en valeur un commentaire (comme le titre donné à un bloc d'instructions ou une note particulièrement importante), vous pouvez l'encadrer en totalité, comme ceci :

```
#####
##  CETTE SECTION EST A VERIFIER DE PRES ##
#####
```

Cela ne vous empêche pas d'écrire de longs commentaires, à condition, bien sûr, que chacune des lignes qu'ils contiennent soit précédée du délimiteur spécial. Mais il faut reconnaître que c'est fastidieux et moins facile à relire. Voici comment pourrait se présenter alors l'exemple de description de programme que nous avons donné dans la section précédente :

```
// nom :          catalogue.php
// description : Ce programme affiche les descriptions des produits.
//              Celles-ci sont conservées dans une base de données.
//              La description à afficher est choisie par le client
//              dans la base de données d'après la catégorie que
//              l'utilisateur a saisie dans un formulaire.
// écrit par :    Jules Martin
// créé le :      23 mars 2004
// modifié le :   15 avril 2004
//              4 juin 2004
```

Bien entendu, les commentaires sont éliminés par PHP de ce qu'il envoie au navigateur de l'utilisateur. Ce dernier ne les voit donc jamais.

Chapitre 7

Briques de base pour l'écriture de programmes en PHP

Dans ce chapitre :

- Envoi de sorties vers une page Web.
- Affectation de valeurs aux variables.
- Incrémentation/décrémentation de variables.
- Arrêt et interruption d'un programme.
- Création et usage de tableaux.
- Emploi des instructions conditionnelles.
- Réalisation de boucles pour les traitements répétitifs.
- Les fonctions.

Dans ce chapitre, vous allez apprendre à créer les briques de base des programmes PHP. Nous verrons les instructions et les structures les plus utilisées et la façon de les construire. Ensuite, au Chapitre 8, nous apprendrons à les employer pour manipuler les informations contenues dans les bases de données.

Instructions simples mais utiles

Une instruction se termine toujours par un point-virgule. (Je ne le dirai jamais assez tant cette ponctuation est souvent oubliée !) Voici quelques-unes des instructions simples mais fréquemment utilisées qu'on rencontre dans un programme PHP :

- ✓ **Instruction** `echo` : Elle génère une sortie à destination du navigateur.
- ✓ **Instruction d'affectation** : Elle affecte une valeur à une variable.
- ✓ **Instruction d'incrémentation/décrémentation** : Elle augmente ou diminue le contenu d'une variable d'une valeur entière.
- ✓ **Instruction** `exit` : Elle met fin à l'exécution du programme.
- ✓ **Appel de fonction** : Elle permet de réutiliser des blocs d'instructions écrits spécialement pour effectuer une certaine tâche.

Nous allons voir de plus près ces instructions dans les sections qui suivent.

echo

Cette instruction génère une sortie à destination du navigateur. Ce dernier l'interprète comme s'il s'agissait de HTML.

Sa forme générale est :

`echo élément,élément,élément... ;`

où *élément* est une constante (numérique ou chaîne de caractères) ou une variable. Leur liste n'est pas limitée en nombre, chacune étant séparée de la suivante par une virgule. Toute chaîne de caractères, quelle qu'elle soit (un espace, par exemple), doit être enclose entre guillemets ou entre apostrophes.



Un nom de variable encadré par des guillemets affiche le contenu de la variable, alors qu'encadré par des apostrophes il affiche le *nom* de la variable. Si `$string` contient "Hello" :

- ✓ `echo "$string";` Affichera : Hello
- ✓ `echo '$string';` Affichera : \$string

En supposant que `$string1` contienne "Hello" et `$string2` "World!", voici quelques exemples d'utilisation de cette instruction :

- ✓ **echo Hello;** Produit le diagnostic (avertissement) : "Use of undefined constant Hello in ...". Affiche : Hello.
- ✓ **echo Hello,World!;** Produit le diagnostic (erreur) : "**Parse error:** parse error, expecting `,' or `;' in ...". Rien n'est affiché, l'exécution du programme n'ayant pas lieu.
- ✓ **echo Hello World!;** Produit le diagnostic (erreur) : "**Parse error:** parse error, expecting `,' or `;' in ...". Rien n'est affiché, l'exécution du programme n'ayant pas lieu.
- ✓ **echo 'Hello World!';** Affiche : Hello World!
- ✓ **echo \$string1;** Affiche : Hello.
- ✓ **echo \$string1,\$string2;** Affiche : HelloWorld!
- ✓ **echo "\$string1 \$string2";** Affiche : Hello World!
- ✓ **echo "Hello",\$string2;** Affiche : HelloWorld!
- ✓ **echo "Hello", "\$string2";** Affiche : HelloWorld!
- ✓ **echo '\$string1','\$string2';** Affiche : HelloWorld!

Un nom de variable peut être composé à partir du nom d'une variable qui existe et d'un suffixe formé d'une simple chaîne de caractères, à condition de placer le nom de la variable entre accolades, comme dans l'exemple suivant :

```
$préfixe = "géo";
echo "La {$préfixe}graphie est enseignée ici";
```

L'exécution de ces deux instructions affiche :

```
La géographie est enseignée ici
```

Si vous omettez les parenthèses, PHP recherchera une variable du nom de \$préfixegraphie, qu'il ne trouvera évidemment pas. Il enverra donc un message d'avertissement à ce sujet, puis affichera simplement le reste de la phrase :

```
La est enseignée ici
```

Lorsque vous voulez afficher une page Web (ou une partie de page Web), vous devez tenir compte des trois éléments suivants :

- ✓ **Le programme PHP.** Les instructions PHP *echo* que vous écrivez.
- ✓ **Le code source HTML.** C'est celui que vous pouvez voir lorsque vous cliquez sur le menu Affichage/Source de votre navigateur. Ce *code source* est celui qui est produit par les instructions *echo*.
- ✓ **La page Web.** C'est ce que vont voir vos utilisateurs. Elle est produite par le code source HTML.



L'instruction *echo* envoie exactement ce que vous y avez mis. Ni plus ni moins. S'il ne s'y trouve pas de balise HTML, le navigateur n'en recevra aucune.

Les caractères de contrôle (tabulation, retour chariot, alinéa...) présents dans une chaîne de caractères qui figure dans une instruction *echo* **ne sont pas** des balises HTML. En conséquence, ils n'ont aucune action sur les sorties affichées par le navigateur et ne font que modifier la façon dont se présente ce qu'il reçoit avant traitement. Si vous voulez par exemple provoquer un saut de ligne dans la sortie produite par PHP, vous devez inclure un caractère spécial (`\n`). Mais cette syntaxe n'envoie *pas* une balise HTML de saut de ligne sur la page Web.

Les Figures 7.1 et 7.2 en montrent un exemple, obtenu à l'aide du Listing 7.1.

Listing 7.1 Exemple permettant de voir les différences entre ce que reçoit le navigateur et ce qu'il affiche.

```
<?php
echo "Hello World!";
echo "Hello World!";
echo "Here I am!";
echo "Hello World!\n";
echo "Here I am!";
echo "Hello World!<br>";
echo "Here I am!";
echo "Hello World!<br>\n";
echo "Here I am!";
?>
```

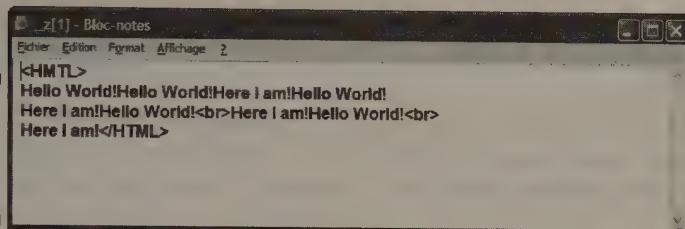


Figure 7.1 :
Ce qui est
reçu par le
navigateur.

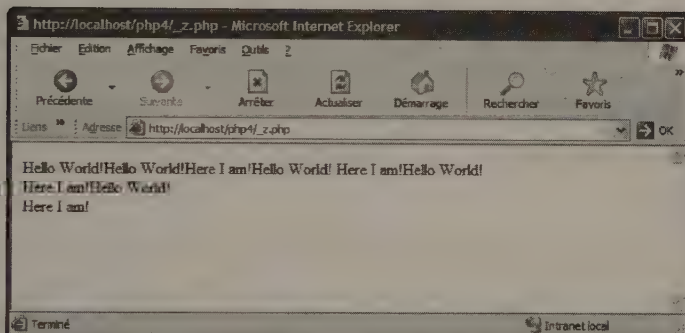


Figure 7.2 :
Ce qui est
affiché par le
navigateur.

Vous pouvez constater que le code `\n` provoque bien un changement de ligne dans la sortie, c'est-à-dire dans le code source HTML. Ce qui se traduit dans la page Web par l'insertion d'un espace. Si vous voulez changer de ligne dans cette page elle-même, il vous faut envoyer une balise HTML adaptée, comme `<br>`. C'est ce que montre le listing précédent.

Instructions d'affectation

C'est au moyen des instructions d'affectation qu'on peut donner des valeurs aux variables. L'opérateur d'affectation est le signe égal (=). Le nom de la variable s'écrit à sa gauche et la valeur à donner à cette variable à sa droite. Exemple :

```
$maVariable = 345.76;
```

Au lieu d'une simple constante, on peut utiliser une variable ou une *expression*, c'est-à-dire une combinaison de constantes et de variables au moyen d'opérateurs du même type. On ne peut pas, en effet, mélanger des éléments de nature différente (valeurs numériques et

chaînes de caractères). Voici quelques exemples d'affectations correctement écrites :

```
$nombre = 2;
$nombre = 2+1;
$nombre = (2 - 1) * (4 * 5) -17;
$nombre2 = $nombre + 3;
chaîne = "Hello World";
chaîne2 = chaîne." encore !";
```

Il faut savoir que, si vous mélangez les deux types, vous n'obtiendrez pas de message d'erreur, mais le résultat ne sera presque jamais celui que vous attendiez. Voici des exemples d'instructions incorrectement écrites :

```
$nombre = 2;
$chaîne = "Hello";

$res1 = $nombre + $chaîne;
$res2 = $nombre.$chaîne;
echo $res1;
echo $res2; ..
```

Voici ce qui sera affiché :

```
2      ($chaîne est considérée comme valant 0)
2Hello ($nombre est considérée comme un caractère)
```

Incrémentation et décrémentation

On utilise souvent une variable comme *compteur*. Par exemple, pour être sûr que le visiteur voit bien le logo de votre entreprise dans sa page Web, vous décidez de l'afficher trois fois. Vous mettez une variable à zéro puis, chaque fois que vous affichez le logo, vous ajoutez 1 à cette variable. Lorsque cette valeur atteint 3, vous savez qu'il est temps d'arrêter cet affichage. Voici ce que vous pourriez écrire :

```
$compteur = 0;
$compteur = $compteur + 1;
echo $compteur;
```

La première fois, vous afficherez 1, les fois suivantes 2, puis 3. Mais il y a moyen d'adopter une forme d'écriture plus concise :

```
$compteur = 0;
$compteur++;
echo $compteur;
```

La suite des deux signes "+" sans espace entre eux est appelée *opérateur d'incrément*. Cet opérateur a pour effet d'ajouter 1 à la variable sur laquelle il porte (qui est écrite à sa gauche).

Symétriquement, il existe un *opérateur de décrémentation* qui se note "-" et qui a pour effet de soustraire 1 à la variable écrite à sa gauche :

```
$compteur--;
```

Il existe une variante de ces deux opérations qui permet d'ajouter ou de soustraire une valeur quelconque à une variable. En voici quelques exemples :

- ✓ `$compteur+=2;` on ajoute 2 à `$compteur`.
- ✓ `$compteur-=3;` on soustrait 3 de `$compteur`.
- ✓ `$compteur*=2;` on multiplie la valeur de `$compteur` par 2.
- ✓ `$compteur/=3;` on divise la valeur de `$compteur` par 3.

exit

Cette instruction a pour effet d'arrêter l'exécution d'un programme à un endroit quelconque, par exemple à la suite d'une erreur ou d'une donnée invalide. Plus aucune instruction n'est exécutée ensuite. Sa forme générale est :

```
exit ("message");
```

message est le message qui sera affiché sur l'écran au moment de l'arrêt. Par exemple :

```
exit ("Le calcul est terminé.");
```

Sa présence est facultative et on peut écrire tout simplement :

```
exit;
```


Il existe une autre instruction pour arrêter un programme, l'instruction `die` qui s'utilise absolument de la même façon que `exit` :

```
die("Ce programme se meurt");
```

Appels de fonctions

Les *fonctions* sont des blocs d'instructions destinés à accomplir telle ou telle tâche. Elles sont écrites de façon à pouvoir être *appelées* autant de fois que nécessaire, sans qu'il soit besoin de réécrire les instructions qu'elles contiennent. On parle quelquefois à ce propos de *sous-programmes*. Une fonction doit être *déclarée* et recevoir un *nom* (sans cela, comment feriez-vous pour l'appeler ?). Voici un exemple d'appel de fonction :

```
maFonction();
```

Supposons que vous ayez écrit une fonction qui affiche, en les extrayant d'une base de données, les noms et prénoms de tous les clients qui résident dans un certain département. Vous écrivez cette tâche sous forme de fonction et lui donnez le nom `afficheClients()`. Pour utiliser cette fonction en pratique, vous devrez lui indiquer le nom du département auquel vous vous intéressez. Vous écrirez par exemple :

```
afficheClients("Val de Marne");
```



Pour distinguer les noms de fonctions des noms de variables, il est d'usage de les faire suivre d'une paire de parenthèses.

Ce qui se trouve à l'intérieur des parenthèses est appelé *argument* de la fonction. Sa valeur est *passée* à la fonction. Il est possible de passer plusieurs arguments dans un appel de fonction, pourvu que cela ait été prévu lors de l'écriture de celle-ci.

Il existe un très grand nombre de fonctions natives dans PHP (environ 2 800 !). Nous en avons déjà rencontré quelques-unes : `unset()`, `date()`, `strtotime()`, etc. Nous ne décrivons évidemment pas toutes ces fonctions, mais nous aurons l'occasion d'en rencontrer bien d'autres dans les chapitres suivants.

Voici par exemple comment dé-définir une variable nommée `$mavariante` par un appel à `unset()` :

```
unset($mavariante);
```

Les tableaux

Les tableaux sont des structures de variables complexes. Dans un *tableau* (*array*, en anglais), on peut ranger un ensemble de variables ordinaires (dites *scalaires*) sous un nom unique. C'est un excellent moyen de conserver un groupe de variables de même nature ou concernant le même objet. Par exemple, vous pouvez conserver dans un tableau trois éléments concernant des chemises : leur taille, leur couleur et leur prix, et appeler le tableau qui les contiendra `$chemisesInfo`. Ce tableau, vous pourrez ensuite le manipuler de diverses façons. Le trier, par exemple. Pour cette dernière tâche, vous verrez plus loin que PHP propose plusieurs fonctions.

Création d'un tableau

La façon la plus simple de créer un tableau est d'ajouter une paire de crochets à la suite d'un nom de variable et de lui affecter une valeur. Pour créer un tableau contenant les différents types d'animaux du premier de nos deux exemples, on pourrait écrire :

```
$animaux[1] = "dragon";
```

Dès lors, il existe un tableau `$dragon[]` qui ne contient pour l'instant qu'une seule valeur.



De même qu'on écrit une paire de parenthèses à droite du nom d'une fonction pour indiquer qu'il s'agit d'un nom de fonction, on a l'habitude d'écrire une paire de crochets à droite d'un nom de tableau pour marquer qu'il s'agit d'un tableau et non d'une variable scalaire (ordinaire).

Pour ajouter d'autres valeurs dans ce tableau, on peut ensuite écrire :

```
$animaux[1] = "chat";  
$animaux[2] = "licorne";
```

D'un autre côté, un tableau peut être considéré comme une liste de couples (ou *paires*) de *clés* et de *valeurs*. Dans l'exemple que nous venons de donner, les clés sont des nombres (1, 2, 3). Plus simplement, on peut écrire :

```
$animaux[] = "dragon";  
$animaux[] = "chat";
```

```
$animaux[] = "licorne";
```

En l'absence d'*indice* (le nombre placé entre crochets), la première valeur adoptée est 0 et les valeurs suivantes sont les entiers 1, 2, 3... Si, ensuite, vous écrivez :

```
echo $animaux[1];
```

vous afficherez bien :

```
chat
```

Au lieu de donner des valeurs numériques aux indices, on peut leur attribuer des *noms* écrits sous forme de chaînes de caractères, comme dans l'exemple suivant qui concerne une liste des chefs-lieux de départements :

```
$cheflieu["Oise"] = "Beauvais";
$cheflieu["Yvelines"] = "Versailles";
$cheflieu["Val de Marne"] = "Créteil";
```

Yvelines est ici la clé du second élément du tableau `$cheflieu[]`, et la valeur qui correspond à cette clé est Versailles.

Il existe un moyen encore plus simple de créer un tableau et de le garnir de quelques valeurs :

```
$cheflieu = array("Beauvais", "Versailles", "Créteil");
```

Si on utilisait des clés alphabétiques à la place des indices numériques, il faudrait écrire :

```
$cheflieu = array("Oise" => "Beauvais",
                 "Yvelines" => "Versailles",
                 "Val de Marne" => "Créteil");
```



S'il n'y a pas ambiguïté, on peut parfaitement utiliser comme délimiteur de chaîne l'apostrophe à la place des guillemets.

Afficher le contenu d'un tableau

Un élément de tableau peut être affiché à l'aide de la commande `echo`, comme ceci :

```
$cheflieu["Yvelines"];
```

Pour inclure cet élément dans une longue instruction `echo` comportant une chaîne placée entre guillemets, vous devrez le délimiter à l'aide d'accolades. Par exemple :

```
echo "La préfecture des Yvelines est {$cheflieu["Yvelines"]}<br>;
```

Vous pouvez visualiser la structure et les valeurs d'un tableau en faisant appel à l'instruction `print_r`. Voici comment afficher le tableau `$cheflieu` :

```
print_r($cheflieu);
```

La sortie va alors indiquer :

```
Array
(
    [Oise] => Beauvais
    [Yvelines] => Versailles
    [Val de Marne] => Créteil
)
```

Pour chaque élément, vous obtenez sa clé et sa valeur.



L'affichage dans la page Web se fait au format HTML. Le navigateur n'affichera donc qu'une seule longue ligne. La présentation ci-dessus est celle qui vous est proposée lorsque vous demandez à voir le code source. Pour obtenir une présentation plus lisible, vous devez envoyer des balises HTML qui demandent au navigateur d'afficher le texte exactement comme il arrive.

Suppression d'éléments dans un tableau

Pour supprimer un élément d'un tableau, il ne suffit pas de lui donner la valeur 0 (dans le cas d'un tableau de valeurs numériques) ou "" (dans le cas d'un tableau de chaînes de caractères), il faut utiliser la fonction native `unset()` que nous avons déjà rencontrée et écrire, par exemple :

```
unset($animaux[2]);
```

En procédant autrement, vous obtiendriez simplement un élément nul ou vide, mais le tableau contiendrait toujours le même nombre d'objets !

Tri de tableaux

Trier un tableau signifie réordonner son contenu selon un certain critère, par exemple, en ordre alphabétique ascendant. (Par défaut, les éléments sont présentés dans l'ordre où ils ont été créés.) Pour cela, PHP propose plusieurs fonctions. La plus simple est `sort()`. Exemple :

```
sort($animaux);
```

Cette fonction trie le tableau selon les valeurs qu'il contient et réassigne de nouvelles valeurs aux clés selon le résultat du tri. S'il s'agit d'un tableau de chaînes de caractères, celles dont l'initiale est un chiffre apparaîtront en tête, suivies par celles dont l'initiale est une majuscule, puis par celles dont l'initiale est une minuscule. Reprenons notre exemple du tableau des animaux créé par :

```
$animaux = array("dragon", "chat", "licorne");
```

Après avoir exécuté l'instruction :

```
sort($animaux);
```

le tableau contiendra :

```
$animaux[0] = "chat";  
$animaux[1] = "dragon";  
$animaux[2] = "licorne";
```

Prenons maintenant le cas d'un tableau doté de clés alphanumériques comme le tableau `$cheflieu[]`. Trions-le au moyen de l'instruction :

```
sort($cheflieu);
```

Nous allons obtenir :

```
$cheflieu[0] = "Beauvais";  
$cheflieu[1] = "Créteil";  
$cheflieu[2] = "Versailles";
```

Les clés ont donc été converties en pointeurs numériques, ce qui n'est sans doute pas ce que nous voulions obtenir. Pour effectuer correctement le tri, il faut utiliser la variante `asort()` :

```
asort($cheflieu);
```


Le tableau devient alors :

```
$cheflieu("Oise") = "Beauvais";
$cheflieu("Val de Marne") = "Créteil";
$cheflieu("Yvelines") = "Versailles";
```

Les valeurs sont classées dans l'ordre alphabétique croissant, et la correspondance avec les clés est parfaitement conservée. Par contre, la situation serait différente si les clés étaient numériques. Supposons que le tableau soit défini ainsi :

```
$cheflieu[0] = "Créteil";
$cheflieu[1] = "Versailles";
$cheflieu[2] = "Beauvais";
```

Exécutons alors l'instruction :

```
asort($cheflieu);
```

Nous allons obtenir le résultat suivant :

```
$cheflieu[2] = "Beauvais";
$cheflieu[0] = "Créteil";
$cheflieu[1] = "Versailles";
```

Les valeurs sont bien triées avec les clés, mais le tableau ainsi produit sera difficilement exploitable !

Le Tableau 7.1 donne une liste d'autres fonctions de tri proposées par PHP.

Tableau 7.1 : Fonctions de tri proposées par PHP.

Fonction de tri	Rôle
<code>sort(\$tableau)</code>	Tri par valeurs en assignant de nouvelles valeurs aux clés.
<code>asort(\$tableau)</code>	Tri par valeurs, les clés étant conservées.
<code>rsort(\$tableau)</code>	Tri par valeurs en ordre inverse, en assignant de nouvelles valeurs aux clés.
<code>arsort(\$tableau)</code>	Tri par valeurs en ordre inverse, les clés étant conservées.
<code>krsort(\$tableau)</code>	Tri sur les clés.
<code>krsort(\$tableau)</code>	Tri en ordre inverse sur les clés.
<code>usort(\$tableau, func)</code>	Tri au moyen de la fonction utilisateur <code>func()</code> (voir plus loin, dans ce même chapitre).

Extraction de valeurs à partir d'un tableau

Vous pouvez accéder directement à n'importe quelle valeur d'un tableau en écrivant, par exemple :

```
$cheflieu_Yvelines = $cheflieu["Yvelines"];
echo $cheflieuYvelines;
```

Ce qui affichera :

```
Versailles
```

Plus simplement, vous pouvez condenser ces deux instructions en une seule :

```
echo $cheflieu["Yvelines"];
```

Dans `echo`, si le nom du tableau se trouve à l'intérieur d'une autre chaîne de caractères, vous devez le placer entre une paire d'accollades, de cette façon :

```
echo "Le chef-lieu des Yvelines est {$cheflieu['Yvelines']}<br>;
```

Si vous placez dans une instruction un élément de tableau qui n'existe pas, un message d'avertissement sera émis (voir à ce sujet le Chapitre 6). Essayons ceci :

```
$Préfecture = $cheflieu["Ivelines"];
```

Le nom du tableau est valide, mais il y a une faute de frappe dans la clé. PHP va réagir en indiquant :

```
Notice: Undefined index: Ivelines in...
```

Cela ne bloque pas l'exécution du script. Les instructions qui suivent continueront à s'exécuter. Par contre, toute occurrence suivante de la variable `$Préfecture` n'affichera qu'un espace vide. Vous pouvez éviter l'apparition du message en plaçant le caractère `@` devant le nom de la variable :

```
@$Préfecture = $cheflieu["Ivelines"];
```

Vous pouvez extraire en une seule fois plusieurs valeurs d'un tableau au moyen de l'instruction `list` qui permet de les ranger dans une suite de variables scalaires. Exemple :

```
$infoChemises = array ("XL", "bleu", 12.0);
sort ($infoChemises);
list($valeur1,$valeur2) = $infoChemises;
echo $valeur1,"<br>";
echo $valeur2,"<br>";
```



Bien que l'écriture de `list` ressemble à celle d'une fonction, syntaxiquement parlant, ce n'est pas une fonction. Ne serait-ce que parce qu'elle s'écrit à gauche du signe `=`. On appelle cette entité une *structure de contrôle*.

La première instruction crée le tableau `$infoChemises[]`. La deuxième trie le tableau. La troisième affecte les première et deuxième valeurs du tableau aux variables scalaires `$valeur1` et `$valeur2`, de la même façon que si on avait écrit :

```
$valeur1 = $infoChemises[0];
$valeur2 = $infoChemises[1]
```

La troisième valeur du tableau, `$infoChemises[2]`, n'est pas utilisée dans cet exemple, puisqu'il n'y a que deux variables dans l'appel de `list`. Ce qui sera affiché est :

```
XL
bleu
```

En effet, "XL" commence par une majuscule, laquelle a priorisé sur l'initiale bas de casse de "bleu", à la suite du tri par la fonction `sort()`.

Dans certains cas, vous vous intéressez aux clés, non aux valeurs, comme dans l'exemple suivant :

```
$infoChemises = array ("taille" => "XL",
                      "couleur" => "bleu",
                      "prix" => 12.0);
$valeur = $infoChemises['taille'];
$célé = key($infoChemises);
echo "$célé: $valeur<br>";
```

Vous verrez s'afficher :

```
taille: XL
```

Vous pouvez extraire toutes les valeurs d'un tableau au moyen de la fonction `extract()`. Chaque valeur est recopiée dans une variable ayant pour nom la valeur de la clé précédée du caractère `$`.

Par exemple, les instructions qui suivent extraient toutes les informations du tableau `infoChemises[]` et les affichent :

```
$infoChemises = array ("taille" => "XL", "couleur" => "bleu", "prix" =>
12.0);
extract($infoChemises);
echo "taille : $taille<br>couleur : $couleur<br>prix : $prix<br>" ;
```

On obtient :

```
taille : XL
couleur : bleu
prix : 12
```



N'oubliez surtout pas de nommer les clés. Sinon, PHP sera incapable de construire les variables correspondantes lors de l'extraction.

Parcours des éléments d'un tableau

Vous aurez souvent à effectuer un certain traitement sur tous les éléments d'un tableau : afficher chaque valeur, la placer dans une base de données ou y ajouter une constante. C'est ce qu'on appelle une *itération*. On dit aussi quelquefois *parcourir* le tableau (en anglais : *traversing the array*). Pour exécuter cette tâche, PHP vous propose les deux moyens suivants :

- ✓ **Manuellement** : Déplacer un pointeur d'un élément à un autre.
- ✓ **Avec `foreach`** : Cette instruction est conçue pour faire automatiquement le travail, du début à la fin du tableau, valeur par valeur.

Procédure manuelle

Pour extraire un élément d'un tableau, vous pouvez utiliser un pointeur. Représentez-vous votre tableau sous la forme d'une liste et imaginez un pointeur désignant l'un des éléments de cette liste. Il reste immobile tant que vous ne le déplacez pas. Pour le déplacer, vous disposez des fonctions suivantes, qui portent sur le tableau dont le nom est placé à l'intérieur des parenthèses :

- ✓ **`current()`** : Désigne l'élément sur lequel est dirigé le pointeur. Celui-ci ne change pas de valeur.

- ✓ `next()` : Déplace le pointeur vers l'avant. Il désigne maintenant l'élément suivant son ancienne position.
- ✓ `previous()` : Déplace le pointeur vers l'arrière. Il désigne maintenant l'élément précédant son ancienne position.
- ✓ `end()` : Déplace le pointeur et le place sur le dernier élément du tableau.
- ✓ `reset()` : Déplace le pointeur et le place sur le premier élément du tableau.

Le Listing 7.2 vous présente quelques exemples d'utilisation de ces fonctions.

Listing 7.2 : Exemples de déplacements manuels d'un pointeur dans un tableau.

```
<?PHP
$cheflieu["Oise"] = "Beauvais";
$cheflieu["Yvelines"] = "Versailles";
$cheflieu["Val de Marne"] = "Créteil";
$valeur = current ($cheflieu);
echo "$valeur<br>";
$valeur = next ($cheflieu);
echo "$valeur<br>";
$valeur = next ($cheflieu);
echo "$valeur<br>";
?>
```

Vous obtiendrez ainsi l'affichage suivant :

```
Beauvais
Versailles
Créteil
```

A moins d'avoir été déplacé auparavant, le pointeur désigne le premier élément du tableau lorsque vous commencez à parcourir celui-ci. Si vous n'êtes pas certain de l'emplacement courant, ou si la réponse obtenue n'est pas conforme à votre attente, réinitialisez le pointeur avant de parcourir le tableau, comme ceci :

```
reset($cheflieu);
```


Pour utiliser cette méthode sur l'ensemble d'un tableau, vous devez avoir une instruction d'affectation et une autre pour l'affichage, et ce pour chacun des éléments (en l'occurrence, les 96 départements de la France métropolitaine).

Cette méthode vous apporte une grande souplesse, puisque vous pouvez naviguer à votre guise dans le tableau, dans l'ordre souhaité. Vous avez la possibilité de revenir en arrière, de vous placer directement à la fin, d'éviter certaines valeurs, et ainsi de suite. Par contre, un parcours systématique du début vers la fin sera réalisé plus efficacement en faisant appel à la structure `foreach`.

Utilisation de `foreach` ()

Cette instruction permet, à chaque exécution, d'accéder à l'élément suivant du tableau et de répéter, pour chacune de ces valeurs, le bloc d'instructions qui la suit. Sa forme générale est la suivante :

```
foreach ($nomTableau as $nomClé => $nomValeur)
{
    bloc d'instructions
}
```

avec :

- ✓ `$nomTableau` : Nom du tableau à explorer.
- ✓ `$nomClé` : Nom de la variable dans laquelle vous voulez placer la clé de l'élément. Cette valeur est facultative. Si vous l'omettez, la valeur sera rangée dans la variable `$nomValeur`.
- ✓ `$nomValeur` : Nom de la variable dans laquelle sera placée la valeur.

Par exemple, les instructions suivantes parcourent le tableau des chefs-lieux et en affichent chaque élément :

```
ksort($cheflieu);

foreach($cheflieu as $dept => $ville)
{ echo "$ville : $dept<br>";
}
```

ce qui affichera :

Beauvais : Oise
Créteil : Val de Marne
Versailles : Yvelines

Si vous aviez écrit :

```
foreach($cheflieu as $ville)
{ echo "$ville<br>";
}
```

vous auriez obtenu :

Beauvais
Créteil
Versailles



Initialement, `foreach` place le pointeur en tête du tableau. Il n'est donc pas nécessaire d'appeler la fonction `reset()`.

Tableaux à plusieurs dimensions

Dans les sections précédentes, je vous ai parlé des tableaux à une seule dimension : ceux qui ne représentent qu'une seule liste d'éléments (paires clé/valeur). Cependant, il faut savoir qu'il est possible de construire et d'utiliser des tableaux à plusieurs dimensions. Par exemple, supposez que vous vouliez placer ces prix de produits dans une variable :

- ✓ chemise, 20.0
- ✓ pantalon, 22.5
- ✓ nappe, 25.6
- ✓ dessus de lit, 50.0
- ✓ lampe, 44.0
- ✓ carpeppe, 75.0

Vous pouvez procéder ainsi :

```
$prixProduit['chemise'] = 20.;
$prixProduit['pantalon'] = 22.5;
$prixProduit['nappe'] = 25.6;
$prixProduit['dessus de lit'] = 50.0;
$prixProduit['lampe'] = 44.;
$prixProduit['carpeppe'] = 75.;
```

Un programme pourra facilement parcourir cette liste pour savoir le prix de chaque article. Mais supposez que vous ayez 3 000 produits. Votre programme devra alors les passer en revue pour trouver celui dont *chemise* ou *carpette* est la clé.

Remarquez que cette liste comporte un mélange d'articles de nature différente : habillement, linge de maison, meubles. Si vous classiez ces produits par catégorie, le tableau à explorer serait de plus petite taille. Pour cela, vous pouvez écrire :

```
$prixProduit['habillement']['chemise'] = 20.;
$prixProduit['habillement']['pantalon'] = 22.5;
$prixProduit['linge de maison']['nappe'] = 25.6;
$prixProduit['linge de maison']['dessus de lit'] = 50.0;
$prixProduit['meubles']['lampe'] = 44.;
$prixProduit['meubles']['carpette'] = 75.;
```

Vous avez ainsi créé ce qu'on appelle un *tableau à plusieurs dimensions*, parce qu'il est en réalité un tableau de tableaux. La Figure 7.3 vous en montre la structure.

Figure 7.3 :
Structure
d'un tableau
à deux
dimensions.

\$prixProduits	Clé	Valeur	
		clé	valeur
habillement		chemise	20.0
		pantalon	22.5
linge de maison		nappe	30.0
		dessus de lit	50.0
meubles		lampe	44.0
		carpette	75.0

La valeur de chacun des éléments du premier rang du tableau \$prixProduit (habillement, linge de maison, meubles) est un couple clé/valeur. On peut concevoir des tableaux avec 3, 4, 5... dimensions ou plus encore. Mais il est difficile de se représenter intellectuellement la signification de tableaux à plus de trois dimensions. Le risque de confusion s'accroît exponentiellement avec le nombre de dimensions !

On extrait les valeurs des tableaux à plusieurs dimensions de la même façon que celles de tableaux simples. Pour accéder directement à une valeur, vous écrirez, par exemple :

```
$prixChemise = $prixProduit['habillement']['chemise'];
```

OU :

```
echo $prixProduit['habillement']['chemise'];
```

Si vous associez une de ces valeurs à une chaîne de caractères entre guillemets dans une instruction `echo`, vous devez la placer entre accolades comme le montre l'exemple suivant :

```
echo "Le prix d'une chemise est  
($prixProduit['habillement']['chemise']) &#8364;<br>";
```



Le signe \$ qui signale le début d'un nom de variable doit être placé immédiatement derrière l'accolade, sans espace intermédiaire.

Vous pouvez parcourir les éléments d'un tableau à plusieurs dimensions au moyen de `foreach()`, de la même façon que pour un tableau simple. Mais, bien entendu, il faut un `foreach()` pour chacune des dimensions, ce qui conduit à les *imbriquer* comme dans cet exemple :

```
echo "<table border cellpadding=5>";
foreach ($prixProduit as $catégorie)
{ foreach ($catégorie as $produit => $prix)
  { $prix_f = sprintf ("%01.2f", $prix);
    echo "<tr><td>$produit</td><td>$prix_f &#8364;</td></tr>";
  }
}
echo "</table>";
```



L'entité de caractère représentant le symbole de l'euro (€) est `€`.

Vous pouvez voir le résultat sur la Figure 7.4.

Voici comment s'exécute ce court programme :

1. La balise `<table border cellpadding=5>` est envoyée au navigateur pour créer un tableau HTML.
2. On extrait la première paire clé/valeur du tableau `$prixProduit` et on place la valeur dans la variable `$catégorie`.
3. On extrait la première paire clé/valeur du tableau `$catégorie`, on place la clé dans la variable `$produit` et la valeur dans la variable `$prix`.

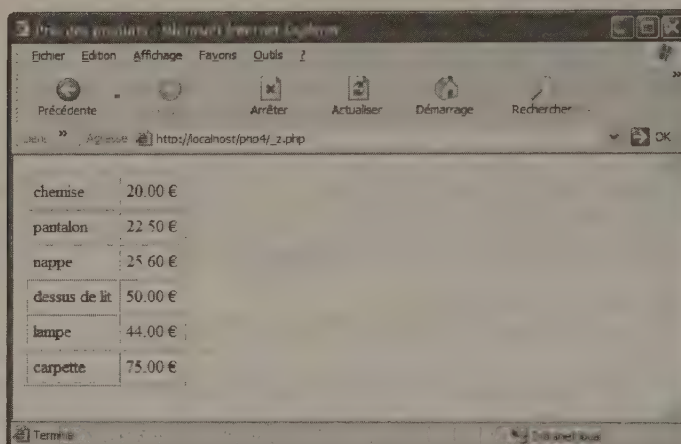


Figure 7.4 :
Affichage
d'éléments
d'un tableau
à deux
dimensions.

4. Le prix est mis en forme par l'instruction `sprintf()`.
5. On envoie au navigateur une ligne du tableau constituée par les balises appropriées et les deux variables précédentes.
6. On extrait la paire clé/valeur suivante du tableau `$prixProduit`.
7. Le prix est mis en forme et on envoie au navigateur une nouvelle ligne du tableau constituée par les balises appropriées et les deux variables extraites de `$prixProduit`.
8. Comme il n'y a plus rien dans le tableau `$catégorie`, la boucle interne se termine et on en sort.
9. On extrait la paire clé/valeur suivante du tableau `$prixProduit` et on place la valeur dans la variable `$catégorie`, qui est un tableau.
10. On répète les étapes 1 à 9 jusqu'à ce que la dernière paire clé/valeur du tableau `$prixProduit` soit utilisée.
11. On sort de la boucle la plus externe et on envoie la balise `</table>` au navigateur pour refermer le tableau HTML.

Autrement dit, le `foreach` externe traite les éléments du premier rang de `$prixProduit`, tandis que le `foreach` interne s'occupe des éléments du second rang correspondant à chaque élément du premier rang.

Instructions conditionnelles usuelles

Une *instruction conditionnelle* est une instruction qui permet d'exécuter ou non l'instruction ou le bloc d'instructions qui la suit lorsque certaines conditions sont remplies. Voici deux types d'instructions conditionnelles courantes :

- ✓ **if** : Définit une condition et en teste le résultat. S'il vaut VRAI, le bloc qui suit est exécuté.
- ✓ **switch** : Définit une suite de conditions possibles et les teste une par une. Les instructions qui suivent celles qui répondent VRAI sont exécutées.

if

Une instruction **if** teste une certaine condition. Si sa valeur est VRAI, le bloc d'instructions qui la suit est exécuté. Sa forme générale est la suivante :

```
if (condition)
{
    bloc d'instructions ...
}
elseif (condition)
{
    bloc d'instructions ...
}
else
{
    bloc d'instructions ...
}
```

Cette instruction comprend trois sections :

- ✓ **if** : Cette section est obligatoire. Elle teste une certaine condition.
 - **Si la condition vaut VRAI** : Le bloc d'instructions qui suit est exécuté. Le programme se déplace ensuite vers l'instruction qui suit cette instruction conditionnelle. Les sections **else** ou **elseif** éventuelles sont ignorées.
 - **Si la condition ne vaut pas VRAI** : Le bloc d'instructions qui suit n'est pas exécuté. Le programme se déplace alors

vers l'instruction qui suit. Il peut s'agir d'un `else` ou d'un `elseif`, ou encore de l'instruction qui suit le bloc conditionnel.

Quel que soit le résultat du test, le programme passe donc ensuite à l'instruction qui suit ce bloc.

- ✓ **elseif** : Cette section est facultative. Elle aussi teste une condition. Vous pouvez écrire autant de sections `elseif` que vous le souhaitez, mais au-delà de deux vous courez le danger de ne plus très bien savoir où vous en êtes de vos tests.
 - **Si la condition vaut VRAI** : Le bloc d'instructions qui suit est exécuté.
 - **Si la condition ne vaut pas VRAI** : Le bloc d'instructions qui suit n'est pas exécuté.

Le déroulement est identique au traitement de `if`.

- ✓ **else** : Cette section, elle aussi, est facultative. Si elle existe et que la condition du `if` ou du `elseif` qui la précède renvoie FAUX, elle est exécutée. Si elle vaut VRAI, cette section est ignorée.

Chacune des conditions peut consister en une simple comparaison ou en une association de comparaisons. Nous avons vu en quoi consistaient ces deux formes dans la section "Comparaisons" du Chapitre 6. Le Listing 7.3 vous propose un exemple d'emploi d'instructions conditionnelles. Supposons que vous ayez quatre versions de votre catalogue en français, anglais, allemand et italien. Vous voulez afficher la langue qui convient en fonction de la nationalité du client. Le listing qui suit définit un message adapté à la langue appropriée à partir d'une variable appelée `$pays`.

Voici comment fonctionne cette séquence :

1. La variable `$pays` est comparée à "Allemagne". S'il y a égalité, `$version` prend la valeur "allemande" et `$message` reçoit un texte en allemand. S'il n'y a pas égalité, ce bloc est ignoré et on passe au test suivant.
2. La variable `$pays` est comparée à "France". S'il y a égalité, `$version` prend la valeur "française" et `$message` reçoit un texte en français. S'il n'y a pas égalité, ce bloc est ignoré et on passe au test suivant.

Listing 7.3 : Exemple d'une cascade d'instructions if.

```

<?PHP
if ($pays == "Allemagne" )
{ $version = "allemande";
  $message = " Sie sehen unseren Katalog auf Deutsch";
}
elseif ($pays == "France" )
{ $version = "française";
  $message = " Vous verrez notre catalogue en français";
}
elseif ($pays == "Italie" )
{ $version = "italienne";
  $message = " Vedrete il nostro catalogo in Italiano";
}
else
{ $version = "anglaise";
  $message = "You will see our catalog in English";
}
echo "Version $version : $message<br>";
?>

```

3. La variable \$pays est comparée à "Italie". S'il y a égalité, \$version prend la valeur "italienne" et \$message reçoit un texte en français. S'il n'y a pas égalité, ce bloc est ignoré et on passe au test suivant.
4. Si aucune des conditions précédentes n'a été remplie, \$version prend la valeur "anglaise" et \$message reçoit un texte en anglais.

Quel qu'ait été le résultat des tests précédents, l'instruction echo est alors exécutée pour afficher le type de version et le message dans cette langue.



Lorsque le bloc d'instructions "protégé" par une instruction conditionnelle ne contient qu'une seule instruction, on peut se dispenser des accolades, voire, si l'instruction est suffisamment courte, l'écrire sur la même ligne. Ainsi, au lieu de :

```

elseif ($pays == "France" )
{ $version = "française";
}

```

on pourrait écrire :

```
elseif ($pays == "France" ) $version = "française";
```

Bien que le programme ainsi écrit permette d'éviter quelques fautes de frappe, il est moins facile à relire lorsque plusieurs instructions conditionnelles se succèdent.

Vous pouvez imbriquer plusieurs `if`. Supposez, par exemple, que vous vouliez prendre contact avec vos clients de Lozère : par e-mail, pour ceux qui utilisent l'Internet ; par courrier postal, pour les autres. Voici comment vous pourriez procéder :

```
if (departClient == "Lozère")
{ if ($email != "")
    $methodeContact = "lettre";
  else
    $methodeContact = "e-mail";
}
else
    $methodeContact = "aucune";
```

Si le client n'habite pas la Lozère, tout le bloc suivant le premier `if` est ignoré et on exécute la toute dernière instruction. S'il habite la Lozère, on regarde si son adresse e-mail existe. (S'il n'en a pas, c'est une chaîne de caractères vide.) Selon la réponse à ce test, on choisit la méthode de contact.

switch

Cette instruction permet de traiter le cas des choix multiples avec plus d'élégance qu'une cascade de `if`. Nous allons reprendre le précédent exemple et le traiter avec un `switch`. Le Listing 7.4 vous présente le programme résultant.

Dans l'instruction `switch`, on commence par indiquer entre parenthèses le nom de la variable qui va être testée à la suite du `switch` proprement dit (ici : `$pays`). On trouve ensuite une succession de blocs `case` dont chacun indique avec quelle constante sera comparée la variable du `switch`. Cette comparaison s'effectue toujours par un test d'égalité. Vient alors le bloc d'instructions à exécuter si le résultat de la comparaison est VRAI.



Ce bloc doit nécessairement se terminer par une instruction `break` qui fait sortir du `switch`, sauf si c'est le dernier de ce bloc d'instruction. Le programme reprend à l'instruction suivant l'accolade

Listing 7.4 : Exemple d'une succession de tests dans une instruction switch.

```

<?PHP
switch ($pays)
{ case "Allemagne" :
    $version = "allemande";
    $message = " Sie sehen unseren Katalog auf Deutsch";
    break;

    case "France" :
        $version = "française";
        $message = " Vous verrez notre catalogue en français";
        break;

    case "Italie" :
        $version = "italienne";
        $message = " Vedrete il nostro catalogo in Italiano";
        break;

    default:
        $version = "anglaise";
        $message = "You will see our catalog in English";
    }
echo "Version $version : $message<br>";
?>

```

refermant le switch. Si on omettait le break, on "tomberait" dans le bloc suivant qui serait systématiquement exécuté.

La clause default commande un bloc d'instructions qui sera exécuté lorsque aucune des comparaisons précédentes (les clauses case) n'aura été satisfaite. Elle est facultative et se place généralement à la fin du bloc. Le break suivant la dernière section case n'est pas non plus obligatoire, mais il est vivement conseillé pour des raisons de clarté et de prudence.

Les boucles

Les boucles sont fréquemment utilisées dans les programmes. Une *boucle* est la répétition d'un bloc d'instructions. Dans certains cas, la boucle se répète un nombre de fois fixé à l'avance. Par exemple, une boucle qui afficherait la liste des départements français métropolitains se répéterait 95 fois. Dans d'autres cas, une boucle est exécutée jusqu'à ce qu'une certaine condition soit remplie. Par exemple,

une boucle qui afficherait la liste de produits commercialisés se répéterait jusqu'à ce que la liste soit épuisée, et ce quel que soit le nombre de ces produits. Il existe trois types de boucles :

- ✓ **for** : Répétition d'un bloc d'instructions un nombre de fois fixé au moyen d'un compteur.
- ✓ **while** : Répétition d'un bloc d'instructions tant qu'une certaine condition est vérifiée. Le test d'itération est effectué au début de la boucle.
- ✓ **do ... while** : Répétition d'un bloc d'instructions tant qu'une certaine condition est vérifiée. Le test d'itération est effectué à la fin de la boucle.

for

La boucle **for** la plus élémentaire est basée sur un compteur. Vous définissez la valeur de départ du compteur, sa valeur finale et la façon dont il va être incrémenté à chaque tour. Sa forme générale est la suivante :

```
for (début; fin; incrément)
{
    ... bloc d'instructions ...
}
```

avec :

- ✓ **début** : Expression définissant la variable qu'utilisera le compteur et sa valeur initiale. On trouve fréquemment ici : `$i = 0` ;. La variable du compteur sera `$i` et elle aura 0 pour valeur initiale. Mais il peut aussi s'agir d'une variable ou d'une combinaison de nombres, comme dans `2+2`.
- ✓ **fin** : Test d'arrêt de la boucle. Tant que sa valeur est égale à VRAI, la boucle continue à tourner. Lorsque l'incrément est positif, on trouve fréquemment ici : `$i < n` ;, `n` étant la première valeur qui ne sera pas utilisée dans la boucle puisque celle-ci se sera arrêtée avant. Il n'est pas indispensable que la variable servant de compteur figure à cet endroit, mais c'est souvent le cas. La condition de fin peut également être définie à l'aide d'une variable.

- ✓ **incrément** : Valeur dont sera incrémentée (ou décrémentée) la variable servant de compteur. On trouve fréquemment ici : `$i++`; qui incrémente `$i` d'une unité. Au lieu de `$i++`, on peut trouver `$i--`; dans le cas d'une boucle où le compteur voit sa valeur décroître à chaque tour.

A titre d'exemple, les instructions qui suivent montrent comment écrire une boucle `for` qui affichera trois fois "Hello World!" :

```
for ($i=0; $i<3; $i++) // on compte de 0 à 2
{ echo "Hello World!<br>";
}
```

Comme on le voit, le compteur n'intervient pas nécessairement dans le bloc d'instructions répété. On aurait également pu écrire cette boucle sous les deux formes qui suivent :

```
for ($i=1; $i<=3; $i++) // on compte de 1 à 3
{ echo "Hello World!<br>";
}
```

```
for ($i=3; $i>0; $i--) // on compte de 3 à 1
{ echo "Hello World!<br>";
}
```

 Comme vous devez vous en douter si vous vous souvenez de ce qui a été dit au début du Chapitre 6, les instructions du bloc qui suit le `for` n'ont nul besoin d'être indentées. Mais il est beaucoup plus facile ainsi de relire le programme.

Les boucles `for` sont particulièrement utiles pour balayer tout un tableau. Pour afficher toute la liste de vos clients contenue dans le tableau `$clients`, vous pouvez écrire :

```
for ($i=0; $i<sizeof($clients); $i++)
{ echo "$clients[$i]<br>";
}
```

La fonction `sizeof()` qui apparaît dans le test d'arrêt renvoie le nombre d'éléments contenus dans un tableau, donc, ici, le nombre de clients à afficher. Le nombre d'itérations correspondra donc exactement à la quantité d'éléments contenus dans le tableau.

 Normalement, le premier indice d'un tableau est 0 (sauf spécification délibérée de votre part). La dernière valeur du compteur à utiliser est le nombre d'éléments du tableau diminué de 1. Oublier cela est

une erreur courante. C'est donc l'un des tout premiers points à vérifier si vous n'obtenez pas le résultat escompté.

Plus sur la boucle for

La structure de la boucle `for` est souple. Elle vous permet de construire des boucles pour de multiples applications et sa forme générale se présente ainsi :

```
for ( instructions de début;
      instructions conditionnelles;
      instructions de fin)
{
    bloc d'instructions;
}
```

Avec :

- ✓ Des instructions de début qui ne sont exécutées qu'une seule fois lorsque la boucle commence.
- ✓ Des instructions conditionnelles qui sont testées lors de chaque itération dans la boucle.
- ✓ Des instructions de fin qui sont exécutées une seule fois lorsque la boucle se termine.

Chacun des trois paramètres de l'instruction `for` est séparé du suivant par un point-virgule. Ils peuvent comporter plusieurs expressions, comme c'est le cas, par exemple, de la séquence suivante qui calcule et affiche la somme des éléments du tableau `$t` comportant 50 éléments :

```
for ($i=0, $s=0; $i<50; $i++)
{ $s = $s + $t[$i];
}
echo "La somme vaut $s<br>"
```

On pourrait aller encore plus loin, en écrivant, pour faire le même calcul :

```
for ($i=0, $s=0; $i<50; $s = $s + $t[$i++])
{
}
echo "La somme vaut $s<br>"
```

Le bloc d'instructions est maintenant vide, le calcul de `$s` s'effectuant dans l'expression d'incrémement. Dans celui-ci, on voit que `$i++` a disparu. L'incrémement s'effectue, en effet, dans l'expression de l'indice elle-même : `[$i++]`.

Et, puisque le bloc est vide, il est inutile d'écrire une paire d'accolades après le `for`. Il suffit d'ajouter un point-virgule à sa suite. On aboutit alors à la forme très (trop) raccourcie suivante :

```
for ($i=0, $s=0; $i<50; $s = $s + $t[$i++]);
echo "La somme vaut $s<br>";
```

Pourquoi "trop" raccourcie ? Parce qu'ainsi on ne voit plus très bien la logique du calcul. D'une façon générale, il faut éviter ce genre de fantaisies d'écriture qui relèvent plutôt d'un certain maniérisme que d'une réelle recherche d'efficacité.

while

La boucle `while` continue de répéter un bloc d'instructions tant que certaines conditions sont vérifiées. Elle est organisée comme suit :

1. Vous définissez une condition.
2. La condition est testée avant d'entrer dans la boucle.
3. Si la condition est vérifiée, on exécute une fois la boucle.
4. On répète les étapes 2 et 3.

La forme générale d'une boucle `while` est la suivante :

```
while (condition)
{
    ... bloc d'instructions
}
```

condition est une expression conforme à toutes celles que nous avons vues jusqu'ici (reportez-vous si nécessaire au Chapitre 6). Elle fait intervenir une variable qui doit évoluer dans le bloc d'instructions, faute de quoi on ne pourrait jamais sortir de la boucle. Voici quelques exemples de conditions :

```
$test <= 10
$test1 == $test2
$a == "oui" and $b != "no"
$name != "Dupont"
```



Si la condition de la boucle `while` n'est pas remplie au départ, on ne passe jamais dans le bloc d'instructions qui suit.



Une erreur fréquente chez les débutants consiste à terminer la ligne sur laquelle est écrite le `while` par un point-virgule. Ce faisant, il est évident qu'on ne pourra jamais entrer dans la boucle. A plus forte raison, en sortir !

La séquence qui suit montre comment *il ne faut pas* écrire une boucle `while` regardant dans un tableau `$clients` s'il s'y trouve quelqu'un du nom de Dupont :

```
$clients = array ("Huang", "Dupont", "Martin");
$testvar = "non";
$k = 0;
while ( $testvar != "oui" )
{ if ($clients[$k] == "Dupont" )
  { $testvar = "oui";
    echo "Dupont est dans le tableau<br>";
  }
  $k++;
}
```

Si Dupont est dans le tableau, la variable de test `$testvar`, définie initialement comme valant "non", prend la valeur "oui" et on affiche :

Dupont est dans le tableau

La condition de la boucle n'étant plus remplie, on en sort. Si Dupont n'est pas dans le tableau, on tourne éternellement dans la boucle `while` puisque `$testvar`, la variable testée, ne change jamais de valeur.

Je vais maintenant vous donner un *bon* exemple de boucle `while` dont on est certain de sortir. Il suffit, pour cela, d'ajouter une condition exprimant qu'on est toujours dans le tableau. Cette condition va s'écrire :

```
$k < sizeof($clients)
```

Pendant que nous y sommes, nous allons ajouter un message indiquant que Dupont n'est pas dans le tableau. C'est le cas lorsqu'on est sorti de la boucle et que `$testvar` contient toujours "non". La séquence devient :

```
$clients = array ("Huang", "Dupont", "Martin");
$testvar = "non";
```



```

$ok = 0;
while ($testvar != "oui" && $ok < sizeof($clients))
{ if ($clients[$ok] == "Dupont")
    { $testvar = "oui";
      echo "Dupont est dans le tableau<br>";
    }
    $ok++;
}
if ($testvar == "non")
    echo "Dupont n'est pas dans le tableau<br>";

```

Détaillons le fonctionnement de cette boucle :

- Initialisations : les variables `$ok` et `$testvar` prennent respectivement les valeurs 0 et "non".**
- On exécute le `while`. `$testvar` ne vaut pas "oui" et `$ok` est égal à 0. La condition formée par l'association de ces deux tests est vérifiée et on entre dans la boucle.**
- Est-ce que `$clients[0]` contient "Dupont" ?**
 - Si c'est non, le bloc d'instructions qui suit est sauté.
 - Si c'est oui, le bloc d'instructions qui suit est exécuté. `$testvar` prend la valeur "oui" et le message "Dupont est dans le tableau" est affiché.
- Dans les deux cas, on fait progresser le compteur : `$ok++`. A la fin du premier tour, `$ok` vaut 1.**
- On répète les étapes 2 à 4 tant que les deux conditions du `while` ne sont pas satisfaites.**
- Lorsqu'on sort de la boucle, il faut voir de quelle façon.**
 - Si `$testvar` vaut "oui", c'est que Dupont était dans le tableau et il n'y a plus rien à faire.
 - Si `$testvar` ne vaut toujours pas "oui", c'est (implicitement) qu'on est sorti du tableau ; il faut alors afficher le message "Dupont n'est pas dans le tableau".



Remarquez que cet exemple aurait aussi bien pu être traité à l'aide d'une boucle `for`. C'est souvent le cas pour beaucoup de boucles, et le choix du type de boucles peut dépendre alors d'autres motivations comme la préférence du programmeur pour telle ou telle

forme d'écriture de boucle. Après tout, l'essentiel n'est-il pas d'écrire un programme correct ?

do ... while

Ce type de boucle ressemble beaucoup à la boucle `while`, à ce détail près que le test de répétition est fait *à la fin* de la boucle et non à son début. On est alors certain d'exécuter au moins une fois le bloc d'instructions qui suit. La forme générale de cette boucle est la suivante :

```
do
{
    ... bloc d'instructions ...
} while (condition);
```

Nous allons réécrire l'exemple précédent avec une boucle `do ... while`. Elle se présente maintenant ainsi :

```
$clients = array ("Huang", "Dupont", "Martin");
$testvar = "non";
$k = 0;
do
{ if ($clients[$k] == "Dupont")
  { $testvar = "oui";
    echo "Dupont est dans le tableau<br>";
  }
  $k++;
}
while ($testvar != "oui" && $k < sizeof($clients));
if ($testvar == "non")
    echo "Dupont n'est pas dans le tableau<br>";
```

Nous pouvons constater que les modifications sont minimales. Nous avons déplacé la ligne contenant le `while` à la fin du bloc d'instructions, et avons écrit `do` devant l'accolade ouvrante qui débute le bloc d'instructions.

Modifions un peu notre petit programme :

```
<?
$clients = array ("Huang", "Dupont", "Martin");
$testvar = "non";
$k = 0;
do
{ if ($clients[$k] == "Dupont")
  { $testvar = "oui";
```

```

    echo "Dupont est dans le tableau<br>";
}
else
{
    echo "$clients[$k], pas Dupont<br>";
}
$k++;
}
while ($testvar != "oui" && $k < sizeof($clients));
?>

```

Il va afficher :

```

Huang, pas Dupont
Dupont est dans le tableau

```

Une fois Dupont trouvé, la variable `$testvar` prend la valeur oui, ce qui va provoquer la sortie de la boucle. Supposons maintenant que la condition initiale soit inversée :

```
$testvar = "oui";
```

Le test est donc faux dès le départ. Dans une boucle `while`, aucune sortie ne serait produite. Avec `do...while`, le bloc est nécessairement exécuté une fois avant que la condition ne soit évaluée. Dans ce cas, la première ligne continuera à être affichée :

```
Huang, pas Dupont
```



Contrairement à ce qui se passe avec la boucle `while`, dans une boucle `do ... while`, la ligne où se trouve le `while` doit être terminée par un point-virgule.

Boucles infinies

Lors de l'étude de ces trois boucles, nous avons rencontré des cas dans lesquels on allait tourner éternellement dans la boucle. Nous avons volontairement créé une telle situation en étudiant la boucle `while` (voir ci-dessus).

En réalité, l'éternité a une valeur finie, car une minuterie incorporée à PHP, et dont la durée est modifiable, arrête l'exécution d'un programme au bout d'un "certain" temps lorsque l'interpréteur PHP ne produit plus rien (on dit alors, dans le jargon de la programmation, qu'il *tourne en rond*). Pour arrêter une boucle dans un programme

PHP, il vous suffit de cliquer sur le bouton Arrêt de la barre d'outils de votre navigateur.



Par défaut, la durée est configurée pour 30 secondes, mais cette valeur peut être modifiée par l'administrateur de PHP.

En dehors des causes provenant d'un point-virgule mal placé ou de l'impossibilité d'évolution de la variable testée dans la condition d'arrêt, il y a une autre cause de blocage : la confusion de l'opérateur de comparaison "==" avec l'opérateur d'affectation "=".

Le simple fait d'écrire `$testvar = "oui"` résulte en une condition ayant la valeur VRAI, puisque "oui" est différent de zéro. En outre, on modifie ainsi la valeur de `$testvar`. Alors que si l'on écrit `$testvar == "oui"`, on ne modifie pas la valeur de `$testvar` et la condition dépendra de la valeur réelle de cette variable.



Il existe un moyen assez simple de contourner ce problème. Si vous écrivez `"oui" == $testvar`, aucune affectation ne sera possible. Certes, cette rédaction est moins logique, mais elle peut vous protéger contre ce type d'erreur. Si vous ne tapez qu'un seul signe d'égalité, une erreur fatale se produira et le programme se terminera tout de suite.

Une autre erreur classique est l'oubli de l'instruction qui incrémente le compteur. Celui-ci ne bouge pas, la condition initiale n'est donc jamais modifiée, et l'on rentre dans une boucle infinie.

Rupture d'une boucle

Dans certains cas, on peut avoir besoin de modifier le cours d'une boucle. Il y a, pour cela, deux motivations possibles :

- ✓ On veut sortir immédiatement de la boucle pour une cause différente de sa condition normale. On utilisera alors l'instruction `break`.
- ✓ On a exécuté une partie des instructions de la boucle. On veut sauter celles qui suivent, puis reprendre l'exécution normale de la boucle. On utilisera alors l'instruction `continue`.

En particulier, `break` est le plus souvent employé dans des instructions `switch` (voir plus haut).

Les instructions qui suivent illustrent les différences existant entre `break` et `continue` :

```
$compteur = 0;
while ($compteur < 5)
{ $compteur++;
  if ($compteur == 3 )
  { echo "break<br>";
    break;
  }
  echo "Fin de la boucle while : compteur=$compteur<br>";
}
echo "Sortie par le break<p>";
```

```
$compteur = 0;
while ($compteur < 5)
{ $compteur++;
  if ($compteur == 3)
  { echo "continue<br>";
    continue;
  }
  echo "Fin de la boucle while : compteur=$compteur<br>";
}
echo "Sortie après le continue<br>";
```

Ces deux groupes d'instructions réalisent deux boucles identiques, à ceci près que le premier met en œuvre un `break` alors que le second contient un `continue`. Voici ce qui est affiché :

```
Fin de la boucle while : compteur=1
Fin de la boucle while : compteur=2
break
Sortie par le break
```

```
Fin de la boucle while : compteur=1
Fin de la boucle while : compteur=2
continue
Fin de la boucle while : compteur=4
Fin de la boucle while : compteur=5
Sortie après le continue
```

La première boucle se termine par l'exécution du `break`, alors que la seconde continue après l'exécution du `continue`. Ce dernier ne fait que sauter l'affichage par l'instruction `echo` lorsque `$compteur` vaut 3, la remontée dans la boucle ayant lieu normalement.

Le `break` est une assurance contre les boucles infinies. Les instructions qui suivent peuvent arrêter une boucle à un point raisonnable :


```
$test_infini++;
if ($test_infini > 100 )
{ break;
}
```

Si vous êtes certain que votre boucle ne doit jamais être répétée plus de 100 fois, ces instructions l'arrêteront avant qu'elle ne devienne infinie. Adoptez n'importe quelle valeur qui vous semblera raisonnable pour la boucle que vous écrirez.

Les fonctions

Dans une application, on a souvent l'occasion d'exécuter la même tâche à plusieurs reprises. Par exemple, afficher le logo de l'entreprise à plusieurs endroits d'une même page Web ou dans différentes pages Web. Supposons qu'on utilise pour cela la séquence d'instructions suivantes :

```
echo '<hr width="50" align="left">','\n';
echo '<br>','\n';
echo '<hr width="50" align="left"><br>','\n';
```

On peut alors créer une fonction utilisateur contenant ces trois instructions et l'appeler `afficheLogo()`. Lorsque le programme aura besoin d'afficher le logo, il suffira alors d'écrire :

```
afficheLogo();
```

Remarquez les parenthèses qui suivent le nom de la fonction. C'est ce qui permet à PHP de savoir qu'il s'agit d'une fonction et non d'une variable.



Contrairement à ce qui se passe pour les variables, le nom d'une fonction **ne doit pas** commencer par un caractère dollar (\$).

Voici quelques-uns des avantages offerts par les fonctions :

- ✓ **Moins de frappe.** Vous n'avez pas à retaper toutes les instructions composant le corps de la fonction, mais seulement son nom.
- ✓ **Relecture plus facile.** On comprend plus facilement ce que signifie `afficheLogo()` que la suite des trois instructions qui composent cette fonction.

- ✓ **Moins d'erreurs.** Une fois que votre fonction est au point, vous ne risquerez plus de faire d'erreurs en la réécrivant chaque fois que vous en aurez besoin.
- ✓ **Plus grandes facilités de changement.** Si vous décidez de modifier la façon dont une tâche est accomplie, il vous suffit de faire ces modifications en un seul endroit sans avoir besoin de rechercher dans un ou plusieurs programmes chacune de ses utilisations. Si, par exemple, vous décidez de modifier le nom de l'image qui constitue le logo, il vous suffit de faire ce changement une seule fois, à l'intérieur de la fonction.

Pour créer une fonction, il faut la *déclarer*, ce qui se fait sous la forme générale suivante :

```
function nomFonction()
{
    ... bloc d'instructions;
    return;
}
```



Attention ! L'instruction s'écrit bien *function* (à l'anglaise, avec un "u") et non *fonction*.

Reprenant notre exemple de l'affichage du logo de l'entreprise, voici de quelle façon vous pouvez en faire une fonction :

```
function afficheLogo()
{ echo '<hr width="50" align="left">','\n';
  echo '<br>','\n';
  echo '<hr width="50" align="left"><br>','\n';
  return;
}
```

Une fois exécutée la dernière instruction du bloc qui constitue la fonction, le programme se poursuit à l'instruction du programme qui suit l'*appel* de la fonction.

On peut forcer un retour prématuré, par exemple à la suite de certaines conditions, au moyen de l'instruction `return`. Pour arrêter l'exécution lorsqu'une variable, \$âge, par exemple, est inférieure à 13, on pourrait écrire :

```
if ($âge < 13) return;
```

Si le retour *normal* est suffisant, il est inutile de placer un `return` comme dernière instruction de la fonction.

L'instruction `return` peut être utilisée pour renvoyer **un** résultat (un seul) au programme qui l'a appelée. Pour cela, on fait suivre le mot `return` de la valeur à renvoyer exprimée par une constante, une variable ou une expression. Nous en verrons une application plus loin dans la section "Passage de valeurs entre une fonction et le programme environnant".

Vous pouvez placer la déclaration des fonctions à n'importe quel endroit du programme, mais il est d'usage de les regrouper en tête, de façon à pouvoir, si besoin, les corriger plus facilement. Il est aussi possible de regrouper dans un fichier séparé toutes les fonctions qu'utilisera votre programme (voire toutes vos applications). Nous reviendrons sur ce point au Chapitre 10.



Contrairement à ce qui se passe pour les noms de variables, les noms de fonctions sont insensibles à la casse. `mafonction()` est identique à `MAFONCTION()` et à `maFonction()`.

L'exemple de fonction que nous venons de donner est simple (rudimentaire, même), car il n'utilise aucune variable et ne partage aucune information avec le programme proprement dit, se contentant d'accomplir exactement la même tâche à chaque appel. Nous allons voir qu'il est possible d'aller plus loin et d'échanger des informations entre une fonction et le programme qui l'utilise.

Les variables et les fonctions

Une fonction peut utiliser des variables *locales* auxquelles elle seule peut accéder et que le programme qui l'environne ne voit normalement pas. Considérons l'exemple suivant :

```
function maFonction()
{ $nom = "Dupont";
  $prénom = "Jules";
  $nomComplet = $prénom . " ." . $nom;
}

maFonction();
echo $nomComplet;
```

La seule chose qui s'affichera sera un avertissement signalant que la variable `$nomComplet` n'est pas définie. Etant créée à l'intérieur de la fonction, elle n'existe pas en dehors de celle-ci.

Si on veut que la communication puisse s'établir, les variables à partager doivent être déclarées dans la fonction au moyen de l'instruction `global` selon le modèle suivant :

```
function maFonction()
{ $nom = "Dupont";
  $prénom = "Jules";
  global $nomComplet;
  $nomComplet = $prénom." ".$nom;
}
```

```
maFonction();
echo $nomComplet;
```

Le programme affiche maintenant :

Jules Dupont

La déclaration `global` doit obligatoirement précéder l'initialisation de la variable concernée, faute de quoi elle n'aurait aucun effet. Dans l'exemple ci-dessus, si elle s'était trouvée après `$nomComplet = $prénom." ".$nom;`, rien n'aurait été affiché.

C'est la même chose dans l'autre sens, à savoir que, dans une fonction, vous ne pouvez pas "voir" les variables définies et utilisées par le programme environnant. Il faut, pour cela, les déclarer comme globales dans la fonction. Exemple :

```
$nom = "Dupont";
$prénom = "Jules";

function maFonction()
{ global $nom, $prénom, $nomComplet;
  $nomComplet = $prénom." ".$nom;
}
```

```
maFonction();
echo $nomComplet;
```

On affichera bien ainsi :

Jules Dupont



Notez que cette particularité permet d'avoir dans une fonction des variables de même nom que les variables du programme environnant, mais utilisées à d'autres fins et ne contenant pas les mêmes valeurs. Tant qu'elles n'ont pas été déclarées à l'intérieur de la fonction par `global`, bien sûr.

Passage de valeurs entre une fonction et le programme environnant

On peut échanger de l'information dans les deux sens entre des fonctions et le programme dans lequel elles se trouvent appelées. Dans l'exemple précédent, les nom et prénom qui composent le nom complet auraient pu être passés à la fonction sans recourir à la déclaration `global`.

Passer des valeurs à une fonction

Pour transmettre des valeurs à une fonction, on utilise une technique consistant à placer des noms de variables à l'intérieur des parenthèses suivant le nom de la fonction dans sa déclaration. On dit que ce sont les *arguments* de la fonction. L'appel se fera alors sous la forme :

```
maFonction(valeur1, valeur2...)
```

Bien entendu, la fonction doit pouvoir récupérer ces valeurs dans des variables. Elle sera donc déclarée de la façon suivante :

```
function maFonction($variable1, $variable2...)
{
    instructions;
    return;
}
```

A titre d'exemple, la fonction présentée dans la séquence suivante détermine les frais de port d'un envoi selon son poids :

```
function FraisDePort($poids)
{ if ($poids < 100)
    return 0;
  elseif ($poids < 500)
    return 2;
  else
    return 5;
```



```

}

$p = 350;
echo "Frais de port pour un envoi de $p grammes : ",
    FraisDePort($p), " €#8364;<br>";

```

On considère trois cas : si le poids est inférieur à 100 grammes, le port est gratuit. S'il est supérieur ou égal à 500 grammes, les frais de port s'élèvent à 5 € Entre les deux, ils sont de 2 €

La variable fictive `$poids` figurant dans les parenthèses de la déclaration de la fonction reçoit la valeur passée entre les parenthèses de l'appel de la fonction et l'utilise pour ses tests. Elle renvoie le montant des frais de port au moyen de l'instruction `return`.

Vous pouvez passer autant d'arguments que vous le souhaitez à une fonction. Pas plus, cependant, que vous n'en avez fait figurer dans la déclaration de la fonction. Ces arguments peuvent être des constantes, des variables ou des expressions. Exemples :

```

FraisDePort(350);
FraisDePort($colis);
FraisDePort($colis * $remise);

```

On peut passer des valeurs dans un tableau. En voici un exemple :

```

function ajouter($nombres)
{ for ($i=0, $s=0; $i<sizeof($nombres); $i++)
    $s = $s + $nombres[$i];
  return $s;
}

```

```
$tableau = array(1,3, 42, 67, 101);
```

```
echo ajouter($tableau);
```

La variable `$nombres` de la fonction reçoit l'adresse du tableau `$tableau`. Autrement dit, tout se passe comme si on avait substitué à l'étiquette `$nombres` l'étiquette `$tableau`. La fonction peut donc facilement faire la somme `$s` des nombres contenus dans ce tableau et la renvoyer au programme qui l'a appelée par l'instruction `return $s;`.

Les valeurs sont passées par position. Autrement dit, la première valeur de la liste est récupérée par le premier argument de la fonction, la seconde valeur par le second argument, et ainsi de suite. Si vos valeurs ne sont pas placées dans le bon ordre, la fonction ne



pourra pas les traiter correctement. Modifions notre exemple d'envoi postal pour y ajouter comme second argument un nom de pays :

```
<?PHP
function FraisDePort($poids, $destination)
{ if ($destination == "France")
  { if ($poids < 100)
    { return 0;
    }
    elseif ($poids < 500)
    { return 2;
    }
    else
    { return 5;
    }
  }
  else
  { if ($poids < 100)
    { return 10;
    }
    elseif ($poids < 500)
    { return 20;
    }
    else
    { return 50;
    }
  }
}

$p = 350;
$destination="Pérou";
echo "Frais de port pour un envoi de $p grammes ($destination): ",
    FraisDePort($p, $destination), "&#8364;<br>";
?>
```

Vous allez obtenir l'affichage suivant :

Frais de port pour un envoi de 350 grammes (Pérou): 20 €

Imaginons alors que, pris d'étourderie, vous inversez les deux arguments :

```
FraisDePort($destination, $p)
```

Vous pourriez parfaitement obtenir un message d'erreur. Mais ici, la réponse sera :

Frais de port pour un envoi de 350 grammes (Pérou): 10 €

En effet, la fonction "pense" que la destination vaut 350. Le premier test est donc FAUX. Elle compare alors le poids (qui vaut en réalité *Pérou*) avec le nombre 100. La chaîne est évaluée dans cette comparaison comme ayant une valeur numérique nulle, et le test est donc bien vérifié. La fonction retourne donc 10.

Si le nombre d'arguments figurant dans l'appel est inférieur à celui qu'attend la fonction, les arguments manquants sont considérés par défaut comme égaux à NULL. Il s'agit là d'une constante PHP de nature particulière, différente de 0. En outre, un message d'erreur (avertissement) sera affiché. En supposant qu'il s'agisse de l'appel de la fonction `maFonction()` et qu'il manque le second argument, on verra s'afficher : "Missing argument 2 for maFonction() ...".

S'il y en a trop, les arguments superflus seront ignorés sans qu'aucun message d'erreur ne soit affiché.

Il est aussi possible de définir des valeurs par défaut pour les arguments qui ne sont pas passés. La déclaration se fait alors comme dans l'exemple suivant :

```
function ajoute_2_nombres($nombre1=1,$nombre2=1)
{
    $total = $nombre1 + $nombre2;
    return $total;
}
```

Si un argument est absent, la fonction se servira de la valeur par défaut. Sinon, elle effectuera le calcul avec la valeur qui lui a été transmise. Par exemple :

```
echo "Total : ";
echo ajoute_2_nombres(2,2);
echo "<br>";
echo "Total : ";
echo ajoute_2_nombres(2);
echo "<br>";
echo "Total : ";
echo ajoute_2_nombres();
```

affichera :

```
Total : 4
Total : 3
Total : 2
```

Renvoi de valeurs par une fonction

Nous avons vu plus haut que, pour renvoyer une seule valeur, il suffisait à une fonction de faire figurer cette valeur à la suite de l'instruction `return`. Exemple :

```
return $somme;
```

A la suite du `return`, on peut écrire une constante, une variable ou une expression. Reprenons l'exemple du calcul des frais de port et modifions-le ainsi :

```
function TotalAPayer($poids, $prix)
{ if ($poids < 100)
    $port = 0;
  elseif ($poids < 500)
    $port = 2;
  else
    $port = 5;
  return $port + $prix;
}

$p = 350;
$q = 30;
echo "Montant à facturer : ", TotalAPayer($p, $q), " &#8364;<br>";
```

Comme on le voit, le programme principal peut incorporer directement l'appel de la fonction `TotalAPayer()` dans une instruction `echo` en même temps que des constantes.



On peut avoir plusieurs instructions `return` à l'intérieur d'une fonction, par exemple dans des instructions conditionnelles ou des boucles. Nous en avons un exemple plus haut.



Une instruction `return` ne peut renvoyer qu'une seule valeur. Mais il est possible de contourner cette restriction : il suffit que cette valeur soit un tableau !

Chapitre 8

Mouvements de données

Dans ce chapitre :

- Connexion à la base de données.
- Extraction d'informations de la base de données.
- Mise en œuvre de formulaires HTML avec PHP.
- Traitement des informations saisies par les utilisateurs dans un formulaire.
- Stockage de données dans une base de données.
- Fonctions de transfert de données avec une base de données.
- Transfert d'informations par fichier.

Dans les précédents chapitres de ce livre, j'ai décrit les outils dont vous aurez besoin pour construire une application de base de données sur le Web. Au Chapitre 4 : comment écrire une requête MySQL ; aux Chapitres 6 et 7 : comment utiliser les briques de base de PHP pour construire votre application. Dans le présent chapitre, je vais vous montrer comment utiliser ces outils pour accomplir les tâches spécifiques que toute application de base de données sur le Web doit effectuer.

Fonctions PHP/MySQL

Le site Web de MySQL propose à l'heure où nous mettons sous presse les versions 4.1, 5.0 et 5.1. La version 5.0 étant la version stable que la plupart vont installer pour exploitation. Les versions 4.0 et 3.23 servent encore sur quelques sites. A partir de PHP 5, vous devez choisir entre deux familles de fonctions pour communiquer avec MySQL : une famille moderne (commençant par **mysqli**) utilisable avec les versions à partir de la 4.1 et une famille de compatibilité (commençant par **mysql**, donc sans **i**) pour MySQL 4.0 et versions antérieures.

Si vous exploitez PHP 5 ou 6 sous Windows, vous devez activer une extension appropriée à la famille désirée, comme indiqué dans l'Annexe B. Sous Linux, Unix et MacOS, vous modifiez une option de configuration, comme expliqué dans l'étape 10 des instructions d'installation dans l'Annexe B.

Avec PHP 4, la famille **mysqli** n'est pas utilisable. Vous devez vous rabattre sur la famille **mysql**, même avec une version récente de MySQL. Certaines fonctions récentes de MySQL ne sont dans ce cas pas exploitables. La famille **mysql** est activée d'office dans PHP 4, ce qui est logique.

Dans ce livre, s'agissant d'une mise à jour, nous avons adopté MySQL 5.0 et la famille **mysqli**. Voici le format générique des fonctions PHP d'accès à MySQL 5.0 :

```
mysqli_nomfonction(valeur,valeur,...);
```



La lettre **i** ajoutée en fin de radical signifie *improved* (amélioré). Le second membre du nom de la fonction est spécifique. C'est en général un verbe d'action. La fonction attend un ou plusieurs paramètres. Ce nombre de paramètres a changé entre les deux générations de fonctions. Voici la syntaxe de deux fonctions utilisées sans cesse dans les exemples de ce chapitre :

```
mysqli_connect(informations de connexion);
mysqli_query($cxn,"Instruction SQL");
```

Avec PHP 4 ou pour un SGBD MySQL 4.0 ou antérieur, les fonctions sont celles-ci :

```
mysql_connect(informations de connexion);
mysql_query("Instruction SQL");
```

Si vous désirez évoluer vers la nouvelle famille de fonctions, vous devrez retoucher les appels de fonction en vous basant sur les comparaisons que propose le Tableau 8.1.

Remarque : Dans la plupart des exemples, nous en avons profité pour remplacer les appels à `mysql_fetch_array()` par des appels à `mysqli_fetch_assoc()`.

Tableau 8.1 : Syntaxe comparée des fonctions mysql et mysqli.

Fonction mysqli	Ancienne fonction mysql
<code>mysqli_connect(\$host,\$user,\$passwd,\$nombase)</code>	<code>mysql_connect(\$host,\$user,\$passwd)</code> puis <code>mysql_select_db(\$nombase)</code>
<code>mysqli_errno(\$cxn)</code>	<code>mysql_errno()</code> ou <code>mysql_errno(\$cxn)</code>
<code>mysqli_error(\$cxn)</code>	<code>mysql_error()</code> ou <code>mysql_error(\$cxn)</code>
<code>mysqli_fetch_array(\$result)</code>	<code>mysql_fetch_array(\$result)</code>
<code>mysqli_fetch_assoc(\$result)</code>	<code>mysql_fetch_assoc(\$result)</code>
<code>mysqli_fetch_row(\$result)</code>	<code>mysql_fetch_row(\$result)</code>
<code>mysqli_insert_id(\$cxn)</code>	<code>mysql_insert_id(\$cxn)</code>
<code>mysqli_num_rows(\$result)</code>	<code>mysql_num_rows(\$result)</code>
<code>mysqli_query(\$cxn,\$sql)</code> <code>mysql_query(\$sql,\$cxn)</code>	<code>mysql_query(\$sql)</code> ou <code>mysql_query(\$sql,\$cxn)</code>
<code>mysqli_select_db(\$cxn,\$dbname)</code>	<code>mysql_select_db(\$dbname)</code>
<code>mysqli_real_escape_string(\$cxn,\$data)</code>	<code>mysql_real_escape_string(\$data)</code>

Etablissement de la connexion

La première chose à faire avant tout transfert de données, est de se connecter à la base de données qu'on veut utiliser. Celle-ci peut se trouver sur le même ordinateur que le programme PHP ou sur une machine différente. Vous n'avez pas besoin de connaître les détails de cette connexion, PHP se charge de tout. Tout ce que vous devez savoir est le nom et l'adresse de la base de données. Représentez-vous la connexion avec une base de données comme s'il s'agissait d'un téléphone : les mots vont d'un interlocuteur à l'autre, mais peu vous importe comment. Il suffit que vous connaissiez le numéro de téléphone de votre interlocuteur.

Une fois que vous êtes connecté à la base de données, vous envoyez une série de requêtes au moyen des fonctions PHP prévues à cet effet. Vous pouvez en envoyer autant que vous le voulez. La connexion reste établie tant que vous ne la refermez pas explicitement ou que le programme se termine. Avec le téléphone, la connexion perdure jusqu'au moment où vous raccrochez votre combiné. C'est pareil.

Connexion au serveur MySQL

Pour communiquer avec une base de données, vous devez tout d'abord vous connecter au serveur MySQL. Pour cela, vous avez besoin de connaître le nom de l'ordinateur où se trouve la base de données, le nom de votre compte MySQL ainsi que le mot de passe correspondant à ce compte. Vous appelez alors la fonction PHP `mysqli_connect()` de la façon suivante :

```
$connexion = mysqli_connect(adresse, compte, mpasse, nombase)
or die ("message");
```

avec :

- ✓ **adresse** : Le nom de l'ordinateur sur lequel se trouve la base de données. Par exemple : `base.monserveur.com`. Sur un serveur personnel (et donc avec EasyPHP), ce serait tout simplement `localhost`. Dans ce cas, vous pouvez même laisser ce champ vide (`""`).
- ✓ **compte** : Le nom de votre compte MySQL. Revoyez à ce sujet le Chapitre 5. Si ce champ est vide (`""`), cela signifie que n'importe quel compte sera considéré comme valide. Pour des raisons évidentes de sécurité, ce n'est pas une bonne idée !
- ✓ **mpasse** : Le mot de passe attaché à ce compte. Si le compte utilisé n'exige pas de mot de passe, laissez ce champ vide (`""`).
- ✓ **nombase** : Le nom de la base de données avec laquelle vous avez besoin de communiquer (paramètre facultatif). Vous pouvez sélectionner la base plus tard avec une autre commande et changer de base active à tout moment.



Avec l'ancienne famille de fonctions `mysql`, vous ne pouviez pas choisir la base lors de la connexion et deviez appeler `mysql_select_db` dans un deuxième temps.

- ✓ **message** : Le message à afficher si la connexion ne peut pas être établie (soit parce que vous n'avez pas fourni les bonnes informations, soit tout bêtement parce que le serveur n'est pas accessible). Pendant la phase de mise au point, vous avez intérêt à prévoir pour ce message un texte explicite du genre : "Connexion au serveur impossible." Plus tard, au moment de l'exploitation de la base de données par vos clients, il faudra être moins technique et dire, par exemple : "Le catalogue de

l'animalerie n'est pas accessible pour le moment. Pouvez-vous réessayer plus tard ?"



Le paramètre `adresse` peut, dans certains cas, être complété par un numéro de port. Presque toujours, ce numéro sera 3306. C'est lui qui sera pris par défaut. Mais, à de rares occasions, l'administrateur de MySQL peut attribuer un numéro différent. Dans ce cas, il est nécessaire de spécifier ce numéro non standard sous la forme, par exemple, de `localhost:8808`.

Lorsque vous appelez la fonction `mysqli_connect()`, celle-ci tente d'établir la connexion avec la base de données en utilisant les paramètres que vous lui avez indiqués. Si cette tentative échoue, le message d'erreur est affiché et le programme s'arrête.

L'exemple qui suit montre comment se connecter à une base de données située sur un serveur personnel :

```
$connexion = mysqli_connect("localhost", "catalogue","", "AniCata")
or die ("Connexion au serveur impossible.");
```



Pour des raisons de sécurité (et de maintenance), il est préférable de placer les paramètres de connexion dans des variables et de faire figurer celles-ci dans l'appel à `mysqli_connect()` de la façon suivante :

```
$hôte = "localhost";
$utilisateur = "catalogue";
$mPasse = "";
$nombase = "AniCata";

$connexion = mysqli_connect($hôte, $utilisateur, $mPasse, $nombase)
or die ("Connexion au serveur impossible.");
```

En fait, si vous voulez accroître la sécurité de cette connexion, vous pouvez placer les trois informations dans un fichier séparé de façon qu'elles n'apparaissent pas en clair dans le programme. Vous en saurez plus sur ce sujet au Chapitre 10.

La variable `$connexion` identifie la connexion établie. Vous pouvez ouvrir plusieurs connexions en même temps, en conservant les informations relatives à chacune dans une variable de nom différent. Une connexion reste ouverte tant que vous ne la refermez pas explicitement ou que l'exécution du programme se poursuit. Pour fermer une connexion, vous appelez la fonction `mysqli_close()` en

lui passant en argument le nom reçu au moment de la connexion initiale. Par exemple :

```
mysqli_close($connexion);
```



Traitement des erreurs MySQL

Lorsqu'une des fonctions servant à dialoguer avec MySQL ne se termine pas normalement, MySQL crée un message d'erreur contenant des informations sur l'erreur qui s'est produite. Cependant, ce message n'est pas envoyé directement au navigateur (à moins que le programme ne le fasse de façon délibérée). Il y a trois façons habituelles d'appeler une fonction MySQL :

- ✓ Appel de la fonction sans traitement d'erreur. La fonction est appelée toute seule, sans que son nom soit suivi de l'appel à `die()`. Exemple :

```
$connexion = mysqli_connect($hôte, $utilisateur, $mpasse, $nombase);
```

Si son exécution échoue (par exemple, le mot de passe est incorrect), la connexion n'est pas établie, mais les instructions qui suivent cet appel dans le programme s'exécutent normalement (enfin, tant qu'elles ne font pas appel à une base de données). Dans la plupart des cas, cette poursuite des opérations est inutile, car on peut se douter que, sans accès à la base de données, le programme ne pourra pas donner de résultats corrects.

- ✓ Appel de la fonction suivi d'un appel à `die()`. L'appel à `die()` indique le texte du message à afficher en cas d'erreur. Exemple :

```
$connexion = mysqli_connect($hôte, $utilisateur, $mpasse, $nombase)
or die ("Connexion au serveur impossible.");
```

Non seulement `die()` affiche le message, mais il met fin à l'exécution du programme, ce qui est une solution de sagesse. C'est à vous de définir les termes du message.

- ✓ Appel de la fonction à l'intérieur d'une structure `if`. Ce `if` va exploiter la valeur renvoyée par la fonction. En cas d'erreur, c'est `FALSE` (FAUX) qui est renvoyé. En cas de réussite, c'est une valeur positive, donc l'équivalent de `TRUE` (VRAI). Exemple :

```
if (! $connexion = mysqli_connect($hôte, $utilisateur, $mpasse, $nombase))
{
    $message = mysqli_error($connexion);
    echo "$message<br>";
    exit;
}
```


En cas d'erreur, ! \$connexion vaut TRUE, aussi le bloc d'instructions qui suit est-il exécuté. Il commence par récupérer le texte du message d'erreur créé par MySQL, en appelant la fonction `mysqli_error()`, puis il affiche ce message. Il ne reste plus qu'à mettre fin au programme, ce qu'on peut faire, au choix, par un appel à `die()` ou à `exit`. Remarquez en passant le point d'exclamation devant la condition. Il s'agit d'une *négation*: le bloc est exécuté uniquement si la fonction `mysqli_connect` ne réussit pas à établir la connexion.

Le niveau du traitement d'erreur que vous devez prévoir dans votre programme dépend de ce que vous craignez de voir arriver au cours de son exécution. Lorsque vous développez une application, vous savez que vous rencontrerez bien plus de problèmes que lorsque vous l'exploitez. Aussi est-il prudent d'afficher le plus de détails possible sur l'erreur qui est survenue. Nous avons déjà évoqué ce problème au début de la section "Connexion au serveur MySQL", à propos du paramètre message. En particulier, pensez à rendre le message aussi explicite que possible afin de localiser rapidement la ligne incriminée. Supposons par exemple que vous utilisiez le compte `root` pour accéder à la base de données et que vous fassiez une petite faute de frappe :

```
$hôte = "localhost";
$utilisateur = "rot";
$mpasse = "";
$nombase = "truc";
if (!$connexion = mysqli_connect($hôte,$utilisateur,$mpasse, $nombase))
{
    $message = mysqli_error($connexion);
    echo "$message<br>";
    die();
}
```

Puisque vous avez oublié un *o* dans `root`, vous allez voir un message d'erreur comme celui-ci :

```
Access denied for user: 'rot@localhost' (Using password: NO)
```

Voilà qui devrait vous aider à trouver l'instruction fautive : il suffit de détecter l'erreur dans le nom de l'utilisateur. Bien entendu, une fois votre programme en service, vos utilisateurs n'ont pas à recevoir un message d'erreur comme celui-ci (votre réputation serait ruinée). Vous modifierez alors le code pour appliquer la seconde méthode afin d'afficher un message personnalisé, comme "Le catalogue en ligne n'est pas disponible actuellement. Veuillez réessayer un peu plus tard."

Sélection de la base de données

Si vous n'avez pas sélectionné la base de données au moment de la connexion avec le serveur MySQL, une fois cette connexion établie, il reste à sélectionner la base à utiliser. Pour cela, il faut appeler la fonction `mysqli_select_db()` de la façon suivante :

```
$db = mysqli_select_db(nomConnexion , nomBase)
    or die (message);
```

avec :

- ✓ **nomConnexion** : La variable qui contient le nom de la connexion à MySQL. Si cet argument n'est pas spécifié, PHP utilise la dernière connexion ouverte.
- ✓ **nomBase** : Nom de la base de données.
- ✓ **message** : Message qui sera affiché si la sélection de la base de données est impossible.



Notez que l'ordre des deux paramètres de cette fonction a été inversé par rapport à `mysql_select_db`, le nom de la connexion devant maintenant être fourni en premier.

Par exemple, vous pouvez sélectionner la base de données des animaux de compagnie, `AniCata`, au moyen de l'instruction suivante :

```
$db = mysqli_select_db($connexion, "AniCata")
    or die ("La base de données ne peut pas être sélectionnée");
```

En cas d'échec, le message est affiché et le programme s'arrête.



Pour des raisons de sécurité, il est préférable de placer le nom de la base de données dans une variable et de faire figurer celle-ci dans l'appel à `mysqli_connect()` de la façon suivante :

```
$base = "AniCata";
$db = mysqli_select_db($connexion, $base)
    or die ("La base de données ne peut pas être sélectionnée");
```

En fait, si vous voulez accroître la sécurité de cette connexion, vous pouvez placer cette information dans un fichier séparé (et dans un emplacement caché), de façon qu'elle n'apparaisse pas en clair dans le programme. Vous trouverez des informations sur ce sujet au Chapitre 10.

La base de données reste sélectionnée tant que vous n'en choisissez pas explicitement une autre par un appel à `mysqli_select_db()`.

Envoi de requêtes MySQL

Une fois que vous vous êtes connecté au serveur MySQL et que vous avez sélectionné votre base de données, vous êtes prêt à envoyer des requêtes MySQL afin de ranger, retrouver ou mettre à jour des données. (Ce qui concerne le langage SQL a été traité au Chapitre 4.)

Pour envoyer une requête SQL à un serveur, placez-la dans une variable et utilisez la fonction `mysqli_query()` de la façon indiquée dans l'exemple suivant :

```
$req = "SELECT * FROM Animal";
$resultat = mysqli_query($connexion, $req)
           or die ("La requête ne peut pas être exécutée");
```

La requête est exécutée sur la base de données couramment sélectionnée pour la connexion ouverte en dernier.

La variable `$resultat` contient les informations renvoyées par le serveur MySQL à la suite de l'exécution de la requête. La nature de cette information dépend du type de la requête envoyée :

- ✓ **Pour les requêtes qui ne renvoient pas d'informations :**
`$resultat` vaut 1 (c'est-à-dire TRUE) si la requête a été exécutée avec succès. Sinon, il vaut FALSE, ce qui permet d'exécuter l'instruction `die()` qui suit éventuellement l'appel. Certaines requêtes, comme INSERT ou UPDATE, ne renvoient pas de données.
- ✓ **Pour les requêtes qui renvoient des informations :**
`$resultat` contient un pointeur vers les informations renvoyées et non pas vers les données elles-mêmes. Parmi les requêtes qui renvoient quelque chose, on peut citer SELECT et SHOW.



A partir de la version MySQL 4.1, si vous utilisez PHP 5 et la famille **mysqli**, vous pouvez émettre plusieurs requêtes en parallèle vers le serveur en les séparant par des signes point-virgule. Cela est possible grâce à la fonction `mysqli_multiple_query`. Le danger est que votre programme perd en sécurité. N'abusez pas de cette possibilité.



Le choix entre guillemets et apostrophes dans l'écriture d'une requête sous la forme `$variable` = chaîne de caractères est délicat. En effet, il faut prendre en compte deux niveaux : l'encadrement de la chaîne de caractères représentant la requête et les délimiteurs des paramètres de la requête en langage SQL. Pour être sûr de ne pas vous tromper, tenez compte des règles suivantes :

- ✓ Employez des guillemets au début et à la fin de la chaîne de caractères définissant la requête.
- ✓ Employez des apostrophes pour encadrer les noms de variables.
- ✓ Employez des apostrophes pour encadrer les valeurs de littéraux.

L'exemple qui suit devrait contribuer à clarifier ces explications :

```
$req = "SELECT prénom from Membre";
$req = "SELECT prénom from Membre WHERE NOM = 'Dupont'";
$req = "UPDATE Membre SET nom = '$nomClient'";
```



Une requête ne doit pas se terminer par un point-virgule. Le seul qu'on puisse trouver dans l'affectation de la requête à la variable `$req` est celui qui termine la chaîne de caractères, autrement dit celui qui termine l'instruction d'affectation de PHP. (Voir les exemples ci-dessus.)

Extraction d'informations d'une base de données

L'extraction d'informations d'une base de données est une opération très courante dans une application de base de données sur le Web. Voici deux de ses principales utilisations :

- ✓ **Se servir de ces informations dans une instruction conditionnelle.** Par exemple, selon le département où réside un client, vous allez lui envoyer tel ou tel message.
- ✓ **Afficher ces informations dans une page Web.** Par exemple, afficher la description d'un produit du catalogue.

Pour effectuer ces manipulations, vous devez commencer par ranger l'information à traiter à l'intérieur d'une ou de plusieurs variables. Pour cela, deux étapes sont nécessaires :

1. **Construire une requête SELECT et l'envoyer à la base de données.** Quand la requête est exécutée, les données sont rangées dans un emplacement provisoire.
2. **Transférer les données depuis cet emplacement jusque dans les variables de votre programme.**

Envoi d'une requête SELECT

Les requêtes SELECT ont été décrites au Chapitre 4. Construisez la requête en fonction de ce que vous voulez extraire et placez la chaîne de caractères qui la représente dans une variable. A titre d'exemple, les deux instructions qui suivent vous montrent comment sélectionner des informations dans la table `Animal` de la base `AniCata` :

```
$rq = "SELECT *FROM Animal";
$res = mysqli_query($connexion, $req)
    or die ("Exécution de la requête impossible ");
```

La fonction renvoie dans la variable `$res` un pointeur vers l'emplacement où se trouvent les informations sélectionnées. On peut se représenter celles-ci comme un tableau formé de lignes de données et de colonnes de champs.

La fonction renvoie un identifiant contenant les informations qui permettront d'accéder à l'emplacement temporaire des données. Dans les instructions précédentes, cet identifiant était placé dans la variable `$result`. En cas d'échec de la fonction (si par exemple la requête est erronée), `$result` contiendra `FALSE`.

Reste maintenant à transférer ces informations dans des variables accessibles au programme.

Transfert et utilisation des informations

La lecture de données d'un endroit temporaire se base sur l'une des fonctions `mysqli_fetch_assoc` ou `mysqli_fetch_row`. La première est la plus usitée ; elle renvoie les données dans un tableau associatif. L'autre, `mysqli_fetch_row`, les renvoie dans un tableau numérique. Dans certains cas, vous aurez besoin de rapatrier dans un tableau associatif et un tableau numérique, auquel cas vous utiliserez `mysqli_fetch_array`.

Toutes ces fonctions lisent une ligne (d'un enregistrement) de données à la fois. Le tableau contenant les données temporaires peut contenir plusieurs lignes de données. Si votre requête `SELECT` sélectionne plusieurs lignes, vous les récupérez une par une dans une boucle appelant `mysqli_fetch_assoc` ou `mysqli_fetch_row`.

Lire une ligne de données

Voici la syntaxe générique de la fonction PHP `mysqli_fetch_assoc` :

```
$ligne = mysqli_fetch_assoc($resptr);
```

L'instruction lit une ligne de données depuis la table de la base de données vers l'emplacement temporaire, puis copie les données dans la variable de type tableau nommée `$ligne`. `$resptr` est un pointeur qui désigne l'emplacement temporaire des données résultantes.



N.d.T. : La fonction `mysql_fetch_array()` a été presque systématiquement remplacée par `mysqli_fetch_assoc()` dans cette nouvelle édition du livre. La principale différence est que la nouvelle fonction ne demande qu'un paramètre. L'ancien second paramètre `type_de_tableau` de `mysql_fetch_array()` ne sert plus. Rappelons qu'il était destiné à préciser le type de tableau et contenait l'un des trois identifiants `MYSQL_NUM`, `MYSQL_ASSOC` ou `MYSQL_BOTH`.

Dans certains cas, le résultat de la requête à la base de données ne procure qu'une seule ligne ; dans d'autres, il peut y en avoir plusieurs. Par exemple, pour contrôler le mot de passe fourni par un utilisateur, il vous suffira d'extraire une seule variable de la base au moyen des instructions suivantes :

```
$passe = "secret"; // mot de passe saisi par
                // l'utilisateur dans un formulaire
$req = "SELECT mPasse FROM Membre WHERE nom='jDupont'";
$res = mysqli_query($connexion, $req)
      or die ("Exécution de la requête impossible");
$ligne = mysqli_fetch_assoc($res);
if ($passe == $ligne['mPasse'])
{ echo "Login correct<br>";
  ... instructions affichant les pages à accès réservé ...
}
else
{ echo "Mot de passe incorrect<br>";
  ... instructions invitant l'utilisateur à essayer
```

```

    un autre mot de passe ...
}

```

Dans ces instructions, notez les points suivants :

- ✓ La requête `SELECT` demande d'extraire un seul champ (`mPasse`) à partir d'une ligne de la base de données, celle repérée par `jDupont`.
- ✓ La fonction `mysqli_fetch_assoc()` renvoie un tableau dont le nom est `$ligne` et dont les intitulés de colonnes sont les clés.
- ✓ L'instruction `if` compare le mot de passe fourni par l'utilisateur (`$passe`) à celui qui vient d'être extrait de la base (`$ligne['mPasse']`). L'opérateur utilisé (`==`) contrôle l'identité des deux renseignements.
- ✓ Si le résultat de cette comparaison est `TRUE`, les pages Web à accès réservé sont affichées (celles de `MembresSeuls`).
- ✓ Si le résultat de cette comparaison n'est pas `TRUE`, les pages à accès réservé ne sont pas affichées. A leur place, un message demande à l'utilisateur d'essayer un autre mot de passe.

 Il existe un raccourci pratique pour utiliser les variables renvoyées par la fonction `mysqli_fetch_assoc()`. C'est la fonction `extract()` qui décompose le tableau en variables ayant le même nom que la champ. Nous allons réécrire la séquence précédente en utilisant cette fonction (les deux lignes modifiées sont imprimées en gras) :

```

$passe = "secret"; // mot de passe saisi par
                // l'utilisateur dans un formulaire
$req = "SELECT mPasse FROM Membre WHERE nom='jDupont'";
$res = mysqli_query($connexion, $req)
    or die ("Exécution de la requête impossible");
$ligne = mysqli_fetch_assoc($res);
extract($ligne);
if ($passe == $mPasse )
{ echo "Login correct<br>";
  ... instructions affichant les pages à accès réservé ...
}
else
{ echo "Mot de passe incorrect<br>";
  ... instructions invitant l'utilisateur à essayer
    un autre mot de passe ...
}

```

Lecture d'une série de lignes dans une boucle

Si l'exécution de votre requête vous a renvoyé plusieurs lignes de données, une boucle vous permettra de les extraire l'une après l'autre afin de les traiter. Cette boucle peut être de type `for` ou `while`, à votre choix (reportez-vous au Chapitre 7 pour plus d'informations). En voici des exemples :

```
while ($ligne = mysqli_fetch_assoc($res))
{
    ... bloc d'instructions
}
```

A chaque appel, la fonction renvoie une ligne de résultat. La boucle se termine lorsqu'il n'y a plus rien à renvoyer. Si vous vous contentez d'afficher un par un les résultats obtenus, vous pouvez écrire :

```
while ($ligne = mysqli_fetch_assoc($res))
{
    extract($ligne);
    echo "$type : $animal<br>";
}
```

Pour bien comprendre le fonctionnement de cette séquence, regardons comment se présentent les informations dans le catalogue des animaux. Supposons qu'il s'y trouve une table appelée `Animal` qui possède quatre colonnes : `animal`, `type`, `description` et `prix` selon le schéma du Tableau 8.1.

Tableau 8.2 : Exemple des données présentes dans la table `Animal`.

animal	type	description	prix
Licorne	Cheval	Corne en spirale sur le front	10 000
Pégase	Cheval	Cheval ailé	15 000
Poney	Cheval	Très petit cheval	500
Dragon d'Asie	Dragon	Corps de serpent	30 000
Dragon médiéval	Dragon	Corps de lézard	30 000
Lion	Chat	Grande taille, crinière	2 000
Griffon	Chat	Corps de lion, tête d'oiseau, ailes	25 000

Le programme `affAnimal.php` que présente le Listing 8.1 sélectionne tous les animaux ayant le type "cheval" et affiche les informations correspondantes dans un tableau HTML. La variable `$typeAnimal` contient le type qui a été saisi par le visiteur dans un formulaire (non montré ici). La Figure 8.1 montre ce qu'on voit sur l'écran du navigateur.

Listing 8.1 : Affichage d'articles du catalogue des animaux.

```
<?php
/* Program: affAnimal.php (petDisplay.php)
 * Desc:   Affiche tous les animaux d'une catégorie.
 */
?>
<html>
<head>
<title>Catalogue des animaux</title>
</head>
<body>
<?php
    $user="catalog";
    $host="localhost";
    $password="";
    $database = "AniCata";

    $connexion = mysqli_connect($host,$user,$password)
        or die ("Connexion au serveur impossible");
    $db = mysqli_select_db($connexion, $database)
        or die ("Sélection de la base impossible");

    $typeAnimal = "cheval"; // ce mot a été saisi par
                           // l'utilisateur dans un formulaire
    $rq = "SELECT * FROM Animal WHERE animalType ='$typeAnimal'";
    $result = mysqli_query($connexion, $rq)
        or die ("Exécution de la requête impossible");

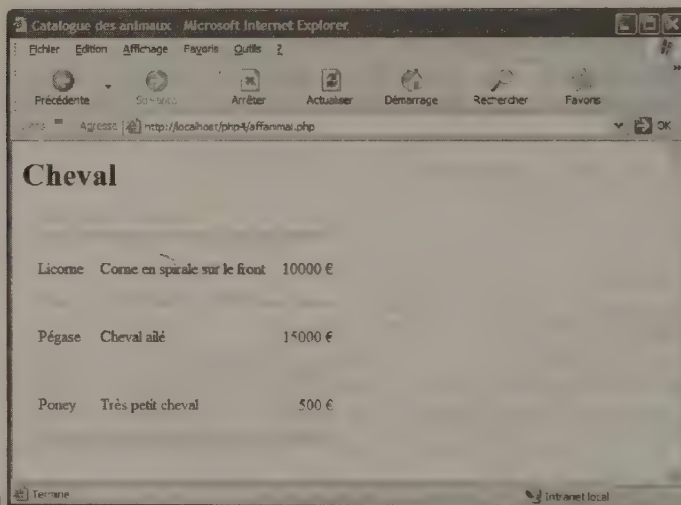
    /* Afficher les résultats dans un tableau */
    $typeAnimal = ucfirst($typeAnimal);
    echo "<h1>$typeAnimal</h1>";
    echo "<table cellpadding='15'>";
    echo "<tr><td colspan='3'><hr></td></tr>";
    while ($ligne = mysqli_fetch_assoc($result))
    { extract($ligne);
      $prix_f = number_format($animalPrix, 2);
      echo "<tr>\n
          <td>$animalNom</td>\n
          <td>$animalPrix</td>\n
          <td>$animalType</td>\n
          </tr>";
    }
    echo "</table>";
}
```

```

        <td>$animalDesc</td>\n
        <td align='right'>$prix_f &#8364;</td>\n
    </tr>\n";
    echo "<tr><td colspan='3'><hr></td></tr>\n";
}
echo "</table>\n";
?>
</body>
</html>

```

Figure 8.1 :
Extraction
des animaux
ayant le type
"cheval" de la
base de
données
AniCata.



Cheval		
Licorne	Corne en spirale sur le front	10000 €
Pégase	Cheval ailé	15000 €
Poney	Très petit cheval	500 €



La fonction `ucfirst()` sert à convertir si nécessaire le premier caractère du type de l'animal en majuscules.

La boucle `while` balaie toutes les lignes des résultats obtenus pour en extraire les informations à afficher dans le tableau HTML. Dans certains cas, il sera plus pratique d'utiliser une boucle `for` à la place d'une boucle `while`. Par exemple, si vous devez utiliser une variable numérique dans la boucle, `for` sera plus indiqué.

Dans ce but, vous devez savoir combien de lignes de données ont été extraites de la base de données à la suite de la requête `SELECT`. Vous obtiendrez cette information par un appel à la fonction `mysqli_num_rows()` :

```
$nbLignes = mysqli_num_rows($résultat);
```


La variable `$nbLignes` contient le nombre de lignes à récupérer. Vous pouvez alors écrire :

```
for ($i=0; $i<$nbLignes; $i++)
{ $ligne = mysqli_fetch_assoc($resultat);
  ... bloc d'instructions ...
}
```

Nous allons reprendre le programme du Listing 8.1 en le modifiant pour qu'il affiche, en tête de chaque ligne, le numéro de l'article. Le Listing 8.2 vous présente cette version modifiée, illustrée par la copie d'écran de la Figure 8.2.

Listing 8.2 : Affichage numéroté d'articles du catalogue des animaux.

```
<?php
/* Program: affAnimalFor.php (petDescripFor.php)
 * Desc:   Affiche la liste numérotée de tous les animaux d'une catégorie.
 */
?>
<html>
<head>
<title>Catalogue des animaux</title>
</head>
<body>
<?php
    $user="catalog";
    $host="localhost";
    $password="";
    $database = "AniCata";

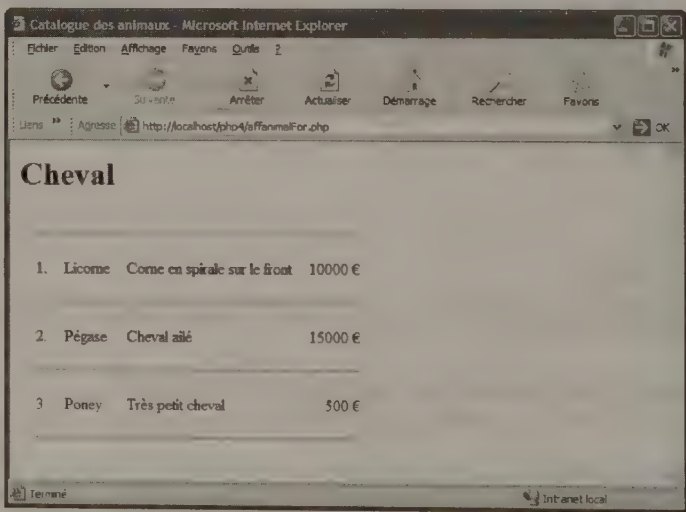
    $connexion = mysqli_connect($host,$user,$password, $database)
        or die ("Connexion au serveur impossible");

    $typeAnimal = "cheval"; // mot saisi dans un formulaire
    $rq = "SELECT * FROM Animal WHERE animalType ='$typeAnimal'";
    $result = mysqli_query($connexion, $rq)
        or die ("Exécution de la requête impossible");
    $nbLignes = mysqli_num_rows($result);

    /* Afficher les résultats dans un tableau */
    $typeAnimal = ucfirst($typeAnimal);
    echo "<h1>$typeAnimal</h1>";
    echo "<table cellpadding='15'>";
    echo "<tr><td colspan='4'><hr></td></tr>";
    for ($i=0; $i<$nbLignes; $i++)
```

```
{ $n = $i + 1; // pour commencer la numérotation à 1
  $ligne = mysqli_fetch_assoc($result);
  extract($ligne);
  $prix_f = number_format($animalPrix, 2);
  echo "<tr>\n
      <td>$n.</td>\n
      <td>$animalNom</td>\n
      <td>$animalDesc</td>\n
      <td align='right'>$prix_f &#8364;</td>\n
    </tr>\n";
  echo "<tr><td colspan='4'><hr></td></tr>\n";
}
echo "</table>\n";
?>
</body>
</html>
```

Figure 8.2 :
Extraction
des animaux
ayant le type
"cheval" de la
base de
données
AniCata. Les
lignes sont
numérotées.



Extraction d'informations à l'aide de fonctions

Dans une application de base de données sur le Web, vous avez besoin d'extraire à plusieurs reprises des données dans un programme, ou dans divers programmes de l'application. Il est donc commode d'écrire une fonction à cet usage.

Chaque fois que le programme a besoin de récupérer des données, il vous suffit d'appeler la bonne fonction. Non seulement vous gagnerez du temps lors de la saisie, mais de surcroît le déroulement du code sera bien plus facile à suivre.

Considérez, par exemple, un catalogue comme celui des animaux AniCata. Le Listing 8.3 vous présente un programme dans lequel on appelle une fonction nommée `extraireInfos()` chaque fois qu'on veut extraire des informations du catalogue pour un animal particulier. Ces informations sont placées dans un tableau, lequel est renvoyé au programme appelant. Celui-ci peut alors en faire ce qu'il veut. Ici, il affichera ces informations.

Listing 8.3 : Fonction d'extraction de données de la base AniCata.

```
<?php
/* Programme : lireLicorne.php (getdata.php)
 * Desc :      Lit des données depuis une base via une fonction
 */
?>
<html>
<head><title>Catalogue des animaux</title></head>
<body>
<?php

    $infosAnimal = extraireInfos("Licorne");    // Appel de fonction

    $prix_f = number_format($infosAnimal['animalPrix'],2);
    echo "<p><b>{$infosAnimal['animalNom']}</b><br>\n
        Description : {$infosAnimal['animalDesc']}<br>\n
        Prix : $prix_f &#8364;\n"

?>
</body></html>

<?php
function extraireInfos($animal)
{
    $user="catalog";
    $host="localhost";
    $password="";
    $database="AniCata";
    $cxn = mysqli_connect($host,$user,$password, $database)
        or die ("Connexion au serveur impossible");
    $query = "SELECT * FROM Animal WHERE animalType='$animal'";
    $result = mysqli_query($cxn,$query)
        or die ("Exécution de la requête impossible");
```

```

    return mysqli_fetch_assoc($result);
}
?>

```

La page Web montrera par exemple ceci :

```

Licorne
Description : Corne en spirale sur le front
Prix : 10000 €

```

Notez les points suivants dans le programme du Listing 8.3 :

- ✓ Le programme est maintenant plus facile à relire par rapport à la version où tous les détails de l'extraction étaient conservés tels quels.
- ✓ L'appel de la fonction utilise l'argument "Licorne" codé sous forme de littéral. Dans la réalité, on le remplacerait plutôt par une variable.
- ✓ Le programme crée la variable `$infosAnimal` pour recueillir les informations extraites de la base. Cette variable est implicitement un tableau parce que les informations qui vont y être placées existent sous cette forme.

La précédente fonction était très simple : elle se contentait de renvoyer une ligne de résultat sous forme de tableau. Mais une fonction peut être bien plus complexe. Dans le programme du Listing 8.4, la fonction `extraireType()` renvoie un tableau à deux dimensions contenant les données concernant tous les animaux du type spécifié.

Listing 8.4 : Affichage d'une liste numérotée des types d'animaux

```

<?php
/* Programme : lireTypes.php (getPets.php)
 * Desc :      Affiche une liste numérotée extraite d'une base.
 */
?>
<html>
<head><title>Catalogue d'animaux</title></head>
<body>
<?php

```

```

$montype = "Cheval";
$animInfo = extraireType($montype);    // Appel de fonction

// Afficher les résultats dans un tableau
echo "<h1>{$montype}</h1>";
echo "<table cellpadding='15'>";
echo "<tr><td colspan='4'><hr /></td></tr>";
for($i=1;$i<=sizeof($animInfo);$i++)
{
    $f_prix = number_format($animInfo[$i]['animalPrix'],2);
    echo "<tr>\n
        <td>{$i}</td>\n
        <td>{$animInfo[$i]['animalNom']}</td>\n
        <td>{$animInfo[$i]['animalDesc']}</td>\n
        <td style='text-align: right'>{$f_prix}</td>\n
    </tr>\n";
    echo "<tr><td colspan='4'><hr /></td></tr>\n";
}
echo "</table>\n";
?>
</body></html>

<?php
function extraireType($aniType)
{
    $user="catalog";
    $host="localhost";
    $passwd="";
    $cxn = mysqli_connect($host, $user, $passwd, "AniCata")
        or die ("Connexion au serveur impossible");
    $req = "SELECT * FROM Pet WHERE animalType='{$aniType}'";
    $result = mysqli_query($cxn,$req)
        or die ("Exécution requête impossible");

    $j = 1;
    while ($ligne = mysqli_fetch_assoc($result))
    {
        foreach ($ligne as $nomColonne => $valeur)
        {
            $tableau[$j][$nomColonne] = $valeur;
        }
        $j++;
    }
    return $tableau;
}
?>

```




Dans l'instruction `return`, le caractère `@` placé juste devant le tableau permet d'éviter l'envoi d'un message d'avertissement dans le cas où la base ne contiendrait aucun animal correspondant au type demandé. Dans ce cas en effet, la boucle `foreach` ne serait pas parcourue, et la variable ne serait donc pas créée.

Voici comment opère ce programme :

1. Il appelle la fonction `extraireType()` à laquelle il passe l'argument "cheval". Le résultat de cet appel sera placé dans la variable `$animInfo`.
2. La fonction se connecte à la base et sélectionne la base de données `AniCata`.
3. La fonction envoie une requête demandant toutes les lignes dans lesquelles la colonne `type` contient `cheval`. Ces informations seront placées dans une zone temporaire vers laquelle pointerait la variable `$result`.
4. Initialisation à 1 d'un compteur : `$j`.
5. Début d'une boucle `while`. La fonction demande à extraire une ligne de la zone temporaire. En cas de réussite, cette ligne est placée dans le tableau `$ligne`. S'il n'y a plus rien, la boucle `while` se termine.
6. Début d'une boucle `foreach`. La boucle explore chaque champ de la ligne.
7. Rangement des valeurs dans `$tableau[]` qui est un tableau à deux dimensions. Sa première champ est un nombre défini par le compteur `$j` qui, la première fois, vaut 1. Tous les champs de la ligne sont rangés avec, en guise de champ, le nom de la colonne où ils se trouvent.
8. Le compteur `$j` est incrémenté d'une unité.
9. Fin de la boucle `while`.
10. Retour au début de la boucle `while`.
11. Répétition des étapes 6 à 11 pour chaque ligne de résultats.
12. Le tableau `$tableau[]` est renvoyé au programme appelant. Il contient toutes les informations relatives aux lignes sélectionnées.

13. `animInfo[]` reçoit ces résultats. La Figure 8.3 montre comment sont structurées ces informations.
14. Ces informations sont envoyées au navigateur après avoir été mises en forme comme un tableau HTML.

<code>infosType [1] [animal]</code>	=	Licorne
<code>[description]</code>	=	Corne en spirale sur le front
<code>[prix]</code>	=	10000
<code>infosType [2] [animal]</code>	=	Pégase
<code>[description]</code>	=	Cheval ailé
<code>[prix]</code>	=	15000
<code>infosType [3] [animal]</code>	=	Poney
<code>[description]</code>	=	Très petit cheval
<code>[prix]</code>	=	500

Figure 8.3 :
Structure des
informations
contenues
dans le
tableau à
deux
dimensions
`infosType[]`.

La page Web résultante est identique à celle de la Figure 8.2 obtenue sans faire appel à une fonction utilisateur. Les deux méthodes de travail sont toujours possibles. Simplement, l'usage de fonctions simplifie la programmation, de même que la compréhension et la maintenance du code que vous écrivez.

Recueil d'informations auprès de l'utilisateur

De nombreuses applications sont conçues pour poser des questions auxquelles l'utilisateur répond en saisissant ses réponses dans un formulaire. Parfois, ces informations sont rangées dans une base de données ; à d'autres moments, elles sont utilisées dans des instructions conditionnelles pour afficher une nouvelle page. Voici quelques exemples de ces tâches :

- ✓ **Commandes en ligne.** Les clients choisissent un produit et donnent leurs coordonnées et les informations concernant leur mode de règlement.
- ✓ **Enregistrement.** Certains sites demandent à leurs visiteurs de fournir des informations particulières pour pouvoir bénéficier

d'avantages réservés : accès à des informations confidentielles, téléchargement de logiciels...

- ✓ **Identification.** L'accès à certains sites est réservé aux membres d'une association qui doivent s'enregistrer pour pouvoir en afficher les pages.
- ✓ **Affichage d'informations sélectionnées.** De nombreux sites proposent aux utilisateurs de spécifier les informations qu'ils veulent consulter. Il peut s'agir par exemple d'une certaine catégorie de produits dans un catalogue en ligne.

Les questions sont posées dans un formulaire ; l'utilisateur y répond dans les boîtes de saisie (ou à l'aide des listes) de ce même formulaire, puis clique sur un bouton pour envoyer ses réponses au serveur Web. C'est alors un autre programme qui prend la main pour traiter ces informations.



Dans ce qui suit, je supposerais que vous savez afficher un formulaire en HTML. Mon propos est de vous montrer comment effectuer ce travail avec PHP, et comment traiter les informations saisies par l'utilisateur dans un formulaire.

PHP et les formulaires HTML

Si vous ne connaissez pas très bien les formulaires HTML, je vous recommande de consulter les pages qui traitent de ce sujet dans un livre consacré à ce sujet si important pour l'interactivité des sites Web. Pour afficher un formulaire avec PHP, vous pouvez choisir entre ces deux façons d'opérer :

- ✓ **Envoyer des commandes HTML au navigateur au moyen d'instructions echo.** Par exemple :

```
echo "<form action='traitement.php' method='POST'>\n
    <input type='text' name='nomcomplet'>\n
    <input type='submit' value='Envoyez'>\n
</form>\n";
```

- ✓ **Ecrire du code HTML pur en dehors des sections PHP.** Lorsqu'il s'agit d'un formulaire statique, il n'y a pas de raison de l'inclure dans une section PHP. Par exemple :

```
<?php
... instructions de la section PHP ...
```

?>

```
<form action="traitement.php" method="POST">
<input type="text" name="nom">
<input type="submit" value ="Envoyer">
</form>
```

```
<?php
... instructions de la section PHP ...
?>
```

Ces deux méthodes afficheront ce qui est reproduit sur la Figure 8.4.

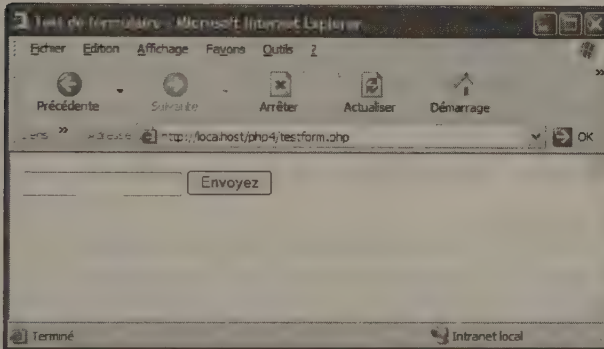


Figure 8.4 :
Exemple de
formulaire
HTML.

Lorsque l'utilisateur clique sur le bouton de validation (légendé par exemple "Envoyer" ou "Soumettre"), les informations contenues dans la boîte de saisie sont placées dans une variable appelée `nom` et envoyées au programme `traitement.php` (celui dont le nom est indiqué comme valeur de l'attribut `action` du formulaire). Ce programme PHP sera alors à même d'accéder au contenu de la variable sous le nom `$nom`. Pour savoir comment traiter les informations provenant du formulaire, consultez la section "Traitement des informations provenant d'un formulaire", plus loin dans ce même chapitre. Mais en attendant, voyons cela d'un peu plus près.

L'information saisie dans le formulaire est récupérée en exécutant un programme qui reçoit ces renseignements. Quel est ce programme ? Celui qui est défini par le paramètre `action` du formulaire. Dans l'exemple précédent, il s'agira donc de `traitement.php`. Celui-ci peut alors afficher, enregistrer ou effectuer toute action utile à partir des données retournées par l'utilisateur lorsqu'il clique sur le bouton `Envoyez`.

Lorsque le visiteur clique sur le bouton de validation, le programme spécifié dans l'attribut action est lancé. Ses instructions ont accès aux données du formulaire via des tableaux intégrés de PHP appelés `$_POST`, `$_GET` et `$_REQUEST`. Ce sont des tableaux dits *superglobaux*. Si vous utilisez la méthode POST, les données sont stockées dans le tableau `$_POST`. Le tableau `$_GET` contient les données transmises via l'adresse URL, y compris donc les champs provenant d'un formulaire via la méthode GET. Enfin, le tableau `$_REQUEST` contient tous les éléments des tableaux `$_POST`, `$_GET` et `$_COOKIES`. Nous parlons des cookies dans le Chapitre 9.

Une fois le formulaire envoyé, le programme qui est exécuté peut récupérer les informations voulues à partir d'un tableau prédéfini adapté. Dans ces tableaux, chaque indice est le nom d'un champ de saisie du formulaire. Supposons que, sur la Figure 8.4, l'utilisateur tape Jean Dupont dans le champ de saisie puis clique sur le bouton Envoyer. Le programme `traitement.php` se lance et peut alors se servir d'une variable de tableau de la façon suivante :

```
$_POST['nomcomplet']
```

Le nom entré dans le formulaire est mis à la disposition du tableau `$_POST` car la balise du formulaire précise `method="POST"`. De plus, le nom donné au champ dans le formulaire HTML est bien celui qui est repris comme indice du tableau :

```
<input type='text' name='nomcomplet'>
```



Les tableaux *superglobaux* `$_POST` et `$_GET` ont été introduits dans PHP 4.1. Auparavant, les données devaient être transmises via des tableaux appelés `$HTTP_POST_VARS` et `$HTTP_GET_VARS`. Si vous utilisez PHP 4.0 ou une version encore plus ancienne, vous êtes forcé d'utiliser ces tableaux longs. Les deux formats de tableaux coexistent jusqu'à PHP 5, mais les tableaux longs ont disparu dans PHP 6. Si vous exploitez de vieux scripts, il faudra donc modifier les références aux tableaux longs du style `$HTTP_POST_VARS` pour qu'elles désignent les tableaux *superglobaux* du style `$_POST`. Une recherche/remplacement soigné devrait suffire pour chaque tableau.

Pour tester un formulaire, il est utile de disposer d'un programme qui affiche tous ses champs. Vous pouvez alors analyser les valeurs transmises par le formulaire afin de contrôler sa mise en forme, les noms des champs et leur contenu. Le Listing 8.5 propose un programme, appelé `traitement.php`, qui se lance automatiquement

lorsque l'utilisateur clique sur le bouton de validation du formulaire précédent.

Listing 8.5 : Script affichant tous les champs d'un formulaire.

```
<?php
/* Nom du script : traitement.php (processform.php)
 * Description : Affiche les données transmises depuis
 *              un formulaire.
 */
echo "<html>
      <head><title>Adresse du client</title></head>
      <body>";
foreach ($_POST as $champ => $valeur)
{
    echo "$champ = $valeur<br>";
}
?>
</body></html>
```

Si le visiteur saisit le nom Jean Dupont dans le formulaire de la Figure 8.4, le résultat affiché est :

nom complet = Jean Dupont

Il n'y a qu'une ligne, car il n'y a qu'un champ dans le formulaire de la Figure 8.4.

Le listing précédent convient à tout formulaire utilisant la méthode POST. Prenons maintenant un cas un peu plus compliqué. Le Listing 8.6 affiche un formulaire contenant plusieurs champs.

Listing 8.6 : Programme affichant un formulaire d'adresse.

```
<?php
/* Nom du programme : afficheForm.php (displayForm.php)
 * Description: Ce script affiche un formulaire qui demande
 *              l'adresse du client.
 */
echo "<html>
      <head><title>Adresse Client</title></head>
      <body>";
$infos = array( "prénom"=>"Prénom :",
```

```

        "nom"=>"Nom :",
        "rue"=>"Adresse :",
        "ville"=>"Ville :",
        "département"=>"Département :",
        "codepostal"=>"Code postal :");
echo "<p align='center'>
    <b>Entrez votre adresse dans le formulaire qui suit.</b><hr>";
echo "<form action='traitement.php' method='POST'>
    <table width='95%' border='0' cellspacing='0'
        cellpadding='2'>\n";
foreach($infos as $champ=>$info)
{
    echo "<tr>
        <td style='text-align: right;
            font-weight: bold'>
            $info</td>
        <td><input type='text' name='$champ' size='65'
            maxlength='65' ></td>
        </tr>";
}
echo "<tr>
    <td colspan='2' style='text-align: center'>
        <input type='submit'
            value='Envoi'>";
echo "</td></tr></table>
    </form>";
?>
</body></html>

```

Remarquez dans ce listing les principaux points suivants :

- ✓ **Un tableau reçoit les intitulés des champs utilisés dans le formulaire.** Les clés sont donc les noms des champs.
- ✓ **Le script `traitement.php` est celui qui est exécuté lorsque le formulaire est validé.** Il reçoit les informations provenant du formulaire afin de les traiter.
- ✓ **Le formulaire est affiché dans un tableau HTML.** Il est important de savoir construire des tableaux HTML afin de mettre correctement en forme vos pages WEB.
- ✓ **Le tableau `$infos` est parcouru dans une boucle `foreach`.** Le code HTML créant chaque ligne de l'affichage est construit lors des différents passages dans la boucle à l'aide des valeurs successives renvoyées par le tableau.



Pour des raisons de sécurité, incluez toujours une balise `maxlength` afin d'indiquer le nombre maximum de caractères que l'utilisateur peut saisir dans chaque champ du formulaire. Vous éviterez ainsi que des personnes mal intentionnées saisissent du code exécutable dans un champ ! Si les informations sont destinées à être enregistrées dans une base de données, affectez comme valeur à `maxlength` la largeur de la colonne correspondante dans la table de destination.



Enregistrer des tableaux longs

Revenons un peu sur le cas de PHP 5. En rappelant tout d'abord que cette version est encore en phase de projet à la date où ce livre est écrit. La version 5 de PHP introduit dans `php.ini` Il est en effet peu probable que vous en ayez l'utilité à moins de vouloir conserver de vieux scripts contenant des variables longues. Ce paramètre est contrôlé par la ligne suivante dans `php.ini` :

```
register_long_arrays = On
```

Dans PHP 5, ce réglage possède par défaut la valeur `On`. Si la compatibilité avec d'anciens scripts n'est pas un impératif, vous devriez lui affecter la valeur `Off` afin d'éviter à PHP un travail inutile.

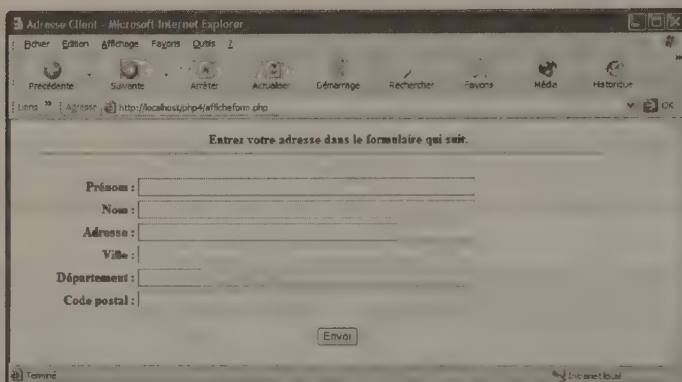
Dans PHP 6, le réglage `register_long_arrays` disparaît de `php.ini` et il n'y a plus de tableaux longs. Si vous utilisez des scripts anciens, vous devez modifier les noms des tableaux longs, comme `$HTTP_POST_VARS`, pour adopter les nouveaux noms de tableaux globaux dans le style `$_POST`.

Lorsque Jean Dupont remplit le formulaire produit par le Listing 8.6, et qui est illustré sur la Figure 8.5, le programme `traitement.php` est exécuté et il produit une sortie comme celle-ci :

```
prénom = Jean
nom = Dupont
rue = 123 Rue des Champs
ville = Paris
département = Seine
codepostal = 75999
```

Dans `traitement.php`, tous les éléments du tableau prédéfini `$_POST` sont affichés sur une ligne, la méthode employée étant ici `POST`, ce qui est le cas de la majorité des formulaires.

Figure 8.5 :
Formulaire
pour la saisie
d'une
adresse
client.



Créer des formulaires dynamiques

PHP apporte de nouvelles possibilités aux formulaires HTML en les dynamisant, notamment grâce à l'emploi de variables. Cela concerne en particulier les points suivants :

- ✓ Affichage dynamique d'informations dans les champs de formulaire.
- ✓ Construction dynamique de listes de sélection.
- ✓ Construction dynamique de boutons radio.
- ✓ Construction dynamique de cases à cocher.

Affichage dynamique d'informations dans les champs de formulaire

Lorsque vous affichez un formulaire dans une page Web, vous pouvez placer des informations à l'intérieur des boîtes de saisie au lieu de vous contenter d'afficher des champs vides. Si, par exemple, la plus grande partie de vos clients réside en France, vous pouvez placer initialement la chaîne de caractères "France" dans la boîte de saisie "Pays de résidence", ce qui évitera au client français d'avoir à saisir ce nom (et de risquer du même coup des erreurs de frappe). Si le client est un étranger, il lui suffira d'écraser cette mention avec le nom du pays où il réside. De plus, le texte saisi automatiquement ne comportera pas d'erreurs de frappe.

Pour initialiser le champ d'une boîte de saisie, vous devez utiliser son attribut `value` de la façon suivante :

```
<input type = "text" name="pays" value="France">
```

Avec PHP, vous pouvez utiliser le contenu d'une variable pour initialiser ce champ au moyen de l'une des deux instructions suivantes :

```
<input type="text" name="pays" value="<?php echo $pays ?>">
```

```
echo "<input type='text' name='pays' value='<$pays'>";
```

Le premier exemple crée un champ d'entrée dans lequel seule la valeur de l'attribut `value` sera initialisée dynamiquement au moyen de `echo $pays`. L'instruction est placée entre deux balises PHP, créant ainsi une mini-section PHP dans le document HTML. Le second exemple génère au moyen d'une instruction `echo` la totalité du champ `<input ...>` du document HTML.

Si vous avez des informations concernant l'utilisateur dans une base de données, vous pouvez vouloir les afficher dans certains champs du formulaire. Cette technique permettra à l'utilisateur de vérifier ces renseignements, et éventuellement de les corriger. Ou bien vous pourriez proposer comme adresse d'expédition de la commande celle qui était utilisée pour le dernier envoi, afin que le client n'ait pas à ressaisir son adresse complète. Le Listing 8.7 présente le détail d'un programme affichant un formulaire contenant des informations extraites d'une base de données. Ce formulaire ressemble à celui de la Figure 8.5, sauf qu'il contient des informations récupérées dans la base et que les champs du formulaire en Figure 8.5 sont vides.

Listing 8.7 : Programme affichant un formulaire HTML

```
<?php
/* Nom du programme : afficheAdresse (displayAddress.php)
 * Description : Le script affiche un formulaire contenant
 *              les adresses lues dans la base de données.
 */
echo "<html>
      <head><title>Adresse du client</title></head>
      <body>";
$infos = array( "prénom"=>"Prénom :",
                "nom"=>"Nom :",
                "rue"=>"Adresse :",
                "ville"=>"Ville :",
```



```

        "département"=>"Département :";
        "codepostal"=>"Code postal :";

$user="admin";
$host="localhost";
$password="";
$databse = "MembresSeuls";
$nomLogin = "jDupont";    // login de l'utilisateur

$connexion= mysqli_connect($host,$user,$password, $databse)
    or die ("Connexion au serveur impossible");

$query = "SELECT * FROM Membre
          WHERE login='$nomLogin'";
$result = mysqli_query($connexion, $query)
    or die ("Exécution requête impossible");
$ligne = mysqli_fetch_assoc($result);

echo "<p align='center'>
      <h1 align='center'>Adresse de $nomLogin</h1>\n";
echo "<br><p align='center'>
      <font size='+1'><b>Merci de vérifier l'adresse ci-dessous et
      de rectifier ce qui est inexact.</b></font>
      <hr>";
echo "<form action='traitement.php' method='POST'>
      <table width='95%' border='0' cellspacing='0'
          cellpadding='2'>\n";
foreach($infos as $champ=>$info)
{
    echo "<tr>
          <td align='right'> <B>{$infos[$champ]} </br></td>
          <td><input type='text' name='$champ'
              value='$ligne[$champ]' size='65' maxlength='65'>
          </td>
          </tr>";
}
echo "</table>
      <div align='center'><p><input type='submit' value='Envoyer'> </p></div>
      </form>";
?>
</body></html>

```

Dans ce programme, vous pouvez remarquer les points suivants :

- ✓ **Le formulaire passe le contrôle au programme**
 traitement.php. C'est ce programme qui traitera les informations contenues dans le formulaire et mettra à jour les éléments correspondants de la base de données, modifiés

éventuellement par l'utilisateur. C'est à vous d'écrire ce programme. Nous reviendrons plus loin dans ce chapitre sur le contrôle des informations saisies dans le formulaire et leur enregistrement dans la base de données.

- ✓ **Les champs du formulaire sont mis en forme dans un tableau HTML.** Le but est de réaliser une présentation plus harmonieuse, donc de faciliter le contrôle demandé à l'utilisateur.
- ✓ **Chacun des champs du formulaire a un nom.** Celui-ci correspond à la variable PHP ayant le même nom (au "\$" initial près). C'est ce qui permettra d'établir la communication avec le programme PHP.
- ✓ **Les noms des champs du formulaire sont les mêmes que ceux des colonnes de la base de données.** On simplifie ainsi les transferts d'informations entre la base de données et le formulaire.
- ✓ **Les valeurs provenant de la base de données sont affichées dans les champs du formulaire grâce au paramètre `value`.** Celui-ci fournit l'information adéquate provenant du tableau \$ligne, qui contient les données provenant de la base.



Pour des raisons de sécurité, pensez à inclure systématiquement un attribut `maxlength` dans vos champs d'entrée de formulaire. Cette valeur limite le nombre de caractères que l'utilisateur est autorisé à saisir dans le champ. Il est recommandé d'affecter à cet attribut la même largeur que la colonne correspondante dans la base de données.

La Figure 8.6 montre ce que ce programme affiche dans la fenêtre du navigateur à partir des éléments contenus dans la base de données.

Construction dynamique de listes de sélection

Parmi les champs de formulaire très utilisés figure la *liste de sélection*. Au lieu de saisir librement une certaine information dans une boîte de saisie, l'utilisateur se voit proposer une liste de valeurs possibles dans une boîte à liste déroulante. Par exemple, dans un catalogue de produits, on peut proposer à l'utilisateur une liste déroulante des produits existants pour qu'il choisisse la catégorie qui l'intéresse. De la même façon, la rubrique "Pays" pourrait proposer une liste de pays à destination desquels il est possible d'expédier

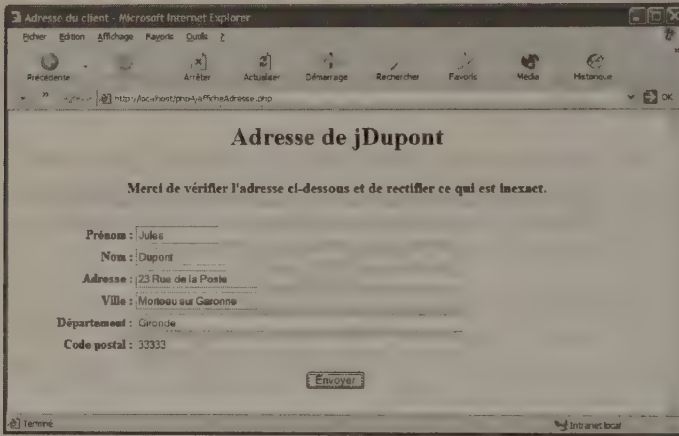


Figure 8.6 :
Formulaire
affichant
l'adresse de
l'utilisateur.

une commande. Enfin, pour donner une date, trois listes de sélection (jour, mois, année) éviteront toute saisie "hasardeuse".



Utilisez le plus souvent possible les listes de sélection. Vous éviterez ainsi les fautes de frappe et/ou d'orthographe toujours possibles dans une boîte de saisie en forme libre.

Voici comment pourrait se présenter une liste HTML de types d'animaux pour la base de données AniCata :

[illegible]

La Figure 8.7 montre comment se présente cette liste de sélection. Remarquez que "chat" est la valeur proposée par défaut, en raison de la présence de l'attribut `selected` dans le champ correspondant.

Lorsque le visiteur clique sur la petite flèche de la boîte de sélection, la liste se déroule, comme on le voit sur la Figure 8.8. Il peut alors sélectionner un autre champ parmi les trois qui lui sont proposés. Vous remarquez que la valeur chat est sélectionnée tant que le visiteur ne choisit pas autre chose.

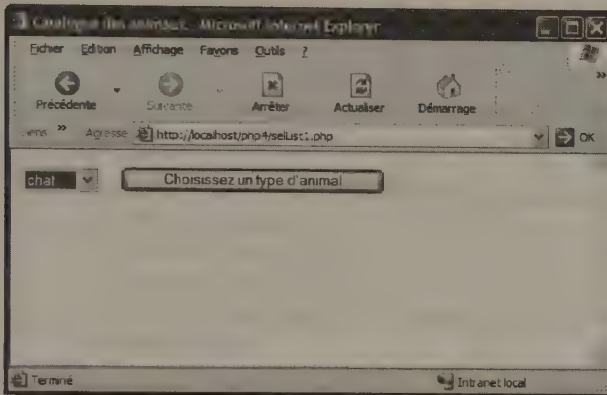


Figure 8.7 :
Boîte à liste
déroulante.

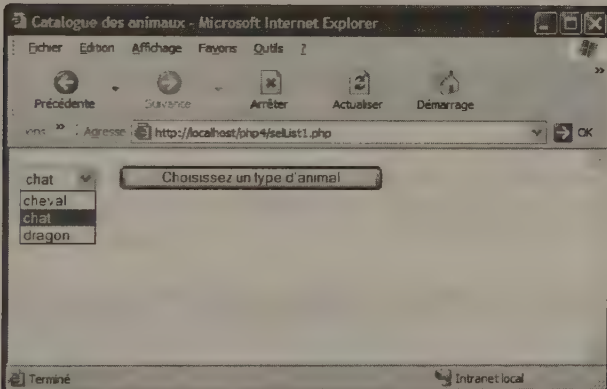


Figure 8.8 :
Les valeurs
possibles
figurent
toutes dans
la boîte à liste
déroulante.

A l'aide de PHP, vous pouvez employer des variables pour afficher des listes de sélection dynamiques. Avec une liste statique, si votre magasin d'animaux décide d'ajouter un nouveau type d'animal à ceux qu'il commercialise déjà, vous devrez modifier à la main cette boîte de sélection. Alors que si la liste est construite dynamiquement à partir de la base de données, elle reflétera toujours l'état des lieux. Le Listing 8.8 montre comment réaliser dynamiquement cette même liste de sélection.

Listing 8.8 : Construction dynamique d'une liste de sélection.

```
<?php
/* Nom du programme : animSelect.php (buildSelect.php)
```

```

* Description : Construction d'une liste de sélection
*               à partir de la base de données.
*/
?>
<html>
<head>
<title>Types d'animaux</title>
</head>
<body>
<?php
    $user="catalog";
    $host="localhost";
    $password="";
    $database = "AniCata";

    $connexion = mysqli_connect($host, $user, $password, $database)
        or die ("Connexion au serveur impossible");

    $req = "SELECT DISTINCT animalType FROM Animal ORDER BY animalType";
    $result = mysqli_query($connexion, $req)
        or die ("Exécution de la requête impossible");

    // Création d'un formulaire contenant une liste de sélection
    echo "<form action='traitement.php' method='POST'>
        <select name='animalType'>\n";

    while ($ligne = mysqli_fetch_assoc($result))
    {
        extract($ligne);
        echo "<option value='$animalType'>$animalType\n";
    }
    echo "</select>\n";
    echo "<input type='submit' value='Choisissez un type d'animal'>
        </form>\n";
?>
</body>
</html>

```

Notez les points suivants concernant ce programme :

- ✓ **Utilisation de DISTINCT dans la requête.** Cette clause oblige MySQL à ne fournir qu'une seule fois chaque type. Si on l'omettait, on recevrait *tous* les types présents, ce qui causerait des doublons.

- ✓ **La clause ORDER BY** renvoie les résultats triés par ordre alphabétique.
- ✓ **Des instructions echo encadrent la boucle while.** Ce sont elles qui créent les balises `<form>` et `<select>`.
- ✓ **L'instruction echo à l'intérieur de la boucle while.** C'est elle qui crée les balises `<option>` pour chaque type d'animal. Aucune de ces entrées n'est marquée `selected`. Il n'y a donc pas de type sélectionné par défaut. Plus exactement, c'est le premier élément par ordre d'apparition qui est présélectionné.
- ✓ **La dernière instruction echo.** C'est elle qui crée les balises fermantes `</form>` et `</select>` et le bouton `submit` sur lequel cliquera le visiteur pour valider son choix et l'envoyer au programme `traitement.php`.

La liste de sélection est identique dans son principe à celle qui avait été créée par le précédent programme (celui du Listing 8.7) et que montrait les Figures 8.7 et 8.8. Mais son contenu est différent, car il est l'exact reflet du catalogue, comme on peut le voir sur la copie d'écran de la Figure 8.9. En particulier, les noms sont triés en ordre alphabétique.

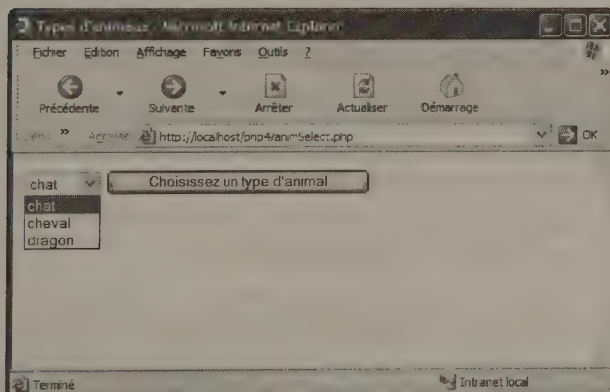


Figure 8.9 :
Liste de
sélection
construite
dynamiquement

Vous pouvez également utiliser des variables PHP pour marquer un choix par défaut dans ce type de liste. Supposons, par exemple, que vous demandiez à l'utilisateur de choisir une date (jour, mois, année) dans trois listes. La plupart des gens vont choisir la date du

jour. Vous souhaitez donc que ce soit elle qui soit "présélectionnée". Le programme du Listing 8.9 construit cette liste :

Listing 8.9 : Création de trois listes avec présélection de la date du jour.

```
<?php
/* Nom du programme : dateSelect.php
 * Description : Le programme affiche une liste de sélection
 *              dans laquelle le client choisit une date.
 */
?>
<html>
<head>
<title>Choix d'une date</title>
</head>
<body>
<?php

// Tableau des mois
$Mois = array(1=> "Janvier", "Février", "Mars", "Avril",
               "Mai", "Juin", "Juillet", "Août",
               "Septembre", "Octobre", "Novembre",
               "Décembre");

$aujourd'hui = time(); // date du jour
$f_aujourd'hui = date("d/n/Y", $aujourd'hui); // mise en forme

echo "<div style = 'text-align: center'>\n";

// Affichage de la date du jour
echo "<p>&nbsp;<h3>Nous sommes aujourd'hui le $f_aujourd'hui
    </h3><hr>\n";

// Création d'un formulaire approprié
echo "<form action='traitement.php' method='POST'>\n";

// Construction de la liste pour les jours
$aujourd'hui_jours = date("d", $aujourd'hui); // $aujourd'hui -> jour
echo "<select name='jour'>\n";
for ($n=1; $n<=31; $n++)
{ echo " <option value=\"$n\";
  if ($aujourd'hui_jours == $n)
  { echo " selected";
  }
  echo "> $n\n";
}
echo "</select>\n";
```

```
// Construction de la liste pour les mois
$aujourd'hui_mois = date("m", $aujourd'hui); // $aujourd'hui -> mois
echo "<select name='mois'>\n";
for ($n=1; $n<=12; $n++)
{ echo "<option value=$n\n";
  if ($aujourd'hui_mois == $n)
  { echo " selected";
  }
  echo "> $Mois[$n]\n";
}
echo "</select>";

// Construction de la liste pour les années
$anDébut = date("Y", $aujourd'hui); // $aujourd'hui -> année
echo "<select name='année'>\n";
for ($n=$anDébut; $n<=$anDébut+3; $n++)
{ echo " <option value=$n";
  if ($anDébut == $n )
  { echo " selected";
  }
  echo "> $n\n";
}
echo "</select>\n";
echo "</form>\n";
?>
</body>
</html>
```

La page Web créée par ce programme est affichée Figure 8.10. La date apparaît en titre au-dessus des trois boîtes de sélection afin que vous puissiez contrôler l'exactitude des présélections. La liste des années est limitée à quatre valeurs, alors que celle des jours en comprend toujours 31 et celle des mois, 12.

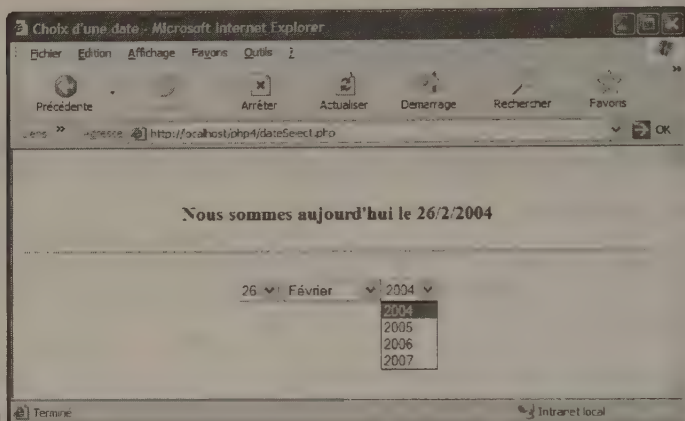


Dans la réalité, ce programme devrait être modifié pour n'afficher que le nombre exact de jours existant dans le mois sélectionné. Attention aux années bissextiles pour février !

Voici les tâches accomplies par ce programme :

1. Création d'un tableau des mois dont les clés sont des numéros commençant à 1 de façon à correspondre à nos habitudes de numérotation des dates.

Figure 8.10 :
Formulaire
proposant
par défaut la
date du jour.



2. Création des variables `$aujourd'hui` (contenant la date du jour) et `$f_aujourd'hui` (contenant la date du jour mise en forme pour l'affichage dans la page Web).
3. Affichage de la date du jour dans un titre HTML de niveau 3 centré dans la page.
4. Construction de la liste de sélection des jours :
 - a. Création de la variable `$aujourd'hui_jours` contenant la date du jour extraite de `$aujourd'hui`.
 - b. Envoi de la balise initiale `<select>` par une instruction `echo`.
 - c. Initialisation d'une boucle `for` dont le compteur variera de 1 à 31.
 - d. Affichage du numéro du jour par une instruction `echo`.
 - e. Si le numéro du jour est égal au jour provenant de `$aujourd'hui`, ajout de la clause `selected`.
 - f. La boucle est répétée jusqu'à la fin (de 1 à 31).
 - g. Envoi de la balise terminale `</select>` par une instruction `echo`.
5. Construction semblable à la précédente de la liste de sélection des mois. Elle va de 1 à 12 et le nom du mois provient du tableau `$Mois`.

6. Construction identique à la précédente de la liste de sélection des années. Elle part de l'année courante et comprend 4 valeurs.
7. Création par une instruction echo de la balise terminale `</select>`.

Construction dynamique de boutons radio

De la même façon, il est facile de construire dynamiquement une liste de boutons radio. Par exemple, vous pouvez afficher une liste de boutons radio pour le catalogue des animaux et demander aux utilisateurs de cliquer sur celui qui correspond au type d'animal qui les intéresse.

Le format HTML d'un bouton radio est le suivant :

```
<input type="radio" name="xxx" value = "yyy">
```

Nous allons construire cette liste pour tous les types d'animaux présents dans le catalogue. Le Listing 8.10 présente le programme qui va accomplir cette tâche.

Listing 8.10 : Construction dynamique d'une liste de boutons radio

```
<?php
/* Nom du programme : dynamBoutonRadio.php (buildRadio.php)
 * Description : Affiche une liste de boutons radio à partir
 *              des informations de la base de données.
 */
?>
<head><title>Types d'animaux</title></head>
<body>
<?php
    $user="catalog";
    $host="localhost";
    $password="";
    $database = "AniCata";

    $connexion = mysqli_connect($host, $user, $password, $database)
        or die ("Connexion au serveur impossible");

    $rq = "SELECT DISTINCT animalType FROM Animal ORDER BY animalType";
    $result = mysqli_query($connexion, $rq);
    or die ("Exécution de la requête impossible");
```



```
// Création d'une liste de boutons radio
echo "<div style='margin-left: .5in; margin-top: .5in'>"
<p>&nbsp;
<p><b>Quel est le type d'animal qui vous intéresse ?</b>
<p>Choisissez-le dans la liste ci-dessous :<n";

// Création d'un formulaire contenant les boutons radio
echo "<form action='traitement.php' method='post'>\n";

while ($ligne = mysqli_fetch_assoc($result))
{
    extract($ligne);
    echo "<input type='radio' name='choix' value='$animalType'>$animalType\n";
    echo "<br>\n";
}

echo "<p><input type='submit' value='Validez votre choix'>"
    </form>\n";

?>
</div>
</body>
</html>
```

La copie d'écran de la Figure 8.11 vous présente le résultat obtenu.

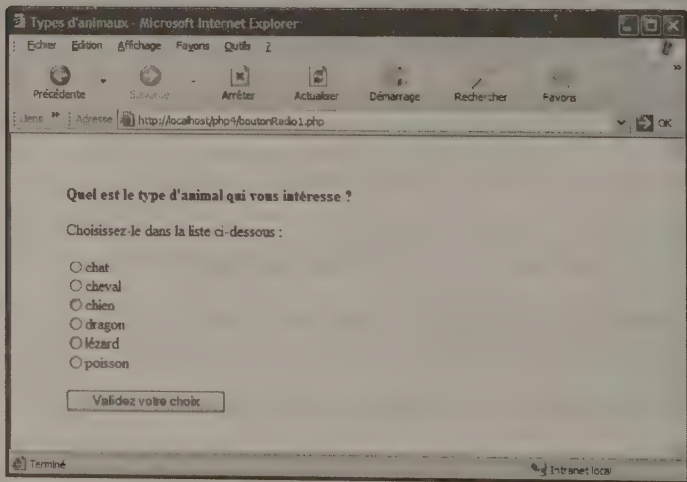


Figure 8.11 :
Liste de
boutons radio
construite
dynamiquement.

Construction dynamique de cases à cocher

Vous pouvez préférer les cases à cocher aux boutons radio. Alors que ceux-ci ne permettent qu'un seul et unique choix, les cases à cocher autorisent un choix multiple. Par exemple, dans une liste de types d'animaux, vous pouvez en choisir deux ou trois. Le programme du Listing 8.11 crée autant de cases à cocher dans une page Web qu'il y a de types d'animaux dans le catalogue.

Listing 8.11 : Construction dynamique de cases à cocher.

```
<?php
/*
 * Nom du programme : dynamCaseCocher.php (buildCheckBox.php)
 * Description : Affiche une liste de cases à cocher à partir
 *               des informations de la base de données.
 */
?>

<html>
<head><title>Types d'animaux</title></head>
<body>
<?php
    $user="catalog";
    $host="localhost";
    $password="";
    $database = "AniCata";

    $connexion = mysqli_connect($host, $user, $password, $database)
        or die ("Connexion au serveur impossible");

    $rq = "SELECT DISTINCT animalType FROM Animal ORDER BY animalType";
    $result = mysqli_query($connexion, $rq)
        or die ("Exécution de la requête impossible");

    // Création d'une liste de cases à cocher
    echo "<div style='margin-left: .5in; margin-top: .5in'>
    <p>&nbsp;
    <p><b>Quel est le type d'animal qui vous intéresse ?</b>
    <p>Choisissez-le dans la liste ci-dessous : \n";

    // Création d'un formulaire contenant les cases à cocher
    echo "<form action='traitement.php' method='POST'>\n";

    while ($ligne = mysqli_fetch_assoc($result))
    {
        extract($ligne);
        echo "<input type='checkbox' name='choix[$animalType]'

```

```

        value='$animalType'>$animalType\n";
    echo "<br>\n";
}
?>
<p>
<input type='submit' value='Validez votre choix'>
</form>
</div>
</body>
</html>

```

Ce programme ressemble beaucoup au précédent. Cependant, on utilise ici une variable dimensionnée (à une seule dimension), `$choix`, pour représenter *les* choix effectués par l'utilisateur. Dans ce tableau, il y aura une paire champ/valeur pour chacun des choix opérés par l'utilisateur. Si ce dernier a coché les cases "Chat", "Chien" et "Poisson", le tableau `$choix[]` contiendra :

```

$choix[Chat] = Chat
$choix[Chien] = Chien
$choix[Poisson] = Poisson

```

Le programme qui traite le formulaire aura à sa disposition les sélections réalisées dans le tableau `$_POST` :

```

$_POST['choix']['chat']
$_POST['choix']['chien']
$_POST['choix']['poisson']

```

La copie d'écran de la Figure 8.12 montre ce qui est affiché dans la fenêtre du navigateur.

Traitement des informations provenant d'un formulaire

Lorsqu'un client remplit les boîtes de saisie et coche les cases d'un formulaire puis clique sur le bouton de type *submit*, il vous envoie des informations. Comment allez-vous les récupérer ?

Dans l'attribut `action` du formulaire, vous indiquez le nom du programme qui va être appelé pour traiter ces informations. Dans les exemples précédents, cet attribut avait pour valeur `traitement.php`. Ce programme doit donc récupérer les informations

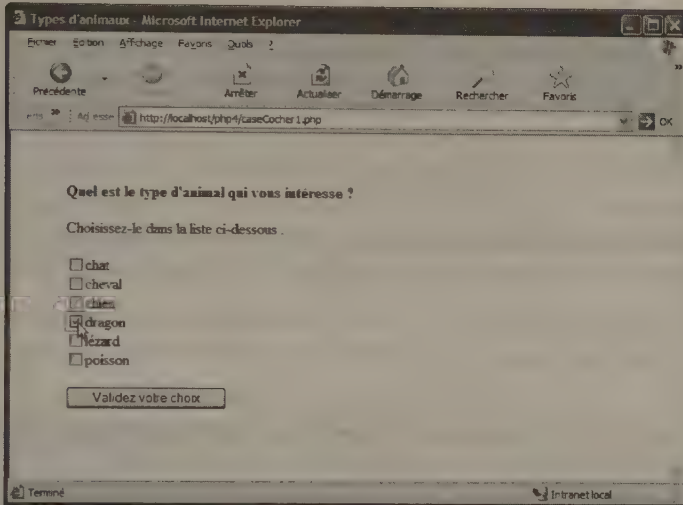


Figure 8.12 :
Liste de
cases à
cocher
construite
dynamiquement.

provenant du formulaire, travail pour lequel PHP est particulièrement bien adapté. Il vous suffit en effet de récupérer le contenu voulu en vous servant des tableaux prédéfinis puis de l'utiliser.

Les valeurs du formulaire sont transmises sous forme de tableaux dont le nom des clés est identique à celui du champ `name` de la balise correspondante du formulaire. Si, par exemple, vous avez envoyé le champ suivant dans votre formulaire :

```
echo "<input type='text' name='prenom'"
```

la valeur saisie par l'utilisateur sera placée dans la variable PHP `$_POST[prenom]`. L'information sélectionnée par l'utilisateur dans une liste déroulante ou un bouton radio est traitée selon le même principe. Prenons le cas d'un groupe de boutons radio comme celui-ci :

```
echo "<input type='radio' name='choix' value='chien'>chien\n";
echo "<input type='radio' name='choix' value='chat'>chat\n";
```

La variable `$_POST[choix]` aura pour valeur la chaîne de caractères "chien" ou "chat" selon le bouton sur lequel aura cliqué l'utilisateur.

Les cases à cocher doivent être traitées d'une façon particulière en raison des possibilités de choix multiples qu'elles offrent. Comme nous l'avons vu dans le programme du Listing 8.11, c'est un tableau qui va contenir les valeurs correspondant aux choix de l'utilisateur

et non plus une variable scalaire. Supposons que votre formulaire propose par exemple les boutons suivants :

```
echo "<input type='checkbox' name='choix[chien]'
      value='chien'>chien\n";
echo "<input type='checkbox' name='choix[chat]'
      value='chat'>chat\n";
```

Vous pouvez alors accéder aux données en utilisant la variable à plusieurs dimensions `$_POST[choix]`, qui va contenir ceci :

```
$_POST[choix][chien] = chien
$_POST[choix][chat] = chat
```



POST ou GET ?

Vous avez le choix entre ces deux valeurs pour l'attribut `method` de la balise `<form>`. Elles passent les données du formulaire de façon différente, et chacune a ses propres avantages et ses inconvénients :

- ✓ **get.** Les informations sont ajoutées au bout de l'URL qui appelle le programme de traitement spécifié comme valeur de l'attribut `action`. Exemple :

```
traitement.php?nom=Dupont&prénom=Jules
```

Cette méthode a pour elle l'avantage de la simplicité et de la rapidité. En revanche, les informations ainsi transmises sont visibles dans la fenêtre du navigateur (ce qui peut constituer un problème du point de vue de la sécurité) et leur nombre est réduit.

- ✓ **POST.** Les informations sont transmises séparément au programme de traitement. Avantage : leur longueur n'est plus limitée. Inconvénient : une légère surcharge du serveur et une rapidité très légèrement dégradée.

Si ces informations ne sont pas traitées en PHP mais par un script CGI, le programme qui va s'en charger doit d'abord les récupérer et les placer dans des variables. La méthode `get` est alors plus facile à implémenter ; c'est pourquoi beaucoup de programmeurs préfèrent cette solution. Mais, comme PHP fait tout ce travail automatiquement, les deux méthodes sont aussi faciles à utiliser l'une que l'autre. Il est donc conseillé de préférer `POST` dont les avantages surpassent les inconvénients, ne serait-ce que du point de vue de la sécurité et du volume théoriquement illimité de données.

Dans certains cas, vous avez besoin d'accéder à tous les champs du formulaire (par exemple pour vérifier que l'utilisateur a bien rempli l'ensemble des zones obligatoires). Comme nous l'avons vu pour le programme `traitement.php` du Listing 8.5, une boucle `foreach` vous permettra de parcourir le tableau prédéfini `$_POST` (ou `$_GET`). La plupart des exemples de programmes et d'instructions proposés dans ce livre font appel à la méthode `POST`. Les clés sont les noms des champs. Reportez-vous à l'encadré "POST ou get ?" pour plus de détails sur ces deux méthodes.

Supposez qu'un programme contienne les instructions suivantes pour afficher un formulaire :

```
echo "<form action='traitement.php' method='POST'>\n";
echo "<input type='text' name='nom' value='Dupont'>\n";
echo "<input type='radio' name='choix' value='chien'>chien\n";
echo "<input type='radio' name='choix' value='chat'>chat\n";
echo "<input type='hidden' name='varcach' value='3'>\n";
echo "<input type='submit' value='Faites votre choix'>\n
    </form>\n";
```

et que `traitement.php` contienne :

```
foreach ($_POST as $cle => $valeur)
{
    echo "$cle, $valeur<br>";
}
```

L'affichage par la boucle `foreach` pourrait se présenter ainsi :

```
nom, Dupont
choix, chien
varcach, 3
```

L'affichage de ces valeurs s'explique de cette façon :

- ✓ **L'utilisateur n'a pas modifié le texte présent dans le champ de texte.** Aussi la valeur "Dupont" est-elle restée inchangée.
- ✓ **L'utilisateur a cliqué sur le bouton radio devant "chien".** Un seul bouton radio d'un groupe peut être choisi.
- ✓ **Une variable cachée, `varcach`, a été transmise.** L'utilisateur ne peut pas modifier sa valeur.

Contrôle des informations

Des erreurs peuvent se glisser dans ce qu'a saisi un utilisateur : erreurs de frappe ou erreurs de logique, voire fautes intentionnelles dans le but de "planter" le programme de dépouillement. Aussi est-il prudent de contrôler les informations reçues avant de les utiliser ou de les enregistrer dans la base de données. C'est ce qu'on appelle *valider* les données. Pour cela, il faut :

- ✓ **Contrôler les champs restés vides.** Certaines informations peuvent être indispensables. Il faut alors afficher un message signalant l'obligation de fournir l'information manquante et réafficher le formulaire.
- ✓ **Contrôler la forme sous laquelle une information a été saisie.** La présence de caractères alphabétiques dans un code postal ou celle de chiffres dans un nom est, à l'évidence, une erreur.

Test des champs vides

Dans un formulaire, il peut se trouver des champs dont le remplissage n'est pas obligatoire et d'autres qui sont indispensables. Vous ne devez donc vérifier que ces derniers et signaler éventuellement l'erreur.

Voici la syntaxe générique de ce genre de test :

```
if ($nom == "")
{ echo "Vous devez indiquer votre nom<br>\n";
  réafficher le formulaire
  exit();
}
echo "Bienvenue dans notre Association<br>"\n";
  afficher le menu
  ... etc ...
```

Dans l'exemple précédent, vous aurez noté la présence d'une instruction `exit` mettant fin au programme. Sans sa présence, l'exécution du programme se poursuivrait inutilement après avoir réaffiché le formulaire en produisant tout de même le message de bienvenue et le menu..

Pour tester l'ensemble des champs d'un formulaire, vous pouvez écrire une boucle explorant toutes les variables du tableau \$ _POST. Par exemple :

```
foreach ($_POST as $valeur)
{ if ($valeur == "")
{ echo "Vous n'avez pas renseigné tous les champs<br>\n";
  réafficher le formulaire
  exit;
}
}
echo "Bienvenue ...";
```



Lorsque vous réaffichez le formulaire, prenez soin d'y replacer les valeurs déjà saisies par l'utilisateur, car il serait sans doute mécontent d'avoir à les taper à nouveau et il risquerait alors de quitter votre site Web pour ne plus jamais y revenir.

Lorsque la saisie de tous les champs n'est pas obligatoire, la séquence précédente doit être modifiée pour ne pas tester les zones facultatives. Admettons que vous n'ayez pas réellement besoin du prénom de la personne, et moins encore de son numéro de fax. Vous devrez alors faire une exception pour ces champs. On pourrait alors aboutir à ceci :

```
foreach ($_POST as $champ => $valeur)
{ if ($champ != "fax" and $champ != "prénom")
{ if ($valeur == "")
{ echo "Vous n'avez pas renseigné tous les champs<br>\n";
  réafficher le formulaire
  exit;
}
}
}
echo "Bienvenue ...";
```

Le if le plus extérieur empêche l'exécution du test (le if intérieur) pour les champs relatifs au fax et au prénom, dans lesquels il est toléré de ne rien indiquer.

En général, le programme devra définir deux tableaux : un pour les noms des champs qui sont vides par erreur et un autre pour ceux dont le contenu est correct, afin de pouvoir afficher les erreurs. Vous verrez plus loin à quoi sert le second tableau.

Dans certains cas, vous pouvez souhaiter dire à l'utilisateur quels sont les champs qui doivent obligatoirement être renseignés.

Le programme `testblanc.php` (voir le Listing 8.12) traite un formulaire contenant quatre champs : prénom, nom, `code_postal` et téléphone, parmi lesquels seul le code postal peut être laissé vide. Il exploite des données saisies dans le script du Listing 8.6.

Avant de lancer le programme du Listing 8.12, vous devez retoucher le programme `afficheForm` pour qu'il l'appelle :

```
echo "<form action='testBlanc.php' method='POST'>
```

Chargez ensuite `afficheForm.php`, saisissez les données et validez pour tester `testBlanc.php`.

Listing 8.12 : Programme testant et signalant les champs vides

```
<?php
/* Nom du programme : testBlanc.php (checkBlank.php)
 * Description : Ce programme contrôle les champs du formulaire
 *              pour rechercher les zones vides.
 */
?>
<html>
<head>
<title>Test de champs vides</title>
</head>
<body>
<?php
    // Définir les champs à tester
    $labels = array ( "prenom" => "Prénom",
                     "nom" => "Nom",
                     "code_postal" => "Code postal",
                     "telephone" => "Téléphone");

    // Contrôler tous les champs sauf celui du code postal
    foreach ($_POST as $champ => $valeur)
    {
        if($champ != "code_postal")
        {
            if( $valeur == "" )
            {
                $tablovide[] = $champ;
            }
        }
    }
    // Fin de la boucle foreach pour $_POST

    // Si l'un des champs est vide, afficher un message
```

```
if(@sizeof($tablovide) > 0) // en cas de champ vide  
{  
    echo "<b>Un des champs obligatoires n'a pas été renseigné.  
        Vous devez saisir :</b><br>";  
    // affichage du nom des informations requises  
    foreach($tablovide as $valeur)  
    {  
        echo "nbsp&nbspnbspnbspnbsp$labels[$valeur]<br>";  
    }  
    // réafficher le formulaire  
    echo "<p>";  
    echo "<form action='$_SERVER[PHP_SELF]' method='POST'>  
        <table>;  
        foreach($labels as $champ => $label)  
        {  
            $dataOk[$champ]=strip_tags(trim($_POST[$champ]));  
            echo "<tr>  
                <td style='text-align: right; font-weight: bold'>  
                    {$labels[$champ]}</td>  
                <td><input type='text' name='$champ' size='65'  
                    maxlength='65' value='$dataOk[$champ]'"</td>  
                </tr>;  
        }  
        echo "<tr>  
            <td colspan='2' style='text-align: center'>  
                <input type='submit' value='Valider la saisie'>;  
        echo "</td></tr></table>  
            </form>;  
        exit();  
    }  
    echo "Tous les champs à saisie obligatoire sont renseignés.";  
?}  
</body></html>
```

Voici comment ce programme est organisé :

1. **Définition d'un tableau d'étiquettes de champs du formulaire.** Ces étiquettes seront utilisées pour afficher les noms des champs qui n'ont pas été remplis.
2. **Bouclage sur les variables passées par le formulaire et vérification des champs vides.** Les variables sont dans le tableau \$_POST. Le champ de code postal n'est pas testé car il n'est pas à saisie obligatoire. Le nom de tout champ non renseigné est placé dans le tableau \$champVide.

3. **Recherche de champ vides.** xamen du contenu du tableau `$champVide` et comptage du nombre de valeurs qu'il contient.
4. Si ce nombre est nul, sauter tout ce qui suit pour aller afficher le message comme quoi tout va bien.
5. Si ce nombre n'est pas nul :
 - a. Afficher un message d'erreur expliquant la cause générale de l'erreur.
 - b. Afficher la liste des informations manquantes. Ce sont les noms contenus dans le tableau `$champVide`.
 - c. Créer un tableau de données corrects (`dataOk`). Nettoyer ces données pour affichage correct dans le formulaire.
 - d. Afficher le formulaire. Il contient les noms des variables dans l'attribut `value` de chaque balise `<input>`. Les informations déjà saisies sont donc réaffichées.
 - e. Exécution de l'instruction `exit;`. L'utilisateur doit cliquer sur le bouton `submit` pour continuer.

La Figure 8.13 montre ce qui sera affiché si l'utilisateur n'a pas renseigné le champ obligatoire "Prénom" et le champ facultatif "Code Postal". Vous noterez que ce dernier n'est pas inclus dans la liste des oublis, puisque sa saisie est facultative et que les informations saisies au départ restent affichées dans les champs du formulaire.

Contrôle du format des informations

Lorsque l'utilisateur saisit des informations dans un formulaire, vous pouvez craindre un certain nombre de fautes de frappe. Vous pouvez détecter quelques-unes de ces erreurs et les signaler en lui demandant de ressaisir les informations concernées. Ainsi, si l'utilisateur a tapé 7612 pour un code postal, le nombre de caractères (4) vous indique déjà que cette entrée est incorrecte.



Vous devez aussi penser à vous protéger contre des utilisateurs malveillants, quelle que soit leur intention réelle. Par exemple, la saisie de balises HTML dans un champ de formulaire risquerait de conduire à un affichage erroné. Pis : la saisie d'un script dans une zone de texte équivaut à entrer un programme dans un champ de formulaire, ce qui est potentiellement dangereux.

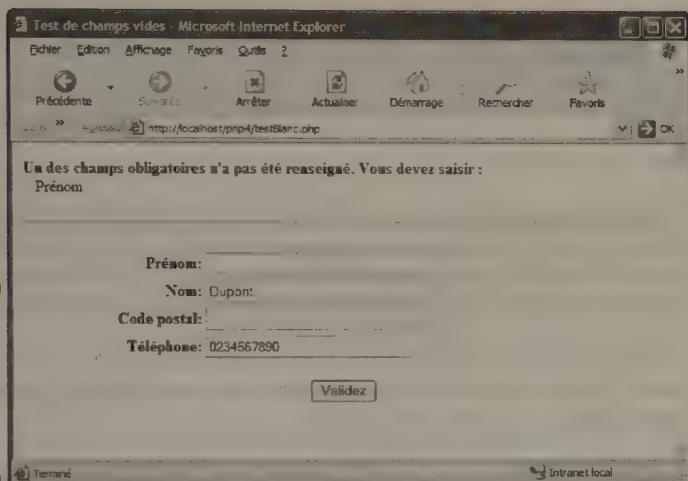


Figure 8.13 :
Résultat du
processus
lorsqu'il
manque des
informations.

L'un des moyens les plus puissants pour tester la validité d'une saisie est probablement l'utilisation d'*expressions rationnelles* (étudiées au Chapitre 6). Vous comparez le contenu d'un certain champ avec un motif. Si le test est concluant, la valeur peut être acceptée. Sinon, la donnée n'est pas valide et l'utilisateur doit la ressaisir.

En général, le test du format d'une saisie se présente sous la forme suivante :

```
if (! ereg("motif", $champ))
{ echo message d'erreur;
  ... réaffichage du formulaire ...
  exit;
}
echo "Bienvenue...";
```

Vous aurez remarqué la présence du point d'exclamation renversant le sens du test conditionnel. De cette façon, les instructions placées dans le bloc qui suit ne seront exécutées que si le test (d'erreur) est positif.

Supposons que vous souhaitiez vérifier la validité d'un nom propre. Vous allez supposer qu'il ne peut pas contenir de chiffres, mais vous devez accepter certains caractères comme le tiret (Saint-Just), l'apostrophe (d'Hauteville) ou l'espace (du Deffand). Une limite de 50 caractères est également plausible. Vous pouvez alors écrire :

```

if (! ereg("[A-Za-z' -]{1,50}", $nom)
{ echo message d'erreur;
... réaffichage du formulaire ...
exit;
}
echo "Bienvenue !";

```



Si vous voulez inclure le tiret (-) dans le groupe des caractères admis (celui qui est placé entre crochets dans le motif), vous devez le placer en tête ou en queue du groupe pour qu'il ne soit pas interprété comme un indicateur d'ensemble (A-Z, par exemple).

Il faut savoir gérer les champs multichoix, car ils constituent une porte d'entrée dérobée pour des visiteurs malveillants. Voici comment tester ce genre de champ avec regex :

```
if( !ereg("(homme|femme)", $sexe)
```

Si le champ contient autre chose que les deux valeurs testées, vous passez au bloc conditionnel if.

Dans la section précédente, nous avons vu comment tester chacun des champs d'un formulaire pour s'assurer qu'il a été renseigné (voir le Listing 8.12). Si vous voulez en outre vérifier qu'il a été saisi sous une forme acceptable, vous devez modifier le programme sous la forme que montre le Listing 8.13.

Le programme du Listing 8.13 (comme celui du Listing 8.12) traite des données saisies dans le formulaire du programme afficheForm (Listing 8.6). Avant de l'essayer, vous devez modifier afficheForm pour appeler testChamps.php dans l'attribut action de la balise form :

```
echo "<form action='testChamps.php' method='POST'>
```

Chargez ensuite la page afficheForm.php, saisissez les données et validez pour déclencher le nouveau programme testChamps.php.

Listing 8.13 : Test de champs vides et de formats de données incorrects.

```

<?php
/* Nom du programme : testChamps.php (checkAll.php)
 * Description : Ce programme contrôle les champs du formulaire
 *              (zones vides, format incorrect).
 */

```

```

?>
<html>
<head>
<title>Test de champs</title>
</head>
<body>
<?php
    /* Définir les champs à tester */
    $labels = array ( "prenom" => "Prénom",
                      "nom" => "Nom",
                      "code_postal" => "Code postal",
                      "telephone" => "Téléphone");

    foreach ($_POST as $champ => $valeur)
    {
        /* Contrôler tous les champs sauf celui du code postal */
        if ( $valeur == "" )
        {
            if ($champ != "code_postal")
            {
                $tabloVide[$champ] = "blanc";
            }
        }
        elseif ($champ == "prenom" or $champ == "nom" )
        { if (!ereg("^[A-Za-z' -]{1,50}$", $_POST[$champ]) )
          {
              $mauvaisFormat[$champ] = "mauvais";
          }
        }
        elseif ($champ == "téléphone")
        {
            if (!ereg("^[0-9]( -){7,20}(((xX)|(ext)|(ex))?[ -]?[0-9]{1,7})?$",
                      $valeur) )
            {
                $mauvaisFormat[$champ] = "mauvais";
            }
        }
    }
} // Fin de la boucle foreach pour $_POST

/* Si l'un des champs est incorrect, afficher un message */
if (@sizeof($tabloVide) > 0 or @sizeof($mauvaisFormat) > 0)
{
    if (@sizeof($tabloVide) > 0)
    {
        /* Message pour information manquante */
        echo "<b>Un des champs obligatoires n'a pas été renseigné.  

        Vous devez saisir :</b><br>";
        /* Affichage du nom des informations requises */
    }
}

```

```

foreach($tabloVide as $champ => $valeur)
{
    echo "&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;{$labels[$champ]}<br>";
} // Fin de la boucle foreach pour les champs vierges
}
if (@sizeof($mauvaisFormat) > 0)
{
    /* Message pour information invalide */
    echo "<b>Un ou plusieurs champs contiennent des informations
        qui semblent incorrectes. Corrigez le format de :</b><br>";
    /* Affiche ule liste des informations incorrectes */
    foreach($mauvaisFormat as $champ => $valeur)
    {
        echo "&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;{$labels[$champ]}<br>";
    }
}

// Réafficher le formulaire
echo "<p>";
echo "<form action='$_SERVER[PHP_SELF]' method='POST'>
    <table>";
foreach($labels as $champ => $label)
{
    $dataOk[$champ]=strip_tags(trim($_POST[$champ]));
    echo "<tr>
        <td style='text-align: right; font-weight: bold'>
            {$labels[$champ]}</td>
        <td><input type='text' name='$champ' size='65'
            maxlength='65' value='$dataOk[$champ]'></td>
        </tr>";
}
echo "<tr>
    <td colspan='2' style='text-align: center'>
        <input type='submit' value='Submit Name and Phone Number'>";
echo "</td></tr></table>
    </form>";
exit();
}

/* Test réussi */
echo "Toutes les données sont correctes.";
?>
</body></html>

```

Voici les différences entre ce programme et celui du Listing 8.12 :

- ✓ **Il existe deux tableaux : un pour les champs vides ; l'autre pour les champs incorrects.** Pour le premier, il s'agit de `$tabloVide[]` ; pour le second, `$mauvaisFormat[]`.
- ✓ **Le programme exécute une boucle sur `$mauvaisFormat[]` pour créer une liste séparée de formats incorrects.** Si un des champs est vide, il affiche un message d'erreur et une liste de ces champs. S'il y a un format incorrect, il crée un second message d'erreur pourvu, lui aussi, de sa propre liste.

La Figure 8.14 vous montre un exemple dans lequel le nom est absent, tandis que le prénom et le numéro de téléphone sont incorrects.

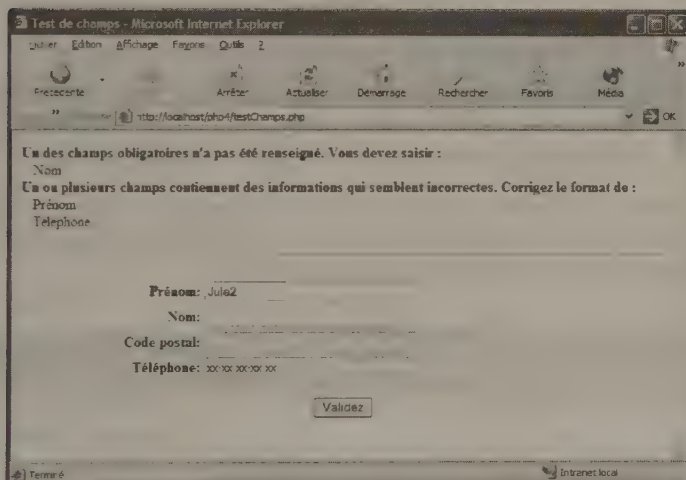


Figure 8.14 :
Autre
exemple de
validation des
données
transmises
par un
formulaire.

Plusieurs boutons submit

Dans un formulaire, vous n'êtes pas limité à un seul bouton de type Submit. Ainsi, pour un bon de commande, vous pouvez très bien prévoir deux boutons : "Envoyer la commande" et "Annuler la commande". Mais, comme vous ne pouvez avoir qu'un seul attribut action dans la balise `<form>`, il faut utiliser un autre moyen pour savoir sur quel bouton l'utilisateur a cliqué afin que PHP puisse traiter correctement cette situation. Le programme suivant va appeler le programme du Listing 8.15 :

```

<?php
/* Programme : deuxBoutonsAffiche.php (displayTwoButtons.php)
 * Description: Affiche un formulaire avec deux boutons
 *              de validation.
 */
?>
<html>
<head><title>Deux boutons (deuxBoutonsAffiche.php)</title></head>
<body>
<?php
echo "<form action='deuxBoutonsTraite.php' method='POST'>
      Nom : <input type='text' name='nom' maxlength='50'><br>
      <input type='submit' name='envoi' value='Afficher Adresse'>
      <input type='submit' name='envoi' value='Afficher Téléphone'>
      </form>";

?>
</body></html>

```

Si les deux boutons Submit ont bien le même nom, leur valeur (ce qu'ils affichent) est différente. Le programme de traitement pourra les distinguer d'après la valeur de cette variable \$envoi.

Dans le programme du Listing 8.15, le choix est offert à l'utilisateur d'afficher l'adresse d'un membre de l'association (dont il vient de saisir le nom) ou son numéro de téléphone.

Listing 8.15 : Traitement de deux boutons Submit.

```

<?php
/* Nom du programme : deux BoutonsTraite.php (processTwoButtons.php)
 * Description : Affiche des informations différentes
 *              selon le bouton submit choisi.
 */
?>

<html>
<head>
<title>Afficher l'adresse ou le numéro de téléphone</title>
</head>
<body>
<?php
    $user="admin";
    $host="localhost";
    $password="";
    $database = "MembresSeuls";

```

```

$connexion = mysqli_connect($host, $user, $password, $database)
    or die ("Connexion au serveur impossible");

if ($_POST['envoi'] == "Afficher Adresse")
{
    $req = "SELECT rue,ville,département,codePostal
        FROM Membre WHERE nom='".$_POST[nom]'" ;
    $result = mysqli_query($connexion, $req)
        or die ("Exécution de la requête impossible.");
    $ligne = mysqli_fetch_assoc($result);
    extract($ligne);
    echo "Adresse : {$ligne['rue']}<br>
        Ville : {$ligne['ville']}<br>
        N° département : {$ligne['département']}<br>
        Code postal : {$ligne['codePostal']}<br>";
}
else
{
    $req = "SELECT tph FROM Membre WHERE nom='".$_POST[nom]'" ;
    $result = mysqli_query($connexion, $req)
        or die ("Exécution de la requête impossible.");
    $ligne = mysqli_fetch_assoc($result);
    echo "Téléphone : {$ligne['tph']}<br>";
}
?>
</body></html>

```

Selon la valeur de la variable \$envoi, le programme exécute l'une des deux branches d'un if.

Insertion d'informations dans une base de données

Une application a presque toujours besoin de stocker dans une base de données des informations recueillies auprès d'un utilisateur via un formulaire approprié. La base MembreSeuls en est un bon exemple. Pour cela, il faut envoyer des requêtes SQL à MySQL (voir à ce sujet le Chapitre 4).

Préparation des informations

Cette opération préliminaire comprend trois étapes :

- ✓ Placer les données dans des variables.
- ✓ S'assurer que ces données sont dans le format attendu par la base de données.
- ✓ Nettoyer les données.
- ✓ Inactiver les données dangereuses.

Placer les données dans des variables

Pour placer des données dans des variables, il suffit d'énumérer les noms de ces variables dans la requête SQL. Avec PHP, cela ne présente aucune difficulté. Nous avons vu comment PHP enregistre les données dans une variable de type tableau utilisant le nom du champ de formulaire comme clé d'accès, cela de façon automatique et invisible. Il vous suffit alors d'utiliser les variables fournies par PHP. A l'occasion, vous pouvez avoir à ajouter d'autres données, souvent déduites de celles que vous avez reçues de l'utilisateur (comme la date du jour ou le numéro d'une commande). Là encore, le passage par une variable qui sera incluse dans la requête résoudra le problème.

Contrôler le format des données

Lors de la création de la base de données, vous avez défini un format particulier pour chacune des informations qui vont s'y trouver, autrement dit pour chaque colonne de chaque table. Il faut évidemment que le type des données que vous avez reçues soit identique. C'est notamment le cas pour les dates dont le format est défini de façon stricte et qui risque de ne pas être celui sous lequel la date a été saisie par l'utilisateur. S'il y a discordance entre le format prévu et le format reçu, la donnée sera probablement stockée mais pas nécessairement sous sa valeur correcte. Voici quelques indications sur la façon dont MySQL va conserver les différents types de données qu'il recevra :

- ✓ **CHAR** ou **VARCHAR** : Chaîne de caractères, comme la plupart des autres informations reçues par MySQL (y compris les nombres et les dates). Si la longueur spécifiée pour **CHAR** est trop courte, les caractères excédentaires seront perdus. C'est pourquoi il est bon de spécifier la même valeur de l'attribut `maxLength` dans le champ de formulaire qui recueille la donnée.



Faites coïncider la longueur maximale `maxlength` de vos champs de saisie de texte des formulaires avec la largeur des colonnes de vos tables de base de données réceptrices. L'utilisateur ne risque ainsi pas de saisir plus de caractères qu'il peut en être stockés dans la base.

- ✓ **INT ou DECIMAL** : Nombres, que cela ait ou non un sens. Ainsi, une date peut-elle être interprétée comme un nombre (2007.00, par exemple, pour l'année en cours). Si MySQL est incapable de trouver un équivalent numérique, il range la valeur 0 dans la colonne considérée.
- ✓ **DATE** : Sous forme d'une date MySQL : AAAA-MM-JJ. L'année peut être exprimée avec quatre chiffres (2007) ou seulement deux (04). La date peut être une chaîne formée de chiffres, ou encore dont les éléments sont séparés par un caractère particulier. Les séparateurs entre les trois valeurs peuvent être le tiret (-), le point (.), le slash (/) ou rien du tout. Voici quelques exemples de formes valides : 20070517, 2007-05-17, 0405017, 2007.05.17... Si la valeur passée ne peut pas être interprétée correctement, c'est 0000-00-00 qui sera enregistré.
- ✓ **ENUM** : La valeur reçue doit correspondre exactement à l'une des valeurs énumérées. Sinon, MySQL mémorise un 0.

Les informations reçues du formulaire sont souvent rangées telles quelles dans la base de données. C'est le cas, par exemple, du nom d'un utilisateur. Parfois, cependant, il est nécessaire d'opérer un traitement préalable comme pour les dates reçues à partir de trois boîtes à liste déroulantes (pour le jour, le mois et l'année, comme nous l'avons vu en étudiant les listes de sélection). L'instruction suivante rassemble ces informations :

```
$expDate = $_POST['expAnnée']."-";
$expDate .= $_POST['expMois']."-";
$expDate .= $_POST['expJour'];
```

Vous pouvez également vouloir standardiser les numéros de téléphone. La façon la plus simple de le faire (pour les numéros français) est de ne conserver que les chiffres en supprimant tous les séparateurs, ce qui peut être accompli en exécutant l'instruction suivante :

```
$téléphone = ereg_replace("[ /](.-)","",$_POST['téléphone']);
```


Ici, la fonction `ereg_replace()` utilise une expression rationnelle pour remplacer les caractères énumérés dans le motif (premier argument) par celui du deuxième argument (chaîne vide), le résultat étant renvoyé dans la variable située à gauche du signe égal. Finalement, tirets, espaces, points, parenthèses et barres obliques sont supprimés.

Nettoyage des données

Certains caractères comme les chevrons ne doivent normalement pas se trouver dans une chaîne de caractères reçue de l'utilisateur par crainte que celui-ci ne veuille glisser un script dans ce qu'il envoie.



En cas de besoin, n'hésitez pas à revoir la section "Recueil d'informations auprès de l'utilisateur", bien plus haut dans ce chapitre.

Mais, dans certains cas, ces caractères peuvent être licites (une formule mathématique, par exemple). Il existe deux fonctions prévues pour "nettoyer" les données :

- ✓ `strip_tags()` : Tout ce qui est compris entre "<" et ">" sera supprimé (ainsi que les chevrons eux-mêmes) de la chaîne de caractères passée en argument. Vous pouvez, néanmoins, spécifier les balises dont vous admettez la présence. L'instruction suivante supprime toutes les balises d'une chaîne de caractères sauf "<i>", "</i>", "" et "" :

```
$nom = strip_tags($nom, "<b></b><i></i>");
```

- ✓ `htmlspecialchars()` : Cette fonction transforme certains signes en entités de caractères pour éviter toute confusion dans un document HTML :

- < devient <
- > devient >
- & devient &

Cela permet d'afficher les chevrons dans une page Web sans risquer de les voir interprétés comme des balises. Voici un exemple d'utilisation :

```
$mot = htmlspecialchars($mot);
```

Si vous refusez absolument que vos utilisateurs puissent taper < ou > dans un champ de saisie de formulaire, utilisez `strip_tags()`. Mais si vous acceptez ces caractères sans vouloir courir de risques inutiles, utilisez `htmlspecialchars()`.

Il existe une autre fonction qui peut s'avérer très utile : `trim()`, qui supprime les espaces superflus aux deux extrémités d'une chaîne de caractères. Elle s'emploie de façon très simple :

```
$mot = trim($_POST['mot']);
```

Inactivation des données

Un visiteur avisé peut saisir des données dans un de vos formulaires de telle sorte que si ces données sont utilisées dans une requête, elles altèrent la requête et donc le résultat qui en découle. Il est possible d'infecter une requête en manipulant les guillemets et apostrophes (délimiteurs). Pour vous protéger de ces injections SQL, il faut neutraliser les délimiteurs (les "échapper") en les préfixant avec une barre oblique inverse (antibarre, *backslash*, `\`). Le couple ainsi formé empêche de considérer le délimiteur comme un symbole spécial, ce qui rend la requête insensible à tout détournement. Voyez aussi la description du type chaîne dans le Chapitre 6.

Les versions de PHP antérieures à la 6 proposaient un mécanisme appelé *magic quotes*. Lorsqu'il était activé, il préfixait systématiquement les délimiteurs dans toutes les chaînes trouvées dans les tableaux `$_POST` et `$_GET` : les apostrophes, les guillemets droits, les antibarres elles-mêmes et les caractères null. L'état du mécanisme dépend de l'option `magic_quotes-gpc` dans `php.ini`. Il est actif par défaut dans PHP 4 et PHP 5, mais n'existe plus dans PHP 6.

Le mécanisme des *magic quotes* convient bien aux programmeurs débutants pour se prémunir contre les injections malveillantes de scripts dans les requêtes SQL. Cependant, toutes les données `$_POST` et `$_GET` sont ainsi traitées, même celles qui ne seront jamais stockées dans une base, ce qui est inefficace. De plus, si vous vous contentez d'afficher des données ou de les transmettre via la messagerie, les antibarres (barres obliques inverses) avant les délimiteurs sont visibles, à moins de penser à les supprimer.

Les programmeurs chevronnés désactivent les *magic quotes* et traitent les chaînes avec des fonctions d'échappement PHP. Si vous utilisez des *magic quotes* dans des programmes exploités sous PHP

4 ou 5, vous devez retoucher les sources avant de les exécuter sous PHP 6.

PHP définit la fonction `mysqli_real_escape_string()` et sa collègue `mysql_real_escape_string()` pour traiter les données des formulaires avant d'émettre une requête MySQL. La fonction est appelée une fois qu'une connexion est établie avec le serveur MySQL. L'identifiant de connexion est transmis à la fonction avec la chaîne à traiter. Si les magic quotes sont actifs, la chaîne traitée sera traitée deux fois, ce qui peut poser problème.

Ajout de nouvelles informations

Pour ajouter de nouvelles informations dans une base de données, on utilise la requête `INSERT` que j'ai présentée au Chapitre 4. Cette requête insère une ligne de données dans une table. Sa forme générale est :

```
$rq = "INSERT INTO nomTable (col1,col2...,colx)
      VALUES ('var1','var2'... , 'varx')";
$result = mysqli_query($connexion, $rq)
      or die ("Exécution de la requête impossible.");
```

Voici par exemple les instructions pour stocker un nom et un numéro de téléphone saisis dans un formulaire :

```
$req= "INSERT INTO Membre (nom,prenom,tph)
      VALUES ('$_POST[nom]', '$_POST[prenom]',
              '$_POST[telephone]')";
$resultat = mysqli_query($connexion, $req)
      or die ("Erreur.");
```



N'insérez jamais des données directement depuis les champs du formulaire dans le tableau `$_POST`. Il faut toujours tester d'abord le format et nettoyer les données comme indiqué plus haut.

Le Listing 8.16 affiche un formulaire de saisie puis appelle le programme du Listing 8.17 qui stocke dans une base de données le nom, le prénom et le numéro de téléphone saisis dans le formulaire.

Listing 8.16 : Programme de saisie de données pour une base de données

```
<?php
/* Nom du programme: saisirTelephone.php (displayPhone.php)
```

```

* Description : Saisit des données dans un formulaire.
*/
echo "<html>
    <head><title>Saisie de numéro de téléphone</title></head>
    <body>";
$labels = array ( "prenom" => "Prénom",
                  "nom" => "Nom",
                  "tph" => "Téléphone");

echo "<h3>Saisissez votre N° de téléphone.</h3>";
echo "<form action='sauverTelephone.php' method='POST'>
    <table>\n";
/* Boucle d'affichage */
foreach($labels as $champ => $label)
{
    echo "<tr>
        <td style='text-align: right;
            font-weight: bold;'> $label</td>
        <td><input type='text' name='$champ' size='65'
            maxlength='65' ></td>
    </tr>";
}
echo "<tr>
    <td colspan='2' style='text-align: center;'>
        <input type='submit'
            value='Valider la saisie'>";
echo "</td></tr></table>
    </form>";
?>
</body></html>

```

Trois champs sont renseignés en sortie de formulaire : nom, prenom et tph.

Listing 8.17 : Programme rangeant des données dans une base de données.

```

<?php
/* Programme : sauverTelephone.php (savePhone.php)
* Description : Contrôle les champs vides et invalides
*               d'un formulaire. Les champs valides sont
*               sauvegardés dans une base de données.
*/
?>
<html>
<head><title>Numéro de téléphone des membres</title></head>

```

```
<body>
<?php
    /* tableau des légendes de champs */
    $labels = array ( "prénom" => "Prénom",
                      "nom" => "Nom",
                      "tph" => "Téléphone");

    /* Scrute les champ reçus */
    foreach($_POST as $champ => $valeur)
    {
        /* Traque les champs vide */
        if( $valeur == "" )
        {
            $tabloVide[] = $champ;
        }

        /* Teste le format des champs */
        elseif( ereg("(nom)", $champ) )
        {
            if(!ereg("[A-Za-z' -]{1,50}$", $valeur) )
            {
                $mauvaisFormat[] = $champ;
            }
        }

        elseif($champ == "tph")
        {
            if(!ereg("[0-9]{1,20}(((xX)|(ext)|(ex))?[ -]?[0-9]{1,7})?$", $valeur) )
            {
                $mauvaisFormat[] = $champ;
            }
        }
    }

} // Fin de foreach sur $_POST

/* Si problème, afficher un message et revenir au formulaire */
if(@sizeof($tabloVide) > 0 or @sizeof($mauvaisFormat) > 0)
{
    if(@sizeof($tabloVide) > 0)
    {
        /* Informations manquantes */
        echo "<b>Un champ au moins est vide.  

            Vous devez remplir :</b><br>";

        /* Liste des données manquantes */
        foreach($tabloVide as $valeur)
        {
            echo "&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;{$labels[$valeur]}<br>";
        }
    }

    if(@sizeof($mauvaisFormat) > 0)
    {
        /* Informations incorrectes */
        echo "<b>Un champ au moins contient une donnée
```



```

incorrecte. Corrigez :</b><br>";
/* Liste des champs incorrects */
foreach($mauvaisFormat as $valeur)
{
    echo "&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;{$labels[$valeur]}<br>";
}
}
/* Réaffichage du formulaire */
echo "<p><hr />";
echo "<h3>Saisissez votre N° de téléphone.</h3>";
echo "<form action='sauverTelephone.php' method='POST'>";
    <table>;
foreach($labels as $champ => $label)
{
    $dataOk[$champ]=strip_tags(trim($_POST[$champ]));
    echo "<tr>
        <td style='text-align: right; font-weight: bold;'>
            $label</td>
        <td><input type='text' name='$champ' size='65'
            maxlength='65' value='$dataOk[$champ]'"></td>
        </tr>";
}
echo "<tr>
    <td colspan='2' style='text-align: center'>
        <input type='submit' value='Valider la saisie'">";
echo "</td></tr></table>";
    </form>";
exit();
}
else // Si les données sont Ok
{
    $user="admin";
    $host="localhost";
    $password="";
    $database = "MembresSeuls";
    $cxn = mysqli_connect($host,$user,$password,$database)
        or die ("Connexion impossible");

    $champsTous = array_keys($labels);
    foreach($champsTous as $champ)
    {
        $dataOk[$champ] = strip_tags(trim($_POST[$champ]));
        if($champ == "tph")
        {
            $dataOk[$champ] = ereg_replace("[ ]( .-)", "", $dataOk[$champ]);
        }
        $dataOk[$champ] = mysqli_real_escape_string($cxn,$dataOk[$champ]);
    }
}

```

```

$req = "INSERT INTO Membre (nom,prenom,tph)
      VALUES ('$dataOk[nom]', '$dataOk[prenom]',
               '$dataOk[tph]')";
$result = mysqli_query($cxn,$req)
        or die ("Requête impossible.");
echo "<h4>Le nouveau membre a été ajouté</h4>";
}
?>
</body></html>

```

Le programme effectue les mêmes vérifications que celles qui ont été détaillées sur le Listing 8.13 `testChamps.php` (champs vides et formats incorrects). Si tout est bon, les données sont nettoyées (débarrassées des espaces excédentaires à chaque bout) puis rangées dans la base de données.

Dans une application, on peut avoir besoin de ranger des données à plusieurs reprises. Une fonction qui ferait ce travail serait donc la bienvenue. En voici une :

```

function stockerFormu($donneesForm,$nomTable, $connexion)
{
    if (!is_array($donneesForm))
    {
        return FALSE;
        exit();
    }
    foreach ($donneesForm as $champ => $valeur)
    {
        $donneesForm[$champ] = trim($donneesForm[$champ]);
        $donneesForm[$champ] = strip_tags($donneesForm[$champ]);
        if ($champ == "tph")
        {
            $donneesForm[$champ] =
                ereg_replace("[" . "-" . "]", "", $donneesForm[$champ]);
        }
        $tableau_champ[] = $champ;
        $tableau_valeur[] = $donneesForm[$champ];
    }
    $champs = implode(",", $tableau_champ);
    $valeurs = implode("'", $tableau_valeur);
    $req = "INSERT INTO $nomTable ($champs)
          VALUES (\'$valeurs\)";
    $result = mysqli_query($connexion, $req)

```

```

        or die ("La requête ne peut pas être exécutée.");
    return TRUE;
}

```

La fonction retourne TRUE si elle se termine après avoir inséré les données sans erreur. Elle commence par vérifier que le premier argument qui lui est passé est un tableau. Dans le cas contraire, elle s'arrête et renvoie FALSE.

Notez que pour que cette fonction fasse correctement son travail, il est nécessaire que les noms des champs du formulaire soient les mêmes que ceux des colonnes de la table de la base de données. On suppose également que la connexion au serveur MySQL est déjà effectuée et que la base de données a été sélectionnée.

Voici comment peut être appelée cette fonction :

```

else // Si les données sont Ok
{
    $user="admin";
    $host="localhost";
    $password="";
    $database = "membresSeuls";
    $cxn = mysqli_connect($host,$user,$password,$database)
        or die ("Erreur");
    if(stockerFormu($dataOk,"tph",$cxn))
        echo "Nouveau membre ajouté<br>";
    else
        echo "Membre non ajouté !<br>";
}
?>
</body></html>

```

Vous aurez remarqué que ce programme est bien plus facile à relire, car une bonne partie des instructions se trouve maintenant dans la fonction. En outre, cette fonction peut marcher avec n'importe quel formulaire, aussi longtemps que les noms de champs de ce formulaire sont les mêmes que ceux des colonnes de la table de la base de données. S'il survient un incident lors de l'exécution de la requête, un message d'erreur doit être affiché.

Mise à jour des informations existantes

On utilise pour cela la requête UPDATE présentée au Chapitre 4. Une mise à jour consiste à actualiser les données des colonnes, et non

pas à ajouter de nouvelles lignes à une ou plusieurs tables. Sa forme générale est :

```
$req = "UPDATE nomTable SET col=valeur WHERE col=valeur";
$result = mysqli_query($connexion, $req)
    or die ("Exécution de la requête impossible.");
```

Par exemple, les instructions pour mettre à jour le numéro de téléphone de Jules Dupont pourraient se présenter ainsi :

```
$prénom = "Jules";           // lu dans le champ du formulaire
$nom = "Dupont";             // lu dans le champ du formulaire
$tph = "01-02-03-03-05";    // lu dans le champ du formulaire
$req = "UPDATE Membre SET tph='$tph'
        WHERE prénom='$prénom'
        AND nom='$nom'";
$result = mysqli_query($connexion, $req)
    or die ("Exécution de la requête impossible.");
```



Si vous ne faites pas figurer une clause `WHERE` dans la requête, le champ défini par `SET` sera modifié dans toutes les lignes de la table. C'est très rarement ce que vous souhaitez faire.



N'insérez jamais des données directement depuis les champs du formulaire dans le tableau `$_POST`. Il faut toujours tester d'abord le format et nettoyer les données comme indiqué plus haut.

Le Listing 8.18 présente un programme nommé `majTelephone.php` destiné à mettre à jour dans une base de données un numéro de téléphone, en se repérant sur le couple prénom/nom également saisi dans le même formulaire que l'exemple précédent, c'est-à-dire le programme `saisirTelephone.php`. Pour le réutiliser, il suffit d'appliquer une retouche au niveau de l'attribut `action` de la balise `form` :

```
echo "<form action='majTelephone.php' method='POST'>
```

Listing 8.18 - Programme modifiant le numéro de téléphone d'un membre identifié par son prénom et son nom.

```
<?php
/* Programme : majTelephone.php (updatePhone.php)
 * Description : Met à jour un N° de téléphone dans la table de base
 *               pour le membre spécifié.
 */
?>
```

```

<html>
<head><title>Mise à jour de table (majTelephone.php)</title></head>
<body>
<?php
    $labels = array ( "prenom" => "Prénom",
                      "nom" => "Nom",
                      "tph" => "Téléphone");

    foreach ($_POST as $champ => $valeur)
    {
        if ( $valeur == "" )
        {
            $tabloVide[] = $champ;
        }
    }

    /* Contrôle du format du N° */
    if(!ereg("^[0-9]( - ){7,20}([xX]|(ext)|(ex))?[ -]?[0-9]{1,7})?$",$_POST['tph']))
    {
        $mauvaisFormat[] = "tph";
    }

    /* Si problème, message et retour au formulaire */
    if (@sizeof($tabloVide) > 0 or @sizeof($mauvaisFormat) > 0)
    {
        if (@sizeof($tabloVide) > 0)
        {
            echo "<b>Il faut saisir :</b><br>";
            foreach($tabloVide as $valeur)
            {
                echo "&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;{$labels[$valeur]}<br>";
            }
        }

        if (@sizeof($mauvaisFormat) > 0)
        {
            echo "<b>Le format du N0 de téléphone est incorrect.<br>";
        }

        echo "<p><hr>";
        echo "<h3>Saisissez votre N°.</h3>";
        echo "<form action='majTelephone.php' method='POST'>
            <table>";

        foreach($labels as $champ=>$label)
        {
            $dataOk[$champ] = strip_tags(trim($_POST[$champ]));
            echo "<tr>
                <td style='text-align: right; font-weight: bold'>
                {$labels[$champ]}</td>
                <td><input type='text' name='$champ' size='65'
                    maxlength='65' value='$_POST[$champ]'"></td>
            </tr>";
        }
    }

```



```

    }
    echo "<tr>
        <td colspan='2' style='text-align: center'>
            <input type='submit' value='Valider la saisie'>";
    echo "</td></tr></table>
        </form>";
    exit();
}
else // Données correctes
{
    $dataOk['tph'] = strip_tags(trim($_POST['tph']));
    $dataOk['tph'] = ereg_replace(")(. -)", "", $dataOk['tph']);

    $user="admin";
    $host="localhost";
    $password="";
    $database = "MembresSeuls";
    $cxn = mysqli_connect($host,$user,$password,$database)
        or die ("Connexion impossible");
    $req = "UPDATE Membre SET tph='$dataOk[tph]'
        WHERE nom='$_POST[nom]'
        AND prenom='$_POST[prenom]'";
    $result = mysqli_query($cxn,$req)
        or die ("Requête impossible : ".mysqli_error($cxn));
    if(mysqli_affected_rows($cxn) > 0)
    {
        echo "<h3>N° de téléphone de {$_POST['prenom']} {$_POST['nom']}
            mis à jour</h3>";
    }
    else
        echo "Aucune mise à jour réalisée !";
}
?>
</body></html>

```

Ce programme est presque identique à celui du Listing 8.17, à ceci près qu'on ne crée par une nouvelle ligne (par INSERT) mais qu'on modifie une entrée existante (avec UPDATE). Si tout est bon, les données sont nettoyées (débarrassées des espaces excédentaires à chaque bout) puis rangées dans la base de données.



Si vous repérez des barres obliques inverses (\) dans la base après une insertion ou une mise à jour, c'est que vos données ont été "échappées" deux fois. C'est sans doute lié au fait que l'option magic quotes est active tout en utilisant `mysqli_real_escape_quotes`.

Désactivez `magic quotes`. Il faut que la requête puisse bien s'exécuter sans pour autant provoquer le stockage de caractères d'échappement parasites dans la base.

Transférer des informations par fichier

Votre site peut proposer à ses utilisateurs de lui envoyer des fichiers : des CV si vous leur proposez un emploi, des images si vous gérez des albums photo, et ainsi de suite. Supposons encore que vous deviez réaliser le catalogue des ventes en ligne du magasin AnimaLux. En plus du texte décrivant chaque produit dans le catalogue, vous avez besoin de photographies d'accompagnement. Vous placez alors un formulaire qui permet aux gens d'AnimaLux de vous transmettre ces images sans que personne n'ait à bouger de chez lui.

Utiliser un formulaire pour charger un fichier

Un formulaire conçu à cet effet peut proposer aux utilisateurs de transférer un fichier. La méthode générale est la suivante :

```
<form enctype="multipart/form-data"
      action="traitement.php" method="POST">
  <input type="hidden" name="MAX_FILE_SIZE" value="30000">
  <input type="file" name="fichier_utilisateur">
  <input type="submit" value="Envoyez le fichier">
</form>
```

Précisons tout cela :

- ✓ **L'attribut `enctype` débute la balise `form`.** Il doit prendre la valeur indiquée ci-dessus, `multipart/form-data`, pour s'assurer que le fichier téléchargé sera traité comme il faut.
- ✓ **Un champ caché envoie la valeur (en octets) du paramètre `MAX_FILE_SIZE`.** Si l'utilisateur tente de transférer un fichier dont la longueur est supérieure à cette donnée, le téléchargement échouera. La taille autorisée peut atteindre 2 Mo. Si cela n'est pas suffisant, vous devrez d'abord modifier manuellement la valeur par défaut enregistrée dans `php.ini`, à la ligne `upload_max_filesize`. Ajuster ensuite le paramètre `MAX_FILE_SIZE` dans votre formulaire.

- ✓ **Le champ de saisie qui effectue le transfert du fichier possède le type `file`.** Vous remarquerez qu'il possède un nom, en l'occurrence `fichier_utilisateur`, exactement comme n'importe quel autre type de champ de formulaire. Le nom de fichier saisi par l'utilisateur est envoyé au programme de traitement qui pourra le récupérer dans un tableau natif appelé `FILES`. Nous verrons cela plus en détail dans la section suivante.

Lorsque l'utilisateur valide le formulaire, le fichier est envoyé vers un emplacement temporaire. Le script qui traite le formulaire doit alors copier ce fichier vers sa destination finale (ou décider d'une autre action). En effet, le document temporaire est détruit dès que le script se termine.

Traiter le fichier téléchargé

Les informations concernant le fichier transféré sont mémorisées dans un tableau natif de PHP appelé `$_FILES`. Il s'agit d'un tableau à plusieurs dimensions, puisque plusieurs références de fichiers sont susceptibles d'y être stockées. Comme d'habitude, vous pouvez récupérer les informations contenues dans `$_FILES` en spécifiant le nom du champ. La structure principale se présente ainsi :

```
$_FILES['nomchamp']['name']
$_FILES['nomchamp']['type']
$_FILES['nomchamp']['tmp_name']
$_FILES['nomchamp']['size']
```

Supposons que vous utilisiez dans votre formulaire le champ suivant pour proposer un transfert de fichier :

```
<input type="file" name="fichier_utilisateur">
```

Si l'utilisateur envoie un fichier de texte appelé `test.txt`, le tableau résultant, exploitable par le programme de traitement, pourrait ressembler à ceci :

```
$_FILES[fichier_utilisateur]['name'] = test.txt
$_FILES[fichier_utilisateur]['type'] = text/plain
$_FILES[fichier_utilisateur]['tmp_name'] = D:\UPLOAD\php92C.tmp
$_FILES[fichier_utilisateur]['size'] = 435
```

Ici, `name` est le nom du fichier qui a été transféré. Sa nature est décrite dans `type`. L'emplacement de l'enregistrement temporaire se

trouve dans `tmp_name` (vous remarquerez que le chemin d'accès est totalement développé). On note enfin que la taille du fichier (`size`) est de 435 octets.

Si la taille du fichier excède la limite autorisée, `tmp_name` contiendra `none` et `size` vaudra 0. Le programme de traitement doit déplacer le fichier temporaire vers son emplacement définitif. Le format général de l'instruction correspondante se présente ainsi :

```
move_uploaded_file(chemin/fichierTmp,chemin/fichierPerm);
```

Le premier argument est disponible dans l'élément de tableau `$_FILES['champ']['tmp_name']`. Le second argument, `chemin/fichierPerm`, définit le chemin d'accès complet pour la version permanente du fichier. Avec les intitulés précédents, le programme de traitement pourrait exécuter par exemple l'instruction ci-dessous :

```
move_uploaded_file($_FILES['fichier_utilisateur']['tmp_name'],
    'c:\data\nouveau_fichier.txt');
```

Le répertoire de destination (ici, `c:\data`) doit exister *avant* de pouvoir y déplacer le fichier. Autrement dit, cette instruction ne crée pas ce répertoire.

Le transfert en action

Le Listing 8.19 montre un exemple de script complet. Le programme affiche un formulaire proposant à l'utilisateur d'envoyer un fichier. Il sauvegarde celui-ci avant d'afficher un message signalant la réussite de l'opération. Ce programme se charge donc à la fois d'afficher le formulaire et de le traiter. Il ne s'intéresse qu'aux fichiers graphiques dont le type est reconnu par le système (type MIME /image) et il contrôle la présence d'une extension adaptée (ce qui n'empêche pas d'ailleurs le transfert de fichiers remplis de mauvaises intentions). Le Listing 8.20 constitue le formulaire de saisie du nom de fichier (code HTML). Enfin, la Figure 8.15 illustre la page Web affichée par cet exemple.

Listing 8.19 : Chargement de fichier avec la méthode Post

```
<?php
/* Nom du script : chargeFich.php (uploadFile.php)
```

```

* Description : Chargement de fichier via HTTP
*               en utilisant un formulaire POST.
*/
if(isset($_POST['Transfert']))                                #1
{
    include("form_charge.inc");
} # endif
else                                                         #2
{
    if($_FILES['pix']['size'] == 0)                            #3
    {
        echo "<b>Le chargement a échoué. Vérifiez la taille
            du fichier. Elle doit être inférieure à 500 Ko.<br>";
        include("form_charge.inc");
        exit();
    }
    if(!ereg("image",$_FILES['pix']['type']))                  #4
    {
        echo "<b>Le fichier envoyé n'est pas une image graphique.
            Essayez avec un autre fichier.</b><br>";
        include("form_charge.inc");
        exit();
    }
    else                                                       #5
    {
        $destination = 'c:\data'. "\\" . $_FILES['pix']['name'];
        $temp_file = $_FILES['pix']['tmp_name'];
        move_uploaded_file($temp_file,$destination);
        echo "<p><b>Transfert du fichier réussi :</b>
            {$_FILES['pix']['name']}
            ({$_FILES['pix']['size']})</p>";
    }
}
?>

```

Certaines lignes du script sont signalées dans le listing par un numéro. Vous pouvez vous y reporter pour suivre les explications ci-dessous :

- 1 Cette instruction if vérifie si le formulaire a déjà été soumis. Dans le cas contraire, il est affiché en incluant le fichier où se trouve le code correspondant. Le fichier inclus est proposé dans le Listing 8.18.

- 2 La ligne débute un bloc `else` qui s'exécute si le formulaire a été soumis. On trouve ici le reste du script, qui sert notamment à traiter le formulaire et le fichier envoyé.
- 3 Cette ligne est une instruction `if` qui teste le bon chargement du fichier. En cas de problème, un message d'erreur est émis et le formulaire est réaffiché.
- 4 Cette instruction `if` vérifie si le fichier envoyé est une image d'un type reconnu. Dans le cas contraire, on émet un message d'erreur et le formulaire est réaffiché.
- 5 Cette ligne débute un bloc `else` qui s'exécute si le fichier a correctement été récupéré. Le fichier est alors déplacé vers sa destination finale (ici, le dossier `c:\data` qui doit exister par avance), et un message signale la bonne fin de l'opération.

Le Listing 8.20 montre le fichier inclus servant à afficher le formulaire.

Listing 8.20 : Fichier inclus affichant le formulaire de chargement de fichier

```

<!-- Nom du programme : form_charge.inc (form_upload.inc)
      Description :Formulaire de transfert de fichier -->
<html>
<head><title>Envoyez votre fichier !</title></head>
<body>
<ol><li>Entrez le nom du fichier contenant l'image du produit
      que vous voulez envoyer, ou utilisez le bouton Parcourir
      pour localiser l'emplacement de ce fichier.</li>
      <li>Lorsque le chemin d'accès exact vers le fichier apparaît
      dans le champ, cliquez sur le bouton Envoyez l'image.</li>
</ol>
<div align="center"><hr>
<form enctype="multipart/form-data"
      action="chargeFich.php" method="POST">
  <input type="hidden" name="MAX_FILE_SIZE" value="500000">
  <input type="file" name="pix" size="60">
  <p><input type="submit" name="Transfert"
      value="Envoyer l'image">
</form>
</body></html>

```

Vous noterez que ce fichier ne contient aucun code PHP, uniquement des balises HTML.

Le formulaire qui permet à l'utilisateur de sélectionner un fichier à envoyer est illustré sur la Figure 8.15. Il propose un champ de texte pour la saisie du nom du fichier, et un bouton Parcourir facilitant si besoin est la navigation vers l'emplacement de l'image.

Figure 8.15 :
Formulaire
permettant à
l'utilisateur
d'envoyer un
fichier
graphique.

Envoyez votre fichier 1 - Microsoft Internet Explorer

Fichier Edition Affichage Plugins Outils ?

Précédente Suivante Arrêter Actualiser Démarage Recherche Favors Média

1. Entrez le nom du fichier contenant l'image du produit que vous voulez envoyer, ou utilisez le bouton Parcourir pour localiser l'emplacement de ce fichier.

2. Lorsque le chemin d'accès exact vers le fichier apparaît dans le champ, cliquez sur le bouton Envoyer l'image

Parcourir

Envoyer l'image

Terminé Intranet local

Chapitre 9

Transfert d'informations d'une page Web à l'autre

Dans ce chapitre :

- Parcours des pages Web par l'utilisateur.
- Transfert d'informations d'une page à l'autre.
- Ajout d'informations au bout d'une URL.
- Regard sur les cookies.
- Les champs cachés de formulaire.
- Les sessions PHP.

La plupart des sites Web comprennent plusieurs pages. Lorsque ces pages sont statiques, il suffit à l'utilisateur de cliquer sur un lien pour aller de l'une à l'autre. Mais les informations ne suivent pas, car chaque page est indépendante de la précédente et de la suivante. Avec les pages dynamiques, il peut s'avérer nécessaire de conserver des informations d'une page à l'autre. Les développeurs expérimentés y parviennent par diverses astuces, par exemple au moyen de formulaires HTM, de scripts CGI (*Commun Gateway Interface*) ou encore de cookies. Toutefois, PHP est un outil bien plus puissant pour passer des informations entre les pages.

Parcours des pages Web par l'utilisateur

Alors qu'avec HTML pur et dur vous n'avez que les liens pour aller d'une page à l'autre, avec PHP vous disposez de trois moyens :

- ✓ **Les liens** : C'est la balise `<a>` (via une instruction `echo`) qui propose un lien à l'utilisateur. Le format général se présente ainsi :

```
<a href="nouvellepape.php">Texte du lien</a>
```

Lorsque l'utilisateur clique sur le texte du lien, le serveur Web envoie le fichier `nouvellepape.php` au navigateur. Cette méthode est employée de façon extensive dans les pages Web écrites en HTML. Si vous avez besoin de réviser un peu ce langage, reportez-vous à un ouvrage spécialisé : cette collection n'en manque pas !

- ✓ **Boutons de validation Submit de formulaire** : Une page peut comporter un ou plusieurs formulaires et chacun peut proposer un ou plusieurs boutons Submit (envoyez !). Lorsque l'utilisateur clique sur l'un d'eux, la page dont l'URL est indiquée comme valeur de l'attribut `action` du formulaire est envoyée au navigateur. Il n'est pas indispensable qu'un formulaire contienne des champs, mais il lui faut au moins un bouton Submit.
- ✓ **La fonction `header()`** : Celle-ci envoie un message au serveur Web lui demandant d'envoyer une nouvelle page sans que l'utilisateur ait à cliquer sur quelque bouton ou lien que ce soit. (*header* peut se traduire en français par *en-tête*).

La fonction d'en-tête PHP `header()` sert à transmettre une nouvelle page au navigateur du visiteur. Aucune action de sa part n'est requise. Une fois l'instruction `header()` exécutée, la nouvelle page est affichée. La forme générale de la fonction `header()` est la suivante :

```
header("Location: URL");
```

("Location:" est un mot clé qui doit être écrit exactement de cette façon.) Le fichier situé à l'emplacement décrit par URL est alors envoyé au navigateur. Voici deux exemples d'utilisation de cette fonction :

```
header("Location: nouvellepape.php");
```

```
header("Location: http://www.monserveur.com/catalog.php");
```

Ceci dit, l'emploi de la fonction `header()` souffre d'une sérieuse limitation : son exécution doit **précéder** l'envoi de toute sortie au navigateur. Vous ne pouvez donc plus l'appeler à partir du moment où vous avez exécuté une seule instruction `echo`. Lisez à ce sujet l'encadré *Instructions devant précéder l'envoi d'informations au navigateur*.

Malgré cette limitation, cette fonction s'avère utile. Vous pouvez exécuter n'importe quelle instruction PHP avant de l'appeler, à condition que cette instruction n'envoie aucune information à afficher au navigateur. Exemple :

```
<?php
    if ($âge < 13)
    { header("Location: CatalogJouets.php");
    }
    else
    { header("Location: CatalogJardin.php");
    }
?>
```

Ces instructions choisissent le catalogue à afficher selon l'âge indiqué par le visiteur de la page.

Transfert d'informations d'une page à l'autre

Les applications Web dynamiques consistent presque toujours en une suite de pages que va successivement parcourir le visiteur. Ce parcours constitue ce qu'on appelle une *session*. Voici quelques exemples de cas où des informations doivent être conservées pendant toute la durée de la session :

- ✓ **Site Web à accès restreint.** Lorsqu'un visiteur doit indiquer son nom de login et son mot de passe pour voir une page, on ne peut pas lui demander de les redonner à chaque page. Ces deux informations (entre autres) doivent donc être conservées tout au long de la session.
- ✓ **Adaptation à différents navigateurs.** Différents navigateurs affichent différemment les mêmes pages. Une fois identifié le navigateur utilisé par le visiteur, vous devez conserver cette information pour lui envoyer uniquement des pages contenant des balises que son navigateur est capable d'interpréter correctement.

Avec PHP, il existe plusieurs moyens de transporter ces informations d'une page à l'autre :

- ✓ **Ajouter les informations à la suite d'une URL.** Comme vous avez pu le lire dans l'encadré traitant des URL, on peut ajouter des informations à cet endroit. C'est facile (et d'ailleurs

couramment employé), mais cette technique convient surtout lorsqu'il y a peu d'informations à faire transiter.

- ✓ **Stocker les données dans des cookies.** Les *cookies* sont des petits fichiers texte envoyés par le serveur Web sur le disque de l'utilisateur. Ils peuvent contenir des couples *nom=valeur*. Ils sont normalement sans danger, mais il y a néanmoins de nombreux utilisateurs qui les refusent. Cette méthode est donc à déconseiller pour le passage de données dynamiques. Mais rien n'interdit à votre site de refuser d'entrer aux personnes qui refusent les cookies !
- ✓ **Utiliser des formulaires HTML.** Cela revient à peu près au même que la première méthode, mais c'est le navigateur (au moment où l'utilisateur clique sur le bouton *Submit*) qui va se charger de confectionner l'URL comportant les informations à passer. Et si l'attribut *method* du formulaire est *post*, le nombre d'informations à faire suivre n'est plus limité comme avec *method="get"*. Cette approche est conseillée lorsque vous devez recueillir des informations auprès des visiteurs.
- ✓ **Mettre en œuvre le mécanisme des sessions proposé par PHP.** Cette nouveauté est apparue avec la version 4 de l'interpréteur PHP. Elle fait appel à des fonctions qui prennent en charge la session de l'utilisateur et enregistrent les informations voulues sur le serveur. Une page Web peut ensuite facilement y accéder. Cette méthode est très pratique, surtout si le nombre de pages susceptibles d'être affichées est important.

Ajout d'informations à la suite d'une URL

L'un des moyens les plus simples de "faire suivre" des informations est de les ajouter sous la forme de couples *nom=valeur* à l'extrémité d'une URL. *nom* représente un nom de variable privé de son \$ initial. C'est PHP qui ajoutera ce caractère lorsqu'il recevra ces informations. *valeur* est la valeur que contient la variable. Vous pouvez ainsi mettre bout à bout plusieurs couples ayant cette structure. Le premier doit être joint à l'URL par un point d'interrogation. Les suivants seront liés les uns aux autres par un et commercial (&). Voici quelques exemples :

```
<form action="pagesuivante.php?dept=Oise" method="post">
```

```
<a href="pagesuivante.php?dept=Oise">Page suivante</a>
```

```
header("Location: pagesuivante.php?dept=Oise&ville=Creil" method="post");
```

Deux raisons peuvent conduire à limiter le passage d'informations via une URL :

- ✓ **Sécurité** : L'URL est affichée avec les paramètres passés à sa suite dans la fenêtre d'adresse du navigateur. S'il s'agit d'informations confidentielles, mieux vaut qu'elles ne soient pas ainsi exposées à la vue du public. Si vous transportez un mot de passe d'une page à la suivante, tout un chacun (enfin, ceux qui regardent par-dessus votre épaule) pourra en prendre connaissance. N'oubliez pas également que l'utilisateur peut enregistrer l'URL dans ses signets (ses *favoris*, pour Internet Explorer).
- ✓ **Longueur de la chaîne de caractères ainsi passée** : Il y a une limite au nombre de caractères qui peuvent être ainsi passés. C'est une restriction du navigateur. Dès que ce nombre est atteint, les caractères excédentaires sont perdus.

Le programme du Listing 9.1 montre comment on peut utiliser l'URL pour voir s'il s'agit ou non du premier affichage du formulaire. Le programme affiche le numéro de téléphone associé à l'identifiant courant et propose au visiteur de le modifier.

Listing 9.1 : Programme permettant à un visiteur de modifier son numéro de téléphone dans une base de données.

```
<?php
/* Nom du programme : changerTel.php (changePhone.php)
 * Description : Affiche le numéro de téléphone lu dans la base de
 *              données et permet à l'utilisateur de le modifier.
 */
?>
<html>
<head><title>Modification d'un numéro de télétp</title></head>
<body>
<?php
    $host="localhost";
    $user="admin";
    $password="";
    $database="MembresSeuls";
    $identifiant= "visiteurtruc"; // De la page précédente
    $cxn = mysqli_connect($host,$user,$password,$database)
        or die ("Problème de connexion");
```

```

if (@$_GET['first'] == "NON") #1
{
    $tph = trim($_POST['tph']);
    if (!ereg("^([0-9]{7,20})$", $tph) or $tph == "") #2
    {
        echo "<h3 style='text-align: center'>
            Numéro de téléphone incorrect.</h3>";
        display_form($identifiant, $tph); #3
    }
    else // N° de téléphone Ok #4
    {
        $query = "UPDATE Membre SET tph='$tph'
            WHERE identifiant='$identifiant'";
        $result = mysqli_query($cxn, $query)
            or die ("Couldn't execute query.");
        echo "<h3>Phone number has been updated.</h3>";
        exit();
    }
}
else // Premier affichage du formulaire #5
{
    $query = "SELECT tph FROM Member WHERE identifiant='$identifiant'";
    $result = mysqli_query($cxn, $query)
        or die ("Requête en échec.");
    $ligne = mysqli_fetch_row($result);
    $tph = $ligne[0];
    afficherFormu($identifiant, $tph); #6
}

function afficherFormu($identifiant, $tph) #7
{
    echo "<div style='text-align: center'>";
    echo "<form action='changePhone.php?first=NON' method='POST'>
        <h4>Contrôle du N° de téléphone et
        correction si nécessaire.</h4><hr />
        <p><b>$identifiant</b> <input type='text' name='tph'
            maxlength='20' value='$tph'></p>
        <p><input type='submit' value='Valider la saisie'></p>
        </form>";
    echo "</div>";
}
?>
</body></html>

```

Remarquez plus spécialement les points suivants :

- ✓ **C'est le même programme qui affiche et traite le formulaire.** Le nom de ce programme est `changerTel.php` et vous retrouvez ce nom dans le paramètre `action` de la balise `<form>`. C'est donc lui qui sera rappelé lorsque l'utilisateur cliquera sur le bouton "Nouveau numéro de téléphone".
- ✓ **Une information est ajoutée à la fin de l'URL.** On lit en effet :

```
action='http://localhost/changerTel.php?premier=NON'
```

Quand l'utilisateur clique sur le bouton "Nouveau numéro de téléphone", `changerTel.php` est exécuté pour la seconde fois et la variable `$first` prend la valeur "NON".

- ✓ **La variable `$first` est testée au début du programme (repère 1).** Elle est récupérée à l'aide du tableau natif `$_GET`. On peut ainsi savoir si on est dans la première ou dans la seconde exécution.
- ✓ **Si `$_GET['premier']` vaut "NON", le numéro de téléphone est testé.** Cette situation n'est possible que si le formulaire a été soumis.
 - a. Si le numéro est valide, il est rangé dans la base de données et le programme se termine (repère 4).
 - b. S'il est incorrect ou absent, un message est affiché puis le formulaire est réaffiché (repère 2).
- ✓ **Si `$_GET['first']` ne vaut pas "NON", le numéro de téléphone est extrait de la base de données.** C'est donc, en effet, la première exécution du programme. Il faut aller lire le numéro dans la base (repère 5).
- ✓ **Le programme définit une fonction pour afficher le formulaire (repère 7).** Cette fonction est appelée aux repères 3 et 6.

La Figure 9.1 montre comment se présente la page Web la première fois que `changerTel.php` s'exécute. Vous remarquerez que l'URL que l'on voit dans la fenêtre d'adresse n'est suivie d'aucune information.

La Figure 9.2 illustre ce qui s'affiche lorsque la saisie initiale de l'utilisateur est incorrecte. Vous constaterez que l'URL que l'on voit dans la fenêtre d'adresse est maintenant suivie de `"?first=NON"`.

Figure 9.1 :
Formulaire
HTML pour
mettre à jour
un numéro de
téléphone.

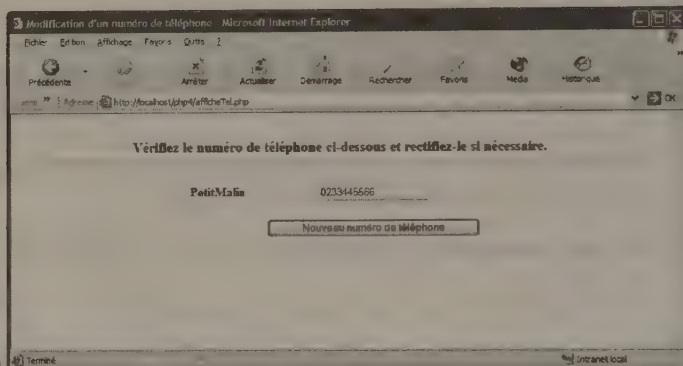
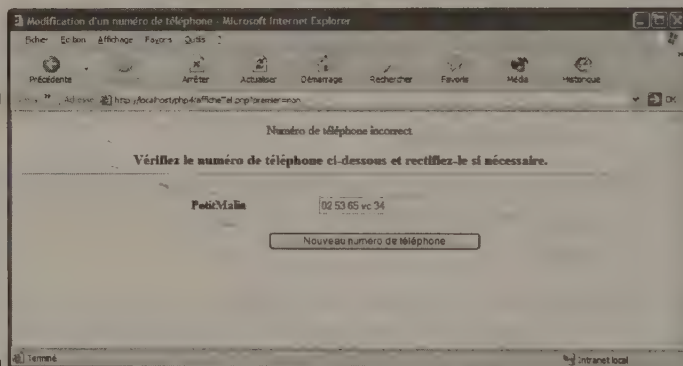


Figure 9.2 :
Message
d'erreur
affiché
lorsque la
saisie de
l'utilisateur
est
incorrecte.



Utilisation d'un cookie pour conserver des informations

Les *cookies* sont de petits fichiers texte contenant des paires *variable=valeur* semblables à celles que l'on peut adjoindre à une URL. Ils sont rangés sur le disque dur de l'utilisateur par le navigateur de celui-ci. Votre application peut alors récupérer un cookie depuis n'importe quelle page Web. Certaines personnes s'en méfient. Peut-être cela vient-il de ce nom un peu étrange, dont l'origine est d'ailleurs un des grands mystères de l'univers.

De prime abord, les cookies semblent pouvoir résoudre le problème du partage de données entre plusieurs pages Web. Ils peuvent même être conservés plus longtemps que la durée d'une session et retrouvés lorsque l'utilisateur revient plus tard (éventuellement un mois

après) sur le même site Web. Le hic est que l'utilisateur est libre de les refuser ou, s'il accepte, de les supprimer plus tard, ce que font bon nombre d'internautes, inquiets de voir quelqu'un qu'ils ne connaissent pas stocker des données (dont ils ignorent le contenu) sur *leur* machine. C'est là une attitude fort compréhensible. Dans ces conditions, on comprend que cette solution, pour intéressante qu'elle paraisse, ne soit pas la plus sûre.

Pour enregistrer un cookie, on a recours à la fonction `setcookie()` dont voici la forme générale :

```
setcookie("variable", "valeur");
```

La variable doit apparaître ici privée de son \$ initial. C'est PHP qui le rétablira lorsqu'il récupérera le cookie. Sous cette forme, le cookie n'est pas persistant (il n'est pas écrit sur le disque dur, mais seulement conservé dans la mémoire de l'ordinateur de l'utilisateur). Il disparaît à la fin de la session. Par exemple, l'instruction suivante :

```
setcookie("département", "Orne");
```

rangera la chaîne de caractères "Orne" dans une variable appelée \$département. Dans toutes les pages de votre application, cette information sera accessible aux programmes via un tableau natif, sous la forme :

```
$_COOKIE['département']
```

PHP récupère automatiquement cette valeur. Un cookie n'est pas accessible dans le programme qui l'a créé, mais seulement à partir du moment où l'utilisateur a changé de page (ou qu'il réaffiche cette page).



Dans la version 4.1 ou antérieure de PHP, vous accédiez aux cookies dans un tableau long appelé `$HTTP_COOKIE_VARS` dont les clés sont les noms des cookies. Ces tableaux longs n'existent plus dans PHP 6. Pour exploiter d'anciens scripts sous PHP 6, vous devez changer les noms de tableaux de `$HTTP_COOKIE_VARS` en `$_COOKIE`.

Si vous voulez que les informations des cookies soient persistantes (qu'elles subsistent après la fin de la session), vous définirez leur date d'expiration de la façon suivante :

```
setcookie("variable", "valeur", date_d_expiration);
```

Cette date est généralement obtenue par un appel aux fonctions `time()` ou `mktime()` de la façon suivante :

- ✓ `time()`. Cette fonction renvoie le groupe date/heure de l'instant présent sous une forme qui n'est lisible que par l'ordinateur. Il faut lui ajouter un délai qui sera exprimé en secondes. Exemples :

```
setcookie("département", "Orne", time()+3600); // dure 1 heure
```

```
setcookie("département", "Orne", time()+3*86400); // dure 3 jours
```

- ✓ `mktime()`. Cette fonction renvoie un groupe date/heure représentant une certaine date sous une date qui n'est lisible que par l'ordinateur. Il faut lui spécifier les coordonnées temporelles dans cet ordre : heures, minutes, secondes, mois, jour, année. Si une de ces valeurs est omise, elle est remplacée par celle de l'instant présent. Exemples :

```
setcookie("département", "Orne", mktime(3,0,0,4,1,2004)  
// expirera le 1er avril 2004 à 3 heures du matin
```

```
setcookie("département", "Orne", mktime(16,0,0,,,  
// expirera aujourd'hui à 16 heures
```

Pour supprimer un cookie, il suffit de lui donner une valeur nulle de l'une des deux façons suivantes :

```
setcookie("nom");  
setcookie("nom", "");
```



La fonction `setcookie()` présente un gros inconvénient : son appel doit *précéder* toute sortie à destination du navigateur. Vous ne pouvez pas l'appeler dans le cours d'un programme après avoir déjà envoyé quelque chose au navigateur de l'utilisateur. Consultez l'encadré *Instructions devant précéder l'envoi d'informations au navigateur*, plus haut dans ce même chapitre.

Passage d'informations via un formulaire HTML

La façon la plus courante de passer des informations d'une page à l'autre est de se servir d'un formulaire. Lorsque l'utilisateur clique sur le bouton Submit, les informations que contient le formulaire

sont passées au programme dont le nom est cité dans l'attribut `action` de la balise `form`. La forme générale se présente ainsi :

```
<form action="traitement.php" method="POST">
    balises pour un ou plusieurs champs
    <input type="submit" value="chaîne">
</form>
```

Comme nous l'avons vu au Chapitre 8, c'est ainsi qu'on recueille des informations en provenance de l'utilisateur. Cependant, on peut profiter d'un formulaire pour convoyer d'autres informations en exploitant les propriétés des *champs cachés* (`type="hidden"`). A la limite, un formulaire pourrait ne contenir que ce type de champ. Mais, pour que ces valeurs soient transmises, le formulaire devra néanmoins posséder une balise de `type="submit"` et il faudra que l'utilisateur clique sur le bouton correspondant.

A titre d'exemple, voici un formulaire ne demandant aucune réponse à l'utilisateur, et qui passera la couleur du fond sous le nom `$couleur` à la page suivante lorsque l'utilisateur cliquera sur le bouton marqué "Page suivante" :

```
$couleur = "bleu"; // couleur de fond
...
echo "<form action='pagesuivante.php' method='post'>
    <input type='hidden' name='couleur' value='$couleur'>
    <input type='submit' value='Page suivante'>
</form>\n";
```

Lorsque l'on clique sur le bouton, le programme `pagesuivante.php` est exécuté et l'élément de tableau `$_POST['couleur']` contient la valeur "bleu".

Le mécanisme de session PHP

Une *session* représente la durée de la visite d'un utilisateur dans un site Web, quelle que soit par ailleurs cette durée. Ce mécanisme est apparu à partir de la version 4.0 de l'interpréteur PHP. Il permet notamment d'associer à la session certaines informations qui resteront accessibles de page en page.

Détails de fonctionnement

Initialement, vous devez appeler une fonction particulière pour définir une session et spécifier les variables qui doivent être

partageables au cours de cette session. Dans les pages suivantes, lorsque vous invoquerez la session en cours, ces variables seront immédiatement accessibles dans le tableau natif `$_SESSION`. Voici le détail de ce mécanisme :

1. **PHP assigne un identificateur (ID) de session à la session qui démarre. Il s'agit d'un nombre unique n'ayant pas de signification particulière. Il est conservé dans une variable native de PHP appelée `PHPSESSID`.**
2. **Les variables de session sont placées dans un fichier sur le serveur Web. Ce fichier a pour nom l'identificateur de session. Avec Linux/UNIX, il est placé dans le répertoire `/tmp`. Sous Windows, il est placé dans le sous-répertoire `sessiondata` du dossier dans lequel PHP est installé.**



L'administrateur de PHP peut modifier l'emplacement de stockage des fichiers de session en éditant la rubrique `session.save_path` de `php.ini`. L'identificateur de session (ID) est passé à chacune des pages qui seront ensuite appelées. Si l'utilisateur accepte les cookies, cet identificateur sera passé par un cookie. Si ce n'est pas le cas, il sera ajouté au bout des URL ou placé dans une variable cachée de formulaire dont l'attribut `method` vaut `post`.

4. **Les variables de session sont automatiquement restaurées au début de chaque page de la session courante. Ces variables sont dès lors utilisables dans la page avec les valeurs qui leur ont été attribuées dans les pages précédentes. PHP les range dans un tableau natif appelé `$_SESSION` dont les clés sont les noms des variables et dont la valeur a été définie dans la page précédente.**



Pour que le mécanisme des sessions fonctionne, il est indispensable que la variable de configuration `track-vars` soit activée. A partir de PHP 4.0.3, c'est l'option par défaut. Pour les versions précédentes, l'option `--enable-track-vars` doit être précisée lors de l'installation de PHP.



Si l'utilisateur a désactivé les cookies, les sessions ne peuvent être exploitées que si `trans-sid` est activé. PHP doit alors avoir été installé avec l'option `--enable-trans-sid`. Nous y reviendrons plus loin dans la section "Utilisation des variables de session".

Ouverture d'une session

Vous devez ouvrir une session dans chaque page Web au moyen de l'appel de fonction suivant :

```
session_start();
```

Cette fonction commence par voir s'il existe un identifiant de session (ID). Si c'est le cas, elle initialise le tableau `$_SESSION`. Sinon, elle commence par créer un nouvel identificateur de session.

Comme les sessions utilisent les cookies lorsque l'utilisateur ne les a pas désactivés, `session_start()` a les mêmes limitations que `setcookie()` : elle doit être appelée avant toute sortie en direction du navigateur. Voyez à ce sujet l'encadré *Instructions devant précéder l'envoi d'informations au navigateur*, plus haut dans ce même chapitre.

Utilisation des variables de session

Vous devez déclarer toutes les variables que vous voulez utiliser au cours d'une même session dans le tableau `$_SESSION`, sous la forme :

```
$_SESSION['nomVariable'] = valeur;
```

La valeur ainsi déclarée devient disponible dans le tableau `$_SESSION` pour toutes les autres pages Web de votre site. Voici par exemple comment mémoriser le département dans lequel habite le visiteur :

```
$_SESSION['département'] = "Oise";
```

Il vous suffit alors de lire le contenu de `$_SESSION['département']` dans une page pour disposer de cette information.

Les deux programmes qui suivent illustrent le mécanisme des sessions. Le premier, `testSession1.php` (Listing 9.2), montre comment se présente la page qui initialise la session. Le second, `testSession2.php` (Listing 9.3), présente la page appelée par `testSession1.php`. Une fois la première page chargée dans votre navigateur, vous allez voir ce que montre la copie d'écran de la Figure 9.3. Modifiez éventuellement le contenu de la boîte de saisie puis cliquez sur le bouton "Page suivante". Vous verrez alors s'afficher la copie d'écran de la Figure 9.4 (qui nous apprend que l'utilisateur a poliment répondu au texte de bienvenue).

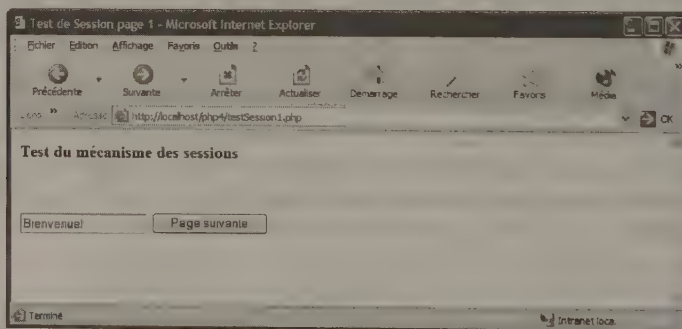
Listing 9.2 : Première page d'une session.

```

<?php
    session_start();
?>
<html>
<head><title>Test de Session page 1</title></head>
<body>
<?php
    $_SESSION['var_session'] = "test";
    echo "<h3>Test du mécanisme des sessions</h3>."
        <form action='testSession2.php' method='POST'>
            <input type='hidden' name='var_form'
                value='Je teste'>
            <input type='text' name='var_util'
                value='Bienvenue!'>
            <input type='submit' value='Page suivante'>
        </form>";
?>
</body></html>

```

Figure 9.3 :
Première
page de la
session.



Vous remarquez que le programme définit trois variables qu'il transmet à la seconde page. Il crée la variable de session `session_var` et affiche un formulaire comportant une variable cachée `form_var` qui est transmise à l'autre page lors de la validation.

Il suffit de charger la page `testSession1.php` dans votre navigateur pour tester l'exemple.

Comme le mécanisme des sessions est différent selon que les cookies sont ou non acceptés par l'utilisateur, vous devriez faire ce test dans les deux cas de figure.

Listing 9.3 : Seconde page d'une session.

```

<?php
    session_start();
?>
<html>
<head><title>Test de Session page 2</title></head>
<body>
<?php
    echo "var_session = {$_SESSION['var_session']}<br>\n";
    echo "var_form = {$_POST['var_form']}<br>\n";
    echo "var_util = {$_POST['var_util']}<br>\n";
?>
</body></html>

```

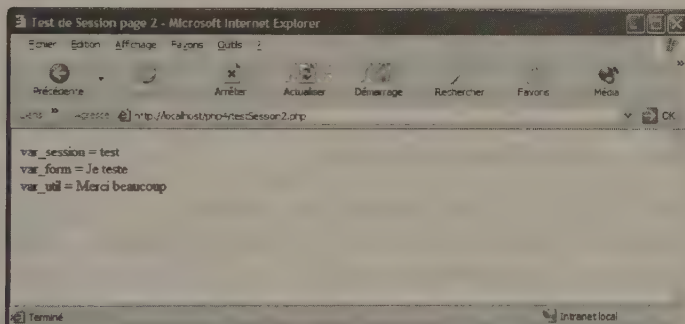


Figure 9.4 :
Seconde
page de la
session.

Voici comment configurer Internet Explorer pour refuser les cookies :

1. Cliquez sur Outils/Options Internet.
2. Dans la boîte de dialogue à plusieurs onglets qui s'ouvre, cliquez sur Confidentialité.
3. Cliquez sur le bouton "Avancé".
4. Cliquez sur Ignorer la gestion automatique.
5. Choisissez Refuser pour les cookies internes et ceux des tierces parties.
6. Cliquez sur OK deux fois de suite.

Voici comment configurer Firefox pour refuser les cookies :

1. Cliquez sur Outils/Options.
2. Accédez à la rubrique Vie privée.
3. Dans la zone Cookies, cliquez sur la case Activer les cookies en sorte qu'elle ne soit pas cochée.
4. Cliquez sur OK.

Si, ensuite, vous voyez s'afficher un diagnostic (cela dépend du niveau de signalisation des erreurs) et constatez que la variable `$var_session` n'affiche pas de valeur, il faut vérifier l'option `trans-sid` de `php.ini`. Cherchez la ligne suivante :

```
session.use_trans_sid = 0
```

Changez le 0 en 1. Si vous ne pouvez pas résoudre le problème, les sessions resteront utilisables, mais vous devrez passer leur identificateur dans des instructions de vos programmes. En effet, PHP ne le fera pas automatiquement si les cookies sont désactivés. Dans ce cas, consultez la section qui suit.



Avec PHP 4.1.2 ou une version plus ancienne, `trans-id` n'est pas disponible, sauf s'il a été activé en insérant l'option `--enable-trans-sid` lors de la compilation de PHP.

Les sessions sans les cookies

PHP teste l'acceptation des cookies par le navigateur de l'utilisateur. S'ils sont acceptés, il agit alors ainsi :

- ✓ Il donne à la variable `$PHPSESSID` la valeur de l'identificateur de session.
- ✓ Il utilise les cookies pour transférer cette variable d'une page à la suivante.

S'ils sont refusés, voici comment il opère :

- ✓ Il définit une constante appelée `SID`. Elle contient une paire `variable=valeur` ressemblant à `PHPSESSID=longue chaîne de caractères`.

- ✓ Selon la valeur de `trans-sid`, l'identificateur de session sera ou non accessible d'une page à la suivante. Si (et seulement si) ce paramètre est actif, l'identificateur de session sera transmis.

Activer ce paramètre a des avantages et des inconvénients. Du côté positif, les sessions peuvent fonctionner même si les cookies sont désactivés. De plus, la programmation des sessions est simplifiée. De l'autre côté, l'identificateur de session sera généralement passé à la suite de l'URL. Dans certaines situations, il est préférable que cet identificateur ne soit pas visible dans la fenêtre d'adresse du navigateur. En outre, il peut alors être enregistré dans ses signets par un utilisateur. Si ce dernier retourne ultérieurement sur le même site avec cet identificateur, cela risque d'occasionner des confusions au niveau de la session, voire de provoquer divers problèmes à cause de l'ancien identifiant de session).

Lorsque trans-sid est actif

S'il découvre que l'utilisateur a bloqué la réception des cookies, l'interpréteur PHP ajoute automatiquement l'identificateur de session à l'URL lors du chargement de toute nouvelle page ou dans un champ caché de formulaire. Si l'utilisateur clique sur un lien pour changer de page, que ce soit à l'aide d'une fonction header ou d'un formulaire dans lequel l'attribut `method` vaut `get`, l'identificateur est ajouté à la fin de l'URL. S'il le fait avec un formulaire dans lequel `method` vaut `post`, l'identificateur est passé dans un champ caché. PHP reconnaît `$PHPSESSID` comme identificateur de session et ne demande aucune programmation particulière dans le script.

L'identificateur de session n'est ajouté à l'URL que dans le cas d'une URL locale, c'est-à-dire concernant le même serveur que celui de la page courante. Si cette URL contient un nom de machine, PHP suppose que la page se trouve sur un autre serveur et n'utilise pas cette méthode. Par exemple :

```
<a href="nouvelleepage.php">
```

entre dans la première catégorie et l'identificateur sera ajouté à l'URL. Mais avec :

```
<a href="http://www.monserveur.com/nouvelleepage.php">
```

PHP n'incorporera pas l'identificateur de session dans l'URL.



Lorsque trans-sid n'est pas actif

S'il découvre que l'utilisateur a bloqué la réception des cookies, l'interpréteur PHP n'ajoute pas l'identificateur de session à l'URL. Il vous appartient alors de procéder vous-même à ce passage d'informations.

Il existe heureusement une constante nommée (voir à ce propos le Chapitre 6) que vous pouvez utiliser pour acheminer l'identificateur de session. Elle s'appelle `SID` et contient un couple `variable=valeur` que vous pouvez ajouter manuellement à l'URL de la façon suivante :

```
<a href="second.php?<?PHP echo SID ?>">Page suivante</a>
```

Cette instruction ajoute un point d'interrogation à l'URL et le fait suivre du couple que représente `SID`. Voici comment se présente l'URL affichée dans la fenêtre d'adresse du navigateur :

```
http://localhost/second.php?PHPSESSID=5accfd985aebf1a8c73e3374727a0122
```

Pour diverses raisons évoquées plus haut, vous pouvez souhaiter que l'identificateur de session ne soit pas affiché dans la fenêtre du navigateur. Il faut alors le passer dans un champ caché de formulaire en utilisant la méthode `post` de la façon suivante :

```
<?
$PHPSESSID = session_id();
echo "<form action='pagesuivante.php' method='post'>
    <input type='hidden' name='PHPSESSID' value='$PHPSESSID'>
    <input type='submit' value='Page suivante'>
  </form>";
?>
```

- ✓ On commence par appeler la fonction `session_id()` qui renvoie l'identificateur de la session courante et on place cette valeur dans la variable `$PHPSESSID`.
- ✓ Il suffit ensuite de donner cette valeur au champ caché du formulaire `$PHPSESSID`.

Dans la page suivante, PHP utilisera automatiquement `$PHPSESSID` pour récupérer les variables de session sans programmation particulière.

Sessions privées

Le mécanisme des sessions est idéal pour les pages Web à accès restreint qui demandent à l'utilisateur voulant les visiter de s'identifier au moyen d'un login et d'un mot de passe. En exploitant le mécanisme des sessions, voici comment implémenter ce type de pages :

1. **Afficher une page de login.**
2. **Si le visiteur a fourni ses deux identificateurs et que ceux-ci ont été reconnus exacts, démarrer une session dans laquelle on crée une variable indiquant qu'il est reconnu.**
3. **Chaque fois que le visiteur change de page, tester cet indicateur.**
4. **S'il existe, afficher la page.**
5. **S'il n'existe pas, revenir à l'étape 1.**

Le test de l'étape 3 peut s'écrire ainsi :

```
<?php
    session_start()
    if (@$_SESSION['login'] != "oui")
    { header("Location: premièrePage.php"); // page de Login
      exit();
    }
?>
```

Ici, `$_SESSION['login']` est une variable de session qui est définie avec la valeur "oui" lorsque l'utilisateur a été reconnu. Si le test échoue, la page de connexion est réaffichée. Sinon, la suite des instructions de la page Web est exécutée normalement.

Fermeture d'une session PHP

Dans les pages Web à accès restreint, on souhaite souvent que les visiteurs se "délogent" lorsqu'ils ont terminé leur visite. Pour fermer la session, il suffit d'écrire :

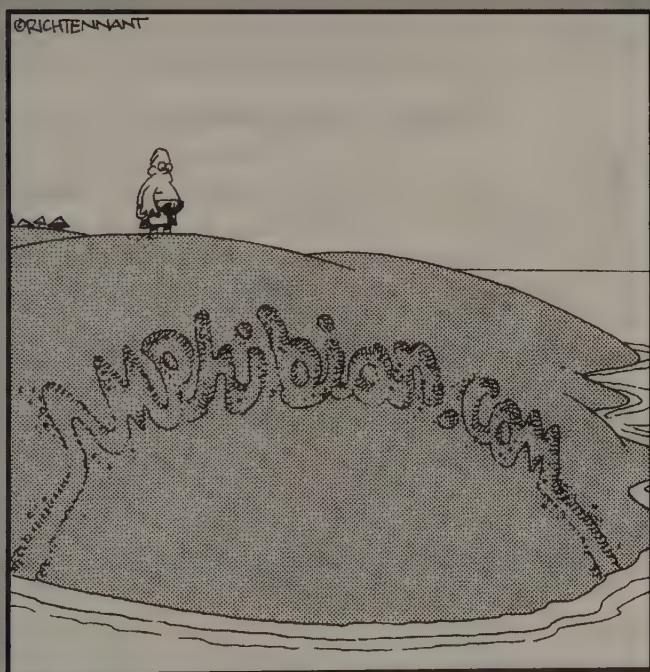
```
session_destroy();
```

Toutes les informations conservées dans le fichier de session sont alors détruites ainsi, naturellement, que leur contenu. L'identificateur de session n'est donc plus disponible pour la page suivante. Mais cela n'affecte pas les variables de la page courante qui conservent leurs valeurs. Si vous voulez les faire disparaître également, vous devez appeler la fonction `unset()` :

```
unset($_SESSION);
```

Quatrième partie

Introduction au développement d'application



Dans cette partie...

Cette partie recèle les informations nécessaires pour la mise en place des applications de ce livre. Vous y trouverez des détails concernant les applications, vous verrez où les trouver, où les placer, comment les assimiler et comment les modifier.

Dans la construction d'applications Web, il est important de garder à l'esprit la notion de sécurité. Ces chapitres présentent les problèmes de sécurité et l'écriture de code sécurisé.

Chapitre 10

Construction d'une application

Dans ce chapitre :

- Installer les applications téléchargées.
- Configurer votre environnement de programmation.
- Personnaliser les applications du livre.
- Planifier le développement de l'application.

Exploitation du code source fourni

Le code des applications de ce livre est téléchargeable sur le site de *First Interactive* (www.efirst.com/EFIindex.asp). Chaque application se trouve dans son propre répertoire. Après copie de tous les fichiers d'un de ces répertoires vers votre espace Web, l'application est exécutable dans votre navigateur.

Choix d'un emplacement

Décompressez l'archive téléchargée, et copiez les fichiers sur votre espace Web. Placez tous les fichiers dans un sous-répertoire de votre espace Web, comme `c:\program files\apache group\apache\http\catalogue`. Les fichiers sont de trois types :

- ✓ **Scripts PHP** : Les fichiers contiennent les scripts en code PHP, qui crée la fonctionnalité de l'application. Les fichiers de script PHP portent l'extension `.php`.
- ✓ **Fichiers inclus** : Ces fichiers sont appelés à l'aide d'instructions `include` écrites dans les scripts PHP. Les fichiers inclus se terminent par l'extension `.inc`.

- ✓ **Classes** : Ces fichiers contiennent des définitions de classe pour les programmes orientés objet. Les fichiers sont appelés au début des scripts PHP à l'aide d'instructions d'insertion `include`. Les fichiers de classes se terminent par l'extension `.class`.

L'application est exécutable sous réserve que tous les fichiers soient réunis dans un même répertoire. Vous voudrez éventuellement organiser les fichiers dans des sous-répertoires. Il faut alors modifier le script, pour employer le chemin d'accès correct pour l'inclusion ou l'appel de fichiers.

Un des fichiers inclus, nommé `Vars.inc`, contient des informations sensibles, nécessaires à l'accès à la base de données MySQL. Sécurisez ce fichier en le plaçant dans votre *répertoire include*, un répertoire où PHP ira chercher les fichiers réclamés par les instructions `include`. Le répertoire `include` peut être situé à l'extérieur de votre espace Web, là où les visiteurs de votre site ne peuvent accéder. On déclare ce répertoire `include` dans le fichier `php.ini`. Cherchez le paramètre `include_path`. Si la ligne commence par un point-virgule (;), supprimez ce point-virgule. Écrivez le chemin d'accès au répertoire à employer comme répertoire `include`. Par exemple :

```
include_path=".;c:\include";           #Windows
include_path=".;/include";             #Linux
```

Ces deux instructions déclarent deux répertoires où PHP doit rechercher les fichiers inclus. Le premier répertoire est représenté par un point (signifiant le répertoire courant), suivi du chemin d'accès au second répertoire. Spécifiez autant de répertoires que vous voulez, PHP y recherchera les fichiers inclus, dans l'ordre où ils sont énumérés. Les chemins de répertoires sont séparés par un point-virgule pour Windows et un deux-points pour Linux.

Lorsque vous n'avez pas accès à `php.ini`, réglez le chemin d'accès dans chaque script avec l'instruction suivante :

```
ini_set("include_path", "c:\hidden");
```



Cette instruction fait pointer `include_path` sur le répertoire spécifié uniquement le temps de l'exécution du programme. Elle ne définit pas le répertoire pour la totalité de votre site Web.

L'application Catalogue de ce livre inclut des images mais elles ne sont pas incluses dans l'archive téléchargeable. Votre catalogue

réclamera ses propres photos de produits. L'application s'attend à trouver les fichiers d'image dans un sous-répertoire nommé Images.

Comprendre le code PHP

Le code PHP des applications ne se compose que d'instructions PHP élémentaires. Il n'exploite pas des concepts ou des instructions PHP évolués. Quiconque a une compréhension élémentaire de PHP peut assimiler le code des applications du livre. Inutile d'être un expert.

Dans chaque application, la majorité du code est incluse dans le (ou les) script PHP principal. Une bonne programmation PHP fait appel à l'écriture de fonctions. Lorsque vous réalisez que vous utilisez le même code plusieurs fois, placez ce code dans une fonction et appelez la fonction aux endroits appropriés du script.

Dans les applications de ce livre, je n'utilise pas les fonctions aussi souvent que j'aurais pu (ou dû). J'ai pensé que vous comprendriez plus facilement le code et son flux en gardant le code dans un seul fichier, plutôt que de devoir sauter des pages pour rechercher les listings des fonctions et revenir ensuite en arrière. Par conséquent, le code est présenté d'une manière plus jointe, il est réparti sur moins de fichiers. Dans les explications du code, vous verrez que j'ai pointé les emplacements où des fonctions seraient les bienvenues.

À la suite de chaque listing, j'explique le code. Les numéros dans les explications renvoient aux numéros de lignes visibles dans le code concerné. Il est ici considéré que vous connaissez le fonctionnement des structures de contrôles dans PHP et savez suivre le flux d'un programme. Vous trouverez une description générale, et des détails pour les blocs de code plus complexes.

Programmation procédurale et programmation orientée objet

Chaque application du livre est écrite en code procédural et en code orienté objet. Cela signifie que l'archive téléchargeable contient deux ensembles de fichiers pour chaque application.

L'idée de fournir les deux types de code suit l'intention de produire un livre utile pour les lecteurs suivants :

- ✓ **Les programmeurs PHP inexpérimentés qui n'ont écrit que du code procédural et doivent construire une application pour un site Web** installeront la version procédurale de l'application.
- ✓ **Les programmeurs expérimentés en code procédural PHP et désirant écrire du code orienté objet** compareront les deux versions pour apprendre à construire du code orienté objet.
- ✓ **Les programmeurs expérimentés en code orienté objet dans un autre langage et qui désirent construire une application orientée objet en PHP** installeront la version orientée objet de l'application.

Les programmations procédurale et orientée objet ne présentent pas simplement une syntaxe différente. La philosophie orientée objet constitue une approche différente de la programmation. Le projet à programmer est modélisé à l'aide d'objets qui représentent les composants du projet de programmation. Les objets stockent des informations et sont capables de réaliser des tâches. Le code qui définit l'objet est stocké dans une classe, utilisable à tout endroit où elle sera utile dans l'application. Le programmeur qui exploite la classe n'a aucun besoin de savoir ce qui se déroule dans la classe où elle réalise ses tâches. Il se contente de l'utiliser. Ainsi, un programmeur peut développer une classe qui sera exploitée par d'autres programmeurs dans leurs programmes.

Le développement d'applications de très grande taille, complexes, implique plusieurs programmeurs ou équipes de programmeurs, et s'avère plutôt difficile sans programmation orientée objet. Avec la programmation orientée objet, les programmeurs sont à même de développer indépendamment des éléments de l'application. Par ailleurs, si quelque chose devait être modifié plus tard, seule la classe concernée serait affectée. Les autres éléments de l'application n'ont pas besoin de changer. Pour les mêmes raisons, la maintenance de l'application est beaucoup plus facile.

Modification du code source

Dans la plupart des cas, vous devrez modifier le code des applications. D'une part, l'aspect des pages Web est plutôt brut. Rien n'excitera vos visiteurs ni ne vous rapportera le prix du Concepteur de l'année. Il faudra donc personnaliser l'aspect des pages. Si vous

ajoutez une de ces applications à un site Web existant, vous la modifierez pour arriver à l'aspect des pages existantes.

Le code source de ce livre est fourni sous forme de fichiers texte simples, vous avez donc la possibilité d'employer tout éditeur de texte de votre choix pour modifier les fichiers de code source PHP. Vous ne serez pas le premier à écrire des scripts avec vi, le Bloc-notes ou WordPad. Il existe cependant des outils qui rendent bien plus facile l'édition de scripts.



Prenez le temps de découvrir les éditeurs de code et les environnements de développement intégré (IDE). Ces outils offrent des fonctions susceptibles de faire gagner énormément de temps dans la construction d'une application. Téléchargez quelques démos, essayez-les et sélectionnez celle qui vous convient le mieux.

Éditeurs de code

Les éditeurs de code offrent de nombreuses fonctions spéciales pour l'écriture de programmes. La plupart des éditeurs de code offrent les fonctions suivantes :

- ✓ **Coloration syntaxique** : Colore en différentes couleurs certaines portions du script, comme les balises HTML, les chaînes de texte, les mots clés, les commentaires, pour qu'elles soient plus faciles à identifier.
- ✓ **Retrait** : Gestion automatique des retraits à l'intérieur des parenthèses et accolades, pour aérer les scripts et en faciliter la lecture.
- ✓ **Numéros de lignes** : Ajout de numéros de lignes temporaires. Cette fonction est importante car les messages d'erreur PHP précisent la ligne où l'erreur se situe. Il serait vraiment dommage de compter 872 lignes à partir du haut du fichier pour trouver la ligne signalée par PHP.
- ✓ **Fichiers multiples** : Vous pouvez garder ouverts plusieurs fichiers à la fois.
- ✓ **Aide à l'insertion de code** : Vous disposerez de boutons pour l'insertion de code, telles que balises HTML ou instructions et fonctions PHP.

- ✓ **Bibliothèque de code** : Il est possible d'enregistrer des portions de votre propre code, que vous insérerez ensuite à volonté en cliquant sur un bouton.

De nombreux éditeurs de code sont disponibles sur Internet, gratuitement ou à petit prix. Voici quelques-uns des éditeurs les plus populaires :

- ✓ **Arachnophilia** : Editeur multi-plateforme écrit en Java. Il est gratuit, de type CareWare (logiciel de bienfaisance).
www.arachnoid.com/arachnophilia
- ✓ **BBEdit** : Editeur conçu pour le Mac. Il coûte 199 \$. La version BBEdition Lite est gratuite. Le développement et l'assistance ne sont plus assurés mais le logiciel reste exploitable légalement.
www.barebones.com/products/bbedit/index.shtml
- ✓ **EditPlus** : Editeur conçu pour Windows. C'est un shareware, dont la licence coûte 30 \$.
www.editplus.com
- ✓ **Emacs** : Emacs fonctionne sous Windows, Linux et UNIX. Il est gratuit.
www.gnu.org/software/emacs/emacs.html
- ✓ **HomeSite** : Conçu pour Windows, il coûte 99 \$.
www.macromedia.com/software/homesite
- ✓ **HTML-Kit** : Autre éditeur Windows utilisable gratuitement.
www.chami.com/html-kit
- ✓ **TextWrangler** : Editeur conçu pour Mac. Il est développé et publié par la même société qui commercialise BBEdition. TextWrangler offre moins de fonctions que BBEdition mais présente la plupart des fonctions majeures utiles aux programmeurs, comme la coloration syntaxique et le retrait automatique. Il est *gratuit*.
www.barebones.com/products/textwrangler/index.shtml
- ✓ **Vim** : Versions étendues gratuites de vi, utilisable sous Windows, Linux, UNIX et Mac OS.

Environnement de développement intégré (IDE)

Un *environnement de développement intégré (IDE)* est un espace de travail totalement dédié au développement d'applications. Il inclut un éditeur de code et d'autres caractéristiques. Voici quelques fonctions présentées par la plupart des IDE :

- ✓ **Débogage** : Fonctions de débogage intégrées.
- ✓ **Aperçu** : Affiche la page Web produite par le script.
- ✓ **Test** : Fonctions de test des scripts.
- ✓ **FTP** : Capacité de se connecter et de télécharger (dans les deux sens) par FTP. Cette fonction garde également trace des fichiers appartenant au site Web et conserve le site à jour.
- ✓ **Gestion de projet** : Organise les scripts en projets, gère les fichiers du projet, et inclut des fonctions d'extraction et d'archivage de fichiers.
- ✓ **Sauvegardes** : Réalise à intervalles réguliers des sauvegardes automatiques de votre site Web.

Les IDE demandent plus de temps de familiarisation que les éditeurs de code. Certains sont plutôt chers mais leur profusion de fonctions peut valoir le coût. Les IDE sont particulièrement utiles lorsque plusieurs personnes écrivent des scripts pour une même application. L'IDE facilite la coordination du projet et rend le code plus compatible.

Voici quelques IDE populaires :

- ✓ **Dreamweaver MX** : Disponible pour les plates-formes Windows et Mac. Il fournit des outils de mise en page visuelle, permettant de créer une page Web en cliquant sur des boutons pour insérer des éléments, en cliquant-glissant les éléments pour les disposer. Dreamweaver écrit le code HTML à votre place. Il intègre aussi cependant l'éditeur HomeSite, si vous désirez rédiger vous-même le code. Il gère également PHP. Dreamweaver coûte environ 500 euros.

www.macromedia.com/dreamweaver

- ✓ **Komodo** : Proposé pour les plates-formes Linux et Windows. C'est un IDE pour les langages open source, dont Perl et Python, ainsi que PHP. Il est commercialisé à 29,95 \$ pour un usage personnel ou éducatif, et 295 \$ pour un usage commercial.

www.activestate.com/Products/Komodo

- ✓ **Maguma** : Disponible uniquement pour Windows. C'est un IDE pour Apache, PHP et MySQL sous Windows. Il existe en deux versions, à des prix différents : Maguma Studio Desktop et Maguma Studio Enterprise, qui offre des fonctions pour les énormes sites avec plusieurs serveurs. Maguma Studio for PHP est une version gratuite ne gérant que le PHP.

www.maguma.com

- ✓ **PHPEdit** : IDE gratuit, disponible uniquement pour Windows.

www.waterproof.fr/products/

- ✓ **Zend Studio** : Est proposé pour les plates-formes Linux et Windows. Cet IDE est produit par les personnes qui ont développé le moteur Zend, qui est le moteur sous-jacent de PHP. Ces gens-là connaissent donc très bien PHP. Zend Studio coûte 195 \$.

www.zend.com/store/products/zend-studio.php

Planifier le développement de l'application

La planification est une part essentielle de la construction d'une application. Le plan de l'application servira d'épreuve de contrôle pour sa construction. Le plan (de même que le planning) doit être suffisamment abouti pour garder votre projet sur les rails, vers son objectif, et pour garantir que tous les éléments et fonctions nécessaires sont inclus.

Même si vous utilisez une des applications de ce livre, il faut commencer par construire votre propre plan. Votre plan vous servira de guide, vous verrez si l'application satisfait telle quelle vos besoins ou s'il faut la modifier, lui ajouter ou supprimer des fonctions pour qu'elle y réponde parfaitement.



Plus votre application est volumineuse et complexe, plus vous devrez être organisé. Une application qui affiche Hello World, comptant cinq lignes de script, écrit par une personne, nécessite peu de planification. Le site Web d'Amazon requiert une incroyable organisation.

Planification du logiciel

L'organisation de l'aspect logiciel de l'application comporte les étapes suivantes :

1. Identifiez l'objectif ou les objectifs de l'application.

L'application doit-elle recueillir des informations de l'utilisateur ? Vendre des produits ? Divertir l'utilisateur ? Créer une communauté d'internautes ?

2. Rédigez une liste des tâches que l'application doit réaliser pour atteindre l'objectif.

Par exemple, si l'objectif est de vendre des produits, l'application doit au moins afficher les informations sur les produits, fournir au client un moyen de sélectionner un produit, de recueillir les informations nécessaires pour remplir la commande, et facturer le client.

3. Planification de la base de données.

Décidez quelles informations sont à stocker. Décidez comment les stocker pour obtenir un accès rapide et facile.

4. Rédigez un plan détaillé des méthodes à employer pour réaliser les tâches généralistes élaborées à l'étape 2.

Par exemple, "recueillir l'information nécessaire pour remplir la commande" peut être développé :

- a. Afficher un formulaire.
- b. Vérifier l'information fournie dans le formulaire.
- c. Stocker l'information dans une base de données.

5. Planification des pages Web.

Combien de pages Web doivent être affichées ? Par exemple, faut-il un formulaire et une page de commentaires ? Une page d'information produit ? Une page qui a l'air d'un échiquier ? Pensez l'aspect et le style des pages Web.

Organisation supplémentaire

Le plan de l'application est une base pour le reste de la planification du projet. Vous devez établir le calendrier du projet. Il faut également un plan de ressources.

Établissement d'un calendrier

La date la plus importante du projet est la date de lancement de l'application. C'est-à-dire la date à laquelle les internautes pourront se servir de l'application. Dans certains cas, on vous impose la date et vous devez déterminer les ressources à mettre en place pour satisfaire le délai. Dans d'autres cas, vous disposez de telles ressources et vous devez estimer la date à laquelle l'application sera exploitable.

Vous pouvez utiliser le plan de l'application pour estimer le nombre d'heures-personnes nécessaires pour construire l'application. L'ensemble dépend aussi du nombre de programmeurs travaillant sur l'application. Un projet de 100 heures occupera un programmeur deux semaines et demie, si le programmeur est optimal 40 heures par semaine. Deux programmeurs mettront théoriquement une semaine et un quart pour terminer l'application.

Pour le calendrier, prévoyez quelque marge pour pallier les problèmes dus à la loi de Murphy : réécrire du code effacé, récupérer le temps perdu en raison de la peste bubonique et des coupures de courant provoquées par la foudre, etc. Pensez aussi au temps de test, ainsi qu'à l'écriture de la documentation.

N'oubliez pas que vous pourrez mener certaines tâches simultanément mais que d'autres seront forcément séquentielles. Par exemple, si une de vos tâches est d'acheter un ordinateur, la programmation ne pourra démarrer avant que l'ordinateur soit là.



Un logiciel de gestion de projet peut être utile pour l'établissement du calendrier. Il garde trace des tâches, des ressources, et des événements importants en cours de route. À partir des tâches, il

produit un graphique chronologique, montrant le *chemin critique* – la série de tâches devant être achevée à tel moment pour que le projet soit mené à bien dans les temps.

Organisation des ressources

Les ressources comprennent les ressources humaines et matérielles. Le plan de votre application et la date de remise du projet déterminent les ressources humaines nécessaires. Votre plan doit planifier la disponibilité des ressources humaines. Si vous devez engager plus de personnes, intégrez ce temps dans votre planning. Si vous comptez employer du personnel existant, planifiez le temps du projet dans leurs agendas.

Assurez-vous que les ressources matérielles seront disponibles au moment voulu. Par exemple, si un nouvel ordinateur est prévu, démarrez le processus d'achat antérieurement, pour que l'ordinateur soit là avant qu'il ne soit réellement nécessaire. Les applications de ce livre nécessitent PHP et MySQL, il faut donc prévoir leur disponibilité. Les logiciels requis sont-ils installés ? Faut-il une mise à jour ? S'ils ne sont pas installés, qui les installera, et qui les administrera ? Quand l'administrateur les rendra-t-il disponibles ?

Insérez la liste des ressources nécessaires, tant humaines que matérielles, dans le plan de votre projet. Pour les projets comme les applications de ce livre, les ressources requises sont du personnel et des ordinateurs. Cependant, pour vos propres projets, de nombreuses autres ressources seront éventuellement nécessaires. Par exemple, des dessins ou photos de produits seront indispensables. Pour un catalogue en ligne, du texte peut aussi être nécessaire. Sans oublier un livre de référence ou deux. La liste des ressources évitera les temps morts passés à attendre la disponibilité des ressources nécessaires.

Chapitre 11

Intégration de la sécurité dans une application

Dans ce chapitre :

Identifier les risques de sécurité.

Vérifier et filtrer les données de sources externes.

Empêcher les attaques par injection de commandes SQL.

Sauvegarde de votre base de données.

La sécurité est une notion importante en informatique en général, et pour les sites Web en particulier. La protection n'est *pas* une fonction en tout ou rien ; il s'agit de toute une étendue, allant de pas de sécurité à une sécurité totale. *Pas de sécurité* est par exemple un ordinateur au milieu d'un centre commercial, avec accès libre. *La sécurité totale* sera alors un ordinateur enfermé dans un coffre, avec impossibilité à quiconque de s'en servir.

La sécurité de votre site Web est quelque part entre les deux. Votre site Web doit être accessible via Internet ; dans le cas contraire, personne ne verra vos pages Web. L'objectif est de limiter la possibilité d'action des visiteurs de votre site, tout en les laissant télécharger les pages et, pour les applications de ce livre, saisir des informations dans un formulaire. Par contre, il n'est pas souhaité que les visiteurs puissent reformater votre disque dur ou supprimer les fichiers de votre site Web.

La sécurité de site Web est un compromis entre mesures de sécurité et facilité d'emploi. Par exemple, si vous réclamez que les internautes commencent par s'identifier sur votre site (login), le site leur devient plus difficile à utiliser. Ils doivent saisir leur ID utilisateur et leur mot de passe, ce qui veut dire qu'ils doivent les mémoriser (ou se souvenir où ils ont noté ces informations). Certains sites

Web réclament malgré cela une identification, pour des raisons de sécurité, même si le site devient plus difficile à employer. Assurez-vous simplement que cette opération de login est réellement nécessaire. Certains visiteurs n'utiliseront pas le site s'il faut s'identifier.

Plus vous ajoutez de sécurité, plus le site devient d'emploi difficile, il ne faut donc pas en faire plus que nécessaire. Évaluez l'importance des données à protéger. Si vous stockez des informations gouvernementales top secret, ou un plan d'accès à une mine de numéros de carte bancaire, il faut un haut niveau de sécurité. Cependant, si vous stockez des recettes, la sécurité est forcément moindre.

PHP et MySQL offrent chacun des fonctions et des problèmes de sécurité. Ces aspects sont étudiés en détail dans ce chapitre. En outre, d'autres aspects de sécurité concernent l'ordinateur hébergeant votre site, et la connexion Internet à ce site. Les problèmes de sécurité informatique et Internet sont le domaine de l'administrateur système, pas du programmeur. Ce livre parle de construction d'applications, pas de sécurité système.

Comprendre les risques de sécurité

Sécurité est un autre terme pour protection. De même que vous protégez votre domicile des intrusions, vous devez en protéger votre ordinateur. Bien que la majorité des visiteurs de votre site Web se compose de clients sans autre intention que de bénéficier des services que vous proposez, tout le monde n'est pas honnête et bien intentionné. Certains ont de mauvais objectifs, comme :

- ✓ **Dérober quelque chose** : Le pirate espère trouver un fichier de numéros de carte bancaire valides, si ce n'est le secret de l'éternelle jeunesse.
- ✓ **Effacer votre site Web** : Le but est ici de détruire votre site. Ou d'y ajouter des graffitis. Certaines personnes trouvent cela drôle. Certains le font pour démontrer qu'ils en sont capables. À moins que vous ayez fait du tort à quelqu'un.
- ✓ **Attaquer vos visiteurs** : La manœuvre consiste à ajouter des éléments sur votre site Web, pour occasionner des dégâts ou voler quelque chose à vos visiteurs.

Dans une conception de site Web, on ne peut donc faire l'impasse sur la sécurité. Il faut penser autant à la sécurité qu'à la fonctionnalité.

Vous devez envisager les possibles mauvais usages de votre site et réfléchir à la protection contre ces méfaits.

Ajout de sécurité aux scripts PHP

PHP sert à construire des sites Web dynamiques. Les sites Web sont, par définition, accessibles via Internet, ce qui les rend sujets à une possible infiltration et au vol. De plus, l'aspect dynamique de PHP permet aux utilisateurs d'ajouter des informations – généralement malveillantes – à votre site Web. Cependant, un programmeur informé peut réduire les risques de sécurité sur son site.

Ne faire confiance à aucune information de source externe



Ne stockez ou n'utilisez pas d'informations d'une source externe sans vérifier qu'elle contient des éléments attendus. C'est le commandement numéro un. Si vous gardez ce commandement en tête, vos applications ne seront pas vulnérables aux problèmes courants de sécurité rencontrés dans de nombreuses applications.

Identifier les sources externes

Dans un script PHP, l'information est stockée et utilisée dans des variables. Le script affecte des valeurs à des variables, à l'aide d'instructions de cette forme :

```
$nom_variable = valeur;
```

La valeur est d'un de ces types :

- ✓ **Valeur littérale** : Nombre ou chaîne, comme dans cette instruction :

```
$num = 1;
$str1 = "Hello";
```

L'information s'origine dans le script, pas à l'extérieur. Ce type de valeur est sûr, il est utilisable tel quel, sans vérification.

- ✓ **Variable** : L'information d'une variable est stockée dans une autre variable, comme dans l'instruction suivante :

```
$nom_de_variable2 = $nom_de_variable1 ;
```

Cette instruction est sécurisée si `$nom_de_variable1` s'origine dans le script. Cependant, si `$nom_de_variable1` contient des informations d'une source externe, elle doit être traitée comme information suspecte. Vérifiez-la avant de la stocker ou de l'exploiter.

Voici quelques sources de données externes :

- ✓ **URL** : PHP lit l'information à la fin de l'URL lorsqu'un fichier est téléchargé. L'information se compose de paires nom/valeur de variable.
- ✓ **Données POST** : PHP lit les données soumises via la méthode POST.
- ✓ **Cookies** : PHP lit les données à partir de cookies. *Les cookies* sont des informations stockées sur l'ordinateur de l'utilisateur, et envoyées au serveur lorsque l'utilisateur accède au site.

Les données reçues de sources externes peuvent contenir n'importe quoi, y compris des informations susceptibles d'endommager votre site Web. Toute information de sources externes doit être vérifiée et filtrée. Le reste de cette section étudie quelques moyens de vérification et filtrage des données.

Spécification de la source des données

Lorsque vous utilisez des données d'une source externe au script, assurez-vous qu'elles proviennent de la source attendue. Par exemple, si vous transmettez des données dans une variable masquée d'un formulaire, veillez à recevoir l'information via un tableau `$_POST`. Par exemple, l'application identifie un client, puis passe la variable d'autorisation, comme `login=yes`, dans une variable masquée du formulaire. Lorsque vous vérifiez si un utilisateur est autorisé à afficher telle page Web, vous devez employer un code comme celui-ci :

```
if(!$_POST['login'] == "yes")  
{
```



```
    echo "Vous n'êtes pas identifié";
}
```

Ainsi, lorsqu'un utilisateur essaye d'accéder à votre site avec l'URL suivante :

```
http://www.votresite.com?login=yes
```

l'utilisateur ne pourra afficher la page car la variable d'autorisation est dans `$_GET['login']`, pas dans `$_POST['login']`.

L'obtention de variables de formulaire d'un tableau `$_POST` est la méthode la plus sûre. Si vous vérifiez la variable d'autorisation dans `$_REQUEST['login']`, l'utilisateur apparaît identifié car les éléments des deux tableaux `$_POST` et `$_GET` sont inclus dans `$_REQUEST`.

Une autre méthode consiste à valider le paramètre `register_globals` dans `php.ini`. Une variable `$login` sera alors disponible. Vous pourrez procéder à la vérification suivante :

```
if($login != "yes")
{
    echo "Vous n'êtes pas identifié";
}
```

Cependant, ce code ne contrôle pas d'où vient l'information. Si l'utilisateur a accédé au site avec la variable de login dans l'URL, il apparaîtra comme identifié.

Le code est plus sécurisé lorsqu'il vérifie la source des données.

Laissez `register_globals` désactivé, ce qui est la configuration par défaut, et obtenez l'information via le tableau superglobal approprié. Cela ne suffit pas pour obtenir un code sécurisé. Cela consolide néanmoins l'application.

Vérifier le type des données externes

Votre script PHP doit vérifier toutes les données reçues de source externe, pour être sûr qu'elles contiennent les informations attendues.

À ce propos, vous pouvez vérifier le type de donnée contenue dans une variable. PHP fournit des fonctions pour cela. Par exemple, si la donnée doit être un entier, vous le vérifiez ainsi :



```
if(!is_int($_POST['age']))
{
    echo "La donnée n'est pas un entier";
}
```

PHP fournit plusieurs fonctions pour contrôler le type de données : `is_array`, `is_bool`, `is_double`, `is_float`, `is_numeric`, `is_scalar`, `is_string`, etc. Ces fonctions vous serviront à vérifier les données de sources externes.

Nettoyage des informations externes

De nombreuses données externes sont dans des chaînes. Les chaînes contiennent tous caractères, y compris susceptibles de provoquer des problèmes dans votre application, votre base de données, ou aux visiteurs de votre site. Par exemple, les balises HTML sont des sources potentielles de problèmes. Un utilisateur peut entrer des balises de script comme `<script>`. Le script peut s'exécuter et réaliser des actions, comme la suppression de tous les fichiers ou d'une base de données.

PHP fournit deux fonctions pour nettoyer les données, les rendant inoffensives :

- ✓ `strip_tags` : cette fonction supprime tout texte entre `<` et `>` dans les données. Elle recherche un `<` ouvrant et supprime le contenu jusqu'à rencontrer le `>` de fermeture ou jusqu'à atteindre la fin de la chaîne. Vous pouvez préciser les balises à conserver. Par exemple, l'instruction suivante supprime toute balise de la chaîne, sauf `<b>` et `<i>` :

```
$last_name = strip_tags($last_name, "<b><i>");
```

- ✓ `htmlspecialchars` : cette fonction convertit en format HTML certains caractères spéciaux ayant une signification en HTML. Voici les conversions opérées

`<` devient `<` ;

`>` devient `>` ;

`&` devient `&` ;

De cette manière, les caractères `<` et `>` peuvent être affichés dans une page Web sans être interprétés comme des balises

HTML. L'instruction suivante change ces caractères spéciaux dans une variable :

```
$last_name = htmlspecialchars($last_name);
```



Si vous ne voulez pas permettre à l'utilisateur de taper des < ou > dans un champ de formulaire, utilisez `strip_tags`. En revanche, si vous autorisez < ou >, vous pouvez les stocker en toute sécurité après les avoir traités avec `htmlspecialchars`.

Vérification des données externes à l'aide d'expressions régulières

Les expressions régulières sont exploitables pour vérifier que des données sont dans un format raisonnable. Si les données n'ont pas de sens, ce n'est pas quelque chose à stocker dans votre base de données. Par exemple, lorsque l'utilisateur tape un nom dans un formulaire, vous pouvez vérifier qu'il ressemble à un vrai nom en le comparant à des motifs. Vous savez qu'un nom se compose principalement de lettres et d'espaces. Les autres caractères valides sont le tiret (-), comme dans *Smith-Jones*, et l'apostrophe ('), comme dans *O'Hara*. Pour vérifier le nom, vous configurez un motif représentant une chaîne ne contenant que des lettres, espaces, tirets et apostrophes. Puis, vous confrontez le nom au motif. Si le nom ne correspond pas – s'il contient des caractères absents du motif, tels que chiffres ou points d'interrogation (?) – ce n'est pas un nom réel.

Les expressions régulières consistent en caractères littéraux et caractères spéciaux. *Les caractères littéraux* sont des caractères normaux, sans autre signification spéciale. Un *c* est un *c*, sans autre signification que d'être une des 26 lettres de l'alphabet. *Les caractères spéciaux* présentent une signification spéciale dans le motif, comme l'astérisque (*) lorsqu'il est utilisé comme caractère générique. Le Tableau 11.1 présente les caractères spéciaux exploitables dans les expressions régulières.

Les caractères littéraux et spéciaux sont combinés pour créer des motifs, qui sont parfois longs et compliqués. Une chaîne est comparée au motif, et si elle concorde, la comparaison est validée.

PHP fournit des fonctions pour la comparaison de chaînes à des expressions régulières. La fonction `ereg()` s'emploie comme ceci :

```
ereg("expr_reg", $nom_de_variable) ;
```

Tableau 11.1 : Caractères spéciaux dans les motifs d'expressions régulières.

Caractère	Signification	Exemple	Correspondance	Pas de correspondance
^	Début de ligne.	^c	chat	mon chat
\$	Fin de ligne.	c\$	tic	ticket
.	Tout caractère individuel.	..	Toute chaîne contenant au moins deux caractères	a, l
?	Le caractère précédent est facultatif.	lapine?s	lapins, lapines	lapons
()	Groupe des caractères littéraux dans une chaîne qui doit correspondre exactement.	lap(ine)s	lapines	lapins
[]	Englobe un groupe de caractères littéraux facultatifs.	lapin[es]	lapin, lapine, lapines	lapons
@-	Représente tous les caractères entre deux caractères.	lap[i-u]n	lapin, lapon	lapen
+	Un ou plusieurs des éléments précédents.	porte[1-3]+	porte111, porte131	porte55, porte131
*	Aucun ou plusieurs exemplaires des caractères précédents.	porte[1-3].*	porte, porte311	porte4, porte445
{ , }	Nombres minimal et maximal de répétitions possibles.	a{2,5}	aa, aaaaa	a, xx3
\	Le caractère suivant est littéral.	m*n	m*n	mon, man
(/)	Plusieurs groupes possibles.	(Jean Jeannot)	Jean, Jeannot	Juan, Je

Le code suivant teste si l'information est un code postal valide :

```
$expr_reg = "^ [0-9] {5} ( \ - [0-9] {4} ) ? $"
if (!ereg($expr_reg, $_POST['zip']))
{
    echo "Code postal invalide<br>";
}
```

L'expression régulière se compose ainsi :

- ^[0-9] {5} – Toute chaîne commençant par cinq chiffres.
- \ - – Un tiret littéral.

[0-9]{4} – Toute chaîne de quatre chiffres de long.

() ? – Groupe les deux dernier caractères du motif et les rend facultatifs.

Voici un autre code utile :

```
$expr_reg = "^,+@.\,com$"
if(!ereg($expr_reg,$_POST['email']))
{
    echo "Adresse e-mail invalide<br>";
}
```

Ce code n'accepte que les adresses électroniques se terminant par .com. (Les adresses électroniques peuvent se terminer par d'autres caractères.) Voici une autre expression régulière :

```
$expr_reg = "[A-Za-z' -]{1-50}$"
if!(ereg($expr_reg,$_POST['last_name']))
{
    echo "Nom invalide<br>";
}
```

Cette expression régulière accepte uniquement lettres, apostrophes, espaces et tirets. Lorsque tout autre caractère est repéré dans la chaîne, le nom (last_name) est rejeté, considéré invalide.

PHP fournit également des expressions régulières compatibles Perl pour les connaisseurs de Perl. Vous disposez de la fonction `preg_match` pour l'emploi avec des expressions régulières compatibles Perl.

Stockage des données

Dans vos scripts, vous aurez fréquemment besoin de stocker et lire des données. Par exemple, dans une application de commande en ligne, vous devez stocker les informations sur le client, telles que nom et adresse, pour l'expédition de la commande, ainsi que les articles commandés par le client. Le stockage des données se fait à l'aide d'un de ces mécanismes :

- ✓ **Fichier texte** : Vous stockez les informations dans un fichier texte sur le serveur Web. Cette solution est rapide et facile. Cependant, quiconque a accès au serveur Web peut accéder au fichier texte.

- ✓ **Cookies** : Une petite quantité de données est stockable dans un cookie. Cependant, l'information est stockée sur l'ordinateur de l'utilisateur, pas sur le serveur Web, ce qui veut dire qu'elle peut y être modifiée. De plus, les pirates ont des techniques pour dérober les cookies, et donc récupérer les informations qui y sont stockées.
- ✓ **Variables de session** : Les informations de session PHP sont stockées dans un fichier sur le serveur Web. Le fichier est stocké en dehors de l'espace Web, personne ne peut donc accéder au fichier de session via le Web.
- ✓ **Base de données** : Vous stockez les informations dans une base de données. C'est la méthode la plus sécurisée car la base de données présente ses propres fonctions de sécurité en plus des fonctions de sécurité de PHP. Cependant, c'est elle qui demande le plus de travail et génère le plus de charge.



Vous devez protéger l'information que vous stockez. Les mesures de protection nécessaires dépendent de l'importance des données stockées. Lorsque vous stockez un numéro de carte bancaire ou de sécurité sociale, il faut une plus grande protection que s'il s'agit de la couleur préférée du client.

Emploi d'appels système

Certains scripts réclament que vous accédiez au système d'exploitation. Aucune des applications de ce livre n'utilise de commandes du système d'exploitation. Certaines applications le font, par exemple pour gérer des fichiers ou de l'espace disque. Vous pouvez réaliser des accès à votre espace disque d'une de ces manières :

- ✓ **Fonctions PHP** : PHP fournit de nombreuses fonctions à utiliser avec les fichiers et répertoires, comme `copy`, `rename`, `delete`, `mkdir`, `readfile`, et de nombreuses autres.
- ✓ **Exécution de commandes système** : PHP donne la possibilité d'exécuter des commandes système. Vous entrez des commandes système comme si vous travailliez directement dans le système d'exploitation. Vous placez la commande entre anti-apostrophes (```) ou vous utilisez une des trois fonctions PHP : `system()`, `exec()`, ou `passthru()`.

Tant que vous exécutez des commandes en utilisant uniquement des données du script, les commandes sont sûres. Si vous utilisez des données de source externe, le danger est grand. Il est plutôt peu souhaitable que l'utilisateur puisse exécuter une commande comme `rm *`, qui supprime tous les fichiers du répertoire courant.

L'exécution de commandes système présente plus de danger. Vous désirez par exemple permettre à l'utilisateur de renommer un fichier. L'utilisateur saisit dans un formulaire le nom de fichier à modifier, puis vous composez l'instruction suivante dans votre script :

```
system("mv $_POST['ancien_nom'] $_POST['nouveau_nom'] ");
```

Supposons que l'utilisateur tape ceci dans le formulaire :

```
fichier1.txt  
fichier2.txt;rm *
```

L'instruction exécutée sera celle-ci :

```
system("mv fichier1.txt fichier2.txt;rm *");
```

Tous les fichiers du répertoire seront effacés.

En d'autres termes, si vous comptez exécuter des commandes système contenant des données d'une source externe, vous devez contrôler soigneusement l'information. Pour des techniques de vérification et nettoyage des données, voir plus haut la section "Ne faire confiance à aucune information de source externe".

L'emploi de fonctions de fichiers système PHP est beaucoup plus sûr que l'exécution de commandes système via les fonctions système. Par exemple, l'opération précédente est réalisable à l'aide de cette instruction :

```
rename($_POST['ancien_nom'],$_POST['nouveau_nom']);
```

La fonction n'accepte que des noms de fichiers valides, elle est donc beaucoup plus sécurisée. Utilisez une fonction PHP à chaque fois qu'il en existe une qui réalise un travail que vous avez à faire. N'utilisez la fonction générale qui permet d'exécuter n'importe quelle commande système que lorsque c'est absolument nécessaire. Et vérifiez très soigneusement toute information externe.

Gestion des erreurs

Les messages d'erreur affichent des informations lorsque des erreurs sont rencontrées. Certaines de ces informations sont utiles aux individus mal intentionnés. Plus ils en savent sur votre système, plus il est probable qu'ils trouvent un moyen d'y pénétrer. Par exemple, un message d'erreur peut indiquer quelle base de données vous utilisez ou comment la page Web est programmée ou conçue.

Pendant le développement de votre application, il faut certes que PHP vous donne le plus d'informations possibles pour vous aider à identifier les problèmes dans votre script. Cependant, une fois le site accessible au public, il ne faut plus que ces informations s'affichent.

Un moyen est d'inhiber les fonctions d'affichage d'erreur dans `php.ini`. Vous pouvez ainsi interdire l'affichage de messages d'erreur dans vos pages Web. Si nécessaire, journalisez les erreurs dans un fichier que vous consulterez. Il est aussi envisageable de désactiver les fonctions d'affichage d'erreur seulement pour des fichiers spécifiques. En d'autres termes, pendant le développement d'un script, validez la gestion d'erreur pour ce script uniquement.

Les paramètres suivants de `php.ini` sont liés à l'affichage des messages d'erreur :

```
display_errors = On (affiche les messages d'erreur dans une page Web)
log_errors = Off (envoie les messages d'erreur vers un fichier d'historique)
error_log = nom_de_fichier (spécifie le nom du fichier d'historique)
```

Un pirate peut délibérément envoyer des erreurs à votre site Web, provoquant l'affichage de messages d'erreur et donc d'informations qui lui seront utiles. Les paramètres suivants sont plus sécurisants que les précédents :

```
display_errors = Off
log_errors = On
error_log = /var/log/php_error_log
```

La désactivation de `display_errors` empêche l'affichage des messages d'erreur dans la page Web. Lorsque vous désactivez `log_errors`, les messages d'erreur sont transmis au fichier d'historique. Le paramètre `error_log` définit le fichier d'historique.

Pendant le développement du script, placez la ligne suivante en haut du script :

```
ini_set("display_errors","On");
```

Cela valide l'affichage des erreurs pour ce script uniquement. Ainsi, vous verrez les erreurs pendant le développement, et supprimerez l'instruction une fois le script accessible au public.

Sécurité MySQL

Les données stockées dans des bases sont plus sécurisées que dans des fichiers ordinaires, des cookies ou des sessions, car le logiciel de gestion de bases de données apporte ses propres fonctions de sécurité. MySQL fournit un système de sécurité pour la protection de vos données, qui restreint l'accès en se basant sur des noms de compte et des mots de passe. En outre, chaque compte comporte des paramètres de privilèges, qui spécifient ce que l'utilisateur a le droit de faire lors de son accès à MySQL. La sécurité MySQL peut sembler compliquée mais ses fonctions de sécurité fournissent une bonne protection des données.

Configuration de comptes et mots de passe

L'accès à une base de données MySQL est un processus en deux étapes :

1. Connexion au serveur MySQL.

Cette étape nécessite un compte d'utilisateur et un mot de passe valides.

Le serveur MySQL peut être situé sur le même ordinateur que vous utilisez ou sur un autre ordinateur auquel vous accédez via un réseau. Il est plus sûr de placer MySQL sur un ordinateur séparé. Dans l'idéal, l'ordinateur MySQL est derrière un pare-feu. Lorsque l'ordinateur MySQL accepte uniquement le trafic interne, comme le trafic provenant du serveur Web, et pas de trafic externe à l'organisation, il est beaucoup plus sécurisé.

2. Accès aux données de la base.

Une requête SQL est utilisée à cet effet. MySQL fournit un système d'autorisations pour spécifier ce qu'un compte peut faire avec les données. Par exemple, un compte est configuré



de façon que les utilisateurs puissent sélectionner des données mais ni en insérer, ni les modifier.

Lors de l'installation de MySQL, certains comptes sont configurés par défaut. Les informations utilisées pour contrôler l'accès aux données sont stockées dans une base de données MySQL nommée `mysql`.

Base de données de sécurité de MySQL

Lors de l'installation de MySQL, une base de données `mysql` est créée. Toutes les informations servant à protéger vos données sont stockées dans cette base : noms de compte, noms d'hôtes, mots de passe et autorisations.

Les autorisations sont stockées en colonnes. Le format de chaque colonne nom est `permission_priv`, où `permission` est une des autorisations de requête que vous pouvez effectuer auprès de MySQL. Par exemple, la colonne contenant les autorisations de modification `ALTER` se nomme `alter_priv`. La valeur dans chaque colonne de permission est Y ou N, ce qui signifie *oui* ou *non*. Ainsi, par exemple, dans la table `user` (décrite ci-après), on trouve une rangée pour chaque compte, avec son champ `alter_priv`. Si le champ `alter_priv` contient Y, le compte est utilisable pour exécuter une requête de modification `ALTER`. Si `alter_priv` contient N, le compte n'a pas la permission d'exécuter une requête `ALTER`.

La base de données `mysql` comporte les tables suivantes :

- ✔ `user` : Stocke les autorisations s'appliquant à toutes les bases et tables. Elle contient une rangée pour chaque compte valide, avec nom d'utilisateur, nom d'hôte et mot de passe. Le serveur MySQL rejette toute connexion pour les comptes absents de cette table.
- ✔ `db` : Stocke les autorisations qui s'appliquent à une base de données particulière. Elle contient une rangée pour la base de données, définissant les autorisations pour un nom de compte et un nom d'hôte. Pour que les permissions soient accordées, le compte doit exister dans la table `user`. Les autorisations données dans la table `user` écrasent les autorisations de cette table.
- ✔ `host` : Contrôle l'accès à une base de données en fonction de l'hôte concerné. La table `host` travaille avec la table `db`. Si une

ligne de la table `db` comporte un champ vide pour l'hôte, MySQL vérifie dans la table `host` si la base en question y possède une ligne. De cette manière, vous permettez l'accès à une base à partir de certains hôtes mais pas des autres. Par exemple, vous avez deux bases de données : `db1` et `db2`. La base `db1` contient des informations très sensibles, seules certaines personnes doivent pouvoir y accéder. La base `db2` recèle des données que tout le monde peut voir. Si une rangée existe dans la table `db` pour `db1`, avec un champ `host` vide, vous pouvez avoir deux lignes pour `db1` dans la table `host`. Une ligne donnant des autorisations aux utilisateurs se connectant à partir d'un hôte particulier, une autre ligne refusant les privilèges aux utilisateurs se connectant à partir de tout autre hôte.

- ✓ `tables_priv` : Stocke les autorisations qui s'appliquent à des tables spécifiques.
- ✓ `columns_priv` : Stocke les autorisations qui s'appliquent à des colonnes spécifiques.

Vous avez la possibilité de consulter et changer les tables de la base `mysql` si vous utilisez un compte possédant les autorisations nécessaires. Il est possible d'exploiter les requêtes SQL comme `SELECT`, `INSERT`, `UPDATE` et autres.

Configuration de comptes

MySQL est installé avec des comptes par défaut, dont l'un est le compte `root`. Dans certains cas, le compte `root` est installé sans mot de passe. Dans d'autres, la procédure d'installation requiert que vous entriez un mot de passe pour ce compte. Il est indispensable que le compte `root` soit protégé par un mot de passe. S'il est installé sans mot de passe, vous devez lui en attribuer un immédiatement. Un pirate essaiera de rentrer via ce compte `root` sur votre système juste pour voir s'il est protégé ou non.

Le compte `root` dispose de toutes les autorisations, y compris la capacité d'éteindre votre serveur. Il faut donc limiter l'accès à ce puissant compte. N'autorisez jamais les utilisateurs à accéder à votre base de données via ce compte.

Vous devrez définir des comptes spécifiques rien que pour l'accès à vos bases de données à partir de PHP. Et vous ne donnerez à ces



comptes que les permissions réellement indispensables. Lorsqu'un script ne fait que lire des données dans la base de données, seule la permission `SELECT` est nécessaire au compte accédant à la base de données. Vous renforcerez encore la sécurité en utilisant différents comptes pour différents rôles. Vous configurerez par exemple un compte avec privilège `SELECT` uniquement, et un autre compte pour l'usage de requêtes `INSERT` mais sans autorisation `SELECT`. Ainsi, si un compte vient à être piraté, les dégâts potentiels seront limités.

Lorsque vous configurez un compte, vous spécifiez le mot de passe, le nom de l'ordinateur autorisé à accéder à la base de données à l'aide de ce compte, et les permissions. Vous pouvez cependant les modifier ensuite à tout moment.

Vous pouvez ajouter ou modifier un compte en modifiant directement la base de données `mysql` à l'aide de requêtes `INSERT` et `UPDATE`. Vous pouvez aussi employer la requête `GRANT`, une requête SQL pour ajouter ou modifier les comptes.

Ajout de comptes avec la requête `GRANT`

Voici le format général d'une requête `GRANT` :

```
GRANT permission (colonnes) ON nom_de_table
    TO nom_de_compte@nom_hôte IDENTIFIED BY 'mot_de_passe'
```

- ✓ **permission (colonnes) :** Vous devez indiquer au moins une permission. Vous pouvez limiter chaque permission à une ou plusieurs colonnes, en indiquant le nom de colonne entre parenthèses à la suite de l'autorisation. Lorsqu'aucun nom de colonne n'est précisé, la permission est accordée sur toutes les colonnes de la ou les tables. Listez autant d'autorisations et de colonnes que nécessaire, séparées par des virgules. Pour la liste des privilèges possibles, consultez le manuel MySQL (dev.mysql.com/doc/mysql/en/privileges-provided.html). Voici un exemple de début de requête `GRANT` :

```
GRANT select (firstName,lastName), update,
    insert (birthdate) ...
```

- ✓ **nom_de_table :** Indique sur quelles tables la permission est accordée. Une table au moins doit être nommée. Vous pouvez lister plusieurs tables, séparées par des virgules. Les valeurs possibles pour `nom_de_table` sont :

`nom_de_table` : Totalité de la table nommée `nom_de_table` dans la base de données active. Vous disposez de l'astérisque (*) pour indiquer que cela s'applique à toutes les tables de la base de données active. Si vous utilisez un astérisque alors qu'aucune base de données n'est sélectionnée, le privilège sera accordé sur toutes les tables de toutes les bases de données.

`nom_de_base.nom_de_table` : La totalité de la table nommée `nom_de_table` dans la base `nom_de_base`. Vous disposez de l'astérisque (*) à utiliser à la place du nom de la base ou de la table pour signifier *toutes*. Ainsi, *.* accorde la permission sur toutes les tables dans toutes les bases de données.

- ✓ `nom_de_compte@nom_hôte` : Si le compte existe déjà, il se voit accorder les autorisations indiquées. Si le compte n'existe pas, il est ajouté. Le compte est identifié par la paire `nom_de_compte` et `nom_hôte`. Lorsqu'un compte existant porte le nom de compte spécifié mais un nom d'hôte différent, le compte existant n'est pas modifié ; un nouveau compte est créé.
- ✓ `mot_de_passe` : C'est le mot de passe à créer ou modifier. Le mot de passe n'est pas obligatoire. Si vous ne voulez pas ajouter ou changer le mot de passe de ce compte, n'écrivez pas la ligne IDENTIFIED BY 'mot_de_passe'.

Voici un exemple de requête GRANT pour ajouter un nouveau compte pour usage dans les programmes PHP avec une base de données nommée Catalog :

```
GRANT select ON Catalog.* TO utilisateur_php@localhost
IDENTIFIED BY 'mot_de_passe_secret'
```

Si un compte existe déjà, vous pouvez ajouter ou changer le mot de passe via la requête GRANT suivante :

```
GRANT select ON * TO utilisateur_php@nom_hôte
IDENTIFIED BY ''
```

Le mot de passe existant est effacé, laissant le compte sans mot de passe – ce qui n'est généralement pas une bonne idée.

Accès à MySQL à partir de scripts PHP

Pour accéder à MySQL à partir de scripts PHP, vous devez employer le compte `nom_de_compte` et `mot_de_passe` que vous avez défini pour usage dans votre script PHP. Par conséquent, les informations sur le compte doivent être accessibles à vos scripts. Vous devez protéger cette information aussi bien que possible.

Ne placez pas dans le script les informations sur le compte. Stockez-les dans un fichier séparé. Utilisez par exemple un fichier comme celui-ci :

```
<?php
$host = "localhost";
$user = "utilisateur_php";
$password = "asdf321";
$database = "Catalog";
?>
```

Insérez (`include`) ensuite le fichier dans votre script. Votre instruction de connexion MySQL aura donc cette allure :

```
$connect = mysqli_connect($host,$user,$password,$database);
```

Placez le fichier dans un endroit sécurisé, à l'extérieur de votre espace Web, de façon que personne ne puisse y accéder à partir du Web. PHP permet de spécifier un répertoire pour les fichiers inclus. Lorsque vous utilisez l'instruction `include` dans un script, PHP recherche le fichier dans le répertoire des fichiers include.

Voici comment déclarer ce répertoire, dans le fichier `php.ini` :

```
include_path = ".:php/includes"      (Linux, Unix)
include_path = ".;c:\php\includes"   (Windows)
```

Vous pouvez spécifier autant de chemins d'accès que vous voulez. Les deux instructions précédentes déclarent deux chemins : le point (`.`) indique le répertoire courant, et `/php/inclus`. Les chemins sont séparés par des deux-points (`:`) dans Linux, des points-virgules (`;`) dans Windows.

Stockez le fichier contenant les informations sur le compte dans le répertoire des fichiers inclus, puis accédez à ce fichier dans le script à l'aide de l'instruction suivante :

```
include("Vars.inc");
```

PHP recherchera `Vars.inc` dans les chemins spécifiés dans l'instruction `include_path`, dans l'ordre d'énumération.

Attaques par injection de commandes SQL

Lorsque vous utilisez des données de source externe pour construire une requête SQL, vous êtes vulnérable à une attaque par injection de commandes SQL. Dans une *attaque par injection de commandes SQL*, le pirate insère des caractères dans une requête SQL pour la transformer en requête affectant votre application.

Emploi d'apostrophes pour modifier votre requête

Lorsque vous utilisez des données non vérifiées provenant d'un formulaire pour construire une requête SQL, vous fournissez au pirate l'occasion rêvée de créer une instruction SQL servant ses intentions. Supposons par exemple que vous avez rédigé le code suivant pour permettre à un utilisateur de s'identifier :

```
$user = $_POST['user_id'];
$password = $_POST['password'];
$sql = "SELECT COUNT(*) FROM Customer
      WHERE user_id='$user' and password='$password'";
```

Si la requête SQL retourne un compte supérieur à 0, c'est que l'identifiant utilisateur (`user_id`) et le mot de passe (`password`) existent dans la base de données, et l'utilisateur est alors connecté. Dans ce cas, les données entrées dans le formulaire par l'utilisateur sont utilisées sans aucune vérification. Supposez qu'un utilisateur entre le nom d'utilisateur suivant dans le formulaire :

```
' or 1=1 --
```

La requête SQL sera alors :

```
SELECT COUNT(*) FROM Customer WHERE user_id='' or 1=1 --
      and password='$password'
```

Cette requête retourne toujours un compte supérieur à 1 car `1=1` est toujours vrai. De plus, le `--` transforme le reste de la requête en commentaire, ignoré par SQL. Par conséquent, cet utilisateur est maintenant connecté.

L'accès non autorisé n'est pas une bonne chose. En fonction de la structure de votre base de données, un pirate pourra éventuellement construire des requêtes occasionnant de sérieux dégâts à la base de données. Par exemple, une requête sera transformée en plusieurs requêtes, dont une créant des dommages. L'emploi de plusieurs requêtes est étudié à la section suivante.

Création de plusieurs requêtes

De nombreuses bases de données permettent l'exécution de plusieurs requêtes à la fois, habituellement séparées par des points-virgules (;). Un pirate peut être capable de saisir des données dans un champ pour créer une requête SQL malveillante. Prenons l'exemple d'une requête écrite à partir des informations saisies par l'utilisateur dans un formulaire. Voici la requête :

```
SELECT email FROM Member WHERE last_name='$_POST['name'] '
```

Disons que l'utilisateur entre le texte ci-dessous dans la case name (Nom) du formulaire :

```
Smith';DROP TABLE Member --
```

La requête devient :

```
SELECT email FROM Member
WHERE last_name='Smith';DROP TABLE Member --
```

Une fois cette requête exécutée, vous vous demanderez où est passée la table Member (Membres).

Ce genre de requête est moins dangereuse dans MySQL que dans d'autres bases de données. Jusqu'à MySQL 4.1, MySQL n'acceptait d'ailleurs pas plusieurs requêtes jointes en une seule. Depuis MySQL 4.1, il est possible d'exécuter plusieurs requêtes mais à l'aide d'une fonction particulière, `mysqli_multi_query`. Lorsque vous n'utilisez pas cette fonction, l'utilisateur ne peut utiliser le type d'attaque présentée dans la requête précédente.

Échappement des apostrophes

Une mesure pour protéger votre base de données est de procéder à l'échappement des apostrophes dans toute information transmise à la base de données. Un \ devant une apostrophe traite cette dernière

comme caractère littéral, pas comme caractère spécial. L'apostrophe est stockée dans la base de données sous forme de caractère, au lieu de déclencher une action. Si l'exemple de la section précédente était échappé, cela donnerait :

```
SELECT COUNT(*) FROM Customer WHERE user_id='\' or 1=1 --
and password='$password'
```

SQL verrait le symbole `\'` comme un caractère ordinaire, pas comme l'apostrophe refermant le champ `user_id`. Serait donc prise en compte comme `user_id` la totalité de la chaîne jusqu'à l'apostrophe devant `$password`. Or, sous cette forme, cette requête SQL n'est pas valide. Elle provoquerait donc une erreur.

PHP fournit deux fonctions pour l'échappement des apostrophes, guillemets et autres caractères :

- ✓ **magic_quotes** : Paramètre de `php.ini`, activé par défaut. Lorsque le paramètre `magic_quotes` est activé, les données reçues par le script PHP sont automatiquement échappées. En d'autres termes, dans l'information lue dans le tableau `$_POST`, les apostrophes et guillemets sont déjà échappés.
- ✓ **La fonction `addslashes`** : Cette fonction PHP échappe notamment les apostrophes. La fonction contraire `stripslashes` supprime les slashes lorsque nécessaire, lorsque vous désirez afficher l'information sur une page Web sans les slashes.

Le paramètre `magic_quotes` est activé par défaut. Cependant, toutes les données ne sont pas transmises à la base de données. Par exemple, lorsque vous affichez des données sans les stocker dans la base, il faut en ôter les slashes. En fait, en général, il est préférable de désactiver `magic_quotes`. Certes ce paramètre est pratique pour protéger les débutants de certaines erreurs dangereuses, l'ajout et la suppression de slashes avec les fonctions citées donnent plus de contrôle sur le script. Il suffit de se souvenir d'ajouter des slashes à toute donnée transmise vers la base de données.

Vérification et filtrage des données

L'échappement des guillemets dans les données envoyées à la base de données n'est pas suffisant pour empêcher toutes les attaques par injection de commandes SQL. Les pirates sont très créatifs. Vous devez vérifier toutes données reçues de sources externes.

Au minimum, vous devez vérifier les points-virgules. Ils sont rarement requis dans les données légitimes. Vous connaissez quelqu'un prénommé Je;an ? Dans la plupart des cas, il est possible de vérifier que l'information est dans un format raisonnable. Pour en savoir plus sur la vérification de vos données, voir la section "Ne faire confiance à aucune information de source externe" plus haut dans ce chapitre.

Sauvegarde des bases de données

Vous devez sauvegarder vos bases de données. Si une attaque détruit vos bases de données, vous devez être capable de les remplacer rapidement. MySQL fournit un utilitaire pour créer des sauvegardes.

Il s'agit de l'outil `mysqldump` situé dans le sous-répertoire `bin` du répertoire où MySQL est installé. Le démarrage de `mysqldump` s'opère en tapant une commande. Dans Windows, vous devez ouvrir une fenêtre de ligne de commande.

Voici la syntaxe de la commande pour sauvegarder une base :

```
mysqldump --user=nom_de_compte--password=mot_de_passe nom_de_base
>chemin/fichier_de_sauvegarde
```

Placez-vous éventuellement dans le répertoire `bin` pour taper la commande. Ou indiquez le chemin d'accès à `mysqldump` en entrant la commande :

```
c:\mysql\bin\mysqldump ...
```

Pour sauvegarder une base nommée `Catalog`, vous utiliserez cette commande :

```
mysqldump --user=utilisateur_php --password=gros_secret Catalog
>../backups/Catalog.bak
```

Après exécution de `mysqldump`, le fichier `/backups/Catalog.bak` contient les requêtes SQL nécessaires pour recréer la base `Catalog`.

Sauvegardez régulièrement, au moins une fois par jour. Si votre base de données change fréquemment, sauvegardez plus souvent. Par exemple, sauvegardez chaque heure vers le répertoire de sauvegarde, et vers un autre ordinateur une fois par jour. Pour les don-

nées extrêmement importantes, il faut plusieurs sauvegardes, dont une stockée hors-site, en cas d'incendie.

Emploi d'un serveur Web sécurisé

Le serveur Web et les navigateurs des internautes communiquent par messages. Le navigateur requiert un fichier contenant l'HTML d'une page Web. Le serveur renvoie un fichier contenant les informations nécessaires. La communication entre serveur et navigateur n'est pas sécurisée. Quelqu'un se trouvant sur Internet entre vous et la personne demandant vos pages Web peut lire les messages échangés. Si votre site recueille ou envoie des numéros de carte bancaire ou autres informations confidentielles, vous devez utiliser un serveur Web sécurisé pour protéger ces données.

Les serveurs Web sécurisés utilisent SSL (Security Sockets Layer) pour protéger les communications vers et en provenance des navigateurs. L'information est *cryptée* (traduite en chaînes codées) avant d'être véhiculée sur le Web. Le logiciel qui reçoit la décode pour reconstituer le contenu original. De plus, le site Web utilise un certificat qui vérifie votre identité. L'emploi d'un serveur Web sécurisé constitue un travail supplémentaire mais indispensable pour certaines applications.

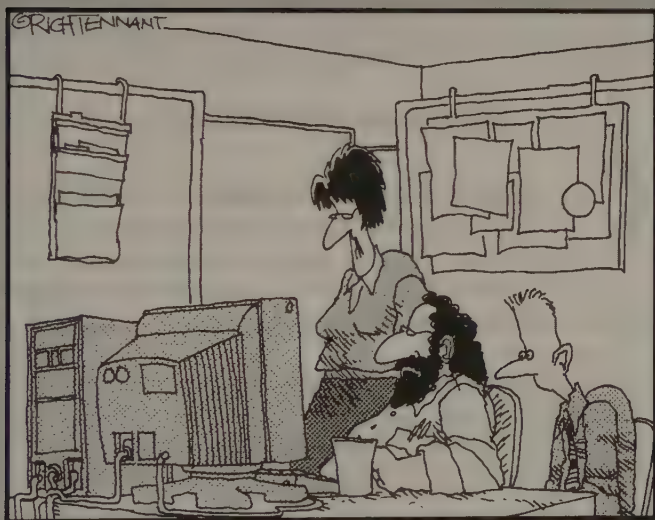


Lorsqu'une communication est sécurisée à l'aide de SSL, l'URL commence par *HTTPS* au lieu de *HTTP*.

Les détails des serveurs Web sécurisés sont spécifiques au serveur Web exploité. Pour en savoir plus sur SSL, visitez le site Web concernant votre serveur Web. Par exemple, si vous utilisez Apache, visitez ces sites open source www.modssl.org et www.apache-ssl.org. Du logiciel commercial est également disponible pour sécuriser le serveur Web Apache. Si vous travaillez avec Microsoft Internet Information Server (IIS), recherchez SSL sur le site Web Microsoft www.microsoft.com.

Cinquième partie

Création d'une application d'authentification d'utilisateur



"Ce que je veux, ce sont des applications Web
et du contenu dynamiques, pas des intentions,
des sous-entendus et des mimiques."

Dans cette partie...

Cette partie propose deux applications qui limitent l'accès à un site Web (ou aux sections d'un site) aux seuls utilisateurs autorisés. Les deux applications exploitent deux approches différentes – procédurale et orientée objet – pour filtrer les utilisateurs.

Chapitre 12

Authentification d'utilisateur avec HTTP

Dans ce chapitre :

Fonctionnement des en-têtes HTTP.

Authentification HTTP avec Apache.

Conception d'un script PHP utilisant l'authentification HTTP des utilisateurs.

Écriture du script PHP utilisant l'authentification HTTP des utilisateurs.

De nombreuses applications réclament à l'utilisateur de s'identifier. Par exemple, la plupart des applications d'achat en ligne réclament à l'utilisateur de s'identifier avant de commander des produits. Il arrive que l'on ne puisse visionner la moindre page d'un site sans avoir auparavant saisi un mot de passe. Parfois, seule une partie du site est ainsi protégée.

Le mécanisme pour l'authentification d'utilisateur est intégré à HTTP (Hypertext Transfer Protocol), le langage servant aux serveurs Web et navigateurs pour communiquer lors du transfert de contenu de pages Web. Pour un rappel sur le World Wide Web et HTTP, voir ci-après la section "Authentification HTTP".

Vous pouvez exploiter les fonctions d'authentification d'utilisateur intégrées à HTTP ou construire votre propre application d'identification à partir de zéro. L'authentification HTTP est plus rapide et plus facile car elle est intégrée, et donc prête à recueillir et vérifier les identifiants et mots de passe d'utilisateurs ; vous n'avez pas de code à écrire pour réaliser cette tâche. Cependant, l'authentification HTTP présente certains inconvénients :

- ✓ **Aspect** : La fenêtre réclamant le mot de passe est présentée par le navigateur de l'internaute. C'est une fenêtre grise ordinaire, où l'utilisateur saisit son identifiant et son mot de passe. Vous ne pouvez contrôler son aspect.
- ✓ **Comportement** : La réponse aux identifiants et mots de passe valides et invalides est gérée par le navigateur. Par exemple, Internet Explorer ne permet que trois entrées invalides, à la suite de quoi il affiche un message d'erreur, alors que Netscape autorise l'utilisateur à continuer à saisir les informations à volonté. Vous ne pouvez pas non plus changer ce comportement.
- ✓ **Mots de passe** : Les ID et mots de passe valides doivent être stockés à l'avance. La fonction d'authentification HTTP compare l'information saisie par l'utilisateur avec celle qui est stockée. L'authentification HTTP ne fournit aucune possibilité à l'utilisateur de s'inscrire en ligne.

Elle n'est donc utile que pour les applications simples. Par exemple, pour protéger des sections de votre intranet, pour ne les rendre accessibles qu'à certains membres du personnel d'un service particulier, comme la comptabilité ou les ventes. Une simple protection par mot de passe, où l'utilisateur doit saisir un mot de passe qui lui a été fourni par la direction de son service, fonctionne bien dans ce genre de situation. Du fait que la page Web est interne à l'entreprise, son aspect est moins crucial que celui des pages Web présentées aux clients. Ce chapitre montre comment développer une application utilisant l'authentification HTTP.

Pour une application plus compliquée, par exemple où l'utilisateur peut s'inscrire en ligne, ou si l'aspect de l'application est important, construisez l'application en écrivant votre propre code pour remplir ces fonctions. Vous trouverez au Chapitre 13 de quoi construire de bout en bout une application d'authentification d'utilisateur.

Authentification HTTP

L'authentification HTTP est intégrée à HTTP (Hypertext Transfer Protocol). HTTP est le langage utilisé par les navigateurs et les serveurs Web pour communiquer. HTTP est la fondation sur laquelle est bâti le Web.

Fonctionnement du Web

Lorsqu'un utilisateur tape une URL dans un navigateur, celui-ci envoie un message HTTP sur le Web, qui demande à voir le fichier spécifié par l'URL. La requête HTTP peut avoir cet aspect :

```
GET /Index.php HTTP/1.1
```

Outre la requête initiale à recevoir le fichier, d'autres messages HTTP sont envoyés. Ce sont les *en-têtes HTTP*. Par exemple, un en-tête est envoyé pour indiquer où se trouve le fichier :

```
GET /index.php HTTP/1.1
```

```
Host: www.monentreprise.com
```

Lorsque le serveur Web de `www.monentreprise.com` reçoit la requête, il recherche dans le répertoire racine de son espace Web un fichier nommé `index.php`. S'il trouve le fichier, le serveur Web envoie certains en-têtes HTTP au navigateur requérant, contenant des informations sur ce qui lui est envoyé, puis le contenu de `index.php`. Par exemple, le serveur Web envoie une ligne d'état comme celle-ci :

```
HTTP/1.1 200 OK
```

Cette ligne d'état informe le navigateur que le fichier a été trouvé. Lorsque le fichier n'est pas trouvé, la ligne d'état suivante est envoyée :

```
HTTP/1.1 404 Not found
```

Des en-têtes supplémentaires sont éventuellement transmis à la suite de la ligne d'état, comme ici :

```
.HTTP/1.1 200 OK
```

```
Date: Mon, 31 May 2004 10:05:44 GMT
```

```
Content-Type: text/html
```

```
Content-length: 210
```

Les en-têtes fournissent la date et l'heure, et indiquent au navigateur quel type d'information attendre, ainsi que la longueur du fichier transmis. Une fois envoyés les en-têtes HTTP, le contenu du fichier demandé est transmis.

Requête d'un fichier protégé par mot de passe

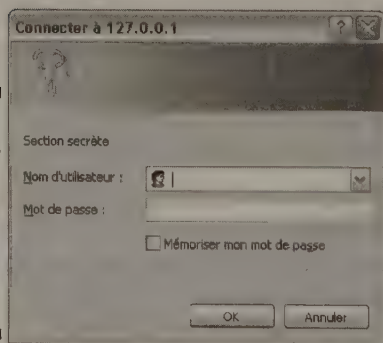
Dans certains cas, lorsque le serveur Web reçoit une demande de fichier, il constate que le fichier est protégé par mot de passe. Par exemple, avec le serveur Web Apache, vous pouvez spécifier que tous les fichiers d'un répertoire nécessitent que l'utilisateur entre un mot de passe avant qu'Apache transmette au navigateur le contenu du fichier. Pour savoir comment désigner les fichiers à protéger par mot de passe, voir plus loin dans ce chapitre.

Lorsque le serveur Web reçoit une requête pour un fichier protégé, il répond avec la ligne d'état suivante :

```
HTTP/1.1 401 Unauthorized
WWW-Authenticate: Basic realm="Domaine"
```

Lorsque le navigateur reçoit la ligne d'en-tête d'authentification, il affiche une boîte de dialogue demandant à l'utilisateur d'entrer un nom et un mot de passe. L'affichage est semblable d'un navigateur à l'autre. La Figure 12.1 montre la boîte de dialogue affichée par Internet Explorer.

Figure 12.1 : Cette boîte de dialogue réclame un nom d'utilisateur et un mot de passe.



Lorsque l'utilisateur a saisi un nom et un mot de passe, le navigateur envoie une seconde requête pour le même fichier, suivie d'un en-tête contenant le nom et le mot de passe entrés par l'utilisateur :

```
GET /SecretFile.php HTTP/1.1
Host: www.monentreprise.com
Authorization: Basic chaîne-de-caractères
```

chaîne-de-caractères contient le nom d'utilisateur et le mot de passe, codés et séparés par un deux-points. Le nom d'utilisateur et le mot de passe sont codés pour les besoins de transmission, pas cryptés pour la sécurité. Si vous écrivez une application à haute sécurité, vous devez employer SSL (Secure Sockets Layer) pour protéger l'information pendant la transmission (voir le Chapitre 11).

Autoriser l'accès

Lorsque le serveur Web reçoit la requête de fichier avec l'en-tête d'autorisation, le nom d'utilisateur et le mot de passe inclus doivent être testés pour déterminer s'ils sont valides. Apache vérifie automatiquement le mot de passe en se basant sur les directives Apache qui indiquent au serveur quelles combinaisons nom d'utilisateur/mot de passe sont valides. Si l'information est valide, le contenu du fichier demandé est affiché. Lorsque la combinaison nom d'utilisateur/mot de passe n'est pas valide, l'utilisateur a la possibilité de ressaisir l'information une ou plusieurs fois selon le navigateur employé. La configuration d'un site pour l'authentification HTTP Apache est décrite à la section "Authentification HTTP avec Apache".

Vous pouvez aussi écrire du code PHP pour vérifier la combinaison nom d'utilisateur/mot de passe transmise dans l'en-tête d'autorisation. L'information est accessible dans le script PHP à l'aide de `$_SERVER['PHP_USER_AUTH']` et `$_SERVER['PHP_USER_PW']`. Vous verrez plus loin dans ce chapitre comment construire une application d'identification avec cette méthode.

Authentification HTTP avec Apache

Il est possible de configurer l'authentification HTTP en utilisant seulement Apache, sans PHP ou MySQL. Apache réclame alors automatiquement un nom d'utilisateur et un mot de passe, et compare l'information saisie par l'utilisateur avec une liste de noms d'utilisateur et mots de passe valides. Bien que cette démarche soit rapide et facile à configurer, elle est aussi un peu simpliste, rigide, de portée et fonction réduites, et lente lorsque le nombre de noms d'utilisateur et mots de passe valides est élevé. Elle convient cependant parfaitement à certains types d'identification d'utilisateur, autorisant une authentification rapide et facile lorsque le nombre d'utilisateurs est restreint et que vous n'avez pas besoin de plus

compliqué. Par exemple, vous désirez protéger un répertoire de votre intranet pour chaque service, remettre à chaque employé un nom d'utilisateur et un mot de passe.

Vous utiliserez des instructions Apache, dites *directives*, pour configurer votre application d'authentification. Les directives vous permettent de spécifier les fichiers protégés par mot de passe et les noms d'utilisateur et mots de passe valides. Le plus simple est de créer un fichier nommé `.htaccess` (notez le point au début du nom de fichier).

Configuration d'Apache

Au démarrage d'Apache, les directives de son fichier de configuration sont exécutées. Le fichier s'appelle habituellement `httpd.conf`, il est situé dans le sous-répertoire `conf` du répertoire où est installé Apache. Ce fichier de configuration autorise généralement l'accès à toutes les pages Web stockées dans votre espace Web – `htdocs` par défaut, à moins que l'emplacement ait été modifié dans le fichier de configuration d'Apache. Pour limiter l'accès à certains fichiers, on place un fichier `.htaccess` dans le répertoire à protéger par mot de passe. Ce fichier contient des directives Apache spécifiant l'authentification. Les directives Apache inscrites dans le fichier `.htaccess` remplacent les directives présentes dans le fichier de configuration exécuté au démarrage d'Apache. Cependant, une directive Apache du fichier de configuration peut indiquer à Apache d'ignorer les commandes du fichier `.htaccess`. Avant de pouvoir protéger des répertoires par mot de passe à l'aide d'un fichier `.htaccess`, vous devez être sûr qu'Apache puisse exécuter les directives écrites dans le fichier `.htaccess`.

Pour cela, ouvrez le fichier `httpd.conf`. Dans Windows, choisissez Démarrer/Programmes/Serveur HTTPD Apache/Configurer Serveur Apache. Cherchez dans le fichier une ligne semblable à la suivante, elle identifie le répertoire racine de votre espace Web :

```
<Directory "C:/Program Files/Apache Group/Apache/htdocs">
```

Cette ligne signifie que les directives Apache trouvées au-dessous ne s'appliquent qu'à ce répertoire. Peu après cette ligne, vous devriez trouver une ligne comme celle-ci :

```
AllowOverride None
```

Cette ligne indique à Apache d'ignorer toute directive de tout fichier `.htaccess`. Vous pouvez changer cette ligne en celle-ci :

```
AllowOverride Authconfig
```

Cette ligne autorise le remplacement des directives d'authentification exécutées au démarrage d'Apache par les directives d'authentification HTTP présentes dans le fichier `.htaccess`.

Création du fichier `.htaccess`

Le fichier `.htaccess` contient les directives Apache nécessaires à l'authentification HTTP. Créez un fichier `.htaccess` (notez que le nom de fichier commence par un point) dans le répertoire à protéger par mot de passe. Le fichier `.htaccess` précise les directives qui s'appliquent au répertoire dans lequel il est stocké, et tout sous-répertoire de ce répertoire. Voici le contenu du fichier :

```
AuthUserFile "c:\secret\htpass
AuthGroupFile /dev/null
AuthName "Service Compta "
AuthType Basic
Require valid-user
```

Voici quelques explications sur le contenu du fichier :

- ✓ La première ligne déclare un fichier contenant la liste des utilisateurs valides de ce répertoire. Les procédures pour créer le fichier des mots de passe sont étudiées dans la section suivante : "Création du fichier des mots de passe".

Stockez le fichier des mots de passe dans un répertoire à l'extérieur de votre espace Web. C'est beaucoup plus sécurisé que dans un répertoire auquel les visiteurs de votre site pourraient accéder. Ici, le répertoire est `c:\secret`, en dehors de `htdocs`.

- ✓ La seconde ligne désigne un fichier de groupe ; ici, aucun fichier de groupe n'existe (indiqué par `/dev/null`). Un fichier de groupe contient la liste des groupes autorisés à accéder aux fichiers du répertoire.
- ✓ `AuthName` est le nom de la section à protéger. Choisissez le nom que vous voulez. Il sera affiché dans la fenêtre où l'utilisateur doit entrer ses nom et mot de passe.



- ✓ AuthType reçoit ici la valeur Basic. L'autre type d'authentification est Digest, plus sécurisé mais également plus compliqué. Nous n'étudions pas l'authentification Digest dans ce livre.
- ✓ La dernière ligne précise que seul un utilisateur valide – c'est-à-dire trouvé dans le fichier spécifié en première ligne – peut accéder aux fichiers de ce répertoire.

Le fichier `.htaccess` s'applique au répertoire dans lequel il est stocké et à tous ses sous-répertoires. Cependant, un autre fichier `.htaccess` placé dans un sous-répertoire de ce répertoire peut écraser les directives de ce fichier `.htaccess`.

Création du fichier des mots de passe

Le fichier des mots de passe contient les paires de nom d'utilisateur/mot de passe valides. Apache fournit un programme pour créer ce fichier de mots de passe. Ce programme, `htpasswd`, est automatiquement installé dans le sous-répertoire `bin` du répertoire où Apache est installé, par exemple `c:\Apache\bin\htpasswd.exe`.

Pour créer un fichier de mots de passe, tapez une commande comme celle-ci sur la ligne de commande :

```
c:\Apache\bin\htpasswd -c c:\secret\htpass janet
```

Cette commande crée un fichier nommé `.htpass` dans le répertoire `c:\secret`. Elle ajoute une ligne au fichier, pour le nom d'utilisateur `janet`, puis vous invite à saisir le mot de passe de l'utilisateur `janet`. Le `-c` de la commande signifie créer, il ne doit être utilisé que lorsque vous créez le fichier. Vous ajouterez ensuite dans le fichier des lignes nom d'utilisateur/mot de passe :

```
c:\Apache\bin\htpasswd c:\secret\htpass boss
```



Si vous utilisez `-c` dans la commande pour ajouter une ligne à un fichier de mots de passe existant, celui-ci sera écrasé par un nouveau fichier portant le même nom ; les lignes du fichier existant seront donc perdues. N'utilisez donc pas `-c` à moins de vouloir créer un nouveau fichier.

Le nom du fichier contenant les noms d'utilisateur valides est à votre choix. Le nom `.htpass`, couramment utilisé, n'est pas obligatoire. Les données stockées dans le fichier ont cette allure :


```
janet:$apr1$Hr.....$DS8EPQBqbxRXt9hUFoq3/
```

Notez que le mot de passe n'est pas stocké dans une forme lisible par l'humain.

Vous avez le droit de stocker le fichier des mots de passe n'importe où sur votre ordinateur mais il est préférable de le placer à l'extérieur de votre espace Web, pour plus de sécurité. Pour indiquer à Apache où se trouve le fichier, écrivez dans le fichier `.htaccess` une ligne semblable à celle-ci :

```
AuthUserFile /usr/local/apache/secret/.htpass
```

Cette commande indique à Apache de rechercher les noms d'utilisateur valides dans le fichier `/usr/local/apache/secret/.htpass`.

L'authentification HTTP Apache en action

Après création des fichiers `.htaccess` et de mots de passe, Apache met en place l'authentification d'utilisateur sans que vous n'ayez plus à intervenir. Lorsque Apache reçoit une requête pour recevoir un fichier d'un répertoire contenant le fichier `.htaccess`, la boîte de dialogue requérant un nom d'utilisateur et un mot de passe est automatiquement affichée. Lorsque l'utilisateur clique sur le bouton pour soumettre le nom et le mot de passe, Apache compare l'information à la liste des paires valides nom d'utilisateur/mot de passe dans votre fichier de mots de passe. S'il trouve une correspondance, il envoie le contenu de la page Web au navigateur de l'utilisateur.

Le comportement du navigateur lorsque le nom d'utilisateur et mot de passe sont absents du fichier des mots de passe varie d'un navigateur à l'autre. La majorité des navigateurs offre à l'utilisateur la possibilité d'annuler en cliquant sur un bouton. Lorsque l'utilisateur clique sur Annuler, le message *Authorization Required* (Autorisation nécessaire) est affiché.

Lorsque l'utilisateur entre une combinaison valide nom d'utilisateur/mot de passe, il est autorisé à visionner toutes les pages Web protégées par mot de passe, pendant la durée de la session. L'utilisateur n'a donc pas à ressaisir le mot de passe à chaque fois qu'un nouveau fichier est demandé au serveur. L'autorisation est valide jusqu'à fermeture du navigateur, ce qui achève la session.

Conception d'une application d'authentification HTTP en PHP

Le reste de ce chapitre étudie l'emploi de l'authentification HTTP dans les scripts PHP. La fonction élémentaire de l'application d'authentification HTTP est de protéger les fichiers. Seuls les utilisateurs qui ont entré des noms d'utilisateur et mots de passe valides sont autorisés à visionner les pages protégées. L'application d'authentification HTTP nécessite un jeu existant de noms d'utilisateur et mots de passe, stocké dans un fichier. L'application compare le nom d'utilisateur et le mot de passe fournis à la liste des informations valides stockées dans un fichier. Lorsque les données fournies correspondent à une paire présente dans le fichier, l'utilisateur se voit accorder l'accès aux fichiers protégés.

Voici les détails de fonctionnalité de l'application :

1. Vérifier si le navigateur a envoyé un nom d'utilisateur et un mot de passe avec la requête pour la page Web protégée.
2. Si un nom d'utilisateur et un mot de passe ne sont pas inclus dans la requête pour la page Web, demander à l'utilisateur d'entrer un nom et un mot de passe.
3. Lorsque l'utilisateur entre un nom et un mot de passe, retransmettre la requête de page Web, en ajoutant un en-tête contenant le nom et le mot de passe saisis par l'utilisateur.
4. Lorsqu'un nom et un mot de passe sont inclus avec la requête de page Web, vérifier si le nom d'utilisateur et le mot de passe sont dans la liste des informations d'utilisateur valides. Si ce n'est pas le cas, afficher un message d'erreur et donner à l'utilisateur l'occasion d'entrer d'autres nom et mot de passe.
5. Lorsque l'utilisateur a entré un nom et un mot de passe valides, affichez le contenu de la page Web dans le navigateur de l'utilisateur.



Pour utiliser l'authentification HTTP avec PHP sous Windows, il faut que PHP soit installé comme module Apache, pas comme CGI. Pour plus d'informations, consultez l'encadré "Emploi de l'authentification HTTP avec PHP sous Windows".

Emploi de l'authentification HTTP avec PHP sous Windows

Un script PHP d'authentification qui utilise des en-têtes HTTP pour réclamer les informations de login à l'utilisateur fonctionne uniquement lorsque PHP est installé comme module Apache, pas comme CGI. Vous pouvez savoir le mode PHP de votre serveur en vérifiant le fichier de configuration Apache `httpd.conf`. Ce fichier est habituellement stocké dans le sous-répertoire `conf` du répertoire où Apache est installé, comme `c:\Program Files\Apache Group\Apache\conf\httpd.conf`.

Ouvrez `httpd.conf`. Il est habituellement possible de l'ouvrir en sélectionnant une commande dans le menu Démarrer/Programmes. Par exemple, sous Windows 2000, choisissez Démarrer/Programmes/Serveur HTTPD Apache/Configurer Serveur Apache/Modifier Fichier de configuration `httpd.conf`. Le fichier de configuration est ouvert dans l'éditeur par défaut, comme le Bloc-notes ou WordPad. Cherchez une de ces deux lignes :

```
LoadModule php5_module "c:/php/php5apache.dll"
Action application/x-httpd-php /php/php-cgi.exe
```

Si vous utilisez Apache 2, la ligne `LoadModule` a cet aspect :

```
LoadModule php5_module "c:/php/php5apache2.dll"
```

La directive de configuration Apache `LoadModule` démarre PHP comme module Apache. La directive `Action` démarre PHP en mode CGI. Dans de nombreux cas, les deux directives sont dans le fichier `httpd.conf` mais l'une d'elles commence par un `#` : c'est donc un commentaire, pas une directive active.

Si la directive `Action` est active (en d'autres termes, la directive `LoadModule` est absente ou est précédée d'un `#`), PHP est installé comme CGI. Pour que l'authentification HTTP fonctionne dans vos scripts PHP, vous devez passer PHP en module Apache. Pour cela, suivez ces étapes :

1. Trouvez dans votre fichier `httpd.conf` la ligne `LoadModule` mentionnée plus haut. Si vous la trouvez précédée d'un `#`, supprimez le `#`. Si vous ne la trouvez pas, ajoutez-la.
2. Recherchez la ligne `Action` mentionnée auparavant, et insérez un `#` en début de ligne.
3. Recherchez la ligne suivante dans le fichier `httpd.conf`, et supprimez le `#` en début de ligne. Si vous ne trouvez pas cette ligne dans le fichier `httpd.conf`, ajoutez-la – sans le `#`.

```
#AddModule mod_php5.c
```

4. Assurez-vous que le fichier `php.ini` est dans votre répertoire système (Win98/XP : `Windows` ; Win2000 : `Winnt`).
5. Vérifiez que le fichier `php_mysql.dll` ou `php_mysqli.dll` est dans votre répertoire système (Win98 : `Windows\system` ; Win2000 : `Winnt\system32` ; WinXP : `Windows\system32`). Si le fichier `dll` de MySQL n'est pas dans votre répertoire système, vérifiez le fichier `php.ini` pour savoir lequel est actif dans votre installation, et copiez le fichier correct dans le répertoire système.
6. Redémarrez Apache.

Il est probable que vous disposiez d'une commande Démarrer/Programmes/ Serveur HTTP Apache/Contrôle du serveur Apache/Redémarrer.

Si vous exploitez une installation PHP 4, les étapes seront légèrement différentes. Par exemple, le module `dll` s'appelle `php4_apache.dll`. Pour plus d'instructions sur PHP 4, consultez le site Web de PHP : www.php.net.

Création d'une base de données d'utilisateurs

Dans l'application d'authentification HTTP, il existe une liste de noms d'utilisateur et mots de passe valides, et les nom d'utilisateur et mot de passe saisis par l'utilisateur sont comparés aux informations stockées, pour vérifier s'ils sont valides. La liste des informations valides peut être stocké dans un fichier ordinaire ou dans une base de données. Dans cet exemple, les informations d'utilisateur valides sont stockées dans une base de données MySQL. Les sections suivantes expliquent comment créer la base de données.

Conception de la base de données d'utilisateurs

Le concept de cette base de données est simple : la base doit contenir un nom d'utilisateur et un mot de passe pour chaque compte d'utilisateur. La date à laquelle le compte a été créé est également une information utile.

Dans cet exemple, la base de données est nommée `UserAccount` (compte utilisateur). Elle contient une table `Valid_User` (utilisateur valide). Les caractéristiques de la table sont présentées au Tableau 12.1. La table contient trois champs : `user_name` (nom

d'utilisateur), password (mot de passe) et create_date (date de création). Tous les champs sont obligatoires ; aucun ne peut être vide. La clé primaire est user_name.

Tableau 12.1 : Table de base de données : Valid_User.

Nom de variable	Type	Description
user_name	CHAR(10)	Nom d'utilisateur du compte (clé primaire)
password	CHAR(255)	Mot de passe du compte
create_date	DATE	Date à laquelle le compte a été ajouté à la table

Création de la base de données d'utilisateurs

L'instruction SQL suivante crée la base de données d'utilisateurs :

```
CREATE DATABASE UserAccount;
```

L'instruction SQL suivante crée la table :

```
CREATE TABLE Valid_User (
    user_name    CHAR(10) NOT NULL,
    password     CHAR(255) NOT NULL,
    create_date  DATE     NOT NULL,
    PRIMARY KEY(user_name) );
```

Accès à la base de données d'utilisateurs

Vous pouvez accéder à la base de données à partir d'un script PHP avec les fonctions MySQL intégrées à PHP. Vous utiliserez les fonctions mysql ou mysqli, selon les versions de PHP et MySQL exploitées selon le module de fonction activé dans le fichier php.ini. Dans ce livre, les applications utilisent les fonctions mysqli. Les différences de versions PHP et MySQL sont expliquées au Chapitre 10.

Vous devez fournir les informations nécessaires aux fonctions MySQL, tels que nom de compte et mot de passe MySQL. Cela n'a pas de lien avec d'autres noms de compte et mots de passe que vous utilisez, par exemple pour vous identifier sur votre ordinateur.



Les fonctions `mysqli` ne sont pas disponibles, dans PHP 4. Vous devez employer les fonctions `mysql`. La syntaxe est légèrement différente.

Pour notre exemple d'application, j'ai stocké les informations nécessaires aux fonctions `mysqli` de PHP dans un fichier séparé, nommé `Vars.inc`. Pour des raisons de sécurité, ce fichier est stocké dans un répertoire en dehors de mon espace Web. Le fichier contient des informations semblables à celles-ci :

```
<?php
    $host = "localhost";
    $user = "admin";
    $passwd = "mdpadmin";
    $database = "useraccount";

?>
```



Notez les balises PHP au début (`<?php`) et à la fin (`?>`) du fichier. Si ces balises ne sont pas incluses, l'information peut s'afficher sur la page Web, aux yeux de tous, ce qui n'est pas souhaitable.

Construction de l'application d'authentification en PHP : démarche procédurale

Un unique script fournira toutes les fonctionnalités nécessaires à cette application. Au démarrage de ce script `Auth.php`, l'utilisateur est invité à entrer un nom d'utilisateur et un mot de passe. Lorsque l'utilisateur clique sur le bouton pour soumettre ses nom et mot de passe, le programme regarde si la combinaison nom d'utilisateur/mot de passe est valide.

Le flux du script est contrôlé par une instruction `if` qui teste si un nom d'utilisateur et un mot de passe ont été saisis. Voici le déroulement général du script :

```
si (le nom d'utilisateur et le mot de passe n'ont pas été soumis)
    Demander à l'utilisateur d'entrer un nom d'utilisateur
    et un mot de passe

sinon (le nom d'utilisateur et le mot de passe ont été soumis)
    1 Regarder si le nom d'utilisateur et le mot de
    passe correspondent à un nom et un mot de passe
    valides dans la base de données d'utilisateurs.
```

- 2 Si le nom d'utilisateur et le mot de passe sont valides, affichez le contenu de la page Web protégée.
- 3 Si le nom d'utilisateur et/ou le mot de passe ne sont pas valides, inviter l'utilisateur à s'identifier de nouveau.

Le Listing 12.1 présente le script de l'application d'authentification HTTP (Auth.php).

Listing 12.1 : Script pour demander et tester un nom d'utilisateur et un mot de passe

```
<?php
/* Prog. : Auth.php
 * Desc. : Programme réclamant un nom d'utilisateur et un
 *         mot de passe, via l'authentification HTTP.
 *         Le programme teste si le nom d'utilisateur et
 *         le mot de passe correspondent à un nom et un mot
 *         de passe stockés dans une base de données MySQL.
 */

// Regarde si l'utilisateur a été invité à entrer son nom
if(!isset($_SERVER['PHP_AUTH_USER'])) #10
{
    header('WWW-Authenticate: Basic realm="Section secrète");
    header('HTTP/1.0 401 Unauthorized'); #13
    exit("Cette page nécessite une authentification!"); #14
} #15

// Vérifie le nom d'utilisateur et mot de passe saisis
else #18
{
    include("Vars.inc"); #20
    $user_name = trim($_SERVER['PHP_AUTH_USER']); #21
    $user_password = trim($_SERVER['PHP_AUTH_PW']);
    $connection = mysql_connect($host,$user,$passwd)
        or die ("Echec de connexion au serveur."); #24
    $db = mysql_select_db($database,$connection)
        or die ("Echec de sélection de la base.");
    $sql = "SELECT user_name FROM Valid_User
            WHERE user_name = '$user_name'
            AND password = md5('$user_password')";
    $result = mysql_query($sql)
        or die("Echec d'exécution de la requête."); #31
    $num = mysql_num_rows($result); #32
    if ($num < 1) // nom d'utilisateur/mot de passe introuvable #33
```

```

    {
        exit("Le nom d'utilisateur ou le mot de passe saisi
            est invalide.<br>");
    }
}
// Contenu de page Web.
include("Welcome.inc");
?>

```

#37
#38
#39
#40

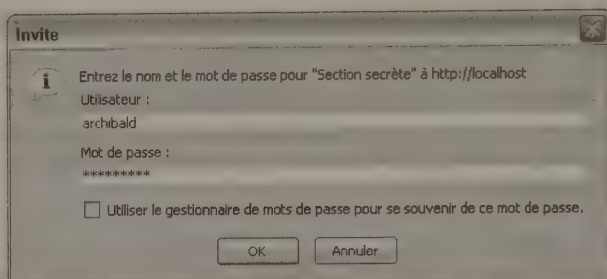
Certaines lignes du Listing 12.1 se terminent par des numéros de lignes. Les explications ci-dessous renvoient aux numéros de ligne du listing :

- #10** Débute un bloc `if` exécuté au démarrage du script, avant que l'utilisateur entre un nom d'utilisateur et un mot de passe. L'instruction `if` teste si un nom d'utilisateur a été entré, en regardant si l'élément `PHP_AUTH_USER` existe dans le tableau superglobal `$_SERVER`. S'il n'existe pas, l'utilisateur n'a pas entré un nom d'utilisateur, et le bloc `if` s'exécute. Le bloc `if` envoie deux en-têtes HTTP (lignes 12 et 13) indiquant au serveur Web que ce fichier est protégé par mot de passe. Le serveur Web envoie alors les en-têtes indiquant au navigateur de demander un nom d'utilisateur et un mot de passe.
- #14** Est exécutée uniquement si l'utilisateur clique sur le bouton Annuler.
- #15** Termine le bloc `if`.
- #18** Débute un bloc `else` qui s'exécute lorsque l'utilisateur a été invité à saisir un nom d'utilisateur et un mot de passe. Le bloc `else` s'exécute si l'élément `PHP_AUTH_USER` existe dans le tableau superglobal `$_SERVER`. Lorsque l'utilisateur clique sur le bouton pour soumettre le nom et le mot de passe, l'élément est créé. Même si l'utilisateur n'a rien tapé dans le champ de nom d'utilisateur, l'élément est créé. Il contient une chaîne vide, ce qui provoque l'exécution du bloc `else`.
- #20** Les lignes 20 à 31 créent et exécutent la requête SQL qui teste si le nom d'utilisateur et mot de passe existent dans la base MySQL de combinaisons nom d'utilisateur/mot de passe valides.

- #20 Inclut le fichier `Vars.inc` qui contient l'information nécessaire pour accéder à la base de données.
- #21 Lit le nom d'utilisateur dans le tableau superglobal `$_SERVER`. Rogne la valeur pour supprimer tout espace tapé par l'utilisateur. La ligne 22 fait de même pour le mot de passe saisi par l'utilisateur.
- #27 Les lignes 27 à 29 créent la requête SQL qui teste le nom d'utilisateur et le mot de passe. Notez que le mot de passe fourni par l'utilisateur est traité par la fonction `md5()`. C'est une fonction MySQL qui code le mot de passe, pour des raisons de sécurité. Le mot de passe stocké dans la base de données a lui aussi été crypté avec la fonction MySQL `md5`. Par conséquent, lorsque vous testez le mot de passe entré par l'utilisateur, vous devez lui appliquer `md5()` avant de le comparer au mot de passe dans la base de données.
- #32 Crée `$num`, qui contient le nombre d'enregistrements trouvés contenant la combinaison nom d'utilisateur et mot de passe saisie par l'utilisateur.
- #33 Débute un bloc `if` qui s'exécute si le nom d'utilisateur et le mot de passe saisis ne sont pas trouvés dans la base de données. Ce bloc `if` affiche un message, puis stoppe le script.
- #37 Termine le bloc `if` de nom d'utilisateur invalide.
- #38 Achève le bloc `else`.
- #39 Le code à partir de cette ligne s'exécute si le nom d'utilisateur et mot de passe sont valides. C'est ici qu'est inséré le contenu de la page Web. Dans l'exemple, il n'affiche qu'un simple message de bienvenue. Vous pouvez évidemment placer ce que vous voulez dans cette page Web, par exemple une liste de liens vers les pages Web de la zone protégée par mot de passe.
- #40 Inclut le fichier contenant la page Web. Ici, le fichier s'appelle `Welcome.inc`. Le Listing 12.2 présente le contenu de `Welcome.inc`.

Lorsque le script du Listing 12.1 est exécuté, il affiche une fenêtre invitant à saisir un nom et un mot de passe (lignes 10 à 15). L'aspect et la fonction exacts de la fenêtre dépendent du navigateur. La Figure 12.2 montre la boîte de dialogue affichée par Firefox.

Figure 12.2 :
La boîte de
dialogue
Firefox
réclamant un
nom
d'utilisateur
et un mot de
passe.



Une fois que l'utilisateur a saisi un nom d'utilisateur et un mot de passe corrects, la page Web est affichée. Dans cette application, une simple page d'accueil est affichée. La page utilise une variable PHP contenant le nom d'utilisateur. Le Listing 12.2 présente le code affichant la page d'accueil.

Listing 12.2 : Code pour afficher la page de bienvenue.

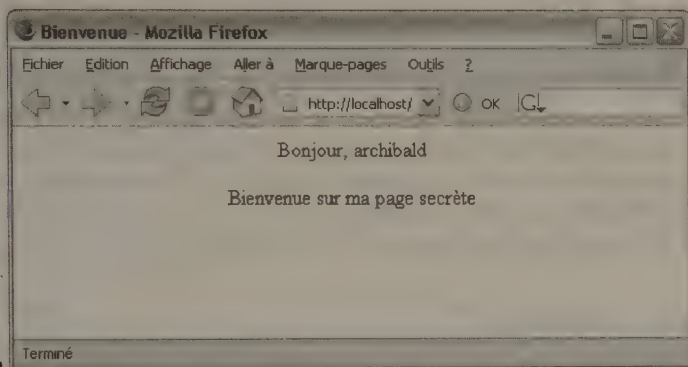
```
<?php
/* Fichier : Welcome.inc
 * Description : code HTML affichant la page Web de bienvenue.
 * Utilise une variable PHP, $user_name.
 */
echo "<html><head><title>Bienvenue</title></head>\n
<body>
<p align='center'>Bonjour, $user_name</p>\n
<p align='center'>Bienvenue sur ma page secrète</p>\n
</body></html>";
?>
```

Lorsque l'utilisateur entre un nom d'utilisateur et un mot de passe valides, la page Web présentée sur la Figure 12.3 est affichée.

Construction de l'application d'authentification en PHP : approche orientée objet

La programmation orientée objet consiste à créer et utiliser des objets pour mettre en place la fonctionnalité de l'application.

Figure 12.3 :
La page Web
affichée
lorsqu'un
utilisateur
entre un nom
d'utilisateur
et un mot de
passe
valides.



On commence par déterminer de quels objets l'application a besoin, puis on écrit les classes qui définissent ces objets, contenant les méthodes nécessaires à l'application. Une fois prêts les objets, on écrit le script qui crée et exploite les objets.

Développement des objets

Cette application d'authentification HTTP doit réclamer à l'utilisateur un nom et mot de passe. Une fois que l'utilisateur a soumis cette information d'identification, l'application doit la comparer avec les informations de comptes d'utilisateur stockées dans une base de données. Si le nom d'utilisateur et le mot de passe sont valides, l'application affiche le contenu de la page Web. Les objets suivants sont nécessaires à l'application :

- ✓ PasswordPrompter : Demande et recueille les informations d'identification de l'utilisateur.
- ✓ Database : Conteneur stockant les données de l'application.
- ✓ Account : Compte d'utilisateur, avec ses informations d'utilisateur associées.
- ✓ WebPage : Page Web à afficher dans le navigateur.

Les détails de chaque objet sont étudiés dans les sections suivantes.

Écriture de la classe PasswordPrompter

L'objet PasswordPrompter, qui affiche la fenêtre d'invite où l'utilisateur entre un nom d'utilisateur et un mot de passe, est central dans l'application. Pour afficher la fenêtre, l'objet PasswordPrompter exploite les fonctions d'authentification HTTP intégrées. Lorsque l'utilisateur soumet l'information, elle devient accessible au script PHP dans le tableau superglobal \$_SERVER.

Les propriétés

La classe PasswordPrompter ne nécessite qu'une propriété.

```
private $realm;
```

\$realm est une chaîne de votre choix, affichée lorsque l'application réclame le nom d'utilisateur et le mot de passe.

Le code

Le Listing 12.3 contient le code complet de la classe PasswordPrompter. Le constructeur et l'unique méthode sont étudiés en détail à la suite du listing. À noter, les numéros de lignes à la fin de certaines lignes de code. Les explications à la suite du listing renvoient à ces numéros de lignes.

Listing 12.3 : Code de la classe PasswordPrompter.

```
<?php
/* Classe : PasswordPrompter
 * Description : Classe affichant une fenêtre réclamant
 *              un mot de passe utilisateur. Le nom d'utilisateur
 *              et le mot de passe tapés dans la fenêtre sont retournés
 *              au script dans le tableau (array) superglobal $_SERVER.
 */
class PasswordPrompter
{
    private $realm; // Chaîne passée au constructeur

    function __construct($realm) #12
    {
        if(is_string($realm))
        {
```

```

        $this->realm = $realm;
    }
    else
    {
        throw new Exception("Cet argument doit être une chaîne.");
    }
}

function displayPrompt()
{
    header
    ("WWW-Authenticate: Basic realm=\"$this->realm\"");
    header('HTTP/1.1 401 Unauthorized'); #28
    // Le code suivant s'exécute lorsque l'utilisateur annule.
    exit("Cette page Web nécessite une authentification."); #30
}
}
?>

```

Le constructeur

Le constructeur attend qu'on lui passe une chaîne. La chaîne est stockée dans la variable *\$realm* (ligne 12). C'est le nom de la section de site protégée par mot de passe. Le nom peut être n'importe quelle chaîne. Il est affiché dans la boîte de dialogue qui invite l'utilisateur à saisir ses informations d'identification. Si l'information passée n'est pas une chaîne, une exception est jetée, accompagnée d'un message.

displayPrompt

Les lignes 26 à 28 envoient des en-têtes HTTP au serveur Web. Le serveur Web répond avec des en-têtes qui indiquent au navigateur d'afficher une fenêtre réclamant un nom et un mot de passe.

Le code de la ligne 30 ne s'exécute que si l'utilisateur clique sur le bouton Annuler.

Exploitation de la classe *PasswordPrompter*

Pour créer un objet *PasswordPrompter* :

```
$prompt = new PasswordPrompter("Section secrète");
```

Lorsque le paramètre passé n'est pas une chaîne, une exception est jetée, accompagnée d'un message.

Écriture de la classe Database

Pour authentifier les informations de login saisies par l'utilisateur, une liste de combinaisons nom d'utilisateur/mot de passe valides doit exister. La classe `Database` définit un objet, où les données seront stockées. Dans cette application, il s'agit d'une base de données MySQL. Le Listing 12.4 présente le code de la classe `Database`.

Les propriétés

Les propriétés de l'objet `Database` stockent les informations nécessaires pour se connecter à la base de données.

```
private $connection;
private $database_name;
private $host;
private $user;
private $password;
```

`$connection` contient un objet représentant la connexion à la base de données. `$database_name` contient le nom de la base de données sélectionnée. Lorsqu'aucune base de données n'a été sélectionnée, `$database_name` est `NULL`. Les propriétés `host`, `user` et `password` ne sont pas exploitées par la classe dans son état actuel. Cependant, dans une autre application, des méthodes supplémentaires pourraient réclamer l'emploi de ces propriétés.

Le code

Le Listing 12.4 présente le code complet de la classe `Database`. Le constructeur et les deux méthodes sont étudiés en détail à la suite de ce listing. À noter, les numéros de lignes à la fin de certaines lignes de code. Les explications à la suite du listing renvoient à ces numéros de lignes.

Listing 12.4 : Code de la classe Database.

```

<?php
/* Classe : Database
 * Description : Classe pour la connexion à une base de données MySQL.
 */
class Database
{
    private $cxn;           // objet de connexion à une base de données
    private $database_name;
    private $host;
    private $user;
    private $password;

    function __construct($filename)
    {
        include("$filename");
        if(! $this->cxn = new mysqli($host,$user,$passwd))           #16
        {
            throw new Exception("Base indisponible.
                                Réessayez ultérieurement.");
            email("dbadmin@mondomaine.com","Problème de base",
                "Le serveur MySQL ne répond pas. ").
                $this->cxn->error());
            exit();
        }
        $this->host = $host;                                         #25
        $this->user = $user;
        $this->password = $passwd;
    }

    function useDatabase($dbname)
    {
        if(! $result = $this->cxn->query("SHOW DATABASES"))         #32
        {
            throw new Exception("Base indisponible.
                                Réessayez ultérieurement.");
            email("dbadmin@mondomaine.com","Problème de base",
                "Le serveur MySQL ne répond pas. "
                . $this->cxn->error());
            exit();
        }
        else                                                         #41
        {
            while($row = $result->fetch_row())
            {
                $databases[] = $row[0];
            }
        }
    }
}

```



```

    }
    if(in_array($dbname,$databases) ||
        in_array(strtolower($dbname),$databases))           #49
    {
        $this->database_name = $dbname;
        $this->cxn->select_db($dbname);
        return TRUE;
    }
    else                                                       #55
    {
        throw new Exception("Base $dbname introuvable.");
        return FALSE;
    }
}

function getConnection()
{
    return $this->cxn;
}

function getDatabaseName()
{
    return $this->database_name;
}
}
?>

```

Le constructeur

Le constructeur crée la connexion et stocke les propriétés. L'hôte, l'utilisateur et le mot de passe doivent être passés au moment de la création du nouvel objet Database, pour que la connexion puisse être établie.

#16 Débute un bloc if qui s'exécute lorsque la connexion à la base de données échoue. Le bloc jette une exception avec affichage d'un message, et transmet un e-mail à l'administrateur de la base de données lui indiquant que la base de données n'est pas accessible, puis quitte le script.

#25 Les lignes 25 à 27 s'exécutent si la connexion à la base de données réussit. Ces lignes stockent les propriétés dans l'objet.

La connexion (`$cnn`) est stockée dans une propriété, de façon qu'elle puisse être utilisée à chaque fois que la connexion à la base est nécessaire.

useDatabase

La méthode `useDatabase` sélectionne une base de données. Le nom de la base de données est passé lors de l'exécution de la méthode. Avant de chercher à la sélectionner, la méthode vérifie que la base de données spécifiée existe.

Les explications suivantes renvoient aux numéros de lignes du Listing 12.4 :

- #32** Débute un bloc `if` qui s'exécute si la requête vers la base de données échoue. Le bloc jette une exception avec affichage d'un message, et transmet un e-mail à l'administrateur de la base de données lui indiquant que la base de données n'est pas accessible, puis quitte le script.
- #41** Débute un bloc `else` qui s'exécute lorsque la requête réussit. La requête retourne la liste des bases de données disponibles. Ce bloc stocke les noms de bases de données dans un tableau nommé `$databases`.
- #48** Débute un bloc `if` qui s'exécute lorsque la base de données à sélectionner existe bien. Le nom de la base de données est stocké dans une propriété. La base de données est sélectionnée. La méthode retourne `TRUE`.
- #55** Débute un bloc `else` qui s'exécute lorsque la base de données passée dans la méthode n'existe pas. Une exception est jetée, accompagnée d'un message indiquant que la base de données est introuvable. La méthode retourne `FALSE`.

getConnection

La méthode `getConnection` retourne la valeur stockée dans la propriété `$connection` (la connexion à la base de données). Elle est utilisable à chaque fois que la connexion à la base de données est nécessaire.

Emploi de la classe Database

Voici comment créer un objet Database :

```
$filename = "Vars.inc";
$db = new Database($filename);
$select = $db->useDatabase("UserAccount");
```

Lorsque l'instruction new ne peut créer un nouvel objet Database, elle jette une exception avec le message suivant :

Base indisponible. Réessayez ultérieurement.

Après ces instructions, \$select contient FALSE si la base de données n'existe pas. Si la sélection de la base de données réussit, \$select contient TRUE.

La base de données sélectionnée reste active jusqu'à en sélectionner une autre ou jusqu'à la fin du script. Vous pouvez changer la base de données sélectionnée à tout moment en utilisant de nouveau la méthode useDatabase.

Écriture de la classe Account

La classe Account stocke et récupère l'information concernant l'utilisateur. Dans cette application, le compte d'utilisateur contient uniquement trois éléments d'information : le nom d'utilisateur, le mot de passe et la date de création du compte. Dans d'autres applications, un compte d'utilisateur stocke des informations supplémentaires, comme l'adresse de l'utilisateur, son téléphone, son numéro de carte bancaire. Dans la base de données de cette application (décrite plus haut dans ce chapitre dans "Création d'une base de données d'utilisateurs"), les informations sur l'utilisateur se résument au nom dans user_name.

Les propriétés

Les propriétés de Account stockent l'information nécessaire pour accéder à la ligne du compte dans la base de données.

```
private $userID = NULL;
private $cxn;
private $table_name;
private $message;
```

\$message est une chaîne stockée par certaines méthodes lors de leur exécution.

Le code

Le Listing 12.5 présente le code de la classe `Account`. Le constructeur et les trois méthodes sont étudiés en détail après le listing. À noter, les numéros de lignes à la fin de certaines lignes de code. Les explications à la suite du listing renvoient à ces numéros de lignes.

Listing 12.5 : Le code de la classe Account.

```
<?php
/* Classe : Account
 * Description : Compte utilisateur stocké dans une base.
 * Représente les informations de comptes stockés
 * dans un enregistrement d'une table.
 */
class Account
{
    private $userID = NULL;
    private $cxn;           // objet de connexion à une base de données
    private $table_name;
    private $message;

    function __construct( mysqli $cxn,$table)
    {
        $this->cxn = $cxn;
        if(is_string($table))
        {
            $sql = "SHOW TABLES LIKE '$table'";
            $result = $this->cxn->query($sql);
            if($result->num_rows > 0)
            {
                $this->table_name = $table;
            }
            else
            {

```

```

        throw new Exception("$table introuvable
                               dans cette base");
        return FALSE;
    }
}
else #32
{
    throw new Exception("Deuxième paramètre :
                          nom de table invalide");
    return FALSE;
}
}

function selectAccount($userID)
{
    $userID = trim($userID); #42
    $sql = "SELECT user_name FROM $this->table_name
            WHERE user_name = '$userID'"; #44
    if(!$result = $this->cxn->query($sql))
    {
        throw new Exception("Exécution de la requête impossible : "
                              . $this->cxn->error());
        return FALSE;
    }
    if($result->num_rows < 1) #51
    {
        $this->message = "Compte $userID
                          inexistant !";
        return FALSE;
    }
    else #57
    {
        $this->userID = $userID;
        return TRUE;
    }
}

function comparePassword($form_password)
{
    if(!isset($this->userID)) #66
    {
        throw new Exception("Pas de compte actuellement sélectionné");
        exit();
    } #71
    $sql = "SELECT user_name FROM $this->table_name
            WHERE user_name = '$this->userID' AND
            password = md5('$form_password')";

```



```

if(!$result = $this->cxn->query($sql))                                #75
{
    throw new Exception("Exécution de la requête impossible : "
                        . $this->cxn->error());
    exit();
}
if($result->num_rows < 1 )                                           #81
{
    $this->message = "Mot de passe incorrect pour le
                    compte $this->userID!";
    return FALSE;
}
else                                                                  #87
    return TRUE;
}

function getMessage()
{
    return $this->message;
}
}
?>

```

Le constructeur

Le constructeur teste la connexion et le nom de la table qui lui sont passés, pour garantir que les paramètres sont dans le format correct, et les stocke dans des propriétés. Il n'existe pas de valeurs par défaut pour ces éléments ; les valeurs doivent être passées lors de la création de l'objet Account.

#16 Stocke, dans la propriété, la connexion à la base de données.

#17 Débute un bloc if qui s'exécute si le nom de la table passée est une chaîne. Le nom de la table est vérifié pour voir si elle existe.

#19 Les lignes 19 à 20 créent et exécutent une requête SQL qui teste s'il existe une table portant le nom passé au constructeur.

#21 Débute un bloc if qui s'exécute si la table existe. Le nom de la table est stocké dans la propriété \$table.

- #25 Débute un bloc `else` qui s'exécute lorsque la table n'existe pas. Le script jette une exception, avec message, et retourne `FALSE`.
- #32 Débute un bloc `else` qui s'exécute si le nom de la table n'est pas une chaîne. Le script jette une exception, avec message, et retourne `FALSE`.

selectAccount

Cette méthode stocke le nom d'utilisateur `user_name` du compte actif. Le nom d'utilisateur est passé lors de l'appel de `selectAccount`. La méthode vérifie si le nom d'utilisateur existe dans la base de données. S'il est trouvé, le nom d'utilisateur est stocké dans `$userID`.

- #42 Le nom d'utilisateur `userID` passé à `selectAccount` est nettoyé des éventuels espaces accidentellement tapés par l'utilisateur avant ou après son nom.
- #43 Les lignes 43 et 44 créent une requête SQL pour sélectionner le compte dont le `user_name` correspond au nom d'utilisateur passé dans `userID`. La propriété `table_name` (nom de la table) est utilisée dans la requête SQL.
- #45 Débute un bloc `if` qui s'exécute lorsque la requête échoue. Une exception est jetée, accompagnée d'un message, et la méthode retourne `FALSE`.
- #51 Débute un bloc `if` qui s'exécute lorsque le nombre de lignes retourné par la requête est inférieur à 1, signifiant que le nom d'utilisateur passé à la méthode n'a pas été trouvé dans la table de la base de données. Un message est stocké dans `$message` et `FALSE` est retournée.
- #57 Débute un bloc `else` qui s'exécute lorsque le nombre de lignes retourné par la requête n'est pas inférieur à 1, signifiant que le nom d'utilisateur a été trouvé dans la table de la base de données. La méthode stocke le nom d'utilisateur dans la propriété `$userID` et retourne `TRUE`.

comparePassword

Cette méthode compare le mot de passe qui lui est passé avec le mot de passe stocké dans le compte actif de la base de données. Cette méthode échoue si aucun compte n'a été sélectionné.

- #66** Débute un bloc `if` qui s'exécute lorsqu'aucun compte n'a été sélectionné. Si aucun nom d'utilisateur n'est stocké dans la propriété `$userID`, le script jette une exception, avec message, et se termine.
- #71** Les lignes 71 à 73 construisent une requête SQL qui sélectionne l'enregistrement contenant le nom d'utilisateur du compte actif et le mot de passe passé à la méthode.
- #74** Débute un bloc `if` qui s'exécute si la requête échoue. Le script jette une exception, avec message, et se termine.
- #80** Débute un bloc `if` qui s'exécute si le mot de passe fourni par l'utilisateur ne correspond pas au mot de passe stocké dans la base de données. C'est-à-dire lorsque la requête retourne moins d'une ligne. La méthode stocke un message et retourne `FALSE`.
- #86** Débute le bloc `else` qui s'exécute si le mot de passe fourni par l'utilisateur concorde avec le mot de passe stocké dans la base de données. Le bloc `else` retourne `TRUE`.

getMessage

Cette méthode retourne le contenu de la propriété `$message`.

Emploi de la classe Account

Pour utiliser la classe `Account`, vous devez lui passer une connexion à une base de données. De plus, la bonne base de données doit être sélectionnée. Par conséquent, pour créer un compte objet, vous pouvez employer des instructions de ce genre :

```
$db = new Database("Vars.inc");
$db->useDatabase("UserAccount");
$sacct = new Account($db->getConnection(), "Valid_User");
```

Lorsque le nouvel objet `Account` est créé, un objet "connexion à une base de données", créé auparavant, lui est passé, de même que le nom de la table où sont stockées les informations sur le compte.

Dans la plupart des cas, on sélectionne un compte spécifique de la base de données avec une instruction comme celle-ci :

```
$sacct->selectAccount("janet");
```

Si le nom d'utilisateur `janet` n'existe pas dans la table de la base de données, `FALSE` est retournée. Un compte doit être sélectionné avant de pouvoir employer `comparePassword`, comme ceci :

```
$acct->comparePassword("secret");
```

Si vous utilisez `comparePassword` avant de sélectionner un compte, le script jette une exception et se termine.

Écriture de la classe `WebPage`

La classe `WebPage` est fréquemment utilisée dans ce livre, à chaque fois qu'une page Web doit être affichée. La classe `WebPage` a pour seule mission l'affichage d'une page Web. La classe attend qu'on lui passe le nom du fichier qui définit la page Web. Si la page Web affiche des informations stockées dans des variables PHP, le tableau des données à afficher dans la page Web doit également être passé.

Les propriétés

Les propriétés de `WebPage` stockent les informations nécessaires pour afficher la page Web.

```
private $filename;  
private $data;
```

`$filename` est le nom du fichier qui définit la page Web – du code HTML plus éventuellement du code PHP pour les éléments de la page Web exploitant des variables PHP. Le fichier qui définit la page Web pour l'application d'authentification présentée dans ce chapitre s'appelle `Welcome.inc`. Ce même fichier a été utilisé dans le code procédural (voir le Listing 12.2).

`$data` est un tableau qui contient les variables PHP de la page Web. Si des informations contenues dans des variables PHP sont affichées sur la page, les variables PHP doivent être passées dans un tableau. Si aucune variable PHP n'est affichée, `$data` peut valoir `NULL`.

Le code

Le Listing 12.6 présente le code de la classe `WebPage`. Le constructeur et l'unique méthode, `displayPage`, sont étudiés en détail après le listing. À noter, les numéros de lignes à la fin de certaines lignes

de code. Les explications à la suite du listing renvoient à ces numéros de lignes.

Listing 12.6 : Le code de la classe WebPage.

```
<?php
/* Classe : WebPage
 * Description : Classe stockant les informations nécessaires
 *               pour afficher une page Web.
 */

class WebPage
{
    private $filename;
    private $data;

    function __construct($filename,$data=NULL)           #12
    {
        if(is_string($filename))                         #14
        {
            $this->filename = $filename;
        }
        else
        {
            throw new Exception("Le nom de fichier doit être une chaîne");
        }
        if($data == NULL or is_array($data))             #22
        {
            $this->data = $data;
        }
        else                                             #26
        {
            throw new Exception("Les données doivent être passées
                                dans un tableau");
        }
    }

    function displayPage()
    {
        #include("fields_summary.inc");
        @extract($this->data);                           #35
        include($this->filename);                         #36
    }
}
?>
```


Le constructeur

Lorsqu'un objet `WebPage` est instancié, le nom de fichier et les données passés au constructeur sont stockés dans les propriétés.

- #11 La définition du constructeur inclut une valeur par défaut pour `$data` : `NULL`. Lorsqu'aucune valeur n'est passée pour les données, `NULL` est stockée dans la propriété `$data`. Cela donne à l'objet la souplesse nécessaire pour stocker et afficher des pages Web contenant de l'HTML et/ou des variables PHP.
- #13 Débute une instruction `if/else` qui teste si le nom de fichier passé est une chaîne. Si c'est une chaîne, elle est stockée dans une propriété. Si ce n'est pas une chaîne, le bloc `else` est exécuté, ce qui jette une exception et termine le script.
- #21 Débute une instruction `if/else` qui teste si les données passées sont dans un tableau. L'instruction `if` s'exécute si les données valent `NULL` ou se présentent sous forme de tableau. Elle stocke les données passées au constructeur dans la propriété `$data`. Le bloc `else` qui commence ligne 26 s'exécute lorsque les données ne sont pas passées dans un tableau. Une nouvelle exception est jetée, avec message, et le programme se termine.

displayPage

Cette méthode affiche la page Web en se basant sur les informations stockées dans les propriétés.

- #34 Extrait les variables PHP du tableau `$data`. Lorsqu'aucune variable PHP n'est utilisée dans la page Web, aucune donnée n'est passée, et `$data` vaut `NULL`. Pour empêcher l'affichage d'une mention lorsque `$data` vaut `NULL`, un `@` est placé devant de la fonction `extract`.
- #35 Inclut le fichier qui définit la page Web, en se basant sur le nom de fichier stocké dans la propriété `$filename`.

Emploi de la classe WebPage

Un objet `WebPage` se crée ainsi :

```
$page1 = new WebPage("Welcome.inc");
$page2 = new WebPage("Welcome.inc",$data);
```

La première instruction sert pour créer un objet `WebPage` lorsque la page Web contient uniquement du code HTML, pas de variables PHP. La seconde instruction crée un objet qui contient les variables PHP à afficher dans la page Web.

Si le second paramètre est passé, ce doit être un tableau. Lorsque le second paramètre n'est pas un tableau (une chaîne ou un entier, par exemple), une exception est jetée, avec le message suivant :

Les données doivent être passées dans un tableau

Écriture du script *Auth-00*

Le script de l'application crée et utilise les objets mettant en place la fonctionnalité de l'application. Pour l'application d'authentification HTTP, le script doit demander à l'utilisateur d'entrer un nom et un mot de passe, puis vérifier que ce nom d'utilisateur et ce mot de passe sont valides. Le Listing 12.7 présente le script de l'application `Auth-00.php`.

Le flux du script est contrôlé par une instruction `if` qui teste si un nom d'utilisateur et un mot de passe ont été entrés, en regardant si le tableau `$_SERVER` contient le nom d'utilisateur. Voici le déroulement général du script :

```
si      (le nom d'utilisateur et le mot de passe n'ont pas été soumis)
        Demander à l'utilisateur d'entrer un nom d'utilisateur
        et un mot de passe

sinon   (le nom d'utilisateur et le mot de passe ont été soumis)
        1 Regarder si le nom d'utilisateur et le mot de
          passe correspondent à un nom et un mot de passe
          valides dans la base de données d'utilisateurs.
        2 Si le nom d'utilisateur et le mot de passe sont
          valides, affichez le contenu de la page
          Web protégée.
        3 Si le nom d'utilisateur et/ou le mot de passe ne
          sont pas valides, inviter l'utilisateur à
          s'identifier de nouveau.
```

Listing 12.7 : Script de l'application créant et utilisant les objets.

```

<?php
/* Programme : Auth-00.php
   Description :   Programme demandant un nom d'utilisateur
                  et un mot de passe. Le programme teste ensuite
                  si le nom d'utilisateur et le mot de passe
                  correspondent à une paire stockée dans une
                  base MySQL.

*/

require_once("PasswordPrompter.class");           #10
require_once("Database.class");
require_once("Account.class");
require_once("WebPage.class");

// Regarde si l'utilisateur a été invité à entrer son nom
if(!isset($_SERVER['PHP_AUTH_USER']))             #16
{
    try
    {
        $prompter = new PasswordPrompter("Section secrète");
        $prompter->displayPrompt();
    }
    catch(Exception $e)
    {
        echo $e->getMessage();
        exit();
    }
}

// Vérifie le nom d'utilisateur et mot de passe saisis
else                                             #31
{
    try                                         #33
    {
        $db = new Database("Vars.inc");       #35
        $db->useDatabase("UserAccount");      #36
    }
    catch(Exception $e)
    {
        echo $e->getMessage();
        exit();
    }                                         #42
    try                                         #43
    {
        $acct = new
            Account($db->getConnection(),"Valid_User"); #46
    }
}

```

```
if(!$acct->selectAccount($_SERVER['PHP_AUTH_USER']))  
{  
    $mess = $acct->getMessage();  
    echo $mess."<br>";  
    exit();  
}  
if(!$acct->comparePassword($_SERVER['PHP_AUTH_PW']) )  
{  
    $mess = $acct->getMessage();  
    echo $mess."<br>";  
    exit();  
}  
}  
catch(Exception $e)  
{  
    echo $e->getMessage();  
    exit();  
}  
}  
$data['user_name'] = $_SERVER['PHP_AUTH_USER'];  
try  
{  
    $welcome_page = new WebPage("Welcome.inc",$data);  
    $welcome_page->displayPage();  
}  
catch(Exception $e)  
{  
    echo $e->getMessage();  
    exit();  
}  
?  

```

Le programme ne contient qu'une seule instruction `if/else` pour demander et tester une paire nom d'utilisateur/mot de passe. Si l'utilisateur n'a pas fourni d'information d'identification, le script réclame un nom et un mot de passe. Lorsque l'utilisateur soumet ses informations d'identification, le nom d'utilisateur et mot de passe sont comparés aux comptes d'utilisateurs valides stockés dans la base de données MySQL. Si les informations sont valides, le contenu de la page Web est envoyé à l'utilisateur. (Les explications suivantes renvoient aux numéros de lignes du Listing 12.7.)

#10 Les lignes 10 à 13 insèrent les fichiers contenant les classes nécessaires à l'application. `require_once` est utilisé de façon

que la classe ne soit pas incluse accidentellement plusieurs fois.

- #16** Débute un bloc `if` qui s'exécute si l'utilisateur n'a pas soumis un nom d'utilisateur. Si `$_SERVER['PHP_AUTH_USER']` n'est pas défini, l'utilisateur n'a pas soumis un mot de passe : un objet `PasswordPrompter` est créé et affiché, avec pour résultat une fenêtre qui réclame un nom d'utilisateur et un mot de passe.
- #31** Débute un bloc `else` qui s'exécute lorsque l'utilisateur saisit des informations d'identification dans la fenêtre de mot de passe HTTP. Le nom d'utilisateur et mot de passe soumis sont accessibles au script dans le tableau superglobal `$_SERVER`, dans les éléments `PHP_AUTH_USER` et `PHP_AUTH_PW`.
- #35** Crée un objet `Database`.
- #36** Sélectionne la base de données qui contient les informations de compte d'utilisateur. Si `useDatabase` échoue (retourne `FALSE`) du fait que "UserAccount" n'existe pas, un message est affiché, et le script se termine.
- #45** Les lignes 45 et 46 créent un objet `Account`.
- #47** Débute un bloc `if` qui sélectionne le compte en se basant sur le nom soumis par l'utilisateur. Si `selectAccount` échoue (retourne `FALSE`) du fait que le nom d'utilisateur n'est pas trouvé dans la base de données, un message est affiché, et le script se termine.
- #52** Achève le bloc `if` qui sélectionne le compte.
- #53** Débute un bloc `if` qui compare le mot de passe soumis par l'utilisateur avec le mot de passe stocké dans la base de données. Si les mots de passe ne correspondent pas (la méthode retourne `FALSE`), un message est affiché, et le script s'achève.
- #58** Achève le bloc `if` qui compare les mots de passe.
- #65** Achève le bloc `else` qui compare les informations d'identification de l'utilisateur avec les informations d'identification valides dans la base de données. Le script ne va au-delà de cette ligne que si les informations d'identification soumises par l'utilisateur sont valides.

- #66 Crée un tableau de données à afficher dans la page Web. Le tableau contient un seul élément : `user_name`.
- #69 Crée un nouvel objet `WebPage` contenant la page Web de bienvenue. Le nom de fichier passé à l'objet `WebPage` est `Welcome.inc`. Le même fichier a été utilisé dans le script procédural (voir plus haut le Listing 12.2).
- #70 Affiche l'objet `WebPage` (la page Web de bienvenue). La page Web affichée est la même que celle affichée par le script procédural (voir la Figure 12.3).

Notez que de nombreuses lignes du script sont dans des blocs `try`. Il est souhaitable de placer dans des blocs `try` les méthodes susceptibles de jeter des exceptions. Lorsqu'une méthode d'objet jette une exception et que vous ne l'interceptez pas (`catch`), vous obtenez une erreur fatale de ce genre :

```
Fatal error: Uncaught exception 'Exception' with message  
'Base indisponible.' in c:\Database.class:18
```


Chapitre 13

Application d'identification d'utilisateur

Dans ce chapitre :

- Conception de la page Web d'identification.
- Création de la base de données pour stocker les informations d'utilisateur.
- Écriture du code procédural de l'application d'identification.
- Développement et emploi des objets pour programmer l'application d'identification.

De nombreux sites Web sont confidentiels ou présentent des sections confidentielles. Ces sites Web réclament que les utilisateurs s'identifient avant d'accéder aux informations confidentielles. Voici deux situations où un site Web limite l'accès :

- ✓ De nombreux commerces en ligne réclament aux clients de s'identifier, de façon que leurs informations puissent être stockées pour les futures transactions. Ces entreprises doivent protéger les informations des clients, particulièrement les données financières, d'un accès public.
- ✓ De nombreux sites Web n'accordent leur accès qu'à certaines personnes. Par exemple, des informations sur l'entreprise seront réservées au personnel de la société ou aux membres d'un certain service. Un autre exemple est lorsque l'information est commercialisée. Cette dernière ne doit donc être accessible qu'aux personnes qui ont payé pour cela.

Lorsque votre site Web a besoin de protection, implémentez une *application d'identification d'utilisateur*. Les applications d'identification d'utilisateur peuvent être simples, par exemple en se contentant d'une application où l'administrateur définit la liste des utilisateurs valides. Quiconque tente d'accéder à un fichier protégé est invité à entrer un nom d'utilisateur et un mot de passe qui sont vérifiés dans la liste des utilisateurs valides. Une application d'identification peut aussi être beaucoup plus compliquée. Elle peut permettre aux visiteurs du site à s'inscrire en configurant leur propre compte. L'application peut recueillir des informations auprès de clients lors de leur inscription. Elle peut proposer à l'utilisateur la capacité de gérer son compte. Les caractéristiques d'une application d'identification peuvent donc être étendues et variées.

L'application d'identification d'utilisateur est une des applications les plus courantes sur le Web, vous avez donc forcément eu l'occasion vous-même d'y être confronté. Ce chapitre montre comment construire votre propre application d'identification d'utilisateur.

Si vous n'avez besoin que d'une simple fenêtre de login, l'application présentée au Chapitre 12 sera éventuellement suffisante ; elle utilise les fonctions d'authentification HTTP intégrées aux navigateurs. L'application d'identification de ce chapitre est plus complexe. Elle permet à l'utilisateur de s'inscrire ou s'identifier lorsqu'il est déjà enregistré. Elle recueille et stocke des informations sur tout utilisateur qui s'inscrit. Elle apporte une page Web d'identification plutôt complexe, avec deux formulaires : un pour l'identification, l'autre pour l'inscription. Si vous devez fournir cette fonctionnalité supplémentaire et contrôler l'aspect de l'application d'identification, ce chapitre vous concerne.

Conception de l'application d'identification

La fonction élémentaire de l'application d'identification est de permettre aux utilisateurs enregistrés d'entrer dans le site Web, et de bloquer l'accès aux utilisateurs non inscrits. L'application permet aussi aux utilisateurs de s'inscrire, et stocke leurs informations dans une base de données. Pour remplir son rôle, l'application d'identification d'utilisateur doit opérer ceci :

- Donner aux clients la possibilité de s'inscrire pour avoir accès au site Web, de s'identifier sur le site lorsqu'ils sont déjà enregistrés.

- ✓ Afficher un formulaire d'inscription qui permet aux nouveaux clients de taper les informations d'inscription.

Les informations à recueillir dans le formulaire sont étudiés à la section suivante, "Création de la base de données d'utilisateurs".

- ✓ Valider l'information soumise dans le formulaire.

Il faut s'assurer que les champs obligatoires ne sont pas vides et que les informations soumises sont dans le format correct.

- ✓ Stocker les informations validées dans la base de données.

- ✓ Afficher un formulaire d'identification qui demande au client inscrit son nom et son mot de passe.

- ✓ Comparer le nom d'utilisateur et le mot de passe aux noms et mots de passe de la base de données. Si concordance il y a, envoyer une page Web au client. Si aucune correspondance n'est trouvée, donner au client la possibilité d'essayer de s'identifier de nouveau.

Création de la base de données d'utilisateurs

L'application a besoin d'une base de données pour stocker les informations d'utilisateur. La base de données est en fait le noyau de cette application. Une application d'identification doit stocker au minimum les noms d'utilisateur et mots de passe. Souvent, on stocke des informations supplémentaires.

Conception de la base de données

La première tâche est de décider quelles informations seront stockées. Au minimum, vous devez stocker un nom d'utilisateur et un mot de passe, que l'utilisateur devra fournir pour s'identifier. Il est également utile de savoir la date de création du compte d'utilisateur. Pour décider quelles informations recueillir lors de l'inscription de l'utilisateur, faites un compromis entre l'envie de collecter toutes les informations potentiellement utiles et l'envie de l'utilisateur de ne pas perdre du temps en formulaires, sans compter sa méfiance à fournir des informations personnelles. Le compromis peut être de laisser facultatives certaines informations ; certains utilisateurs les renseigneront, d'autres les laisseront vierges.

Il reste que certaines informations sont indispensables au fonctionnement du site Web. Par exemple, l'utilisateur comprend facilement qu'un site qui va lui transmettre quelque chose a besoin de son nom et de son adresse. En revanche, il ne verra peut-être pas pourquoi il lui serait indispensable de fournir son numéro de téléphone. D'ailleurs, lorsque l'on rend obligatoire le champ du numéro de téléphone, certains utilisateurs entrent des faux numéros. Par conséquent, à moins d'avoir un public obligé, tel que des employés, qui devront fournir tout ce que vous demandez, réfléchissez attentivement aux informations à recueillir. Un internaute agacé quitte facilement un site Web. Rien à voir avec le client qui a fait des dizaines de kilomètres pour arriver à une grande surface et tourne des heures pour trouver une place de parking. L'internaute peut repartir d'un clic.

Dans l'exemple d'application de ce chapitre, le site Web est une boutique en ligne qui vend des produits. Donc, vous devez recueillir du client ses informations de contact, et son numéro de téléphone, au cas où vous deviez le contacter concernant sa commande. La majorité des clients fournit sans souci son numéro de téléphone aux commerçants Web honorables, sachant que des problèmes avec une commande peuvent justifier que le vendeur les contacte. Le reste de cette section étudie les détails des informations et leur stockage dans une base de données MySQL.

La base de données ne contient qu'une table. Les informations client sont stockées dans la table, à raison d'un *enregistrement* (ligne) par client. Les champs de cette table sont présentés au Tableau 13.1. La table contient 12 champs. Les trois premiers champs, `user_name`, `password` et `create_date`, sont obligatoires et ne peuvent être vides. Les champs restants contiennent les nom, adresse, numéro de téléphone et de fax du client. Ils peuvent être vides. Le premier champ, `user_name`, est la clé primaire.

Création de la base de données

Il est possible de créer la base de données MySQL avec l'instruction SQL suivante :

```
CREATE DATABASE CustomerDirectory;
```

L'instruction SQL suivante crée la table :

Tableau 13.1 : Table de base de données : Customer.

Nom de variable	Type	Description
user_name	VARCHAR(20)	Nom d'utilisateur du compte (clé primaire)
create_date	DATE	Date à laquelle le compte a été ajouté à la table
password	VARCHAR(255)	Mot de passe du compte
email	VARCHAR(50)	Adresse électronique du client
last_name	VARCHAR(50)	Nom de famille du client
first_name	VARCHAR(40)	Prénom du client
street	VARCHAR(50)	Adresse du client
city	VARCHAR(50)	Ville du client
state	CHAR(2)	Code d'État (deux lettres)
zip	CHAR(10)	Code postal, cinq chiffres ou cinq chiffres + 4 (USA par exemple)
phone	CHAR(15)	Numéro de téléphone où joindre le client
fax	CHAR(15)	Numéro de fax du client

```

CREATE TABLE Customer (
  user_name    VARCHAR(20)    NOT NULL,
  create_date  DATE          NOT NULL,
  password     VARCHAR(255)   NOT NULL,
  last_name    VARCHAR(50),
  first_name   VARCHAR(40),
  street       VARCHAR(50),
  city         VARCHAR(50),
  state        CHAR(2),
  zip          CHAR(10),
  email        VARCHAR(50),
  phone        CHAR(15),
  fax          CHAR(15),
  PRIMARY KEY(user_name) );

```

Accès à la base de données

PHP fournit des fonctions MySQL pour accéder à la base de données à partir du script PHP. Les fonctions MySQL reçoivent les informations nécessaires pour accéder à la base de données, comme un nom de compte et un mot de passe MySQL. Cela n'a aucun rapport

avec d'autres nom de compte et mot de passe susceptibles de vous concerner, par exemple un mot de passe pour s'identifier sur l'ordinateur.



Plusieurs nouvelles caractéristiques sont présentes dans MySQL 4.1. Pour accéder aux nouvelles caractéristiques, vous devez utiliser les fonctions `mysqli` au lieu des fonctions `mysql`. Pour exploiter les fonctions `mysqli`, vous devez cependant utiliser PHP 5. Les fonctions `mysqli` ne sont pas disponibles dans PHP 4. Il reste possible d'exploiter les fonctions `mysql` et PHP 4 pour travailler avec MySQL 4.1 mais sans bénéficier des nouvelles caractéristiques. Les fonctions `mysqli` sont similaires aux fonctions `mysql`. Voir les informations de versions MySQL et PHP au Chapitre 10. Pour en savoir plus sur le module `mysqli` (MySQL Improved), allez sur www.php.net/manual/en/ref.mysqli.php.

Dans cette application, j'ai stocké les informations nécessaires aux fonctions `mysql` PHP dans un fichier séparé, `Vars.inc`. Pour des raisons de sécurité, ce fichier est stocké dans un répertoire en dehors de mon espace Web. Le fichier contient des informations semblables à celles-ci :

```
<?php
    $host = "localhost";
    $user = "admin";
    $passwd = "mdpadmin";
    $database = "CustomerDirectory";
?>
```



Notez les balises PHP au début (`<?php`) et à la fin (`?>`) du fichier. Si ces balises ne sont pas incluses, l'information s'affiche sur la page Web, aux yeux de tous, ce qui n'est pas souhaitable.

Ajout de données à la base de données

Cette base de données est prévue pour contenir des données entrées par les clients – pas par vous. À l'origine, la base sera donc vide. Cela dit, pendant les tests de votre application, le programme ajoutera une ou des rangées à la base de données.

Création de la page Web d'identification

Le client s'identifie sur votre site Web protégé via un formulaire HTML de page Web. On fait appel à deux formulaires : un pour

permettre aux nouveaux clients de s'inscrire, un autre pour que les clients inscrits puissent s'identifier.

Conception de la page Web d'identification

Il est probable que vous ayez déjà rencontré sur le Web des designs différents de pages d'identification. Vous avez donc peut-être déjà des idées pour votre propre page d'identification. Le design présenté ici n'est certes pas le seul possible, simplement un que j'apprécie. Ne vous gênez pas pour le modifier en tout ou partie.

Ici, les deux formulaires sont présentés sur une même page. Ils sont affichés côte à côte. Chaque formulaire a son propre titre, ses champs de formulaire, son bouton d'envoi. Le formulaire d'identification permet aux personnes d'entrer leur nom d'utilisateur et leur mot de passe ; le formulaire d'inscription demande beaucoup plus d'informations au client. La Figure 13.1 montre l'aspect de la page d'identification dans un navigateur.

Figure 13.1 shows a screenshot of a web browser displaying a client identification and registration page. The browser window is titled "Page d'identification du client" and is running Microsoft Internet Explorer. The address bar shows the URL "http://localhost/jchap13/ProcedureM/login.php". The page content is divided into two main sections side-by-side.

The left section is titled "Clients déjà inscrits : identifiez-vous ici". It contains two input fields: "Nom d'utilisateur" and "Mot de passe". Below these fields is a button labeled "Identification".

The right section is titled "Nouveaux clients : inscrivez-vous ici". It contains several input fields: "Nom d'utilisateur", "Mot de passe", "E-mail", "Nom", "Prénom", "Adresse", "Ville", "État" (with a dropdown menu currently showing "Alabama"), "Code postal", "Téléphone", and "Télécopie". Below these fields is a button labeled "Inscription".

At the bottom of the page, there is a small line of text: "Envoyez vos questions et commentaires à shah@goodminds.com". The browser's status bar at the very bottom shows "Terminé" and "Intranet local".

Figure 13.1 : La page d'identification affichée par l'application d'identification d'utilisateur.

Le code de la page Web d'identification est le même que si vous utilisiez la démarche procédurale ou orientée objet pour construire votre application. Ce code est stocké dans des fichiers séparés, inclus lorsque l'application a besoin d'afficher la page d'identification. Le code qui définit la page Web est donc séparé du code PHP qui fournit la logique de l'application.

Le code de la page d'identification se compose de deux fichiers : d'une part, le code qui définit l'aspect de la page, d'autre part le code qui fournit les informations spécifiques à la page.

Écriture du code de la page d'identification

La page d'identification affiche deux formulaires : un formulaire d'identification et un formulaire d'inscription, côte à côte. Le code qui crée la page est dans deux fichiers séparés :

- ✓ `double_form.inc` : Contient le code qui définit l'aspect de la page Web. Il produit une page Web avec deux formulaires côte à côte. Il est utilisable pour créer toute page Web nécessitant l'affichage de deux formulaires côte à côte. Ce fichier ne comprend pas d'informations spécifiques, tels que les noms et valeurs des champs texte affichés dans les formulaires. Pour créer la page Web, vous devez utiliser un autre fichier en conjonction avec celui-ci. L'autre fichier contient les informations spécifiques de la page Web, tels que les noms de champs.
- ✓ `fields_login.inc` : Contient les informations spécifiques de la page Web d'identification. Utilisé en conjonction avec `double_form.inc`, il affiche une page d'identification de client. Un autre fichier, avec des champs différents, est exploitable avec `double_form.inc` pour créer une page Web qui affiche des formulaires dont les champs sont sans rapport avec l'identification de clients.

Le reste de cette section présente les détails de ces fichiers. Le second fichier est court, et plus facile à comprendre, je l'étudie donc en premier (voir le Listing 13.1). Puis, pendant l'explication du premier fichier (`double_form.inc`), celui du Listing 13.2, je ferai référence aux informations contenues dans `fields_login.inc`.

Écriture du fichier `fields_login.inc`

Le fichier du Listing 13.1 définit sept tableaux contenant les informations spécifiques affichées dans la page Web d'identification.

- ✓ `$page` : Éléments affichés en haut et en bas de la page. Ces éléments englobent la totalité de la page, pas juste un des formulaires.
- ✓ `$elements_1` : Éléments affichés en haut et en bas du premier formulaire (celui de gauche). Ce tableau contient le texte à afficher en haut et en bas du formulaire, et le texte à afficher sur le bouton d'envoi.
- ✓ `$elements_2` : Éléments similaires, pour le second formulaire (celui de droite).
- ✓ `$fields_1` : Noms et étiquettes des champs à afficher dans le premier formulaire. Les clés du tableau sont les noms de champs du formulaire ; les valeurs du tableau sont les étiquettes affichées à côté des champs.
- ✓ `$length_1` : Longueurs des champs du premier formulaire. Il n'est pas indispensable de définir les longueurs des champs mais rien ne vous en empêche. Par exemple, vous pouvez attribuer la même longueur à tous les champs. Je préfère définir les longueurs de champs pour des raisons de sécurité ; cela restreint le nombre de caractères que l'utilisateur peut taper dans les champs, ce qui réduit les possibilités offertes au pirate pour entrer des données malveillantes dans les formulaires.
- ✓ `$fields_2` : Noms et étiquettes des champs du second formulaire. Les clés du tableau sont les noms de champs. Ces champs étant stockés dans la base de données, les clés du tableau sont les noms utilisés dans la table de la base de données. Les valeurs du tableau sont les étiquettes affichées à côté des champs. Par exemple, pour le nom de champ de base `first_name`, l'étiquette de formulaire est beaucoup plus claire si elle affiche *Prénom*.
- ✓ `$length_2` : Longueurs des champs du second formulaire. Les longueurs des champs sont celles des champs de la base de données.

La configuration des éléments et des champs dans ce fichier séparé, au lieu de les inclure dans le fichier avec le code HTML du formulaire, simplifie grandement la création et la maintenance du formulaire.

Listing 13.1 Fichier contenant les tableaux nécessaires à la page d'identification

```
<?php
/* Fichier : fields_login.inc
 * Description : Tableaux avec noms des champs et éléments de
 *              de formulaire pour la page Web d'identification.
 */
$page = array( "title" => "Page d'identification du client",
               "top"    => "",
               "bottom" => "Envoyez vos questions et commentaires
                           à admin@mondomaine.com",
               );

$elements_1 = array( "top" => "Clients déjà inscrits :
                           <span style='font-size: 80%;
                           font-weight: 100%;'>
                           <i>identifiez-vous ici</i></span>",
                     "bottom" => "",
                     "submit" => "Identification"
                     );

$elements_2 = array( "top" => "Nouveaux clients :
                           <span style='font-size: 80%;
                           font-weight: 100%;'>
                           <i>Inscrivez-vous ici</i></span>",
                     "bottom" => "",
                     "submit" => "Inscription"
                     );

$fields_1 = array( "fusername" => "Nom d'utilisateur",
                  "fpassword"  => "Mot de passe"
                  );

$length_1 = array( "fusername" => "10",
                  "fpassword"  => "10"
                  );

$fields_2 = array( "user_name"   => "Nom d'utilisateur",
                  "password"     => "Mot de passe",
                  "email"       => "E-mail",
                  "first_name"  => "Nom",
                  "last_name"   => "Prénom",
                  "street"      => "Adresse",
```

```

        "city"           => "Ville",
        "state"          => "État",
        "zip"            => "Code postal",
        "phone"          => "Téléphone",
        "fax"            => "Télécopie"
    );
$length_2 = array("user_name" => "20",
                  "password"   => "8",
                  "email"      => "55",
                  "first_name" => "40",
                  "last_name"  => "40",
                  "street"     => "55",
                  "city"       => "40",
                  "zip"        => "10",
                  "phone"      => "15",
                  "fax"        => "15"
                );
?>

```

Les tableaux sont définis dans un format structuré. Il serait possible d'utiliser beaucoup moins d'espace pour définir les tableaux mais ce format donne des valeurs claires, faciles à modifier si nécessaire.

Notez que certaines valeurs sont vides, comme `$element_1["bottom"]="`. Dans cette page Web en particulier, je ne voulais pas inclure du texte en haut de la page ou en bas des formulaires. Cependant, pour un autre formulaire exploitant `double_form.inc`, vous insérerez peut-être des valeurs pour ces éléments.

Écriture de `double_form.inc`

Le script `double_form.inc` (Listing 13.2) inc contient le code qui définit l'aspect de la page Web.

Il contient le code HTML des formulaires et des tableaux pour organiser la page. Le code inclut les variables pour les endroits où des informations spécifiques sont nécessaires. Les valeurs des variables sont fournies par le précédent fichier `fields_login.inc`. Par exemple, le script contient la ligne suivante, qui définit le titre de la page Web :

```
<head><title><?php echo $page['title']?></title></head>
```

La variable \$page['title'] se trouve dans le fichier fields_login.inc, où elle vaut "Page d'identification client".

Listing 13.2 : Script construisant les deux formulaires HTML côte à côte

```
<?php
/* Fichier : double_form.inc
 * Description : Code d'une page Web affichant
 *      deux formulaires HTML, côte à côte dans un tableau.
 */
include("functions.inc");                                #6
?>
<head><title><?php echo $page['title']?></title></head>
<body style="margin: 0">
<h1 align="center"><?php echo $page['top'] ?></h1>
<hr size="10" noshade>
<table border="0" cellpadding="5" cellspacing="0">
<?php
#####
# Formulaire 1 #
#####                                                    #16
?>
<tr>
<td width="33%" valign="top">
<p style="font-size: 110%; font-weight: bold">
<?php echo $elements_1['top']?></p>
<!-- Début du formulaire 1 (gauche) -->
<form action=<?php echo $_SERVER['PHP_SELF']?>
      method="POST">
<table border="0">
<?php                                                    #26
if (isset($GLOBALS['message_1']))                        #27
{
    echo "<tr>
        <td colspan='2'
            style=\"font-weight: bold;
                font-style: italic;
                font-size: 90%; color: red\">
                {$_GLOBALS['message_1']}<p></td></tr>\n";
    }
foreach($fields_1 as $field => $value)                    #36
{
    if(ereg("pass",$field))                               #38
        $type = "password";
    else
        $type = "text";
    echo "<tr><td style=\"text-align: right;
```

```

        font-weight: bold\">{$value}</td>
<td><input type='{$type}' name='{$field}'
        value='{$length_1[{$field}]}'
        size='{$length_1[{$field}]}'
        maxsize='{$length_1[{$field}]}'>
</td></tr>\n";
}
?>
#49

<tr>
<td colspan="2" style="text-align: center" >
<br />
<input type="submit" name="Button"
        value="<?php echo $elements_1['submit']?>">
</td></tr>
</table>
</form>
</td>

<!-- Colonne séparant les deux formulaires -->
<td style="background-color: gray"></td>
<?php
#####
# Formulaire 2 #
#####
?>
#63
#66

<td width="67%">
<p style="font-size: 110%; font-weight: bold">
<?php echo $elements_2['top']?>
<!-- Début du formulaire 2 (côté droit) -->
<form action="<?php echo $_SERVER['PHP_SELF']?>"
        method="POST">
<p>
<table border="0" width="100%">
<?php
if (isset($GLOBALS['message_2']))
{
    echo "<tr>
        <td colspan='2'
            style=\"font-weight: bold; font-style: italic;
            font-size: 90%; color: red\">
            {$GLOBALS['message_2']}<p></td></tr></td>";
}
foreach($fields_2 as $field => $value)
{
    if($field == "state")
    {
        echo "<tr><td style=\"text-align: right;
            font-weight: bold\">État</td>";

```



```

        <td><select name='state'>;
$stateName=getStateName(); #92
$stateCode=getStateCode();
for ($n=1;$n<=50;$n++) #94
{
    $state=$stateName[$n]; #96
    $scode=$stateCode[$n];
    echo "<option value='$scode'";
    if ($scode== @$_POST['state'])
        echo " selected";
    echo ">$state\n";
}
echo "</select>";
} #104
else
{
    if(ereg("pass",$field))
        $type = "password";
    else
        $type = "text"; #110
    echo "<tr><td style='text-align: right;
        font-weight: bold\">$value</td>
        <td><input type='$type' name='$field'
            value=\"\".@@$field.\"\"
            size='{length_2[$field]}'
            maxsize='{length_2[$field]}'>
        </td></tr>"; #117
} #118
} #119
?>

<tr><td colspan="2" style="text-align: center">
    <p style="margin-top: .05in">
        <input type="submit" name="Button"
            value="<?php echo $elements_2['submit']?>"
        </td></tr>
</table>
</form>
</td>
</tr>
</table>
<hr size="10" noshade>
<div style="text-align: center; font-size: 75%">
<?php echo $page['bottom']?>
</body></html>

```

Les numéros de lignes ci-dessous renvoient au Listing 13.2 :

- #6 Inclut un fichier contenant des fonctions utilisées dans ce script. Le Listing 13.3 présente le contenu du fichier `functions.inc`.
- #8 Les lignes 8 à 12 constituent une section HTML. Le code HTML définit le haut de la page Web. Notez que les lignes 8 et 10 comportent un petit passage en PHP qui affiche le contenu d'une variable. Les valeurs des variables se trouvent dans le fichier `fields_login.inc`. La ligne 12 commence la construction du tableau qui organise la page Web.
- #18 Les lignes 18 à 25 contiennent le code HTML qui ouvre la première cellule de tableau, y affiche le texte, puis génère la balise de formulaire. Un second tableau, inséré dans cette première cellule, est démarré. Il contiendra les champs de formulaire.
- #26 Débute la section PHP qui produit les champs de formulaire.
- #27 Débute un bloc `if` qui affiche un message. Si l'élément `"message_1"` existe dans le tableau `GLOBALS`, `message_1` est affiché. Le message est écrit dans le tableau global du script de l'application lorsque des erreurs sont détectées dans l'information saisie par l'utilisateur. Lorsque le formulaire est affiché pour la première fois, avant que l'utilisateur entre quoi que ce soit ou lorsqu'aucune erreur ne se produit, l'élément `GLOBALS message_1` n'existe pas.
- #36 Les lignes 36 à 49 contiennent un bloc `foreach` qui affiche tous les champs du formulaire. L'instruction `foreach` traverse le tableau `$fields_1` qui est défini dans le fichier `fields_login.inc`.
- #38 Débute une instruction `if` qui définit une valeur pour `$type`, variable utilisée dans le code du champ de saisie. Le champ de saisie est de type `text` pour tous les champs sauf le mot de passe, qui est de type `password`.
- #42 Les lignes 42 à 48 contiennent l'instruction `echo` qui affiche le code HTML du champ. L'instruction `echo` est exécutée une fois pour chaque élément du tableau `$fields_1`. Dans cette instruction, des variables PHP sont utilisées pour les informations spécifiques, telles que les noms de champs.

- #51** Débute une section HTML (lignes 51 à 59) pour afficher le bouton d'envoi du formulaire 1 et fermer les balises de ce même formulaire.
- #62** Code HTML affichant une colonne séparant les deux formulaires.
- #67** Section HTML (lignes 67 à 75) ouvrant la deuxième cellule du tableau, affiche le texte en haut de la cellule, et génère la balise de formulaire. Un deuxième tableau, dans cette seconde cellule, est démarré pour contenir les champs de formulaire.
- #76** Débute la section PHP qui produit les champs de formulaire.
- #77** Débute un bloc `if` qui affiche un message. Si l'élément `"message_2"` existe dans le tableau `GLOBALS`, `message_2` est affiché. Le message est écrit dans le tableau global du script de l'application lorsque des erreurs sont détectées dans l'information saisie par l'utilisateur. Lorsque le formulaire est affiché pour la première fois, avant que l'utilisateur entre quoi que ce soit ou lorsqu'aucune erreur ne se produit, l'élément `GLOBALS message_2` n'existe pas.
- #85** Les lignes 85 à 119 contiennent un bloc `foreach` qui affiche tous les champs du formulaire. L'instruction `foreach` traverse le tableau `$fields_2` qui est défini dans le fichier `fields_login.inc`.
- #87** Débute un bloc `if` qui s'exécute uniquement pour le champ `state`. Au lieu d'un simple champ, ici c'est la liste des États américains qui est affichée. Les lignes 96 à 97 exploitent les fonctions incluses à la ligne 6.
- #105** Débute un bloc `else` qui s'exécute pour tout champ sauf le champ `state`.
- #107** Débute une instruction `if` qui définit une valeur pour `$type`, variable utilisée dans le code du champ de saisie. Le champ de saisie est de type `text` pour tous les champs sauf le mot de passe, qui est de type `password`.
- #111** Les lignes 111 à 117 contiennent l'instruction `echo` qui produit le code HTML du champ. L'instruction `echo` est exécutée une fois pour chaque élément du tableau `$fields`. Dans cette

instruction, des variables PHP sont utilisées pour les informations spécifiques, telles que les noms de champs.

- #121** Débute une section HTML qui affiche le bouton d'envoi du formulaire 2, ferme les balises du formulaire 2, et affiche le texte en bas de la page Web.

Dans `double_form.inc`, le champ `state` du formulaire 2 est une liste déroulante. Le code qui crée la liste utilise deux fonctions stockées dans le fichier `functions.inc` inclus à la ligne 6. Le Listing 13.3 montre le code des deux fonctions.

Listing 13.3 : Fonctions créant des tableaux de noms et codes d'États américains.

```
<?php
function getStateCode()
{
    $stateCode = array(1=> "AL" ,
        "AK" ,
        "AZ" ,
        ...
        "WY" );
    return $stateCode;
}

function getStateName()
{
    $stateName = array(1=> "Alabama",
        "Alaska",
        "Arizona",
        ...
        "Wyoming" );
    return $stateName;
}
?>
```

Les fonctions sont appelées dans les lignes 92 et 93 du Listing 13.2. Les tableaux créés par ces fonctions sont utilisés pour créer la liste déroulante d'États dans l'instruction `for` qui débute en ligne 94 du Listing 13.2.

Affichage de la page Web d'identification

Le fichier `double_form.inc` est utilisé en conjonction avec `fields_login.inc` pour afficher la page Web d'identification. Même si la démarche pour afficher le formulaire peut être différente dans le script procédural et dans le script orienté objet, le code qui affiche le formulaire est le même :

```
include("fields_login.inc");  
include("double_form.inc");
```

Le fichier contenant les tableaux de données utilisés lors de l'affichage du formulaire doit être inclus auparavant, pour être disponible dès que nécessaire.

Par ailleurs, pour afficher du contenu dans les champs de formulaire, ce contenu doit être également disponible. Par exemple, l'utilisateur soumet un formulaire comportant une erreur et vous voulez réafficher le formulaire qu'il a rempli : il vous faut disposer des informations que l'utilisateur venait de taper. Pour cela, on utilise une variable dans le paramètre `value` de la balise `input`. Par exemple : `value="$first_name"`. Les variables que vous exploitez dans les balises `input` doivent aussi être disponibles.

Lorsque le client soumet le formulaire, l'information qu'il a tapée est passée au script, dans le tableau superglobal `$_POST`. Si vous réaffichez le formulaire, vous pouvez récupérer les informations dans ce tableau via la fonction PHP `extract` :

```
extract($_POST);
```

Ici, ce code affiche le formulaire :

```
extract($_POST);  
include("fields_login.inc");  
include("double_form.inc");
```

Création de l'application d'identification : démarche procédurale

L'application d'identification a un script principal organisé en trois sections de base :

- ✓ Une section qui s'exécute au premier affichage de la page d'identification, avant que l'utilisateur clique sur un bouton.
- ✓ Une deuxième section qui s'exécute lorsque l'utilisateur clique sur le bouton Identification.
- ✓ Une troisième section qui s'exécute lorsque l'utilisateur clique sur le bouton Inscription.

Une instruction switch contrôle le flux du programme, selon le bouton déclenché. Voici la structure générale du script :

```
switch (Button)
```

```
case "Identification":
```

- 1 Regarde si le nom d'utilisateur est dans la base de données. Si non, réafficher le formulaire, avec un message d'erreur.
- 2 Tester si le mot de passe est correct. Si non, réafficher le formulaire avec un message d'erreur.
- 3 Lorsque l'identification réussit, afficher la page Web protégée

```
case "Inscription":
```

- 1 Tester si tous les champs sont renseignés. Si non, réafficher le formulaire avec un message d'erreur.
- 2 Tester si l'information est dans le format correct
Si non, réafficher le formulaire avec un message d'erreur
- 3 Lorsque l'information est correcte, la stocker dans la base de données.
- 4 Lorsque l'inscription réussit, affichez la page Web protégée.

```
case "default":
```

- Affiche la page Web d'identification avec les champs vides.

Le cas par défaut (default) s'exécute lorsque ni le bouton Identification ni le bouton Inscription n'est déclenché.

Écriture du script de l'application

Le Listing 13.4 montre le script de l'application d'identification.

Listing 13.4 : Code de l'application d'identification.

```

<?php
/* Program: Login.php
 * Desc:      Script principal de l'application d'identification
 *            d'utilisateur. Elle fournit deux possibilités :
 *            (1) identification avec un nom d'utilisateur existant
 *            (2) inscription d'un nouvel utilisateur. Les noms
 *            d'utilisateurs et les mots de passe sont stockés
 *            dans une base MySQL.
 */

session_start();                                     #9
include("functions_main.inc");                       #10
$stable_name = "Customer";                          #11
$next_program = "SecretPage.php";

switch (@$_POST['Button'])                           #14
{
    case "Identification":                           #16
        $cxn = Connect_to_db("Vars.inc");
        $sql = "SELECT user_name FROM $stable_name
                WHERE user_name='$_POST[fusername]'"; #19
        $result = mysqli_query($cxn,$sql)
                or die("Exécution requête 1 impossible"); #21
        $num = mysqli_num_rows($result);
        if($num == 1)                                #23
        {
            $sql = "SELECT user_name FROM $stable_name
                    WHERE user_name='$_POST[fusername]'
                    AND password=md5('$_POST[fpassword]')";
            $result2 = mysqli_query($cxn,$sql)
                    or die("Exécution requête 2 impossible.");
            $row = mysqli_fetch_assoc($result2);        #30
            if($row)                                   #31
            {
                $_SESSION['auth']="yes";                #33
                $_SESSION['logname'] = $_POST['fusername']; #34
                header("Location: $next_program");      #35
            }
        }
        else                                          #37
        {
            $message_1="Le nom d'utilisateur '$_POST[fusername]'
                        existe mais vous n'avez pas entré le
                        bon mot de passe ! Réessayez.<br>";
            extract($_POST);
            include("fields_login.inc");
            include("double_form.inc");
        }
}

```

#45

```

}
elseif ($num == 0) // nom d'utilisateur introuvable #46
{
    $message_1 = "Le nom d'utilisateur entré n'existe #47
                pas ! Réessayez.<br>";
    include("fields_login.inc");
    include("double_form.inc");
}
break; #54
case "Inscription": #55
/* Check for blanks */
foreach($_POST as $field => $value) #57
{
    if ($field != "fax")
    {
        if ($value == "")
        {
            $blanks[] = $field;
        }
    }
} #66
if(isset($blanks)) #67
{
    $message_2 = "Les champs suivants sont vides.
                Merci d'entrer les informations demandées : ";
    foreach($blanks as $value)
    {
        $message_2 .="$value, ";
    }
    extract($_POST);
    include("fields_login.inc");
    include("double_form.inc");
    exit();
} #79
/* validate data */
foreach($_POST as $field => $value) #81
{
    if(!empty($value)) #83
    {
        if(ereg("name",$field) and
            !ereg("user",$field) and !ereg("log",$field))
        {
            if (!ereg("[A-Za-z' -]{1,50}$", $value))
            {
                $errors[] = "$value n'est pas un nom valide.";
            }
        }
    }
    if(ereg("street",$field)or ereg("addr",$field) or

```

```

        eregi("city",$field))
    {
        if(!ereg("^[A-Za-z0-9.,' -]{1,50}$",$value))
        {
            $errors[] = "$value n'est pas une adresse
                        ou ville valide.";
        }
    }
    if(ereg("state",$field))
    {
        if(!ereg("^[A-Za-z]",$value))
        {
            $errors[] = "$value n'est pas un état valide .";
        }
    }
    if(ereg("email",$field))
    {
        if(!ereg("^.+@.+\\.+.$",$value))
        {
            $errors[] = "$value n'est pas un e-mail
                        valide.";
        }
    }
    if(ereg("zip",$field))
    {
        if(!ereg("^[0-9]{5,5}(\\-[0-9]{4,4})?$",$value))
        {
            $errors[] = "$value n'est pas un code postal valide.";
        }
    }
    if(ereg("phone",$field) or eregi("fax",$field))
    {
        if(!ereg("^[0-9](xX -){7,20}$",$value))
        {
            $errors[] = "$value n'est pas un numéro
                        de téléphone valide. ";
        }
    }
    }
}
#132
foreach($_POST as $field => $value)
#134
{
    if($field != "Button")
    {
        if($field == "password")
        {
            $password = strip_tags(trim($value));
        }
    }
}

```

```

        else
        {
            $fields[]=$field;
            $value = strip_tags(trim($value));
            $values[] = addslashes($value);
            $$field = $value;
        }
    }
}
if(@is_array($errors)) #150
{
    $message_2 = "";
    foreach($errors as $value)
    {
        $message_2 .= $value." Merci de réessayer<br />";
    }
    include("fields_login.inc");
    include("double_form.inc");
    exit();
}
$user_name = $_POST['user_name']; #139

/* regarde si le n'om d'utilisateur existe */
$cxn = Connect_to_db("Vars.inc");
$sql = "SELECT user_name FROM $table_name
        WHERE user_name='$user_name'"; #158
$result = mysqli_query($cxn,$sql)
        or die("Exécution de la requête impossible.");
$num = mysqli_num_rows($result); #169
if ($num > 0) #170
{
    $message_2 = "$user_name existe déjà. Choisissez un
        autre nom d'utilisateur.";
    include("fields_login.inc");
    include("double_form.inc");
    exit();
}
else #178
{
    $today = date("Y-m-d"); #180
    $fields_str = implode("",$fields);
    $values_str = implode("'",",$values);
    $fields_str .=",create_date";
    $values_str .=":"'",$today";
    $fields_str .=",password";
    $values_str .=":"'",$md5("'.$password.'")";
    $sql = "INSERT INTO $table_name ";
    $sql .= "('$fields_str.'";

```



```

$sql .= " VALUES ";
$sql .= "(".$','. ' '. $values_str.")";
mysqli_query($cxn,$sql) or die(mysqli_error($cxn));           #165
$_SESSION['auth']="yes";                                       #192
$_SESSION['logname'] = $user_name;                             #193
/* envoie un e-mail au nouveau client */
$emess = "Votre inscription s'est correctement déroulée. ";
$emess .= " Vos nom d'utilisateur et mot de passe sont : ";
$emess .= "\n\n\t$user_name\n\t";
$emess .= "password\n\n";
$emess .= "Si vous avez des questions ou problèmes,";
$emess .= " écrivez à service@notreboutique.com";             #201
$subj = "Votre inscription client";                             #202
#$mailsend=mail("$email","$subj","$emess");                   #203
header("Location: $next_program");                             #204
}
break;                                                         #206

default:                                                       #208
    include("fields_login.inc");
    include("double_form.inc");
}
?>

```

Les numéros dans les explications suivantes renvoient aux numéros de lignes du Listing 13.4 :

- #9** Débute une session PHP.
- #10** Inclut le fichier contenant la fonction `Connect_to_db`, utilisée plus tard dans le script.
- #11** Déclare une variable qui contient le nom de la table de la base de données où est stockée l'information sur le client.
- #12** Déclare une variable contenant le nom du script constituant la page Web d'ouverture de votre site Web protégé. Le script s'exécute lorsque l'identification du client réussit.
- #14** Débute l'instruction `switch` qui englobe le reste du script. L'instruction `switch` teste la valeur de l'élément `Button` du tableau superglobal `$_POST`. L'élément `Button` n'existe que lorsque l'utilisateur a cliqué sur le bouton Envoyer d'un des formulaires.

- #16 Débute le cas (case) qui s'exécute lorsque l'élément Button a la valeur Identification – c'est-à-dire lorsque l'utilisateur a cliqué le bouton d'envoi intitulé Identification. Les instructions des lignes 14 à 52 font partie du cas Identification. Ce bloc case compare le nom d'utilisateur et le mot de passe avec les noms et mots de passe stockés dans la base de données.
- #17 Connecte à la base de données, en utilisant la fonction incluse ligne 10.
- #18 Les lignes 18 à 21 construisent puis exécutent une requête SQL pour sélectionner dans la base de données l'enregistrement comportant le nom d'utilisateur soumis par le client.
- #22 Vérifie combien d'enregistrements trouvés correspondent au nom d'utilisateur soumis par le client. Les valeurs possibles sont 0 ou 1.
- #23 Débute un bloc if qui s'exécute si 1 enregistrement a été trouvé. Ce bloc vérifie si l'utilisateur a entré le mot de passe correct.
- #25 Les lignes 25 à 30 construisent puis exécutent une requête SQL pour sélectionner l'enregistrement contenant le nom d'utilisateur et le mot de passe soumis par le client. Une fois la requête exécutée, regardent combien d'enregistrements ont été trouvés.
- #31 Débute un bloc if qui s'exécute lorsqu'un enregistrement a été trouvé, c'est-à-dire lorsque le mot de passe est correct. Deux variables de session sont définies et le contenu de la page Web protégée est affiché.
- #37 Débute un bloc else qui s'exécute lorsqu'aucun enregistrement n'a été trouvé, c'est-à-dire lorsque le mot de passe est erroné. Un message d'erreur est composé et la page Web d'identification est réaffichée avec le message d'erreur.
- #47 Débute un bloc elseif qui s'exécute lorsqu'aucun enregistrement n'a été trouvé avec le nom d'utilisateur soumis par l'utilisateur. Un message d'erreur est composé et la page Web d'identification est réaffichée avec le message d'erreur.
- #55 Débute le bloc case qui s'exécute lorsque Button a la valeur Inscription, ce qui veut dire que le client a cliqué sur le

bouton d'envoi Inscription. Les instructions des lignes 53 à 180 compose le bloc Inscription (inscription).

- #57** Débute une boucle `foreach` qui vérifie si chaque champ du formulaire est vide. Si le nom de champ n'est pas `fax` (qui n'est pas un champ obligatoire), le nom de champ de tout champ avec valeur vide est ajouté au tableau `$blanks`.
- #67** Débute un bloc `if` qui s'exécute lorsque `$blanks` est un tableau – c'est-à-dire lorsque des champs ont des valeurs vides. Un message d'erreur est composé qui contient les noms des champs vides. Le formulaire est réaffiché avec ce message d'erreur.
- #81** Débute une boucle `foreach` qui vérifie le format des données de chaque champ.
- #83** Débute un bloc `if` qui s'exécute lorsque la valeur n'est pas vide. Les lignes 57 à 79 du script traitent l'information des champs vides. Par conséquent, tout champ vide à ce stade est un champ qui peut être vide car ce n'est pas un champ obligatoire. Le test de format n'est pas nécessaire sur les champs vides.

Le bloc `if` (83 à 133) vérifie chacun des champs contenant des données, pour garantir que l'information fournie est dans un format acceptable. Un bloc `if` est exécuté lorsque des noms de champs spécifiques sont détectés, il compare la valeur de chaque champ avec une expression régulière spécifique au champ. Si le contenu du champ ne correspond pas à l'expression régulière, un message d'erreur approprié est stocké dans le tableau `errors`.

- #134** Débute une boucle `foreach` qui traite chaque champ, supprimant tous espaces initiaux ou terminaux et toutes balises HTML. Le mot de passe est stocké dans la variable `password`. Tous les autres champs sont stockés dans deux tableaux. `$fields` contient les noms de champs. Aucun traitement supplémentaire n'est nécessaire ici. `$values` contient les valeurs à insérer dans la base de données, avec guillemets ayant été échappés. De plus, chaque valeur est stockée dans une variable nommée d'après le nom de champ.



Si le paramètre `magic_quotes` est activé dans le fichier `php.ini`, inutile d'échapper les guillemets ici. Pour ce qui concerne les

données POST, les guillemets sont échappés automatiquement. Cependant, vous aurez plus de contrôle si vous désactivez le paramètre `magic_quotes` et échappez vous-même les guillemets dans le script. Quel que soit le moyen choisi, il est important que les guillemets soient échappés avant stockage des données dans la base de données. Pour une étude sur l'échappement des guillemets pour se protéger contre les attaques par injection de commandes SQL, voir le Chapitre 11.

- #151** Débute un bloc `if` qui s'exécute lorsque le tableau `$errors` existe, ce qui veut dire qu'au moins une erreur a été trouvée. Un message d'erreur est composé, et le formulaire est réaffiché avec le message d'erreur.
- #165** Les lignes 165 à 170 composent puis exécutent une requête pour sélectionner l'enregistrement contenant le nom d'utilisateur entré par l'utilisateur. Les doublons de noms d'utilisateur ne sont pas autorisés dans la base de données.
- #171** Débute un bloc `if` qui s'exécute lorsqu'un enregistrement est trouvé, signifiant que le nom d'utilisateur est déjà utilisé. Un message d'erreur est composé, et la page d'identification est réaffichée avec le message d'erreur.
- #179** Débute un bloc `else` qui s'exécute lorsqu'aucun enregistrement n'est trouvé, signifiant que le nom d'utilisateur n'est pas déjà exploité et qu'il est donc disponible.
- #181** Stocke la date du jour.
- #182** Les lignes 182 à 192 composent puis exécutent la requête SQL qui insère le nouvel enregistrement dans la base de données. Les tableaux `$fields` et `$values` sont convertis en chaînes avec la fonction `implode`. Les chaînes sont alors employées pour créer une requête `INSERT` pour ajouter l'enregistrement de l'utilisateur fraîchement inscrit.
- #193** Stocke une variable de session indiquant que l'utilisateur s'est identifié avec succès.
- #194** Stocke une variable de session avec le nouveau nom d'utilisateur.
- #196** Les lignes 196 à 204 composent et envoient un message électronique au nouvel utilisateur.

#206 Affiche le contenu de la page Web protégée.

#209 Débute le bloc `case` par défaut. Ce bloc exécute si aucun des deux précédents cas n'est vérifié. C'est-à-dire si l'utilisateur n'a cliqué ni sur le bouton Identification ni sur Inscription. Le bloc affiche la page Web d'identification avec champs de formulaire vierges.

Le script de l'application d'identification appelle une fonction pour se connecter à la base de données. Le fichier contenant le code de la fonction est inclus à la ligne 9. Le Listing 13.5 présente le code de la fonction.

Listing 13.5 : La fonction `Connect_to_db`

```
<?php
/* Fonction: Connect_to_db
 * Description : - Connecte à une base MySQL. Le nom d'un
 *               fichier contenant les variables de la base
 *               est passé à la fonction.
 */
function Connect_to_db($filename)
{
    include($filename);
    $connection = mysqli_connect($host, $user,$passwd)
        or die ("Connexion impossible au serveur.");
    $db = mysqli_select_db($connection,$database)
        or die ("Connexion impossible à la base de données.");
    return $connection;
}
?>
```

Cette fonction lit les informations dont elle a besoin dans un fichier séparé. Le nom de fichier est passé lors de l'appel de la fonction. Le contenu du fichier est semblable à celui-ci (il s'agira évidemment de vos propres informations) :

```
<?php
$host = "localhost";
$user = "admin";
$passwd = "";
$database = "CustomerDirectory";
?>
```


Pour des raisons de sécurité, stockez ce fichier en dehors de votre espace Web. Lorsque vous réglez le paramètre `include_path` dans votre fichier `php.ini` pour pointer vers un emplacement en dehors de votre espace Web, vous pouvez y stocker les fichiers, puis les inclure sans nécessité d'un chemin d'accès complet.

Protection de vos pages Web

Les pages Web de votre site ou section de site Web protégé ne sont pas différentes des autres pages Web. Vous désirez juste les mettre hors de portée des membres non identifiés. Pour cela, vous commencez une session et regardez si l'utilisateur est identifié, en haut de chaque page.

Lorsque l'utilisateur s'identifie via le script d'application PHP `Login.php` décrit dans la précédente section, une session est démarrée, et la valeur "yes" est stockée dans une variable de session :

```
$_SESSION['auth'] = "yes";
```



Vous emploierez probablement vos propres nom de variable et valeur, ainsi que votre propre nom de fichier. `Vars.inc` est plutôt clair, de même que `$auth = "yes"`. Il est préférable que le nom et la valeur soient moins faciles à deviner. Employez quelque chose hors du sujet, comme `Fleur.inc`, `$Brad="Pitt"`.

Donc, en haut de chaque script, vous vérifiez si la variable de session `auth` est définie pour l'utilisateur en question. Vous devez ajouter les instructions suivantes en haut de chaque script qui ne doit être disponible que pour les utilisateurs identifiés :

```
session_start();
if (@$_SESSION['auth'] != "yes")
{
    header("Location: Login.php");
    exit();
}
```

Lorsque `session_start` s'exécute, PHP cherche une session existante.

L'instruction `if` regarde si la variable de session `auth` est réglée sur `yes`. Si ce n'est pas le cas, ou si elle n'existe carrément pas, cela veut dire que l'utilisateur n'est pas identifié, auquel cas le script affiche la page Web d'identification et se termine. Sinon, si `auth` vaut `yes`,

l'utilisateur est identifié : le script continue pour afficher la page Web.

Création de l'application d'identification : approche orientée objet

La programmation orientée objet nécessite la création et l'emploi d'objets pour mettre en place les fonctionnalités. On commence par déterminer de quels objets l'application a besoin, puis on écrit les classes qui définissent ces objets, contenant les méthodes nécessaires à l'application. Une fois les objets prêts, on écrit le script qui crée et exploite les objets.

Développement des objets

L'application d'identification doit réaliser les tâches suivantes :

- ✓ Recueillir et traiter les informations d'un formulaire.
- ✓ Lorsqu'un client s'inscrit, stocker les informations dans une base de données.
- ✓ Vérifier un mot de passe.
- ✓ Autoriser l'utilisateur authentifié à visiter les pages du site Web pendant la durée de sa session d'utilisateur.
- ✓ Envoyer un e-mail à l'utilisateur.

La liste d'objets suivante reflète les tâches que l'application doit réaliser :

- ✓ **WebForm** : Un formulaire, central dans cette application. Le formulaire permet au client de s'inscrire ou de saisir son nom et mot de passe s'il est déjà inscrit. La classe `WebForm` fournit le formulaire de l'application. Il recueille et traite l'information tapée par l'utilisateur.
- ✓ **Database** : L'application stocke les informations sur le client dans une base de données. La classe `Database` fournit le conteneur qui stocke les données.

- ✓ **Account** : L'information entrée par le client doit être associée à un compte client. La classe `Account` stocke et récupère les informations sur le client.
- ✓ **Session** : En principe, le client doit pouvoir visionner plusieurs pages pendant sa visite sans s'identifier à chaque page. La durée pendant laquelle un client navigue sur votre site Web, du moment où il s'identifie, est appelée *session*. La classe `Session` permet au visiteur de rester identifié pendant toute une session.
- ✓ **Email** : L'application envoie un e-mail au client qui vient de s'inscrire, pour lui signaler que son enregistrement s'est déroulé avec succès. La classe `Email` stocke et gère le message électronique.

Les détails de chaque objet sont étudiés dans les sections suivantes.

Écriture de la classe `WebForm`

Un des objets les plus importants est le formulaire. La définition de l'objet formulaire est codée dans la classe `WebForm`. Cette classe est une classe générale utilisable pour afficher tout formulaire, et recueillir et traiter les données de tout formulaire, pas seulement le formulaire d'identification exploité dans cette application.

La classe `WebForm` affiche un formulaire sur une page Web, recueille l'information, la reformate, contrôle le format des données saisies, et réaffiche le formulaire lorsque des données incorrectes sont détectées. `WebForm` possède quatre propriétés, un constructeur et 13 méthodes.

Les propriétés

Les propriétés de `WebForm` stockent les informations nécessaires pour définir et gérer le formulaire et ses données. Voici les propriétés :

```
private $form;
private $fields;
private $data;
private $not_required;
```

Les deux premières propriétés sont les noms des fichiers nécessaires pour afficher le formulaire. Ces fichiers sont décrits dans la section "Écriture du code de la page d'identification", plus haut dans ce chapitre.

Les deux dernières propriétés sont des tableaux. `$data` contient les informations soumises par le client dans le formulaire. La clé de chaque élément est un nom de champ – le nom de champ utilisé dans le formulaire et la table de la base de données. `$not_required` est un tableau contenant les noms de champs des champs autorisés à être vides. La méthode `checkForBlanks` utilise ce tableau pour identifier les champs qui ne doivent pas générer d'erreur lorsqu'ils ne sont pas renseignés.

Le code

Le listing 13.6 présente le code de la classe `WebForm`. Chaque méthode est étudiée en détail après le listing. Remarquez les numéros de lignes à la fin de certaines lignes de code. Les explications à la suite du listing renvoient à ces numéros de lignes.

Listing 13.6 : La classe WebForm

```
<?php
/* Classe : WebForm
 * Description : Classe qui collecte, stocke et traite
 * les informations d'un formulaire HTML.
 */

class WebForm
{
    private $form;           //nom de fichier
    private $fields;         //nom de fichier
    private $data;           //tableau (array)
    private $not_required;   //tableau (array)

    function __construct($form,$fields,$data=NULL)
    {
        if(is_string($form) and is_string($fields))
        {
            $this->form = $form;
            $this->fields = $fields;
        }
        else

```

```

    {
        throw new Exception("Les 2 premiers paramètres doivent
                               être des noms de fichiers");
    }
    if($data == NULL OR is_array($data))                #26
    {
        $this->data = $data;
    }
    else                                                #30
    {
        throw new Exception("Les données du formulaire doivent
                               être passées dans un tableau");
    }
}

function setFieldsNotRequired($not_required)            #37
{
    if(!is_array($not_required))
    {
        throw new Exception("Les champs doivent être passés
                               dans un tableau");
    }
    else
    {
        $this->not_required = $not_required;
    }
}

function displayForm()
{
    @extract($this->data);
    include($this->fields);
    include($this->form);
}

function getAllFields()
{
    return $this->data;
}

function checkForBlanks()
{
    if(sizeof($this->data) < 1 )                        #64
        throw new Exception("Pas de données de formulaire disponibles");

    foreach($this->data as $key => $value)              #67
    {
        if($value == "")                              #69

```



```

    {
        $match = false;
        if(is_array($this->not_required)) #72
        {
            foreach($this->not_required as $field) #74
            {
                if($field == $key)
                {
                    $match = true;
                }
            }
        }
        if(!$match) #82
        {
            $blanks[] = $key;
        }
    }
    if(isset($blanks)) #88
        return $blanks;
    else
        return TRUE;
}

function verifyData()
{
    if(sizeof($this->data) < 1 ) #96
        throw new Exception("Pas de données de formulaire disponibles");
    foreach($this->data as $key => $value) #98
    {
        if(!empty($value)) #100
        {
            if(ereg("name",$key) and !ereg("log",$key)
               and !ereg("user",$key)) #103
            {
                $result = $this->checkName($value); #105
                if(is_string($result))
                    $errors[$key] = $result; #107
            } #108
            if(ereg("addr",$key) or ereg("street",$key)
               or ereg("city",$key)) #110
            {
                $result = $this->checkAddress($value);
                if(is_string($result))
                    $errors[$key] = $result;
            }
            if(ereg("email",$key)) #116
            {

```

```

        $result = $this->checkEmail($value);
        if(is_string($result))
            $errors[$key] = $result;
    }
    if(ereg("phone",$key)or ereg("fax",$key))                #122
    {
        $result = $this->checkPhone($value);
        if(is_string($result))
            $errors[$key] = $result;
    }
    if(ereg("zip",$key))                                       #128
    {
        $result = $this->checkZip($value);
        if(is_string($result))
            $errors[$key] = $result;
    }
    if(ereg("state",$key))                                     #134
    {
        $result = $this->checkState($value);
        if(is_string($result))
            $errors[$key] = $result;
    }
}
}
if(isset($errors))                                           #142
    return $errors;
else
    return TRUE;
}

function trimData()
{
    foreach($this->data as $key => $value)
    {
        $data[$key] = trim($value);
    }
    $this->data = $data;
}

function stripTagsFromData()
{
    foreach($this->data as $key => $value)
    {
        $data[$key] = strip_tags($value);
    }
    $this->data = $data;
}

```

```

function checkName($field)
{
    if(!ereg("^[A-Za-z' -]{1,50}$", $field))
    {
        return "$field n'est pas un nom valide.
                Réessayez.";
    }
    else
        return TRUE;
}

function checkAddress($field)
{
    if(!ereg("^[A-Za-z0-9.,' -]{1,50}$", $field))
    {
        return "$field n'est pas une adresse valide.
                Réessayez.";
    }
    else
        return TRUE;
}

function checkZip($field)
{
    if(!ereg("^[0-9]{5}(\-[0-9]{4})?", $field))
        return "$field n'est pas un code postal valide.
                Réessayez.";
    else
        return TRUE;
}

function checkPhone($field)
{
    if(!ereg("^[0-9](Xx -){7,20}$", $field))
        return "$field n'est pas un numéro de téléphone valide.
                Réessayez.";
    else
        return TRUE;
}

function checkEmail($field)
{
    if(!ereg("^.+@.+\\..+$", $field))
        return "$field n'est pas une adresse électronique valide.
                Réessayez.";
    else
        return TRUE;
}

```

```

function checkState($field)
{
    if(!ereg("[A-Za-z]", $field))
        return "$field n'est pas un état valide.
            Réessayez.";
    else
        return TRUE;
}
}
?>

```

Le constructeur

Le *constructeur* vérifie si les informations passées à la classe sont du bon type de données. Si c'est le cas, il stocke les données. Sinon, il jette une exception, avec le message approprié. À la ligne 16, le constructeur vérifie si les deux premiers paramètres, qui doivent être des noms de fichiers, sont des chaînes. Ligne 26, il vérifie si les données passées dans le troisième paramètre sont bien dans un tableau. Il peut aussi s'agir de NULL, ce qui permet le tout premier affichage du formulaire, pour que le client puisse y entrer des informations. On crée donc un objet *WebForm* avec une de ces instructions :

```

$form = new WebForm("fichier1.inc", "fichier2.inc", $_POST);
$form = new WebForm("fichier1.inc", "fichier2.inc");

```

Si vous utilisez la seconde forme, la propriété *\$data* vaut NULL. Cela ne provoque pas d'erreur.

setFieldsNotRequired

La méthode *setFieldsNotRequired* contrôle si le paramètre passé est un tableau. Si c'est le cas, elle le stocke dans une propriété. Sinon, elle jette une exception avec le message approprié.

displayForm

La méthode *displayForm* affiche le formulaire. Elle extrait les données de la propriété *\$data*. Un @ est utilisé pour supprimer les

messages d'erreur, pour que le formulaire puisse être affiché sans aucune donnée. Les deux fichiers qui définissent le formulaire sont alors inclus. Ces deux fichiers peuvent définir tout type de formulaire, avec les champs et éléments à employer. Dans cette application, j'utilise les fichiers décrits plus haut, qui définissent une page Web d'identification comportant deux formulaires.

getAllFields

La méthode `getAllFields` retourne le tableau de propriétés `$data` qui contient les données du formulaire.

checkForBlanks

La méthode `checkForBlanks` vérifie chaque champ du formulaire pour voir s'il contient des informations. Si des champs vides invalides sont trouvés, elle retourne un tableau contenant les noms de champs de ces champs vides. L'explication suivante fait référence aux numéros de lignes du Listing 13.6 :

- #64 Regarde si `$data` contient un tableau de données. Si ce n'est pas le cas, jette une exception indiquant que le formulaire ne contient aucune donnée.
- #67 Débute une boucle `foreach` qui traverse le tableau `$data`. La boucle `foreach` regarde s'il existe des champs vides. Un tableau est construit pour contenir les noms des champs vides.
- #69 Débute un bloc `if` qui s'exécute si le champ est vide. Le bloc se termine à la ligne 86.
- #71 Règle la variable `$match` sur `false`.
- #72 Débute un bloc `if` qui s'exécute si des noms de champs sont stockés dans `$not_required`. Si c'est le cas, le champ courant est comparé à chaque valeur dans `$not_required`, et si une occurrence est trouvée, `$match` est réglée sur `true`. Une fois que le champ courant a été comparé avec les noms de champs dans `$not_required`, la ligne 82 vérifie si une occurrence a été trouvée. Si `$match = false`, le nom du champ courant n'est pas dans le tableau des champs pouvant être vides, ce qui veut dire que le champ n'est pas autorisé à être

vide : le nom de champ est ajouté à un tableau nommé `$blanks`.

- #88** Après traitement de tous les champs vides, une instruction `if` vérifie si des noms de champs ont été ajoutés au tableau `$blanks`. Si c'est le cas, le tableau est retourné. Sinon, si le tableau `$blanks` est vide, la méthode retourne `TRUE`, ce qui veut dire que tous les champs sont corrects.

verifyData

La méthode `verifyData` vérifie chaque champ pour garantir que l'information soumise est dans un format raisonnable. Par exemple, vous savez que "salut" n'est pas un format raisonnable pour un code postal. Un code postal doit être dans un de ces deux formats : soit 12345 ou 12345-6789 (USA par exemple). Lorsque vous contrôlez que l'information soumise par le client est dans un format raisonnable, vous pouvez aussi détecter des fautes. Il est aussi possible d'empêcher les pirates de saisir du code malveillant dans vos champs de formulaire.

La vérification de données est affaire de compromis. Il faut intercepter autant d'informations erronées que possible. En revanche, il ne faut pas empêcher le passage des informations légitimes. Par exemple, si vous limitez le numéro de téléphone à des chiffres, tout numéro de téléphone avec tiret ou parenthèses – comme dans (0) 155567890 – sera considéré comme une erreur. Pensez soigneusement, pour chaque champ, au type d'information à autoriser ou non.

Cette méthode compare le contenu de champs spécifiques avec des expressions régulières correspondant aux informations autorisées dans ces champs. L'explication suivante fait référence aux numéros de lignes du Listing 13.6 :

- #96** Vérifie si `$data` contient des données. Si ce n'est pas le cas, la méthode jette une exception avec un message approprié.
- #98** Débute une boucle `foreach` qui traverse le tableau dans la propriété `$data`. La boucle `foreach` vérifie le format de tous les champs de données. Toute erreur de format détectée est ajoutée au tableau `$errors`.
- #100** Débute un bloc `if` qui s'exécute lorsque le champ n'est pas vide. Les champs vides ne sont pas vérifiés car ils seraient systématiquement incorrects. Cependant, certains champs



sont autorisés à être vides. La méthode `checkForBlanks` est habituellement utilisée avant cette méthode pour intercepter tout champ vide alors qu'il n'est pas autorisé à l'être. Dans ce bloc `if`, le contenu de chaque champ est comparé avec une expression régulière appropriée.

- #102** Débute un bloc `if` qui s'exécute si le nom de champ contient `"name"`, comme dans `first_name`, contient ni `"log"` ni `"user"`. Les règles pour les noms sont plus restrictives que pour les noms d'utilisateur, aussi cette instruction exclut-elle les noms de champs qui sont probablement des noms d'utilisateur. À la ligne 105, le contenu du champ est passé à la méthode `checkName`, qui compare la valeur à l'expression régulière pour les noms. Lorsque `checkName` approuve la valeur, elle retourne `TRUE`. Si la valeur ne correspond pas à l'expression régulière, `checkName` retourne un message d'information. La méthode `checkName` est étudiée plus loin dans cette section.

La ligne 106 vérifie la valeur retournée par `checkName`. Si c'est une chaîne, ce qui veut dire que la valeur n'a pas été validée, la chaîne est ajoutée au tableau `$errors`.

- #109** Débute un bloc `if` qui s'exécute si le nom de champ contient `"addr"` (adresse) ou `"street"` (rue) ou `"city"` (ville). Traite le contenu du champ comme décrit pour la ligne 102. La méthode `checkAddress` est appelée.
- #116** Débute un bloc `if` qui s'exécute si le nom de champ contient `"email"`. La méthode `checkEmail` est appelée.
- #122** Débute un bloc `if` qui s'exécute si le nom de champ contient `"phone"` ou `"fax"`. La méthode `checkPhone` est appelée.
- #128** Débute un bloc `if` qui s'exécute si le nom de champ contient `"zip"` (code postal). La méthode `checkZip` est appelée.
- #134** Débute un bloc `if` qui s'exécute si le nom de champ contient `"state"` (état). La méthode `checkState` est appelée. Bien que le champ `state` soit une liste déroulante, il faut vérifier son format. Un pirate peut changer le champ en champ de texte et y insérer du code malveillant.
- #142** Une fois traitées les valeurs de champ, la méthode vérifie si des chaînes ont été ajoutées au tableau `$errors`. Si c'est le cas, le tableau est retourné. Si ce n'est pas le cas, ce qui veut

dire que le contenu de tous les champs est correct, la méthode retourne TRUE.

Cette classe `WebForm` vérifie les champs qui sont couramment recueillis dans les formulaires, telles que adresses et numéros de téléphone. Pour utiliser cette classe pour vérifier les données de formulaire comportant des champs moins courants, comme l'âge, l'anniversaire, les loisirs préférés, vous ajouterez des méthodes à la classe pour vérifier ces informations spécifiques.

Ou, si vous pensez employer fréquemment différents formulaires, il est possible de créer des classes pour représenter chaque formulaire utilisé. Vous aurez donc une classe `WebForm` contenant les méthodes de la classe `WebForm` de cet exemple, hormis les méthodes vérifiant les données. Puis, vous aurez deux classes (ou plus), comme `WebFormAddress` et `WebFormPersonal`, ayant `WebForm` comme parent et contenant uniquement les méthodes pour vérifier les données.

trimData, stripTagsFromData

Une fonction PHP est appliquée à chaque valeur de la propriété `$data`. Les valeurs résultantes sont stockées dans `$data`. La fonction `trim` supprime les espaces initiaux et terminaux d'une chaîne. La fonction `strip_tags` supprime les balises HTML de la chaîne, ce qui est important pour la sécurité.

checkName, checkAddress, checkZip, checkPhone, checkEmail, checkState

Ces méthodes comparent les valeurs qui leur sont envoyées, avec des expressions régulières correspondant aux types de données attendus. Par exemple, `checkZip` vérifie si la chaîne passée se compose de cinq chiffres éventuellement suivis d'un tiret et de quatre chiffres.

Si la comparaison avec l'expression régulière échoue, un message est retourné, indiquant que le champ est invalide. Si la chaîne est valide, la méthode retourne TRUE.

Écriture de la classe Database

La classe Database fournit la connexion à la base de données où sont stockées les informations sur le client. La classe Database a été développée au Chapitre 12 (pour son code, voir le Listing 12.4).

Voici les méthodes fournies par la classe Database :

- ✓ **Le constructeur** : Crée une connexion à une base de données MySQL. Attend qu'on lui passe un nom de fichier, contenant le nom d'hôte, le nom de compte et le mot de passe nécessaires pour accéder à MySQL. Un objet Database est créé avec l'instruction suivante :

```
$db = new Database("Vars.inc");
```

- ✓ **useDatabase** : Sélectionne une base de données et stocke le nom de la base de données. S'attend à ce qu'on lui passe un nom de base de données. Vérifie si la base de données existe, retourne un message si elle n'existe pas. La méthode est utilisée ainsi :

```
$db->useDatabase("Customer");
```

- ✓ **getConnection** : Retourne la connexion établie et stockée dans le constructeur. La méthode est utilisée ainsi :

```
$db->getConnection();
```

Écriture de la classe Account

La classe Account indique quel enregistrement de la table contient les informations sur le compte client. La classe Account utilisée dans cette application a été développée au Chapitre 12 (voir le Listing 12.5). Je lui est cependant ajouté une méthode pour ce chapitre. Il s'agit de la méthode `createNewAccount`. Le Listing 13.7 présente le code de cette nouvelle méthode.

Résumé des méthodes de la classe Account

- ✓ **Le constructeur** : crée une connexion à la table MySQL contenant les informations de compte. Attend qu'on lui passe une connexion à une base de données et un nom de table. La méthode vérifie si les paramètres passés sont au bon

format. Elle vérifie également si la table de la base de données existe.

Un objet Account est créé avec l'instruction suivante :

```
$acct = new Account($connection,"nomdetable");
```

Un objet Account est souvent créé à fin d'utilisation avec un objet Database. Il est possible de le créer avec les instructions suivantes :

```
$db = new Database("Vars.inc") ;  
$acct = new Account($db->getConnection,"nomdetable") ;
```

- ✓ **selectAccount** : cette méthode stocke le nom d'utilisateur **user_name** du compte actif. Le nom d'utilisateur est passé lors de l'appel de **selectAccount**. La méthode vérifie si le nom d'utilisateur existe dans la base de données. Elle est utilisée avec l'instruction suivante :

```
$acct->getConnection();
```

- ✓ **getMessage** : cette méthode retourne le contenu de la propriété **\$message**. Elle est utilisée avec l'instruction suivante :

```
$acct->getMessage();
```

- ✓ **comparePassword** : cette méthode compare le mot de passe qui lui est passé avec le mot de passe stocké dans le compte actif de la base de données. Cette méthode échoue si aucun compte n'a été sélectionné. Utilisez **selectAccount** pour sélectionner un compte actif avant d'utiliser cette méthode. Elle est utilisée avec l'instruction suivante :

```
$acct->comparePassword("secret");
```

- ✓ **createNewAccount** : cette méthode stocke les informations sur le client dans un nouvel enregistrement de la table, ce qui crée un nouveau compte. L'information est passée à cette méthode dans un tableau. Cette méthode considère que les données ont déjà été nettoyées et vérifiées, qu'elles sont prêtes à être stockées dans la base de données. Elle est utilisée avec l'instruction suivante :

```
$acct->createNewAccount($data_array);
```


createNewAccount

Cette méthode stocke les informations sur le client dans un nouvel enregistrement de la table de la base de données. Cette méthode accepte les données à partir d'un tableau et les stocke sans les vérifier. Les données doivent donc être nettoyées et validées avant de les passer à la méthode.

Chaque élément du tableau passé à la méthode stocke les informations sur le client, avec le nom de champ servant de clé. Cette méthode commence par créer deux tableaux : l'un contenant les clés, l'autre contenant les informations sur le client. Elle reformate ensuite chaque tableau en chaîne. Enfin, elle construit la requête SQL pour insérer le nouvel enregistrement, en plaçant les chaînes à l'emplacement approprié de la requête, et exécute la requête pour créer le nouvel enregistrement.

Le Listing 13.7 présente le code de `createNewAccount`.

Listing 13.7 : Code de la méthode `createNewAccount`.

```
function createNewAccount($data)
{
    if(!is_array($data))                                     #97
    {
        throw new Exception("Les données doivent être dans un tableau.");
        return FALSE;
    }
    foreach($data as $field => $value)                       #102
    {
        if($field != "password" and $field != "Button")
        {
            $fields[] = $field;
            $values[] = addslashes($value);
        }
    }
    $str_fields = implode($fields, ",");                    #110
    $str_values = "'".implode($values, "','");              #111
    $today = date("Y-m-d");                                 #112
    $str_fields .= "create_date";
    $str_fields .= "password";
    $str_values .= "\".\"$today\"";
    $str_values .= "\".md5(\"{$data['password']}\")\"";
    $sql = "INSERT INTO $this->table_name ($str_fields)
           VALUES ($str_values)";
    if(!$this->cxn->query($sql)                             #119
```

```

{
    throw new Exception("Exécution de la requête impossible : "
                        . $this->cxn->error());
    return FALSE;
}
else
{
    return TRUE;
}
}

```

Les chiffres dans les explications suivantes renvoient aux numéros de lignes du Listing 13.7 :

- #97** Débute un bloc if qui s'exécute si les données passées ne sont pas dans un tableau, auquel cas une exception est jetée.
- #102** Débute une boucle foreach qui traverse le tableau de données qui lui a été passé. La boucle crée deux tableaux : \$fields qui contient les noms de champs ; \$values qui contient les contenus des champs.
- #104** Écarte le champ password (mot de passe), soumis plus tard à un traitement spécial, et Button, qui ne doit pas être ajouté aux tableaux car ce n'est pas réellement un nom de champ.
- #106** Construit le tableau \$fields avec les noms de champs issus des clés du tableau de données.
- #107** Construit le tableau \$values. Les valeurs de ce tableau devant être insérées dans la base de données, les guillemets sont échappés avant ajout des valeurs au tableau.

Si le paramètre magic_quotes est activé dans le fichier php.ini, inutile d'échapper ici les guillemets. Pour ce qui concerne les données POST, les guillemets sont échappés automatiquement. Cependant, vous aurez plus de contrôle si vous désactivez le paramètre magic_quotes et échappez vous-même les guillemets dans le script. Quel que soit le moyen choisi, il est important que les guillemets soient échappés avant stockage des données dans la base de données. Pour une étude sur l'échappement des guillemets pour se protéger contre les attaques par injection de commandes SQL, voir le Chapitre 11.



- #110 Appelle la fonction PHP `implode`, qui transforme le tableau `$fields` en chaîne, avec une virgule entre les noms de champs.
- #111 Appelle la fonction PHP `implode`, qui transforme le tableau `$values` en chaîne, avec un guillemet, une virgule, puis un second guillemet (" , ") entre les valeurs. Notez qu'un guillemet, pas une apostrophe, est ajouté au début de la chaîne. Cette chaîne forme la section `VALUES` de la requête, où les valeurs doivent être entre guillemets.
- #112 Stocke la date du jour dans `$today`.
- #113 Les lignes 113 à 116 ajoutent les noms de champs et les valeurs des champs `create_date` et `password` aux chaînes appropriées.
- #117 Les lignes 117 et 118 créent la requête SQL qui insère les données dans la table de la base de données.
- #119 Débute un bloc `if` qui s'exécute si la requête échoue. Le bloc `if` jette une nouvelle exception, avec le message d'erreur MySQL.
- #125 Débute un bloc `else` qui s'exécute si la requête réussit à stocker les informations du nouveau compte. Le bloc retourne `TRUE`.

Écriture de la classe *Session*

La classe `Session` est utilisée de façon que le client n'ait pas à s'identifier dans chaque page du site Web protégé. La classe `Session` débute une session PHP et stocke les variables de session.

Les propriétés

La classe `Session` n'a besoin que d'une seule propriété, qui contient un message stocké par une méthode.

```
private $message ;
```

Le code

Le listing 13.8 présente le code de la classe `Session`. Le constructeur et chacune des quatre méthodes sont étudiés en détail après le listing. À noter, les numéros de lignes à la fin de certaines lignes de code. Les explications à la suite du listing renvoient à ces numéros de lignes.

Listing 13.8 : Code de la classe `Session`.

```
<?php
/* Classe : Session
 * Description : Ouvre et gère une session PHP.
 */

class Session
{
    private $message;

    function __construct()
    {
        session_start();
    }

    function getVariable($varname)
    {
        if(isset($_SESSION['$varname']))           #17
            return $_SESSION['$varname'];
        else                                       #19
        {
            $this->message = "Cette variable n'existe pas dans
                           cette session";

            return FALSE;
        }
    }

    function storeVariable($varname,$value)
    {
        if(!is_string($varname))                 #29
        {
            throw new Exception("Le paramètre 1 n'est pas un
                               nom de variable valide.");

            return FALSE;
        }
        else                                       #35
            $_SESSION['$varname'] = $value;
```

```

    }

    function getMessage()
    {
        return $this->message;
    }

    function login(Account $acct,$password)                                #44
    {
        if(!$acct->comparePassword($password))                            #46
        {
            return FALSE;
        }
        $this->storeVariable("auth","yes");                                #47
        return TRUE;
    }
}
?>

```

Le constructeur

Le constructeur débute la session PHP. La fonction PHP `session_start` vérifie si une session existe déjà. Si ce n'est pas le cas, elle débute une nouvelle session. Si c'est le cas, elle continue la session existante. Le constructeur n'attend pas qu'on lui passe des données. L'instruction pour créer un objet `Session` est donc

```
$sess = new Session();
```

getVariable

Cette méthode retourne la valeur d'une variable de session PHP stockée. Elle vérifie si la variable existe dans la session. Si c'est le cas, la méthode retourne la valeur de la variable. Si la variable n'existe pas, la méthode retourne `FALSE` et stocke un message d'information.

storeVariable

Cette méthode stocke une variable de session PHP. La méthode attend deux valeurs : une chaîne qui est le nom de la variable, et la

valeur de la variable. Les chiffres ci-après renvoient aux numéros de lignes du Listing 13.8 :

- #29 Débute un bloc `if` qui s'exécute lorsque le première paramètre *n'est pas* une chaîne. Le bloc jette une exception avec un message indiquant que le paramètre n'est pas un nom de variable valide.
- #35 Débute un bloc `else` qui s'exécute si le paramètre *est* une chaîne. Le bloc stocke l'information dans le tableau superglobal `$_SESSION` et utilise comme clé le nom de la variable.

getMessage

Cette méthode retourne le contenu de la propriété `$message`.

login

Cette méthode identifie un compte Account dans la session.

- #44 Notez que la méthode attend deux arguments : un objet Account et une chaîne contenant un mot de passe. Le nom de l'objet attendu est inclus dans la signature de la méthode. Si `$acct` n'est pas un objet Account, une erreur fatale se produit :

```
Fatal error : Argument 1 must be an object of class Account in
c:\Session.class on line 39
```

- #46 Appelle la méthode `comparePassword` de l'objet Account qui a été passé à la méthode `login`. Si la méthode `comparePassword` échoue, la méthode `login` retourne `FALSE`.
- #47 Si la méthode `comparePassword` n'échoue pas, la méthode `login` stocke une variable de session PHP nommée `auth` avec la valeur "yes". Durant la session, cette variable peut être vérifiée dans les autres pages pour voir si l'utilisateur est identifié. Si vous le préférez, vous pouvez changer cette méthode pour stocker un autre nom de variable et une autre valeur. En fait, vous pouvez rendre la méthode plus générale, en passant le nom et la valeur de la variable d'autorisation au lieu de les coder en dur directement dans la méthode.

#48 Après stockage de la variable d'autorisation, la méthode login retourne TRUE.

Écriture de la classe Email

Lorsqu'un nouveau client s'inscrit avec succès, l'application envoie un message électronique de confirmation à l'adresse d'e-mail fournie par le client.

Les propriétés

La classe Email stocke l'information nécessaire pour envoyer un message électronique.

```
private $message;  
private $addr;  
private $subj;
```

\$message renferme le contenu du message. \$addr contient l'adresse d'e-mail à laquelle le message sera envoyé. \$subj contient le texte servant de ligne d'objet dans le message électronique.

Le code

Le Listing 13.9 présente le code de la classe Email. Les quatre méthodes sont étudiées en détail après le listing. À noter, les numéros de lignes à la fin de certaines lignes de code. Les explications à la suite du listing renvoient à ces numéros de lignes.

Listing 13.9 : Code de la classe Email.

```
<?php  
/* Classe : Email  
 * Description : Stocke un message électronique.  
 */  
class Email  
{  
    private $message;  
    private $addr;  
    private $subj;  
  
    function setMessage($message)
```

```
{
    if(!is_string($message))
        throw new Exception("Le message doit être une chaîne");
    else
    {
        $this->message = $message;
        return TRUE;
    }
}
```

```
function setAddr($addr)
{
    if(!is_string($addr))
    {
        throw new Exception("L'adresse doit être une chaîne.");
        return FALSE;
    }
    else
    {
        $this->addr = $addr;
        return TRUE;
    }
}
```

```
function setSubj($subj)
{
    if(!is_string($subj))
        throw new Exception("L'objet doit être une chaîne");
    else
    {
        $this->subj = $subj;
        return TRUE;
    }
}
```

[illegible]

```

        throw new Exception("L'objet, l'adresse et le corps
                               du message sont obligatoires.");
    return FALSE;
}
}
}
?>

```

Le constructeur

La classe `Email` n'a pas besoin de constructeur car aucune action n'est à réaliser lors de la création de l'objet `Email`.

setSubj, setAddr, setMessage

Ces méthodes stockent l'information nécessaire pour envoyer le message électronique. Chaque méthode regarde si l'information passée est une chaîne. Si ce n'est pas le cas, elle jette une exception avec un message d'information. Si c'est le cas, elle stocke l'information dans la propriété appropriée et retourne `TRUE`.

sendEmail

Cette méthode envoie le message électronique.

#49 Débute un bloc `if` qui s'exécute lorsque toutes les informations requises sont disponibles. Si aucune des propriétés requises n'est vide, l'e-mail est envoyé. Lorsque l'envoi de l'e-mail réussit, la méthode retourne `TRUE`. Si l'envoi échoue, une exception est jetée, avec un message.

#58 Débute un bloc `else` qui s'exécute si une des propriétés est vide. Une exception est jetée avec un message.

Cette classe `Email` est très simple. Vous verrez facilement où des méthodes supplémentaires peuvent être utiles. Par exemple, une méthode permettant de stocker plusieurs adresses électroniques peut être utile. Une autre méthode intéressante servirait à insérer des en-têtes dans le message électronique, comme un en-tête `from` pour indiquer l'expéditeur. Dans le cas présent, les méthodes suffisent.

Écriture du script de l'application d'identification

Après écriture du code de classe nécessaire à l'application d'identification, vous allez écrire le script de l'application qui crée et utilise les objets. Voici la structure générale du script de l'application :

```

si (le formulaire n'a pas été affiché et soumis auparavant)
    Afficher la page Web d'identification avec les champs
    de formulaire vierges

sinon (si le formulaire a été soumis par l'utilisateur)

    si(l'utilisateur a soumis le formulaire d'identification)
        1 Tester si tous les champs sont renseignés. Si non,
        réafficher le formulaire avec un message d'erreur.
        2 Regarde si le nom d'utilisateur est dans la base
        de données. Si non, réafficher le formulaire, avec
        un message d'erreur.
        3 Tester si le mot de passe est correct. Si non,
        réafficher le formulaire avec un message d'erreur.
        4 Lorsque l'identification réussit, afficher la page
        Web protégée

    sinon, si (l'utilisateur a soumis le formulaire d'inscription)
        1 Tester si tous les champs sont renseignés. Si non,
        réafficher le formulaire avec un message d'erreur.
        2 Tester si l'information est dans le format correct
        Si ce n'est pas le cas, afficher de nouveau le formulaire,
        avec un message d'erreur.
        3 Lorsque l'information est correcte, la stocker dans
        la base de données.
        4 Afficher la page Web protégée.

```

Le programme de l'application crée les objets et se sert de leurs méthodes pour réaliser ces tâches. Le Listing 13.10 présente le script de l'application.

Listing 13.10 : Le script de l'application d'identification.

```

<?php
/* Programme : Login-00.php
 * Description : Script d'application d'identification
 *              d'utilisateur. Le programme affiche la

```



```

*           page Web d'identification. Les informations
*           d'inscription d'un nouvel utilisateur sont
*           validées et stockées dans une base de
*           données. Le mot de passe d'un client existant
*           est comparé au mot de passe valide stocké
*           dans la base de données.
*/
require_once("WebForm.class");                                #9
require_once("Account.class");
require_once("Database.class");
require_once("Session.class");
require_once("Email.class");

try                                                         #15
{
    $form =
        new WebForm("double_form.inc","fields_login.inc",$_POST);
}
catch(Exception $e)
{
    echo $e->getMessage();
    exit();
}
//Le formulaire est affiché pour la première fois, vierge.//
if (!isset($_POST['Button']))                             #26
{
    $form->displayForm();
    exit();
}
// Traite le formulaire qui a été soumis avec des infos utilisateur //
else                                                         #32
{
    $sess = new Session();                                  #34
    try
    {
        $db = new Database("Vars.inc");                    #37
        $db->useDatabase("CustomerDirectory");              #38
        $acct = new Account($db->getConnection(),"Customer");
    }
    catch(Exception $e)
    {
        echo $e->getMessage()."\n<br>";
        exit();
    }

    // Le formulaire d'identification a été soumis //
    if (@$_POST['Button'] == "Identification")              #48
    {

```

```

try
{
    $blanks = $form->checkForBlanks();
}
catch(Exception $e)
{
    echo $e->getMessage();
    exit();
}
if(is_array($blanks))
{
    $GLOBALS['message_1'] =
        "User name or Password was blank.
        Please enter both.";
    $form->displayForm();
    exit();
}
try
{
    if(!$acct->selectAccount($_POST['fusername']))
    {
        $GLOBALS['message_1'] = $acct->getMessage().
            " Réessayez.";
        $form->displayForm();
        exit();
    }
    if(!$sess->login($acct,$_POST['fpassword']))
    {
        $GLOBALS['message_1'] = $acct->getMessage().
            " Réessayez.";
        $form->displayForm();
        exit();
    }
    header("Location: SecretPage.php");
    exit();
}
catch(Exception $e)
{
    echo $e->getMessage();
}
}

// Le formulaire d'inscription a été soumis //
elseif($_POST['Button'] = "Inscription")
{
    $not_required[] = "fax";
    try
    {

```

```

        $form->setFieldsNotRequired($not_required);           #98
        $blanks = $form->checkForBlanks();                   #99
    }
    catch(Exception $e)
    {
        echo $e->getMessage();
    }
    if(is_array($blanks))                                     #105
    {
        $GLOBALS['message_2'] =
            "Les champs obligatoires suivants sont vides.
            Merci de saisir les informations obligatoires : ";
        foreach($blanks as $value)
        {
            $GLOBALS['message_2'] .="$value, ";
        }
        $form->displayform();
        exit();
    }
    $form->trimData();                                       #117
    $form->stripTagsFromData();                             #118
    try
    {
        $errors = $form->verifyData();                       #121
    }
    catch(Exception $e)
    {
        echo $e->getMessage();
    }
    if(is_array($errors))                                    #127
    {
        $GLOBALS['message_2'] = "";
        foreach($errors as $value)
        {
            $GLOBALS['message_2'] .="$value<br> ";
        }
        $form->displayform();
        exit();
    }
    $newdata = $form->getAllFields();                         #137
    try
    {
        if($acct->selectAccount($newdata['user_name']))      #140
        {
            $GLOBALS['message_2'] =
                "Nom d'utilisateur déjà utilisé.
                Choisissez-en un autre.";
            $form->displayForm();
        }
    }

```

```

    exit();
}
if(!$acct->createNewAccount($newdata))           #148
{
    echo "Impossible de créer le nouveau compte.
        Réessayez.";
    exit();
}
$sess->storeVariable("auth","yes");               #154
$sess->storeVariable("logname",$newdata['user_name']);
$em = new Email();                               #156
$em->setAddr($newdata['email']);
$em->setSubj("Votre inscription client");
$emess = "Votre inscription s'est correctement déroulée.";
$emess .= " Vos nom d'utilisateur et mot de passe sont : ";
$emess .= "\n\n\t{$newdata['user_name']}\n\t";
$emess .= "{$newdata['password']}\n\n";
$emess .= "Si vous avez des questions ou problèmes,";
$emess .= " écrivez à service@notreboutique.com";
$em->setMessage($emess);
#$em->sendEmail();                               #167
}
catch(Exception $e)
{
    echo $e->getMessage();
    exit();
}
header("Location: SecretPage.php");
}
?>

```

Dans ce script, de nombreuses instructions sont englobées dans des blocs try/catch. Lorsqu'une méthode jette une exception qui n'est pas interceptée, une erreur fatale se produit :

```

Fatal error: Uncaught exception 'Exception' with message
'Base indisponible.' in c:\Database.class:56

```

Il faut donc intercepter toute exception jetée par une méthode, dans la méthode elle-même ou dans le script qui l'utilise.

Les explications suivantes renvoient aux numéros de lignes du Listing 13.10 :

#9 Les lignes 9 à 16 insèrent les fichiers nécessaires.

- #15** Débute un bloc `try/catch` qui crée l'objet `WebForm`.
- #26** Débute un bloc `if` qui s'exécute lorsqu'aucun bouton n'est cliqué, signifiant que le formulaire n'a pas encore été soumis. Le bloc affiche la page Web d'identification, avec les champs de formulaire vierges.
- #32** Débute un bloc `else` qui s'exécute si un bouton *est* cliqué, signifiant que l'utilisateur a soumis le formulaire. Ce bloc opère le traitement de formulaire et l'authentification de mot de passe.
- #34** Crée un objet `Session`.
- #37** Les lignes 37 et 38 créent un objet `Database` et sélectionnent la base de données correcte.
- #39** Crée un objet `Account`.
- #48** Débute un bloc `if` qui s'exécute lorsque l'utilisateur soumet le formulaire d'identification. Ce bloc teste si le nom d'utilisateur et le mot de passe soumis sont valides.
- #52** Recherche les champs vides dans le formulaire d'identification. Aucun ne peut être vide.
- #59** Débute un bloc `if` qui s'exécute si des champs sont vides. Un message d'erreur est composé, et le formulaire est réaffiché. Notez que le message d'erreur est stocké dans le tableau `$GLOBALS` de façon que la méthode `WebForm` ait accès au message.
- #69** Débute un bloc `if` qui s'exécute lorsque le nom d'utilisateur *n'est pas* trouvé dans la base de données. Un message d'erreur est composé, le formulaire est réaffiché, et le script se termine.
- #76** Débute un bloc `if` qui s'exécute lorsque le mot de passe du formulaire ne correspond pas au mot de passe stocké dans la base de données pour cet utilisateur. Un message d'erreur est composé, et le formulaire est réaffiché.
- #83** Affiche une page Web protégée. Le nom `SecretPage.php` n'est qu'un exemple de nom. Vous emploierez le nom du script générant la page que vous désirez afficher une fois identifié l'utilisateur.

- #93 Débute un bloc `elseif` qui s'exécute lorsque l'utilisateur soumet le formulaire d'inscription. Ce bloc traite et stocke les données des champs de formulaire.
- #95 Crée un tableau contenant le nom du champ qui peut être vide. Dans ce cas, `fax` est le seul champ qui peut être laissé vide.
- #98 Définit le nom du champ autorisé à être vide.
- #99 Recherche les champs vides du formulaire. Un tableau des noms de champs vides est retourné. Si `fax` est vide, il est ignoré.
- #105 Débute un bloc `if` qui s'exécute si le tableau `$blank` contient des éléments – c'est-à-dire, si des champs sont vides. Un message d'erreur est composé, et le formulaire est réaffiché. Notez que le message d'erreur est stocké dans le tableau `$GLOBALS` de façon que la méthode `WebForm` ait accès au message.
- #117 Supprime les blancs initiaux et terminaux dans tous les champs.
- #118 Supprime les balises HTML dans les champs.
- #121 Vérifie que les données sont au format correct. Si des données sont incorrectement formatées, les méthodes retournent un tableau de messages d'erreur.
- #127 Débute un bloc `if` qui s'exécute si le tableau `$errors` contient des éléments – c'est-à-dire lorsque des champs contiennent des données erronées. Un message d'erreur est composé, et le formulaire est réaffiché avec le message d'erreur.
- #137 Lit les données de l'objet `WebForm`. Il est nécessaire de stocker les données de l'objet. Les données ne sont pas stockées à partir du tableau `$_POST`, qui contient les données que l'utilisateur a saisies dans le formulaire, car elles ont pu être modifiées au niveau des lignes 120 et 121.
- #140 Débute un bloc `if` qui s'exécute lorsque le nom d'utilisateur a été trouvé dans la base de données. Les doublons de noms d'utilisateur ne sont pas autorisés. Un message d'erreur est composé, et le formulaire est réaffiché.

- #148 Débute un bloc if qui s'exécute si la méthode `createNewAccount` échoue. Un message d'erreur est affiché, et le script se termine.
- #154 Stocke la variable de session indiquant que l'utilisateur s'est identifié avec succès. Le script atteint cette ligne uniquement lorsqu'aucune erreur n'a été rencontrée.
- #155 Stocke le nom d'utilisateur dans une variable de session pour usage ultérieur en cours de session.
- #156 Les lignes 156 à 167 créent et envoient un message électronique au client lui signalant que son nouveau compte a été bien créé.
- #174 Affiche une page Web protégée. Le nom `SecretPage.php` n'est qu'un exemple de nom. Vous emploierez le nom du script générant la page que vous désirez afficher une fois identifié l'utilisateur.

Protection de vos pages Web

Les pages protégées de votre site Web ne sont pas différentes des autres pages Web. L'idée est ici de limiter leur affichage aux seuls utilisateurs identifiés. Pour cela, en haut de chaque page, vous vérifiez si l'utilisateur est identifié.

Lorsque l'utilisateur s'identifie via le script d'application `Login-00.php` décrit dans la section précédente, une session est démarrée, et la valeur "yes" est stockée dans une variable de session, comme ceci :

```
$sess->setVariable("auth", "yes");
```

Il est possible de vérifier cette variable de session `$auth` en haut de chaque page protégée, pour voir si elle vaut "yes". Si c'est le cas, l'utilisateur est identifié. Pour vérifier la variable de session `$auth`, ajoutez les instructions suivantes au début de chaque script :

```
require_once("Session.class");
$sess = new Session();
if($sess->getVariable("auth") != "yes")
{
    header("Location: Login-00.php");
    exit();
}
```

Lorsque vous créez l'objet session, PHP regarde s'il existe déjà une session en cours pour cet utilisateur. Si c'est le cas, les variables de session actuelle sont rendues disponibles au script.

L'instruction `if` teste si la variable de session `$auth` vaut "yes". Lorsque `$auth` ne vaut pas "yes" ou que `$auth` n'existe pas, l'utilisateur n'est pas identifié, alors le bloc `if` est exécuté : l'utilisateur est mené à la page Web d'identification et ce script s'achève. Si `$auth` vaut "yes", le script continue pour afficher le contenu de la page Web.

Ajout de caractéristiques à l'application

L'application d'identification présentée dans ce chapitre fournit une fonctionnalité élémentaire d'identification. D'autres fonctions peuvent y être ajoutées. Voici quelques fonctions courantes des applications d'identification non étudiées dans ce chapitre :

- ✓ **Bouton mot de passe oublié** : Il est quasiment sûr que des utilisateurs oublieront leur mot de passe. Il est alors pratique de fournir un bouton sur lequel l'utilisateur cliquera lorsqu'il ne se souvient plus de son mot de passe. Certaines applications envoient le mot de passe à l'utilisateur par e-mail, certaines présentent une page où ce dernier peut créer un nouveau mot de passe.

Pour envoyer par e-mail le mot de passe extrait de la base de données, vous devez employer une autre fonction de cryptage de mot de passe, car `md5()`, utilisée dans cette application, est une fonction de cryptage à sens unique. Vous ne pouvez pas lire le mot de passe dans sa forme d'origine. Le mot de passe est protégé de tous, même de vous. Pour un moyen de cryptage à deux sens, pour pouvoir déchiffrer le mot de passe et le transmettre à l'utilisateur, voyez les fonctions AES et DES de MySQL ou la fonction `mcrypt` de PHP.

Au lieu de lire le mot de passe et le transmettre à l'utilisateur, qui constitue une procédure non sécurisée, vous pouvez présenter à l'utilisateur une page Web où il pourra créer un nouveau mot de passe. Cependant, il faut être sûr que seul le véritable propriétaire du compte pourra changer le mot de passe. Lors de la création du compte, de nombreuses applications posent une question secrète à l'utilisateur, dont la réponse est stockée dans la base. Cette question est du type

"Nom de jeune fille de votre mère". Avant toute modification du compte, cette réponse est réclamée à l'utilisateur.

- ✓ **Gestion de compte** : Il arrive que l'on déménage et que l'on change de numéro de téléphone. Une adresse électronique peut aussi changer. Il sera pratique de permettre à l'utilisateur de modifier les informations stockées dans son compte. Il faut dans ce cas fournir un bouton "Gérer votre compte" qui affiche une page Web où l'utilisateur pourra changer ses adresse, numéro de téléphone, mot de passe, etc.

Avant d'ajouter de telles fonctions, je vous suggère de vous familiariser avec l'application telle quelle. Lorsqu'elle fonctionnera correctement, vous pourrez ajouter des fonctions, une à la fois. Ne modifiez pas trop de choses en même temps. Il est plus facile de déboguer une fonction à la fois.



En général, l'ajout de fonctions à une application orientée objet est plus facile qu'à une application procédurale. L'un des points forts de la programmation orientée objet est de pouvoir ajouter du code sans modifier le code existant. Si vous pensez que votre application grossira à l'avenir, il est probablement plus judicieux de travailler sur notre exemple d'application orientée objet.

Sixième partie

Création d'applications de vente en ligne



"Mme Lamont, depuis combien de temps
envoyez-vous des factures mentionnant le port,
la manutention et la TVA sous les appellations
Ceci, Cela, et Autres bricoles ?"

Dans cette partie...

Cette partie présente deux applications liées à la vente en ligne. La première application affiche un catalogue de produits (Chapitre 14). La seconde application permet au client d'acheter des produits en ligne (Chapitre 15). Pour chaque application, deux méthodes différentes sont présentées – procédurale et orientée objet.

Chapitre 14

Application de catalogue en ligne

Dans ce chapitre :

- Conception des pages affichant les produits.
- Création de la base de données stockant les informations produit.
- Écriture du code procédural pour l'application catalogue.
- Développement et emploi des objets pour programmer l'application catalogue.

Le catalogue en ligne est une des applications les plus courantes sur le Web. Que le site soit proposé par une entreprise individuelle proposant une poignée de produits ou une énorme société avec des milliers de produits, le principe est le même. Le client veut pouvoir visionner les produits.

Sur de nombreux sites, les clients peuvent acheter les produits du catalogue directement en ligne. Ce chapitre présente une application catalogue sans fonction d'achat en ligne. L'application étudiée gère uniquement l'affichage du catalogue. L'application présentée au chapitre suivant fournit cette capacité d'acheter en ligne.

Conception de l'application de catalogue en ligne

La fonction élémentaire de l'application de catalogue en ligne est d'afficher des produits. Si un magasin n'offre qu'une dizaine de produits, il est possible de les afficher sur une même page. Cependant, il est fréquent que tous les produits ne puissent être raisonnablement groupés sur une seule page. Habituellement, les produits

sont classés par catégories. Lorsque le nombre de produits est faible, un classement sur un niveau de catégorie est suffisant. En revanche, si le magasin propose un grand nombre de produits, il faut exploiter deux, trois niveaux de catégories ou plus pour catégoriser les produits en catégories assez concises pour être affichées. L'exemple de ce chapitre est une boutique qui vend des articles d'alimentation. Deux niveaux de catégories sont utilisés. Les articles sont d'abord catégorisés selon un premier niveau de catégorie : fruits, légumes, herbes aromatiques, et ainsi de suite. Au second niveau du premier niveau fruits, on trouvera pommes, oranges, et cerises. Le produit peut être Reinette ou Granny Smith, auquel cas il sera dans la catégorie fruits : pommes.

Lorsque les produits sont catégorisés, le catalogue en ligne affiche généralement en premier une page montrant les catégories disponibles. Le client choisit une catégorie pour en visionner les articles. S'il y a plusieurs niveaux de catégories, le client choisit dans les différents niveaux de catégories, au fur et à mesure, jusqu'à atteindre la page de l'article.

Même avec plusieurs niveaux de catégories, il se peut qu'il existe un grand nombre d'articles dans le dernier niveau. Par exemple, un grand de l'habillement dispose de nombreux articles dans la catégorie "Robes" ou même "Robes de soirée". Il est courant, dans ce cas, de n'afficher qu'un certain nombre de produits par page, souvent dix. Le client clique alors sur un bouton pour voir le jeu d'articles suivant ou précédent.

Pour mettre en place cette possibilité, l'application de catalogue en ligne doit :

- ✓ **Afficher les catégories d'articles parmi lesquelles l'utilisateur peut choisir.**
- ✓ **Afficher les produits de la catégorie sélectionnée par l'utilisateur.** Elle affichera toutes les informations produit nécessaires au client (prix, description, etc.). Lorsque la liste des produits est longue, l'application affichera les articles page par page.

Création de la base de données du catalogue

L'application requiert une base de données pour stocker les informations produits. La base de données est le catalogue, le noyau de

l'application. La base de données stocke les noms d'articles, numéros de commandes, description, prix et toute autre information pertinente, telle que taille, couleur, etc.

Conception de la base de données du catalogue

La première tâche est de déterminer les informations à stocker. Ce que l'on stocke dépend du type d'article. Vous devez stocker toute information susceptible d'être nécessaire au client pour décider quel produit acheter. Le propriétaire de la boutique, qui connaît ses produits et ce que ses clients ont besoin de savoir, vous fournira ces informations, ainsi que des images des produits. Voici quelques informations à stocker :

- ✓ **Nom du produit** : Il est évident que le client a besoin de cette information.
- ✓ **Identifiant produit** : Dans la plupart des cas, le nom du produit n'est pas unique, il faut donc stocker un *identifiant produit*, numéro unique qui identifie chaque produit.
- ✓ **Description du produit** : Description textuelle du produit.
- ✓ **Taille** : Un produit peut être vendu dans diverses tailles. Même lorsqu'une seule taille existe, le client a éventuellement besoin de connaître les dimensions, pour diverses raisons. Par exemple, vous ne vendez qu'une seule table de salon : le client doit en connaître les dimensions, pour savoir si elle ira bien dans sa pièce.
- ✓ **Couleur** : Un article peut exister en plusieurs couleurs.
- ✓ **Prix** : Information indispensable !
- ✓ **Disponibilité de l'article** : Le client veut également savoir quand le produit a été ajouté au catalogue, s'il est en stock ou quand est prévu son approvisionnement.

Vous pouvez aussi ajouter dans la base de données des informations qui vous seront réservées. Par exemple, des informations sur les entreprises fournissant le produit. Ces informations seront stockées dans la base de données mais n'ont pas à être montrées au client.

La boutique de notre exemple s'appelle Alimentation. Pour l'instant, elle vend des fruits et légumes mais le propriétaire espère bientôt proposer d'autres articles.

La base de données ne contient qu'une table. Les informations produits sont stockées à raison d'une ligne par article. Les champs de cette table sont présentés au Tableau 14.1.

Tableau 14.1 : Table de base de données : Customer.

Nom de variable	Type	Description
catalog_number	INT(6)	Les numéros d'identification produit sont affectés séquentiellement par MySQL (clé primaire).
name	VARCHAR(40)	Nom de l'article.
added_date	DATE	Date d'ajout du produit au catalogue.
category	VARCHAR(20)	Nom de catégorie de premier niveau.
type	VARCHAR(20)	Nom de catégorie de second niveau.
description	VARCHAR(255)	Description du produit.
price	DECIMAL(7,2)	Prix du produit. Les prix sont à la livre (500 g).
pix	VARCHAR(20)	Nom du fichier graphique qui contient l'image du produit.

Le tableau comporte huit champs. À l'exception de la description, tous les champs sont requis, ils ne peuvent être vides. Le champ Description peut être laissé vierge. Elle sera éventuellement ajoutée plus tard.

Le champ catalog_number est le numéro d'article identifiant le produit de manière unique. Ce numéro sert lorsque le client commande le produit. C'est un champ AUTO_INCREMENT : MySQL lui affecte un numéro séquentiel lors de l'ajout du produit à la base de données. Dans certaines boutiques, un identifiant significatif (une référence) est saisi à la place d'un simple numéro séquentiel.

Le champ pix contient un nom de fichier par défaut. Lorsqu'un nom de fichier n'est pas saisi, c'est le fichier d'image par défaut (Missing.jpg) qui est retenu. Cette image indique "image not available" (image indisponible).

Création de la base de données du catalogue

L'instruction SQL suivante crée cette base de données :

```
CREATE DATABASE FoodCatalog;
```

L'instruction SQL suivante crée la table :

```
CREATE TABLE Food (
  catalog_number INT(6) NOT NULL AUTO_INCREMENT,
  name VARCHAR(20) NOT NULL,
  added_date DATE NOT NULL,
  category VARCHAR(20) NOT NULL,
  type VARCHAR(20) NOT NULL,
  description VARCHAR(255),
  price DECIMAL(7,2) NOT NULL,
  pix VARCHAR(20) NOT NULL DEFAULT "Missing.jpg",
  PRIMARY KEY(catalog_number) );
```

Accès à la base de données d'alimentation

PHP fournit des fonctions MySQL pour accéder à la base de données à partir du script PHP. Les fonctions MySQL reçoivent les informations nécessaires pour accéder à la base de données, tel qu'un nom de compte MySQL et un mot de passe. Cela n'a pas de lien avec d'autres noms de compte et mots de passe que vous utilisez, comme un mot de passe pour vous identifier sur le système.

PHP fournit deux jeux de fonctions MySQL :

- ✓ **mysql** : Fonctions MySQL développées pour les versions de MySQL jusqu'à 4.0. Bien que vous puissiez continuer à employer ces fonctions dans les versions plus récentes de MySQL, elles ne donnent pas accès à certaines caractéristiques évoluées de MySQL. Les fonctions sont au format `mysql_action-opérée()`, comme `mysql_connect()` et `mysql_query()`. En principe, vous avez utilisé PHP et MySQL avant de lire ce livre, vous devriez donc connaître ces fonctions.
- ✓ **mysqli** : Fonctions MySQL Improved (MySQL amélioré) développées pour exploiter les caractéristiques évoluées de MySQL 4.1 et versions ultérieures. L'extension MySQL Improved n'est disponible que dans PHP 5, pas PHP 4. Les fonctions sont au format `mysqli_action-opérée()`, comme

`mysqli_connect()` et `mysqli_query()`. De plus, l'extension MySQL Improved propose certaines classes intégrées, pour l'exploitation d'objets lors de l'emploi de votre base de données.

MySQL 4.1 est aujourd'hui la version recommandée sur le site Web MySQL : nous utiliserons donc dans ce chapitre les fonctions MySQL Improved. Les fonctions procédurales sont employées pour la construction des programmes procéduraux. Pour la construction des programmes orientés objet, ce sont les classes orientées objet qui le sont.



Si vous utilisez PHP 4 ou pour quelque autre raison voulez employer les fonctions `mysql` au lieu des fonctions `mysqli`, vous aurez quelques petites modifications à apporter à la syntaxe. Les fonctions `mysqli` sont similaires aux fonctions `mysql`. Pour plus d'informations sur les fonctions, voir le manuel PHP sur www.php.net/manual/en/ref.mysqli.php et www.php.net/manual/en/ref.mysql.php.

Pour notre exemple d'application, j'ai stocké les informations nécessaires aux fonctions `mysqli` de PHP dans un fichier séparé, nommé `Vars.inc`. Ce fichier est stocké dans un répertoire en dehors de mon espace Web, pour des raisons de sécurité. Le fichier contient des informations semblables à celles-ci :

```
<?php
    $host = "localhost";
    $user = "admin";
    $passwd = "xy.34W";
    $database = "FoodCatalog";
?>
```



Notez les balises PHP au début (`<?php`) et à la fin (`?>`) du fichier. Si vous n'insérez pas ces balises, l'information s'affiche sur la page Web, aux yeux de tous, ce qui n'est pas souhaitable.

Ajout de données à la base de données

Cette base de données contiendra les informations sur les produits.

Création des pages Web du catalogue

Le catalogue en ligne nécessite deux types de pages Web. L'une affiche un index des catégories de produits, où le client choisit la

catégorie qui l'intéresse. Si votre catalogue comporte des sous-catégories, il est possible d'afficher la page d'index plusieurs fois, une pour chaque niveau de catégorie. Le second type de page est la page produit, qui affiche les informations produit des articles de la catégorie sélectionnée.

Conception des pages Web du catalogue

Les catalogues en ligne abondent sur le Web. Vous en avez rencontré beaucoup, avec des aspects et comportements différents. Cependant, différents designs sont susceptibles de fournir les mêmes fonctions. Vous savez peut-être déjà exactement quel design appliquer mais gardez à l'esprit que le design le plus fonctionnel dépend en grande partie du type et de la quantité d'articles au catalogue.

Le catalogue présenté dans ce chapitre propose de l'alimentation. Les informations à afficher pour chaque article sont le nom, la description, le prix et une photo. Ces données iront sans problème sur une ligne ou deux à l'écran. D'autres produits réclameront plus ou moins d'espace d'affichage. Certains catalogues affichent un produit par page.

Il faut concevoir deux types de pages : une page d'index qui affiche les catégories et une page produit qui affiche les produits d'une catégorie.

Conception de la page d'index

La page d'index affiche les catégories dans un formulaire, de telle sorte que l'utilisateur puisse sélectionner une catégorie. Dans notre exemple, les catégories sont affichées dans le formulaire sous forme de boutons d'options. La Figure 14.1 montre la page d'index du catalogue en ligne affichée dans un navigateur.

Le code de la page d'index est stocké dans des fichiers séparés qui sont inclus lorsque l'application a besoin d'afficher cette page. Le code qui définit la page Web est donc séparé du code PHP qui fournit la logique de l'application.

Le code de la page d'index du catalogue se compose de deux fichiers : d'une part, le code qui définit l'aspect de la page, d'autre part le code qui fournit les informations spécifiques à la page.

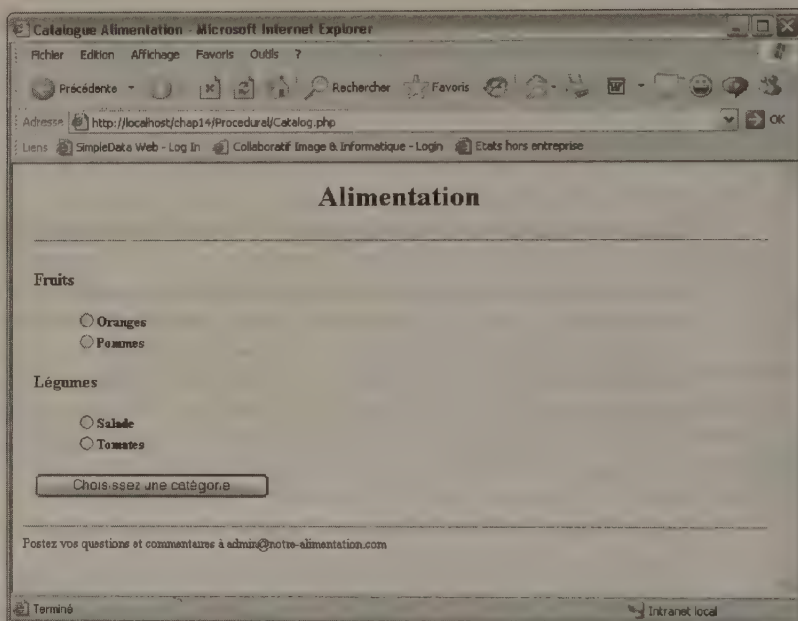


Figure 14.1 :
La page
d'index du
catalogue en
ligne.

Conception de la page produits

La page produits d'un catalogue doit afficher les articles pour que le client dispose de toutes les informations qui s'y rapportent. Si tous les produits ne rentrent pas sur une seule page, il doit y avoir autant de pages produits que nécessaire pour présenter tous les produits de la catégorie. Certains catalogues affichent uniquement la liste des articles, avec un lien vers des pages présentant des informations supplémentaires.

Dans notre exemple de catalogue, l'information concernant chaque article tient sur une ligne ou deux : plusieurs produits peuvent donc être affichés sur une page. En bas de la page, un formulaire présente des boutons servant à afficher la page suivante, la précédente, ou pour revenir à la page des catégories. La Figure 14.2 présente la page produits du catalogue.

Le code de la page produits est stocké dans des fichiers séparés, comme pour la page d'index : d'une part le fichier qui définit l'aspect de la page, d'autre part le fichier qui fournit les informations spécifiques à la page.

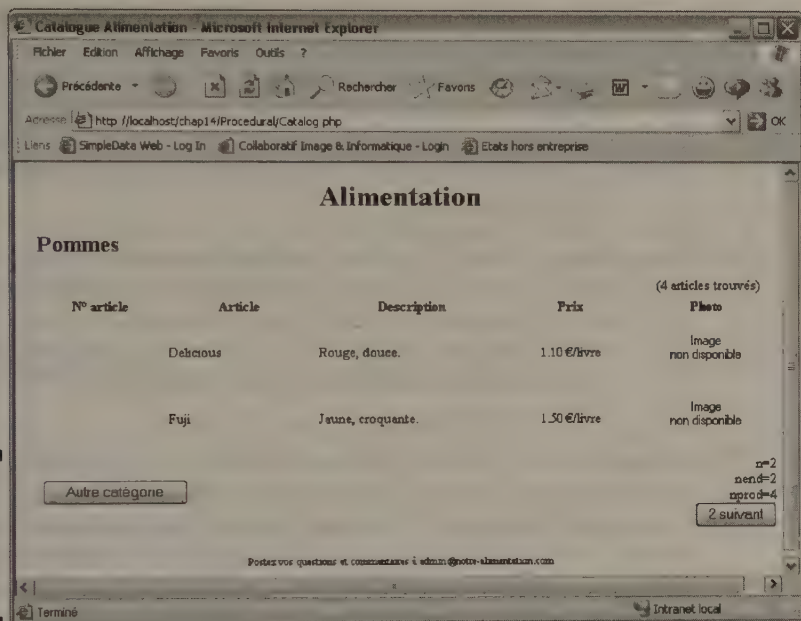


Figure 14.2 :
La page
produits du
catalogue en
ligne.

Écriture du code de la page d'index

La page d'index du catalogue présente un simple formulaire qui contient la liste de catégories. Le catalogue Alimentation présente deux niveaux de catégories. Cependant, le catalogue comportant peu de catégories à ce stade, ces deux niveaux de catégories peuvent être affichés sur une seule page d'index. Certains catalogues comportent de nombreuses catégories et seules les catégories de premier niveau sont alors affichées dans la page d'index. Pour afficher les catégories de second niveau, le client doit cliquer sur une catégorie de premier niveau. Dans notre exemple de catalogue Alimentation, il n'y a aucune nécessité d'afficher séparément les niveaux de catégories.

Le code qui crée la page d'index se trouve dans deux fichiers séparés :

- ✓ `catalog_index_page.inc` : Contient le code qui définit l'aspect de la page Web. Il génère une page Web contenant un formulaire énumérant les catégories. Les catégories de premier niveau sont présentées sous forme de titres. Les catégories de second niveau sont énumérées sous chaque catégorie de

premier niveau. Chaque catégorie de second niveau est présentée sous forme d'un bouton d'option : le client clique donc sur la catégorie de produits qu'il veut visionner. Ce fichier ne comporte pas d'informations spécifiques comme les noms de catégories affichés à côté des boutons d'options. Un autre fichier doit être utilisé en conjonction avec celui-ci pour créer la page Web.

- ✓ `fields_index_page.inc` : Contient les informations spécifiques à la page d'index du catalogue. Utilisé en conjonction avec `catalog_index_page.inc`, il génère un formulaire où le client sélectionne une catégorie. Pour créer une page Web différente, il suffira d'exploiter un autre fichier avec `catalog_index_page.inc`.

Le reste de cette section présente les détails de ces fichiers. Le second fichier est court, plus facile à comprendre, aussi est-il étudié en premier (Listing 14.1). Ensuite, lors de l'explication du premier fichier, `catalog_index_page.inc` au Listing 14.2, je ferai référence au contenu de `fields_index_page.inc`. Ces deux fichiers sont utilisés tant dans l'application procédurale qu'orientée objet.

Écriture de `fields_index_page.inc`

Le code du Listing 14.1 fournit les informations spécifiques à la page Web. Dans notre exemple, un seul tableau (array) est nécessaire. Le tableau `$page` contient les éléments affichés en haut et en bas de la page.

La configuration des éléments et des champs dans ce fichier séparé, au lieu de les inclure dans le fichier avec le code HTML du formulaire, simplifie grandement la création et la maintenance du formulaire. Les champs et éléments sont facilement modifiables dans ce fichier.

Listing 14.1 : Tableaux nécessaires à la page d'index.

```
<?php
/* Fichier : fields_index_page.inc
 * Description : Construit les tableaux nécessaires pour afficher
 *              les catégories d'articles du catalogue.
 */
$page = array( "title" => "Catalogue Alimentation",
              "top"   => "Alimentation",
```

```

"bottom" => "Postez vos questions et commentaires
à admin@notre-alimentation.com",
);
?>

```

Notez que le tableau `$page` est défini dans un format structuré. Il pourrait être défini avec beaucoup moins d'espace mais ce format donne des valeurs plus claires et plus faciles à modifier si besoin.

Écriture de `catalog_index_page.inc`

Ce script contient le code qui définit l'aspect de la page Web. Il contient le code HTML des formulaires et des tableaux pour organiser la page. Le code inclut les variables pour les endroits où des informations spécifiques sont nécessaires. Les valeurs des variables sont fournies dans le précédent fichier, `fields_index_page.inc`. Par exemple, le script contient la ligne suivante, qui définit le titre de la page Web :

```
<head><title><?php echo $page['title']?></title></head>
```

La variable `$page['title']` se trouve dans le fichier `fields_index_page.inc`, où elle vaut "Catalogue Alimentation".

Listing 14.2 : Script définissant la page d'index du catalogue.

```

<?php
/* Fichier : catalog_index_page.inc
 * Description : Affiche les catégories du catalogue.
 */
?>
<html>
<head><title><?php echo $page['title'] ?></title></head>           #7
<body>
<?php
/* Affiche le texte avant le formulaire */
echo "<div style='margin-left: .1in'"
<h1 align='center'> {$page['top']}</h1><hr>";                       #12

/* Crée le formulaire contenant la liste de sélection */
echo "<form action='$_SERVER[PHP_SELF]' method='POST'>\n";
foreach($food_categories as $key => $subarray)                     #16

```

```

[
    echo "<h3>$key</h3>";
    echo "<ul>";
    foreach($subarray as $type)                                #20
    {
        echo "<input type='radio' name='interest'
            value='$type'><b>$type</b><br>\n";
    }
    echo "</ul>";
}
echo "<p><input type='submit' name='Products'                #26
    value='Choisissez une catégorie'>\n
    </form>\n";                                              #29
?>
</div>
<hr><font size="-1"><?php echo $page['bottom'] ?>            #32
</body></html>

```

Les numéros suivants renvoient aux numéros de lignes du Listing 14.2 :

- #7** Inclut une courte section PHP qui affiche le titre.
- #12** Inclut une courte section PHP qui affiche le titre supérieur dans la page Web.
- #15** Affiche l'HTML de la balise de formulaire débutant le formulaire.
- #16** Débute une boucle foreach qui traverse le tableau `$food_categories`. La boucle affiche le contenu du tableau. La boucle externe affiche les catégories de premier niveau. La boucle se termine ligne 26.
- #20** Débute une boucle foreach interne qui affiche les catégories de second niveau.
- #27** Les lignes 27 à 29 affichent l'HTML qui affiche le bouton d'envoi et termine le formulaire. Le bouton d'envoi est nommé `Products` (Produits en anglais) car il déclenche l'affichage des produits.
- #32** Affiche l'élément "inférieur" du tableau `$page`.

Écriture du code de la page produits

La page produits du catalogue affiche une liste de détails sur les articles. Les articles sont affichés page par page. Un petit formulaire à la fin de chaque page affiche des boutons d'envoi de formulaire, pour aller à la page suivante, la précédente et la page d'index.

Le code générant la page produits se trouve dans deux fichiers séparés :

- ✓ `catalog_product_page.inc` : Contient le code qui définit l'aspect de la page Web. produit une page Web énumérant les articles dans un tableau (HTML). À la suite des tableaux, un petit formulaire est affiché : il ne comporte pas de champ, juste des boutons d'envoi de formulaire. Des champs masqués du formulaire passent les informations nécessaires à la page suivante. Ce fichier ne contient pas d'informations sur les articles. Un autre fichier doit être utilisé en conjonction avec celui-ci pour créer la page Web.

Ce fichier est un peu différent dans l'application procédurale et dans l'application orientée objet. Les différences sont signalées à la suite du listing.

- ✓ `fields_product_page.inc` : Contient les informations spécifiques à la page produits du catalogue. Lorsqu'il est utilisé avec `catalog_product_page.inc`, ce fichier affiche les informations sur les articles. Pour générer une page Web qui affiche différentes informations sur les produits, il suffit d'exploiter un autre fichier avec `catalog_product_page.inc`.

Ce fichier est le même pour l'application procédurale et l'application orientée objet.

Le reste de cette section présente les détails de ces fichiers. Le second fichier est court, plus facile à comprendre, aussi est-il étudié en premier (Listing 14.3). Ensuite, lors de l'explication du premier fichier, `catalog_product_page.inc` (Listing 14.4), je ferai référence au contenu de `fields_product_page.inc`.

Écriture de `fields_product_page.inc`

Le code du Listing 14.3 construit trois tableaux (array) contenant les informations spécifiques affichées dans la page produits.

- ✓ `$page` : Contient les éléments affichés en haut et en bas de la page.
- ✓ `$table_heads` : Contient les éléments fournissant les titres du tableau (HTML) affichant les articles.

Il est facile de visualiser et modifier les champs et éléments dans ce fichier séparé.

Listing 14.3 : Tableaux nécessaires à la page produits

```
<?php
/* Fichier : fields_products_page.inc
 * Description : Définit les variables et construit les tableaux
 *              nécessaires à l'affichage de la page produits du catalogue.
 */
$page = array( "title" => "Catalogue Alimentation",
              "top"    => "Alimentation",
              "bottom" => "Postez vos questions et commentaires
                          à admin@notre-alimentation.com",
              );
$table_heads = array("catalog_number" => "N° article",
                    "name"            => "Article",
                    "description"      => "Description",
                    "price"            => "Prix",
                    "pix"              => "Photo",
                    );
?>
```

`$page` et `$table_heads` sont créés via des instructions de tableaux structurées, faciles à lire et mettre à jour.

Écriture de `catalog_product_page.inc`

Le Listing 14.4 présente le code de `catalog_product_page.inc`. Comme le fichier `catalog_index_page.inc` du Listing 14.2, celui-ci définit l'aspect de la page Web. Il inclut le code HTML qui affiche la page. Il inclut également de petites sections pour afficher les données de `fields_products_page.inc`.

Le Listing 14.4 présente le fichier utilisé pour l'application procédurale. À la suite du listing, j'explique les changements devant y être apportés pour l'employer dans l'application orientée objet.

Listing 14.4 : Script définissant la page produits du catalogue.

```
<?php
/* Fichier : catalog_product_page.inc
 * Description : Affiche les articles de la
 *               catégorie choisie.
 */
?>
<html>
<head><title><?php echo $page['title'] ?></title></head>
<body>
<?php
echo "<div style='margin-left: .1in; margin-right: .1in'>
    <h1 align='center'>{$page['top']}</h1>
    <p><font size='+2'><b>{$_POST['interest']}</b></font>";
echo "    <p align='right'>({$n_products} articles trouvés)"; #14
echo "<table border = '0' cellpadding = '3' width = '100%'>";
echo "<tr>";
foreach($table_heads as $heading) #17
{
    echo "<th>{$heading}</th>";
}
echo "</tr>";
for ($i=$n_start;$i<=$n_end;$i++) #22
{
    echo "<tr>";
    echo "<td align='right'>{$products[$i]['food_id']}</td>";
    echo "<td>{$products[$i]['name']}</td>";
    echo "<td>{$products[$i]['description']}</td>";
    echo "<td style='text-align: center'>
        {$products[$i]['price']} €/livre</td>";
    echo "<td style='text-align: center'>
        <img src='images/{$_products[$i]['pix']}'
        width='70' height='55'></td>";
    echo "</tr>";
}
echo "<form action='$_SERVER[PHP_SELF]' method='POST'>"; #35
echo "<input type='hidden' name='n_end' value='{$n_end}'>";
echo "<input type='hidden' name='interest'
    value='{$_POST[interest]}'>";
echo "<tr>
    <td colspan='2'>
        <input type='submit'

```

```

        value='Autre catégorie'></td>
    <td colspan='3' align='right'>";
echo "n=$n_per_page<br>";
echo "nend=$n_end<br>";
echo "nprod=$n_products<br>";
    if($n_end > $n_per_page)                                #44
    {
        echo "<input type='submit' name='Products'
            value='Préc.'>";
    }
    if($n_end < $n_products)                                  #49
    {
        echo "<input type='submit' name='Products'
            value='$n_per_page suivant'>";
    }
echo "</td></form></tr></table>";                            #54
echo "<p style='text-align: center; font-size: 75%'>
    {$page['bottom']}";
?>
</div></body></html>

```

Les numéros suivants renvoient aux numéros de lignes du Listing 14.4 :

- #8** Inclut une courte section PHP qui affiche le titre.
- #11** Les lignes 11 à 14 affichent la section supérieure de la page Web, dont une ligne qui indique la catégorie présentée (lignes 12 et 13) et une ligne signalant le nombre total d'articles de cette catégorie (ligne 14).
- #15** Affiche la balise de tableau qui commence le tableau des articles.
- #17** Débute une boucle `foreach` affichant les en-têtes de colonnes dans la première ligne du tableau.
- #22** Débute une boucle `for` traversant les articles du tableau `$products`. Cette boucle génère une ligne de tableau pour chaque produit.
- #24** Les lignes 24 à 33 affichent chaque élément d'information dans la ligne d'un article.

- #35 Affiche la balise débutant le formulaire. Le formulaire se trouve à la dernière ligne du tableau des articles, il contient les boutons Suivant, Précédent, Autre catégorie.
- #36 Les lignes 36 à 38 créent des variables masquées qui passent les informations nécessaires à la page suivante : \$n_end et la catégorie actuellement affichée.
- #40 Crée une grande cellule pour y afficher le bouton d'envoi Autre catégorie. Le bouton ne reçoit pas de nom.
- #43 Crée une grande cellule sur la droite, pour y afficher les boutons d'envoi Suivant et Précédent.
- #44 Débute une instruction `if` qui affiche le bouton Précédent. Si la page affichée n'est pas la première page, le bouton Précédent est affiché. Le bouton Précédent est nommé Products (pour Produits).
- #49 Débute une instruction `if` qui affiche le bouton Suivant. Si le dernier produit n'a pas encore été affiché, le bouton Suivant est affiché. Le bouton Suivant est nommé Products (pour Produits). Du fait que les deux boutons Suivant et Précédent sont nommés Products, le script de l'application teste la valeur du bouton Products pour décider quels produits afficher dans la page des produits.
- #55 Affiche les informations en bas de page.

Modification de `catalog_product_page.inc` pour l'application orientée objet

Le Listing 14.4 affiche la page des produits dans l'application écrite en code procédural. Le fichier nécessite quelques modifications pour fonctionner avec le code orienté objet.

Les lignes 24 à 33 affichent les informations produits sur chaque ligne. Dans le Listing 14.4, les informations sont affichées à partir d'un tableau nommé `$products`. Ce tableau est construit à partir des données extraites de la base, dans le script `Catalog.php` (Listing 14.5). Pour l'application orientée objet, ces lignes doivent être modifiées de façon à afficher les données à partir d'un objet. L'objet est créé dans le script `Catalog-oo.php` (Listing 14.8).

Pour utiliser le fichier avec l'application orientée objet, remplacez les lignes 24 à 33 du Listing 14.4 par les lignes suivantes, et enregistrez le fichier sous le nom `catalog_product_page-oo.inc`.

```
echo "<tr>";
echo "<td align='right'>{$all[$i]->catalog_number}</td>\n";
echo "<td>{$all[$i]->name}</td>\n";
echo "<td>{$all[$i]->description}</td>\n";
echo "<td>{$all[$i]->price} /lb</td>\n";
echo "<td><img src='images/({$all[$i]->pix})'></td>\n";
echo "</tr>";
```

Affichage des pages Web du catalogue

Deux fichiers sont utilisés conjointement pour afficher chaque page Web. Les fichiers `catalog_index_page.inc` et `fields_index_page.inc` affichent la page des catégories ; les fichiers `fields_products_page.inc` et `catalog_product_page.inc` (ou `catalog_product_page-oo.inc`) affichent la page des produits.

La page d'index est affichée par le code suivant :

```
include("fields_index_page.inc");
include("catalog_index_page.inc");
```

Dans l'application procédurale, la page produits est affichée à l'aide du code suivant :

```
include("fields_products_page.inc");
include("catalog_product_page.inc");
```

Dans l'application orientée objet, elle l'est avec celui-ci :

```
include("fields_products_page.inc");
include("catalog_product_page-oo.inc");
```

Le script procédural et le script orienté objet diffèrent légèrement mais la démarche générale reste la même. Un fichier contient le code pour l'aspect, approprié à tout catalogue. Les informations spécifiques au catalogue sont fournies dans un fichier séparé, inclus lors de l'inclusion du fichier régissant l'aspect.

Création de l'application de catalogue en ligne : démarche procédurale

L'application catalogue dispose d'un script principal. Il est organisé en deux sections : l'une affiche la page d'index, l'autre affiche la page produits. La section de la page d'index s'affiche lors du démarrage de l'application, avant que quelque bouton ne soit déclenché.

Lorsque l'utilisateur clique sur un bouton, le script affiche une page Web dont le contenu dépend du bouton déclenché. Voici une vue d'ensemble de la structure du script :

si (Bouton)

Le bouton Product a été cliqué,
 1 Tester si une catégorie a été sélectionnée.
 Si non, afficher de nouveau la page d'index.
 2 Afficher la page produits.

Le bouton Product n'a pas été cliqué,
 1 Afficher la page d'index.

Le Listing 14.5 présente le code du catalogue en ligne.

Listing 14.5 : Code du catalogue en ligne.

```
<?php
/* Programme : Catalog.php
 * Description :   Affiche un catalogue de produits. Affiche deux
 *                pages différentes : une page d'index qui présente
 *                les catégories et une page produits qui est affichée
 *                lorsque le client choisit une catégorie.
 */
include("functions_main.inc");                                #8
$N_per_page = 2;                                             #9
if(isset($_POST['Products']))                                #10
{
    if(!isset($_POST['interest']))                            #12
    {
        header("location: Catalog.php");
        exit();
    }
    else                                                        #17
    {
        if(isset($_POST['n_end']))                            #19
        {
```



```

        if($_POST['Products'] == "Préc.") #21
        {
            $n_start = $_POST['n_end'] - ($n_per_page);
        }
        else #25
        {
            $n_start = $_POST['n_end'] + 1;
        }
    }
    else #30
    {
        $n_start = 1;
    }
    $n_end = $n_start + $n_per_page - 1; #35
    $connect = connect_to_db("Vars.inc"); #36
    $query_food = "SELECT * FROM Food WHERE
                    type='$_POST[interest]' ORDER BY name";
    $result = mysqli_query($connect,$query_food)
        or die ("query_food: ".mysqli_error($connect)); #39
    $n=1;
    while($row = mysqli_fetch_assoc($result)) #41
    {
        foreach($row as $field => $value) #43
        {
            $products[$n][$field]=$value;
        }
        $n++;
    }
    $n_products = sizeof($products); #49
    if($n_end > $n_products)
    {
        $n_end = $n_products;
    }
    include("fields_products_page.inc"); #54
    include("catalog_product_page.inc");
}
}
else #58
{
    $cxn = connect_to_db("Vars.inc"); #60
    $query = "SELECT DISTINCT category,type FROM Food
              ORDER BY category,type";
    $result = mysqli_query($cxn,$query)
        or die ("Exécution de la requête impossible
                ".mysqli_error($cxn)); #65
    while($row = mysqli_fetch_array($result)) #66
    {
        $food_categories[$row['category']][]=$row['type'];
    }
}

```

```

}
include("fields_index_page.inc");
include("catalog_index_page.inc");
}
?>
#70

```

Dans les explications suivantes, les numéros renvoient aux numéros de lignes du Listing 14.5 :

- #8** Inclut un fichier qui contient la fonction `Connect_to_db` – fonction utilisée tout au long du livre. Dans ce script, la fonction est utilisée à la ligne 36 et à la ligne 60.
- #9** Définit `$n_per_page`, le nombre de produits à afficher sur une même page. `$n_per_page` est ici réglé sur 2, valeur inhabituellement faible. Il est plus courant de choisir la valeur 10, cela dépend en fait du nombre de produits que l'on peut afficher sur une même page. Vous pouvez définir ce nombre sur toute valeur entière.
- #10** Débute une instruction `if` qui s'exécute lorsque le client clique sur le bouton d'envoi nommé `Products`.
- #12** Débute une instruction `if` qui s'exécute lorsque le client n'a *pas* sélectionné de catégorie dans le formulaire. La page des catégories est de nouveau affichée.
- #17** Débute une instruction `else` qui s'exécute lorsque le client a choisi une catégorie dans le formulaire. La page produits est affichée.
- #19** Débute une instruction `if` qui s'exécute si la page affichée n'est pas la première page. Vérifie quel bouton a été enfoncé. Si c'est le bouton Précédent qui a été enfoncé, `$n_start` est réglé sur le début de la page précédente. Si le bouton Suivant a été enfoncé, `$n_start` est réglé sur le produit après le dernier produit affiché sur la dernière page.
- #30** Débute un bloc `else` qui s'exécute au tout premier affichage de la page produits. Règle `n_start` sur 1.
- #35** Règle `$n_end` sur `$n_start` plus le nombre de produits à afficher dans la page, moins un.

- #36** Les lignes 36 à 39 construisent et exécutent une requête qui lit les informations concernant tous les articles de la catégorie sélectionnée.
- #40** Les lignes 40 à 48 construisent un tableau nommé `$products`, qui contient les informations sur tous les articles sélectionnés.
- #49** Règle `$n_products` sur le nombre de produits dans la catégorie.
- #50** Débute une instruction `if` qui vérifie que `$n_end` est supérieur au nombre de produits. Lorsque `$n_end` est supérieur au nombre total de produits, `$n_end` est réglé sur le dernier produit.
- #54** Les lignes 54 à 55 affichent la page des produits.
- #58** Débute un bloc `else` qui s'exécute si le bouton Product n'a pas été cliqué. Aucun bouton ou le bouton Autre catégorie a été cliqué. Ce bloc affiche la page des catégories.
- #60** Les lignes 60 à 65 construisent et exécutent une requête qui extrait toutes les catégories de la base de données.
- #66** Les lignes 66 à 69 créent un tableau des articles d'alimentation.
- #70** Les lignes 70 à 71 affichent la page des catégories.

Le script appelle une fonction pour se connecter à la base de données. C'est le fichier `functions_main.inc`, appelé à la ligne 8, qui contient le code de cette fonction. Le listing 14-6 présente le code de la fonction.

Listing 14.6 : La fonction `Connect_to_db`.

```
<?php
/* Fonction: Connect_to_db
 * Description :      Connecte à une base MySQL. Le nom d'un
 *                  fichier contenant les variables de la base
 *                  est passé à la fonction.
 */
function Connect_to_db($filename)
{
    include($filename);
```

```

$cxn = mysqli_connect($host, $user,$passwd)
        or die ("Connexion impossible au serveur");
$db = mysqli_select_db($cxn,$database)
        or die ("Connexion impossible à la base de données.");
return $cxn;
}
?>

```

Cette fonction lit les informations dont elle a besoin dans un fichier séparé. Le nom de fichier est passé lors de l'appel de la fonction. Le contenu du fichier est de cet ordre :

```

<?php
$host = "localhost";
$user = "admin";
$passwd = "admin";
$database = "FoodCatalog";
?>

```

Pour des raisons de sécurité, stockez ce fichier en dehors de votre espace Web. Après avoir défini le paramètre `include_path` du fichier `php.ini` sur cet emplacement en dehors de votre espace Web, les fichiers que vous y stockerez pourront être inclus sans indiquer de chemin d'accès.

Création de l'application de catalogue en ligne : approche orientée objet

La programmation orientée objet nécessite la création et l'emploi d'objets pour mettre en place les fonctionnalités de l'application. Identifiez en premier lieu les objets nécessaires à l'application. Écrivez ensuite les classes qui définissent les objets, y compris les méthodes dont l'application a besoin. Une fois les objets prêts, on écrit le script qui crée et exploite les objets.

Développement des objets

L'application de catalogue en ligne affiche des informations extraites d'une base de données. Deux types de données sont issues de cette base : la liste des catégories ou la liste des détails sur les produits d'une catégorie particulière. Pour afficher un catalogue, l'application

a besoin d'un seul objet : un objet `Catalog`. Les détails de l'objet `Catalog` sont étudiés dans les sections suivantes.

Écriture de la classe `Catalog`

La définition de l'objet `Catalog` est codée dans la classe `Catalog`. Cette classe est une classe générique, exploitable pour gérer l'affichage de tout catalogue, pas seulement le catalogue d'alimentation présenté dans ce chapitre.

La classe `Catalog` gère la connexion à la base de données où sont stockées les informations produits, et retourne ou affiche si nécessaire les informations produits. La classe `Catalog` possède cinq propriétés, un constructeur et cinq méthodes.

Les propriétés

Les propriétés stockent les informations nécessaires pour définir et gérer le formulaire et ses données.

```
private $cxn;  
private $catalog_name;  
private $host;  
private $user;  
private $password;
```

La première propriété est la connexion au serveur de base de données MySQL. La deuxième propriété est le nom du catalogue, qui est aussi le nom de la base de données où sont stockées les informations produits. Les propriétés restantes fournissent les informations nécessaires pour accéder à la base de données.

Le code

Le Listing 14.7 présente le code complet de la classe `Catalog`. Chaque méthode est étudiée en détail à la suite du listing. À noter, les numéros de lignes à la fin de certaines lignes de code. Les explications des six sous-sections à la suite du Listing 14.7 renvoient aux numéros de lignes de ce listing.

Listing 14.7 : Code de la classe Catalog.

```

<?php
/* Classe : Catalog
 * Description : Classe contenant les informations produits.
 */
class Catalog
{
    private $cxn;
    private $catalog_name;
    private $host;
    private $user;
    private $password;

    function __construct($filename)
    {
        if(is_string($filename))
        {
            include("$filename");
        }
        else
        {
            throw new Exception("Le paramètre n'est pas un nom de fichier");
        }
        $this->cxn = new mysqli($host,$user,$passwd);           #23
        if(mysqli_connect_errno())                             #24
        {
            throw new Exception("Base de données inaccessible.
                                Réessayez ultérieurement.");
            exit();
        }
        $this->host = $host;
        $this->user = $user;
        $this->password = $passwd;
    }

    function selectCatalog($database)
    {
        $db = $this->cxn->select_db($database);                 #37
        if(mysqli_errno($this->cxn))                             #38
        {
            if(mysqli_errno($this->cxn) == 1049)                 #40
            {
                throw new Exception("$database n'existe pas");
                exit();
            }
            else
            {

```

```

        throw new Exception("Base de données inaccessible.
                               Réessayez ultérieurement.");
        exit();
    }
}
$this->catalog_name = $database;
}

function getCategoriesAndTypes()
{
    $sql = "SELECT DISTINCT category,type FROM Food
            ORDER BY category,type";
    if(!$result = $this->cxn->query($sql))
    {
        throw new Exception(mysqli_error($this->cxn));
        exit();
    }
    while($row=$result->fetch_array())
    {
        $array_cat_type[$row['category']][]=$row['type'];
    }
    return $array_cat_type;
}

function getAllofType($type)
{
    if(is_string($type)) #73
    {
        $sql = "SELECT * FROM Food WHERE type='$type'
                ORDER BY name";
    }
    else
    {
        throw new Exception("$type n'est pas une chaîne.");
        exit();
    }
    if(!$result = $this->cxn->query($sql)) #83
    {
        throw new Exception(mysqli_error($this->cxn));
        exit();
    }
    $n=1;
    while($row=$result->fetch_object()) #89
    {
        $array_all[$n] = $row;
        $n++;
    }
    return $array_all;
}

```

```

}

function displayCategories()
{
    $food_categories = $this->getCategoriesAndTypes();
    include("fields_index_page.inc");
    include("catalog_index_page.inc");
}

function displayAllofType($type,$page)
{
    if(is_string($type))
    {
        $all = $this->getAllofType($type);
    }
    else
    {
        throw new Exception("$type n'est pas une chaîne.");
        exit();
    }
    if(is_int($page))
    {
        $n_per_page = $page;
    }
    else
    {
        throw new Exception("$page n'est pas un entier.");
        exit();
    }
    $n_products = sizeof($all);
    if(isset($_POST['n_end'])) #125
    {
        if($_POST['Products'] == "Préc.") #127
        {
            $n_start = $_POST['n_end'] - ($n_per_page);
        } #131
        else
        {
            $n_start = $_POST['n_end'] + 1;
        }
    }
    else #136
    {
        $n_start = 1;
    }
    $n_end = $n_start + $n_per_page - 1; #140
    if($n_end >= $n_products) #141
    {

```

```

        $n_end = $n_products;
    }
    include("fields_products_page.inc");
    include("catalog_product_page-oo.inc");
}
?>

```

Le constructeur

Le constructeur crée une connexion au serveur MySQL. Il obtient les nom d'hôte, nom de compte et mot de passe MySQL à partir d'un fichier ; le nom de fichier est passé au moment où l'objet est instancié. Le fichier peut être stocké dans le répertoire include, défini dans le fichier `php.ini`. Le constructeur vérifie si l'information passée à la classe est du bon type de données. Si c'est le cas, il inclut le fichier ; sinon, il jette une exception.

Pour la connexion à MySQL, ce sont les objets fournis par l'extension MySQL Improved (mysqli) qui sont utilisés. La ligne 23 crée une connexion qui est stockée dans la propriété `$cxn`. La ligne 24 commence une instruction `if` qui vérifie si la connexion a abouti. Si elle a échoué, une exception est jetée.

Le nom d'hôte, le nom de compte et le mot de passe sont stockés dans des propriétés.

Il est possible de créer un objet `Catalog` avec l'instruction suivante :

```
$food_catalog = new Catalog("Vars.inc");
```

selectCatalog

La méthode `selectCatalog` définit la base de données à utiliser pour l'accès MySQL. Le nom de la base de données est passé. La connexion stockée est utilisée pour définir la base de données active.

#37 Définit la base de données.

#38 Débute un bloc `if` qui s'exécute lorsque la sélection de la base de données n'a pas réussi. Le bloc vérifie si l'erreur est une 1049, qui signale que la base de données n'existe pas. Si

l'erreur est une 1049, une exception est jetée pour informer l'utilisateur que le nom de la base de données fourni ne correspond pas à une base de données valide. Si le numéro d'erreur n'est pas 1049, l'exception créée génère un message d'erreur plus générique.

Cette méthode est exploitable à tout moment pour changer de base de données active.

getCategoryAndTypes

La méthode `getCategoryAndTypes` lit les catégories et types, et les retourne dans un tableau multidimensionnel. Elle crée et exécute la requête `SELECT` appropriée. Une instruction `while` construit un tableau multidimensionnel à partir du résultat. La méthode retourne le tableau.

getAllofType

La méthode `getAllofType` lit les informations de tous les produits d'une sous-catégorie donnée (type). Le type est passé à la méthode.

- #73** Débute un bloc `if/else` qui vérifie si le nom du type passé est une chaîne. Si c'est le cas, le bloc `if` crée une requête `SELECT` avec le type passé. Si ce n'est pas le cas, le bloc `else` jette une exception.
- #83** Exécute la requête. Si la requête échoue, une nouvelle exception est jetée.
- #89** Débute une boucle `while` qui construit un tableau d'objets à partir du résultat. Chaque produit (ligne) est un objet. Les propriétés d'objet sont les noms de champs, ils contiennent les informations dans la cellule.

displayCategories

La méthode `displayCategories` affiche la page Web des catégories. Un tableau multidimensionnel des catégories de produits est créé à l'aide de la méthode `getCategoriesAndTypes`. La méthode affiche la page des catégories en incluant deux fichiers : un fichier contient les variables supplémentaires nécessaires au catalogue d'alimentation, l'autre fichier contient le code qui définit l'aspect de la page Web.

Ces deux fichiers sont étudiés à la section "Création des pages Web du catalogue", plus haut dans ce chapitre.

displayAllofType

La méthode `displayAllofType` affiche tous les produits d'un type donné. La méthode obtient les informations produits à l'aide de la méthode `getAllofType`. La méthode affiche les produits en incluant deux fichiers séparés : l'un fournit certaines variables supplémentaires, l'autre fournit le code pour définir l'aspect de la page Web. Les deux fichiers sont étudiés à la section "Création des pages Web du catalogue", plus haut dans ce chapitre.

- #125 Débute une instruction `if` qui s'exécute si la page affichée n'est *pas* la première page. Donne à `$n_start` une valeur basée sur le bouton enfoncé, Précédent ou Suivant.
- #127 Débute un bloc `if` qui s'exécute si le bouton Précédent a été cliqué. `$n_start` est défini pour revenir au début de la page précédente.
- #131 Débute un bloc `else` qui s'exécute si le bouton Précédent n'a pas été cliqué. Le bouton Suivant a été enfoncé. `$n_start` est réglé sur la valeur de produit après le dernier produit affiché sur la dernière page.
- #136 Débute un bloc `else` qui s'exécute au tout premier affichage de la page produits. Règle `n_start` sur 1.
- #140 Règle `$n_end` sur `$n_start` plus le nombre de produits à afficher dans la page, moins un.
- #141 Débute une instruction `if` qui vérifie que `$n_end` est supérieur au nombre de produits. Lorsque `$n_end` est supérieur au nombre total de produits, `$n_end` est réglé sur le dernier produit.
- #145 Les lignes 145 à 146 affichent la page des produits.

Écriture du script de l'application catalogue

Une fois écrit tout le code de classe nécessaire à l'application de catalogue en ligne, on écrit le script qui crée et utilise les objets

fournissant la fonctionnalité de l'application. Voici la structure générale du script de l'application :

si (Bouton)

Le bouton Products a été enfoncé,

- 1 Tester si une catégorie a été sélectionnée. Si non, afficher de nouveau la page d'index.
- 2 Afficher la page produits.

Le bouton Products n'a pas été enfoncé,

- 1 Afficher la page d'index.

La seconde possibilité (le bouton Products n'a pas été enfoncé) signale soit que c'est la première fois que la page est affichée, soit que l'utilisateur a cliqué le bouton Autre catégorie.

Le programme de l'application crée les objets et exploite leurs méthodes pour réaliser les tâches décrites dans la section précédente (qui décrit la structure). Le Listing 14.8 présente le script de l'application.

Listing 14.8 : Le script de l'application d'identification

```
<?php
/* Programme : Catalog-oo.php
 * Description :   Affiche un catalogue de produits. Affiche deux
                  pages différentes : une page d'index qui présente
                  les catégories et une page produits qui est affichée
                  lorsque le client choisit une catégorie.
 */
require_once("Catalog.class");
if(isset($_POST['Products']))                                #9
{
    if(!isset($_POST['interest']))                          #11
    {
        header("location: Catalog-oo.php");
        exit();
    }
    else                                                    #16
    {
        try
        {
            $foodcat = new Catalog("Vars.inc");              #20
            $foodcat->selectCatalog("FoodCatalog");          #21
            $foodcat->displayAllofType($_POST['interest'],2);
```

```

    }
    catch(Exception $e)
    {
        echo $e->getMessage();
        exit();
    }
}
else
{
    try
    {
        $foodcat = new Catalog("Vars.inc");
        $foodcat->selectCatalog("FoodCatalog");
        $foodcat->displayCategories();
    }
    catch(Exception $e)
    {
        echo $e->getMessage();
        exit();
    }
}
?>

```

#31
#35
#36
#37

Dans ce script, de nombreuses instructions sont dans des blocs `try/catch`. Si une méthode jette une exception et que celle-ci n'est pas interceptée, l'erreur fatale suivante se produit :

```

Fatal error: Uncaught exception 'Exception'
with message 'Base indisponible.'
in c:\Database.class:56

```

Il faut donc intercepter toute exception jetée par une méthode, dans la méthode elle-même ou dans le script qui l'utilise.

Les explications suivantes renvoient aux numéros de lignes du Listing 14.8 :

#9 Débute une instruction `if` qui s'exécute si un bouton nommé `Products` a été enfoncé.

#11 Débute un bloc `if` qui s'exécute si le client n'a *pas* choisi une catégorie de produits. Le client est renvoyé à la page Web des catégories.

- #16** Débute un bloc `else` qui s'exécute si l'utilisateur a sélectionné une catégorie.
- #20** Les lignes 20 à 22 affichent la page Web des produits. La ligne 20 crée un nouvel objet `Catalog` ; la ligne 21 sélectionne le catalogue à employer ; la ligne 22 affiche les produits.
- #31** Débute un bloc `else` qui s'exécute si l'utilisateur n'a pas cliqué sur un bouton nommé `Products`. Ce bloc affiche la page des catégories. Il le fait lors du premier appel au script, lorsque l'utilisateur n'a cliqué sur aucun bouton. Il est également exécuté lorsque l'utilisateur clique sur le bouton `Autre catégorie` dans la page produits car ce bouton n'est pas nommé `Products`.
- #35** Les lignes 35 à 37 affichent la page Web des catégories. La ligne 35 crée un nouvel objet `Catalog` ; la ligne 36 sélectionne le catalogue à employer ; la ligne 37 affiche les catégories.

Extension de la classe Catalog

La classe `Catalog` de ce chapitre fournit les fonctions essentielles à notre exemple d'application. Cependant, l'on voit rapidement où des méthodes supplémentaires seront bienvenues, peut-être même obligatoires. Par exemple, la classe a besoin d'une méthode capable de lire individuellement les champs d'un article particulier, de façon à pouvoir lire le prix des oignons rouges ou des pommes reinettes. Voici un exemple d'appel d'une telle méthode :

```
getPrice("oignons rouges")
```

Parmi les autres méthodes utiles : une pour ajouter un produit au catalogue, une autre pour supprimer des produits du catalogue.

L'un des bénéfices de la programmation orientée objet est la facilité d'ajout de nouvelles fonctions aux applications.

Chapitre 15

Application de panier d'achat

Dans ce chapitre :

- Conception de l'application de panier d'achat.
- Création de la base de données de commande en ligne.
- Écriture du code procédural de l'application de panier d'achat.
- Développement et emploi des objets pour programmer l'application de panier d'achat.

Internet fournit d'infinies occasions d'acheter des choses. Ce chapitre présente un système de commande en ligne qui affiche les informations sur les produits d'un catalogue et fournit un panier pour acheter des articles. Au Chapitre 14, vous avez étudié un catalogue en ligne. Cette application de catalogue en ligne présente les informations produits aux clients potentiels. Dans ce chapitre, l'application de panier d'achat apporte la capacité d'acheter les produits du catalogue. Le panier d'achat n'est pas une application autonome. Il a besoin du catalogue pour fournir les articles que le client est susceptible d'y placer.

Conception de l'application de panier d'achat

Il existe de nombreuses façons pour implémenter un panier d'achat. Votre première tâche est de décider comment mettre en place le vôtre.

Décisions élémentaires de conception d'application

Vous devez opérer certaines décisions de conception fondamentales touchant à la programmation avant de l'interface utilisateur. Voici quelques considérations élémentaires :

- ✓ **Identification client** : De nombreuses boutiques obligent le client à s'inscrire et/ou s'identifier avant de pouvoir acheter des produits. L'inscription du client fournit à la boutique des informations sur lui, telles que numéro de téléphone et adresse électronique. L'obligation d'identification autorise en outre des fonctions impossibles à fournir sans ce processus d'identification. Par exemple, vous ne pouvez proposer au client de suivre ses commandes sans l'identifier au préalable. Si le client ne devait pas s'identifier, rien n'empêcherait d'autres clients de visionner les commandes des autres.

Cela dit, de nombreux clients fuient les inscriptions. Certains clients supposent que les informations fournies leur vaudront des appels marketing ou e-mails indésirables. D'autres n'ont pas la patience d'accomplir le processus, il passeront donc à un autre site qui leur semble plus facile. En résumé, l'obligation d'identification coûte quelques ventes à la boutique.

L'application présentée dans ce chapitre ne réclame pas l'identification du client. N'importe qui peut acheter les produits. Les Chapitres 3 et 4 ont présenté des applications de login, que vous pourrez ajouter à cette application si vous désirez identifier les clients.

- ✓ **Méthodes d'achat** : De quelle manière le client peut acheter les produits ? La méthode la plus facile est d'envoyer les informations de commande via e-mail au service des ventes, puis de facturer le client. Il est aussi possible de demander au client de payer avant de lui expédier les produits. Par ailleurs, la plupart des sites Web acceptent le paiement directement sur le site. Les paiements se font majoritairement par carte bancaire. Certains sites acceptent les paiements PayPal, en plus ou au lieu du paiement par carte bancaire. PayPal est un site Web qui propose aux internautes des comptes utilisables pour envoyer ou recevoir de l'argent via Internet. Par exemple, un compte PayPal est exploitable pour payer pour des achats sur eBay. Outre la possibilité d'ouvrir un compte, le

site Web de PayPal propose des outils que vous pouvez utiliser sur votre site pour accepter facilement le paiement par PayPal. Voir www.paypal.com.

L'application de ce chapitre n'accepte que les cartes bancaires.

- ✓ **Traitement de carte bancaire :** L'acceptation de paiements par carte soulève des problèmes de sécurité. Puisque le client va vous transmettre un numéro de carte bancaire, vous devez implémenter la sécurité SSL (Secure Socket Layers), présentée au Chapitre 11. Si vous stockez les numéros de carte bancaire, la sécurité devra être renforcée. Le stockage des numéros de carte bancaire permet au client de réaliser des achats plus rapides (puisque ses informations de carte sont dans un fichier) mais augmente les possibilités pour les pirates de dérober ces informations. Certains clients ne veulent pas que l'on stocke leur informations de carte sur un site Web. Une possible solution, exploitée par certaines boutiques en ligne, est de permettre au client de décider si ses informations de carte bancaire seront stockées. •

L'application de ce chapitre n'enregistre pas les informations de carte bancaire. Les données de la carte sont reçues, utilisées, puis détruites, elles ne sont pas stockées dans la base de données.

- ✓ **Frais de port :** L'envoi des produits aux clients vous coûte de l'argent. La solution la plus simple est de s'en tenir à un tarif unique de port. L'ajout de ce montant au total de la commande est une opération simple de programmation. La solution la plus difficile est d'essayer de calculer le véritable coût d'expédition, qui dépend du mode de livraison choisi par le client et de la distance entre votre code postal et celui du client. Les clients apprécieront un coût établi de façon précise mais cette programmation demande plus de temps et d'effort.

L'application de ce chapitre facture des frais de port qui consistent en un forfait par article.

- ✓ **Panier d'achat :** Il existe divers mécanismes pour stocker le panier pendant que le client poursuit ses achats, avant validation de la commande. Le client doit pouvoir ajouter et supprimer des articles du panier jusqu'à être satisfait de son

contenu et passer la commande. Voici les techniques les plus courantes pour stocker le panier :

- **Table de base de données** : Plus sûr mais plus de travail.
- **Cookies** : Il se peut que le client ait désactivé l'emploi des cookies.
- **Variables de session** : Moins sécurisé sur un serveur partagé.
- **Fichier texte** : Facile mais moins sécurisé.

D'autres méthodes moins communes sont parfois exploitées.

L'application de ce chapitre stocke le panier de deux manières. Le programme procédural stocke les articles du panier dans la base de données MySQL. Le programme orienté objet stocke les articles du panier dans une variable de session.

Conception de la fonctionnalité de l'application

La fonction élémentaire de l'application de panier d'achat est de recueillir l'information nécessaire pour mener à terme les achats du client. L'application doit :

- ✓ **Afficher les produits, pour que le client puisse choisir les articles à acheter.** Cette étape est réalisée par l'application de catalogue en ligne, étudiée au Chapitre 14. Pour permettre l'achat en ligne, il faut cependant ajouter quelques fonctions au catalogue. Ces fonctions supplémentaires font l'objet de ce chapitre.
- ✓ **Garder trace des produits sélectionnés par le client.** À tout moment, le client doit pouvoir visionner ce qu'il a déjà sélectionné. Il doit également pouvoir supprimer des articles.
- ✓ **Recueillir l'information nécessaire à l'expédition des articles.** Il vous faut donc le nom et l'adresse du client. Un numéro de téléphone est aussi bienvenu en cas de problème. Une adresse électronique sera utile à la communication. L'application doit aussi collecter les informations nécessaires au calcul des frais de port.

- ✓ **Recueillir l'information nécessaire pour facturer le client.** L'application collecte les informations de carte bancaire, une adresse de facturation, et le nom associé à la carte bancaire. Dans notre exemple, les informations d'expédition et de facturation sont supposées être identiques. Cela a permis de simplifier l'exemple. Cela dit, dans un cas véritable, vous ne pouvez faire cette supposition.
- ✓ **Fournir un récapitulatif au client.** Le client doit pouvoir visionner les informations qu'il a entrées, à n'importe quelle étape du processus, et avoir la possibilité de les corriger.

Création de la base de données pour le panier d'achat

La base de données pour le panier stocke les informations sur les commandes. Elle stocke des données générales sur la commande, comme les nom et adresse du client, ainsi que les articles sélectionnés. Un autre détail important est la date de la commande. Il faut aussi stocker les informations produits qui apparaissent dans le catalogue en ligne. L'application de ce chapitre présente des produits du catalogue d'alimentation étudié au Chapitre 14.

Conception de la base de données pour le panier d'achat

Notre exemple utilise une base nommée `OnlineOrders`. Cette base contient deux tables. L'une stocke l'information générale sur la commande : nom et adresse, numéro de commande, etc. L'autre stocke une rangée pour chaque article de la commande, liée à la première table par le numéro de commande.

De plus, puisque l'application doit afficher les produits, elle doit accéder à la base de données du catalogue.

Conception de la table `Customer_Order`

La table `Customer_Order` (commande client) contient des informations liées à la commande dans sa globalité (Tableau 15.1).



Il est interdit de nommer les tables avec des mots réservés de MySQL. Cette table aurait pu être nommée `Order` (commande) mais c'est un mot réservé de MySQL. Le nom de table `Order` génère une erreur de syntaxe MySQL. Vous pourrez passer des heures à relire votre requête en cherchant où se trouve le problème. Pour la liste des mots réservés, visitez <http://dev.mysql.com/doc/mysql/en/reserved-words.html>.

Tableau 15.1 : Table de base de données : `Customer_Order`.

Nom de variable	Type	Description
<code>order_number</code>	<code>INT(6)</code>	Entier attribué par <code>AUTO_INCREMENT</code> (clé primaire).
<code>order_date</code>	<code>DATE</code>	Date à laquelle la commande a été ajoutée à la table.
<code>shipping_fee</code>	<code>DECIMAL(9,2)</code>	Coût total de livraison de la commande.
<code>sales_tax</code>	<code>DECIMAL(9,2)</code>	Total TVA de la commande.
<code>submitted</code>	<code>ENUM('yes', 'no')</code>	Statut de la commande.
<code>ship_name</code>	<code>VARCHAR(50)</code>	Expédié à : nom.
<code>ship_street</code>	<code>VARCHAR(50)</code>	Adresse (rue).
<code>ship_city</code>	<code>VARCHAR(50)</code>	Ville où la commande sera expédiée.
<code>ship_state</code>	<code>CHAR(2)</code>	Code d'état (deux lettres, pour les USA).
<code>ship_zip</code>	<code>CHAR(10)</code>	Code postal. (Cinq chiffres ou code postal+4.)
<code>email</code>	<code>CHAR(50)</code>	Adresse électronique du client.
<code>phone</code>	<code>CHAR(20)</code>	Numéro de téléphone du client.

Ici, le numéro de commande est un entier attribué séquentiellement par MySQL. Il serait possible d'utiliser un numéro de commande avec des nombres et/ou lettres significatifs, à base de dates ou codes des services de l'entreprise.

Les frais de port et la TVA sont stockés dans la commande. Ils peuvent être certes calculés, mais le taux peut changer ultérieurement. Lorsque vous consultez une commande, vous voulez savoir quelle TVA a été appliquée au moment de la commande.

Conception de la table *Order_Item*

La table *Order_Item* (article de commande) contient des informations sur chaque article de la commande (Tableau 15.2).

Tableau 15.2 : Table de base de données : *Order_Item*.

Nom de variable	Type	Description
order_number	INT(6)	Lien vers la table <i>Customer_Order</i> (clé primaire 1).
item_number	INT(4)	Nombre attribué à chaque article (clé primaire 2).
catalog_number	INT(8)	Nombre affecté au produit dans le catalogue.
quantity	DECIMAL(7,2)	Montant commandé.
price	DECIMAL(9,2)	Prix de l'article.

La table *Order_Item* comporte cinq champs. Les deux premiers champs constituent ensemble la clé primaire. Le prix est stocké, de façon que le véritable prix payé pour cet article puisse être retrouvé plus tard, même si le prix a changé entre temps.

Conception de la table *Food*

L'application utilise la table *Food* (aliments) du catalogue en ligne étudié au Chapitre 14. (En particulier, voir le Tableau 14.1.) L'application aurait pu accéder à la table de cette base de données. Cependant, j'ai préféré ajouter la table *Food* à la base *OnlineOrders* (étudiée dans ce chapitre) pour simplifier l'exemple.

Création de la base de données pour le panier d'achat

Vous créerez la base MySQL à l'aide d'une des méthodes étudiées au Chapitre 10. L'instruction SQL suivante crée cette base de données :

```
CREATE DATABASE OnlineOrders;
```

Voici les instructions SQL pour créer les tables :

```

CREATE TABLE Customer_Order (
  order_number INT(6) NOT NULL AUTO_INCREMENT,
  order_date DATE NOT NULL,
  shipping_fee DECIMAL(9,2),
  sales_tax DECIMAL(9,2),
  submitted ENUM("yes", 'no'),
  ship_name VARCHAR(50),
  ship_street VARCHAR(50),
  ship_city VARCHAR(50),
  ship_state VARCHAR(2),
  ship_zip VARCHAR(10),
  email VARCHAR(50),
  phone VARCHAR(20),
  PRIMARY KEY(order_number) );

```

Tous ces champs sont requis pour achever le traitement de la commande. Cependant, seuls les deux premiers champs sont déclarés NOT NULL. Lorsque l'application insère pour la première fois la commande dans la base de données, des valeurs sont insérées uniquement dans les deux premiers champs. À ce stade, les champs restants sont vides ; les valeurs de ces champs seront ajoutées plus tard. Par conséquent, ces champs doivent être autorisés à être vides. Le script de l'application PHP doit assurer que les champs contiennent les informations appropriées.

```

CREATE TABLE Order_Item (
  order_number INT(6) NOT NULL,
  item_number INT(5) NOT NULL,
  catalog_number INT(6) NOT NULL,
  quantity DECIMAL(7,2) NOT NULL,
  price DECIMAL(9,2) NOT NULL,
  PRIMARY KEY(order_number,item_number) );

```

```

CREATE TABLE Food (
  catalog_number INT(6) NOT NULL AUTO_INCREMENT,
  name VARCHAR(20) NOT NULL,
  added_date DATE NOT NULL,
  category VARCHAR(20) NOT NULL,
  type VARCHAR(20) NOT NULL,
  description VARCHAR(255) NOT NULL,
  price DECIMAL(7,2) NOT NULL,
  pix VARCHAR(20) NOT NULL DEFAULT "Missing.jpg",
  PRIMARY KEY(catalog_number) );

```

Accès à la base de données du panier d'achat

PHP fournit des fonctions MySQL pour accéder à la base de données à partir du script PHP. Les fonctions MySQL reçoivent les informations nécessaires pour accéder à la base de données, tel qu'un nom de compte MySQL et un mot de passe. Ces nom de compte et mot de passe ne sont pas apparentés à tout autre nom de compte ou mot de passe que vous possédez, par exemple un mot de passe pour vous identifier sur le système.

PHP fournit deux jeux de fonctions MySQL : `mysql` et `mysqli`. Les fonctions `mysqli` servent à exploiter les caractéristiques ajoutées à MySQL version 4.1. Les fonctions `mysql` restent utilisables avec la 4.1 mais vous ne bénéficierez pas des caractéristiques récentes. L'extension `mysql` ou `mysqli` est activée lors de l'installation de PHP. Pour employer les fonctions `mysqli`, vous devez disposer de PHP 5.

MySQL 4.1 étant actuellement la version recommandée sur le site Web de MySQL, j'utilise les fonctions MySQL Improved (`mysqli`) dans ce chapitre. Les fonctions procédurales sont employées pour la construction des programmes procéduraux. Pour la construction des programmes orientés objet, ce sont les classes orientées objet qui le sont.

Si vous utilisez PHP 4 ou pour quelque autre raison désirez employer les fonctions `mysql` au lieu des fonctions `mysqli`, il faudra apporter quelques modifications de syntaxe. Pour plus d'informations sur les fonctions, voir le manuel PHP sur www.php.net/manual/en/ref.mysqli.php et www.php.net/manual/en/ref.mysql.php.

Ajout de données à la base de données du panier d'achat

La table `Food` contient les informations sur les produits. Vous devez ajouter vous-même ces données dans la base, notre exemple d'application ne le permet pas. Pour ajouter des articles à ce catalogue d'alimentation, employez le client MySQL installé avec MySQL, ou toute application d'administration MySQL (comme `phpmyadmin` [www.phpmyadmin.net] ou MySQL Administrator, téléchargeable sur www.mysql.com/products/administrator/index.html). Reste aussi la possibilité d'écrire votre propre application en PHP.



Les informations de commande sont ajoutées à la base de données par l'application Panier d'achat. Lorsque les clients valident leur commande, cette dernière, ainsi que les informations sur les articles, est ajoutée à la table appropriée.

Création des pages Web du panier d'achat

L'application de panier d'achat fournit au client des informations produits, extraites d'un catalogue en ligne semblable à celui du Chapitre 14. Le client sélectionne des articles du catalogue et les place dans le panier. Lorsque le client est satisfait du contenu de son panier, il valide la commande : l'application élabore la commande, elle recueille les informations de livraison et stocke les articles choisis.

Conception des pages Web du panier d'achat

L'application de panier d'achat affiche cinq pages Web, dans cet ordre :

1. **Informations produits** : L'application affiche les informations produits extraites du catalogue en ligne (décrit au Chapitre 14). Le catalogue affiche deux types de pages : la page des catégories et la page des informations produits. La page des catégories est la même que celle du Chapitre 14. La page produit est similaire mais des éléments lui ont été ajoutés, nécessaires pour l'achat en ligne.
2. **Panier d'achat** : La page Web du panier d'achat affiche les articles actuellement dans le panier.
3. **Formulaire d'expédition** : Lorsque le client valide la commande, l'application affiche un formulaire pour recueillir l'adresse de livraison et les informations de carte bancaire.
4. **Page de récapitulatif** : La page de récapitulatif affiche toutes les données de la commande, y compris l'adresse.
5. **Page de confirmation** : Une fois acceptées les informations de carte bancaire, l'application affiche une page de confirmation, qui signale l'acceptation de la commande et fournit toutes informations nécessaires au client. Autre possibilité :

lorsque le client annule la commande, une page d'annulation est affichée.

Conception de la page Web des informations produits

Le Chapitre 14 a présenté l'application de catalogue en ligne, qui affiche les articles extraits d'un catalogue. L'application de ce chapitre-ci affiche également les articles du catalogue. Deux types de pages sont affichés. Une des pages est la page des catégories d'articles (voir la Figure 14.2 au chapitre précédent). Cette page est la même dans ce chapitre que dans l'application de catalogue en ligne du chapitre précédent.

Le second type de page affiche les informations produits de la catégorie choisie par le client. La page produit de l'application panier d'achat est semblable à la page produit décrite au chapitre précédent (voir la Figure 14.2), à l'exception de l'ajout de quelques éléments (Figure 15.1).

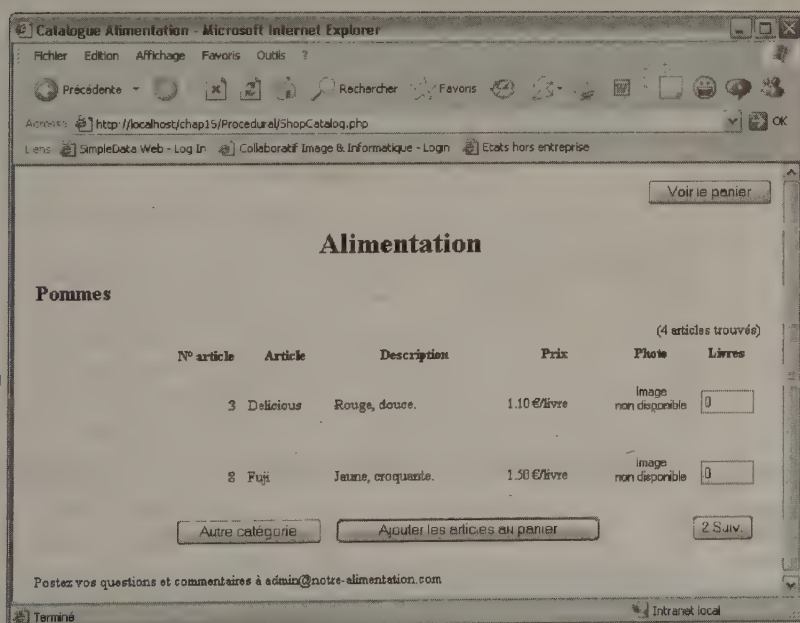


Figure 15.1 : La page produit affichée par l'application de commande en ligne.

Voici les ajouts opérés :

- ✓ **Bouton Voir le panier :** Un bouton d'envoi de formulaire Voir le panier a été ajouté dans l'angle supérieur droit de la page, il permet au client de visionner le contenu actuel de son panier d'achat. Ce bouton a également été ajouté à la page des catégories.
- ✓ **La colonne Livres :** Permet au client d'entrer le poids désiré pour chaque article. Les articles sont affichés avec un poids initial nul. Le client peut changer ce poids.
- ✓ **Bouton Ajouter les articles au panier :** Un bouton d'envoi de formulaire Ajouter les articles au panier a été ajouté.

Ce bouton ajoute au panier les articles choisis par le client.

Conception de la page Web du panier d'achat

L'application affiche les articles actuellement dans le panier (Figure 15.2).

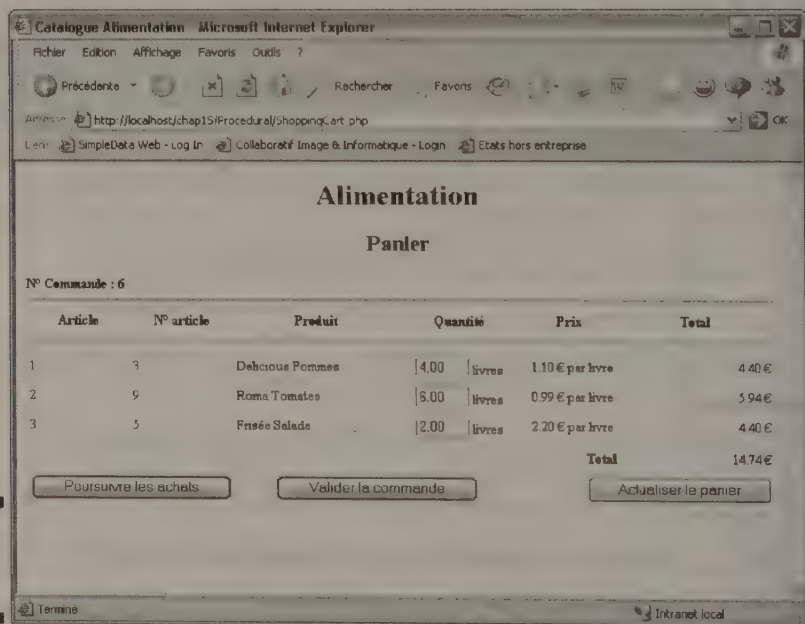


Figure 15.2 :
Le panier
d'achat.

Le panier d'achat présente trois boutons :

- ✓ **Poursuivre les achats** : Ramène le client à la première page du catalogue.
- ✓ **Valider la commande** : Valide la commande avec les articles actuellement dans le panier.
- ✓ **Actualiser le panier** : Permet au client de modifier le contenu actuel du panier. Il suffit de modifier le poids d'un article, puis de cliquer sur ce bouton. Le panier d'achat est réaffiché, avec les quantités mises à jour. Lorsque le poids entré est 0 (zéro), l'article est supprimé du panier.

Sur l'illustration, le panier comporte trois articles, qui ont été ajoutés au fur et à mesure de la navigation dans les pages produits.

Conception de la page du formulaire d'expédition

L'application recueille l'information nécessaire pour traiter et expédier la commande, via le formulaire de la Figure 15.3.

The screenshot shows a web browser window titled "Commande d'alimentation - Microsoft Internet Explorer". The address bar shows the URL "http://localhost/chap15/Procedure/ProcessOrder.php?from=cert". The browser's toolbar includes buttons for "Précédente", "Rechercher", "Favours", and "OK". The page content is titled "Commande d'alimentation" and includes the instruction "Merci de fournir les informations ci-dessous". The form fields are as follows:

- E-mail:
- Nom:
- Adresse:
- Ville:
- État:
- Code postal:
- Téléphone:
- Type de carte de crédit:
- Numéro de carte de crédit:
- Date d'expiration:

A "Poursuivre" button is located at the bottom of the form. The browser's status bar at the bottom shows "Terminé" and "Intranet local".

Figure 15.3 :
Le formulaire
collectant les
informations
pour
l'expédition.

Il s'agit d'un formulaire simplifié pour l'exemple. Dans la réalité, vous devrez probablement recueillir un nom et une adresse de facturation, en plus du nom et de l'adresse d'expédition. Vous proposerez aussi plusieurs modes de livraison, etc.

Conception de la page de récapitulatif

L'application affiche un récapitulatif de la commande, de façon que le client puisse se rendre compte des éventuelles erreurs et les corriger (Figure 15.4).

Figure 15.4 :
La page de
récapitulatif
de la
commande.

Récapitulatif de commande d'alimentation

Destinataire: Archibald Laigo
Les Flandres
La Mense, AL 72000-
00 22 33 44 55
archibald@larago.com

Numéro de commande : 6

	Article	N° article	Produit	Quantité	Prix	Total
1	3		Délicieuses Pommes	4.00	1.10 € par livr.	4.40 €
2	9		Roses Tomates	6.00	0.99 € par livr.	5.94 €
3	5		Pâtes Salades	2.00	2.20 € par livr.	4.40 €
Sous-total						14.74 €
TVA						0.00 €
Port						0.75 €
Total						15.40 €

Terminé

La page de récapitulatif présente quatre boutons :

- ✓ **Poursuivre les achats** : Ramène le client à la première page du catalogue tout en conservant les informations actuelles de la commande.
- ✓ **Modifier infos livraison** : Ramène le client au formulaire d'information sur l'expédition, où il pourra modifier les informations de livraison à sa guise.
- ✓ **Annuler la commande** : Annule complètement la commande.

- ✓ **Valider la commande** : Valide la commande affichée dans cette page de récapitulatif. Après cette validation, le client ne peut plus opérer de changements à sa commande.

La page de récapitulatif présente aussi la TVA.

Les frais de port se montent à 25 centimes par article. Ainsi, le port pour ces trois articles est de 75 centimes. Ce forfait a été choisi pour simplifier l'exemple.

Conception de la page de confirmation

La page de confirmation est spécifique à la boutique. Elle peut être une simple répétition de la page de récapitulatif. Une page de confirmation indique au client que la commande a été acceptée. Elle peut en outre signaler la date d'expédition de la commande ; elle indique aussi souvent le numéro de la commande, que le client pourra utiliser pour le suivi de commande. Cet exemple ne comporte pas de page de confirmation ou d'annulation particulière. Ce sera à vous de l'ajouter à l'application.

Écriture du code de la page des informations produits

La page produits du catalogue affiche une liste de détails sur les articles. Les articles sont affichés page par page. Chaque page affiche des boutons d'envoi pour aller à la page suivante, la précédente, et revenir à la page d'index.

Le code générant la page des produits se trouve dans deux fichiers séparés :

- ✓ `shopping_product_page.inc` : Contient le code qui définit l'aspect de la page Web. Produit une page énumérant les produits sous forme de tableau. Un autre fichier doit être utilisé en conjonction avec ce fichier pour créer la page. L'autre fichier contient les informations spécifiques à la page.
- ✓ `fields_products_page.inc` : Contient les informations spécifiques à la page produits du catalogue. Utilisé conjointement avec `shopping_product_page.inc`, il affiche les informations sur les produits. Pour afficher des produits différents, il vous faudra exploiter un autre fichier avec `shopping_product_page.inc` (voir le Chapitre 14).

Ces deux fichiers ont été décrits en détail au Chapitre 14. Dans ce chapitre, j'ai ajouté quelques éléments à la page Web, pour permettre les commandes en ligne. Cette section ne décrit que les éléments ajoutés.

Écriture de `fields_products_page.inc`

Le fichier `fields_products_page.inc` est quasiment identique au fichier décrit au Chapitre 14. Il construit deux tableaux contenant les informations spécifiques affichées dans la page des produits.

- ✓ `$page` : Contient les éléments affichés en haut et en bas de la page.
- ✓ `$table_heads` : Contient les éléments fournissant les titres du tableau (HTML) affichant les articles.

L'unique différence entre ce fichier et celui du Chapitre 14 est un élément supplémentaire dans le tableau `$table_heads`. Il est identifié dans l'explication suivant le Listing 15.1.

Les fichiers procédural et orienté objet sont identiques.

Listing 15.1 : Tableaux nécessaires à la page produits.

```
<?php
/* Fichier : fields_products_page.inc
 * Description : Définit les variables et construit les tableaux
 *              nécessaires à l'affichage de la page produits du catalogue.
 */
#$n_per_page = 2;
$page = array( "title" => "Catalogue Alimentation",
              "top"    => "Alimentation",
              "bottom" => "Postez vos questions et commentaires
                          à admin@notre-alimentation.com",
              );
$table_heads = array("catalog_number" => "N° article",
                    "name"             => "Article",
                    "description"      => "Description",
                    "price"            => "Prix",
                    "pix"              => "Photo",
                    "quantity"         => "Livres",
                    );
?>
```

La page produits présente une colonne supplémentaire. Cette nouvelle colonne permet au client d'entrer la quantité désirée pour l'article. Il a donc fallu ajouter la colonne au tableau \$table_heads. Il s'agit de la colonne amount (quantité).

Écriture de shopping_product_page.inc

Les fichiers shopping_product_page.inc et shopping_product_page-oo.inc sont très similaires aux fichiers shopping_product_page.inc et shopping_product_page-oo du Chapitre 14. Cependant, quelques ajouts étaient requis pour permettre la commande en ligne. Le Listing 15.2 présente le fichier après ajouts. Cette section ne décrit que les lignes qui ont changé. Pour une description complète de toutes les lignes, voir la section "Écriture du code de la page produits" au Chapitre 14.

Le fichier pour l'application procédurale est légèrement différent de celui pour l'application orientée objet. Je présente d'abord le fichier procédural, puis décrit les changements nécessaires pour employer le fichier dans l'application orientée objet.

Le fichier du Listing 15.2 définit l'aspect de la page des produits.

Listing 15.2 : Fichier définissant la page des produits.

```
<?php
/* Fichier : shopping_product_page.inc
 * Description : Affiche les articles de la
 *              catégorie choisie.
 */
?>
<html>
<head><title><?php echo $page['title'] ?></title></head>
<body>
<?php
echo "<form action='ShoppingCart.php' method='POST'>\n
    <p style='text-align: right'>\n
    <input type='submit' name='Cart'
        value='Voir le panier'>\n
    </form>\n";
echo "<div style='margin-left: .1in; margin-right: .1in'>\n
    <h1 style='text-align: center'>{$page['top']}</h1>\n
    <p style='font-size: 150%'><b>{$_POST['interest']}</b>";
echo "<p align='right'>($n_products articles trouvés)\n";
#10
#15
```

```

echo "<table border='0' cellpadding='5' width='100%'\>\n";
echo "<tr>";
foreach($table_heads as $heading)
{
    echo "<th>$heading</th>";
}
echo "</tr>\n";
echo "<form action='$_SERVER[PHP_SELF]' method='POST'\>\n";
for($i=$n_start;$i<=$n_end;$i++)
{
    echo "<tr>";
    echo "<td align='right'>
        {$products[$i]['catalog_number']}</td>\n";
    echo "<td>{$products[$i]['name']}</td>\n";
    echo "<td>{$products[$i]['description']}</td>\n";
    echo "<td>{$products[$i]['price']} €/livre</td>\n";
    echo "<td>
        <img src='images/{$products[$i]['pix']}'></td>\n";
    echo "<td style='text-align: center'><input type='text'
        name='item{$products[$i]['catalog_number']}'
        value='0'.size='4'></td>\n";
    echo "</tr>";
}
echo "<input type='hidden' name='n_end' value='$n_end'\>\n";
echo "<input type='hidden' name='interest'
    value='$_POST[interest]'\>\n";
echo "<tr>
    <td colspan='2'><input type='submit'
        value='Autre catégorie'></td>\n";
echo "<td colspan='2' style='text-align: center'>
    <input type='submit' name='Products'
        value='Ajouter les articles au panier'>";
echo "<td colspan='2' style='text-align: right'\>\n";
    if($n_end > $n_per_page)
    {
        echo "<input type='submit' name='Products'
            value='Préc.'>\n";
    }
    if($n_end < $n_products)
    {
        echo "<input type='submit' name='Products'
            value='$n_per_page Suiv.'>\n";
    }
echo "</td></form></tr></table>\n";
echo "<p style='font-size: 75%; text-align: center'>
    {$page['bottom']}</p>";
?>
</div></body></html>

```

Dans les explications suivantes, les numéros renvoient aux numéros de lignes du Listing 15.2. Ce chapitre n'étudie que les éléments ajoutés au fichier. Pour plus d'informations sur le fichier d'origine, reportez-vous aux explications du Listing 14.4.

- #11 Les lignes 11 à 15 ajoutent le bouton Voir le panier dans l'angle supérieur droit de la page produits. Ce bouton affiche le contenu actuel du panier.
- #38 Les lignes 38 à 40 ajoutent un champ de saisie dans chaque ligne d'article, où le client peut entrer la quantité commandée. Le nom du champ est construit à l'aide du numéro d'article dans le catalogue. Par exemple, le nom du champ sera `item11` ou `item4002`. La quantité par défaut est 0 (zéro). Le client peut la changer à volonté. Le nom du champ est passé au tableau `$_POST`, avec la quantité. Le programme de traitement obtient le numéro de produit à partir du nom de champ.
- #49 Les lignes 49 à 51 ajoutent un bouton supplémentaire Products à la page des produits. Il porte la mention Ajouter les articles au panier.

Modification de `shopping_product_page.inc` pour usage avec l'application orientée objet

Le Listing 15.2 affiche la page des produits dans l'application écrite en code procédural. Le fichier nécessite quelques modifications pour fonctionner avec le code orienté objet.

Les lignes 30 à 41 affichent les informations produits sur chaque ligne. Dans le Listing 15.2, les informations sont affichées à partir d'un tableau nommé `$products`. Ce tableau est construit à partir de l'information dans la base de données, dans le script `ShopCatalog.php` (voir plus loin le Listing 15.11). Pour l'application orientée objet, ces lignes doivent être modifiées de façon à afficher les données à partir d'un objet. L'objet est créé dans le script `Orders-oo.php` (voir plus loin le Listing 15.18).

Pour utiliser le fichier avec l'application orientée objet, remplacez les lignes 30 à 41 du Listing 15.2 par les lignes suivantes, et nommez le nouveau fichier `shopping_product_page-oo.inc` :

```
echo "<tr>";
echo "<td align='right'>{$all[$i]->catalog_number}</td>\n";
echo "<td>{$all[$i]->name}</td>\n";
```

```

echo "<td>{$all[$i]->description}</td>\n";
echo "<td>\${$all[$i]->price} /lb</td>\n";
echo "<td align='center'>
    <img src='images/{$all[$i]->pix}'></td>\n";
echo "<td style='text-align: center'><input type='text'
    name='item{$all[$i]->catalog_number}'
    value='0' size='4'></td>\n"; #40
echo "</tr>";

```

Comme vous pouvez le constater, les variables qui étaient des tableaux (`$products[$i]['name']`) sont devenues des objets (`$all[$i]->name`).

Écriture du code de la page du panier d'achat

La page du panier d'achat affiche les articles actuellement dans le panier. Le client peut changer la quantité commandée. Il a aussi la possibilité de revenir au catalogue pour ajouter des articles supplémentaires. Il peut, enfin, valider sa commande.

Le panier d'achat n'est pas implémenté de la même manière dans le code procédural et le code orienté objet. Pour présenter les deux méthodes, j'ai stocké les articles dans la table MySQL pour le code procédural, mais dans une variable `SESSION` pour le code orienté objet. Cela a pour conséquence que le code affichant ces pages Web est légèrement différent. Les sections suivantes présentent le code pour les deux méthodes.

Écriture des fichiers de page Web du panier d'achat : méthode procédurale

Le code qui crée la page du panier d'achat se trouve dans deux fichiers séparés :

- ✓ `table_page.inc` : contient le code qui définit l'aspect de la page Web. Produit une page énumérant les produits sous forme de tableau. Un autre fichier doit être utilisé en conjonction avec ce fichier pour créer la page. L'autre fichier contient les informations spécifiques à la page.

- ✓ `fields_cart.inc` : contient les informations spécifiques à la page Web du panier. Utilisé conjointement avec `table_page.inc`, affiche les éléments stockés dans le panier.

Écriture de `fields_cart.inc`

Le Listing 15.3 présente le code qui définit les variables pour la page du panier.

Listing 15.3 : Code définissant les variables pour le panier d'achat.

```
<?php
/* Fichier : fields_cart.inc
 * Description : Fournit les informations nécessaires
 *              pour afficher le panier.
 */
$page = array( "title" => "Catalogue Alimentation",
              "top"    => "Alimentation",
              "top2"   => "Panier",
              "bottom" => "Postez vos questions et commentaires
                          à admin@notre-alimentation.com",
              );

$table_headers = array( "Article", "N° article", "Produit",
                       "Quantité", "Prix", "Total" );

$order_number = $_SESSION['order_number']; #14
$table_name = $order_number;
$cxn = connect_to_db("Vars.inc"); #16
$sql_1 = "SELECT * FROM order_item
        WHERE order_number='$order_number'";
$result = mysqli_query($cxn,$sql_1)
        or die("sql_1: ".mysqli_error($cxn));
$num_row = mysqli_num_rows($result); #21
if($num_row < 1) #22
{
    echo "Votre panier est vide<br>\n
        <a href='ShopCatalog.php'>Poursuivre les achats</a>\n";
    exit();
}

$num=1; #28
while($row = mysqli_fetch_assoc($result)) #29
{
    foreach($row as $field => $value)
    {
        if($field != "order_number")
        {
            $items[$num][$field]=$value;
```

```

        if($field == "catalog_number")
        {
            $sql_2 = "SELECT name,type FROM food WHERE
                    catalog_number = '$row[catalog_number]'";
            $result2 = mysqli_query($cxn,$sql_2)
                or die("sql_2: ".mysqli_error($cxn));
            $row = mysqli_fetch_row($result2);
            $items[$n]["name"]=$row[0]." ".$row[1];
        }
    }
    }
    $n++;
}
?>

```

Les numéros de lignes dans le Listing 15.3 correspondent aux explications numérotées ci-dessous :

- #14** Récupère le numéro de commande dans une variable de session. Le numéro de commande a été stocké dans la variable de session lorsque la commande a été stockée dans la base de données, c'est-à-dire lorsque l'utilisateur a cliqué sur le bouton Ajouter les articles au panier. Le numéro de commande identifie la commande dans la base de données.
- #16** Les lignes 16 à 20 lisent les articles dans la table `Order_Item` de la base de données. (En d'autres termes, ces lignes lisent les articles actuellement stockés dans le panier d'achat.)
- #21** Règle `$n_rows` sur le nombre d'éléments trouvés pour cette commande dans la base de données.
- #22** Débute un bloc `if` qui affiche un message et un lien lorsqu'aucun article n'existe dans la base de données pour le numéro de commande spécifié.
- #28** Définit un compteur indiquant le nombre d'articles.
- #29** Débute une boucle `while` qui crée un tableau `$items` contenant toutes les informations sur les articles. Le nom de l'aliment est lu dans le catalogue et ajouté au tableau.

Écriture de table_page.inc

Le Listing 15.4 présente le code qui définit l'aspect de la page du panier.

Listing 15.4 : Code affichant le contenu du panier.

```
<?php
/*Fichier :   table_page.inc
 *Description :   Définit une page HTML affichant des articles et
 *               des prix. La quantité et le prix sont multipliés
 *               pour obtenir le prix total.
 *               La somme des totaux des articles est également opérée.
 */
echo "<html>
      <head><title>{$page['title']}</title></head>\n
      <body>\n";
echo "<h1 style='text-align: center'>{$page['top']}</h1>\n";
echo "<h2 style='text-align: center'>{$page['top2']}</h2>\n";
echo "<p style='font-weight: bold'>N° Commande : $table_name<hr>\n";
echo "<table border = '0' style='width: 100%'>\n";
echo "<form action='$_SERVER[PHP_SELF]' method='POST'>";
echo "<tr>";
foreach($table_headers as $header)
{
    echo "<th>$header</th>\n";
}
echo "</tr>";
echo "<tr><td colspan='6'><hr></td></tr>\n";
for($i=1;$i<=sizeof($items);$i++)
{
    echo "<tr>";
    echo "<td style='width: 10%'>{$items[$i]['item_number']}</td>\n";
    echo "<td style='width: 10%'>{$items[$i]['catalog_number']}</td>\n";
    echo "<td >{$items[$i]['name']}</td>\n";
    echo "<td style='text-align: center; width: 20%'>
          <input type='text' name='quantity['
              value='{$items[$i]['quantity']}'
              size='4'> livres</td>\n";
    $f_price = number_format($items[$i]['price'],2);
    echo "<td style='text-align: right; width: 17%'>
          $f_price € par livre.</td>\n";
    $total = $items[$i]['quantity'] * $items[$i]['price'];
    $f_total = number_format($total,2);
    echo "<td style='text-align: right'>$f_total €</td></tr>\n";
    @$order_total = $order_total + $total;
}
```

#25

#40

#44

```

$fb_order_total = number_format($fb_order_total,2);
?>
<tr><td colspan='5'
    style='text-align: right; font-weight: bold'>Total</td>
    <td style='text-align: right; line-height: 200%'>
        <?php echo $fb_order_total ?> €</td></tr>
<input type='hidden' name='order_number'
    value='<?php echo $fb_order_number ?>'>
<tr><td colspan='2' style='text-align: left'>
    <input type='submit' name='Cart'
        value='Poursuivre les achats'></td>
<td colspan='2' style='text-align: center'>
    <input type='submit' name='Cart'
        value='Valider la commande'></td>
<td colspan='2' style='text-align: right'>
    <input type='submit' name='Cart' value='Actualiser le panier'></td>
</tr></table></form></body></html>

```

Les numéros dans les explications suivantes renvoient aux numéros de lignes du Listing 15.4 :

- #25** Débute une boucle `for` qui traverse le tableau `$items`, pour afficher chaque ligne du panier. La boucle se termine en ligne 45.
- #27** Les lignes 27 à 39 affichent chaque article de l'élément courant du tableau `$items`. Le prix est formaté sous forme de montant en euros.
- #40** Les lignes 40 à 43 affichent le prix total. La ligne 40 calcule le coût total, en multipliant le prix par la quantité. La ligne 41 présente le coût total en euros. Les lignes 42 à 43 affichent la ligne de l'article dans la page du panier.
- #44** Ajoute le total de l'article en cours, pour cumuler le total du panier.
- #48** Les lignes 48 à la fin sont le code HTML qui affiche le total du panier et les boutons d'envoi.

Écriture de table-page-oo.inc

Le Listing 15.6 présente le code qui définit les variables pour la page du panier.

Listing 15.6 : Code affichant le contenu du panier.

```
<?php
/*Fichier :   table_page-oo.inc
 *Description :   Définit une page HTML affichant des articles et
 *               des prix. La quantité et le prix sont multipliés
 *               pour obtenir le prix total.
 *               La somme des totaux des articles est également opérée.
 */
echo "<html>
    <head><title>{$page['title']}</title></head>\n"
    <body>\n";
echo "<h1 style='text-align: center'>{$page['top']}</h1>\n";
echo "<h2 style='text-align: center'>{$page['top2']}</h2>\n";
echo "<p style='font-weight: bold'>{$this->message}<hr>\n";
echo "<table border = '0' style='width: 100%'\n";
echo "<form action='$_SERVER[PHP_SELF]' method='POST'>";
if(sizeof($this->items) > 0)                                     #17
{
    echo "<tr>";
    foreach($table_headers as $header)
    {
        echo "<th>$header</th>\n";
    }
    echo "</tr>";
    echo "<tr><td colspan='6'><hr></td></tr>\n";
    for($i=0;$i<sizeof($this->items);$i++)                       #26
    {
        echo "<tr>";
        echo "<td width='5%'>".($i+1)."</td>\n";
        $cat = new catalog("Vars.inc");                         #30
        $cat->selectCatalog("OnlineOrders");
        $cat_no = $this->items[$i]->getCatalogNumber();
        $food_name = $cat->getName($cat_no);
        echo "<td width='5%'>$cat_no</td>\n";
        echo "<td>". $food_name."</td>\n";                       #34
        echo "<td width='20%' style='text-align: center'>
            <input size='4' type='text' name='item$cat_no'
                value='". $this->items[$i]->getQuantity().
                "' > livres</td>\n";
        echo "<td width='15%' style='text-align: right'> $.
            number_format($this->items[$i]->getPrice(),2).
```

```

        " par livre</td>\n";
$total = $this->items[$i]->getQuantity() *
        $this->items[$i]->getPrice();
echo "<td style='text-align: right; width: 7%'"> $.
        number_format($total,2)."/>\n";
@$order_total = $order_total + $total;
}
$f_order_total = number_format($order_total,2);
echo "<tr><td colspan='5'
        style='text-align: right;
        font-weight: bold'">Total</td>
        <td style='text-align: right;
        line-height: 200%'">$$f_order_total</td></tr>\n";
écho "<tr><td colspan='2' style='text-align: left'">
        <input type='submit'
        value='Poursuivre les achats'"></td>\n";
echo "        <td colspan='2' style='text-align: center'">
        <input type='submit' name='Ship'
        value='Valider la commande'"></td>\n";
echo "        <td colspan='2' style='text-align: right'">
        <input type='submit' name='Cart'
        value='Actualiser le panier'"></td>\n";
echo "</tr>\n";
}
else
{
    echo "<hr>";
    echo "<tr><td colspan='5' style='text-align: left'">
        <input type='submit'
        value='Poursuivre les achats'"></td>\n";
}
?>
</table></form>

```

Les numéros suivants renvoient aux numéros de lignes du Listing 15.6 :

- #17** Débute un bloc if qui s'exécute s'il existe un ou plusieurs éléments dans le panier d'achat. La variable `$items` est un tableau d'objets articles, stocké dans la classe `ShoppingCart`, étudiée plus loin dans ce chapitre. La classe `ShoppingCart` est créée par le script de l'application. Le bloc if affiche la page du panier. Le bloc if se termine ligne 65.

- #26 Débute une boucle `for` qui traverse le tableau `$items` et affiche dans la page Web l'information pour chaque article. La boucle `for` se termine ligne 48.
- #30 Les lignes 30 à 33 obtiennent des informations en utilisant un objet `Catalog`. Par exemple, la ligne 33 devient le nom de l'article en utilisant la méthode `getName` de la classe `Catalog`. Les lignes 34 à 35 affichent les informations obtenues.
- #44 Les lignes 43 à 44 cumulent le total du panier.
- #49 Les lignes 49 à la fin affichent le total du panier et les bouton d'envoi.
- #66 Débute un bloc `else` qui s'exécute lorsque le panier ne comporte aucun élément. Le bloc affiche un bouton qui ramène le client à la page des catégories du catalogue.

Écriture du code du formulaire d'expédition

Le formulaire d'informations d'expédition recueille les informations nécessaires à l'expédition des produits. Il collecte le nom, l'adresse, le numéro de téléphone et autres informations. Le code qui affiche le formulaire est dans deux fichiers séparés :

- ✓ `single_form.inc` : Contient le code qui définit l'aspect de la page Web contenant le formulaire. Pour créer le formulaire Web, vous devez utiliser un autre fichier en conjonction avec celui-ci. L'autre fichier contient les informations spécifiques, comme les noms de champs.
- ✓ `fields_ship_info.inc` : Contient les tableaux et variables exploités par `single_form.inc` pour afficher le formulaire Web d'informations d'expédition.

Écriture de `fields_ship_info.inc`

Le fichier `fields_ship_info.inc` fournit les informations nécessaires pour afficher le formulaire Web d'informations d'expédition. Le fichier (Listing 15.7) définit six tableaux contenant les informations spécifiques affichées dans le formulaire d'informations d'expédition.

- ✓ `$page` : Contient les éléments affichés en haut et en bas de la page.
- ✓ `$ship_info` : Contient les éléments fournissant les noms de champs et étiquettes pour le formulaire d'expédition. Par exemple, pour le nom de champ `email`, l'étiquette associée, affichée à gauche de la case dans le formulaire, est `E-mail`.
- ✓ `$cc_types` : Contient les éléments qui fournissent les types de cartes bancaires acceptées. Dans le formulaire, les types de cartes bancaires sont affichés via une liste déroulante.
- ✓ `$length` : Contient les longueurs autorisées pour chaque champ du formulaire.
- ✓ `$elements` : Contient les éléments affichés en haut et en bas du formulaire. Ce tableau ne contient que le texte à afficher sur le bouton d'envoi.
- ✓ `$months` : Contient les mois de l'année. Les mois sont affichés dans une liste déroulante, pour indiquer la date d'expiration de la carte bancaire.

En outre, le fichier définit une variable contenant la date, et plusieurs variables contenant les informations d'expédition actuellement stockées dans la base de données et qui sont affichées dans les champs du formulaire.

Le fichier utilisé avec le script procédural est légèrement différent de celui utilisé avec le script orienté objet. Le Listing 15.7 présente le fichier exploité par l'application procédurale. À la suite du listing, vous trouvez les changements requis pour employer ce fichier dans le script orienté objet.

Listing 15.7 : Fichier contenant les variables pour le formulaire d'expédition.

```
<?php
/* Fichier : fields_ship_info.inc
 * Description : Contient des tableaux contenant les noms de
 * champs et éléments de formulaire pour la page
 * Web d'identification.
 */
$page = array( "title" => "Commande d'alimentation",
              "top"    => "Commande d'alimentation",
```

```

"top2" => "Merci de fournir les
          informations ci-dessous",
"bottom" => "Postez vos questions et commentaires
            à admin@notre-alimentation.com",
);
$ship_info = array("email"      => "E-mail",
                  "ship_name"   => "Nom",
                  "ship_street" => "Adresse",
                  "ship_city"   => "Ville",
                  "ship_state"  => "État",
                  "ship_zip"    => "Code postal",
                  "phone"       => "Téléphone",
                  "cc_type"     => "Type de carte de crédit",
                  "cc_number"   => "Numéro de carte de crédit",
                  "cc_exp"      => "Date d'expiration"
                );
$cc_types = array("visa"       => "Visa",
                  "mc"         => "Master Card",
                  "amex"       => "American Express"
                );
$length = array("email"      => "55",
                "ship_name"   => "40",
                "ship_street" => "55",
                "ship_city"   => "40",
                "ship_zip"    => "10",
                "phone"       => "15",
                "cc_number"   => "20"
                );
$elements = array("submit"    => "Poursuivre");
$months = array(1=> "Janvier", "Février", "Mars",
                "Avril", "Mai", "Juin", "Juillet",
                "Août", "Septembre",
                "Octobre", "Novembre", "Décembre"
                );
$today = time("Y-m-d");
if(!isset($_POST['state']))
{
    $connect = connect_to_db("Vars.inc");
    $sql = "SELECT
            ship_name,ship_street,ship_city,ship_state,ship_zip,
            phone,email FROM Customer_Order WHERE
            order_number = '{$_SESSION['order_number']}'";
    $result = mysqli_query($connect,$sql)
            or die("Erreur : ".mysqli_error($connect));
    $n = mysqli_num_rows($result);
    if($n > 0)
    {
        $row = mysqli_fetch_assoc($result);

```

#47

#54

#55


```

extract($row);
}
}
?>

```

Dans les explications suivantes, les numéros renvoient aux numéros de lignes du Listing 15.7 :

#45 Débute un bloc `if` qui s'exécute s'il n'y a pas de données POST. Aucune donnée POST n'existe lorsque l'utilisateur clique sur le bouton Modifier infos livraison dans la page de récapitulatif. Ce bloc lit les informations d'expédition dans la base de données.

#47 Les lignes 47 à 53 créent et exécutent une requête SQL qui sélectionne les informations d'expédition dans la base de données. `Connect_to_db` est une fonction utilisée fréquemment dans ce livre. Elle est décrite au Chapitre 12. La fonction est stockée dans le fichier `functions_main.inc`, inclus dans les trois scripts d'applications, décrits à la section "Création de l'application de panier d'achat : démarche procédurale."

#54 Teste si des informations d'expédition ont été trouvées.

#55 Débute un bloc `if` qui s'exécute si des informations d'expédition ont été trouvées. Si c'est le cas, l'information est extraite et placée dans des variables avec les noms de champs.

Modification de `fields_ship_info.inc` pour usage avec l'application orientée objet

Les 44 premières lignes du fichier sont les mêmes dans les versions procédurale et orientée objet.

En revanche, les variables créées pour la version procédurale dans les lignes restantes de ce fichier sont, dans l'application orientée objet, créées dans une classe. Par conséquent, pour employer le fichier dans l'application orientée objet, vous devez supprimer les lignes restantes (45 à 57). Nommez le nouveau fichier `fields_ship_info-oo.inc`.

Écriture de *single_form.inc*

Le fichier `fields_ship_info.inc`, ou `fields_ship_info-oo.inc`, fournit l'information nécessaire pour afficher le formulaire Web d'informations d'expédition. Le fichier `single_form.inc` définit l'aspect du formulaire d'expédition. Le Listing 15.8 présente ce fichier. Le même fichier est exploité dans le script procédural et orienté objet.

Listing 15.8 : Fichier définissant le formulaire d'informations d'expédition

```
<?php
/* Fichier : single_form.inc
 * Description : Contient le code d'une page Web affichant
 *              un formulaire HTML.
 */
include("functions.inc");
echo "<head><title> {$page['title']} </title></head>\n
      <h2 align='center'>{$page['top']} </h2>\n
      <p style='font-style: italic;
                font-weight: bold'>{$page['top2']}\n
      <form action='{$_SERVER['PHP_SELF']}' method='POST'>
      <table border='0' cellpadding='5' cellspacing='0'>\n";
if(isset($GLOBALS['message']))                                #13
{
    echo "<tr>
          <td colspan='2'
              style=\"font-weight: bold; font-style: italic;
                    font-size: 90%; color: red\">
              {$GLOBALS['message']}</td></tr>";
}
foreach($ship_info as $field => $value)                        #21
{
    if($field == "ship_state")                                  #23
    {
        echo "<tr><td style=\"text-align: right;
                        font-weight: bold\">État</td>
              <td><select name='ship_state'>";
        $stateName=getStateName();                             #28
        $stateCode=getStateCode();
        for ($n=1;$n<=50;$n++)
        {
            $state=$stateName[$n];
            $scode=$stateCode[$n];
            echo "<option value='$scode'";
            if($scode == @$_POST['state'] ||
```

```

        $score == @$ship_state)
            echo " selected";
        echo ">$state\n";
    }
    echo "</select>";
}
elseif($field == "cc_type") #42
{
    echo "<tr><td style=\"text-align: right;
        font-weight: bold\">Type de carte de crédit</td>
        <td><select name='cc_type'>";
    foreach($cc_types as $field => $value)
    {
        echo "<option value='$field'";
        echo ">$value\n";
    }
    echo "</select>";
}
elseif($field == "cc_exp") #54
{
    echo "<tr><td style=\"text-align: right;
        font-weight: bold\">Date d'expiration</td>
        <td><select name='cc_exp_mo'>";
    for($n=1;$n<=12;$n++)
    {
        echo "<option
            value='$n'>{$months[$n]}\n";
    }
    echo "</select>\n";
    echo "<select name='cc_exp_da'>";
    for($n=1;$n<=31;$n++)
    {
        echo " <option value='$n'>$n\n";
    }
    echo "</select>\n";
    echo "<select name='cc_exp_yr'>";
    $start_yr = date("Y", $today);
    for($n=$start_yr;$n<=$start_yr+5;$n++)
    {
        echo " <option value='$n'>$n\n";
    }
    echo "</select>\n";
}
else #79
{
    echo "<tr><td style=\"text-align: right;
        font-weight: bold\">$value</td>
        <td><input type='text' name='$field'

```

```

        value="".@$field.'"
        size='{length[$field]}'
        maxsize='{length[$field]}'
    </td></tr>";
}
#89
?>
<tr><td colspan="2" style="text-align: center">
    <p style="margin-top: .05in">
        <input type="submit" name="Summary"
            value="<?php echo $elements['submit']?>">
    </td></tr></table></form></body></html>

```

Les numéros dans les explications suivantes renvoient aux numéros de lignes du Listing 15.8 :

- #13** Débute un bloc `if` qui vérifie si un message est stocké dans le tableau `$GLOBALS`. Si c'est le cas, le bloc affiche le message au début du formulaire. Le script de l'application stocke le message lorsqu'une condition d'erreur est rencontrée.
- #21** Débute une boucle `foreach` qui traverse le tableau `$ship_info`. Le tableau `ship_info` est construit dans `fields_ship_info.inc`.
- #23** Débute un bloc `if` qui s'exécute lorsque le nom de champ est `ship_state`. Les lignes 23 à 39 affichent une liste déroulante contenant les noms d'états. Les lignes 28 à 29 appellent des fonctions stockées dans le fichier `functions.inc`, inclus ligne 6.
- #42** Débute un bloc `elseif` qui s'exécute lorsque le nom de champ est `cc_type`. Les lignes 41 à 52 créent une liste déroulante contenant les types de cartes bancaires acceptés.
- #54** Débute un bloc `elseif` qui s'exécute lorsque le nom de champ est `cc_exp`. (Ce champ contient la date d'expiration de la carte bancaire.) Les lignes 53 à 77 créent une liste déroulante de dates.
- #79** Débute un bloc `else` qui s'exécute pour tous les autres champs. Des cases de saisie sont affichées dans le formulaire pour tous les autres champs.

- #91 Première ligne d'une section HTML affichant le bouton d'envoi et la balise de fin du formulaire.

Écriture du code pour la page de récapitulatif

La page Web de récapitulatif présente la commande finale au client. Ce dernier peut ainsi vérifier les articles sélectionnés et les informations d'expédition. Il peut envoyer la commande ou la modifier. Le code qui affiche la page de récapitulatif se trouve dans deux fichiers séparés :

- ✓ `summary_form.inc` : contient le code qui définit l'aspect de la page de récapitulatif. Un autre fichier doit être utilisé en conjonction avec ce fichier pour fournir les informations spécifiques.
- ✓ `fields_summary.inc` : contient les tableaux et variables utilisés par `summary_form.inc` pour afficher la page de récapitulatif.

Écriture de `fields_summary.inc`

Le fichier `fields_summary.inc` fournit l'information nécessaire pour afficher la page de récapitulatif. Le Listing 15.9 définit deux tableaux contenant les informations spécifiques affichées dans la page de récapitulatif.

- ✓ `$page` : Contient les éléments affichés en haut et en bas de la page.
- ✓ `$table_headers` : Contient les titres de colonnes du tableau présentant au client le récapitulatif de la commande.

De plus, le fichier récupère le numéro de commande à partir de la session, le stocke dans `$order_number`, définit une variable avec le forfait de base des frais de port de la commande, et définit un nom pour le tableau.

Le fichier utilisé avec le script procédural est légèrement différent de celui utilisé avec le script orienté objet. Le Listing 15.9 montre le fichier à utiliser dans l'application procédurale. À la suite du listing, vous trouvez les changements requis pour employer ce fichier avec le script orienté objet.

Listing 15.9 : Fichier fournissant les informations pour la page de récapitulatif.

```

<?php
/* Fichier : fields_summary.inc
 * Description : Construit les tableaux nécessaires pour afficher
 *              un récapitulatif de la commande.
 */
$page = array("title" => "Récapitulatif de commande d'alimentation",
              "top"    => "Récapitulatif de commande d'alimentation",
              "top2"   => "Récapitulatif de commande",
              "table"  => "Numéro de commande : ",
              "bottom" => "Postez vos questions et commentaires
                          à admin@notre-alimentation.com",
              );
$table_headers = array( "Article", "N° article", "Produit",
                        "Quantité", "Prix", "Total");
$order_number = $_SESSION['order_number'];
$shipping_rate = .25;
$table_name = $order_number;
$cxn = connect_to_db("Vars.inc");
$sql_ord = "SELECT * FROM order_item
            WHERE order_number='$order_number'";
$result = mysqli_query($cxn,$sql_ord)
        or die("sql_ord: ".mysqli_error($cxn));
$n_row = mysqli_num_rows($result);
if($n_row < 1)
{
    echo "Votre panier est vide<br>\n
        <a href='ShopCatalog.php'>Poursuivre les achats</a>\n";
    exit();
}
$n=1;
while($row = mysqli_fetch_assoc($result))
{
    foreach($row as $field => $value)
    {
        if($field != "order_number")
        {
            $items[$n][$field]=$value;
            if($field == "catalog_number")
            {
                $sql_name = "SELECT name,type FROM food WHERE
                            catalog_number = '$row[catalog_number]'";
                $result2 = mysqli_query($cxn,$sql_name)
                        or die("sql_name: ".mysqli_error($cxn));
                $row = mysqli_fetch_row($result2);
                $items[$n]["name"]=$row[0]." ".$row[1];
            }
        }
    }
}

```

```

    }
  }
  $n++;
}
?>
#49

```

Les numéros dans les explications suivantes renvoient aux numéros de lignes du Listing 15.9 :

- #15** Récupère le numéro de commande à partir de la session et le stocke dans `$order_number`.
- #16** Stocke le forfait de base des frais de port dans une variable.
- #17** Stocke le nom du tableau affiché dans la page de récapitulatif.
- #18** Les lignes 18 à la fin créent un tableau (array) d'informations de commande, qui sera affiché dans le formulaire de récapitulatif. Les lignes 18 à 22 établissent la connexion à la base de données, et sélectionnent tous les articles actuels de la commande. La ligne 23 règle `$n_row` sur le nombre d'articles retournés.
- #24** Débute un bloc `if` qui s'exécute lorsqu'aucun article n'est trouvé. Le bloc affiche un message et fournit un lien qui ramène le client au catalogue.
- #30** Les lignes 30 à 50 créent le tableau `$items` qui contient les informations sur l'article.

Modification de `fields_summary.inc` pour usage dans l'application orientée objet

Les 17 premières lignes du fichier sont les mêmes dans les versions procédurale et orientée objet.

En revanche, les variables créées pour la version procédurale dans les lignes restantes de ce fichier sont, dans l'application orientée objet, créées dans une classe. Par conséquent, pour employer le fichier dans l'application orientée objet, vous devez supprimer les lignes restantes (18 à 50). Nommez le nouveau fichier `fields_summary-oo.inc`.

Écriture de *summary_page.inc*

Le fichier `summary_page.inc` affiche la page Web de récapitulatif (Listing 15.10). Le même fichier est utilisé dans le script procédural et orienté objet.

Listing 15.10 : Fichier définissant la page de récapitulatif.

```
<?php
/*Fichier :    summary_page.inc
 *Description :  Définit une page HTML affichant
 *              le récapitulatif de la commande.
 */
echo "<html>
      <head><title>{$page['title']}</title></head>\n
      <body>\n";
echo "<h2 style='text-align: center'>{$page['top']}</h2>\n";
echo "<p style='position: absolute; margin-top: .25in;
      font-weight: bold'>Destinataire :</p>";                                     #11
echo "<p style='position: absolute; margin-top: .25in;
      margin-left: .75in'>{$ship_name}<br>";
echo "$ship_street<br>
      $ship_city, $ship_state $ship_zip<br>
      $phone<br>
      $email<br>";                                                                #17
echo "<div style='margin-top: 1.5in'>";
echo "<p style='font-weight: bold'>
      <b>{$page['table']}</b> $table_name\n";
echo "<table border = '0' style='width: 100%'>\n";
echo "<form action='$_SERVER[PHP_SELF]' method='POST'>";
echo "<tr>";
foreach($table_headers as $header)
{
    echo "<th>$header</th>\n";
}
echo "</tr>";
for($i=1;$i <=sizeof($items);$i++)                                           #29
{
    echo "<tr>";
    echo "<td width='10%'>$i</td>";
    echo "<td width='10%'>
        {$items[$i]['catalog_number']}</td>";
    echo "<td>{$items[$i]['name']}</td>";
    echo "<td>{$items[$i]['quantity']}</td>";
    $f_price = number_format($items[$i]['price'],2);
    echo "<td style='text-align: right; width: 17%'>
        $f_price € par livre.</td>\n";
```

```

$total = $items[$i]['quantity'] * $items[$i]['price'];
$f_total = number_format($total,2);
echo "<td style='text-align: right'>$f_total €</td>\n";
echo "</tr>";
$order_subtotal = $order_subtotal + $total;
}
$order_subtotal = number_format($order_subtotal,2); #46
if(substr($ship_zip,0,5) > 75000
    && substr($ship_zip,0,5) < 80000) #48
{
    $taxrate = .0700;
}
else
{
    $taxrate = 0.0; #55
}
$sales_tax = $order_subtotal * $taxrate; #56
$f_sales_tax = number_format($sales_tax,2);
$shipping = $shipping_rate * sizeof($items); #58
$f_shipping = number_format($shipping,2);
$order_total = $order_subtotal + $sales_tax + $shipping; #60
$f_order_total = number_format($order_total,2); #61
echo "<tr><td colspan='5' style='text-align: right;
    font-weight: bold'>Sous-total</td>
    <td style='text-align: right; line-height: 200%'>
        $f_order_subtotal €</td></tr>\n";
echo "<tr><td colspan='5'
    style='text-align: right; font-weight: bold'>
    TVA</td>
    <td style='text-align: right; line-height: 50%'>
        $f_sales_tax €</td></tr>\n";
echo "<tr><td colspan='5' style='text-align: right;
    font-weight: bold'>Port</td>
    <td style='text-align: right; line-height: 50%'>
        $f_shipping €</td></tr>\n";
echo "<tr><td colspan='5'
    style='text-align: right; font-weight: bold'>
    Total</td>
    <td style='text-align: right; line-height: 300%'>
        $f_order_total €</td></tr>\n";
echo "<tr><td colspan='2' style='text-align: left'>
    <input type='submit'
        value='Poursuivre les achats'></td>\n";
echo "    <td colspan='1' style='text-align: center'>
    <input type='submit' name='Ship'
        value='Modifier infos livraison'></td>\n";
echo "    <td colspan='1' style='text-align: right'>
    <input type='submit' name='Final'

```

```

                                value='Annuler la commande'></td>\n";
echo "      <td colspan='2' style='text-align: right'>
                                <input type='submit' name='Final'
                                value='Valider la commande'></td>\n";
echo "</tr></table></form>\n";
?>

```

Dans les explications suivantes, les numéros renvoient aux numéros de lignes du Listing 15.10 :

- #10** Les lignes 10 à 17 affichent les informations d'expédition en haut de la page de récapitulatif.
- #18** Les lignes 18 à 28 affichent le haut du formulaire et les titres des colonnes du tableau.
- #29** Les lignes 29 à 45 affichent les articles de la commande dans la page de récapitulatif.
- #47** Débute une instruction `if/else` qui définit le taux de taxe.
- #55** La TVA est calculée en multipliant le coût total des articles par le taux de taxe.
- #58** Le montant du transport est obtenu en multipliant le nombre d'articles par le forfait de base des frais de port par article.
- #60** Le total de la commande est calculé en additionnant le total de chaque article, la TVA et le montant du transport.
- #62** Les lignes restantes affichent le total des articles, les frais de port, la TVA, le total de la commande, puis les quatre boutons d'envoi.

Création de l'application de panier d'achat : démarche procédurale

L'application de panier d'achat comporte trois scripts :

- ✓ **ShopCatalog.php** : Gère et affiche les pages Web du catalogue. Lorsque le client clique sur le bouton Ajouter les articles au panier, les informations sur l'article sont stockées dans la base de données et le panier est affiché.

- ✓ `ShoppingCart.php` : Gère et affiche le panier d'achat. Met à jour le panier lorsque le client clique sur le bouton Actualiser le panier. Lorsque le client clique sur le bouton Valider la commande, le script de l'application affiche le formulaire d'informations d'expédition.
- ✓ `ProcessOrder.php` : Traite le formulaire d'informations d'expédition, stocke les informations de commande dans la base de données et affiche le formulaire récapitulatif. Il traite la carte bancaire et les informations de commande lorsque le client clique sur le bouton Valider la commande.

Écriture de *ShopCatalog.php*

Le premier script de l'application de panier d'achat affiche le catalogue et stocke dans le panier les articles choisis par le client.

`ShopCatalog.php` est organisé sous forme d'instructions if imbriquées, basées sur le bouton sur lequel le client a cliqué. Voici une vue d'ensemble de la structure du script :

si (le bouton nommé Products a été cliqué)

si (bouton = "Ajouter les articles au panier")

1. Déterminer le numéro de commande

Si une commande en cours existe,
obtenir le numéro. Si non, créer une
nouvelle commande dans la base de
données et définir le nouveau numéro
de commande pour qu'il devienne le
numéro de commande en cours.

2. Stocker dans la base de données les
articles sélectionnés.

3. Passer le contrôle à `ShoppingCart.php`,
qui affiche le panier d'achat.

sinon (si le bouton n'est pas Ajouter les articles au panier)

Afficher la page des produits du catalogue

sinon (le bouton nommé Products n'a pas été cliqué)

afficher la page des catégories du catalogue

Ce script s'exécute lorsqu'un de ces événements se produit :

- ✓ **Le client entre l'URL de `ShopCatalog.php` dans le navigateur.**
C'est le premier script de l'application, il s'exécute donc normalement. Dans ce cas, aucun bouton n'a été cliqué, le

script parvient donc à la dernière instruction `else` et affiche la page d'index du catalogue.

- ✓ **Le client clique sur le bouton Ajouter les articles au panier.** Ce bouton est nommé `Products`, aussi le script entre-t-il dans le premier bloc `if`. Le second `if` vérifie la valeur du bouton. Le bouton concorde, le script entre donc dans le second bloc `if`, où il ajoute les éléments à une commande existante ou crée une nouvelle commande si aucune commande en cours n'existe. Il débute ensuite le second script, `ShoppingCart.php`, qui affiche le panier d'achat.
- ✓ **Le client clique sur Suivant ou Précédent.** Ces boutons sont nommés `Products`, le script entre donc dans le premier bloc `if`. Cependant, la valeur du bouton ne correspond pas dans l'instruction `if` interne : le script entre donc dans le bloc `else` interne, où il affiche les produits suivants ou précédents du catalogue.
- ✓ **Le client clique sur le bouton Autre catégorie.** Ce bouton n'a pas de nom, le script parvient donc à la dernière instruction `else` et affiche la page d'index du catalogue.

Le Listing 15.11 présente le code de `ShopCatalog.php` – le premier script de l'application de commande en ligne.

Listing 15.11 : Premier script d'application du panier d'achat

```
<?php
/* Programme : ShopCatalog.php
 * Description :   Affiche un catalogue de produits. Affiche deux
                  pages différentes : une page d'index qui présente
 *               les catégories et une page produits qui est affichée
 *               lorsque le client choisit une catégorie. Cette
 *               version est utilisée avec un panier pour l'achat en ligne.
 */
$n_per_page = 2;
session_start();
#echo "<pre>";
#var_dump($_SESSION);
#echo "</pre>";

include_once("functions_main.inc");
if(isset($_POST['Products']) &&
    isset($_POST['interest']))
```

```

if($POST['Products'] == "Ajouter les articles au panier")
{
    if(!isset($_SESSION['order_number']))
    {
        $connect = connect_to_db("Vars.inc");
        $today = date("Y-m-d");
        $sql_order = "INSERT INTO Customer_Order (order_date)
                        VALUES ('$today')";
        $result = mysqli_query($connect,$sql_order)
                    or die("sql_order".mysqli_error($connect));
        $order_number = mysqli_insert_id($connect);
        $_SESSION['order_number'] = $order_number;
        $n_items = 0;
    }
    else
    {
        $order_number = $_SESSION['order_number'];
        $n_items = $_SESSION['n_items'];
    }
    foreach($_POST as $field => $value)
    {
        if(substr($field,0,4) == "item" && $value > 0)
        {
            $n_items++;
            $catalog_number =
                substr($field,4,strlen($field)-4);
            $connect = connect_to_db("Vars.inc");
            $sql_price = "SELECT price FROM Food WHERE
                        catalog_number='".$catalog_number'";
            $result = mysqli_query($connect,$sql_price)
                        or die("sql_price:".mysqli_error($connect));
            $row = mysqli_fetch_assoc($result);
            $sql_item = "INSERT INTO Order_Item
                        (order_number,item_number,catalog_number,
                        quantity,price) VALUES
                        ($order_number,$n_items,$catalog_number,
                        $value,{row['price']})";
            $result = mysqli_query($connect,$sql_item)
                        or die("sql_item:".mysqli_error($connect));
        }
    }
    $_SESSION['n_items'] = $n_items;
    header("Location: ShoppingCart.php");
    exit();
}
else
{

```

```

if(isset($_POST['n_end']))                                     #64
{
    if($_POST['Products'] == "Préc.")                       #66
    {
        $n_start = $_POST['n_end'] - ($n_per_page);
    }
    else                                                       #70
    {
        $n_start = $_POST['n_end'] + 1;
    }
}
else                                                           #75
{
    $n_start = 1;
}
$n_end = $n_start + $n_per_page - 1;                         #79
$connect = connect_to_db("Vars.inc");                        #80
$query_food = "SELECT * FROM Food WHERE
               type='$_POST[interest]' ORDER BY name";
$result = mysqli_query($connect,$query_food)
        or die ("query_food: ".mysqli_error($connect));      #84
$n=1;
while($row = mysqli_fetch_assoc($result))                    #86
{
    foreach($row as $field => $value)                         #88
    {
        $products[$n][$field]=$value;
    }
    $n++;
}
$n_products = sizeof($products);                             #94
if($n_end > $n_products)
{
    $n_end = $n_products;
}
include("fields_products_page.inc");
include("shopping_product_page.inc");
}
}
else
{
    $connect = connect_to_db("Vars.inc");                     #105
    $sql_cat = "SELECT DISTINCT category,type FROM Food
               ORDER BY category,type";
    $result = mysqli_query($connect,$sql_cat)
            or die("sql_cat: ".mysqli_error($connect));
    while($row = mysqli_fetch_array($result))                 #110
    {

```

```
$food_categories[$row['category']][]=$row['type'];
}
include("fields_index_page.inc");
include("catalog_index_page.inc");
}
?>
```

Les explications suivantes renvoient aux numéros de lignes du Listing 15.11 :

- #10** Définit le nombre d'articles à afficher sur une page.
- #12** Ouvre une session. Le client demeure dans une session durant tout le processus de commande en ligne.
- #14** Les lignes 14 à 15 débutent un bloc `if` qui s'exécute si le bouton Products est trouvé dans le tableau `$_POST` et si le client a choisi une catégorie. Le bloc `if` se poursuit jusqu'à la ligne 102.
- #17** Débuté un bloc `if` qui s'exécute lorsque l'utilisateur clique sur le bouton Ajouter les articles au panier. Le bloc `if` se poursuit jusqu'à la ligne 61.
- #19** Débuté une instruction `if/else` qui règle le numéro de commande et le nombre d'articles dans le panier. Si aucun numéro de commande n'est trouvé dans la session, le bloc `if` insère une nouvelle commande dans la base de données. La date du jour est insérée. MySQL insère un numéro de commande séquentiel. La ligne 27 stocke le numéro de la nouvelle commande dans `$order_number`. La ligne 28 stocke le numéro de la nouvelle commande dans la session. Aucun article n'a encore été ajouté à la commande, `$n_items` est donc réglé sur 0 (zéro). Si un numéro de commande est trouvé, le bloc `else` récupère, dans la session, le numéro de commande et le nombre d'articles actuellement dans le panier.
- #36** Débuté une boucle `foreach` qui traverse le tableau `$_POST`. La boucle se termine en ligne 57.
- #38** Débuté un bloc `if` qui s'exécute pour tout champ du tableau contenant la sous-chaîne "item" et a une valeur supérieure à 0. La valeur est la quantité saisie par le

client. Le nom de champ contient le numéro de produit (numéro dans le catalogue) de l'article. Le bloc `if` entre les articles dans la table `order_item`. À la ligne 37, le numéro de produit est extrait du nom de champ. Le prix est lu dans le catalogue. Les informations sur l'article sont insérées dans la base de données. Le bloc `if` se termine ligne 56.

#58 Stocke le nouveau nombre d'articles dans la session.

#59 Exécute le script `ShoppingCart.php`, qui affiche le panier.

#62 Débute un bloc `else` qui s'exécute lorsque la valeur du bouton `Products` n'est pas `Ajouter les articles au panier`. La valeur du bouton est `Précédent` ou `Suivant`. Le bloc règle les numéros d'articles pour le premier et le dernier articles affichés et construit un tableau `$products` contenant les informations sur les produits. La page produits est affichée.

#103 Débute un bloc `else` qui s'exécute lorsque le bouton `Products` n'a pas été cliqué. L'utilisateur n'a pas cliqué ou alors sur un bouton sans nom ou avec un nom différent. La page d'index du catalogue est affichée.

Écriture de *ShoppingCart.php*

Le second script de l'application de panier d'achat gère et affiche le panier. Le client peut modifier la quantité des articles affichés. Si la quantité est passée à 0 (zéro), l'article est supprimé du panier. Le script est organisé sous forme d'une instruction `switch`, qui exécute du code selon la valeur du bouton que le client a cliqué. Voici une vue d'ensemble de la structure du script :

`si (aucun numéro de commande n'existe dans la session)`

Afficher un message disant que le panier est vide, et un lien qui ramène l'utilisateur à la page d'index du catalogue.

`selon (switch) (la valeur du bouton nommé Cart)`

`cas : Cart = "Poursuivre les achats"`
exécuter `ShopCatalog.php`, qui affichera la première page du catalogue.

`cas : Cart = Actualiser le panier`
1. Mettre à jour les quantités dans la base

- de données.
 - 2. Supprimer tous les articles dont la quantité vaut 0.
 - 3. Renuméroter séquentiellement les articles.
 - 4. Réafficher le panier d'achat.
- cas : Cart = Valider la commande
 Exécuter le script ProcessOrder.php, qui affiche le formulaire d'informations d'expédition
 cas par défaut :
 afficher le panier

Le Listing 15.12 présente le code de ShoppingCart.php – le deuxième script de l'application de panier d'achat.

Listing 15.12 : Le script ShoppingCart de l'application de panier d'achat.

```
<?php
/* Programme : ShoppingCart.php
 * Description :   Gère et affiche le panier.
 */
session_start();                                     #5
include("functions_main.inc");
if(!isset($_SESSION['order_number'])
    or empty($_SESSION['order_number']))              #8
{
    echo "Votre panier est vide<br>\n
    <a href='ShopCatalog.php'>Poursuivre les achats</a>\n";
    exit();
}
switch (@$_POST['Cart'])                              #14
{
    case "Poursuivre les achats":                      #16
        header("Location: ShopCatalog.php");
        break;
    case "Actualiser le panier":                       #19
        $connect = connect_to_db("Vars.inc");
        $order_number = $_SESSION['order_number'];
        $n = 1;
        /* Met à jour les quantités dans la base de données */
        foreach($_POST['quantity'] as $field => $value) #23
        {
            $sql_quant = "UPDATE Order_Item SET quantity='$value'
                           WHERE item_number= '$n'
                           AND order_number='$order_number'";
            $result = mysqli_query($connect,$sql_quant)
                or die("sql_quant: ".mysqli_error($connect));
            $n++;
        }
    }
```

```

    }
/* Efface tout article dont la quantité est nulle */                                     #33
$sql_del = "DELETE FROM Order_Item
           WHERE quantity= '0.00'
           AND order_number='$order_number'";
$result = mysqli_query($connect,$sql_del)
        or die("sql_del: ".mysqli_error($connect));
/* Renumérote les articles dans la base. 1. Met les articles dans
   un tableau. 2. Efface tous les articles de la base.
   3. Ré-insère les articles avec de nouveaux numéros d'articles. */ #41
$sql_getnew = "SELECT * from Order_Item
              WHERE order_number='$order_number'";
$result = mysqli_query($connect,$sql_getnew)
        or die("sql_getnew: ".mysqli_error($connect));
$n_rows = mysqli_num_rows($result);
if($n_rows < 1)                                                                    #47
{
    echo "Votre panier est vide<br>\n
    <a href='ShopCatalog.php'>Poursuivre les achats</a>\n";
    exit();
}
while($row = mysqli_fetch_assoc($result))                                           #53
{
    $items_new[]=$row;
}                                                                                     #56
$sql_del2 = "DELETE FROM Order_Item
            WHERE order_number='$order_number'";
$result = mysqli_query($connect,$sql_del2)
        or die("sql_del2: ".mysqli_error($connect));
for($i=0;$i<sizeof($items_new);$i++)                                              #61
{
    $sql_ord = "INSERT INTO Order_Item
               (order_number,item_number,catalog_number,
                quantity,price) VALUES
               ($order_number,$i+1,
                {$items_new[$i]['catalog_number']},
                {$items_new[$i]['quantity']},
                {$items_new[$i]['price']})";
    $result = mysqli_query($connect,$sql_ord)
        or die("sql_ord: ".mysqli_error($connect));
}                                                                                     #72
$_SESSION['n_items'] = $i;                                                         #73
include("fields_cart.inc");                                                         #74
include("table_page.inc");
break;
case "Valider la commande":                                                         #77
    header("Location: ProcessOrder.php?from=cart");
    exit();

```

```

break;
default:
    include("fields_cart.inc");
    include("table_page.inc");
    break;
}
?>
#81

```

Dans les explications suivantes, les numéros renvoient aux numéros de lignes du Listing 15.12 :

- #5** Débute une session, pour gérer la commande du client.
- #7** Débute un bloc `if` qui s'exécute lorsqu'aucune commande en cours n'existe. Affiche un message et un lien vers la page d'index du catalogue.
- #14** Débute une instruction `switch` basée sur la valeur d'un bouton nommé `Cart`.
- #16** Débute le bloc `case` qui s'exécute lorsque le bouton est `Poursuivre les achats`. Le bloc affiche la page d'index du catalogue.
- #19** Débute le bloc `case` qui s'exécute si le bouton est `Actualiser le panier`.
- #23** Débute une boucle `foreach` qui met à jour dans la base de données les quantités de chaque article.
- #33** Les lignes 33 à 38 suppriment dans la base de données tous les articles dont la quantité est 0.
- #42** Les lignes 42 à 46 sélectionnent les articles restants, à partir de la base de données.
- #47** Débute un bloc `if` qui s'exécute lorsqu'aucun article n'est trouvé dans la base de données. Le bloc `if` affiche un message et un lien vers le catalogue.
- #53** Débute une boucle `while` qui crée un nouveau tableau (`$items_new`) contenant les articles restants lus dans la base de données.

#57 Supprime dans la base de données tous les articles de la commande courante.

#61 Débute une boucle `for` qui insère dans la base de données tous les articles du tableau (`$items_new`, créé à la ligne 53), avec des numéros séquentiels d'articles. La boucle se termine en ligne 72.

#73 Stocke dans la session le nombre actuel d'articles.

#74 Les lignes 74 et 75 affichent le panier d'achat.

#77 Débute le bloc `case` qui s'exécute lorsque la valeur du bouton est Valider la commande. Le bloc exécute le troisième script de l'application : `ProcessOrder.php`.

#81 Débute le bloc `case` par défaut. Le bloc affiche le panier d'achat.

Écriture de *ProcessOrder.php*

Le troisième script de l'application de panier d'achat traite la commande lorsque le client la valide. Le script recueille les informations d'expédition, vérifie les informations saisies par le client, et affiche le formulaire récapitulatif. Lorsque le client clique sur un bouton dans le formulaire récapitulatif, le script accepte et traite la commande et affiche une page de confirmation. Le script est organisé en une série d'instructions `if/elseif`, qui exécutent du code selon le nom et la valeur du bouton enclenché par le client. Voici une vue d'ensemble de la structure du script :

```
si (aucun numéro de commande n'existe dans la session)
    Afficher un message disant que le panier est vide,
    et un lien qui ramène l'utilisateur à la page
    d'index du catalogue.
```

```
si (le script a été démarré à partir du panier)
    Afficher le formulaire d'informations d'expédition
sinon si (nom du bouton = "Summary")
```

1. Vérifier si le formulaire contient des champs vides. Si des champs vides sont trouvés, réafficher le formulaire.
2. Vérifier le format des champs de formulaire. Si des données invalides sont détectées, réafficher le formulaire.
3. Insérer les informations d'expédition dans la

base de données.

4. Afficher le formulaire récapitulatif.

sinon si (nom du bouton = "Ship")

1. Mettre à jour les quantités dans la base de données
2. Supprimer tous les articles dont la quantité vaut 0.
3. Renuméroter séquentiellement les articles
4. Réafficher le panier d'achat

sinon si (nom du bouton = "Final")

selon (valeur du bouton)

cas : "Poursuivre les achats"

Exécuter ShopCatalog.php

cas : Annuler la commande

Afficher la page Web d'annulation

Détruire session

cas : Valider la commande

Régler le statut de la commande sur validé

Traiter les informations de carte bancaire

Transmettre la commande pour qu'elle

soit honorée

Afficher la page Web de confirmation de

commande

Le Listing 15.13 présente le code de ProcessOrder.php – le troisième script de l'application de commande en ligne.

Listing 15.13 : Le script ProcessOrder de l'application de panier d'achat

```
<?php
/* Programme : ProcessOrder.php
 * Description :  Traite la commande lorsqu'elle est soumise.
 */
session_start();                                     #5
include("functions_main.inc");
if(!isset($_SESSION['order_number']))                #7
{
    echo "N° de commande introuvable<br>\n";
    header("Location: ShopCatalog.php");
    exit();
}
if(@$_GET['from'] == "cart")                           #13
{
    include("fields_ship_info.inc");
    include("single_form.inc");
    exit();
}
elseif(isset($_POST['Summary']))                       #19
{
```

```

foreach($_POST as $field => $value)                                #21
{
    if ($value == "")
    {
        $blanks[] = $field;
    }
}
if(isset($blanks))
{
    $message = "Les champs suivants sont vides.
                Merci d'entrer les informations requises : ";
    foreach($blanks as $value)
    {
        $message .="$value, ";
    }
    extract($_POST);
    include("fields_ship_info.inc");
    include("single_form.inc");
    exit();
}
foreach($_POST as $field => $value)                                #41
{
    if($field != "Summary")
    {
        if(ereg("name",$field))
        {
            if (!ereg("[A-Za-z' -]{1,50}$",$value))
            {
                $errors[] = "$value n'est pas un nom valide.";
            }
        }
        if(ereg("street",$field) or ereg("addr",$field) or
           ereg("city",$field))
        {
            if(!ereg("^[A-Za-z0-9.,' -]{1,50}$",$value))
            {
                $errors[] = "$value n'est pas une adresse ou ville
                               valide.";
            }
        }
        if(ereg("state",$field))
        {
            if(!ereg("[A-Za-z]", $value))
            {
                $errors[] = "$value n'est pas un état valide.";
            }
        }
        if(ereg("email",$field))
    }
}

```

```

if(!ereg("^.+@\.\.\.$", $value))
{
    $errors[] = "$value n'est pas un e-mail valide.";
}
}
if(ereg("zip", $field))
{
    if(!ereg("[0-9]{5,5}(\-[0-9]{4,4})?$", $value))
    {
        $errors[] = "$value n'est pas un code postal valide.";
    }
}
if(ereg("phone", $field))
{
    if(!ereg("[0-9](xX -){7,20}$", $value))
    {
        $errors[] = "$value n'est pas un n° de téléphone valide. ";
    }
}
if(ereg("cc_number", $field))
{
    $value = trim($value);
    $value = ereg_replace(' ', '', $value);
    $value = ereg_replace('-', '', $value);
    $_POST['cc_number'] = $value;
    if($_POST['cc_type'] == "visa")
    {
        if(!ereg("[4]{1,1}[0-9]{12,15}$", $value))
        {
            $errors[] = "$value n'est pas un n° Visa valide. ";
        }
    }
    elseif($_POST['cc_type'] == "mc")
    {
        if(!ereg("[5]{1,1}[0-9]{15,15}$", $value))
        {
            $errors[] = "$value n'est pas un n°
            Mastercard valid . ";
        }
    }
}
else
{
    if(!ereg("[3]{1,1}[0-9]{14,14}$", $value))
    {
        $errors[] = "$value n'est pas un n°
        American Express valide. ";
    }
}
}

```

```

    }
  }
  $$field = strip_tags(trim($value));
}
}
if(@is_array($errors))
{
  $message = "";
  foreach($errors as $value)
  {
    $message .= $value." Réessayez<br />";
  }
  include("fields_ship_info.inc");
  include("single_form.inc");
  exit();
}
/* Traite les données lorsque tous les champs sont corrects */
foreach($_POST as $field => $value)
{
  if(!ereg("cc_", $field) && $field != "Summary" )
  {
    $value = addslashes($value);
    $updates[] = "$field = '$value'";
  }
}
$update_string = implode($updates, ",");
$sql_ship = "UPDATE Customer_Order SET $update_string
            WHERE order_number='{$_SESSION['order_number']}'";
$cxn = connect_to_db("Vars.inc");
$result = mysqli_query($cxn, $sql_ship)
        or die(mysqli_error($cxn));
extract($_POST);
include("fields_summary.inc");
include("summary_page.inc");
}
elseif(isset($_POST['Ship']))
{
  include("fields_ship_info.inc");
  include("single_form.inc");
}
elseif(isset($_POST['Final']))
{
  switch ($_POST['Final'])
  {
    case "Poursuivre les achats":
      header("Location: ShopCatalog.php");
      break;
    case "Annuler la commande":

```

```

#include("fields_cancel.inc");
#include("cancel_message.inc");
unset($_SESSION['order_number']);
session_destroy();
exit();
break;
case "Valider la commande":                                #171
    $cxn = connect_to_db("Vars.inc");
    $sql = "UPDATE Customer_Order SET submitted='yes'
        WHERE order_number='{$_SESSION['order_number']}'";
    $result = mysqli_query($cxn,$sql)
        or die("Error: ".mysqli_error($cxn));
    #processCCInfo();                                       #177
    #sendOrder();
    #include("fields_accept.inc");                          #179
    #include("accept_message.inc");
    #email();                                              #181
    session_destroy();                                     #182
    break;
}
}
?>

```

Dans les explications suivantes, les numéros renvoient aux numéros de lignes du Listing 15.13 :

- #5** Début une session pour la commande en cours.
- #7** Début un bloc if qui s'exécute s'il n'y a pas de commande en cours. Affiche un message et un lien vers le catalogue.
- #13** Début un bloc if qui s'exécute lorsque l'utilisateur clique, dans le panier, sur le bouton Valider la commande. Le bloc affiche le formulaire d'informations d'expédition.
- #19** Début un bloc elseif qui s'exécute lorsque l'utilisateur clique sur le bouton nommé Summary (récapitulatif), c'est-à-dire le bouton libellé Poursuivre dans le formulaire d'informations d'expédition. Le bloc elseif traite les informations du formulaire d'informations d'expédition. Les lignes 21 à 132 vérifient les champs du formulaire. (Les champs de formulaire sont étudiés plus en détail au Chapitre 13.)

- #21 Les lignes 21 à 40 vérifient la présence de champs vides et réaffichent le formulaire si des champs vides sont trouvés.
- #41 Les lignes 41 à 132 contrôlent le format des données saisies par l'utilisateur. Si des formats invalides sont détectés, le formulaire est réaffiché, avec un message d'erreur.
- #134 Débute une boucle `foreach` qui crée un tableau `$update` contenant les informations d'expédition. Ce tableau est utilisé plus tard pour construire l'instruction SQL qui ajoute les informations d'expédition à la base de données.
- #136 Débute un bloc `if` qui s'exécute si le champ ne contient pas d'informations de carte bancaire. Cette application ne stocke pas les informations de carte bancaire dans la base de données. Par conséquent, si le formulaire est réaffiché, le client doit ressaisir ses informations de carte bancaire.
- #142 Crée une chaîne contenant les informations d'expédition.
- #143 Les lignes 143 à 147 créent et exécutent l'instruction SQL qui ajoute les informations d'expédition à la base de données.
- #148 Les lignes 148 à 150 affichent la page Web de récapitulatif.
- #152 Débute un bloc `elseif` qui s'exécute lorsque le bouton s'appelle Ship. Cette condition est vérifiée lorsque l'utilisateur clique sur le bouton Modifier infos livraison dans la page de récapitulatif. Le bloc affiche le formulaire d'informations d'expédition avec les informations d'expédition actuellement stockées dans la base de données.
- #157 Débute un bloc `elseif` qui s'exécute lorsque l'utilisateur clique sur un bouton nommé Final. Ces boutons sont affichés dans la page Web de récapitulatif.
- #159 Débute une instruction `switch` basée sur le bouton Final sur lequel le client a cliqué.
- #161 Débute le bloc `case` qui s'exécute lorsque la valeur du bouton Final est Poursuivre les achats. Le bloc exécute le script `ShopCatalog.php`, qui affiche la page d'index du catalogue.

- #164 Débute le bloc `case` qui s'exécute lorsque la valeur du bouton Final est Annuler la commande. Le bloc affiche une page Web d'annulation, en incluant deux fichiers, et détruit la session. Notez que les deux instructions incluent un symbole de commentaire (`#`) en début de ligne. Ces deux instructions sont en commentaires car la page Web d'annulation n'est pas fournie dans ce chapitre, pour des raisons d'espace. Vous devrez donc écrire une page d'annulation propre à votre processus de commande.

- #171 Débute le bloc `case` qui s'exécute lorsque la valeur du bouton Final est Valider la commande. Le bloc règle l'état de la commande avec `Submitted='yes'`.

- #177 Appelle une fonction qui traite les informations de carte bancaire. Je ne fournis pas cette fonction car elle dépend de la société de traitement de carte bancaire à laquelle vous ferez appel. Ce prestataire vous fournira toutes les informations nécessaires pour écrire la fonction. En général, la fonction envoie les informations de carte bancaire au prestataire, et reçoit en retour un code qui indique l'acceptation ou le rejet du paiement avec cette carte. Notez que l'instruction à un symbole de commentaire (`#`) en début de ligne, de façon qu'elle ne s'exécute pas réellement. Elle n'est là qu'à titre d'exemple d'instruction à employer.

- #178 Appelle une fonction qui envoie les informations de commande à la personne/service responsable de l'exécution et l'expédition de la commande. Cette fonction dépend de vos procédures internes. Cette instruction est en commentaire car je ne fournis pas la fonction.

- #179 Affiche une page de confirmation ou de rejet de commande en incluant deux fichiers. Les fichiers ne sont pas fournis, aussi les instructions inclues sont-elles en commentaires. Vous devrez écrire vos propres fichiers à inclure ici.

- #181 Appelle une fonction qui envoie un e-mail au client. Cet appel de fonction est en commentaire car je ne fournis pas la fonction `email`. Vous devrez écrire une fonction qui crée et envoie un message électronique. L'envoi d'un e-mail est présenté au Chapitre 13.

- #182** Détruit la session. Après avoir cliqué sur le bouton Valider la commande dans la page de récapitulatif, le client ne peut plus opérer de changement à la commande.

Création de l'application de panier d'achat : approche orientée objet

La programmation orientée objet nécessite la création et l'emploi d'objets pour mettre en place les fonctionnalités de l'application. Identifiez en premier lieu les objets nécessaires à l'application. Écrivez ensuite les classes qui définissent les objets, y compris les méthodes dont l'application a besoin. Une fois les objets prêts, on écrit le script qui crée et exploite les objets.

Développement des objets

L'application de panier d'achat doit afficher les produits du catalogue. Elle stocke les choix du client dans un panier d'achat. De même, elle stocke dans une base de données les informations d'expédition de la commande et les articles commandés. La liste d'objets ci-dessous reflète les tâches à réaliser :

- ✓ **Catalog** : La classe `Catalog` retourne et affiche les informations produits à la demande.
- ✓ **Database** : L'application stocke dans une base de données les informations sur les produits. La classe `Database` fournit le conteneur qui stocke les données.
- ✓ **Item** : Articles de la commande. Les articles sont stockés dans le panier et dans la base de données de commande. La classe `Item` stocke et lit les informations sur l'article.
- ✓ **ShoppingCart** : Le panier contient les articles sélectionnés par le client. Le client peut ajouter et supprimer des articles.
- ✓ **Order** : Les informations d'expédition et de carte de crédit de la commande doivent être associées aux articles de la commande. La classe `Order` stocke et lit les informations sur une commande stockée dans la base de données.
- ✓ **WebForm** : Un formulaire est utilisé pour recueillir les informations d'expédition et de carte de crédit du client. La classe

WebForm fournit le formulaire de l'application. Elle recueille et traite les informations saisies par le client.

- ✓ **WebPage** : La classe `WebPage` affiche une page Web qui inclut des informations issues de variables PHP. La classe `WebPage` est fréquemment utilisée dans ce livre, à chaque fois qu'une page Web doit être affichée.
- ✓ **Email** : L'application envoie un e-mail au client qui a passé commande, indiquant si la commande a été acceptée et fournissant d'autres informations sur la commande. La classe `Email` stocke et gère le message électronique.

Les sections suivantes étudient chaque classe en détail.

Écriture de la classe *Catalog*

La classe `Catalog` gère la connexion à la base de données où sont stockées les informations produits. Elle retourne ou affiche les informations produits à la demande. Cette classe a été développée au Chapitre 14. Pour cette application de panier d'achat, je lui ai ajouté deux méthodes (pour le code d'origine de la classe `Catalog`, voir le Listing 14.7). Ces nouvelles méthodes, `getName` et `getPrice`, sont décrites plus loin dans cette section.

Voici les méthodes de la classe `Catalog` :

- ✓ **Le constructeur** : Crée une connexion à une base de données MySQL. Le constructeur attend qu'on lui passe le nom du fichier qui contient les nom d'hôte, nom de compte et mot de passe nécessaires pour accéder à MySQL. L'instruction suivante crée un objet `Database` :

```
$db = new Database("Vars.inc");
```

- ✓ **useDatabase** : Sélectionne une base de données et stocke le nom de la base de données. La méthode s'attend à ce qu'on lui passe un nom de base de données. Elle vérifie si la base de données existe, et renvoie un message si elle n'existe pas.
- ✓ **getConnection** : Retourne la connexion établie et stockée dans le constructeur.
- ✓ **getName** : Retourne le nom du produit. Cette méthode attend qu'on lui passe un numéro de produit du catalogue. Cette

méthode est nouvelle dans ce chapitre. Le Listing 15.14 présente son code.

- ✓ `getPrice` : Retourne le prix. Cette méthode attend qu'on lui passe un numéro de produit du catalogue. Cette méthode est nouvelle dans ce chapitre. Le Listing 15.14 présente son code.

Le Listing 15.14 présente le code des méthodes `getName` et `getPrice`.

Listing 15.14 : Les nouvelles méthodes de la classe `Catalog`.

```
function getName($catalog_number)
{
    if(ereg("[0-9]*",$catalog_number))
    {
        $sql = "SELECT name,type FROM Food
WHERE catalog_number='$catalog_number'";
    }
    else
    {
        throw new Exception("$catalog_number n'est pas un
                                numéro d'article du catalogue.");
        exit();
    }
    if(!$result = $this->cxn->query($sql))
    {
        throw new Exception(mysqli_error($this->cxn));
        exit();
    }
    $name = $result->fetch_assoc();
    return "{$name['name']}" . " {$name['type']}";
}

function getPrice($catalog_number)
{
    if(ereg("[0-9]*",$catalog_number))
    {
        $sql = "SELECT price FROM Food
WHERE catalog_number='$catalog_number'";
    }
    else
    {
        throw new Exception("$catalog_number n'est pas un
                                numéro d'article du catalogue.");
        exit();
    }
    if(!$result = $this->cxn->query($sql))
```



```

{
    throw new Exception(mysqli_error($this->cxn));
    exit();
}
$price = $result->fetch_assoc();
return "{$price['price']}";
}

```

La méthode getName

La méthode `getName` retourne le nom du produit, sous forme nom-espace-type. Par exemple, dans cette application, la méthode retourne `Fuji Pommes` ou `Sanguine Oranges`.

La méthode vérifie que le numéro de produit passé contient uniquement des chiffres. Dans d'autres applications, le numéro de produit peut présenter un format différent, avec des lettres ou autres caractères. L'instruction `if` doit tester le format du numéro de produit de façon aussi détaillée que possible.

Si le format du numéro de produit est correct, une requête SQL est construite pour sélectionner les informations nécessaires dans la base de données. La requête est exécutée. Les informations retournées par la requête sont ajoutées à une chaîne, dans le format correct. Les informations formatées sont retournées. Voici comment appeler la méthode :

```
$product_name = $catalog->getName("1004");
```

où "1004" est le numéro du produit.

La méthode getPrice

La méthode `getPrice` retourne le prix du produit. Une requête SQL est construite et exécutée pour sélectionner le prix dans la base de données. La méthode retourne le prix. Voici la syntaxe pour appeler la méthode :

```
$product_price = $catalog->getPrice($catalog_number);
```

Écriture de la classe Item

La classe Item est une classe fondamentale. Articles de la commande. L'objet Item stocke et lit les informations sur un article choisi par le client.

Les propriétés

Les propriétés de Item stockent les informations sur l'article.

```
private $catalog_number;  
private $quantity;  
private $name;  
private $price;
```

La première propriété est le numéro nécessaire pour localiser l'article dans la base de données du catalogue. La deuxième propriété est la quantité saisie pour l'article par le client. Les propriétés restantes sont le nom et le prix de l'article, informations obtenues à partir du catalogue.

Le code

Le Listing 15.15 présente le code de la classe Item. Les détails sur chaque méthode sont présentés à la suite du listing.

Listing 15.15 : La classe Item.

```
<?php  
/* Classe : Item  
 * Description : Représente un article de la commande.  
 */  
class Item  
{  
    private $catalog_number;  
    private $quantity;  
    private $name;  
    private $price;  
  
    function __construct($cat_no,$quantity)  
    {  
        if(is_string($cat_no) && is_numeric($quantity))  
        {
```

```

        $this->quantity = $quantity;
        $this->catalog_number = $cat_no;
        $cat = new Catalog("Vars.inc");
        $cat->selectCatalog("OnlineOrders");
        $this->name = $cat->getName($cat_no);
        $this->price = $cat->getPrice($cat_no);
    }
    else
    {
        throw new Exception("Le paramètre n'est pas un numéro
            d'article ni une quantité valides");
    }
}
,
function getCatalogNumber()
{
    return $this->catalog_number;
}

function getQuantity()
{
    return $this->quantity;
}

function getPrice()
{
    return $this->price;
}

function getName()
{
    return $this->name;
}
}
?>

```

Le constructeur

Le constructeur recueille et stocke les informations de l'article. Le numéro de produit et la quantité sont passés lors de la création d'un nouvel objet `Item`. Le constructeur stocke le numéro de produit et la quantité dans les propriétés de l'objet `Item`. Il lit les deux propriétés restantes dans la base de données du catalogue et les stocke dans les propriétés de `Item`.

Voici comment créer un objet Item (un article) :

```
$item1 = new Item(5007,3)
```

getCatalogNumber, getQuantity, getPrice, getName

Ces méthodes retournent les informations correspondantes de l'article. Voici comment les utiliser :

```
$price = getPrice();
```

Écriture de la classe ShoppingCart

Le panier est un composant majeur de l'application de panier d'achat. Il contient les articles sélectionnés par le client.

Les propriétés

Les propriétés du panier stockent les articles du panier, avec les informations nécessaires au panier pour être affiché correctement.

```
private $items = array();  
private $message;  
private $n_items = 0;
```

La première propriété est un tableau d'objets qui contient les articles actuellement dans le panier. La deuxième propriété est un message qui apparaît lors de l'affichage du panier. La troisième propriété est le nombre d'articles actuellement dans le panier.

Le code

Le Listing 15.16 présente le code de la classe ShoppingCart. Chaque méthode est étudiée en détail à la suite du listing.

Listing 15.16 : La classe ShoppingCart.

```
<?php  
/* Classe : ShoppingCart  
 * Description : Crée un panier d'achat (une structure  
 * contenant des articles).
```

```

*/
class ShoppingCart
{
    private $items = array();
    private $message;
    private $n_items = 0;

    function __construct()
    {
        if(isset($_SESSION['items']))
        {
            $this->items = $_SESSION['items'];
            $this->n_items = sizeof($this->items);
        }
        $this->message = "Votre panier contient {$this->n_items} articles.";
    }

    function addItem(Item $item)
    {
        $this->items[] = $item;
        $_SESSION['items'] = $this->items;
        $this->n_items++;
        $this->message = "Votre panier contient {$this->n_items} articles.";
    }

    function getAllItems()
    {
        return $this->items;
    }

    function getMessage()
    {
        return $this->message;
    }

    function displayCart($file_fields,$file_page)
    {
        if(is_string($file_fields) ||
           is_string($file_page))
        {
            include($file_fields);
            include($file_page);
        }
        else
        {
            throw new Exception("Requiert deux noms de fichiers valides");
        }
    }
}

```



```

function updateCart($new_array)
{
    if(is_array($new_array))
    {
        foreach($new_array as $field => $value)
        {
            if(ereg("item",$field) && $value > 0)                #51
            {
                $cat_no = substr($field,4);
                $items_new[] = new Item($cat_no,$value);
            }
        }
        $this->items = @$items_new;
        $_SESSION['items'] = $this->items;
        $this->n_items = sizeof($this->items);
        $this->message = "Votre panier contient {$this->n_items} articles.";
    }
    else
    {
        throw new Exception("Le paramètre n'est pas un tableau");
    }
}
}
?>

```

Le constructeur

Le constructeur définit les trois propriétés. Les valeurs par défaut définissent un tableau vide et un nombre d'articles égal à 0. Le constructeur recherche un tableau d'articles dans la session. S'il le trouve, il le stocke dans la propriété `$items` et règle le nombre d'articles sur la grandeur du tableau. Si aucun article n'est stocké dans la session, les propriétés conservent les paramètres par défaut, c'est-à-dire pour un panier vide. La troisième propriété, `$message`, contient une chaîne de texte indiquant le nombre d'articles dans le panier.

L'objet est créé sans passage d'argument, comme ceci :

```
$cart = new ShoppingCart();
```

Lorsque le panier est créé, il est soit nouveau, et donc vide, soit il contient les articles stockés dans la session.

addItem

La méthode `addItem` ajoute un élément au panier. Elle s'attend à recevoir un objet `Item`. Elle ajoute l'article au tableau des articles et stocke le nouveau tableau dans la variable de session. Cette méthode incrémente aussi le nombre d'articles stockés et met à jour le message pour indiquer le nouveau nombre d'articles. Cette méthode s'emploie ainsi :

```
$cart->addItem($item5);
```

où la variable `$item5` contient un objet `item`.

getAllItems, getMessage

Les méthodes `getAllItems` et `getMessage` lisent les propriétés spécifiées. `getAllItems` retourne le tableau d'objets `Item` stocké dans les propriétés du panier. `getMessage` retourne le message stocké. Aucune de ces deux méthodes n'attend d'argument.

displayCart

La méthode `displayCart` affiche le panier dans une page Web. Les noms des fichiers fournissant les champs et définissant la page doivent être passés à la méthode. Vous pouvez utiliser les méthodes comme ceci :

```
$cart = new ShoppingCart();
$cart->displayCart("fields_cart-oo.inc", "table_page.inc");
```

updateCart

La méthode `updateCart` met à jour un panier existant. Elle attend un tableau contenant les informations sur tous les articles devant être dans le panier actualisé. La méthode remplace le tableau d'articles existant par le nouveau tableau d'articles créé à partir des informations passées à la méthode.

Le tableau passé à la méthode doit contenir des clés au format `itemnnnn` où `nnnn` est le numéro de produit d'un article. La valeur est la quantité demandée pour l'article. Voici un exemple de ce qu'un tableau peut contenir :

```
$item_array[item1003] = 1  
$item_array[item27] = 2.5  
$item_array[item673] = 1.7
```

Le tableau `$_POST` (qui est envoyé lorsque l'utilisateur clique sur le bouton d'envoi dans la page du panier d'achat) contient des éléments similaires, comme :

```
$item_array[item1003] = 1  
$item_array[item27] = 2.5
```

Pour chaque élément du tableau, la méthode extrait le numéro de produit à partir de la clé, passe le numéro de produit et la quantité pour créer un objet `Item`, et ajoute le nouvel objet à un tableau d'objets. Lorsque tous les éléments ont été ajoutés au nouveau tableau d'objets, celui-ci est stocké dans la propriété d'objet `$items` et dans la variable de session. Le nombre d'articles est incrémenté et stocké.

Écriture de la classe Database

La classe `Database` fournit la connexion à la base de données où sont stockées les informations sur le client. Cette classe a été développée au Chapitre 12. Pour en étudier le code, voir le Listing 12.4.

Voici les méthodes fournies par la classe `Database` :

- ✓ **Le constructeur** : Crée une connexion à une base de données MySQL. Le constructeur attend qu'on lui passe un nom de fichier contenant les nom d'hôte, nom de compte et mot de passe nécessaires pour accéder à MySQL. Un objet `Database` est créé avec l'instruction suivante :

```
$db = new Database("Vars.inc");
```

- ✓ **useDatabase** : Sélectionne une base de données et stocke le nom de la base de données. La méthode s'attend à ce qu'on lui passe un nom de base de données. Elle vérifie si la base de données existe, et renvoie un message si elle n'existe pas.
- ✓ **getConnection** : Retourne la connexion établie et stockée dans le constructeur.

Écriture de la classe Order

La commande contient toutes les informations nécessaires pour mener au bout les achats du client. Elle contient les informations d'expédition et de carte de crédit, ainsi que les informations sur chaque article commandé. Les informations sur la commande sont stockées dans une base de données.

Les propriétés

Les propriétés de Order stockent les informations sur la commande.

```
private $order_number;
private $cxn;
private $table;
```

La première propriété est le numéro nécessaire pour identifier la commande dans la base de données. Les deux propriétés restantes contiennent les informations nécessaires pour accéder à la commande dans la base de données.

Le code

Le Listing 15.17 présente le code de la classe Order. À la suite du listing, vous trouverez une étude de chaque méthode.

Listing 15.17 : La classe Order.

```
<?php
/* Classe : Order
 * Description : Classe contenant les commandes.
 */

class Order
{
    private $order_number;
    private $order_info;
    private $order_items=array();
    private $cxn;
    private $table;

    function __construct(mysql $cxn,$table)
    {
```

```

$this->cxn = $cxn;
if(is_string($table))
{
    $this->table = $table;
}
else
{
    throw new Exception("$table n'est pas un nom de table valide.");
}
}

function createOrder()
{
    $today = date("Y-m-d");
    $sql = "INSERT INTO $this->table
            (order_date) VALUES ('$today')";
    if($result = $this->cxn->query($sql))
    {
        $this->order_number = $this->cxn->insert_id;
        $_SESSION['order_number'] = $this->order_number;
    }
    else
    {
        throw new Exception("Base de données inaccessible.
                             Réessayez ultérieurement");
    }
}

function getOrderNumber()
{
    return $this->order_number;
}

function addCart(ShoppingCart $cart)
{
    foreach($cart->getAllItems() as $n => $item)
    {
        $cat_no = $item->getCatalogNumber();
        $quantity = $item->getQuantity();
        $price = $item->getPrice();
        $sql = "INSERT INTO Order_Item
                (order_number,catalog_number,
                 quantity,item_number,price)
                VALUES
                ($this->order_number,$cat_no,
                 $quantity,($n+1),$price)";
        $result = $this->cxn->query($sql);
    }
}

```



```

    }

function selectOrder($order_number)
{
    if(is_int($order_number))
    {
        $this->order_number = $order_number;
    }
    else
    {
        throw new Exception("$order_number n'estpas un entier.");
    }
}

function getOrderInfo()
{
    $sql = "SELECT * FROM $this->table
           WHERE order_number='$this->order_number'";
    if($result = $this->cxn->query($sql))
    {
        return $result->fetch_assoc();
    }
    else
    {
        throw new Exception("Base de données inaccessible.
                             Réessayez ultérieurement");
    }
}

function getItemInfo()
{
    $sql = "SELECT item_number,catalog_number,quantity,price
           FROM order_item
           WHERE order_number='$this->order_number'";
    if($result = $this->cxn->query($sql))
    {
        $n=1;
        while($row=$result->fetch_assoc())
        {
            foreach($row as $field => $value)
            {
                $item[$n][$field] = $value;
            }
            $cat = new Catalog("Vars.inc");
            $cat->selectCatalog("OnlineOrders");
            $item[$n]['name'] = $cat->getName($item[$n]['catalog_number']);
            $n++;
        }
    }
}

```

```

        return $item;
    }
    else
    {
        throw new Exception("Base de données inaccessible.
                               Réessayez ultérieurement");
    }
}

function updateOrderInfo($data)
{
    if(!is_array($data))
    {
        throw new Exception("Les données doivent être dans un tableau.");
        exit();
    }
    $sql = "UPDATE $this->table SET ";
    foreach($data as $field => $value)
    {
        if(ereg("ship",$field) || $field == "phone"
            || $field == "email")
        {
            $data_array[] = "$field='$value'";
        }
    }
    $sql .= implode($data_array,',');
    $sql .= "WHERE order_number='".$this->order_number'";
    if(!$result = $this->cxn->query($sql))
    {
        throw new Exception("Base de données inaccessible.
                               Réessayez ultérieurement");
    }
    return true;
}

function displayOrderSummary($field_info,$field_page)
{
    $data = $this->getOrderInfo();
    $items = $this->getItemInfo();
    extract($data);
    if(is_string($field_info) and is_string($field_page))
    {
        include($field_info);
        include ($field_page);
    }
    else
    {
        throw new Exception("Les paramètres ne sont pas des

```

```

        noms de fichiers valides.");
    }
}
?>

```

Le constructeur

Le constructeur stocke les informations nécessaires pour se connecter à la base de données. Il s'attend à ce qu'on lui passe une connexion et un nom de table. La connexion est une connexion à une base de données fournie par un objet Database. Le constructeur stocke les informations dans les propriétés de Order.

Voici comment créer un objet Order :

```

$db = new Database("Vars.inc");
$db->selectDatabase("OnlineOrders");
$order = new Order($db->getConnection(),"Customer_Order");

```

createOrder

La méthode `createOrder` insère une nouvelle commande dans la base de données de commandes, à l'aide d'une requête SQL. La date du jour est stockée. C'est la fonction MySQL `AUTO_INCREMENT` qui génère le numéro de commande. Le numéro de la nouvelle commande est stocké dans la propriété de la commande et dans une variable de session. La méthode retourne `true` si la création de la commande réussit.

selectOrder

La méthode `selectOrder` règle le numéro de commande dans Order sur le numéro de commande qui lui est passé. Toute information lue plus tard dans la base de données le sera en se basant sur la propriété numéro de commande.

addCart

La méthode `addCart` ajoute à la commande les articles du panier. La méthode s'attend à ce qu'on lui passe un objet `ShoppingCart`. La méthode utilise une boucle `foreach` pour traverser tous les articles du panier. Les informations à partir de chaque objet article sont utilisées pour construire une requête SQL qui insère l'article dans la table `order_item`.

getOrderInfo

La méthode `getOrderInfo` retourne un tableau associatif contenant toutes les informations issues de la table `customer_order`. Il contient le nom et l'adresse, le téléphone, la date de création et d'autres informations liées à la commande. Elle crée une requête SQL `SELECT` qui sélectionne toutes les informations. Aucun paramètre n'est attendu.

getItemInfo

La méthode `getItemInfo` retourne un tableau multidimensionnel contenant toutes les informations de tous les articles de la commande. Les informations pour article du tableau incluent les champs de la table `order_item`, plus le nom du produit lu dans le catalogue.

La méthode exécute une requête qui récupère toutes les informations de la table `order_item`. Une boucle `while` traite tour à tour chaque ligne d'article. Une boucle `foreach` s'exécute pour chaque ligne, ajoutant chaque champ à l'élément de tableau, d'où l'obtention d'un tableau multidimensionnel. Une fois terminée la boucle `foreach`, la méthode récupère le nom du produit dans le catalogue et l'ajoute au tableau.

updateOrderInfo

La méthode `updateOrderInfo` actualise les informations d'expédition dans la table `customer_order` de la base de données. La méthode attend un tableau d'informations d'expédition, comme le tableau `$_POST` venant du formulaire d'informations d'expédition. Une requête SQL `UPDATE` est créée à partir des données du tableau `$data`.

La requête est construite avec une section d'ouverture et des phrases pour chaque champ, comme `ship_name=John Smith`. Chaque

phrase actualisée est ajoutée à un nouveau tableau, `$data_array`. Une fois que tous les champs ont été traités et ajoutés au nouveau tableau, la fonction `implode` est utilisée pour convertir le tableau en une chaîne adaptée à l'usage dans la requête. La clause `WHERE` est alors ajoutée à la fin de la requête.

La méthode est utilisable dans une instruction de ce genre :

```
$order->updateOrderInfo($_POST);
```

`updateOrderInfo` retourne `true` lorsque les informations sont actualisées avec succès.

displayOrderSummary

La méthode `displayOrderSummary` affiche un récapitulatif de la commande. Les noms des deux fichiers qui définissent la page de récapitulatif sont passés. La page de récapitulatif est affichée en incluant les deux fichiers.

Écriture de la classe WebForm

L'objet `WebForm` est utilisé pour afficher et traiter le formulaire d'informations d'expédition. La classe `WebForm` est étudiée au Chapitre 13. Pour son code, voir le Listing 13.6.

Voici les méthodes de classe `WebForm` exploitées par l'application de panier d'achat :

- ✓ **Le constructeur** : Stocke les propriétés nécessaires pour afficher correctement le formulaire. Deux fichiers – un fichier de données et un fichier qui définit l'aspect – sont requis. Les deux noms de fichiers sont passés lors de la création de l'objet `WebForm`, et stockés dans deux propriétés. Le passage des données des champs de formulaire n'est pas obligatoire, ce qui donne alors des champs de formulaires vides. Il est possible de créer l'objet à l'aide d'une de ces instructions :

```
$form = new WebForm("fichier.inc", "fichier2.inc", $_POST);
$form = new WebForm("fichier1.inc", "fichier2.inc");
```

- ✓ **displayForm** : Cette méthode affiche le formulaire. Elle extrait les données de la propriété `$data`. Un `@` est utilisé pour supprimer les messages d'erreur, pour que le formulaire

puisse être affiché sans aucune donnée. Le formulaire est affiché en incluant les deux fichiers qui le définissent. Ces deux fichiers peuvent définir tout type de formulaire, avec les champs et éléments à votre choix. Pour cette application, j'utilise les fichiers décrits plus haut dans ce chapitre – `fields_ship_info-oo.inc` et `single_form.inc` – qui définissent le formulaire d'informations d'expédition.

- ✓ `checkForBlanks` : Vérifie chaque champ du formulaire pour voir s'il contient des informations. Si la méthode trouve des champs vides invalides, elle retourne un tableau contenant leurs noms.
- ✓ `verifyData` : Cette méthode vérifie chaque champ pour garantir que les informations soumises sont dans un format raisonnable. Par exemple, vous savez que "salut" n'est pas un format raisonnable pour un code postal. Cette méthode vérifie les informations de champs spécifiques à l'aide d'expressions régulières définissant les informations autorisées dans ce champ. Si des données invalides sont détectées dans les champs, la méthode retourne un tableau contenant des messages identifiant les problèmes.
- ✓ `trimData, stripTagsFromData` : Une fonction PHP est appliquée à chaque valeur de la propriété `$data`. Les valeurs résultantes sont stockées dans `$data`. La fonction `trim` supprime les espaces initiaux et terminaux d'une chaîne. La fonction `strip_tags` supprime les balises HTML de la chaîne, ce qui est important pour la sécurité.

Écriture de la classe *WebPage*

Dans ce livre, la classe `WebPage` est utilisée à chaque fois qu'il faut afficher une page Web. L'objet `WebPage` est une page Web qui affiche des informations issues de variables PHP, ainsi que du code HTML. Le code qui définit la page Web est dans un fichier séparé. Pour afficher la page Web, il faut inclure ce fichier. La classe `WebPage` a été développée au Chapitre 12. Son code est présenté par le Listing 12.6.

La classe `WebPage` comporte les méthodes suivantes :

- ✓ **Le constructeur** : Stocke un nom de fichier et les données nécessaires pour afficher la page. S'attend à recevoir le nom du fichier qui contient le code HTML définissant la page Web.

S'attend également à ce qu'on lui passe un tableau contenant toutes données devant être affichées dans la page. Pour créer un objet `WebPage`, utilisez une instruction semblable à celle-ci :

```
$web_page1 = new WebPage("define_page.inc", $data);
```

- ✓ `displayPage` : Affiche la page Web. La méthode extrait les informations du tableau `$data` et inclut le fichier HTML.

Écriture de la classe `Email`

Lorsqu'un client passe commande avec succès, l'application envoie un message de confirmation à l'adresse e-mail fournie par le client. (Pour des détails sur la classe `Email`, voir le Chapitre 13 ; pour son code, voir le Listing 13.9.)

Écriture du script de l'application de panier d'achat

Après écriture du code de classe nécessaire à l'application de panier d'achat, passons au script. Il crée et utilise les objets pour mettre en place la fonctionnalité de l'application. Le script de l'application est organisé sous forme d'instructions `if` imbriquées. Les instructions `if` s'exécutent selon le nom et la valeur du bouton d'envoi. Le Tableau 15.3 présente les boutons utilisés dans l'application. Le tableau indique le texte visible sur le bouton lui-même, la page Web où le bouton apparaît, et le nom donné au bouton dans la balise `input`.

Le pseudo-code ci-dessous donne une vue d'ensemble du script de l'application :

```
si (nom du bouton = Products)
    Afficher la page des produits de la catégorie
    sélectionnée.
sinon si (nom du bouton = Cart)
    si (Bouton = Actualiser le panier)
        1. Mettre à jour les articles et quantités dans
           le panier.
        2. Afficher le panier.
    sinon si (Bouton = Ajouter les articles au panier)
        1. Ajouter au panier les articles sélectionnés.
        2. Afficher le panier.
sinon si (nom du bouton = Ship)
```

Tableau 15.3 : Boutons affichés par l'application de panier d'achat.

Nom affiché	Page Web	Nom
Autre catégorie	Index du catalogue	Products
Poursuivre les achats	Produits du catalogue	(sans nom)
Poursuivre les achats	Panier d'achat	(sans nom)
Poursuivre les achats	Récapitulatif	(sans nom)
Ajouter les articles au panier	Produits du catalogue	Cart
Précédent	Produits du catalogue	Cart
Suivant	Produits du catalogue	Cart
Valider la commande	Panier d'achat	Ship
Actualiser le panier	Panier d'achat	Cart
Poursuivre	Formulaire d'informations d'expédition	Summary
Modifier infos livraison	Récapitulatif	Ship
Valider la commande	Récapitulatif	Final
Annuler la commande	Récapitulatif	Final

1. Si commande en cours existe, lire le numéro de commande dans la variable de session. Si ce n'est pas le cas, créer une nouvelle commande.
2. Afficher le formulaire d'informations d'expédition.
sinon si (nom du bouton = Summary)
 1. Vérifier si le formulaire d'information contient des champs vides. Si des champs vides sont trouvés, réafficher le formulaire, avec un message d'erreur.
 2. Vérifiez le format des informations du formulaire. Si des informations invalides sont trouvées, réafficher le formulaire, avec message d'erreur
3. Ajouter les informations d'expédition dans la base de données.
4. Ajouter les articles du panier à la base de données.
5. Afficher le formulaire de récapitulatif.
sinon si (nom du bouton = Final)
 - si (Bouton = "Valider la commande")
 1. Mettre à jour le statut de la commande (elle est validée).
 2. Traiter les informations de carte bancaire.
 3. Envoyer la commande pour exécution et expédition.
 4. Envoyer e-mail de confirmation au client.

```

5. Afficher page Web de confirmation.
sinon si (Bouton = "Annuler la commande")
    1. Mettre à jour le statut de la commande (elle
       est annulée).
    2. Afficher la page Web d'annulation.
    3. Détruire la session.
sinon
    Afficher la page d'index du catalogue

```

Le programme de l'application crée les objets, et utilise leurs méthodes pour réaliser les tâches décrites dans le pseudo-code ci-dessus. Le Listing 15.18 présente le code du script de l'application.

Listing 15.18 : Le script de l'application de panier d'achat.

```

<?php
/* Programme : Orders-oo.php
 * Description :   Gère toutes les fonctions de
 *                l'application de commande en ligne. Le bouton
 *                d'envoi de formulaire est testé pour
 *                déterminer quelle section du programme il faut
 *                exécuter.
 */
require_once("Item.class");
require_once("Catalog.class");
require_once("ShoppingCart.class");
require_once("WebForm.class");
require_once("WebPage.class");
require_once("Order.class");
require_once("Database.class");
include("functions_main.inc");
session_start();                                     #16
if(isset($_POST['Products']) && isset($_POST['interest']))
{
    try
    {
        $catalog = new Catalog("Vars.inc");
        $catalog->selectCatalog("OnlineOrders");
        $catalog->displayAllofType($_POST['interest'],2);
    }
    catch(Exception $e)
    {
        echo $e->getMessage();
        exit();
    }
}
}
elseif(isset($_POST['Cart']))                         #31

```

```

{
    $cart = new ShoppingCart();
    if($_POST['Cart'] == "Actualiser le panier") #34
    {
        try
        {
            $cart->updateCart($_POST);
        }
        catch(Exception $e)
        {
            echo $e->getMessage();
            exit();
        }
    }
    elseif($_POST['Cart'] == "Ajouter les articles au panier") #46
    {
        foreach($_POST as $field => $value) #48
        {
            if(ereg("item",$field) && $value > 0)
            {
                try
                {
                    $cat_no = substr($field,4);
                    $item = new Item($cat_no,$value); #54
                    $cart->addItem($item);
                }
                catch(Exception $e)
                {
                    echo $e->getMessage();
                    exit();
                }
            }
        }
    }
    try
    {
        $cart->displayCart("fields_cart-oo.inc",
            "table_page-oo.inc"); #69
    }
    catch(Exception $e)
    {
        echo $e->getMessage();
        exit();
    }
}
elseif(isset($_POST['Ship'])) #77
{
    try

```



```

{
    $db = new Database("Vars.inc");
    $db->useDatabase("OnlineOrders");
    $order = new Order($db->getConnection(),
                      "Customer_Order");
    if(isset($_SESSION['order_number'])) #85
    {
        $order->selectOrder($_SESSION['order_number']);
    }
    else
    {
        $order->createOrder();
    }
    $ord = $order->getOrderNumber();
    $info = $order->getOrderInfo();
    $form = new WebForm("single_form.inc",
                      "fields_ship_info-oo.inc",$info);
    $form->displayForm();
}
catch(Exception $e)
{
    echo $e->getMessage();
    exit();
}
}
elseif(isset($_POST['Summary'])) #105
{
    try
    {
        $form = new
            WebForm("single_form.inc",
                  "fields_ship_info-oo.inc",$_POST);
        $blanks = $form->checkForBlanks();
    }
    catch(Exception $e)
    {
        echo $e->getMessage();
    }
    if(is_array($blanks)) #118
    {
        $GLOBALS['message'] = "Les champs suivants sont
                               vides. Merci d'entrer les informations
                               requises : ";
        foreach($blanks as $value)
        {
            $GLOBALS['message'] .="$value, ";
        }
        $form->displayform();
    }
}

```

```

        exit();
    }
    $form->trimData();
    $form->stripTagsFromData();
    try
    {
        $errors = $form->verifyData();
    }
    catch(Exception $e)
    {
        echo $e->getMessage();
    }
    if(is_array($errors))
    {
        $GLOBALS['message'] = "";
        foreach($errors as $value)
        {
            $GLOBALS['message'] .="$value<br> ";
        }
        $form->displayForm();
        exit();
    }
    try
    {
        $db = new Database("Vars.inc");
        $db->useDatabase("OnlineOrders");
        $order =
            new Order($db->getConnection(),"Customer_Order");
        $order->selectOrder($_SESSION['order_number']);

        // Ajoute les infos de livraison dans la base de données
        $order->updateOrderInfo($_POST);

        // Ajoute les articles dans la base
        $cart = new ShoppingCart();
        $order->addCart($cart);

        // affiche le formulaire de récapitulatif
        $order->displayOrderSummary("fields_summary-oo.inc",
                                   "summary_page.inc");
    }
    catch(Exception $e)
    {
        echo $e->getMessage();
    }
}
elseif(isset($_POST['Final']))
{

```

```

if($_POST['Final'] == "Valider la commande") #176
{
    $db = new Database("Vars.inc");
    $db->useDatabase("OnlineOrders");
    $order = new Order($db->getConnection(),
        "Customer_Order");
    $order->selectOrder($_SESSION['order_number']);
    if(processCC()) #180
    {
        $order->setSubmitted(); #182
        $order->sendToShipping(); #183
        $order->sendEMailtoCustomer(); #184
        $confirm = new webPage("confirm_page.inc",$data);
        $confirmpage->displayPage(); #186
    }
    else #188
    {
        $order->cancel();
        $unapp = new webPage("not_accepted_page.inc",$data);
        $unapp->displayPage();
        unset($_SESSION['order_number']);
        unset($_SESSION);
        session_destroy();
    }
}
else #198
{
    $order->cancel();
    $cancel = new webPage("cancel.inc",$data);
    $cancel->displayPage();
    unset($_SESSION['order_number']);
    unset($_SESSION);
    session_destroy();
}
}
else #208
{
    $catalog = new Catalog("Vars.inc");
    $catalog->selectCatalog("OnlineOrders");
    $catalog->displayCategories();
}
?>

```

Dans les explications suivantes, les numéros renvoient aux numéros de lignes du Listing 15.18 :

- #16 Ouvre une session existante ou, si aucune session n'existe, ouvre une nouvelle session.
- #17 Débute un bloc `if` qui s'exécute si le client clique sur le bouton `Products` alors qu'une catégorie de produits est sélectionnée. Le bloc affiche la page des produits pour la catégorie sélectionnée.
- #31 Débute un bloc `elseif` qui s'exécute lorsque l'utilisateur clique sur le bouton `Cart`.
- #33 Crée un objet `ShoppingCart`. Le constructeur de classe `ShoppingCart` recherche des articles dans la session. Si des articles existants sont trouvés dans la variable de session `$items`, ils sont chargés dans l'objet `Cart` (panier). Si `$items` n'est pas trouvé dans la session, le panier est vide.
- #34 Débute un bloc `if` qui s'exécute lorsque l'utilisateur clique sur le bouton `Actualiser le panier`. Le bloc met à jour les quantités et articles dans le panier d'achat.
- #46 Débute un bloc `elseif` qui s'exécute lorsque l'utilisateur clique sur le bouton `Ajouter les articles au panier`. La ligne 48 débute une boucle `foreach` qui traverse le tableau `$_POST`. Lorsqu'une clé d'élément de tableau contient `item` et que la quantité est supérieure à 0, le numéro de produit dans le catalogue est extrait du nom de champ (ligne 54) et un objet `Item` (article) est créé (ligne 55) et ajouté au panier (ligne 56).
- #68 Les lignes 68 à 69 affichent le panier d'achat.
- #77 Débute un bloc `elseif` qui s'exécute lorsque l'utilisateur clique sur le bouton `Ship`. Ce bloc affiche le formulaire d'informations d'expédition.
- #81 Les lignes 81 à 84 créent un objet `Order` (commande).
- #85 Débute une instruction `if/else` qui définit le numéro de commande. Si le numéro de commande est trouvé dans une variable de session, la propriété concernée de l'objet `Order` est réglée sur le numéro de commande de la session. Si ce n'est pas le cas, une nouvelle commande est créée, avec un nouveau numéro de commande.

- #93 Les lignes 93 à 97 affichent le formulaire d'informations d'expédition, avec les informations d'expédition de cette commande, extraites de la base de données.
- #105 Débute un bloc `elseif` qui s'exécute lorsque l'utilisateur clique sur le bouton Récapitulatif. Ce bloc traite l'information du formulaire d'informations d'expédition.
- #109 Les lignes 109 à 128 regardent si le tableau `$_POST` contient des champs de formulaire vierges. Si des champs vides sont trouvés, le formulaire est réaffiché, avec un message d'erreur.
- #129 Les lignes 129 à 148 contrôlent le format des données dans les éléments du tableau `$_POST`. Si des données invalides sont détectées, le formulaire est réaffiché avec un message d'erreur.
- #151 Les lignes 151 à 158 ajoutent les informations d'expédition dans la base de données. Cette ligne n'est pas atteinte tant que les informations d'expédition ne sont pas validées.
- #161 Les lignes 161 à 162 ajoutent les articles du panier dans la base de données des commandes.
- #165 Les lignes 165 à 166 lisent les données dans la base de données et affichent la page Web de récapitulatif.
- #173 Débute un bloc `elseif` qui s'exécute lorsque l'utilisateur clique sur le bouton Final. Ce bloc traite la commande finale. Le client peut annuler ou valider la commande. Lorsqu'il valide la commande, celle-ci est approuvée ou rejetée.
- #175 Débute un bloc `if` qui s'exécute lorsque l'utilisateur clique sur le bouton Valider la commande. Le bloc crée un objet `Order`.
- #182 Débute un bloc `if` qui s'exécute lorsque le société de traitement de carte bancaire approuve le paiement. La fonction `processCC` n'est pas fournie dans ce chapitre. Vous devrez écrire cette fonction vous-même car elle diffère d'un prestataire de traitement de carte bancaire à l'autre. Votre prestataire vous indiquera ce que doit réaliser la fonction. En général, on transmet les informations de carte bancaire vers le serveur du prestataire, qui

traite le paiement par carte. Un code indiquant la réussite ou l'échec sera retourné vers votre fonction.

- #184 Règle sur approuvé l'état de la commande, dans la base de données.
- #185 Transmet la commande, pour exécution et expédition. Cette méthode n'est pas incluse dans la classe `Order` de ce chapitre. Vous devrez l'écrire, pour le suivi de commande selon les procédures de votre entreprise.
- #187 Les lignes 187 et 188 affichent une page Web de confirmation. Vous devez écrire `confirmation_page.inc` pour inclure les informations que vous désirez fournir à vos clients, comme la date d'expédition, le mode de livraison, etc.
- #190 Débute un bloc `else` qui s'exécute lorsque le paiement par carte n'est pas approuvé. Le bloc annule la commande, affiche une page de commande refusée, et détruit la session. Le fichier `not_accepted_page.inc` n'est pas fourni dans ce chapitre. Vous devrez écrire l'HTML de ce fichier pour afficher les informations que vous désirez présenter au client en cas de refus.
- #200 Débute un bloc `else` qui s'exécute lorsque le client clique sur le bouton Annuler la commande. Le bloc annule la commande, affiche une page d'annulation, et détruit la session. Vous devrez écrire le fichier `cancel_page.inc`.
- #210 Débute un bloc `else` qui s'exécute lorsque l'utilisateur ne clique pas sur un bouton ou clique sur un bouton sans nom. Le bloc affiche la page d'index du catalogue.

Ajout de caractéristiques à l'application

L'application de panier d'achat présentée dans ce chapitre fournit la fonctionnalité élémentaire pour les commandes en ligne. Cependant, il est possible d'ajouter d'autres fonctions. Voici quelques fonctions courantes dans les applications de panier d'achat et qui ne sont pas mises en place dans ce chapitre :

- ✓ **Identification (login)** : Comme évoqué au début de ce chapitre, de nombreux commerces en ligne réclament au client de s'identifier avant de pouvoir commander. Pour ajouter cette

caractéristique, servez-vous de l'application d'identification du Chapitre 13.

Certains sites permettent au client d'acheter des articles sans s'identifier mais proposent la possibilité d'enregistrer leur profil de façon à extraire rapidement ces informations lors de la prochaine commande.

Notre exemple n'enregistre pas les informations de carte bancaire. Certains utilisateurs font confiance aux marchands connus et leur laissent stocker leurs numéros de carte bancaire. Cela autorise un passage de commande plus rapide. Proposer ce stockage facultatif d'informations de carte bancaire constitue pour vous une difficulté supplémentaire mais plus de convivialité pour le client. Il appréciera que vous lui demandiez son avis avant de stocker son numéro de carte bancaire.

- ✓ **Suivi de commande** : Les clients apprécient de pouvoir suivre leurs commandes. Or il est difficile de fournir cette caractéristique sans identification du client. Si vous autorisez un client à voir les détails d'une commande tout simplement en entrant le numéro de commande, comment empêcher les autres clients de visionner également cette commande ?
- ✓ **Stock** : Dans certaines entreprises, la base de données qui contient les informations sur les articles comporte également les données de stock. Le client a ainsi une information immédiate concernant la disponibilité de tel article. Cela sert aussi à indiquer à l'acheteur que le stock pour un certain article est faible.

Avant d'ajouter de telles fonctions, maîtrisez d'abord l'exemple d'application proposé. Lorsqu'il fonctionnera correctement, commencez à ajouter les fonctions une seule à la fois. Ne modifiez pas trop de choses en même temps. Il est plus facile de régler les problèmes un à la fois.

En général, l'ajout de fonctions à une application orientée objet est plus facile qu'à une application procédurale. L'un des points forts de la programmation orientée objet est de pouvoir ajouter du code sans modifier le code existant. Si vous pensez que votre application est amenée à grossir, il est plus judicieux de la construire de façon orientée objet.

Septième partie

Création d'autres applications utiles



"Les ingénieurs se sont nourris de Coca
et de bâtonnets de fromage pendant tout le
développement du produit, mais si vous voulez
mettre juste 'soda et bâtonnets de fromage'
dans le Manuel Utilisateurs, pas de problème.
On n'est pas trop rigides, ici."

Dans cette partie...

Cette partie propose une application dite Système de gestion de contenu ou CMS (pour Content Management System). Un CMS permet à l'utilisateur de publier facilement ses documents sur un site Web.

Chapitre 16

Création d'un système de gestion de contenu

Dans ce chapitre :

- Vue d'ensemble d'un système de gestion de contenu (CMS).
- Conception d'un CMS élémentaire.
- Autorisations spécifiques à des zones.
- Téléchargement de fichiers vers le serveur et gestion de ces fichiers.
- Conception d'un modèle objet.

Système de gestion de contenu (CMS) est un terme général pour désigner un site Web dont le contenu, le flux et/ou l'aspect est gérable en passant par une interface Web administrative. Un CMS permet à un non-technicien d'actualiser facilement le contenu d'un site Web. Cela réduit le recours systématique à un programmeur pour adapter le code à chaque élément que l'utilisateur souhaite publier.

Conception de l'application CMS

Avant de concevoir l'interface utilisateur, vous devez prendre quelques décisions fondamentales de conception de programmation :

- ✓ **Types de contenu :** L'objectif d'un CMS est de donner le contrôle du contenu du site Web à l'utilisateur. Cependant, le concepteur du CMS doit décider en amont de certaines caractéristiques. Par exemple, certains CMS fournissent un éditeur wysiwyg et une hiérarchie complexe pour organiser les sections du site Web. Ce chapitre vous présentera un exemple de CMS intranet. Il fournit une zone pour chaque

service d'une entreprise. La zone de chaque service possède des sections Actualités, Événements, Réunions, FAQ. La hiérarchie et les types de contenu de cet exemple sont réduits mais la conception du CMS facilite l'ajout d'un nouveau service ou l'ajout d'un nouveau type de contenu (sans code supplémentaire).

- ✓ **Autorisations** : Un CMS doit habituellement savoir qui utilise le site Web, de façon à afficher les menus appropriés. Un *intranet* est habituellement constitué d'un groupe d'applications Web internes permettant le partage d'informations au sein d'une entreprise. Dans l'exemple de CMS de ce chapitre, tout membre d'un service donné peut créer ou modifier le contenu Web de son service. Un utilisateur ne peut modifier le contenu d'un service auquel il n'appartient pas. Par exemple, un employé du service Ventes ne peut toucher au contenu Web du service Informatique. Le CMS doit donc disposer d'un écran d'identification. Les comptes d'utilisateurs doivent préciser le service auquel appartient chaque utilisateur. Le programme d'identification utilisé pour cette application a été étudié au Chapitre 13. L'emploi de ce programme d'identification dans cette application nécessite quelques changements mineurs, détaillés plus loin dans ce chapitre.
- ✓ **Gestion de téléchargement vers le serveur** : Les intranets mettent à disposition des employés un certain nombre de documents. Pour permettre aux utilisateurs de partager des documents, vous devez leur fournir un moyen de télécharger des fichiers vers le serveur. Notre exemple de CMS autorise cette opération. Il garde trace du nom du fichier et du contenu auquel il est associé. Par exemple, si un utilisateur veut publier dans la FAQ la question "Comment s'inscrire au plan d'épargne salariale?", il doit avoir la possibilité de répondre à la question et de poster tous documents apparentés, par exemple des formulaires qui seront utiles aux employés.

Création de la base de données CMS

La base de données stocke des informations sur les utilisateurs, services, types de contenu, éléments de contenu, et téléchargements. La table des utilisateurs sert à déterminer le service auquel appartient un utilisateur. La table des services fournit des informations sur chaque service : nom, description du

service. La table des types de contenu définit les types de contenu que le site Web peut gérer. Les tables des contenus et des téléchargements stockent les détails, les éléments qui serviront aux utilisateurs. Le CMS autorise l'utilisateur à modifier un contenu lorsque ce contenu relève du service auquel appartient l'utilisateur.

Conception de la base de données CMS

L'exemple d'application de ce chapitre exploite une base de données nommée IntranetCMS. Voici les cinq tables composant cette base :

- ✓ **Dept_User** : Stocke les informations sur les utilisateurs, notamment le service où ils travaillent.
- ✓ **Department** : Stocke le nom et la description de chaque service.
- ✓ **Content_Type** : Stocke un ID (identifiant) et une description de chaque type de contenu que l'utilisateur peut publier.
- ✓ **Content** : Stocke les informations sur un contenu : titre, description, date de création, auteur, et autres informations.
- ✓ **Content_Download** : Stocke les noms de fichier des documents téléchargeables. Chaque document est lié à un élément de la table **Content**.

Dans chaque table, une colonne auto-incrémentée permet le lien entre toutes les informations. La colonne `dept_id` est utilisée dans les tables **Department**, **Dept_User** et **Content**. Le champ `content_id` (colonne auto incrément de la table **Content**) lie un contenu à tous les téléchargements associés, qui sont stockés dans la table **Content_Download**.

Conception de la table Dept_User

La table **Dept_User** (voir le Tableau 16.1) contient les informations utilisateurs, notamment le `dept_id` qui lie chaque utilisateur à un service.

Les champs `user_name` et `password` seront utilisés pour accéder au site. La colonne `dept_id` servira lorsque le CMS devra décider si un utilisateur peut ajouter ou modifier le contenu d'une section du site Web.

Tableau 16.1 : Table de base de données : Dept_User.

Nom de variable	Type	Description
user_name	VARCHAR(255)	Identifiant utilisateur (clé primaire)
dept_id	INT	Identifiant service (voir le Tableau 16.2)
first_name	VARCHAR(255)	Prénom de l'utilisateur
last_name	VARCHAR(255)	Nom de l'utilisateur
password	VARCHAR(255)	Mot de passe de l'utilisateur
create_date	TIMESTAMP	Date (et heure) de création de l'enregistrement utilisateur
email	VARCHAR(255)	Adresse électronique

Conception de la table Department

La table `Department` (Tableau 16.2) est une table simple stockant le nom et une courte description pour chaque service. La colonne `dept_id` est un identifiant également exploité dans les autres tables, elle permet l'association de contenu et d'utilisateurs avec un service.

Tableau 16.2 : Table de base de données : Department.

Nom de variable	Type	Description
dept_id	SERIAL	ID unique du service (clé primaire)
description	VARCHAR(255)	Rôle du service dans l'entreprise
name	VARCHAR(255)	Nom du service



SERIAL est un alias de `BIGINT UNSIGNED NOT NULL AUTO_INCREMENT`. Vous vous éviterez de la frappe en cet alias pour créer une colonne auto incrément. Cependant, n'oubliez pas que le stockage requis pour un `BIGINT` est de 8 octets, alors que celui d'un `INT` est de 4 octets. Pour des informations sur l'espace occupé par les types de données MySQL : <http://dev.mysql.com/doc/mysql/en/storage-requirements.html>.

On aurait pu coder en dur la liste des services dans le code PHP mais cela ne facilite pas l'ajout de nouveaux services. En stockant les informations de service dans la base de données, vous améliorez la

gestion du site Web. C'est bien là l'objectif d'un système de *gestion* de contenu. Ce sont les utilisateurs et le système – pas les programmeurs – qui doivent pouvoir gérer les informations. Si vous aviez à réaliser la maintenance de tous les contenus des sites Web que vous construisez, vous ne feriez plus que ça : changer des petits bouts de site Web pour vos clients.

Conception de la table *Content_Type*

Le Tableau 16.3 montre la structure de table permettant au CMS de garder trace des différents types de contenu postables sur le site Web.

Tableau 16.3 : Table de base de données : *Content_Type*.

Nom de variable	Type	Description
type_id	SERIAL	Identifiant du type de contenu (clé primaire)
name	VARCHAR(255)	Description du type de contenu



La liste des types de contenu aurait pu être codée en dur dans le code PHP du site Web mais, ici encore, en gérant ce genre d'informations via une base de données, vous rendez le site Web suffisamment souple pour s'accommoder de futurs types de contenu.

Conception de la table *Content*

La table *Content* est une table générique servant à contenir tout type de contenu. Dans un CMS plus complexe, cette tâche peut ne pas être réalisable avec une seule table. Pour notre exemple, cela suffit. Le Tableau 16.4 présente les colonnes de la table *Content*. Plusieurs attributs sont associés à chaque élément de contenu, comme la date et l'heure auxquelles le contenu a été créé, et la date et l'heure de dernière modification du contenu.

La table *Content* utilise une colonne **TEXT** pour la description détaillée car les types de données **VARCHAR** et **CHAR** ont une longueur limitée à 255 (dans MySQL 4.x et versions précédentes). Une colonne **TEXT** peut contenir $2 + 2^{16}$ octets. La description est le gros du contenu, elle est affichée via un élément **TEXTAREA** dans le formulaire HTML.

Tableau 16.4 : Table de base de données : Content.

Nom de variable	Type	Description
content_id	SERIAL	Identifiant de contenu (clé primaire)
dept_id	INT	Identifie le service auquel ce contenu appartient (voir le Tableau 16.2)
content_type	INT	Identifie le type de contenu stocké dans cet enregistrement (voir le Tableau 16.3)
title	VARCHAR(255)	Description courte du contenu
description	TEXT	Description détaillée du contenu
content_date	DATE	Date à laquelle le contenu est pertinent ou qu'un événement se produit
create_date	TIMESTAMP	Date à laquelle le contenu a été créé
created_by	VARCHAR(255)	Nom de l'utilisateur qui a créé ce contenu
last_upd_date	TIMESTAMP	Date de dernière modification du contenu
last_upd_by	VARCHAR(255)	Nom de l'utilisateur ayant modifié ce contenu

Conception de la table Content_Download

Une liste des documents téléchargeables est associée à chaque contenu de la table Content (Tableau 16.4). Le Tableau 16.5 montre la table qui lie un document à un contenu. C'est le nom du fichier qui sera affiché.

Tableau 16.5 : Table de base de données : Content_Download.

Nom de variable	Type	Description
download_id	SERIAL	Identifiant de document téléchargeable (clé primaire)
content_id	INT	Identifie le contenu auquel appartient ce document téléchargeable (voir le Tableau 16.4)
name	VARCHAR(255)	Le nom du fichier téléchargé vers le serveur

Dans notre exemple de CMS, la table `Content_Download` est très simple. Pour améliorer la caractéristique de téléchargement, il est possible de stocker la taille du fichier, son type, etc. Dans l'exemple, j'ai mis en place une simple capacité de téléchargement montant et descendant.

Création de la base de données CMS

L'instruction SQL suivante crée cette base de données :

```
CREATE DATABASE IntranetCMS;
USE IntranetCMS;
```

Voici les instructions SQL pour créer les tables :

```
DROP TABLE IF EXISTS Department;
```

```
CREATE TABLE Department (
    dept_id      SERIAL,
    name         VARCHAR(255) NOT NULL,
    description  VARCHAR(255) NOT NULL,

    PRIMARY KEY (dept_id)
);
```

```
DROP TABLE IF EXISTS Content_Type;
```

```
CREATE TABLE Content_Type (
    type_id      SERIAL,
    name         VARCHAR(255) NOT NULL,

    PRIMARY KEY (type_id)
);
```

```
DROP TABLE IF EXISTS Content;
```

```
CREATE TABLE Content (
    content_id   SERIAL,
    dept_id     INT          NOT NULL,
    content_type INT          NOT NULL,
    title       VARCHAR(255) NOT NULL,
    description  TEXT         NOT NULL,
    content_date DATE,
    create_date  TIMESTAMP
                DEFAULT now() NOT NULL,
    created_by  VARCHAR(255) NOT NULL,
    last_upd_date TIMESTAMP  NOT NULL,
```

```

last_upd_by  VARCHAR(255) NOT NULL,

PRIMARY KEY(content_id)
);

DROP TABLE IF EXISTS Dept_User;

CREATE TABLE Dept_User (
  user_name    VARCHAR(255)
              UNIQUE NOT NULL,
  dept_id      INT          NOT NULL,
  first_name   VARCHAR(255) NOT NULL,
  last_name    VARCHAR(255) NOT NULL,
  password     VARCHAR(255) NOT NULL,
  create_date  TIMESTAMP
              DEFAULT now() NOT NULL,
  email        VARCHAR(255),

  PRIMARY KEY(user_name)
);

DROP TABLE IF EXISTS Content_Download;

CREATE TABLE Content_Download (
  download_id  SERIAL,
  content_id   INT(6)    NOT NULL,
  file_name    VARCHAR(255) NOT NULL
);

```



Pourquoi détruire les tables avant de les créer ? Lorsque je développe un nouveau site Web, je trouve utile de disposer d'un fichier exploitable pour recréer rapidement la structure de la base de données d'un seul coup. J'utilise ensuite tout ou partie de ce "script" au fur et à mesure pour régler finement la structure de la base de données. L'instruction `drop table if exists` m'épargne du temps lorsque je copie et colle le *langage de définition de données* (pour data definition language, DDL) dans la fenêtre du client MySQL. Les instructions DDL sont un sous-ensemble d'instructions SQL utilisé pour créer ou modifier des objets de base de données et leur structure.

Les instructions SQL suivantes remplissent la table `Content_Type` avec les types de contenu qui seront employés dans le CMS :

```

INSERT Content_Type (name) values ("Actualités");
INSERT Content_Type (name) values ("Événements");
INSERT Content_Type (name) values ("FAQ");
INSERT Content_Type (name) values ("Réunions");

```

Les instructions SQL suivantes remplissent la table Department avec les services de la société fictive de cet exemple :

```
INSERT Department (name, description)
  values ("Ressources humaines",
         "Rassembler les énergies.");

INSERT Department (name, description)
  values ("Ventes", "Vente des produits
         de l'entreprise");

INSERT Department (name, description)
  values ("Distribution", "Notre travail :
         que les produits arrivent chez le
         client");

INSERT Department (name, description)
  values ("Informatique", "Construire les
         applications qui nous rendent
         performants.");
```

Accès à la base de données CMS

Dans notre exemple de CMS, comme dans les précédents chapitres, les références à la base de données sont stockées dans un fichier nommé Vars.inc. Ce fichier contient votre compte et mot de passe MySQL. Par exemple :

```
<?php
    $host = "localhost";
    $user = "admin";
    $passwd = "admin";
    $database = "IntranetCMS";
?>
```

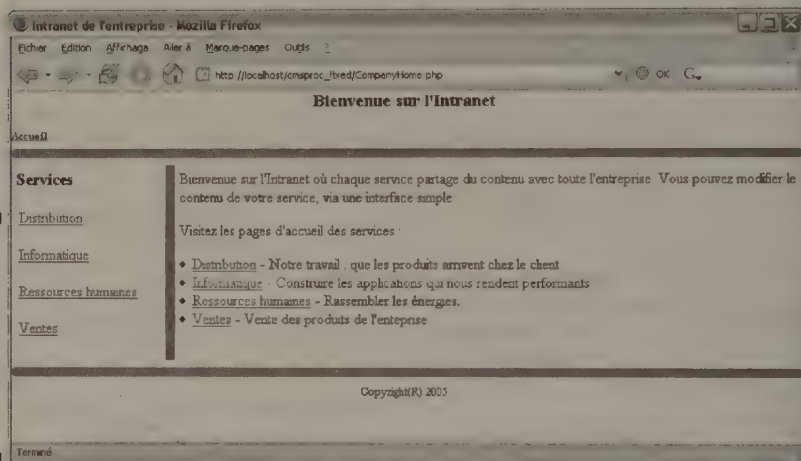
Conception des pages Web du CMS

L'application CMS a une page d'identification et des pages de contenu, avec cinq niveaux de navigation :

- 1. Identification : Avant de pouvoir parcourir le contenu, l'utilisateur doit s'inscrire et s'identifier.** J'ai pris la page d'identification du Chapitre 13 et l'ai légèrement modifiée. Les changements sont notifiés par le Listing 16.1.

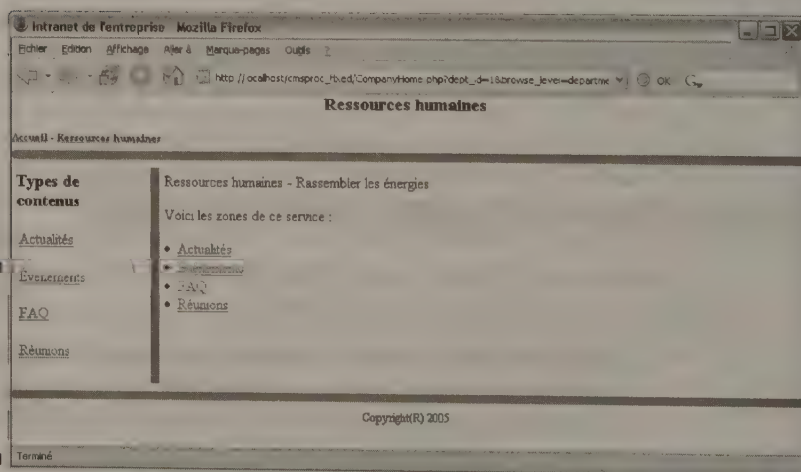
- 2. Page d'accueil :** La page d'accueil affiche la liste des services présents sur l'intranet de l'entreprise. Les descriptions des services sont affichées dans la partie principale de la page ; sur la gauche de la page, on trouve des liens vers les pages intranet des services (Figure 16.1).

Figure 16.1 : La page d'accueil de l'intranet présente la liste des services de l'entreprise.



- 3. Page de service :** A partir de la page d'accueil, l'utilisateur clique sur un service. Dans la page de niveau service, les types de contenu sont énumérés dans la section principale de la page et sur la gauche (Figure 16.2).

Figure 16.2 : Types de contenu pour le service Ressources humaines.



4. **Page de liste de contenus :** dans la page de service, l'utilisateur clique sur une zone de contenu, comme **Actualités**, **Événements**, **FAQ**. La zone de contenu présente la liste des contenus concernant le service, et dont le type correspond à ce que l'utilisateur a choisi à l'étape précédente. Sur la Figure 16.3, c'est les questions de la FAQ du service Ressources humaines qui sont énumérées. Lorsque l'utilisateur affichant cette page n'est pas membre du service Ressources humaines, il ne voit pas les liens pour modifier ou supprimer le contenu. (Les liens Modifier et Effacer se trouvent dans la colonne de droite.)

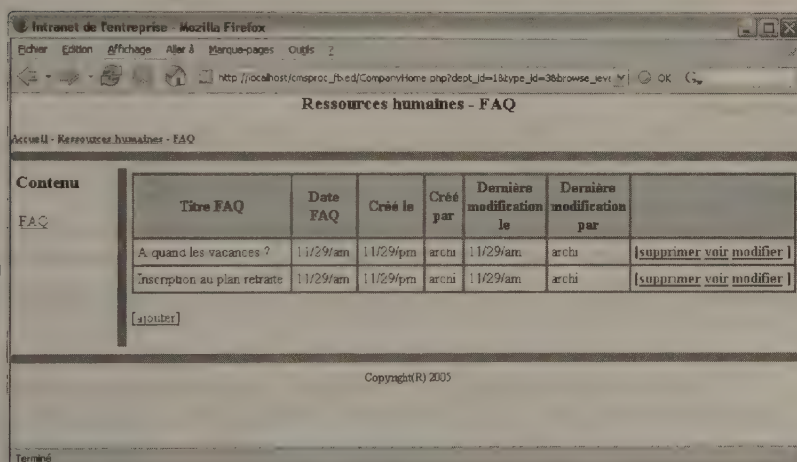


Figure 16.3 : La page de la FAQ du service Ressources humaines.

5. **Page de détail de contenu :** A partir de la page listant les contenus, l'utilisateur peut afficher les détails d'un contenu particulier, assortis des éventuels documents téléchargeables. Le côté gauche de la page énumère les documents téléchargeables associés à ce contenu ; la partie principale de la page présente les détails du contenu, dont la date de création et l'auteur du contenu. La Figure 16.4 présente les détails d'un élément de la FAQ du service Ressources humaines. Cela n'est visible que par les utilisateurs appartenant au service Ressources humaines. Ceux qui ne sont pas des ressources humaines peuvent seulement lire le contenu, présenté comme sur la Figure 16.5.

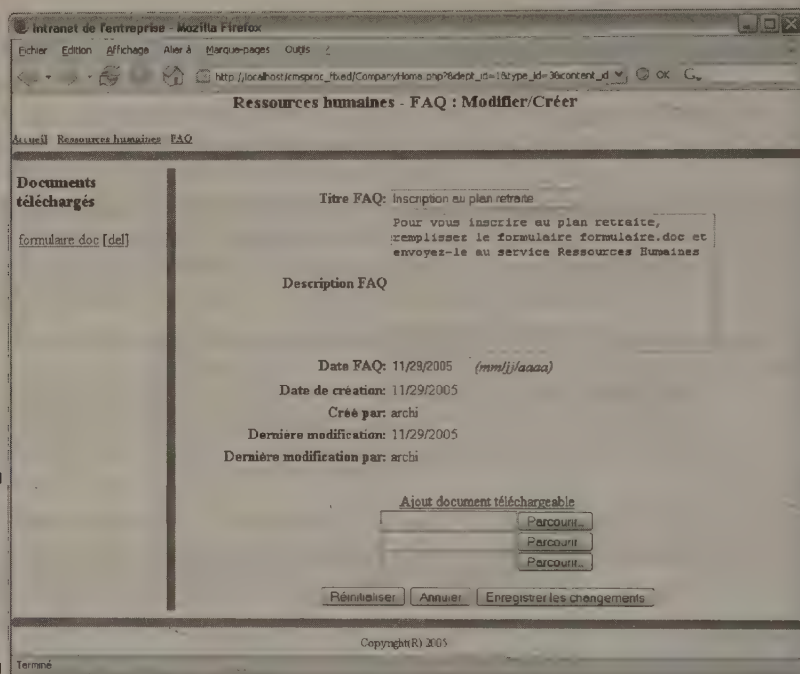


Figure 16.4 :
Page de
détail de la
FAQ des
ressources
humaines.

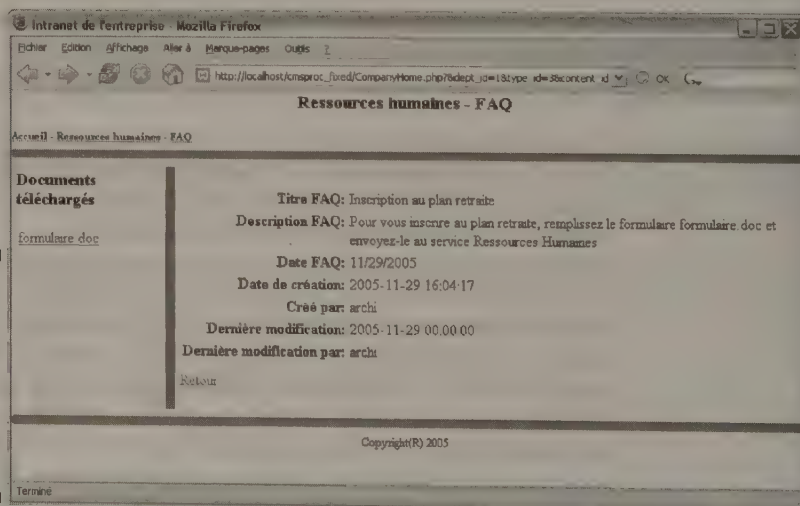


Figure 16.5 :
Page de
détail de la
FAQ des
ressources
humaines, en
mode lecture
seule.

Création de l'application CMS : démarche procédurale

Le fichier `CompanyHome.php` contient la logique pour organiser les données de l'affichage principal du CMS. Il détermine si l'utilisateur visionne la page principale, une page de niveau service, une page de liste de contenus, ou une page de détail. Il remplit les éléments du tableau `$page`. Le tableau `$page` sert dans `company.inc`, pour construire l'affichage en HTML. Ce site intranet CMS impose aux utilisateurs d'être inscrits et identifiés. Le fichier inclus `company.inc` produit l'HTML avec les données définies par `CompanyHome.php`.

Écriture du code d'identification

L'exemple de ce chapitre utilise un système très simple d'authentification. L'application considère que les utilisateurs appartenant à un service ont permission de créer, modifier et supprimer tout contenu de ce service. En outre, l'application permet à l'utilisateur de choisir, au moment de son identification, le service dans lequel il travaille. Dans le monde réel, cela n'est pas aussi souple. Il faut mettre en place un système capable d'identifier correctement les utilisateurs et de leur appliquer les privilèges appropriés. Pour garder les choses simples, nous réutilisons le code `Login.php` du Chapitre 13 (Listing 13.4). Quelques modifications sont cependant nécessaires ; elles sont présentées par le Listing 16.1.

Listing 16.1 : Modifications dans `Login.php`.

```

$stable_name = "Dept_User";           #11
$next_program = "CompanyHome.php";    #12

$sql = "SELECT user_name,dept_id FROM $stable_name" #25

$_SESSION['user_dept'] = $row['dept_id'];      #33
$_SESSION['user_name'] = $row['user_name'];    #34

$_SESSION['user_dept'] = $dept_id;            #192
$_SESSION['user_name'] = $user_name;          #193

```

Voici la description des lignes modifiées dans le fichier `Login.php` du Chapitre 13 :

- #11** Les tables de base de données exploitées dans l'application CMS sont différentes des tables utilisées au Chapitre 13. La ligne 11 est modifiée pour employer la table `Dept_User` conçue plus haut dans ce chapitre.
- #12** La page vers laquelle `Login.php` redirigera l'utilisateur après l'identification ou l'inscription est la page d'accueil de l'application CMS, `CompanyHome.php`.
- #25** L'application stocke le nom d'utilisateur et l'ID de service dans la session, pour usage ultérieur dans l'application. Le champ `dept_id` est ajouté à la requête `SELECT` démarrée sur cette ligne, de façon que l'ID de service puisse être stocké dans la session.
- #33** Stocke l'ID de service dans la session.
- #34** Stocke le nom d'utilisateur dans la session. Les lignes 33 et 34 stockaient différentes données dans le `Login.php` du Chapitre 13.
- #192** [et **#193**] Le Chapitre 13 stockait un indicateur dans la session pour indiquer que l'utilisateur avait été authentifié (`auth`), et il stockait l'ID utilisateur (`logname`). Ici, l'application CMS stocke le nom d'utilisateur (`user_name`) et l'identifiant de service (`dept_id`) dans la variable de session. Le `user_name` est pratique lorsque le CMS a besoin de stocker des données sur la personne qui ajoute ou modifie le contenu. L'ID de service (`dept_id`) permet au CMS de garder trace du service pour lequel l'utilisateur a permission d'ajouter ou modifier le contenu.

Le Listing 16.2 est une copie du Listing 13.1 du Chapitre 13, à laquelle quelques modifications ont été apportées. La différence la plus importante est que la liste de services est construite par connexion à la base de données et interrogation de la table `Department`. La liste de services est nécessaire pour la liste déroulante `Service` de la page d'identification, qui autorise l'utilisateur à choisir le service dont il est membre.

Listing 16.2 : Fichier contenant les tableaux nécessaires à la page d'identification.

```
<?php
/* Fichier : fields_login.inc
 * Description : Tableaux avec noms des champs et éléments de
 *              de formulaire pour la page Web d'identification.
 */
include_once("functions_main.inc");

$page = array( "title" => "Page d'identification",
               "top"    => "",
               "bottom" => "Envoyez vos questions et commentaires
                           à admin@mondomaine.com",
               );

$elements_1 = array( "top" => "Déjà inscrit :
                           <span style=\"font-size: 80%;
                           font-weight: 100%\">
                           <i>Identification</i></span>",
                     "bottom" => "",
                     "submit" => "Identification"
                     );

$elements_2 = array( "top" => "Nouveaux-venus:
                           <span style=\"font-size: 80%;
                           font-weight: 100%\">
                           <i>Inscrivez-vous</i></span>",
                     "bottom" => "",
                     "submit" => "Inscription"
                     );

$fields_1 = array( "fusername" => "Nom utilisateur",
                  "fpassword" => "Mot de passe"
                  );

$length_1 = array( "fusername" => "10",
                  "fpassword" => "10"
                  );

$types_1 = array( "fusername"    => "text",
                  "fpassword"    => "password"
                  );

$fields_2 = array( "user_name"    => "Nom utilisateur",
                  "password"      => "Mot de passe",
                  "email"        => "Adresse électronique",
                  "first_name"    => "Prénom",
                  "last_name"     => "Nom",
                  "dept_id"       => "Service"
                  );
```



```

$types_2 =    array("user_name"    => "text",
                    "password"     => "password",
                    "email"        => "text",
                    "first_name"   => "text",
                    "last_name"    => "text",
                    "dept_id"      => "select"
                );
$length_2 =   array("user_name"    => "20",
                    "password"     => "8",
                    "email"        => "55",
                    "first_name"   => "40",
                    "last_name"    => "40",
                );

$options =     array();

$conconnection = Connect_to_db("Vars.inc");
$results = mysqli_query($conconnection, "SELECT dept_id, name
                                         FROM Department
                                         ORDER BY name");
while($row = mysqli_fetch_assoc($results)) {
    $options['dept_id'][$row['dept_id']] = $row['name'];
}

?>

```

Le fichier `double_form.inc` (Listing 16.3) est une copie légèrement modifiée du Listing 13.2 du Chapitre 13. La différence la plus importante est qu'il existe maintenant du code construisant la liste déroulante des services, à partir de la ligne 85.

Listing 16.3 : Définition de deux formulaires HTML côte à côte.

```

<?php
/* Fichier : double_form.inc
 * Description : Code d'une page Web affichant
 *              deux formulaires HTML, côte à côte dans un tableau.
 */
?>
<head><title><?php echo $page['title']?></title></head>
<body style="margin: 0">
<h1 align="center"><?php echo $page['top'] ?></h1>
<hr size="10" noshade>

```

```

<table border="0" cellpadding="5" cellspacing="0">
<?php
#####
# Formulaire 1 #
#####
?>

<tr>
<td width="33%" valign="top">
    <p style="font-size: 110%; font-weight: bold">
        <?php echo $elements_1['top']?></p>
    <!-- Début du formulaire 1 (gauche) -->
    <form action=<?php echo $_SERVER['PHP_SELF']?>
        method="POST">
        <table border="0">
<?php
if (isset($GLOBALS['message_1'])) : #27
{
    echo "<tr>
        <td colspan='2'
            style=\"font-weight: bold;
                font-style: italic;
                font-size: 90%; color: red\">
                {$GLOBALS['message_1']}<p></td></tr>\n";
}
foreach($fields_1 as $field => $value) #36
{
    $type = $types_1[$field];
    echo "<tr><td style=\"text-align: right;
        font-weight: bold\">$value</td>
        <td><input type='$type' name='$field'
            value='\".@$$field.\"'
            size='{ $length_1[$field]}'
            maxsize='{ $length_1[$field]}'>
        </td></tr>\n";
}
?>

<tr> #48 -->
    <td colspan="2" style="text-align: center" >
        <br />
        <input type="submit" name="Button"
            value="<?php echo $elements_1['submit']?>">
        </td></tr>
</table>
</form>
</td>

<!-- Colonne séparant les deux formulaires -->
<td style="background-color: gray"></td>
    
```

```

<?php
#####
# Formulaire 2 #
##### #63
?>

<td width="67%">
  <p style="font-size: 110%; font-weight: bold">
    <?php echo $elements_2['top']?>
    <!-- Début du formulaire 2 (côté droit) -->
    <form action=<?php echo $_SERVER['PHP_SELF']?>
      method="POST">

    <p>
    <table border="0" width="100%">

<?php
if (isset($GLOBALS['message_2'])) #74
{
  echo "<tr>
    <td colspan='2'
      style=\"font-weight: bold; font-style: italic;
      font-size: 90%; color: red\">
      {$GLOBALS['message_2']}<p></td></tr>";
}
foreach($fields_2 as $field => $value) #82
{
  $type = $types_2[$field];
  if($type == "select") #85
  {
    echo "<tr><td style=\"text-align: right;
      font-weight: bold\">{$fields_2[$field]</td>
      <td><select name='{$field}'>";
    foreach ($options[$field] as $opt_id => $opt_name)
    {
      echo "<option value='{$opt_id}'";
      if (@$_GET[$field] == $opt_id)
        echo " selected";
      echo ">{$opt_name}\n";
    }
    echo "</select>";
  }
  else
  {
    echo "<tr><td style=\"text-align: right; #101
      font-weight: bold\">{$value}</td>
      <td><input type='{$type}' name='{$field}'
        value='\". @${$field}.\"'
        size='{${length_2[$field]}}'
        maxsize='{${length_2[$field]}}'>
      </td></tr>";
  }
}

```

```

    }
  }
  ?>

  <tr>
    <td colspan="2" style="text-align: center">
      <p style="margin-top: .05in">
        <input type="submit" name="Button"
          value="<?php echo $elements_2['submit']?>">
      </td></tr>
    </table>
  </form>
</td>
</tr>
</table>
<hr size="10" noshade>
<div style="text-align: center; font-size: 75%">
  <?php echo $page['bottom']?>
</body></html>

```

Dans les explications suivantes, les numéros renvoient aux numéros de lignes du Listing 16.3 (double_form.inc) :

- #27** Vérifie l'existence d'un message d'erreur dans le tableau `$_GLOBALS`. Si c'est le cas, le message d'erreur est affiché.
- #36** Pour chaque élément du tableau `$fields_1` qui est utilisé dans le formulaire d'identification, un élément de saisie de formulaire est construit.
- #48** À la ligne 48, le bouton d'envoi est affiché. Ce bouton, lorsqu'il est cliqué, soumet le formulaire à `Login.php` : le nom et le mot de passe de l'utilisateur seront évalués.
- #63** Le formulaire créé après la ligne 63 est le formulaire d'inscription.
- #74** L'appel de fonction `isset` vérifie l'existence d'un message d'erreur dans le tableau `$_GLOBALS`. Si c'est le cas, le message d'erreur est affiché.
- #82** L'instruction `foreach` débute une boucle traversant les éléments devant être affichés dans le formulaire d'inscription, comme défini par le tableau `$fields_2`. La ligne 84 regarde le type de l'élément HTML du champ dans le tableau `$types_2` (défini dans `fields_login.inc`).

- #85** Ce bloc de code crée la liste déroulante des services. Dans un CMS véritable, il faut une sécurité renforcée. Dans l'exemple de ce chapitre, on fait confiance à l'utilisateur sur le choix de son service. Rappelons en effet que tout utilisateur associé à un service a les droits d'administration du contenu de ce service. Un CMS grandeur nature devra insérer une autre couche d'administration, avec un "super utilisateur" à même d'accorder ou révoquer les privilèges administratifs.
- #101** Dans l'HTML aux environs de la ligne 101, un élément de saisie de formulaire est construit. La longueur de l'élément est définie dans le tableau `$length_2` (dans `fields_login.inc`).
- #111** À la ligne 111, le bouton d'envoi est affiché. Ce bouton, s'il est cliqué, soumet le formulaire à `Login.php` qui traitera les informations d'inscription. Si la validation réussit, l'utilisateur est redirigé vers la page d'accueil de l'intranet. Si une erreur est trouvée pendant la validation des informations d'inscription, la page d'identification est réaffichée et les erreurs sont indiquées à l'écran, en rouge.

Écriture de `CompanyHome.php`, fichier d'accès aux données

`CompanyHome.php` a pour rôle de définir les données utilisées par `company.inc`, fichier qui affiche l'interface HTML. `CompanyHome.php` est structuré sous forme d'une instruction `switch`, avec des blocs `case` pour chaque niveau de navigation. Le niveau de navigation reflète le niveau auquel navigue l'utilisateur dans la hiérarchie du site, en commençant par la page d'accueil et en descendant vers le niveau de détail de contenu. Le niveau de navigation est passé dans l'URL. L'instruction `switch` teste le niveau de navigation et exécute le bloc `case` approprié. Voici une vue d'ensemble de la structure du script :

selon niveau de navigation (`browse_level`)

```
cas "home":
1 Obtenir la liste des services dans la
  table Department.
2 Utiliser la liste des services pour construire
  les liens vers les services (à gauche).
3 Utiliser la liste des services pour construire
  le corps principal du texte de la page Web, qui
```


comprendra la description du service.

cas "department":

- 1 Obtenir la liste des types de contenu reconnus par le CMS, à partir de la table Content_Type.
- 2 Utiliser la liste des types de contenu pour construire les liens de gauche vers les pages de types de contenu du service sélectionné.
- 3 Utiliser la liste des types de contenu pour construire les liens vers les pages de types de contenu du service sélectionné.

cas "content":

- 1 Obtenir la liste des contenus en se basant sur le service et le type de contenu choisis par l'utilisateur.
- 2 Si aucun contenu n'existe, afficher un message le signalant.
- 3 Si du contenu existe, lister les articles dans un tableau.
- 4 Si l'utilisateur a des autorisations administratives pour ce service, afficher les liens permettant d'ajouter ou modifier le contenu.

cas "details":

- 1 Obtenir la liste des détails de contenus, en se basant sur le service, le type de contenu et le contenu sélectionnés par l'utilisateur.
- 2 Si l'utilisateur est administrateur, afficher un formulaire avec les éléments permettant à l'utilisateur de télécharger des fichiers vers le serveur.
- 3 Afficher les fichiers téléchargeables dans la section gauche de la page Web.

Le Listing 16.4 présente le code PHP configurant les données utilisées pour afficher les pages Web.

Listing 16.4 : Lecture des données de service et de contenu dans la base MySQL

```
<?php
/* Programme : CompanyHome.php
 * Description : Affiche une page Web à quatre niveaux :
 *      1) page d'accueil, 2) page de service, 3) page
 *      des contenus 4) page des détails de contenu.
 */
if (!isset($_SESSION))
```

#7

```

    session_start();

    include_once("functions_main.inc");

    $page = array(                                     #12
        "title"      => "Intranet de l'entreprise",
        "header"     => "Intranet de l'entreprise",
        "bottom"     => "Copyright(R) 2005",
        "left_nav_links" => array(),
        "body_links"  => array(),
        "col_headers" => array(),
        "data_rows"   => array(),
    );

    $admin = FALSE;
    $base_url = "CompanyHome.php";
    $trail = "<a href='$base_url'>Accueil</a>";          #24

    if (!isset($_SESSION['user_name']))
        header("Location: Login.php");                #27
    else
    {
        if (isset($_SESSION['user_dept'])
            && isset($_GET['dept_id']))
        {                                              #32
            $admin = $_SESSION['user_dept'] == $_GET['dept_id'];
        }

        $cxn = Connect_to_db("Vars.inc");
        $left_nav_links = array();
        $page["browse_level"] =                        #38
            isset($_GET['browse_level']) ?
                $_GET['browse_level'] : "home";

        switch ($page["browse_level"])                #42
        {
            case "home":
                $sql = "SELECT name, dept_id, description
                        FROM Department
                        ORDER BY name";
                $results = mysqli_query($cxn, $sql);
                $body_links = "";
                while($row = mysqli_fetch_assoc($results)) #50
                {
                    $link = "$base_url?dept_id=" . $row['dept_id']
                        . "&browse_level=department";
                    $page["left_nav_links"][$link] = $row['name'];
                    $body_links .= "<li><a href=\"$link\">";
                }
            }
        }
    }

```

```

        . "\">" . $row['name'] . "</a> - "
        . $row['description'];
    }
    $page["left_nav_header"] = "Services"; #59
    $page["top"] = "Bienvenue sur l'Intranet";
    $page["body_text"] = "Bienvenue sur l'Intranet "
        . "où chaque service partage du contenu avec "
        . "toute l'entreprise. Vous pouvez modifier le "
        . "contenu de votre service, via une interface "
        . "simple.<p>Visitez les pages d'accueil des "
        . "services : $body_links";

    break;

    case "department": #70
        $dept_id = $_GET['dept_id'];
        $sql = "SELECT name, dept_id, description
            FROM Department
            WHERE dept_id = $dept_id ORDER BY name";
        $results = mysqli_query($cxn, $sql);
        $row = mysqli_fetch_assoc($results);
        $dept_name = $row['name'];
        $dept_desc = $row['description'];

        $page["left_nav"] = "$dept_name Content";
        $page["body_text"] = "$dept_name - $dept_desc";
        $sql = "SELECT a.name, a.type_id,
            count(b.content_id)
            FROM Content_Type a
            LEFT OUTER JOIN Content b on
                a.type_id = b.content_type
                and b.dept_id = $dept_id
            GROUP BY a.name, a.type_id ORDER BY name";
        $results = mysqli_query($cxn, $sql);

        $body_links = "";
        while($row = mysqli_fetch_assoc($results)) #92
        {
            $link = "$base_url?dept_id=$dept_id"
                . "&type_id=" . $row['type_id']
                . "&browse_level=content";
            $page["left_nav_links"][$link] = $row['name'];
            $body_links .= "<li><a href=\"\" . $link
                . "\">" . $row['name'] . "</a>";
        }
        $page["left_nav_header"] = "Types de contenus";
        $page["top"] = $dept_name;
        $page["body_text"] = "$dept_name - $dept_desc "

```

```

        . "<p>Voici les zones de ce "
        . "service : $body_links";
$trail .= " - <a href='$base_url?dept_id=$dept_id"
        . "&browse_level=department'$dept_name</a>";
break;

case "content":
#110
    $dept_id = $_GET['dept_id'];
    $type_id = $_GET['type_id'];
    $sql = "SELECT a.name, a.type_id, b.title,
                b.description, b.content_date,
                b.create_date, b.created_by,
                b.last_upd_date, b.last_upd_by,
                c.name as dept_name, content_id
            FROM Content_Type a, Department c
            LEFT OUTER JOIN Content b on
                a.type_id = b.content_type
                and a.type_id = b.content_type
                and b.dept_id = $dept_id
                and b.content_type = $type_id
            WHERE c.dept_id = $dept_id
            ORDER BY content_date DESC";
    $results = mysqli_query($cxn, $sql);

    $body_links = "";
    $content_count = 0;
    $page["body_text"] = "";

    while($row = mysqli_fetch_assoc($results))
#132
    {
        if (!isset($area_name) && $type_id == $row["type_id"])
        {
            $area_name = $row["name"];
            $dept_name = $row["dept_name"];
        }
        $link = "$base_url?dept_id=$dept_id"
            . "&type_id=" . $row['type_id']
            . "&browse_level=content";
        $page["left_nav_links"][$link] = $row['name'];

        if (!isset($row["content_id"]))
#144
            continue;

        $content_id = $row["content_id"];

        $content_count++;
        $link = "$base_url?dept_id=$dept_id"
            . "&type_id=$type_id&browse_level=content";

```

```

$page["left_nav_links"][$link] = $row['name'];
$page["data_rows"][] = $row;

}
if ($content_count == 0)                                     #156
{
    $page["body_text"] = "Pas de contenu $area_name
    dans $dept_name";
}
if ($admin)                                                  #161
{
    $page["body_text"] .= "<p>[<a href='$base_url?dept_id=$dept_id"
    . "&browse_level=details&type_id=$type_id"
    . "&content_id='>ajouter</a>]";
}
$page["col_headers"]["title"] = "Titre $area_name";
$page["col_headers"]["content_date"] = "Date $area_name";
$page["col_headers"]["create_date"] = "Créé le";
$page["col_headers"]["created_by"] = "Créé par";
$page["col_headers"]["last_upd_date"] =
    "Dernière modification le";
$page["col_headers"]["last_upd_by"] =
    "Dernière modification par";

$page["left_nav_header"] = "Contenu";                       #176
$page["top"] = "$dept_name - $area_name";
$trail .= " - <a href='$base_url?dept_id=$dept_id"
    . "&browse_level=department'>$dept_name</a>";
$trail .= " - <a href='$base_url?dept_id=$dept_id"
    . "&browse_level=content"
    . "&type_id=$type_id'>$area_name</a>";
break;

case "details":                                             #185
    $dept_id = $_GET['dept_id'];
    $type_id = $_GET['type_id'];

    $sql = "SELECT a.name as dept_name, b.name
    FROM Department a, Content_Type b
    WHERE b.type_id = $type_id
    and a.dept_id = $dept_id
    ORDER BY name";
    $results = mysqli_query($cxn, $sql);
    $body_links = "";
    $content_count = 0;

while($row = mysqli_fetch_assoc($results))                 #198
{

```



```

$area_name = $row["name"];
$dept_name = $row["dept_name"];

if (!isset($row["content_id"]))                                #203
    continue;

$content_count++;
$link = "$base_url?dept_id=$dept_id"
    . "&type_id=$row['type_id']"
    . "&browse_level=content";
$page["left_nav_links"][$link] = $row['name'];
$body_links .= "<li><a href=\"\" . $link
    . "\">\" . $row['name'] . \"</a>\"";
}
$create_date = date("m/d/y", time());
$created_by = $_SESSION["user_name"];
$last_upd_by = $_SESSION["user_name"];

$content_id = $_GET["content_id"];
$edit = $admin && (@$_GET["edit"] == "true"
    || $content_id == "");

if ($content_id != "")                                         #222
{
    Connect_to_db("Vars.inc");
    $sql = "SELECT content_id, dept_id, content_date,
        content_type as type_id, title,
        description, create_date,
        created_by, last_upd_date, last_upd_by
    FROM Content
    WHERE content_id = $content_id";
    $results = mysqli_query($cxn, $sql);
    if ($row = mysqli_fetch_assoc($results))
    {
        foreach ($row as $key => $value)
            $$key = $value;
    }
    $sql = "SELECT download_id, file_name
    FROM Content_Download
    WHERE content_id = $content_id";

    $results = mysqli_query($cxn, $sql);
    while($row = mysqli_fetch_assoc($results))                 #242
    {
        $download_id = $row["download_id"];
        $file_name = $row["file_name"];
        $link = "files/$download_id/$file_name";
        $page["left_nav_links"][$link] = $file_name;
    }
}

```

```

        if ($edit)
            #249
            $page["left_nav_links"][$link] .= "</a>
            [<a href=\"Admin.php\"
            . \"?action=DeleteDownload&download_id=$download_id\"
            >del</a>]";
        }
    }

    foreach ($_GET as $name => $value)
        #257
        $$name = $value;

    $edit = $admin && (@$_GET["edit"] == "true" || $_content_id == "");

    $page["top"] = "$dept_name - $area_name";

    if ($edit)
        #264
        {
            $page["body_text"] = "<center><u>Ajout document téléchargeable</u>";
            for ($i = 0; $i < 3; $i++)
            {
                $page["body_text"] .=
                "<br><input type='file' name='upload_file$i'>";
            }

            $page["body_text"] .= "
            </center> <p />
            <center>
            <input type='reset' name='action'
            value ='Réinitialiser'>
            <input type='submit' name='action'
            value ='Annuler'>
            <input type='submit' name='action'
            value ='Enregistrer les changements'>
            </center>";

            $page["top"] .= " : Modifier/Créer";
        }
    else
    {
        $page["body_text"] =
        "<a href='javascript:history.go(-1)''>Retour</a>";
    }

    $page["left_nav_header"] = "Documents téléchargés";
    $trail .= " - <a href='$base_url?dept_id=$dept_id'
    . "&browse_level=department'$dept_name</a>";
    $trail .= " - <a href='$base_url?dept_id=$dept_id"

```

```

        . "&browse_level=content"
        . "&type_id=$type_id'>$area_name</a>";

    break;
}

include("company.inc");
}
?>

```

Dans les explications suivantes, les numéros renvoient aux numéros de lignes de `CompanyHome.php` (Listing 16.4) :

- #7 Les lignes 7 et 8 s'assurent qu'une session a été démarrée. L'appel `isset` à la ligne 7 est utilisé car `Admin.php`, qui comporte également un appel `session_start`, utilise ce fichier. Sans la vérification `isset` de la variable `$_SESSION`, une notification s'affichera, indiquant qu'une session a déjà été démarrée et que l'appel à `session_start()` sera donc ignoré. La notification s'affiche lorsque le niveau de rapport d'erreur `error_reporting` (défini dans le fichier `php.ini`) inclut le niveau `E_NOTICE`.
- #12 Les lignes 12 à 19 définissent quelques chaînes et tableaux qui seront exploités dans `company.inc` pour afficher la page Web. Modifiez éventuellement les variables `title`, `header` et `bottom` pour afficher le nom de votre société. Les éléments `left_nav`, `body_links`, `col_headers` et `data_rows` sont des listes de données.
- #24 Définit une variable nommée `$trail` (piste). Cette chaîne sera utilisée pour construire une suite de liens représentant l'arborescence du site traversée par l'utilisateur. Sur la Figure 16.3, plus haut dans ce chapitre, la piste affiche Accueil, puis un nom de service (Ressources humaines), puis la zone de contenu en cours (FAQ).
- #27 Les lignes 27 et 28 vérifient que l'utilisateur est inscrit et identifié. Vous pouvez supprimer ces lignes si vous désirez ouvrir le site aux utilisateurs non inscrits. Certains intranet ne réclament pas de login sauf lorsque l'utilisateur tente d'entrer dans une partie administrative du site.
- #32 Les lignes 30 à 34 définissent la variable `$admin` sur `TRUE` ou `FALSE`. La variable `$admin`, définie ligne 22, est utilisée pour

déterminer si un utilisateur a des privilèges administratifs dans la zone du site Web actuellement visitée.

#38 Les lignes 38 à 40 définissent la variable `browse_level` (un élément du tableau `$page`). Le niveau de navigation détermine si l'utilisateur visionne :

- La page d'accueil de l'entreprise (`browse_level` vaut "home").
- La page d'accueil d'un service (`browse_level` vaut "department").
- Une liste de contenus (`browse_level` vaut "content").
- La vue détaillée d'un contenu (`browse_level` vaut "details").

#42 La ligne 42 contient une instruction `switch` qui exécute un bloc dépendant du niveau d'arborescence auquel l'utilisateur navigue.

#50 La ligne 50 recueille les noms de services composant l'entreprise. Le design de la table `Department` (Tableau 16.2) autorise l'ajout en souplesse de nouveaux services (ou leur éventuelle suppression).

#59 Les lignes 59 à 66 remplissent quelques variables de niveau `page`. Le niveau (`browse_level`) "home" représente la page d'accueil principale (voir plus haut la Figure 16.1). À ce niveau, les services sont énumérés. Les éléments `left_nav_header`, `top` et `body_text` du tableau `$page` sont définis ici. Pour améliorer la gestion du CMS (qu'il soit moins dépendant du programmeur), il faut que le CMS obtienne ces chaînes de la base de données. Bien sûr, pour que le CMS soit véritablement gérable pour l'utilisateur, il faut construire une interface pour modifier ces chaînes.

#70 Les lignes 70 à 88 servent à construire l'affichage de niveau service (voir plus haut la Figure 16.2). Le SQL s'achevant ligne 88 utilise une clause `OUTER JOIN` pour s'assurer que tous les types de contenu seront lus dans la table `Content_Type`. Si une jointure ordinaire (`INNER JOIN`) était utilisée ici, aucune ligne ne serait retournée dans le cas où il n'existerait pas de contenu dans la table `Content` pour ce service. La clause `OUTER JOIN` lit les types de contenu de la table `Content_Type`, que des contenus existent ou non dans la table `Content`.

- #92 Débute une boucle traversant la liste des types de contenu. Les lignes 94 à 96 construisent le lien vers la liste des contenus.
- #110 Débute un bloc qui construit l'affichage de niveau contenu (voir plus haut la Figure 16.3).
- #132 Débute une boucle traversant la liste des contenus. Lorsque l'exécution est à ce stade du code, le programme connaît le service et le type de contenu où navigue actuellement l'utilisateur.
- #144 Vérifie `content_id`. Si `content_id` n'est pas définie, la boucle poursuit avec l'itération suivante. La variable `content_id` peut valoir null, du fait de la clause `LEFT OUTER JOIN`. Le code construit les liens de gauche vers les types de contenu, qu'il existe ou non des contenus dans la table `Content`. La ligne 153 affecte l'élément `data_rows` du tableau `$page` aux résultats de la requête.
- #156 La ligne 156 vérifie le nombre de contenus traités dans la boucle précédente. S'il n'existe pas de contenu, un message est affiché de façon que l'utilisateur ne se trouve pas en face d'une page vierge.
- #161 Débute un bloc de code affichant un lien administratif "Ajouter" si l'utilisateur est administrateur du service en cours de navigation.
- #176 Les lignes 176 et 177 définissent des éléments d'affichage indiquant à l'utilisateur que le niveau de données en cours de navigation est le niveau contenu. Les lignes 178 à 181 définissent la piste de liens indiquant à l'utilisateur où il se trouve dans l'arborescence du site Web.
- #185 Débute un bloc de code construisant l'affichage de niveau détails de contenu (Figures 16.4 et 16.5).
- #198 Débute une boucle traversant la liste des contenus. Lorsque l'exécution est à ce stade du code, le programme connaît le service, le type de contenu et le contenu exact jusqu'où l'utilisateur a navigué.
- #203 À la ligne 203 (comme à la ligne 144), un contrôle de `content_id` est effectué. Si la variable `content_id` n'est pas définie, la boucle poursuit avec son itération suivante.

- #222 Si `$content_id` contient une valeur, le code sait que l'utilisateur modifie un contenu existant. Lorsque cette variable `$content_id` est vide, l'utilisateur est en train de créer un nouveau contenu.
- #242 Débute un bloc de code qui construit la liste des téléchargements associés à un contenu.
- #249 Débute un bloc de code qui ajoute un lien administratif "effacer" lorsque l'utilisateur est administrateur du service en cours. Le lien "effacer" permet à l'utilisateur de supprimer le document téléchargeable.
- #257 Débute une boucle traversant le tableau `$_GET` et définit des variables basées sur les éléments du formulaire HTML soumis.
- #264 Débute un bloc construisant les liens pour les actions administratives. Les utilisateurs non-administrateurs du service ne verront pas ces liens administratifs, ils seront limités à la lecture des contenus.
- #297 Marque la fin du script `CompanyHome.php`. À ce stade de l'exécution du programme, les données nécessaires pour construire l'HTML ont été définies. Le fichier `company.inc` est alors inclus pour élaborer réellement l'affichage HTML.

Écriture de *company.inc*, fichier d'affichage HTML

Le fichier précédent, `CompanyHome.php` (Listing 16.4), opère la plupart du travail pour déterminer à quel endroit se trouve l'utilisateur dans la hiérarchie du site Web, s'il s'agit d'un administrateur, et le titre de la page Web. Le fichier suivant, `company.inc` (Listing 16.5), opère le travail d'affichage. Il analyse les listes de données définies dans `CompanyHome.php` et construit l'HTML.

Listing 16.5 : Elaboration de l'affichage de l'accueil et du service

```
<?php
/* Fichier : company.inc
 * Description : Contient le code d'une page Web
 * affichant les informations d'entreprise et de services.
 */
```

```

include_once("functions_main.inc"); #6
?>
<html>
<head><title><?php echo $page['title']?></title></head>
<body style="margin: 0">
<h3 align="center"><?php echo $page['top'] ?></h3>
<div style="font-size: 70%; font-weight: bold">
    <?php echo $trail ?></div>
<hr size="10" noshade>
<table border="0" cellpadding="5" cellspacing="0">
<?php
    #####
    ## Navigation gauche ##
    ##### #16
?>
    <tr>
        <td width="20%" valign="top" >
            <p style="font-size: 110%; font-weight: bold">
                <?php echo $page['left_nav_header']?></p>
            <table border="0">
<?php
    foreach($page["left_nav_links"] as $link => $label) #27
    {
        echo "<tr><td >"
            . "<a href=\"\$link\">\$label<p><p></td></tr>\n";
    }
    if (sizeof($page["left_nav_links"]) == 0)
        echo "<i>aucun élément</i>";
?>
        </table>
    </td>

    <!-- Colonne entre les deux formulaires -->
    <td style="background-color: gray"></td>
<?php
    #####
    ## Contenu principal ##
    #####
?>
        <td width="80%" valign="top">
            <form method="POST" action="Admin.php"
                enctype="multipart/form-data">
<?php

if ($page["browse_level"] == "details") #50
{
    include("fields_content.inc");
    include("content_form.inc");

```

```

}
else if (@$content_count > 0)                                     #55
{
    echo "<table cellpadding='3' cellspacing='3'
        width='100%' bgcolor='gray'>
        <tr bgcolor='lightgray'>\n";
    foreach ($page["col_headers"] as $key => $display)
    {
        echo "<th >$display</th>\n";
    }
    echo "<th nowrap>&nbsp;&nbsp;&nbsp;</th>\n";
    echo "</tr>\n";
    foreach ($page["data_rows"] as $row)                         #66
    {
        echo "<tr bgcolor=white>\n";
        foreach ($page["col_headers"] as $key => $display)
        {
            if (ereg("date", $key))
                $row[$key] = date("m/j/a", strtotime($row[$key]));
            echo "<td nowrap>".$row[$key]."</td>\n";
        }
        echo "<th nowrap>[";
        if ($admin)                                              #76
        {
            echo "<a href=\"Admin.php?action=delete\"
                . "&dept_id=$dept_id&type_id=$type_id&content_id=\"
                . $row["content_id"] . "\">supprimer</a>\n";
        }
        echo "<a href=\"CompanyHome.php?\"
            . "&dept_id=$dept_id&type_id=$type_id&content_id=\"
            . $row["content_id"] . "&browse_level=details&edit=false\">\"
            . "voir</a>\n";
        if ($admin)                                             #86
        {
            echo "<a href=\"CompanyHome.php?\"
                . "&dept_id=$dept_id&type_id=$type_id&content_id=\"
                . $row["content_id"] . "&browse_level=details&edit=true\">\"
                . "modifier</a>\n";
        }
        echo "]"</th></tr>\n";
    }
    echo "</table>\n";
}

echo $page["body_text"];
?>
</form>
</td>

```

```

    </tr>
</table>
<hr size="10" noshade>
<div style="text-align: center; font-size: 75%">
<?php echo $page['bottom']?>
</body>
</html>

```

Dans les explications suivantes, les numéros renvoient aux numéros de lignes de `company.inc` (Listing 16.5) :

- #6** La ligne 6 garantit que le fichier contenant le code de connexion à la base de données ne sera inclus qu'une fois.
- #16** Débute une ligne HTML et la colonne de gauche qui contient soit les services (lorsque l'utilisateur est à la page d'accueil), soit les types de contenu (lorsque l'utilisateur est dans une page de service), soit les téléchargements disponibles (lorsque l'utilisateur est sur une page de détails de contenu).
- #27** Débute la boucle des liens de gauche. Lorsqu'aucun lien n'existe, le message `aucun contenu s'affiche`.
- #50** Lorsque l'utilisateur visionne un contenu, l'HTML est construit par les fichiers inclus.
- #55** Si plusieurs contenus sont énumérés, un tableau HTML listant les contenus disponibles est construit. Les lignes 57 à 65 définissent le début du tableau HTML.
- #66** Débute la boucle qui construit dans le tableau HTML une ligne pour chaque contenu.
- #76** Lorsque l'utilisateur est administrateur, un lien pour supprimer le contenu est ajouté.
- #86** Si l'utilisateur est administrateur, un lien pour modifier le contenu est ajouté.

Écriture du code pour le détail de contenu

Le site Web est conçu de telle sorte que l'utilisateur naviguera jusqu'aux détails. La page d'accueil et la page de service ne présentent pas les détails des contenus. La page de détail de contenu affiche toutes les données associées à un même et unique contenu.

Écriture de `fields_content.inc` (configuration des champs de la page de détail)

Écriture de `fields_content.inc` (Listing 16.6) configure les éléments à présenter dans le formulaire affichant le contenu. Le tableau associatif `$fields` fait correspondre les ID d'éléments du formulaire avec les noms affichés. Certains noms de formulaire sont laissés vides car ils sont masqués.

Listing 16.6 : Configuration des éléments et types utilisés pour construire l'affichage.

```
<?php
/* Fichier : fields_content.inc
 * Description : Contient des tableaux (arrays) avec les
 * noms de champs et les éléments de formulaire
 * des pages de contenu.
 */
include_once("functions_main.inc");

$fields = array("content_id" => "",
               "dept_id"    => "",
               "type_id"    => "",
               "title"      => "Titre $area_name",
               "description" =>
                   "Description $area_name",
               "content_date" => "Date $area_name",
               "create_date"  => "Date de création",
               "created_by"   => "Créé par",
               "last_upd_date" => "Dernière modification",
               "last_upd_by"  => "Dernière modification par"
            );

$types = array("content_id"    => "hidden",
               "dept_id"       => "hidden",
               "type_id"       => "hidden",
               "content_date"  => "date",
               "title"         => "text",
               "description"   => "textarea",
               "create_date"   => "datelabel",
               "created_by"    => "label",
               "last_upd_date" => "datelabel",
               "last_upd_by"   => "label"
            );
```



```

foreach($fields as $field => $value)                                #21
{
    $type = $types[$field];
    if ($type != "hidden" && !$edit)
    {
        $type = $type == "date" ? "datelabel" : "label";
    }
    switch ($type) {                                                #28
        case "hidden":
            echo "<input type='hidden' "
                . "name=\"\$field\" value=\"\".@$$field.\">";
            break;

        case "datelabel":
            if (!isset($$field) || $$field == "")
                break;
            $$field = date("m/d/Y", time($$field));

        case "label":
            echo "<tr><td nowrap valign=top
                style=\"text-align: right;
                font-weight: bold\">{$value}</td>
                <td valign=top>\".@$$field.\"</td></tr>";
            break;

        case "date":
            if (isset($$field) && $$field != "")
                $$field = date("m/d/Y", time($$field));

        case "text":
            echo "<tr><td valign=top nowrap
                style=\"text-align: right;
                font-weight: bold\">{$value}</td>
                <td valign=top>
                <input type='\$type'
                    name='\$field'
                    value='\".@$$field.\"'
                    size='{length[\$field]}'
                    maxsize='{length[\$field]}'>";

            if ($type == "date")
                echo " <i>(mm/jj/aaaa)</i>";
            echo "</td></tr>";
            break;

        case "textarea":
            echo "<tr><td nowrap style=\"text-align: right;
                font-weight: bold\">{$value}</td>
    
```

```

        <td><textarea name='$field' cols=40 rows=8>"
    . @$field
    . "</textarea>
        </td></tr>";
    }
}
?>
<input type="hidden" name="browse_level" value="details">
    <tr><td colspan="2" style="text-align: center">
        <p style="margin-top: .05in">
</table>

```

Dans les explications suivantes, les numéros renvoient aux numéros de lignes de `content_form.inc` (Listing 16.7) :

- #10** Débute un bloc qui, si `message_2` est défini dans le tableau `$_GLOBALS`, affiche le message d'erreur stocké dans l'élément de tableau.
- #19** La variable `$edit` est réglée sur `TRUE` ou `FALSE`. Si l'utilisateur est administrateur et qu'il a cliqué sur le lien Modifier pour accéder à la page de détail, `$edit` est réglée sur `TRUE` et la page est affichée en mode d'édition. Sinon, `$edit` est réglée sur `FALSE` et la page apparaît en lecture seule.
- #21** Débute une boucle traversant les éléments du formulaire. L'HTML est construit, en se basant sur les attributs définis dans le tableau `$types`, et si l'utilisateur a le droit de modifier le contenu.
- #28** Débute une instruction `switch` qui détermine le type de l'élément et construit l'HTML en se basant sur ce type.

Écriture de Admin.php (manipulation des données)

Lors de l'écriture de code prévu pour opérer des changements aux données, vous ne serez jamais trop exigeant dans votre contrôle des droits de modification des données de l'utilisateur, de la validité des données soumises, de la validité des relations entre les données apparentées. Une certaine validation est opérée dans cet exemple de CMS mais, pour la simplicité, un ensemble complet n'a pas été mis

en place. Dans l'idéal, vous devriez étudier chaque ligne de code et vous demander si quelqu'un pourrait intentionnellement ou accidentellement contourner les vérifications. Vous disposez de la fonction intégrée `assert` pour vérifier des hypothèses de code pendant le débogage.

L'intelligence du CMS réside dans le fichier `Admin.php`. C'est dans ce code que s'opère l'ajout, la modification et la suppression des contenus. Le formulaire construit dans le fichier `content_form.inc` poste ses éléments de formulaire vers `Admin.php`. `Admin.php` doit valider les données, rediriger l'utilisateur vers l'écran suivant et enregistrer les données dans la base de données.

Voici le flux élémentaire du fichier PHP d'administration (`Admin.php`) (le Listing 16.8 en présente par ailleurs le code complet) :

Boucle pour traverser les éléments de formulaire soumis.

Déterminer l'action de l'utilisateur :

selon (action)

cas "delete":

- 1 Supprimer dans la table `Content` les détails du contenu que l'utilisateur demande d'effacer.
- 2 Supprimer dans la table `Content_Download` tout téléchargement associé avec le contenu que l'utilisateur supprime.

cas "Save Changes":

- 1 Organiser et valider les éléments de formulaire soumis
- 2 Si l'utilisateur enregistre un nouveau contenu, insérer une nouvelle ligne dans la table `Content`.
- 3 Si l'utilisateur enregistre un contenu existant, actualiser la ligne dans la table `Content`.
- 4 Boucler dans les fichiers qui ont été téléchargés vers le serveur et ajouter leurs détails à la table `Content_Download`.

cas "DeleteDownload":

- 1 Supprimer un contenu de la table `Content_Download`

Listing 16.8 : Actualisation de la base de données et enregistrement des fichiers téléchargés vers le serveur.

```

<?php
/* Fichier: Admin.php
 * Description: Réalise les tâches de manipulation de données,
 *             comme la création, la modification ou la suppression
 *             de contenu
 */
session_start();
include_once("functions_main.inc");

foreach ($_POST as $name => $value)                                #9
    $$name = $value;
foreach ($_GET as $name => $value)                                #11
    $$name = $value;

if (!isset($action))                                             #14
    header("Location: CompanyHome.php");

if (!isset($create_date))                                        #17
    $create_date = date("Y-m-d", time());
else
    $create_date = time($create_date);
if (!isset($content_date))
    $content_date = date("Y-m-d", time());
else
    $content_date = strtotime($content_date);

$content_date = date("Y-m-d", $content_date);
$last_upd_date = date("Y-m-d", time());

if (!isset($created_by))
    $created_by = $_SESSION["user_name"];

$last_upd_by = $_SESSION["user_name"];

$cxn = Connect_to_db("Vars.inc");
switch ($action)                                                  #35
{
    case "delete":                                              #37
        $sql = "DELETE FROM Content
                WHERE content_id=$content_id";

        mysqli_query($cxn, $sql);
        $sql = "DELETE FROM Content_Download
                WHERE content_id=$content_id";
        mysqli_query($cxn, $sql);

```



```

break;

case "Enregistrer les changements":
    $message_2 = "";
    #48

    if ($content_date <= 0)
        $message_2 = "<li>Date de contenu invalide";

    if ($title == "")
        $message_2 .= "<li>Le titre ne peut être vierge";

    if ($message_2)
        $message_2 = "Merci de corriger ces erreurs : $message_2";

    if ($message_2 != "")
    {
        include("CompanyHome.php");
        exit();
    }
    if ($content_id)
    {
        $sql = "UPDATE Content
                SET title = '$title',
                    description = '$description',
                    content_date = '$content_date',
                    last_upd_date = '$last_upd_date',
                    last_upd_by = '$last_upd_by'
                WHERE content_id = $content_id";
        #65
    }
    else
    {
        $sql = "INSERT Content (dept_id, content_type,
                                title, description, content_date,
                                create_date, created_by,
                                last_upd_date, last_upd_by)
                VALUES ($dept_id, $type_id, '$title',
                        '$description', '$content_date',
                        '$create_date', '$created_by',
                        '$last_upd_date', '$last_upd_by')";
        #75
    }

    Connect_to_db("Vars.inc");
    #87

    mysqli_query($cxn, $sql);

    if (!$content_id)
        $content_id = mysqli_insert_id($cxn);
    #92

```

```

foreach ($_FILES as $file)                                #94
{
    $file_name = $file["name"];
    if ($file["size"] <= 0)
        continue;

    $sql = "INSERT Content_Download (content_id, file_name)
           VALUES ($content_id, '$file_name')";
    mysqli_query($cxn, $sql);
    $file_id = mysqli_insert_id($cxn);                    #103
    $dest_dir = "files".DIRECTORY_SEPARATOR.$file_id;
    $dest_file = $dest_dir.DIRECTORY_SEPARATOR.$file_name;

    if(!file_exists($dest_dir))                            #107
    {
        if(!mkdir($dest_dir, 0700, TRUE))
            die ("Archivage du document impossible dans $dest_dir");
    }

    if (!file_exists($dest_file))                          #113
    {
        if (!move_uploaded_file($file["tmp_name"], $dest_file))
            die ("Archivage du document impossible dans $dest_dir");
    }
}

break;
case "DeleteDownload":                                    #121
    $sql = "SELECT a.dept_id, a.content_type
           FROM Content a, Content_Download b
           WHERE b.download_id=$download_id
                 AND a.content_id = b.content_id";

    $results = mysqli_query($cxn, $sql);

    $row = mysqli_fetch_assoc($results);
    $dept_id = $row["dept_id"];
    $type_id = $row["content_type"];

    $sql = "DELETE FROM Content_Download
           WHERE download_id=$download_id";
    mysqli_query($cxn, $sql);

    break;
case "Annuler":
    break;
}

```

#132

```
$query_str = "browse_level=content" #142
. "&dept_id=$dept_id&type_id=$type_id";
header("Location: CompanyHome.php?$query_str");
?>
```

Dans les explications suivantes, les numéros renvoient aux numéros de lignes de Admin.php (Listing 16.8) :

- #9** Débute une boucle traversant les éléments du formulaire qui ont été soumis à Admin.php à l'aide de la méthode de formulaire POST.
- #11** Débute une boucle traversant les éléments du formulaire qui ont été soumis à Admin.php à l'aide de la méthode de formulaire GET.
- #14** Lorsque le formulaire de content_form.inc est soumis à Admin.php, il doit fournir une action. Sans action (définie dans la variable \$action), le code ne sait pas si l'utilisateur veut ajouter, supprimer ou modifier un contenu. Par conséquent, si aucune action n'est fournie, l'utilisateur est renvoyé à la page d'accueil via la directive d'en-tête (header). La directive d'en-tête "Location : chemin d'accès" indique au navigateur de l'utilisateur d'aller à un autre endroit. Il existe d'autres en-têtes bien utiles lors de la conception de sites Web dynamiques. La directive d'en-tête "Pragma : no-cache" indique au navigateur comment gérer la mise en cache. La directive d'en-tête "Expires : temps GMT" assure que le contenu sera rafraîchi régulièrement. (Ces en-têtes ne sont pas utilisés dans ce chapitre.)
- #17** La date de création des nouveaux contenus est élaborée à partir de plusieurs fonctions : time, strtotime et date. La fonction time crée un entier indiquant les millisecondes écoulées depuis le 1 janvier 1970, GMT (également dénommée Époque UNIX). La fonction strtotime sert à analyser une chaîne de date générique pour en extraire l'information d'heure. (Un @-1 retourné par cette fonction signifie que PHP ne peut déterminer l'heure de la chaîne passée en variable.) La fonction date prend une variable time et lui applique une mise en forme.

- #35 Débute un bloc `switch` qui détermine l'action que l'utilisateur veut réaliser sur un contenu. Il peut s'agir d'un ajout de nouveau contenu, d'un ajout de téléchargement, de la suppression d'un contenu, de la suppression d'un téléchargement, ou de la modification des détails d'un contenu.
- #37 Débute un bloc de code supprimant un contenu. Les téléchargements associés sont également effacés (lignes 42 à 44).
- #48 Débute un bloc de code enregistrant les modifications apportées à un contenu. Les lignes 49 à 58 valident certains éléments du formulaire soumis. À la ligne 63, si des problèmes sont détectés lors de la validation du formulaire soumis, l'utilisateur est renvoyé à la page de détails de contenu.
- #65 Les lignes 65 à 74 construisent la requête SQL pour actualiser le contenu existant que l'utilisateur demande à modifier. Le code sait qu'il s'agit d'un contenu à modifier (et pas un nouveau contenu) car la vérification de la variable `$content_id` aboutit.
- #75 Les lignes 75 à 85 construisent la requête SQL pour créer un nouveau contenu.
- #87 Les lignes 87 à 89 établissent la connexion à la base et exécutent la requête SQL pour créer ou actualiser un contenu.
- #92 Les lignes 91 et 92 remplissent la variable `$content_id` lorsque le contenu est un nouvel élément. La fonction `mysql_insert_id` lit dans la base de données la valeur de la colonne d'identité de la ligne la plus récemment insérée.
- #94 Débute une boucle qui enregistre chaque document téléchargé vers le serveur que l'utilisateur a associé. Sur la Figure 16.4 (voir plus haut), on voit une section Ajouter des documents. Pour cela, l'utilisateur doit employer le bouton Parcourir. Par défaut, trois documents téléchargeables sont affichés, mais vous pouvez en ajouter après enregistrement des détails du contenu, si l'utilisateur retourne à l'affichage du contenu en mode modification.
- #103 Le fichier est stocké dans le système de fichiers en utilisant un chemin d'accès qui inclut l'identifiant de la ligne (`$file_id`) dans la table `Content_Download`. À noter, la cons-

tante `DIRECTOR_SEPARATOR`. Cette constante intégrée rend votre code portable sur différents systèmes d'exploitation. Si vous codez en dur un antislache (`\`) comme séparateur de répertoire (comme dans `C:\www\fichiers`), puis que vous devez porter votre code sur un serveur Unix ou Mac, le code échouera du fait d'un chemin d'accès considéré invalide.

- #107 Vérifie l'existence du répertoire de destination. Ce répertoire doit être unique. Il faut donc insérer ici un code de contrôle d'erreur supplémentaire pour afficher une erreur système si le répertoire `$dest_dir` existe déjà. Le répertoire est créé à la ligne 109.

- #113 Vérifie l'existence du fichier de destination. Il n'est pas censé exister mais votre code PHP ne doit jamais faire trop de suppositions sur l'état du système. À la ligne 115, le fichier qui a été téléchargé vers le serveur par l'utilisateur est déplacé vers la destination que le CMS a construite, le répertoire `$dest_dir`. L'appel à la fonction `move_uploaded_file` est nécessaire car PHP supprimera tout fichier téléchargé vers le serveur et stocké dans l'emplacement temporaire (déterminé par la valeur `upload_tmp_dir` dans le fichier `php.ini`). Le code PHP de notre exemple ne limite pas la taille des fichiers téléchargés vers le serveur. Vous pouvez le faire de deux manières :
 - En changeant le paramètre `upload_max_filesize` dans le fichier `php.ini`.
 - En ajoutant au formulaire HTML un champ d'entrée masqué `MAX_FILE_SIZE`, avec saisie d'une valeur (en octets) dans ce champ d'entrée HTML.

- #121 Bloc de code pour supprimer un document téléchargeable. La première requête SQL récupère dans la base de données les valeurs `dept_id` et `type_id` du document téléchargeable à effacer. Ainsi, la page suivante affichée dispose de suffisamment d'informations pour conserver l'emplacement de l'utilisateur dans l'arborescence du site.

- #132 Supprime de la base de données les informations se rapportant au document téléchargeable. Notez que cela n'efface pas le fichier du système de fichiers. Pour cela, le code doit lire le chemin d'accès du document téléchargeable (avant effacement de la ligne dans la base de données), puis utiliser la fonction `unlink` pour supprimer le fichier véritable.

#142 À ce stade de l'exécution du code, toutes les manipulations de données ont été opérées, et le code redirige l'utilisateur (via la directive d'en-tête `Location`) vers la page d'accueil.

Création de l'application CMS : approche orientée objet

Lors de la conception d'un système orienté objet, le concepteur peut définir le modèle objet en premier – propriétés, méthodes et constructeurs des objets – sans écrire tout le code qui se trouvera dans les objets. En commençant par concevoir les objets de haut niveau, les concepteurs, programmeurs et utilisateurs pourront, conceptuellement, voir le système et, c'est ce que l'on espère, trouver les failles de conception avant d'écrire un gros volume de code. En outre, lorsqu'un concept de modèle objet existe, les tâches de codage peuvent être scindées entre plusieurs programmeurs. Tant que le programmeur connaît les méthodes d'un objet, il est capable d'écrire du code exploitant cet objet, même si c'est un autre qui a écrit l'objet.

Écriture du modèle objet

La section "Les objets" présente les classes qui constituent le modèle objet du CMS de ce chapitre. Le code orienté objet réutilise une partie du code procédural.



Si vous consultez le DDL défini plus haut dans ce chapitre (voir la section "Conception de la table `Content_Type`"), vous verrez une corrélation directe entre les tables de la base de données et les classes. Cela dit, le modèle objet d'une application ne reflète pas toujours la structure de la base de données exploitée.

Notre exemple de CMS se compose de services, zones de contenu, contenus, et documents téléchargeables. Chacune de ces entités peut être représenté à l'aide d'un objet. Par exemple, un objet `Department` (service) aura des méthodes pour fournir le nom et la description d'un service.

Les objets

Les classes `Department`, `ContentType`, `ContentItem` et `ContentDownload` aident à constituer le modèle objet du CMS. Voici leur description, ainsi que celles d'autres classes :

- ✓ **Department** : Cette classe représente un service de l'entreprise. L'exemple de CMS présenté gère uniquement le nom du service, sa description, et son identifiant (ID) sous-jacent dans la base de données. Cette classe ne possède que des méthodes en lecture seule, car notre exemple de CMS ne dispose pas de mécanisme pour modifier la liste des services. La liste des services est fournie par la table `Department`. Une amélioration possible de l'exemple de CMS serait un module d'administration permettant aux administrateurs d'ajouter de nouveaux services.
- ✓ **ContentType** : Dans le CMS, un utilisateur peut visionner diverses rubriques, comme Actualités, Événements, FAQ. La classe `ContentType` permet de catégoriser les contenus. La classe contient un ID et un nom. La liste des types de contenus est fournie par la table `Content_Type`. Une amélioration possible de l'exemple de CMS serait un module d'administration permettant aux administrateurs d'ajouter de nouveaux types de contenu.
- ✓ **ContentItem** : Cette classe encapsule les données créées ou modifiées par les utilisateurs. Les données sous-jacentes sont stockées dans la table `Content` de la base de données. Cette classe possède des fonctions `save` et `delete` pour refléter dans la base de données les changements apportés à l'objet.
- ✓ **ContentDownload** : Cette classe, représentant les données stockées dans la table `Content_Download`, expose un nom de fichier et un ID.
- ✓ **BaseInfo** : Les classes ci-dessus ont certaines choses en commun : un nom et un identifiant. La classe `BaseInfo` est le dénominateur commun entre les autres classes. Cette classe vous donne la possibilité de simplifier les autres classes car le code commun a été centralisé dans une classe.
- ✓ **Database** : Cette classe encapsule l'établissement de la connexion à la base de données. La fonction `getConnection` sert à obtenir la connexion à la base de données. Un avantage de la

programmation orientée objet (POO) est que l'implémentation des détails peut se trouver dans une *boîte noire*. En d'autres termes, le code qui utilise la classe `Database` n'a pas besoin de savoir comment la classe établit les connexions à la base de données. Le programmeur qui utilise les classes ne se préoccupe que des méthodes proposées par les objets.

- ✓ **WebForm** : La classe `WebForm` fournit les données à l'application. Elle recueille et traite l'information tapée par l'utilisateur.

Création de méthodes statiques de recherche

Comment transformer des données de la base en une représentation objet dans votre code PHP ? Il existe plusieurs démarches. Il est possible de construire un unique fichier PHP pour rechercher des données et instancier les objets appropriés. Vous placez le code d'instanciation de l'objet à l'endroit où le besoin d'objet existe. Dans l'exemple de ce chapitre, c'est la démarche de factorisation qui est employée. Une *méthode de factorisation* a pour rôle de prendre des paramètres de recherche et de retourner le ou les objets désirés. Les méthodes de factorisation ont un rôle simple, sans ambiguïté : elles débitent des objets. Les *méthodes de recherche* ont une portée statique. Cela signifie que les méthodes de factorisation ne requièrent pas un objet instancié pour s'exécuter. Or, pour les méthodes de classe ordinaires, l'objet doit être instancié (créé) avec le mot clé `new` avant que ses méthodes soient utilisables. Il n'est pas nécessaire d'exécuter les méthodes statiques dans un contexte d'objet.

Les méthodes statiques de recherche des objets du modèle objet de ce chapitre se trouvent dans le code des objets. Par conséquent, chaque classe a pour rôle de trouver les données sous-jacentes de la base de données et d'instancier les objets qui représentent ces données. Par exemple, l'objet `Department` possède une méthode `findAll` qui retourne un tableau d'objets `Department`. La méthode `findById` retourne un objet (ou `NULL` si, dans la base, aucune donnée n'a été trouvée correspondant à l'ID fourni en paramètre dans l'appel de méthode). Le nom de chaque méthode de recherche commence par `find` (chercher).

Une méthode statique est appelée à l'aide du format `NomDeClasse::NomDeMéthodeStatique()`. L'appel est différent de celui d'une méthode objet, qui ressemble à `$NomDeVariable->NomDeMéthode()`. Voici

quelques autres exemples de code utilisant des appels à des méthodes objets et des appels à des méthodes statiques :

```
# Instantiation d'objet
$instance_dobjet = new MyObject();

# Appel de méthode d'objet
echo $instance_dobjet->getDescription();

# Appel de méthode statique, retourne un
# tableau d'objets
$tableau_dobjets = MyObject::findAllData();

# Appel de méthode statique, retourne un objet
$instance_dobjet = MyObject::findById($id);

# Appel de méthode d'objet
echo $instance_dobjet->getName();
```

Création de méthodes de consultation et de configuration

Les objets révèlent leurs détails par le biais de *méthodes de consultation* (getter methods). Ces méthodes, dont le nom commence habituellement par *get*, retournent les attributs d'un objet. Si vous conceviez un objet *Personne*, vous mettriez par exemple en place ces méthodes de consultation : *getPrenom*, *getNom*, *getNumeroID*, *getAge*, etc. *Les méthodes de configuration* (setter methods), au contraire, sont des méthodes qui permettent de définir les attributs d'un objet. Pour un objet *Personne*, on pourrait avoir les méthodes de configuration *setPrenom*, *setNom*, *setService*, *setAge*, etc.

Écriture d'une classe de données élémentaire

Lors de la création d'un modèle objet, il est souvent possible de *factoriser* (ou *généraliser*) dans une classe de base les attributs et fonctions communs. Dans cet exemple de CMS, chaque objet de données a un identifiant et un nom. Ces propriétés, ainsi qu'une fonction utilitaire statique pour obtenir une connexion à une base de données, constitueront la classe de base, *BaseInfo.class*.

Les propriétés

Les propriétés de `BaseInfo` stockent l'ID et le nom d'un élément.

```
protected $id;  
protected $name;
```

La première propriété, `$id`, est l'identifiant sous-jacent de la colonne. La propriété `$name` est un nom ou titre pour l'élément. Notez que ces propriétés sont *protected* (protégées) : elles ne sont pas accessibles aux utilisateurs de l'objet. En revanche, les sous-classes ont accès à ces propriétés.

Le code

Le Listing 16.9 présente le code de la classe `BaseInfo`. Chaque méthode est étudiée en détail après le listing.

Listing 16.9 : La classe `BaseInfo`.

```
<?php  
/* Nom : BaseInfo.class  
 * Description : Classe de base à partir de laquelle  
 * les objets de données peuvent être étendus.  
 * L'identifiant (ID) et le nom sont communs à  
 * tous les objets de données. En outre, la  
 * connexion à la base de données peut être  
 * obtenue de cette classe.  
 */  
include_once("Database.class");  
  
class BaseInfo  
{  
    protected $id;  
    protected $name;  
  
    function __construct($id, $name)  
    {  
        $this->id = $id;  
        $this->name = $name;  
    }  
  
    protected static function getConnection()  
    {  
        $db = new Database("Vars.inc");
```



```

$db->useDatabase("IntranetCMS");
return $db->getConnection();
}

public function getId()
{
    return $this->id;
}

public function getName()
{
    return $this->name;
}

}
?>

```

Le constructeur

Le constructeur nécessite un identifiant et un nom. Ces paramètres sont utilisés pour affecter des valeurs aux propriétés de l'objet.

getConnection

Cette méthode instancie un objet Database et utilise sa méthode `getConnection` pour obtenir une connexion à la base de données. La méthode est statique car les méthodes de recherche, statiques également, doivent obtenir une connexion à une base de données pour réaliser une recherche.

getId, getName

La valeur `$id` est exposée par la fonction `getId`. En programmation orientée objet, essayez de déclarer des fonctions pour consulter et configurer les attributs internes au lieu d'exposer les variables de classe en les rendant publiques. Cela renforce la garantie que la classe contrôle strictement son état interne. La méthode `getName` retourne le nom de l'objet.

Écriture de la classe *Department*

La classe *Department* encapsule les informations définissant un service de l'entreprise. Il s'agit du nom du service, de sa description, de son ID.

Les propriétés

Les propriétés de l'objet *Department* stockent les informations sur un seul service de l'entreprise. Voici ces propriétés :

```
protected $description;
```

Notez que cette classe définit une propriété alors qu'elle en possède réellement trois. Les deux autres propriétés (ID et name) sont définies dans la classe *BaseInfo*. La classe *Department* étant une sous-classe de la classe *BaseInfo*, la classe *Department* hérite des propriétés de la classe *BaseInfo*.

Le code

Le Listing 16.10 présente le code de la classe *Department*. Chaque méthode est étudiée en détail après le listing.

Listing 16.10 : La classe *Department*.

```
<?php
/* Nom : Department.class
 * Description : Classe contenant les détails sur un service
 *              et les contenus de ce service.
 */
include_once("BaseInfo.class");

class Department extends BaseInfo
{
    protected $description;                                     #10

    function __construct($id, $name, $desc)
    {
        parent::__construct($id, $name);                       #14
        $this->description = $desc;
    }

    public function getDescription()                             #18
```

```

{
    return $this->description;
}

public static function findAll()
{
    $cxn = parent::getConnection();
    $sql = "SELECT dept_id, name, description
           FROM Department
           ORDER BY name";
    $results = $cxn->query($sql);
    if (!$results)
    {
        throw new Exception("Problème d'accès aux données : ",
                             $cxn->error);
    }
    $depts = array();
    while ($row = $results->fetch_assoc())
    {
        $depts[] = new Department($row['dept_id'],
                                   $row['name'], $row['description']);
    }
    return $depts;
}

public static function findById($id)
{
    $cxn = parent::getConnection();
    $sql = "SELECT dept_id, name, description
           FROM Department
           WHERE dept_id = $id";
    $results = $cxn->query($sql);
    if (!$results)
    {
        throw new Exception("Problème d'accès aux données : " +
                             $cxn->error);
    }
    if ($row = $results->fetch_assoc())
    {
        return new Department($row['dept_id'],
                               $row['name'], $row['description']);
    }
    return NULL;
}
}

```

Le constructeur

La classe de base n'a pas toutes les propriétés nécessaires pour représenter un service, aussi une propriété supplémentaire est-elle déclarée pour représenter la description détaillée d'un service (ligne 10). Rappelons que les propriétés `$id` et `$name` de la classe de base `BaseInfo` sont également accessibles dans cette classe. Le constructeur est nécessaire pour gérer la propriété supplémentaire `$description`. Le constructeur de la classe parent est appelé à la ligne 14 pour configurer les propriétés `$id` et `$name`.

getDescription

À la ligne 18, cette nouvelle méthode expose la propriété `$description`.

findAll, findById

Ces méthodes sont des fonctions de recherche statiques. Dans le modèle objet de cet exemple, le code qui effectue la recherche se trouve dans les classes elles-mêmes. La fonction `findAll` retourne un tableau des objets instanciés satisfaisant les paramètres de recherche. À la ligne 25, une connexion à la base de données est obtenue en utilisant une fonction définie dans la classe parent. À la ligne 36, une boucle remplit le tableau retourné avec de nouveaux objets `Department`. `findById` retourne un objet s'il existe dans la base de données une ligne correspondant à l'identifiant.

Écriture de la classe ContentType

Cette classe est très simple. Elle représente un type de contenu disponible dans le CMS. Un type de contenu possède uniquement un ID et un nom. Ces propriétés ont déjà été codées dans la classe de base, aussi le seul code supplémentaire dans cette classe est le code pour gérer la recherche et l'instanciation des objets `ContentType`.

Les propriétés

La classe `ContentType` ne définit pas de propriétés, elle utilisera les propriétés définies dans la *super classe* (ou *classe parent*), les propriétés `$id` et `$name`.

Le code

Le Listing 16.11 contient le code complet de la classe `ContentType`. Chaque méthode est étudiée en détail après le listing.

Listing 16.11 : La classe `ContentType`.

```
<?php
/* Nom : ContentType.class
 * Description : Classe contenant les détails sur un type
 *              de contenu accessible dans le CMS.
 */
include_once("BaseInfo.class");

class ContentType extends BaseInfo
{
    public static function findAll() #11
    {
        $cxn = parent::getConnection();
        $sql = "SELECT type_id, name
                FROM Content_Type
                ORDER BY name";
        $results = $cxn->query($sql);
        if (!$results)
        {
            throw new Exception("Problème d'accès aux données : " +
                                $cxn->error());
        }
        $types = array();
        while ($row = $results->fetch_assoc())
        {
            $types[] = new ContentType($row['type_id'],
                                       $row['name']);
        }
        return $types;
    }

    public static function findById($id) #32
    {
        $cxn = parent::getConnection();
        $sql = "SELECT type_id, name
                FROM Content_Type
                WHERE type_id = $id";
        $results = $cxn->query($sql);
        if (!$results)
        {
```



```

        throw new Exception("Problème d'accès aux données : " +
                               $cxn->error());
    }
    if ($row = $results->fetch_assoc())
    {
        return new ContentType($row['type_id'], $row['name']);
    }
    return NULL;
}

}
?>

```

Le constructeur

Cette classe ne définit pas explicitement un constructeur car celui défini dans la classe parent (`BaseInfo.class`) satisfait également les besoins de cette classe. Il n'existe donc pas de code de constructeur dans ce fichier de classe.

findAll, findById

Ces méthodes sont des fonctions de recherche statiques. `findAll` (ligne 11) retourne un tableau d'objets, et `findById` (ligne 32) retourne un objet (si la recherche trouve une rangée dans la base de données).

Écriture de la classe ContentItem

La classe `ContentItem` (élément de contenu) est la classe de base représentant les différents contenus disponibles dans le CMS. Cette classe est extensible car plus ou moins de détails peuvent être associés aux divers contenus.

Les propriétés

Les propriétés de l'objet `Department` stockent les informations sur un seul service de l'entreprise.

```
protected $description;
protected $content_date;
protected $create_date;
protected $created_by;
protected $last_upd_date;
protected $last_upd_by;
```

Les propriétés `$id` et `$name` sont héritées de la classe `BaseInfo`, aussi ne sont-elles pas redéfinies ici. La propriété `$description` est une chaîne qui représente le corps principal d'un contenu. La propriété `$content_date` indique la date à laquelle le contenu sera pertinent ; dans le cas d'un contenu d'événement, la propriété `$content_type` est utilisée pour signifier la date de l'événement indiqué. Les propriétés `$created_by`, `$create_date`, `$last_upd_date` et `$last_upd_by` sont utilisées pour l'audit élémentaire du contenu.

Le code

Le Listing 16.12 présente le code de la classe `ContentItem`. Chaque méthode est étudiée en détail après le listing.

Listing 16.12 : La classe `ContentItem`.

```
<?php
/* Nom : ContentItem.class
 * Description : Classe contenant les détails sur
 *              un contenu particulier.
 */
include_once("BaseInfo.class");

class ContentItem extends BaseInfo
{
    protected $description;
    protected $content_date;
    protected $create_date;
    protected $created_by;
    protected $last_upd_date;
    protected $last_upd_by;
}
```

#10

```

function __construct($id, $title, $desc,                                     #17
    $content_date, $created_on, $created_by,
    $last_upd_on, $last_upd_by)
{
    parent::__construct($id, $title);

    $this->description = $desc;
    $this->content_date = $content_date;
    $this->create_date = $created_on;
    $this->created_by = $created_by;
    $this->last_upd_date = $last_upd_on;
    $this->last_upd_by = $last_upd_by;
}

public function setName($name)                                           #31
{
    $this->name = $name;
}

public function getDescription()                                         #36
{
    return $this->description;
}

public function setDescription($desc)
{
    $this->description = $desc;
}

public function getContentDate()
{
    return date("m/d/y", strtotime($this->content_date));
}

public function setContentDate($content_date)
{
    $this->content_date = strtotime($content_date);
}

public function getCreationDate()
{
    return date("m/d/y", strtotime($this->create_date));
}

public function setCreationDate($create_date)
{
    $this->create_date = strtotime($create_date);
}

```

```

public function getCreatedBy()
{
    return $this->created_by;
}

public function getLastUpdDate()
{
    return date("m/d/y", strtotime($this->last_upd_date));
}

public function getLastUpdBy()
{
    return $this->last_upd_by;
}

public function setLastUpdBy($last_upd_by)
{
    $this->last_upd_by = $last_upd_by;
}

public static function findById($content_id)                                     #86
{
    $cxn = parent::getConnection();
    $sql = "SELECT content_id, title, description,
                content_date, create_date,
                created_by, last_upd_date, last_upd_by
            FROM Content
            WHERE content_id = $content_id";

    $results = $cxn->query($sql);
    if (!$results)
    {
        throw new Exception("Problème d'accès aux données : " +
                               $cxn->error);
    }
    $row = $results->fetch_assoc();

    return ContentItem::getContentItem($row);                                     #103
}

public static function findByDeptType($dept_id,
                                       $content_type)
{
    $cxn = parent::getConnection();
    $sql = "SELECT content_id, title, description,
                content_date, create_date,
                created_by, last_upd_date, last_upd_by
            FROM Content

```

```

        WHERE content_type = $content_type
            and dept_id = $dept_id";
$results = $cxn->query($sql);
if (!$results)
{
    throw new Exception("Problème d'accès aux données : " +
        $cxn->error);
}
$ret_array = array();
while ($row = $results->fetch_assoc()) #123
    $ret_array[] = ContentItem::getContentItem($row);

return $ret_array;
}

public static function getContentItem($row) #129
{
    if (!$row)
        return NULL;

    return new ContentItem($row['content_id'], $row['title'],
        $row['description'], $row['content_date'],
        $row['create_date'], $row['created_by'],
        $row['last_upd_date'], $row['last_upd_by']);
}

public function save($dept_id = NULL, $type_id = NULL) #140
{
    if (isset($this->id)) #142
    {
        $sql = "UPDATE Content
            SET title = '$this->name',
                description = '$this->description',
                content_date = '$this->content_date',
                last_upd_date = now(),
                last_upd_by = '$this->last_upd_by'
            WHERE content_id = $this->id";
    }
    else #152
    {
        $sql = "INSERT Content (dept_id, content_type,
            title, description, content_date,
            create_date, created_by,
            last_upd_date, last_upd_by)
            VALUES ($dept_id, $type_id,
                '$this->name',
                '$this->description', '$this->content_date',
                now(), '$this->created_by',

```



```

        now(), '$this->last_upd_by');"
    }

    $cxn = parent::getConnection();
    $cxn->query($sql);

    if (!isset($this->id))
        $this->id = $cxn->insert_id;

    return $this->id;
}

public function delete() #174
{
    if (!isset($this->id))
        return;

    $sql = "DELETE FROM Content
           WHERE content_id = $this->id";
    $cxn = parent::getConnection();
    $results = $cxn->query($sql);
    if (!$results)
    {
        throw new Exception("Problème de suppression de données : " +
                             $cxn->error);
    }

    #188
    $sql = "DELETE FROM Content_Download
           WHERE content_id = $this->id";
    $results = $cxn->query($sql);
    if (!$results)
    {
        throw new Exception("Problème de suppression de données : " +
                             $cxn->error);
    }
}

}
?>

```

Le constructeur

Le constructeur configure les propriétés de la classe `ContentItem` à la ligne 17. Rappelons que les propriétés `$id` et `$name` ne sont pas définies dans cette classe car elles le sont déjà dans la classe `parent`.

Dans le constructeur, le constructeur de la classe parent est également appelé pour que les propriétés de la classe de base soient correctement configurées.

setName, setDescriptionn, setContentDate, setCreationDate, setLastUpdBy

Les méthodes de configuration changent l'état interne de l'objet. Par exemple, un appel à `setName` à la ligne 31 change la propriété `$name`, qui est réellement définie dans la classe `BaseInfo`. Ces fonctions de configuration n'effectuent pas de modifications persistantes aux données dans la base de données. Cela relève de la responsabilité des fonctions `save` et `delete`.

getContentDate, getCreationDate, getLastUpdBy, getLastUpdDate

Les méthodes de consultation exposent les propriétés de l'objet. Sans ces méthodes, le monde extérieur serait incapable de lire les détails des propriétés d'objets. À la ligne 36, la méthode `getDescription` retourne la description détaillée du contenu.

findById, findByDeptType

La ligne 106 commence la définition d'une fonction qui retourne un tableau d'objets `ContentItem` correspondants à un service et un type de contenu donnés, alors que la méthode `findById` (ligne 86) retourne un seul objet `ContentItem` en se basant sur l'ID du contenu.

save, delete

La fonction `save` a pour rôle d'enregistrer un objet `ContentItem`, nouveau ou modifié, dans la base de données ; la fonction `delete` supprime ses détails de la base de données. À la ligne 142, l'appel de fonction `isset` permet à la fonction `save` de décider si l'objet `ContentItem` (`$this`) est un objet nouveau ou existant dans la base de données. Si l'ID de l'objet n'a pas encore été défini, le code considère que `ContentItem` est un nouveau contenu.

getContentItem

La fonction `getContentItem` est utilisée par les fonctions de recherche pour instancier un seul objet `ContentItem` à partir d'une seule rangée retournée par la base de données, comme à la ligne 103. La fonction `getContentItem` simplifie le code qui instancie un nouvel objet.

Écriture de la classe ContentDownload

Il s'agit d'un objet simple, qui représente une seule ligne dans la base de données. Un élément `ContentDownload` représente une rangée dans la table `Content_Download`. Notez que cet objet ne définit pas un constructeur car le constructeur défini dans la classe parent remplit les besoins de cette classe. Seules quelques fonctions de recherche supplémentaires sont nécessaires, ainsi que des méthodes de manipulation de données.

Les propriétés

La classe `ContentDownload` ne définit aucune propriété, elle utilise les propriétés définies dans la classe parent : `$id` et `$name`.

Le code

Le Listing 16.13 présente le code de la classe `ContentDownload`. Chaque méthode est étudiée en détail après le listing.

Listing 16.13 : La classe ContentDownload.

```
<?php
/* Nom : ContentDownload.class
 * Description : Classe contenant les détails sur un
 *             fichier téléchargeable associé à un contenu.
 */
include_once("BaseInfo.class");

class ContentDownload extends BaseInfo
{
    public static function findById($content_id)           #10
    {
        $ret_arr = array();
```

```

$cxn = parent::getConnection();
$sql = "SELECT download_id, file_name
        FROM Content_Download
        WHERE content_id = $content_id";
$results = $cxn->query($sql);
if (!$results)
{
    throw new Exception("Problème d'accès aux données : " +
                        $cxn->error);
}
while($row = $results->fetch_assoc())
{
    $download_id = $row["download_id"];
    $file_name = $row["file_name"];

    $ret_arr[] =
        new ContentDownload($download_id, $file_name);
}

return $ret_arr;
}

public static function findById($file_id)                                #36
{
    $ret_arr = array();

    $cxn = parent::getConnection();
    $sql = "SELECT download_id, file_name
            FROM Content_Download
            WHERE download_id = $file_id";
    $results = $cxn->query($sql);
    if (!$results)
    {
        throw new Exception("Problème d'accès aux données : " +
                            $cxn->error);
    }
    if ($row = $results->fetch_assoc())
    {
        $download_id = $row["download_id"];
        $file_name = $row["file_name"];

        return new ContentDownload($download_id, $file_name);
    }

    return NULL;
}

public function save()                                                #61

```

```

{
    $sql = "INSERT Content_Download (content_id, file_name)
        VALUES ($this->id, '$this->name')";

    $cxn = parent::getConnection();
    $results = $cxn->query($sql);
    if (!$results)
    {
        throw new Exception("Problème d'enregistrement des données : " +
            $cxn->error);
    }

    $this->id = $cxn->insert_id; #74

    return $this->id;
}

public function delete() #79
{
    $sql = "DELETE FROM Content_Download
        WHERE download_id=$this->id";

    $cxn = parent::getConnection();
    $results = $cxn->query($sql);
    if (!$results)
    {
        throw new Exception("Problème de suppression de données : " +
            $cxn->error);
    }
}
}
?>

```

Le constructeur

Cette classe ne définit pas explicitement un constructeur car celui défini dans la classe parent (`BaseInfo.class`) satisfait également les besoins de cette classe. L'écriture de code orienté objet peut épargner un gros volume de saisie.

findByContentId, findById

Ces méthodes sont des fonctions de recherche statiques. La méthode `findByContentId` (ligne 10) retourne un tableau d'objets ;

`findById` (ligne 36) retourne un objet (si la recherche trouve une rangée dans la base de données).

save, delete

Les méthodes `save` (ligne 61) et `delete` (ligne 79) veillent à ce que les données soient enregistrées ou supprimées dans la base de données.

Écriture de la classe Database

La classe `Database` fournit la connexion à la base de données où les données d'utilisateur, de service et de contenu sont stockées. Cette classe a été développée au Chapitre 12. Pour en étudier le code, voir le Listing 12.4.

Voici les méthodes fournies par la classe `Database` :

- ✓ **Le constructeur** : Cette méthode établit une connexion à une base de données MySQL. Il faut lui passer les nom d'hôte, nom de compte et mot de passe nécessaires pour accéder à MySQL. Un objet `Database` est créé avec l'instruction suivante :

```
$db = new Database($host,$user,$password);
```

- ✓ **useDatabase** : Cette méthode sélectionne une base de données et stocke son nom. Elle réclame un nom de base de données. Elle vérifie également si la base de données existe, et retourne un message si ce n'est pas le cas.
- ✓ **getConnection** : Cette méthode retourne la connexion établie et stockée dans le constructeur.

Écriture de la classe WebForm

La classe `WebForm` sert à afficher la page présentant les formulaires d'identification et d'inscription. Elle est étudiée au Chapitre 13. Pour son code, voir le Listing 13.6.

Voici les méthodes de la classe `WebForm` utilisée dans cette application :

- ✓ **Le constructeur** : Il stocke les propriétés nécessaires pour afficher correctement le formulaire. Deux fichiers – un fichier de données et un fichier qui définit l'aspect – sont requis. Les deux noms de fichiers sont passés lors de la création de l'objet `WebForm`, et stockés dans deux propriétés. Le passage des données des champs de formulaire n'est pas obligatoire, ce qui produit alors des champs de formulaire vides. Il est possible de créer l'objet à l'aide d'une de ces instructions :

```
$form = new WebForm("fichier.inc","fichier2.inc",$_POST);
$form = new WebForm("fichier.inc","fichier.inc");
```

- ✓ **displayForm** : Cette méthode affiche le formulaire. Elle extrait les données de la propriété `$data`.
- ✓ **checkForBlanks** : Cette méthode vérifie chaque champ du formulaire pour voir s'il contient des informations. Si des champs vides invalides sont trouvés, elle retourne un tableau contenant les noms de champs de ces champs vides.
- ✓ **verifyData** : Cette méthode vérifie chaque champ pour garantir que les informations soumises sont dans un format raisonnable. Si des données invalides sont détectées dans tout champ, la méthode retourne un tableau contenant les messages identifiant les problèmes.
- ✓ **trimData, stripTagsFromData** : Une fonction PHP est appliquée à chaque valeur de la propriété `$data`. Les valeurs résultantes sont stockées dans `$data`. La fonction `trim` supprime les espaces initiaux et terminaux d'une chaîne. La fonction `strip_tags` supprime les balises HTML de la chaîne, ce qui est important pour la sécurité.

Écriture du code de la page d'identification

La page d'identification orientée objet (`Login-00.php`) du Chapitre 13 (Listing 13.10) a été modifiée pour ce chapitre. Le Listing 16.14 met en lumière les changements.

`Login-00.php` nécessite trois classes supplémentaires non utilisées par les autres programmes de l'application CMS : `Account.class`, `Session.class`, et `Email.class`. Ces classes ne sont pas décrites dans ce chapitre. Pour en savoir plus sur ces classes et la manière dont elles sont employées dans l'application d'identification, voir la section sur l'application orientée objet du Chapitre 13.

Listing 16.14 : Changements dans Login-00.php.

```

Ligne 38 :
$db->useDatabase("intranetCMS"); #38
Ligne 39 :
$sacct = new Account($db->getConnection(), "Dept_User");
Lignes 83 et 174 :
header("Location: CompanyHome-00.php");
Ligne 154 :
$sess->storeVariable("user_name", $newdata['user_name']);
$sess->storeVariable("user_dept", $newdata['dept_id']);

```

Suit une description des lignes modifiées par rapport au fichier Login-00.php du Chapitre 13 :

- #38** Change le nom de la base de données.
- #39** Le nom de la table de la base de données stockant les détails sur l'utilisateur a été changé en Dept_User.
- #83** [et **#174**] En cas d'identification ou d'inscription réussie, le navigateur est redirigé vers le programme principal, CompanyHome-00.php. La directive d'en-tête envoie une commande au navigateur. Si l'HTML est envoyé au navigateur avant la directive d'en-tête, l'appel de la directive provoquera une erreur car tous les en-têtes doivent être envoyés avant que du contenu soit affiché.
- #154** Au Chapitre 13, le programme stockait un indicateur dans la session pour indiquer que l'utilisateur avait été authentifié ("auth") ; il stockait aussi l'utilisateur ID ("logname"). L'application CMS de ce chapitre stocke le nom d'utilisateur ("user_name") et l'identifiant de service ("dept_id") dans la variable de session.

Écriture de fields_content.inc et content_form.inc

Les fichiers fields_content.inc (voir le Listing 16.6) et content_form.inc (voir le Listing 16.7) de la section de code procédural peuvent être ré-utilisés dans l'approche orientée objet.

Écriture du code d'affichage

CompanyHome-00.php est très semblable au CompanyHome.php vu plus haut dans ce chapitre (Listing 16.4). Le code objet gère les détails de lecture et stockage dans la base de données : le code d'affichage s'affranchit de la nécessité de connaître la structure de la base. C'est un des bels aspects du code orienté objet : vous pouvez créer des couches d'abstraction rendant possible de modifier les rouages internes des objets sans avoir à changer les éléments exploitant ces objets dans le modèle objet.

Écriture de CompanyHome-00.php

CompanyHome-00.php (Listing 16.15) a pour rôle de configurer les éléments de données utilisés par company-00.inc, un fichier qui compose l'interface HTML.

Listing 16.15 : Configuration des données d'affichage (CompanyHome-00.php).

```
<?php
/* Programme : CompanyHome-00.php
 * Description : Affiche une page Web à quatre niveaux :
 *      1) page d'accueil, 2) page de service, 3) page
 *      des contenus 4) page des détails de contenu.
 */

if (!isset($_SESSION))
    session_start();

include_once("functions_main.inc");
include_once("Department.class");                                #11
include_once("ContentType.class");
include_once("ContentItem.class");
include_once("ContentDownload.class");                          #14

$page = array(
    "title"      => "Intranet de l'entreprise",
    "header"     => "Intranet de l'entreprise",
    "bottom"     => "Copyright(R) 2005",
    "left_nav_links" => array(),
    "body_links"  => array(),
    "col_headers" => array(),
    "data_rows"   => array(),
);

$admin = FALSE;
```

```

$base_url = "CompanyHome-00.php";
$strail = "<a href='$base_url'>Home</a>";

if (!isset($_SESSION['user_name']))
    header("Location: Login-00.php"); #31
else
{
    if (isset($_SESSION['user_name'])
        && isset($_GET['dept_id']))
    {
        $admin = $_SESSION['user_dept'] == $_GET['dept_id'];
    }

    $left_nav_links = array();

    $page["browse_level"] =
        isset($_GET['browse_level']) ?
            $_GET['browse_level'] : "home";

    switch ($page["browse_level"])
    {
        case "home": #48
            Try
            {
                $departments = Department::findAll(); #51
            }
            catch(Exception $e)
            {
                echo $e->getMessage();
                exit();
            }
            $body_links = "";
            foreach ($departments as $department) #59
            {
                $link = "$base_url?dept_id=" . $department->getId()
                    . "&browse_level=department";
                $page["left_nav_links"][$link] =
                    $department->getName();
                $body_links .= "<li><a href=\"$link\" . $link
                    . \">\" . $department->getName() . "</a> - "
                    . $department->getDescription();
            }
            $page["left_nav_header"] = "Services";
            $page["top"] = "Bienvenue sur l'Intranet";
            $page["body_text"] = "Bienvenue sur l'Intranet "
                . "où chaque service partage du contenu avec "
                . "toute l'entreprise. Vous pouvez modifier le "
                . "contenu de votre service, via une interface "

```



```

    . "simple.<p>Visitez les pages d'accueil des "
    . "services : $body_links";
break;

case "department":
    $dept_id = $_GET['dept_id'];
    try
    {
        $department = Department::findById($dept_id);
    }
    catch(Exception $e)
    {
        echo $e->getMessage();
        exit();
    }
    $dept_name = $department->getName();
    $dept_desc = $department->getDescription();

    $page["left_nav"] = "$dept_name Content";
    $page["body_text"] = "$dept_name - $dept_desc";

    $body_links = "";
    $content_types = ContentType::findAll();
    foreach ($content_types as $content_type)
    {
        $link = "$base_url?dept_id=$dept_id"
            . "&type_id=" . $content_type->getId()
            . "&browse_level=content";
        $page["left_nav_links"][$link] =
            $content_type->getName();
        $body_links .= "<li><a href=\"\" . $link
            . "\">\" . $content_type->getName() . "</a>";
    }
    $page["left_nav_header"] = "Types de contenus";
    $page["top"] = $dept_name;
    $page["body_text"] = "$dept_name - $dept_desc "
        . "<p>Voici les zones de ce "
        . "service : $body_links";
    $strail .= " - <a href='$base_url?dept_id=$dept_id'
        . "&browse_level=department'$dept_name</a>";
break;

case "content":
    $dept_id = $_GET['dept_id'];
    $type_id = $_GET['type_id'];
    try
    {
        $department = Department::findById($dept_id);
    }

```

```

    }
    catch(Exception $e)
    {
        echo $e->getMessage();
        exit();
    }
    $dept_name = $department->getName();
    $body_links = "";
    $page["body_text"] = "";
    try
    {
        $content_types = ContentType::findAll();
    }
    catch(Exception $e)
    {
        echo $e->getMessage();
        exit();
    }
    foreach ($content_types as $content_type)
    {
        $link = "$base_url?dept_id=$dept_id"
            . "&type_id=" . $content_type->getId()
            . "&browse_level=content";
        $page["left_nav_links"][$link] =
            $content_type->getName();
        if ($content_type->getId() == $type_id)
            $area_name = $content_type->getName();
    }
    $page["data_rows"] =
        ContentItem::findByDeptType($dept_id, $type_id);
    if (sizeof($page["data_rows"]) == 0)
    {
        $page["body_text"] = "Pas de contenu $area_name
            dans $dept_name";
    }
    if ($admin)
    {
        $page["body_text"] .=
            "<p>[<a href='$base_url?dept_id=$dept_id'
            . "&browse_level=details&type_id=$type_id"
            . "&content_id=" . $content_id . ">ajouter</a>]";
    }
    $page["col_headers"]["title"] = "Titre $area_name";
    $page["col_headers"]["content_date"] =
        "Date $area_name";
    $page["col_headers"]["create_date"] = "Créé le";
    $page["col_headers"]["created_by"] = "Créé par";
    $page["col_headers"]["last_upd_date"] =

```

```

"Dernière modification le";
$page["col_headers"]["last_upd_by"] =
"Dernière modification par";
$page["left_nav_header"] = "Contenu";
$page["top"] = "$dept_name - $area_name";
$strail .= " - <a href='$base_url?dept_id=$dept_id"
. "&browse_level=department'>$dept_name</a>";
$strail .= " - <a href='$base_url?dept_id=$dept_id"
. "&browse_level=content"
. "&type_id=$type_id'>$area_name</a>";
break;

case "details":
    $dept_id = $_GET['dept_id'];
    $type_id = $_GET['type_id'];
    $department = Department::findById($dept_id);
    $dept_name = $department->getName();
    $content_type = ContentType::findById($type_id);
    $area_name = $content_type->getName();
    $body_links = "";
    $create_date = date("m/d/y", time());
    $created_by = $_SESSION["user_name"];
    $last_upd_by = $_SESSION["user_name"];
    $content_id = $_GET["content_id"];
    $edit = $admin && (@$_GET["edit"] == "true"
        || $content_id == "");
    if ($content_id != "")
    {
        $content_item = ContentItem::findById($content_id);
        $title = $content_item->getName();
        $content_date = $content_item->getContentDate();
        $description = $content_item->getDescription();
        $create_date = $content_item->getCreationDate();
        $created_by = $content_item->getCreatedBy();
        $last_upd_date = $content_item->getLastUpdDate();
        $last_upd_by = $content_item->getLastUpdBy();
        $downloads =
            ContentDownload::findByContentId($content_id);
        foreach($downloads as $download)
        {
            $download_id = $download->getId();
            $file_name = $download->getName();
            $link = "files/$download_id/$file_name";
            $page["left_nav_links"][$link] = $file_name;
            if ($edit)
                $page["left_nav_links"][$link] .= "</a>
                [<a href='\"Admin-00.php\" .
                \"?action=DeleteDownload&download_id=$download_id&\"

```

```

        . "dept_id=$dept_id&type_id=$type_id\"
    >del</a>]";
}
}
foreach ($_GET as $name => $value)
    $name = $value;
$edit = $admin && (@$_GET["edit"] == "true" ||
    $content_id == "");
$page["top"] = "$dept_name - $area_name";
if ($edit)
{
    $page["body_text"] = "<center><u>Ajout document téléchargeable</u>";
    for ($i = 0; $i < 3; $i++)
    {
        $page["body_text"] .=
            "<br><input type='file' name='upload_file$i'>";
    }
    $page["body_text"] .= "
    </center> <p />
    <center>
        <input type='reset' name='action'
            value ='Réinitialiser'>
        <input type='submit' name='action'
            value ='Annuler'>
        <input type='submit' name='action'
            value ='Enregistrer les changements'>
    </center>";
    $page["top"] .= " Modifier/Créer";
}
else
{
    $page["body_text"] =
        "<a href='javascript:history.go(-1)'">Retour</a>";
}
$page["left_nav_header"] = "Documents téléchargés";
$trail .= " - <a href='$base_url?dept_id=$dept_id"
    . "&browse_level=department'">$dept_name</a>";
$trail .= " - <a href='$base_url?dept_id=$dept_id"
    . "&browse_level=content"
    . "&type_id=$type_id'">$area_name</a>";

break;
}
include("company-00.inc");
}
?>

```

Voici une description des lignes numérotées de `CompanyHome-00.php` (Listing 16.15) :

- #11** Les lignes 11 à 14 garantissent que les définitions de classe sont incluses, de façon que le modèle objet détaillé plus haut peut être mis en action.
- #31** Ici, l'application orientée objet d'identification du Chapitre 13 est sollicitée.
- #48** L'affichage de la page d'accueil est géré dans ce bloc de l'instruction `switch`.
- #51** L'appel de la méthode statique trouve tous les services de l'intranet. Le tableau retourné par cette fonction contient des objets `Department`.
- #59** Chaque objet `Department` est examiné dans cette boucle, pour afficher le nom du service et sa description dans la page d'accueil.
- #79** L'affichage de niveau service est géré dans ce bloc de l'instruction `switch`.
- #83** À ce stade de l'exécution du programme, l'utilisateur a cliqué sur un service. La variable `$department` devient une instantiation de la classe `Department` au retour de la fonction `findById`.
- #90** Les lignes 90 et 91 lisent les détails concernant le service.
- #97** La ligne 97 devient la liste des types de contenus du CMS, chaque type de contenu étant représenté par un objet `ContentType`.
- #98** Débute une boucle traversant les objets `ContentType`. Les détails d'objets seront utilisés pour construire les liens vers les zones de contenu du service choisi.
- #117** L'affichage de la liste des contenus est géré dans ce bloc de l'instruction `switch`.
- #122** Cet appel de fonction statique récupère un objet représentant le service sélectionné.
- #134** L'appel de fonction statique trouve les types de contenu gérés par le système.

- #151 À la ligne 151, la liste des contenus est ajoutée à une variable de niveau page, de façon que `company-00.inc` puisse construire l'affichage des données des objets.
- #153 Débute un bloc de code qui affiche un message s'il n'existe pas de contenu correspondant au service et au type de contenu sélectionnés.
- #183 Bloc de code exécuté lorsqu'un utilisateur affiche les détails d'un contenu.
- #186 Les lignes 186 à 189 obtiennent les détails pour l'affichage, à partir des objets.
- #198 Si la variable `$content_id` contient une valeur, les détails du contenu sont lus en utilisant la fonction de recherche à la ligne 199.
- #200 Les lignes 200 à 206 affichent les détails du contenu, extraits de l'objet à l'aide des fonctions de consultation.
- #207 Les noms des documents téléchargeables associés au contenu sélectionné sont lus.
- #209 Débute la boucle qui construit les liens vers les fichiers téléchargeables, à base des détails obtenus des objets `ContentDownload`.

Écriture de `company-00.inc` (code d'affichage principal)

Ici, les objets définis dans `CompanyHome-00.php` (Listing 16.16) sont sollicités pour construire l'affichage HTML. Hormis l'emploi de ces objets, il n'y a guère de différence avec la version procédurale de ce fichier, `company.inc`.

Listing 16.16 : Création de l'affichage HTML

```
<?php
/* Fichier : company-00.inc
 * Description : Contient le code d'une page Web
 * affichant les informations d'entreprise et de services.
 */
include_once("functions_main.inc");
?>
<html>
<head><title><?php echo $page['title']?></title></head>
```

```

<body style="margin: 0">
<h3 align="center"><?php echo $page['top'] ?></h3>
<div style="font-size: 70%; font-weight: bold">
    <?php echo $trail ?></div>
<hr size="10" noshade>
<table border="0" cellpadding="5" cellspacing="0">
<?php
#####
## Navigation gauche ##
#####
?>
<tr>
    <td width="20%" valign="top" >
        <p style="font-size: 110%; font-weight: bold">
            <?php echo $page['left_nav_header']?></p>
            <table border="0">
<?php
foreach($page["left_nav_links"] as $link => $label)
{
    echo "<tr><td >"
        . "<a href=\"$link\">$label<p><p></td></tr>\n";
}
if (sizeof($page["left_nav_links"]) == 0)
    echo "<i>aucun élément</i>";
?>
        </table>
    </td>

    <!-- Colonne entre les deux formulaires -->
    <td style="background-color: gray"></td>
<?php
#####
## Contenu principal ##
#####
?>
    <td width="80%" valign="top">
        <form method="POST" action="Admin-00.php"
            enctype="multipart/form-data">
<?php

if ($page["browse_level"] == "details")
{
    include("fields_content.inc");
    include("content_form.inc");
}
else if (sizeof($page["data_rows"]) > 0)
{
    echo "<table cellspacing='3' cellpadding='3'
        width='100%' bgcolor='gray'>

```

```

        <tr bgcolor='lightgray'>\n";
foreach ($page["col_headers"] as $key => $display)
{
    echo "<th >$display</th>\n";
}
echo "<th nowrap>&nbsp;  </th>\n";
echo "</tr>\n";
foreach ($page["data_rows"] as $content_item)
{
    echo "<tr bgcolor=white>\n";
    echo "<td nowrap>" . $content_item->getName() . "</th>\n";
    echo "<td nowrap>" . $content_item->getContentDate()
        . "</th>\n";
    echo "<td nowrap>" . $content_item->getCreationDate()
        . "</th>\n";
    echo "<td nowrap>" . $content_item->getCreatedBy()
        . "</th>\n";
    echo "<td nowrap>" . $content_item->getLastUpdDate()
        . "</th>\n";
    echo "<td nowrap>" . $content_item->getLastUpdBy()
        . "</th>\n";
    #68

    echo "<th nowrap>[";
    if ($admin)
    {
        echo "<a href=\"Admin-00.php?action=delete\"
            . "&dept_id=$dept_id&type_id=$type_id&content_id="
            . $content_item->getId() . "\">supprimer</a>\n";
    }
    echo "<a href=\"CompanyHome-00.php?\"
        . "&dept_id=$dept_id&type_id=$type_id&content_id="
        . $content_item->getId() . "&browse_level=details&edit=false\">"
        . "voir</a>\n";
    if ($admin)
    {
        echo "<a href=\"CompanyHome-00.php?\"
            . "&dept_id=$dept_id&type_id=$type_id&content_id="
            . $content_item->getId() . "&browse_level=details&edit=true\">"
            . "modifier</a>\n";
    }
    echo "]</th></tr>\n";
}
echo "</table>\n";
}

echo $page["body_text"];
?>

</form>
</td>

```

```

</tr>
</table>
<hr size="10" noshade>
<div style="text-align: center; font-size: 75%">
<?php echo $page['bottom']?>
</div>
</body>
</html>

```

Voici une description des lignes numérotées dans `CompanyHome-00.php` (Listing 16.16) :

#68 Les lignes 68 à 79 montrent l'emploi des objets pour obtenir les détails de contenu. Pour chaque colonne affichée dans le tableau HTML, une fonction différente est utilisée pour obtenir la valeur. Gardez à l'esprit que les données dans cet objet correspondant aux données stockées dans la base de données. Cela dit, ce code d'affichage ne sait rien de ces données, comment elles sont stockées, comment elles sont lues. Tant que les objets peuvent être sollicités, l'affichage peut être construit.

Écriture de `fields_content.inc` et de `content_form.inc`

Les fichiers `fields_content.inc` et `content_form.inc` du code procédural peuvent être ré-utilisés pour l'approche orientée objet.

Écriture de `Admin-00.php` (code de manipulation des données)

Le code de manipulation des données est plus simple que dans le code procédural, car le code SQL n'est pas exécuté dans le code PHP d'administration. Le PHP administratif, `Admin-00.php`, opère sur des objets.

Les fonctions `save` et `delete` des objets sont utilisées pour modifier les données sous-jacentes. Cela diffère de ce que l'on a dans le fichier procédural `Admin.php`, où le SQL pour manipuler les données est défini dans le PHP lui-même. Le code orienté objet supprime la nécessité du code SQL dans le fichier PHP d'administration. Dans le Listing 16.17, le code PHP utilise simplement des appels de méthode pour manipuler les données sous-jacentes.

Listing 16.17 : Actualisation de la base de données et enregistrement des fichiers téléchargés vers le serveur.

```

<?php
/* Fichier: Admin-00.php
 * Description: Réalise les tâches de manipulation de données,
 *             comme la création, la modification ou la suppression
 *             de contenu
 */
session_start();
include_once("functions_main.inc");
include_once("ContentItem.class");
include_once("ContentDownload.class");

foreach ($_POST as $name => $value)
    $$name = $value;
foreach ($_GET as $name => $value)
    $$name = $value;
if (!isset($action))
    header("Location: CompanyHome-00.php");

if (!isset($create_date))
    $create_date = date("Y-m-d", time());
else
    $create_date = strtotime($create_date);
if (!isset($content_date))
    $content_date = date("Y-m-d", time());
else
    $content_date = strtotime($content_date);

$content_date = date("Y-m-d", $content_date);
$last_upd_date = date("Y-m-d", time());

if (!isset($created_by))
    $created_by = $_SESSION["user_name"];

$last_upd_by = $_SESSION["user_name"];

switch ($action)
{
    case "delete":
        $content_item = ContentItem::findById($content_id);
        if (isset($content_item))
            try
            {
                $content_item->delete();
            }
            catch(Exception $e)

```



```

    {
        echo $e->getMessage();
        exit();
    }

break;

case "Enregistrer les changements":
    $message_2 = "";
    if ($content_date <= 0)
        $message_2 = "<li>Date de contenu invalide";
    if ($title == "")
        $message_2 .= "<li>Le titre ne peut être vierge";
    if ($message_2)
        $message_2 = "Merci de corriger ces erreurs : $message_2";
    if ($message_2 != "")
    {
        include("CompanyHome-00.php");
        exit();
    }
    if ($content_id) #65
    {
        try
        {
            $content_item = ContentItem::findById($content_id);
        }
        catch(Exception $e)
        {
            echo $e->getMessage();
            exit();
        }
        $content_item->setName($title);
        $content_item->setDescription($description);
        $content_item->setLastUpdBy($last_upd_by);
        $content_id = $content_item->save();
    }
    else #81
    {
        $content_item = new ContentItem(NULL,
            $title, $description, $content_date,
            $create_date, $created_by,
            $last_upd_date, $last_upd_by);
        $content_id = $content_item->save($dept_id, $type_id);
    }
    foreach ($_FILES as $file) #89
    {
        $file_name = $file["name"];
        if ($file["size"] <= 0)

```

```

        continue;
        $download = new ContentDownload($content_id,
                                         $file_name);

        $file_id = $download->save();
        $dest_dir = "files".DIRECTORY_SEPARATOR.$file_id;
        $dest_file = $dest_dir.DIRECTORY_SEPARATOR.$file_name;
        if(!file_exists($dest_dir))
        {
            if(!mkdir($dest_dir, 0700, TRUE))
                die ("Archivage du document impossible dans $dest_dir");
        }
        if (!file_exists($dest_file))
        {
            if (!move_uploaded_file($file["tmp_name"],
                                    $dest_file))
                die ("Archivage du document impossible dans $dest_dir");
        }
    }

    break;

case "DeleteDownload":
    try
    {
        $download = ContentDownload::findById($download_id);
        $download->delete();
    }
    catch(Exception $e)
    {
        echo $e->getMessage();
        exit();
    }

    break;

case "Annuler":
    break;
}

$query_str = "browse_level=content"
            . "&dept_id=$dept_id&type_id=$type_id";
header("Location: CompanyHome-00.php?$query_str");
?>

```

Voici la description des lignes de code numérotées de Admin-00.php (Listing 16.17) :

- #8 Les deux types de données modifiables par les utilisateurs du CMS, représentés par les classes `ContentItem` et `ContentDownload`, sont utilisées dans `Admin-00.php`.
- #37 Le bloc de code débutant ligne 37 utilise la fonction statique de recherche de la classe `ContentItem` pour obtenir l'objet de contenu que l'utilisateur souhaite supprimer. La fonction `delete` de l'objet est appelée pour supprimer les données sous-jacentes dans la base de données. `Admin-00.php` n'a pas besoin de savoir comment cela est accompli car l'approche orientée objet permet d'encapsuler de tels détails dans le code de la classe.
- #65 Débute un bloc de code qui met à jour un contenu existant.
- #81 Débute un bloc de code qui crée un nouveau contenu.
- #89 Débute une boucle qui enregistre dans la base de données les fichiers téléchargés vers le serveur. De nouveau, il y a abstraction dans `Admin-00.php` des détails sur l'enregistrement des informations dans la base de données.
- #114 C'est ici que sont consultés les détails d'un fichier téléchargeable. Une fois trouvées les informations, elles sont supprimées via la fonction `delete` de l'objet `ContentDownload`.

Amélioration du système de gestion de contenu

Le CMS de ce chapitre est très générique. Il gère plusieurs types de contenu, mais l'interface utilisateur, les données, et le modèle objet ne diffèrent guère d'un type de contenu à l'autre. Le développement du modèle objet peut être poussé, de façon que, par exemple, un objet actualité `News` et un objet événement `Event` puissent posséder plus d'attributs pertinents par rapport à ces types de contenu. Par exemple, le type de contenu `Event` pourrait être implémenté de façon que les utilisateurs puissent s'inscrire à un ou plusieurs événements.

Un CMS bien conçu doit organiser le contenu de telle sorte que, si le volume de données croît, la navigation et la recherche de contenu ne deviennent pas fastidieuses. La pagination est utile pour simplifier la navigation dans de longues listes de contenu. Par ailleurs, une recherche par mot clé est confortable, elle permet à l'utilisateur de trouver rapidement plusieurs contenus distincts mais apparentés.



Huitième partie

Les dix commandements



"D'accord... bien... je pense que nous savions tous jusqu'à où serait mené Jerry par le développement de cette carte de site."

Dans cette partie...

Le Chapitre 17 propose des conseils pour le développement d'application, basés sur mes expériences. Ils vous serviront éventuellement de raccourcis dans votre voyage pour devenir un développeur Web.

Le Chapitre 18 présente une liste de sites Web où vous trouverez des bibliothèques de code et autres informations utiles sur les applications.

Chapitre 17

Dix conseils pour le développement d'application

Dans ce chapitre :

- Écrire du code facile à maintenir.
- Trouver de l'aide sur les listes de discussion.

Une application peut se composer d'un unique script de cinq lignes affichant Hello World! Dans ce cas, le développement de l'application est simple. Cependant, les applications sont en réalité plus complexes, nécessitant souvent de nombreux scripts et de nombreux programmeurs travaillant en coordination. Le développement d'une application complexe peut être très délicat. Ce chapitre présente quelques suggestions de pratiques de codage qui rendront le développement plus fluide, quelle que soit la complexité de l'application.

Première chose, planifiez

Calmez votre envie d'en découdre immédiatement, c'est-à-dire de commencer tout de suite à écrire du code. Un plan, sur papier, est essentiel. Le plan répondra aux questions suivantes :

- ✓ Quelles tâches l'application doit-elle réaliser ?
- ✓ Quelles méthodes de programmation utiliserez-vous pour implémenter chaque tâche ?

- ✓ De quelles façons les tâches vont-elles interagir les unes avec les autres ?
- ✓ Quelle est la facilité de maintenance de votre application ?
- ✓ Comment ajouterez-vous plus tard des fonctions à l'application ?
- ✓ De quelle manière l'application satisfait-elle les besoins de l'utilisateur ?

Rien n'est plus pénible que de se rendre compte, à mi-chemin du codage, qu'un important facteur a été oublié et qu'il faut tout reprendre.

Soyez homogène

De nombreuses décisions sont prises au cours du codage d'une application. Prenez chaque décision une bonne fois pour toutes, soyez homogène dans le reste de l'application. Par exemple, prenez au début une décision sur les conventions de nommage, puis utilisez ces conventions le reste du temps. Si vous commencez à écrire les variables tout en lettres minuscules et les noms de fonctions avec majuscules initiales sur chaque mot, dès le début de l'application ou d'une classe, utilisez ces mêmes conventions dans toute l'application ou dans toutes les classes. L'homogénéité facilite la compréhension et la maintenance du code.

Testez le code de manière incrémentale

Un programme se compose de nombreux blocs de code. La création et le test d'une application se déroulent mieux lorsque chaque petit bloc de code est testé en tant qu'entité. Testez chaque morceau simple de code jusqu'à ce qu'il se comporte comme attendu. Ajoutez des fonctions au code une à la fois, en testant chaque nouveau morceau de code au fur et à mesure. Par exemple, si une tâche de programme nécessite une boucle dans une boucle, construisez et testez une première boucle. Lorsqu'elle fonctionne, ajoutez l'autre boucle.

Pensez à ceux qui prendront la relève

Gardez votre code aussi simple et facile à lire que possible. Un jour, quelqu'un essaiera de comprendre votre code, pour en opérer la maintenance ou le modifier. Tout code qui est difficile à comprendre est un problème, quelle que soit son intelligence apparente ou sa compacité.

Utilisez des constantes

Les constantes simplifient grandement la maintenance. Lorsque vous utilisez une même valeur plusieurs fois dans une application, comme un taux de TVA ou un nom d'entreprise, stockez-la dans une constante et exploitez cette constante dans l'application. Cela sert deux rôles. Tout d'abord, le nom de constante est beaucoup plus descriptif qu'un occulte nombre. En d'autres termes, `Taux_TVA` est plus descriptif que `19,6`. Ensuite, lorsque la valeur change, vous n'avez à la modifier qu'une seule fois, au début du programme, au lieu de devoir la remplacer à tous les endroits de l'application où elle apparaît.

Écrivez du code réutilisable

Soyez à l'affût des possibilités d'écriture de code réutilisable. À chaque fois que vous constatez que vous rédigez le même code dans différentes portions de l'application, écrivez une fonction. Elle vous fera gagner du temps et facilitera la lecture du programme. Avec le temps, vous bâtirez une bibliothèque de fonctions exploitables dans différentes applications.

Bien sûr, si vous rédigez du code orienté objet, vous écrivez forcément du code réutilisable.

Séparez la mise en page et la fonction

La maintenance est plus facile lorsque le code de la mise en page Web est séparé du code de la logique du programme. Vous pouvez écrire le code qui affiche la page dans un fichier séparé, générant du code HTML. Le script PHP principal qui contient la logique du programme inclura le fichier contenant le code d'affichage, au

moment où la page doit être affichée. Une fois opérée cette séparation, il est beaucoup plus facile de changer l'aspect d'une page Web. Vous n'avez en effet à changer que le code d'affichage, dans le fichier de mise en page ; vous n'avez pas besoin de changer le code de la logique du programme.

Ne réinventez pas la roue

Avant de passer des heures à écrire un morceau de code refusant de réaliser ce que vous en attendez, cherchez sur Internet si ce code existe. Des milliers de développeurs PHP partagent leur code. Il existe des sites qui proposent du code PHP pour diverses fonctions, peut-être exactement celle que vous avez à l'esprit.

Si votre application est courante, il est même possible que vous en trouviez une toute faite. De nombreuses applications PHP, gratuites ou payantes, sont disponibles sur Internet. On trouve des CMS, des paniers d'achat, des forums, etc. Recherchez sur Google une solution complète avant de vous embarquer dans la construction de la vôtre.

Utilisez fréquemment les listes de discussion, mais intelligemment

Bien des utilisateurs PHP et MySQL sont prêts à vous aider si vous êtes bloqués. Tirez avantage de cette ressource. Cependant, assurez-vous que votre question est réfléchie. "J'ai essayé d'accéder à ma base de données et je n'y suis pas arrivé. Que dois-je faire?" n'est pas un message auquel on répondra. Si quelqu'un répond, ce sera pour vous pointer l'inutilité de ce message qui ne fournit aucun détail. Dans tout message demandant de l'aide, insérez les informations suivantes :

- ✓ Indiquez les logiciel et versions utilisés. Par exemple, Windows XP, PHP 5.0.2, MySQL 4.1.7, Apache 1.3.27.
- ✓ Expliquez ce que vous avez opéré, en détail. Si le problème est que le code produit un message d'erreur, ou que le code ne donne pas les résultats attendus, insérez le code. Si le code est énorme, insérez juste assez de code pour montrer le problème. Ou, éventuellement, construisez un bloc simplifié du code qui pose problème.

- ✓ Précisez ce qu'aurait dû être le résultat.
- ✓ Indiquez ce qui s'est réellement produit, à la différence du résultat attendu.

N'oubliez pas de vous documenter avant de poser une question. Lisez le manuel. Faites des recherches sur Google. Postez sur la liste uniquement lorsque les autres ressources ont échoué.

Voici les listes les plus utiles sur PHP et MySQL (en anglais).

- ✓ www.php.net/mailling-lists.php
- ✓ <http://lists.mysql.com>

Documentez tout

Pendant l'écriture d'une application, les détails semblent gravés dans votre esprit de façon indélébile. Cependant, six mois plus tard, lorsque vous désirerez opérer des changements à l'application, vous serez étonné de vous rendre compte que vous ne vous souvenez plus de ces détails. Ne comptez pas sur votre mémoire. Notez toutes vos décisions. Insérez des commentaires sans compter dans vos scripts.

Chapitre 18

Dix sources de code PHP

Dans ce chapitre :

Trouver des bibliothèques de code.

Autres ressources utiles aux programmeurs PHP.

Un avantage de PHP est que ses développeurs ont la volonté de partager leur code. Il existe plusieurs bibliothèques de code en ligne. Vous pouvez y glaner du code déposé par des programmeurs PHP expérimentés. Quel que soit votre besoin, vous trouverez souvent du code exploitable tel quel ou avec un peu de modification. Inutile de réinventer la roue pour chaque tâche de programmation.

SourceForge.net

www.sourceforge.net

SourceForge.net est le plus grand référentiel de code et d'applications open source sur Internet. Vous y trouverez du logiciel pour tous les rôles. De nombreux projets sur SourceForge sont de grandes applications logicielles, que vous pouvez télécharger et utiliser. Par exemple, phpmyadmin, populaire application exploitée pour gérer les bases de données MySQL, est disponible sur SourceForge.

Vous n'avez pas besoin de vous identifier sur SourceForge pour télécharger du logiciel. Pour trouver du logiciel écrit en PHP, suivez ces étapes :

1. Cliquez sur l'onglet **Software Map**, en haut.
2. Cliquez sur **Programming Language** (langage de programmation) dans la colonne de droite.

3. Cliquez sur PHP dans la liste des langages de programmation classés par ordre alphabétique, sur la gauche.

À l'heure de la traduction de cet ouvrage, SourceForge.net présentait 12 175 projets en PHP.

WeberDev

<http://weberdev.com>

WeberDev est une des ressources les plus complètes pour programmeurs PHP. Démarré en 1998 sous forme d'une page pour présenter des exemples, le site a énormément grossi, il comprend des portions de code et des programmes complexes. Voici les grandes catégories d'exemples présentés : PHP, MySQL, JavaScript, Bases de données. Vous pouvez télécharger le code sans vous identifier sur le site. Les commentaires des utilisateurs sur les exemples sont affichés avec le code. WeberDev présente également des articles, manuels et forums de discussion, propose un concours de codage, des modèles, des blogs PHP, et d'autres ressources.

Vous pouvez vous inscrire (vous recevrez un mot de passe). Une fois inscrit, vous avez la possibilité d'ajouter du code, de poster dans les forums de discussion, d'ajouter des commentaires aux exemples de code, de participer au concours de code (doté de prix), de recevoir une lettre d'information, des notifications lorsque de nouveaux exemples sont ajoutés au site, et de bénéficier d'autres avantages. Pour vous inscrire, vous devrez fournir vos nom et adresse électronique. Vous pouvez demander à ne pas recevoir d'e-mail du site.

PHP Classes

www.phpclasses.org

PHP Classes est un dépôt recelant des centaines de classes écrites en PHP. Pour trouver des classes, vous pouvez effectuer une recherche, ou parcourir par catégorie ou par auteur. Les notes des utilisateurs sont affichées avec les informations sur les classes. Il n'est pas obligatoire de s'identifier pour télécharger du code. Ce site Web propose également des comptes rendus de livres PHP.

Codewalkers

<http://codewalkers.com>

Le site Codewalkers s'intitule lui-même "Une ressource pour les développeurs PHP et SQL". Vous trouverez du code dans les catégories suivantes : Content Management (Gestion de contenu), Database Related (Base de données), Date and Time (Date et heure), Discussion Boards (Forums de discussion), E-Mail, File Manipulation (Manipulation de fichiers), Link Farm (Liens), Look and Feel (Aspect, interface), Miscellaneous (Divers), Searching (Recherche), Site Navigation (Navigation), Statistics and Counters (Statistiques et compteurs) et User Management (Gestion d'utilisateurs). Chaque code est assorti d'une note donnée par les visiteurs du site. La galerie de code contient du code procédural et orienté objet. Le site offre également des didacticiels, des comptes rendus de livres et de logiciels, un concours de code, et des forums de discussions.

Le site propose l'inscription comme membre. Il est inutile de s'inscrire pour télécharger du code mais il le faut pour pouvoir proposer du code. Vous pouvez poster sur les forums sans vous identifier au préalable mais vous devrez vous inscrire pour bénéficier de certaines fonctions évoluées des forums. L'inscription nécessite un ID utilisateur et un mot de passe de votre choix. Une adresse électronique est également requise. D'autres informations sont demandées mais non obligatoires.

PHP Builder

www.phpbuilder.com

PHP Builder est un site Web contenant diverses ressources pour codeurs PHP. Il fournit une bibliothèque de portions de code, des scripts, et des fonctions pour un large éventail d'usages. Vous pouvez rechercher du code dans des catégories telles que bases de données, calendriers, paniers d'achat, jeux, graphismes, etc. De plus, vous trouverez des actualités et une liste d'articles utiles.

HotScripts.com

www.hotscripts.com

HotScripts est un répertoire de ressources de programmation. Ce site propose informations et scripts pour PHP, CGI, Perl, JavaScript et ASP. Presque 10 000 scripts sont actuellement proposés.

Le descriptif de chaque programme donne des informations sur le logiciel, plus un lien vers sa page Web. Chaque descriptif inclut également des comptes rendus et des appréciations des utilisateurs.

Zend

<http://zend.com>

Le site Web Zend inclut une galerie de code. Zend est l'entreprise qui développe le moteur PHP. Il n'est pas obligatoire de s'identifier pour télécharger du code. Parmi les catégories de code PHP proposées, on trouve les maths, les bases de données, les algorithmes. Les utilisateurs peuvent donner des notes aux logiciels.

De nombreuses autres ressources sont disponibles, notamment certains excellents articles. Zend, en partenariat avec WeberDev (présenté plus haut dans ce chapitre), sponsorise un concours de code PHP.

PHP Freaks

www.phpfreaks.com

PHP Freaks offre environ 600 scripts, dans plusieurs catégories. Il fournit un accès rapide aux dix scripts les plus populaires, et aux scripts les plus récents. Le code est noté. Il n'est pas obligatoire de s'identifier pour télécharger les scripts. Cependant, pour ajouter un script, vous devrez fournir un nom et une adresse électronique valides.

Le site fournit également articles et didacticiels, actualités et forums.

PX : PHP Code Exchange

<http://px.sklar.com>

PX permet aux utilisateurs de poster et télécharger du code PHP. C'est un site Web simple, qui donne juste accès au code. La page d'accueil se compose d'une liste de catégories à parcourir, et d'une case pour effectuer une recherche. Il n'est pas obligatoire d'être identifié pour télécharger du code mais vous devrez le faire pour poster ou noter le code.

Free PHP et MySQL Hosting Directory

www.oinko.net/freephp

Ce site propose une liste d'hébergements Web gratuits qui proposent du PHP. Les hébergeurs sont notés de une à cinq étoiles.

Neuvième partie

Annexes



Dans cette partie...

Dans les deux premières annexes, vous allez trouver des instructions pour installer MySQL, PHP et Apache. L'Annexe C, quant à elle, est consacrée à EasyPHP et à PHPMyAdmin, utilitaire de manipulation des données contenues dans une base MySQL.

Annexe A

Installation de MySQL

Bien que MySQL puisse tourner sur de nombreuses plates-formes, je vais décrire ici son installation sous UNIX/Linux/Mac et Windows, ce qui concerne la majorité des sites de l'Internet. Commencez par lire toutes ces instructions avant de démarrer une installation.

MySQL peut être installé à partir de paquetages précompilés appelés RPM (sous Linux) ou fichiers binaires. Ces paquetages sont disponibles pour la plupart des systèmes d'exploitation : Linux, Windows, Mac, FreeBSD, de nombreuses variantes d'UNIX, etc. S'il en existe un pour le système d'exploitation que vous utilisez, prenez-le. Il sera ainsi plus facile d'installer MySQL. Mais si vous n'en trouvez aucun (ou si vous avez des besoins très spécifiques), vous pouvez néanmoins procéder à une installation de MySQL à partir de son code source.

Si vous rencontrez des problèmes pour lancer le serveur MySQL après l'installation, regardez ce qu'indique le journal d'erreurs. Il doit se trouver dans le répertoire `data` et possède l'extension `.err`.

MySQL sous Windows

Dans la plupart des cas, lorsque vous téléchargez et installez MySQL, le serveur est lancé automatiquement. Si ce n'est pas le cas ou si vous avez besoin de l'arrêter et de le redémarrer pour toute autre raison, vous pouvez recourir à un utilitaire appelé `WinMySQLadmin` qui est installé en même temps que MySQL. J'en reparlerai plus loin dans la section "Lancement du serveur MySQL". Vous pouvez aussi l'utiliser pour commander le lancement automatique de MySQL au moment de l'amorçage de votre ordinateur.

Téléchargement et installation de MySQL

Pour installer MySQL sous Windows, suivez ces étapes :

1. **Affichez dans votre navigateur la page située à l'URL <http://www.mysql.fr>.**
2. **Cliquez sur le lien Serveur MySQL Community à droite.**
3. **Descendez vers le bas de la page jusqu'à atteindre la section Windows Downloads.**

A l'heure où nous mettons sous presse, c'est la version MySQL 5.0 qui est conseillée.

4. **Repérez le lien de téléchargement Pick a mirror de la version Windows désirée (Windows Essentials suffit à la plupart des besoins).**

Une page vous montre la liste des sites Web à partir desquels vous allez pouvoir procéder.

5. **Cliquez sur le lien d'un site miroir (FTP ou http) situé en France.**

Une boîte de dialogue standard s'ouvre.

6. **Choisissez l'option vous permettant de sauvegarder le fichier.**

Une boîte de saisie vous permet d'indiquer le répertoire dans lequel vous voulez sauvegarder le fichier.

7. **Sélectionnez le répertoire destinataire (par exemple, `c:\downloads`) et cliquez sur Enregistrer.**

Une fois le téléchargement terminé, vous verrez un fichier .MSI dans ce répertoire. Ce fichier contient tous les fichiers MySQL. Son nom est par exemple `mysql-5.0.37-win32.msi`.

8. **Vérifiez l'intégrité du fichier téléchargé.**

Reportez-vous à la section "Contrôle d'intégrité d'un fichier téléchargé", en fin d'annexe.

9. **Double-cliquez sur le nom ou l'icône du fichier d'installation .msi.**

L'écran d'accueil apparaît (Figure A.1). **Note :** si vous faites cette installation sous Windows NT, 2000 ou XP, assurez-vous que vous êtes loggé sous un compte ayant les droits d'accès d'administrateur.

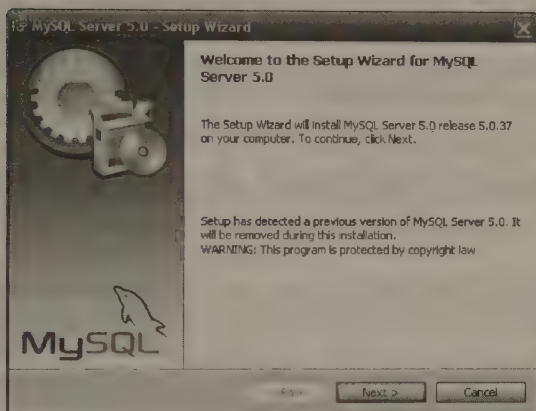


Figure A.1 :
Écran
d'accueil du
programme
d'installation
de MySQL.

10. Cliquez sur Next.

Vous devez choisir le type d'installation.

11. Choisissez Typical et cliquez sur Next.

Un écran indique les paramètres d'installation proposés.

12. Cliquez sur Install.

L'installation se déroule puis vous passez à une proposition d'inscription.

13. Choisissez Skip Signup sauf si vous désirez vous inscrire puis cliquez sur Next.

La page Wizard Completed apparaît (Figure A.2).

14. Si c'est la première installation de cette version du serveur, choisissez Configure the MySQL Server Now.

Si vous procédez à une mise à jour de MySQL Server (par exemple de MySQL 5.0.18 vers 5.0.37), il n'est peut-être pas nécessaire de le reconfigurer. L'assistant reprendra la configuration existante. Si la mise à jour saute de nombreuses

sous-versions (par exemple, de MySQL 5.0 vers MySQL 5.1), vous devez utiliser l'assistant de configuration.

15. Cliquez sur Finish.

Si vous avez choisi de configurer avec l'assistant, celui-ci démarre. Le déroulement de cette configuration fait l'objet de la section suivante. Si vous avez refusé cette étape, vous avez terminé d'installer MySQL.

Exécution de l'assistant de configuration MySQL

Une fois installé, MySQL doit encore être configuré. Vous devez choisir un mot de passe pour le compte d'accès principal à MySQL appelé *root* et créé d'office, mais sans mot de passe. Vous devez aussi démarrer le serveur et le configurer pour qu'il soit lancé automatiquement au démarrage de l'ordinateur.

MySQL est doté d'un assistant qui est lancé en fin d'installation des fichiers (il faut accepter l'option *Configure the MySQL Server Now*). Voyons comment l'utiliser en dehors de la procédure d'installation depuis le sous-menu de MySQL.

1. Choisissez Démarrer/Programmes/MySQL/MySQL Server 5.0L/MySQL Server Instance Config Wizard.

L'assistant de configuration apparaît (Figure A.3).

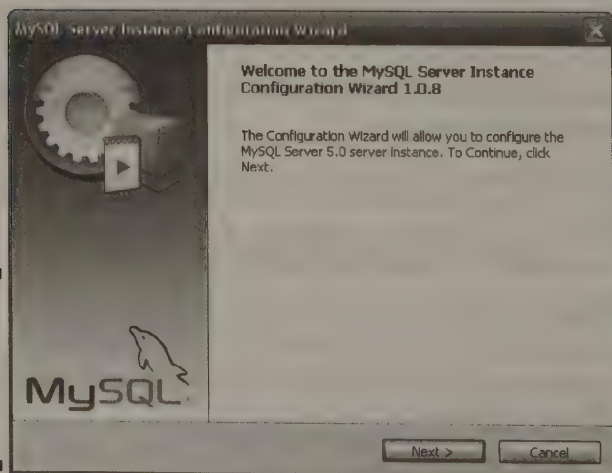


Figure A.3 :
Ecran
d'accueil de
l'assistant de
configuration
MySQL.

2. Si plus d'une version de MySQL est détectée, vous devez choisir laquelle vous voulez configurer. Cliquez ensuite sur Next.

La page MySQL Server Configuration Types apparaît.

3. Choisissez Standard Configuration puis cliquez sur Next.

La page Windows Options apparaît.

4. Choisissez Install as a Windows Service.

Cette approche est impossible sous Windows 98/Me. Vous devez choisir **Include bin Directory in Windows PATH** puis passer à l'étape 7.

5. Dans la zone de saisie de nom de service Service Name, saisissez `mysql50`.
6. Sélectionnez l'option de lancement automatique Launch the MySQL Server Automatically.
7. Cliquez sur Next.

La page des options de sécurité apparaît (Figure A.4).

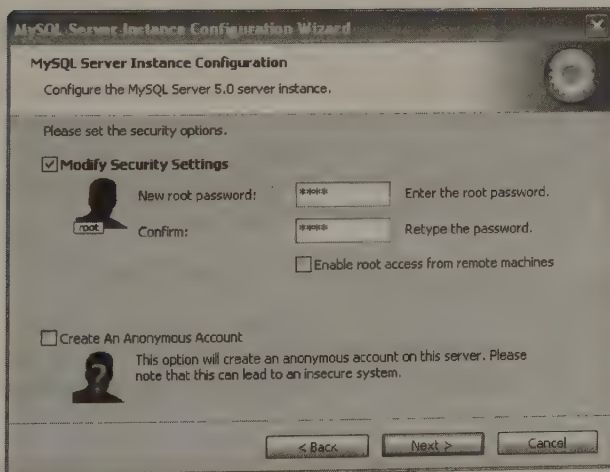


Figure A.4 :
Page de
configuration
des options
de sécurité.

8. Choisissez l'option Modify Security Settings.



9. Dans la zone de saisie New Root Password, saisissez votre mot de passe pour le compte racine *root* et répétez-le dans la zone Confirm box.

Vous définissez ici le mot de passe principal d'accès au serveur MySQL. Il est requis pour accéder à la base de données MySQL. Souvenez-vous bien de ce mot de passe.

10. Si vous préparez un environnement de développement d'applications pour votre usage personnel, vous pouvez sans danger activer l'option de création d'un compte anonyme Create an Anonymous Account.

Si la base de données risque de pouvoir être atteinte depuis le réseau Internet, il ne faut absolument pas créer un tel compte anonyme. Le risque d'intrusion est trop grand.

11. Cliquez sur Next.

Vous passez à l'étape d'exécution.

12. Cliquez sur le bouton Execute.

Un message confirme la fin de la configuration.

Lancement et arrêt du serveur MySQL

Une fois le serveur configuré pour démarrer automatiquement lorsque vous allumez votre machine, il arrivera de temps à autre que vous ayez à l'arrêter puis à le redémarrer manuellement. C'est par exemple le cas lors d'une mise à jour de MySQL. Il faut arrêter le serveur avant de lancer la mise à jour.

Windows NT/2000/XP

Voici comment arrêter et démarrer le serveur MySQL sous l'une de ces versions de Windows (avec les menus normaux de ces versions) :

1. Choisissez Démarrer/Panneau de configuration/Outils d'administration/Services.

La liste des services installés apparaît.

2. Descendez jusqu'à la ligne du service MySQL.

Des liens apparaissent dans le volet gauche.

3. Cliquez sur Arrêter ou sur Démarrer.

Si vous ne trouvez pas le service MySQL dans la liste, vous devez utiliser l'assistant de configuration pour implanter le service.

Windows 98/Me

Sous Windows 98/Me, vous ne pouvez pas implanter MySQL en tant que service. Voici comment le démarrer manuellement :

1. Ouvrez une fenêtre de commande (fenêtre MS-DOS).

Ouvrez le sous-menu Démarrer/Programmes/Accessoires.

2. Passez dans le sous-dossier bin de celui dans lequel MySQL a été installé.

Par exemple, tapez la commande `cd c:\Program Files\MySQL\MySQL Server 5.0\bin`.

3. Tapez la commande `mysqld`.

Si la commande échoue, essayez `mysqld-nt`. Le choix entre les deux lanceurs dépend de la version de MySQL.

Si le serveur a démarré, il n'affiche aucun message pour le confirmer. Vous devez absolument laisser ouverte la fenêtre (même repliée) pendant l'activité du serveur. Si vous la fermez, le serveur est arrêté, même si cela prend parfois quelques secondes. Si le démarrage échoue, un message d'erreur explique pourquoi.

4. Testez l'activité du serveur.

Ouvrez une autre fenêtre de commande (voir l'étape 1). Passez dans le répertoire (dossier) de l'étape 2 et saisissez la commande `mysql`. Si le serveur ne fonctionne pas, un message d'erreur le confirmera.

Arrêt manuel

Parfois, vous aurez du mal à arrêter le serveur. Dans ce cas, vous pouvez forcer son arrêt manuellement ainsi :

1. Ouvrez une fenêtre de commande (fenêtre MS-DOS).

Ouvrez le sous-menu Démarrer/Programmes/Accessoires.

2. Passez dans le sous-dossier bin de celui dans lequel MySQL a été installé.

Par exemple, tapez la commande `cd c:\Program Files\MySQL\MySQL Server 5.0\bin`.

3. Tapez la commande suivante :

```
mysqladmin -u root -p shutdown
```

Le nom du compte est `root`. L'option `-p` demande de saisir un mot de passe. Si vous n'avez pas défini de mot de passe pour ce compte (c'est déconseillé), omettez cette option.

Sous Linux/UNIX

Dans la plupart des distributions Linux, le SGBD MySQL est installé d'office ou bien son installation est proposée dans les options principales. Les distributions Fedora, SuSE, Ubuntu, et d'autres, comportent un utilitaire pour télécharger puis installer la plus récente version de MySQL sans effort de votre part. Cette approche assistée est généralement plus efficace que le téléchargement manuel suivi de l'installation comme nous l'avons vu sous Windows. Si vous n'avez pas accès à ce confort, cherchez toujours sur le site Web de MySQL s'il existe un installateur pour votre distribution.

Vous pouvez installer MySQL sous Linux à partir de fichiers au format RPM (disponibles sur le site de MySQL). Ces fichiers archives RPM garantissent une installation facile sous Linux. Nous en expliquons la procédure dans la section suivante. Sous UNIX, vous pouvez installer le fichier binaire correspondant à votre variante d'UNIX, et donc adapté aux procédures d'installation de ce système. A l'heure où nous mettons sous presse, les binaires MySQL pour UNIX sont proposés pour Solaris, HP-UX, AIX, SCO, SGI Irix, OpenBSD, FreeBSD et quelques autres UNIX.

Si vous ne trouvez ni fichiers RPM, ni fichiers binaires, vous pouvez toujours installer les fichiers source de MySQL. Nous expliquons la procédure dans une section ultérieure.

Installation avec les fichiers RPM (Linux uniquement)

RPM signifie *Red Hat Package Manager* ("gestionnaire de paquets pour Red Hat"), mais ce format prévu à l'origine pour la distribution Red Hat a été adapté à d'autres distributions Linux. C'est de cette façon que l'installation sera la plus facile à effectuer. Vous allez trouver ci-après les instructions qui vont vous permettre de télécharger le fichier RPM. Toutefois, peut-être se trouve-t-il déjà sur les CD de votre distribution Linux ? Vous risquez cependant de ne disposer que d'une version ancienne. Le téléchargement d'une version RPM récente est donc toujours conseillé.

- 1. Affichez dans votre navigateur la page située à l'URL `http://www.mysql.fr`.**
- 2. Cliquez sur le lien *Serveur MySQL Community* à droite.**
- 3. Descendez vers le bas de la page jusqu'à atteindre la section *Linux Downloads*.**

A l'heure où nous mettons sous presse, c'est la version MySQL 5.0 qui est conseillée.

- 4. Repérez le lien de téléchargement *Pick a mirror de la version Linux désirée du fichier Server approprié (x86 ou AMD64)*.**

Une page vous montre la liste des sites Web à partir desquels vous allez pouvoir procéder. Notez que vous devez télécharger deux fichiers : le RPM du serveur et celui du client MySQL.

- 5. Cliquez sur le lien d'un site situé en France.**

Une boîte de dialogue standard s'ouvre.

- 6. Choisissez l'option vous permettant de sauvegarder le fichier.**

Une boîte de saisie vous permet d'indiquer le répertoire dans lequel vous voulez sauvegarder le fichier.

- 7. Sélectionnez le répertoire choisi pour ce téléchargement. (Par exemple, `/usr/src/mysql`.) Cliquez sur *Save*.**

8. Répétez les étapes 4 à 7 pour télécharger le fichier RPM Client (x86 ou AMD64) dans le même répertoire.
9. Placez-vous dans le répertoire où ont été téléchargés les fichiers.

Tapez, par exemple :

```
cd /usr/src/mysql
```

Vous verrez deux fichiers dans le répertoire :

- MySQL suivi d'un numéro de version et de l'extension .i386.rpm. Par exemple : MySQL 5.0.21-0.i386.rpm.
- Un autre de même nom mais avec client dans son intitulé. Par exemple : MySQL client-5.0.21-0.i386.rpm.

10. Vérifiez l'intégrité du fichier téléchargé.

Reportez-vous à la section "Contrôle d'intégrité d'un fichier téléchargé", en fin d'annexe.

11. Installez le RPM au moyen de la commande suivante :

```
rpm -i liste_des_paquetages
```

Ainsi, en suivant l'exemple donné à l'étape 9, la commande serait :

```
rpm -i MySQL-server-5.0.21-0.i386.rpm MySQL-client-5.0.21-0.i386.rpm
```

Cette commande installe les paquetages MySQL. Elle définit le compte MySQL et le nom de groupe dont vous aurez besoin et crée le répertoire de données `/var/lib/mysql`. Elle lance également le serveur MySQL et crée les entrées appropriées dans `/etc/rc.d` afin que MySQL démarre automatiquement au moment de l'amorçage de votre ordinateur.

12. Pour vérifier que MySQL est bien en activité, tapez :

```
bin/mysqladmin --version
```

Vous devriez voir s'afficher le numéro de version de votre serveur MySQL.

Installation avec les fichiers source

Avant que vous ne décidiez d'installer MySQL à partir des fichiers source, assurez-vous qu'il n'existe pas de fichiers binaires convenant à votre système d'exploitation. Les fichiers binaires sont précompilés sous forme de paquetages tout prêts à être installés. C'est un moyen très pratique et fiable d'installer MySQL.

Pour installer MySQL à partir des fichiers source, vous devez les télécharger, les compiler et installer les programmes qui en résultent. Rassurez-vous : cette mise en œuvre est plus simple qu'il n'y paraît. Lisez la totalité des instructions qui suivent avant de commencer votre installation.

Pour installer MySQL à partir du code source, parcourez les étapes suivantes :

- 1. Procédez comme décrit pour les fichiers binaires RPM dans les étapes 1 à 6 de la section précédente, sauf que vous devez aller trouver la section des fichiers source tout en bas de liste (Source downloads).**

Le fichier à télécharger s'intitule Compressed GNU TAR archive.

- 2. Sélectionnez le répertoire de destination.**

La convention stipule `/usr/local`. Il est souhaitable d'utiliser cette adresse lorsque c'est possible.

- 3. Une fois le téléchargement terminé, placez-vous dans le répertoire où ont été téléchargés les fichiers. Tapez, par exemple : `cd -/usr/local`. Vous y voyez un fichier appelé `mysql- suivi du numéro de version et de .tar.gz` (par exemple : `mysql-5.0.37.tar.gz`). Ce fichier est appelé *tarball*.**

- 4. Vérifiez l'intégrité du fichier téléchargé.**

Reportez-vous à la section "Contrôle d'intégrité d'un fichier téléchargé", en fin d'annexe.

- 5. Créez un identificateur d'utilisateur et de groupe pour MySQL à l'aide des commandes :**

```
groupadd mysql
useradd -g mysql mysql
```

La syntaxe de ces commandes peut différer légèrement selon les versions d'UNIX. Elles peuvent aussi être appelées `adduser` et `addgroup`.

Note : assurez-vous que vous êtes connecté sous un compte ayant les droits d'accès nécessaires pour ajouter des utilisateurs et des groupes.

6. Décompressez le tarball en tapant :

```
gunzip -c nom_du_fichier | tar -xvf -
```

Par exemple :

```
gunzip -c mysql-5.0.27.tar.gz | tar -xvf -
```

Vous devez, pour cela, être loggé sous un compte autorisé à créer des fichiers dans `/usr/local`.

7. Placez-vous dans le nouveau répertoire.

Par exemple, tapez : `cd mysql-5.0.37`.

8. Tapez ce qui suit :

```
./configure --prefix=/usr/local/mysql
```

Plusieurs lignes vont s'afficher qui vous signalent que la configuration s'effectue. Cela peut prendre un certain temps.

9. Tapez `make`.

Plusieurs lignes vont s'afficher. La dernière vous signale que le `make` est terminé. Cela peut prendre un certain temps.

10. Tapez `make install`.

Cette commande s'exécute rapidement.

Note : il peut être nécessaire de lancer cette commande en tant que `root`.

11. Tapez ce qui suit :

```
scripts/mysql_install_db.
```

Cette commande lance un script qui initialise vos bases de données MySQL.

- 12. Vérifiez que le possesseur et le groupe des membres de vos répertoires MySQL sont corrects. Pour définir le possesseur, exécutez les commandes suivantes :**

```
chown -R root /usr/local/mysql
chown -R mysql /usr/local/mysql/data
chgrp -R mysql /usr/local/mysql
```

Ces commandes définissent `root` comme le possesseur de tous les répertoires sauf `data` dont le possesseur est défini comme `mysql`. Tous les répertoires de MySQL appartiennent au groupe `mysql`.

- 13. Modifiez votre système d'exploitation de façon que MySQL soit automatiquement lancé lors de l'amorçage de votre machine, en copiant le fichier `mysql.server` à partir de `/usr/local/mysql/support-files` vers l'emplacement où se trouvent les fichiers de démarrage de votre système d'exploitation.**
- 14. Pour tester MySQL, vous pouvez lancer le serveur à la main sans qu'il soit nécessaire de redémarrer votre machine en tapant :**

```
bin/safe_mysqld --user=mysql &
```

- 15. Pour vous assurer que MySQL est bien en activité, tapez :**

```
bin/mysqladmin --version
```

Vous devriez voir s'afficher le numéro de version de votre serveur MySQL.

Sous MacOS

Vous pouvez télécharger MySQL sous la forme d'un paquetage binaire PKG pour Mac OS X 10.2 (Jaguar) et versions ultérieures. Avec Mac OS X 10.1 ou antérieur, ce paquetage sera inutilisable. Vous devrez charger une archive tarball et procéder à l'installation de MySQL à partir du code source (voir la section précédente).

- 1. Procédez comme décrit pour les fichiers binaires RPM dans les étapes 1 à 6 de la section précédente, sauf que vous devez aller trouver la section des fichiers pour MacOS : Mac OS X (package format) downloads.**

Le fichier à télécharger s'intitule Mac OS X 10.4, (PowerPC, 64-bit).

2. Sélectionnez le répertoire de destination.

La convention stipule `/usr/local`. Il est souhaitable d'utiliser cette adresse lorsque c'est possible.

3. Une fois le téléchargement terminé, placez-vous dans le répertoire où ont été téléchargés les fichiers (par exemple `/usr/local`).

Vous y voyez un fichier appelé `mysql standard` suivi du numéro de version et de `.dmg`. Si nécessaire, changez l'extension en `.dmg`.

4. Vérifiez l'intégrité du fichier téléchargé.

Reportez-vous à la section "Contrôle d'intégrité d'un fichier téléchargé", en fin d'annexe.

5. Créez un identificateur d'utilisateur et de groupe pour MySQL.

Sur les versions les plus récentes de Mac, ce groupe et cet utilisateur devraient déjà exister.

6. Montez l'image disque par un double clic sur son icône dans le Finder.

7. Faites un double clic sur l'icône du paquetage pour installer MySQL.

L'installation devrait s'effectuer dans le répertoire `/usr/local/mysql` suivi du numéro de version. L'installateur crée également un lien symbolique `/usr/local/mysql/` pointant vers le répertoire où se trouve MySQL. Dans le même temps, la base de données est initialisée par l'exécution du script `mysql_install_db` qui va ajouter un compte MySQL appelé `root`.

8. Si nécessaire, changez le propriétaire du répertoire MySQL.

Le possesseur du répertoire dans lequel MySQL est installé (par exemple `/usr/local/mysql-5.0.27`) devrait appartenir à `root`. Le sous-répertoire des données (tel que `/usr/local/mysql-5.0.27/data`) devrait dépendre du compte `mysql`. C'est

lui aussi qui devrait posséder ces deux répertoires. Si ces définitions ne sont pas correctement établies, modifiez-les à l'aide des commandes suivantes :

```
sudo chown -R root /usr/local/mysql-5.0.27
sudo chown -R mysql /usr/local/mysql-5.0.27/data
sudo chown -R root /usr/local/mysql-5.0.27/bin
```

9. Démarrez le serveur SQL en saisissant les commandes suivantes :

```
cd /usr/local/mysql
sudo ./bin/mysqld_safe
(Si nécessaire, tapez votre mot de passe ici.)
Frappez Ctrl-Z.
bg
Frappez Ctrl-D ou tapez exit.
```

Vous lancez ainsi manuellement le serveur. Autrement dit, la même procédure devra être recommencée lors de chaque démarrage de l'ordinateur. Pour éviter ce travail fastidieux, il vous faut installer l'option MySQL Startup Item, qui fait partie du disque image d'installation mais se trouve dans un paquetage séparé. Pour cela, faites un double clic sur l'icône MySQLStartupItem.pkg.

Pour arrêter le serveur MySQL placez-vous dans le sous-répertoire bin du dossier d'installation de MySQL, puis tapez :

```
mysqladmin -u root -p shutdown
```

L'option -p n'a d'utilité que si vous devez absolument saisir un mot de passe.



Contrôle d'intégrité d'un fichier téléchargé

Le site Web de MySQL propose deux techniques pour que vous puissiez vous assurer qu'un fichier que vous venez de télécharger n'a pas été altéré dans un but malveillant par un cyberdélinquant. Vous pouvez choisir entre la méthode MD5, basée sur une comparaison entre deux sommes de contrôle, et la méthode PGP. La méthode MD5 étant la plus simple, c'est celle que nous allons présenter.

Sur chaque page de téléchargement, vous remarquez à côté du nom du fichier que vous téléchargez une longue chaîne de caractères illisible. Il s'agit d'une signature. Le format est dans le style suivant :

MD5: 6112f6a730c680a4048dbab40e4107b3

Une fois que vous disposez en local du fichier MySQL que vous avez téléchargé, vous devez appliquer un petit programme sur ce fichier pour calculer la somme de contrôle MD5. Vous obtenez alors une chaîne qui doit être strictement identique à celle fournie sur le site pour le même fichier. Sous Linux et UNIX (et donc MacOS), l'utilitaire de calcul MD5 est installé d'office. Sous Windows, vous devrez télécharger cet outil et l'installer. Voyez par exemple à l'adresse www.formilab.ch/md5/.

Pour vérifier la signature MD5 d'un fichier, vous saisissez la commande suivante dans une fenêtre d'invite ou de Terminal en vous plaçant d'abord dans le dossier (répertoire) contenant le fichier à tester :

```
md5 nomdefichier
```

nomdefichier désigne le fichier téléchargé, par exemple `md5 mysql-essential-5.0.37-win32.msi`. Sous Windows, il vous faudra peut-être copier le fichier reçu vers le dossier contenant l'outil MD5 (du style `md5.exe`) et passer dans ce dossier avant de lancer la commande mentionnée ci-dessus.

La signature qui est produite et affichée doit être strictement identique à celle affichée sur la page de téléchargement du fichier sur le site MySQL.

Il existe un utilitaire simple et distribué en logiciel libre (gratuit) sous Windows qui propose une interface graphique pour tester les signatures MD5, ce qui évite de devoir saisir des commandes dans une fenêtre d'invite. Cet outil se trouve à l'adresse suivante :

www.nullriver.com/index/products/winmd5sum

Vous pouvez vérifier les téléchargements concernant Apache et PHP de la même manière.

Configuration de MySQL

Lors de son démarrage, MySQL lit un fichier de configuration. Si vous utilisez un installateur ou les réglages par défaut, vous n'aurez probablement pas besoin d'ajouter quoi que ce soit à ce fichier. Si, cependant, vous installez MySQL ailleurs que dans le répertoire par

défaut ou si vous voulez que les bases de données soient enregistrées dans un emplacement spécifique, vous devrez adapter votre fichier de configuration. ini ou my.cnf. Il se trouve dans votre dossier système (Windows ou WINNT si vous êtes sous Windows, ou encore /etc sous Linux/UNIX/Mac). On y trouve des lignes comme celles-ci :

```
[mysqld]
```

```
# Port TCP/IP que va écouter le serveur MySQL
port=3306
```

```
# Chemin d'accès au répertoire d'installation. Tous les chemins
# sont en général relatifs à celui-ci.
basedir="C:/Program Files/MySQL/MySQL Server 5.0/"
```

```
# Chemin d'accès à la racine de la base de données
datadir="C:/Program Files/MySQL/MySQL Server 5.0/Data/"
```

La ligne `basedir` indique au serveur où MySQL est installé. La ligne `datadir` spécifie l'emplacement des bases de données. Le caractère `#` signale une ligne inactive, autrement dit transformée en commentaire. Pour changer le port de communication entre le serveur et les requêtes SQL, supprimez ce caractère et redéfinissez si nécessaire le numéro du port.

Annexe B

Installation de PHP

Bien qu'il existe des versions de PHP pour de nombreuses plates-formes, je me contenterai de décrire l'installation sur Linux/UNIX/Mac et sous Windows qui constituent la grande majorité des sites Web. PHP peut tourner sur plusieurs serveurs Web, mais les instructions que je vais donner sont orientées vers Apache et IIS, parce qu'ils couvrent quelque 90 % des sites Web de l'Internet. Vous trouverez des instructions concernant d'autres systèmes d'exploitation à l'URL <http://www.php.net>.

La procédure qui suit concerne les versions 5 et 6 de PHP. Pour installer une version plus ancienne, lisez bien le fichier de commentaires `install.txt` au préalable.

Sous Linux/UNIX/Mac avec Apache

Cette section est consacrée aux instructions d'installation de PHP versions 5 et 6 sous UNIX, Linux et MacOS.

Sous Linux/UNIX

Vous pouvez installer PHP en tant que module d'Apache ou comme un interpréteur autonome. Si vous considérez PHP comme un langage de script dans une page Web dialoguant avec une base de données, mieux vaut adopter la première solution, car PHP est plus rapide et plus sûr lorsque c'est un module d'Apache. Dans ce livre, je ne parlerai pas de PHP considéré comme un interpréteur autonome.

Avant d'installer PHP, il faut tout naturellement télécharger les fichiers source, les compiler et installer les programmes résultant de

cette compilation. Ce processus n'est pas aussi technique et intimidant qu'on pourrait le craindre. Dans les sections qui suivent, nous verrons point par point les étapes à parcourir. Commencez par lire l'ensemble des étapes avant de démarrer le processus d'installation.



Pour les utilisateurs de Linux seulement : PHP pour Linux existe tant sous forme de RPM que de fichiers source. Sous forme de RPM, il se trouve peut-être sur l'un des CD de votre distribution Linux. Cependant, ce mode d'installation ne vous permet pas de contrôler les options avec lesquelles PHP a été compilé. Par exemple, vous allez avoir besoin d'installer un PHP assurant le support de MySQL, ce qui ne sera peut-être pas le cas pour votre RPM. Comme MySQL est très populaire, c'est toutefois généralement le cas. Mais peut-être y aura-t-il d'autres options dont vous n'avez nul besoin. En conséquence, la façon la plus simple et la plus efficace d'installer PHP est de partir des fichiers source. Si vous connaissez bien les RPM, vous pouvez vous risquer à installer celui de PHP. Mais vous ne trouverez pas ici d'instructions dans ce but.

Avant de commencer

Avant de commencer à installer PHP, contrôlez les points suivants :

- ✓ **Le module Apache `mod_so` est installé.** C'est généralement le cas. Pour afficher une liste des modules installés, tapez :

```
httpd -l.
```

Il se peut qu'il soit nécessaire que vous vous trouviez dans le répertoire où est situé `httpd` pour que cette commande fonctionne. La sortie liste généralement une longue suite de modules. Le seul qui vous intéresse pour PHP est `mod_so`. S'il n'est pas chargé, vous devrez réinstaller Apache en utilisant l'option `enable-module=so`.

- ✓ **L'utilitaire `apxs` est installé.** Normalement, il est installé en même temps qu'Apache. Vous devriez trouver un fichier appelé `apxs`. Mais si Apache a été installé au moyen d'un RPM, ce n'est peut-être pas le cas. Certains RPM ne contiennent, en effet, que deux modules : un pour le serveur de base Apache et l'autre pour les outils de développement Apache. Peut-être faut-il installer ce dernier, car c'est lui qui contient `apxs`.
- ✓ **La version d'Apache est récente (1.3 ou 2.0 ou plus récent).** Pour savoir ce qu'il en est, tapez la commande :


```
httpd -v
```

Il se peut qu'il soit nécessaire que vous vous trouviez dans le répertoire où est situé `httpd` pour que cette commande fonctionne.



À l'heure où nous écrivons ceci, le site Web de PHP Web déconseille d'utiliser Apache 2 avec PHP sous Linux/UNIX. Pour un site de production, mieux vaut en rester à Apache 1.3. Tenez-vous au courant de la finalisation des liens fonctionnels entre PHP et Apache 2 en visitant de temps à autre le site www.php.net/manual/en/install.unix.apache2.php.

Installation

Pour installer PHP sous Linux/UNIX avec un serveur Apache, parcourez les étapes suivantes :

1. **Affichez dans votre navigateur la page située à l'URL** `http://www.php.net`.
2. **Cliquez sur Downloads.**
3. **Sélectionnez la plus récente version du code source de PHP.** Au moment où ces lignes sont écrites, il s'agit de la 5.2.0 ou de la 6.0.0.

Une boîte de dialogue s'ouvre. Le fichier à récupérer comporte de nombreux fichiers compressés dans un fichier d'archive *tarball*.

4. **Choisissez l'option vous permettant de sauvegarder le fichier.**

Une boîte de saisie vous permet d'indiquer le répertoire dans lequel vous voulez sauvegarder le fichier.
5. **Sélectionnez le répertoire choisi. (Par exemple, `/usr/src`.) Cliquez sur Save.**
6. **Une fois le téléchargement terminé, placez-vous dans le répertoire où se trouvent les fichiers source. (Par exemple, avec la commande `cd -/usr/src`.)**

Vous voyez un fichier nommé `php-` suivi du nom de la version et de `tar.gz`.

7. Vérifiez l'intégrité du fichier téléchargé.

Voyez à ce sujet la section "Contrôle d'intégrité d'un fichier téléchargé" en fin d'Annexe A.

8. Décompressez le tarball. Pour la version 6.0.0, la commande à exécuter est :

```
gunzip -c php-6.0.0.tar.gz | tar -xf -
```

Un nouveau répertoire, appelé php-6.0.0, est créé. Il contient plusieurs sous-répertoires.

9. Placez-vous dans le nouveau répertoire créé lors de la décompression. Par exemple :

```
cd php-6.0.0
```

10. Tapez l'une des deux commandes de configuration suivantes :

```
./configure --with-mysql=DIR --with-apxs  
./configure --with-mysql=DIR --with-apxs
```

Cette commande s'attend à ce que MySQL soit installé dans son répertoire par défaut (/usr/local). Si ce n'est pas le cas, incluez le chemin d'accès à MySQL dans les options. Par exemple :

```
--with-mysql=/user/home/local
```

Si vous utilisez la variante moderne `mysql`, indiquez le chemin d'accès vers le fichier `mysql_config`.

Avec la version 4.0 de MySQL, il convient de remplacer `mysql` par `mysql`. `DIR` indique le chemin vers le répertoire de MySQL. Si vous utilisez `with-mysql`, vous devez indiquer le chemin d'accès à MySQL, par exemple :

```
--with-mysql=/user/local/mysql
```

Si vous utilisez Apache 2, l'option devra s'écrire `with-apxs2`.

Vous allez voir s'afficher de nombreuses lignes. Attendez que la configuration soit terminée. Cela peut prendre plusieurs minutes. En cas d'échec, la commande de configuration va afficher un message d'explication. En général, le problème concerne un logiciel manquant. Si vous voyez par exemple



que le programme `wyz` version 5.6 est indispensable, alors que c'est la version 4.2 qui est trouvée, vous devrez mettre à jour le logiciel dont PHP a besoin.

Si l'utilitaire `apxs` n'est pas installé là où il le devrait, vous verrez s'afficher un message d'erreur signalant qu'il n'a pas pu être trouvé. Dans ce cas, recherchez l'endroit où est installé `apxs` (`find / -name apxs`) et incluez son chemin d'accès dans l'option `with-apxs` de la commande de configuration :

`--with-apxs=/usr/sbin/apxs` (ou `/usr/local/apache/bin/apxs`). Avec Apache 2, l'option est `--with-apxs2=/usr/sbin/apxs`.

11. Tapez : `make`. Vous allez voir s'afficher plusieurs lignes. Attendez que ces sorties soient terminées. Cela peut prendre plusieurs minutes.

12. Tapez : `make install`.

Sous Mac OS X

Avec la version 4.3 de PHP, l'installation sur Mac OS X n'est pas plus compliquée qu'avec Linux/UNIX. Vous devez charger les fichiers source, les compiler, puis installer les programmes ainsi obtenus. Les étapes à suivre sont décrites pas à pas dans cette section. Lisez-les avant de commencer le travail de façon à bien les comprendre et à ne pas vous trouver bloqué au milieu de la procédure.

Avant de commencer

Si vous voulez utiliser PHP avec Apache pour votre site Web, ce dernier doit déjà être installé. C'est le cas de la plupart des systèmes Mac OS X.

Avant de commencer à installer PHP, contrôlez les points suivants :

- ✓ **Vérifiez que votre version d'Apache est récente.** Pour savoir ce qu'il en est, tapez la commande :

```
httpd -v
```

Il se peut qu'il soit nécessaire que vous vous trouviez dans le répertoire où est situé `httpd` pour que cette commande fonctionne.



Au moment où ce livre est écrit, l'utilisation conjointe de PHP et d'Apache 2 en est encore au stade expérimental. Il est donc préférable d'en rester à Apache 1.3. Renseignez-vous sur cette question en consultant l'adresse www.php.net/manual/en/install.unix.apache2.php.

- ✓ **Le module Apache `mod_so` est installé.** C'est généralement le cas. Pour afficher une liste des modules installés, tapez :

```
httpd -l
```

Il se peut qu'il soit nécessaire que vous vous trouviez dans le répertoire où est situé `httpd` pour que cette commande fonctionne. La sortie liste généralement une longue suite de modules. Le seul qui vous intéresse pour PHP est `mod_so`. S'il n'est pas chargé, vous devrez réinstaller Apache.

- ✓ **L'utilitaire `apxs` est installé.** Normalement, il est installé en même temps qu'Apache. Vous devriez trouver un fichier appelé `apxs`. Il se trouve généralement dans le répertoire `/usr/sbin/apxs`. Si vous localisez ce fichier, `apxs` est installé. Sinon, il ne l'est pas.
- ✓ **Les fichiers du kit de développement sont installés.** Ils se trouvent sur un CD venant en complément de la distribution principale de Mac OS X. Si vous ne le possédez pas, il reste possible de télécharger des outils sur le site Web Apple Developer Connection, à l'adresse developer.apple.com/tools/macosxtools.html.

Installation

Pour installer PHP sur Mac, parcourez les étapes suivantes :

1. Affichez dans votre navigateur la page située à l'URL <http://www.php.net>.
2. Cliquez sur Downloads.
3. Sélectionnez la plus récente version du code source de PHP. Au moment où ces lignes sont écrites, il s'agit de la 6.0.0. Une boîte de dialogue s'ouvre.
4. Choisissez l'option vous permettant de sauvegarder le fichier. Une boîte de saisie vous permet d'indiquer le répertoire dans lequel vous voulez sauvegarder le fichier.

5. Sélectionnez le répertoire choisi. (Par exemple, `/usr/src`.) Cliquez sur **Save**.
6. Une fois le téléchargement terminé, placez-vous dans le répertoire où se trouvent les fichiers source. (Par exemple, avec la commande `cd -/usr/src`.) Vous voyez un fichier nommé `php-` suivi du nom de la version et de `tar.gz`. On appelle ce fichier un *tarball*. Si ce fichier a été automatiquement décompacté par Stuffit Expander, vous devriez trouver un répertoire appelé `php-6.0.0`. Dans ce cas, passez directement à l'étape 8.
7. Vérifiez l'intégrité du fichier téléchargé.

Voyez à ce sujet la section "Contrôle d'intégrité d'un fichier téléchargé" en fin d'Annexe A.

8. Décompressez le fichier d'archive *tarball*.

Pour la version 6.0.0, la commande à exécuter est :

```
tar xvfz php-6.0.0.tar.gz
```

Un nouveau répertoire, appelé `php-6.0.0`, est créé. Il contient plusieurs sous-répertoires.

9. Placez-vous dans le nouveau répertoire créé lors de la décompression. Par exemple :

```
cd php-6.0.0
```

10. Tapez la commande de configuration `configure` :

Cette commande possède la syntaxe `./configure` suivie de toutes les options nécessaires. La configuration minimale se compose des éléments suivants :

- **Emplacement** : Le Mac ne place pas les fichiers dans les emplacements par défaut prévus par PHP. Vous devez donc lui indiquer où se trouvent ces fichiers. Utilisez les options suivantes :

```
--prefix=/usr
--sysconfdir=/etc
--localstatedir=/var
--mandir=/usr/share/man
```


- **Option zlib** : `--with-zlib`.
- **Option Apache** : Si vous installez PHP pour qu'il fonctionne avec Apache, ajoutez l'option `--with-apxs` (ou `--with-apxs2` avec Apache 2).

L'option de configuration la plus probable devrait donc être construite ainsi :

```
./configure --prefix=/usr --sysconfdir=/etc
--localstatedir=/var --mandir=/usr/share/man
--with-apxs --with-zlib
```

Vous avez également besoin d'une option pour le support MySQL. Elle se présente comme suit :

```
--with-mysql=DIR
--with-mysqli=DIR
```

Avec `with-mysqli`, indiquez le chemin d'accès au fichier nommé `mysql_config`.

Vous utilisez la variante `mysql` pour MySQL 4.0 et les versions encore plus anciennes ; utilisez `mysqli` à partir de la version MySQL 4.1. `DIR` indique le chemin d'accès au répertoire MySQL. Avec `with-mysql`, indiquez le chemin d'accès au répertoire d'installation de `mysql` comme ceci :

```
--with-mysql=/user/local/mysql
```

La commande de configuration peut être entrée sur une seule ligne. Pour la décomposer, tapez un `\` (et un seul) à la fin de chaque ligne.

Vous allez voir s'afficher de nombreuses lignes. Attendez que la configuration soit terminée. Cela peut prendre plusieurs minutes.



Si l'utilitaire `apxs` n'est pas installé là où il le devrait, vous verrez s'afficher un message d'erreur signalant qu'il n'a pas pu être trouvé. Dans ce cas, recherchez l'endroit où est installé `apxs` (`find / -name apxs`) et incluez son chemin d'accès dans l'option `with-apxs` de la commande de configuration :

```
--with-apxs=/usr/sbin/apxs (ou /usr/local/apache/bin/
apxs).
```

Bien d'autres options sont possibles, par exemple pour définir la base de données que vous utilisez ou les répertoires dans lesquels PHP est installé. Nous y reviendrons plus loin dans la section intitulée "Options d'installation".

11. Tapez : `make`. Vous allez voir s'afficher plusieurs lignes. Attendez que ces sorties soient terminées. Cela peut prendre plusieurs minutes.
12. Tapez : `sudo make install`.

Autres méthodes d'installation sur Mac

La plupart des versions de MacOS X depuis la X.3 sont dotées de PHP préinstallé, mais parfois non encore activé. Si PHP est bien installé mais semble ne pas fonctionner, reportez-vous à la section ultérieure "Configuration d'Apache pour PHP". En général, il suffit d'appliquer une retouche au fichier `httpd.conf`.

Certaines versions de MacOS X sont livrées avec un paquetage précompilé de PHP dont l'installation est encore plus facile qu'un fichier à compiler. En revanche, vous n'avez plus dans ce cas la possibilité de choisir l'étendue des fonctions PHP activées (vous en aurez trop ou trop peu). Si vous craignez de compiler des fichiers source sur Mac, visitez la page <http://www.entropy.ch/software/macosx/php/> et voyez ce qui est proposé.

Options d'installation

Les sections précédentes vous ont présenté les étapes que vous devez parcourir pour installer rapidement PHP avec les options nécessaires pour exécuter les applications de ce livre. Cependant, vous pouvez souhaiter installer PHP de façon différente : par exemple, tous les programmes PHP et les fichiers installés dans leurs emplacements par défaut, mais PHP lui-même ailleurs.

Ou bien vous avez peut-être l'intention d'utiliser des logiciels complémentaires. Si vous souhaitez configurer PHP pour des besoins spécifiques, vous pouvez ajouter des options à la commande de configuration de l'étape 10 de l'installation sous Linux/UNIX ou MacOS X. En général, l'ordre de ces options est indifférent. Le Tableau B.1 montre les options les plus courantes pour PHP. Pour en connaître la liste complète, tapez : `configure -help`.

Tableau B.1 : Options de configuration de PHP.

Option	Rôle
<code>prefix=PREFIX</code>	Définit le répertoire principal de PHP comme étant <code>PREFIX</code> . Par défaut, c'est <code>/usr/local</code> .
<code>exec-prefix=EPREFIX</code>	Installe des fichiers dépendant de l'architecture dans <code>EPREFIX</code> . Par défaut, <code>EPREFIX</code> est identique à <code>PREFIX</code> .
<code>bindir=DIR</code>	Installe des exécutables utilisateur dans <code>DIR</code> . Par défaut, c'est <code>EPREFIX/bin</code> .
<code>infodir=DIR</code>	Installe de la documentation dans <code>DIR</code> . Par défaut, c'est <code>PREFIX/info</code> .
<code>mandir=DIR</code>	Installe les fichiers <i>man</i> dans <code>DIR</code> . Par défaut, c'est <code>PREFIX/man</code> .
<code>with-config-file-path=DIR</code>	Recherche le fichier de configuration <code>php.ini</code> dans <code>DIR</code> . Sans cette option, PHP le cherchera dans son emplacement par défaut, généralement <code>/usr/local/lib</code> .
<code>disable-libxml</code>	Désactive le support XML qui est inclus par défaut.
<code>enable-ftp</code>	Active le support de FTP.
<code>enable-magic-quotes</code>	Active l'échappement automatique des guillemets et apostrophes par un antislash (<code>\</code>).
<code>with-apxs=FILE</code>	Construit un module Apache partagé au moyen de l'utilitaire <code>apxs</code> qui se trouve en <code>FILE</code> . Par défaut, <code>FILE</code> vaut <code>apxs</code> .
<code>with-apxs2=FILE</code>	Construit un module Apache 2 partagé au moyen de l'utilitaire <code>apxs</code> qui se trouve en <code>FILE</code> . Par défaut, <code>FILE</code> vaut <code>apxs</code> .
<code>with-mysql=DIR</code>	Active le support des bases de données MySQL (versions 4.0 et antérieures). Par défaut, le répertoire de MySQL est <code>/usr/local</code> .
<code>with-mysqli=DIR</code>	Active le support des bases de données MySQL (versions 4.1 et postérieures). Par défaut, le répertoire de MySQL est <code>/usr/local</code> . <code>DIR</code> doit indiquer le chemin d'accès au fichier <code>mysql_config</code> qui a été installé avec la version 4.1. Disponible seulement à partir de la version 5 de PHP 5.
<code>with-openssl=DIR</code>	Active le support d'OpenSSL pour un serveur sécurisé. Demande la version 0.9.5 ou plus récente d'OpenSSL.
<code>with-ori8=DIR</code>	Active le support d'Oracle version 7 ou plus. Par défaut, <code>DIR</code> se trouve dans la variable d'environnement <code>ORACLE_HOME</code> .

Option	Rôle
<code>with-oracle=DIR</code>	Active le support des anciennes versions d'Oracle. Par défaut, DIR se trouve dans la variable d'environnement <code>ORACLE_HOME</code> .
<code>with-pgsql=DIR</code>	Active le support des bases de données PostgreSQL. Par défaut, DIR vaut <code>/usr/local/pgsql</code> .
<code>with-servlet=DIR</code>	Inclut le support des <i>servlets</i> . DIR est le répertoire d'installation de base du JSDK. L'extension Java doit être construite sous forme de librairie dynamique partagée (<i>shared dl</i>).

Configurer Apache pour PHP

Vous devez configurer Apache pour qu'il reconnaisse les fichiers PHP comme le fait PHP lui-même. Un fichier de configuration Apache appelé `httpd.conf` existe sur votre système. Il peut se trouver dans `/etc` ou dans `/usr/local/apache/conf`. Vous devez éditer ce fichier pour que PHP fonctionne correctement.

Effectuez les modifications suivantes :

1. Ouvrez le fichier `httpd.conf` pour le modifier.
2. Configurez Apache pour charger le module PHP.

Trouvez la série d'instructions `LoadModule` et parmi elles la ligne suivante :

```
LoadModule php6_module libexec/libphp6.so
```

Si vous ne la trouvez pas, ajoutez-la. Si elle existe, mais avec un caractère dièse (`#`) en tête, supprimez ce dernier.

Pour PHP 5, la ligne s'écrit :

```
LoadModule php5_module libexec/libphp5.so
```

3. Configurez Apache pour qu'il reconnaisse les extensions PHP.

Vous devez indiquer à Apache quels sont les fichiers pouvant contenir du code PHP. Recherchez la section descriptive

d'AddType. Vous pouvez trouver une ou plusieurs lignes de ce type. Pour PHP, c'est :

```
AddType application/x-httpd-php .php
```

Cette ligne signale à Apache que les scripts PHP ont pour extension `.php`. Si vous ne la trouvez pas, ajoutez-la. Si elle existe, mais avec un caractère dièse (`#`) en tête, supprimez ce dernier.

4. **Lancez (s'il ne tourne pas) ou relancez (si vous l'aviez arrêté) le serveur Apache httpd.**

Pour cela, il existe un script appelé `apachectl` ou `httpd.apache`. Il peut se trouver dans le répertoire `/bin` du répertoire où Apache a été installé (par exemple : `/usr/local/apache/bin`). Vous pouvez lancer le serveur en tapant `apachectl start`, le relancer en tapant `apachectl restart`, ou l'arrêter en tapant `apachectl stop`. Il se peut que vous deviez auparavant arrêter le serveur avant de pouvoir le relancer.

Installation de PHP sous Windows

PHP peut tourner sous Windows NT/2000/XP. Sous Windows 98/Me, vous pouvez programmer mais pas gérer un site Web. Windows 95 n'est plus supporté depuis la version 4.3.0 de PHP. Windows 3.1 n'a jamais été supporté.

Pour installer PHP 5 ou 6 avec support de MySQL sous Windows, vous devez télécharger un fichier Zip contenant tous les fichiers de PHP. Exécutez les étapes suivantes :

1. **Affichez dans votre navigateur la page située à l'URL `http://www.php.net`.**
2. **Cliquez sur Downloads.**
3. **Allez à la section Windows Binaries. Cliquez sur le lien de téléchargement de l'installateur correspondant à la version la plus récente de PHP (c'est la 6.0.0 au moment où ces lignes sont écrites).**
4. **Cliquez sur le lien de choix du site miroir et choisissez celui qui est le plus proche de votre emplacement géographique.**

Une boîte de dialogue s'ouvre.

5. Cliquez sur le bouton **Enregistrer** pour sauvegarder le fichier.

Une boîte de dialogue vous permet de choisir le répertoire de téléchargement.

6. Naviguez jusqu'au répertoire de destination (un répertoire temporaire dédié à cet usage) et cliquez sur **Enregistrer**.

Une fois le téléchargement terminé, vous trouvez dans ce répertoire le fichier nécessaire qui s'appelle `php-` suivi de son numéro de version et de `-win32.exe`. Pour la version courante, ce nom est donc : `php-6.0.0-win32.exe`.

7. Vérifiez l'intégrité du fichier téléchargé.

Voyez à ce sujet la section "Contrôle d'intégrité d'un fichier téléchargé" en fin d'Annexe A.

8. Procédez à l'extraction des fichiers de l'archive `.zip` dans le répertoire d'installation désiré de PHP (par exemple `C:\php`).

Si vous double-cliquez sur le nom du fichier `.zip`, cela doit lancer l'utilitaire déclaré sur votre système pour extraire les fichiers `.zip` (WinZip, PKZIP ou 7Zip). La destination `C:\php` est conseillée parce que de nombreux fichiers de configuration sont prédéfinis avec ce chemin. Cela vous évitera donc d'avoir à les modifier.

Il est fortement déconseillé d'installer PHP dans un répertoire dont le chemin d'accès comporte des espaces comme `Program Files/PHP`. Des problèmes peuvent en résulter.

Vous disposez dorénavant d'un répertoire avec plusieurs sous-répertoires peuplés avec les fichiers de l'archive.

9. Copiez le fichier dont MySQL a besoin dans le répertoire principal de PHP.

Ce fichier se situe dans le sous-répertoire `ext` de celui d'installation de PHP. Copiez l'un des deux fichiers suivants, selon votre version de MySQL :



ext\php_mysql4i.dll (MySQL 4.1 et suivants)
 ext\php_mysql.dll (MySQL 4.0 ou plus ancien)

Le fichier doit être placé dans le répertoire principal de PHP (par exemple C:\php).



Un autre fichier est requis pour le support de MySQL. Il s'agit d'une librairie dynamique (un exécutable). Ce fichier appelé libmysql.dll devrait déjà se trouver dans le répertoire principal de PHP. Si vous ne l'y trouvez pas, vous devez aller le chercher dans le répertoire d'installation de MySQL ou son sous-répertoire bin (cherchez par exemple dans c:\Program Files\MySQL\MySQL Server 5.0\bin).



De temps à autre, PHP réclame des librairies DLL qu'il ne parvient pas à trouver et affiche un message d'erreur approprié. La DLL manquante se trouve en général dans le sous-répertoire ext. Il suffit de le recopier dans le répertoire principal de PHP.

10. Configurez votre serveur Web.

Voyez la section suivante à ce sujet.

11. Configurez PHP.

Voyez la dernière section de cette annexe à ce sujet.

Configurer votre serveur Web pour PHP

Votre serveur Web doit être reconfiguré sous Windows pour reconnaître les fichiers de scripts PHP et pouvoir les exécuter. Sachez que vous ne pouvez lancer les deux serveurs Apache et IIS en même temps avec les mêmes numéros de ports. Soit vous arrêtez un des deux serveurs, soit vous modifiez les ports associés à l'un des deux.

Voyons tour à tour les deux principaux cas de figure, d'abord avec Apache puis avec IIS.

Configurer Apache pour PHP

Avec Apache, il est nécessaire de faire cette configuration manuellement. Pour cela, suivez les étapes ci-après :

1. Editez le fichier `httpd.conf`.

Cliquez sur Démarrer/Programmes/Apache Web Server/Configure Apache Server/Edit Configuration.

Si Edit Configuration ne figure pas dans votre menu Démarrer, recherchez le fichier `httpd.conf` sur votre disque dur. Il est généralement dans le répertoire où Apache a été installé, plus précisément dans un sous-répertoire appelé `conf` (par exemple : `c:\program files\Apache group\Apache\conf`). Ouvrez ce fichier dans un éditeur tel que le Bloc-notes ou Wordpad.

2. Activez le module PHP.

Cherchez la section de l'instruction `module` et trouvez la ligne suivante :

```
#LoadModule php6_module "c:/php/php6apache2.dll"
```

Supprimez le signe `#` de neutralisation du début de la ligne pour l'activer. Si vous utilisez la version 5 de PHP, la ligne se lit ainsi :

```
LoadModule php5_module "c:/php/php5apache2.dll"
```

Si vous utilisez Apache 1.3 et non Apache 2, le nom de module est `php6apache.dll` ou `php5apache.dll`.

3. Indiquez à Apache quels sont les fichiers contenant du code PHP.

Recherchez dans `httpd.conf` une section contenant des instructions `AddType`. Elle peut déjà en contenir quelques-unes pour d'autres logiciels. Celle qui concerne PHP s'écrit :

```
AddType application/x-httpd-php .php
```

Recherchez cette ligne. Si elle est précédée d'un caractère dièse (`#`), supprimez-le. Si elle n'existe pas, créez-la. Elle indique à Apache que le code PHP se trouve dans tous les fichiers ayant une extension `.php`. Vous pouvez spécifier n'importe quelle extension ou suite d'extensions.

Cette ligne indique à Apache que les fichiers à extension `.php` sont du type applicatif `application/x-httpd-php`. Apache sait ainsi qu'il doit transmettre ces fichiers au module PHP.

4. Lancez ou relancez Apache.

Vous pouvez le lancer en tant que service sous WindowsNT/2000/XP en cliquant sur Démarrer/Programmes/Apache Web Server/Control Apache Server et en choisissant **Stop** puis **Start**. Sous Windows 95/98/ME, il faut le lancer en cliquant sur Démarrer/Programmes/Apache Web Server/Management.

Dans certains cas, relancer Apache n'est pas suffisant. Vous devez commencer par l'arrêter, puis ensuite le relancer. En outre, votre machine est probablement configurée pour lancer automatiquement Apache à son démarrage. Une autre solution consiste donc à l'arrêter puis à réamorcer l'ordinateur pour relancer Apache.

Configurer IIS pour PHP

Voici comment configurer IIS pour travailler avec PHP :

1. Activez la console de gestion de IIS.

Ouvrez le Panneau de configuration. Choisissez la catégorie Performances et maintenance, puis Outils d'administration. Faites un double clic sur l'icône Services Internet (IIS).

2. Dans le panneau de gauche, activez la rubrique Sites Web. Choisissez votre site dans le panneau de droite (par exemple Site Web par défaut).

3. Cliquez avec le bouton droit de la souris et choisissez l'option Propriétés.

4. Activez l'onglet Répertoire de base.

5. Cliquez sur le bouton Configuration.

6. Activez l'onglet Mappages. Si l'extension `.php` n'apparaît pas dans la liste des extensions, cliquez sur le bouton Ajouter.

7. Dans le champ Exécutable, entrez le chemin d'accès complet à l'interpréteur PHP. Par exemple : `c:\php\php.exe`. Vous pouvez vous servir du bouton Parcourir pour le localiser.

8. Dans le champ Extension, entrez `.php`. Cette extension sera associée aux scripts PHP.

9. Cochez la case Moteur de script.

10. Validez.

Répétez les étapes 7 à 10 si vous voulez ajouter d'autres extensions (par exemple, .phtml).

Configurer PHP pour MySQL

Le comportement de PHP est notamment régi par un fichier de configuration appelé `php.ini`. En cas d'absence de ce fichier, il sert de valeurs par défaut.

Le fichier `php.ini` se trouve normalement dans l'un des emplacements suivants :

Sous Windows, dans le dossier appelé `winnt` (pour Windows 2000/NT) ou `windows` (toutes les autres versions).

Sous UNIX/Linux/Mac, dans le répertoire `/usr/local/lib`.

Si vous ne trouvez pas `php.ini` au bon endroit, copiez le modèle `php.ini-dist`. Renommez ensuite cette version en `php.ini`.



Si une ancienne version de PHP était déjà installée sur votre machine, commencez par renommer `php.ini` **avant** d'installer la nouvelle version de ce fichier. Vous pourrez ainsi récupérer vos réglages spécifiques pour redéfinir la configuration de PHP.



Au fil du livre, nous avons vu comment adapter certains paramètres de PHP (par exemple dans le Chapitre 4 pour ce qui concerne les niveaux d'alerte en cas d'erreur). Nous rappellerons simplement ici comment activer le support de MySQL, sachant que l'essentiel des réglages par défaut donne généralement satisfaction.

Voici comment configurer PHP pour activer le support de MySQL :

1. Ouvrez `php.ini` afin de l'éditer (vous devez disposer de droits d'administration suffisants).
2. Modifiez les paramètres désirés.

Les étapes 3, 4 et 5 présentent les paramètres qu'il est toujours nécessaire de modifier si vous utilisez l'environnement spécifié.

- 3. Seulement si vous utilisez PHP 5 ou une version antérieure, vous devez désactiver les magic quotes.**

Cherchez la ligne suivante :

```
magic_quotes-gpc On
```

Remplacer On par Off.

- 4. Seulement si vous utilisez PHP 5/6 sous Windows, activez le support de mysqli ou celui de mysql.**

Cherchez la liste des extensions et trouvez la ligne de l'extension mysqli :

```
;extension=php_mysqli.dll
```

Si vous utilisez une version de PHP antérieure à la 5.0 ou une version de MySQL antérieure à la 4.1, cherchez la ligne suivante :

```
;extension=php_mysql.dll
```

Vous remarquez le point-virgule (;) en début de ligne. Il faut le supprimer pour activer l'extension. Si vous ne trouvez pas cette ligne dans `php.ini`, il faut l'insérer.

- 5. Uniquement dans le cas où vous utilisez PHP avec le serveur IIS, désactivez l'option force redirect. Localisez la ligne :**

```
;cgi.force_redirect = 1
```

Supprimez le point-virgule en début de ligne et changez le 1 en 0 pour obtenir :

```
cgi.force_redirect = 0
```

- 6. Enregistrez `php.ini` et refermez la fenêtre de l'éditeur.**

- 7. Il peut être nécessaire de relancer le serveur Apache pour que la nouvelle configuration soit reconnue.**

En général, les autres paramètres de configuration de PHP conviennent, mais vous devrez peut-être en retoucher certains dans des cas particuliers. Nous présentons ces retouches au fichier `php.ini` tout au long du livre lorsque le besoin s'en fait sentir. Par exemple, les

actions de gestion d'erreur de PHP peuvent être modifiées en intervenant dans ce fichier. Les réglages et leurs effets ont été indiqués dans le Chapitre 4.

Si vous modifiez `php.ini` mais que les effets ne sont pas ceux que vous espériez sur le fonctionnement de PHP, il y a deux causes probables. Sachez tout d'abord qu'il faut arrêter et redémarrer le serveur Web pour que la modification soit prise en compte. Ensuite, vous n'avez peut-être pas modifié le bon exemplaire du fichier `php.ini` (ce n'est pas celui que lit PHP). Vous devez vérifier où PHP s'attend à pouvoir lire ce fichier. Lorsque vous testez PHP via la méthode `phpinfo()` comme montré dans le Chapitre 2, PHP affiche les valeurs de nombreuses variables et options. Vers le début de la liste, une option se nomme *Configuration File Path*. Elle indique le chemin d'accès au répertoire dans lequel PHP cherche les fichiers de configuration. Si le chemin se termine avec un nom de fichier, c'est le fichier que PHP exploite pour sa configuration. Si le chemin se termine sur un nom de sous-répertoire, c'est que PHP cherche à cet endroit mais n'a pas trouvé le fichier. Il utilise dans ce cas sa configuration par défaut.

Annexe C

EasyPHP et PHPMyAdmin

Si vous avez parcouru le contenu des Annexes A et B qui détaillent l'installation de PHP et de MySQL, sans doute avez-vous été effrayé par le nombre de manipulations à effectuer avant de pouvoir exécuter la moindre opération sur une base de données. On se croirait revenu au bon (?) vieux temps de MS-DOS, il y a une vingtaine d'années, lorsqu'il fallait taper une suite de commandes ésoériques aux options incompréhensibles pour effectuer la moindre tâche. Passe encore pour les Linuxiens convaincus qui ont

N.d.T. : Calmez vos ardeurs, chers lecteurs. La version actuelle la plus récente d'EasyPHP (1.8) ne contient ni PHP 5 ni MySQL 5. Voici son contenu :

Apache 1.3.33.

PHP 4.3.10.

MySql 4.1.9.

phpMyAdmin 2.6.1.

Vous pouvez procéder à une mise à jour de PHP en lisant la procédure fournie dans la Foire aux questions :

<http://www.easypHP.org/faq.php3#50>

Très prochainement, sortira la version 2.0 d'EasyPHP qui réunit des versions récentes de MySQL et de PHP.

conservé l'habitude de ces manipulations et qui y trouvent un certain plaisir, mais ceux qui sont passés sous les fourches caudines de Bill Gates ne savent plus que cliquer sur des icônes.

Pratiquants de Windows, reprenez espoir ! Cette annexe n'est pas l'Enfer mais le Paradis. Nous allons vous montrer qu'on peut installer PHP et MySQL plus simplement encore qu'on installe Word, pour peu qu'on dispose du package tout ficelé appelé EasyPHP.

Pourquoi installer PHP et MySQL ?

En toute rigueur, cela n'a absolument rien d'indispensable. Quelques fournisseurs d'accès (mais pas tous) vous proposent d'utiliser ces logiciels en vous connectant sur leur site. Dans ces conditions, plus rien à installer chez vous : c'est le fournisseur d'accès qui a fait tout le sale boulot à votre place. Revers de la médaille, vous allez devoir télécharger vos scripts dans le répertoire qu'il vous a attribué et consommer du temps de télécommunication. Si vous avez l'ADSL ou le câble, cela n'a guère d'importance. Mais si vous n'avez qu'une connexion par modem, vous allez très vite consommer votre forfait. En outre, les délais de mise au point vont s'accroître et, en cas de panne ou de surcharge des lignes téléphoniques, vous serez condamné à l'inaction.

Avoir PHP et MySQL chez soi constitue donc un avantage marqué, non seulement pour l'apprentissage, mais plus tard pour l'écriture et la mise au point des programmes. Toutefois, n'allez pas croire que vous allez pouvoir exploiter ces programmes en ouvrant votre serveur personnel sur le monde entier. Comme nous l'avons expliqué au Chapitre 2, il ne faut pas y compter. Pour l'exploitation, vous devrez **toujours** recourir aux bons offices d'un professionnel, que ce soit un fournisseur d'accès ou un hébergeur payant.

Donc, la cause est entendue, vous devez avoir PHP et MySQL sous la main, sur votre propre machine.

EasyPHP, qu'est-ce ?

Allez sur le site <http://www.easyphp.org>. Vous y verrez la page reproduite sur la Figure C.1, page dont le premier paragraphe se lit ainsi :

EasyPHP installe et configure automatiquement un environnement de travail complet permettant de mettre en œuvre toute la puissance et la souplesse qu'offrent le langage dynamique PHP et son support efficace des bases de données. EasyPHP regroupe un serveur Apache, une base de données MySQL, le langage PHP ainsi que des outils facilitant le développement de vos sites ou de vos applications.

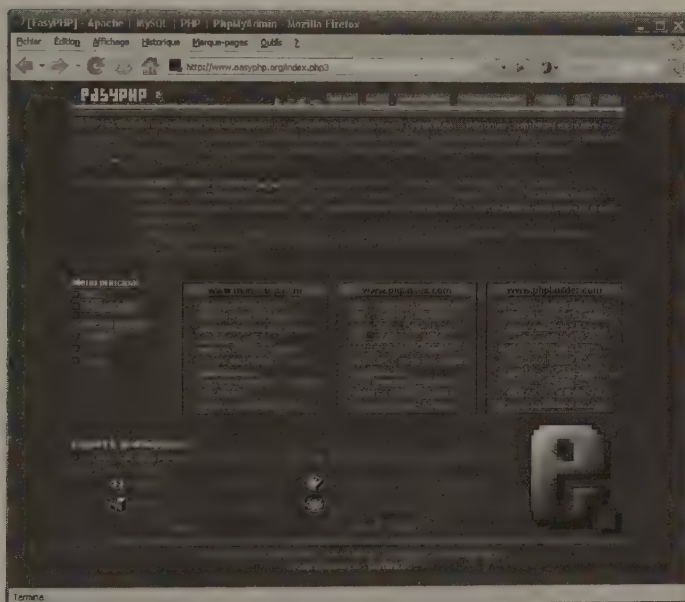


Figure C.1 :
La page
d'accueil du
site Web de
EasyPHP.

A la date où ce livre est écrit, la version courante possède la référence 1.8.0. Elle comprend, réunis en un unique fichier d'un peu plus de 10 Mo, les logiciels suivants :

- ✓ Apache 1.3.33
- ✓ PHP 4.3.10
- ✓ MySQL 4.1.9
- ✓ PHPMyAdmin 2.6.1 (outil d'administration que nous présentons *in fine* dans cette même annexe).

Ce ne sont pas les toutes dernières déclinaisons en date de chacun des programmes, mais le principal est là. Vous disposez bien des

versions principales stables actuellement diffusées. Donc, tout va bien.

Téléchargement de EasyPHP

Pour télécharger ce package, cliquez sur le menu "téléchargements", tout en haut de la Figure C.1. Dans la liste des chargements qui vous sont proposés, choisissez la première ligne, celle qui correspond à la version actuelle (soit EasyPHP 1.8.0 à la date de rédaction du livre). Il n'y a de toute façon aucune raison de choisir une ancienne version.

Choisissez ensuite le répertoire de votre machine dans lequel vous voulez que ce fichier atterrisse et attendez un "certain temps" (un peu moins d'une heure avec une connexion par modem, beaucoup moins avec l'ADSL ou le câble).

Installation de EasyPHP



Ne prenez aucun risque. Dans votre répertoire Windows, commencez par renommer les fichiers `php.ini` et `my.ini` (ajoutez par exemple une seconde extension, `.old`). Faites de même pour le lien `My`. De cette façon, vous ne perdrez pas votre configuration actuelle et vous pourrez facilement la récupérer plus tard.

Vous aurez ainsi téléchargé un fichier qui s'appelle `EasyPhp1-8_setup.exe` (dans le cas de la version 1.8) dont vous allez lancer l'exécution de la manière la plus classique qui soit : en double-cliquant sur son nom dans l'Explorateur de Windows. Suivez la procédure d'installation comme pour tout autre logiciel. Mais n'allez pas trop vite. Suivez scrupuleusement les recommandations qui vous sont faites dans l'écran d'information présenté sur la Figure C.2.



Sous Windows NT/2000/XP, MySQL est installé en tant que service. Celui-ci **doit** être arrêté avant d'installer EasyPHP. Pour cela, ouvrez le Panneau de configuration, puis choisissez successivement les catégories Performance et maintenance puis Outils d'administration. Ouvrez la console Services. Faites défiler la liste de droite pour repérer la ligne MySQL. Activez-la, puis cliquez sur le lien Arrêter le service. Si vous utilisez le serveur Microsoft IIS, faites de même pour le service Administration IIS. Si vous utilisez Apache comme serveur Web, arrêtez-le (voir à ce sujet l'Annexe B).

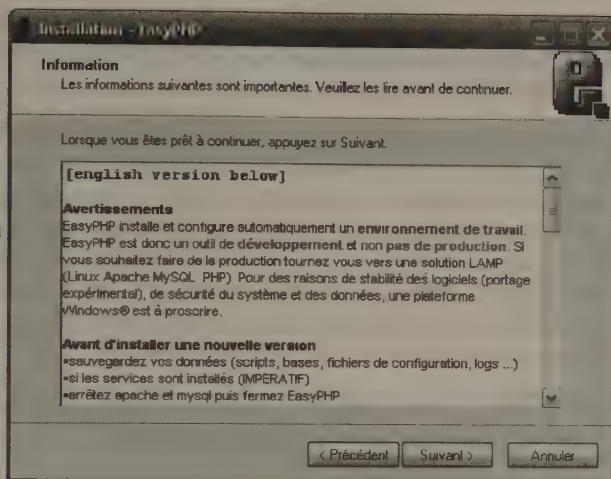


Figure C.2 :
Avant de poursuivre l'installation de EasyPHP, prenez toutes vos précautions.

Choisissez le dossier de destination. Il s'agit par défaut de C:\Program Files\EasyPHP1-8. Je vous conseille vivement de ne rien changer à cela. Poursuivez normalement la procédure.

Une fois les fichiers copiés, lisez soigneusement le nouvel écran d'information qui s'affiche. Il serait même bon de copier ce texte et de l'imprimer.

A la fin, vous aurez sous les yeux la fenêtre de fin d'installation. Supprimez la coche placée devant : "Ouvrir la page d'accueil". Cliquez sur le bouton "Terminer", et c'est tout.

Eh oui, vous venez en moins de cinq minutes, sans aucune manipulation délicate ou complexe, d'installer le serveur personnel Apache, l'interpréteur PHP et le SGBD MySQL ! Incroyable, peut-être, mais vrai.

Utilisation de EasyPHP

Pour lancer EasyPHP, ouvrez votre menu Démarrer. Choisissez tous les programmes, puis localisez le groupe EasyPHP 1.8. Cliquez sur le nom de commande EasyPHP 1.8. C'est parti !

La présence de EasyPHP se manifeste par son logo affiché du côté droit de la barre des tâches. Tous les éléments ne sont peut-être actifs dès le démarrage. Pour le savoir :

- ✓ Cliquez sur l'icône d'EasyPHP avec le bouton droit de la souris. Dans le menu qui s'affiche, choisissez Configuration, puis EasyPHP.
- ✓ Vérifiez maintenant l'état des services Apache et MySQL (voir la Figure C.3.). Si nécessaire, cliquez sur le bouton correspondant, puis choisissez l'option Démarrer. Profitez-en pour décider si vous voulez démarrer EasyPHP lors de l'ouverture de Windows et activer les serveurs en tant que services XP (ce n'est peut-être pas une bonne idée si vous voulez revenir ensuite facilement à votre configuration précédente : vous avez tout le temps de réfléchir à la question).
- ✓ Lorsque vous avez terminé, cliquez sur Appliquer, puis réduisez la fenêtre dans la barre des tâches.

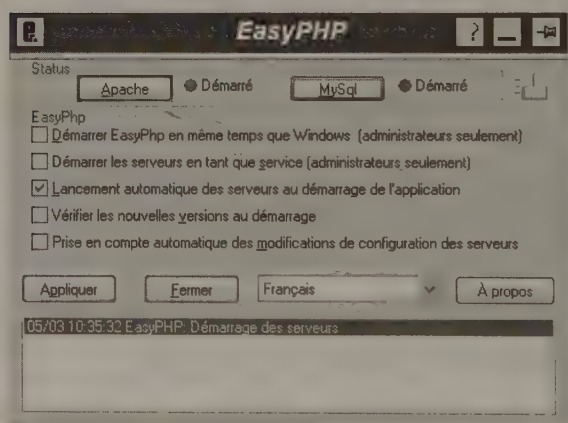


Figure C.3 :
Configuration
de base pour
EasyPHP.

Une fois Apache et MySQL opérationnels, un petit point rouge clignotera au bout de la branche inférieure du "e" dans l'icône d'EasyPHP.

Si vous avez retenu (comme nous vous y avons invité) les options d'installation par défaut, PHP est installé dans le répertoire `C:\Program Files\EasyPHP1-8`, l'interpréteur PHP (`php.exe`) étant dans le sous-répertoire `\php`, MySQL (`mysql.exe`) dans le sous-répertoire `\MySQL\bin`, et enfin Apache dans le sous-répertoire `\Apache`. Les scripts PHP devront se trouver dans le sous-répertoire `C:\Program Files\EasyPHP\www`.

Pour vérifier que tout est en état de marche, saisissez (avec le Bloc-notes de Windows, par exemple) le script de trois lignes suivant que vous sauvegarderez sous le nom de `salut.php` dans `C:\Program Files\EasyPHP\www` :

```
<?
echo "<h2>Salut, les copains !</h2>";
?>
```

Lancez alors votre navigateur et, dans sa fenêtre d'adresse, tapez :

`localhost/salut.php`

Si tout se passe bien, vous allez voir s'afficher ce qui est reproduit sur la Figure C.4.

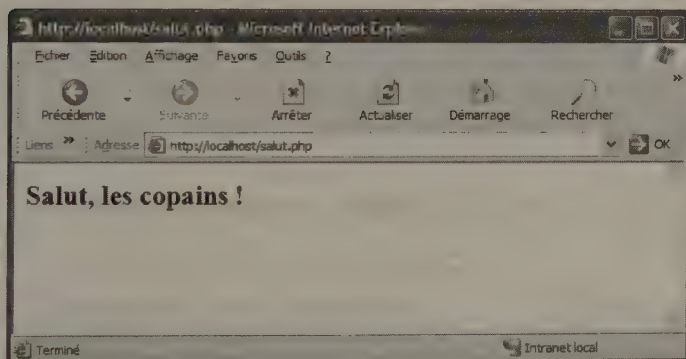


Figure C.4 :
Exécution du
petit script
d'essai de
EasyPHP.

Pour arrêter PHP et MySQL, cliquez du bouton droit sur l'icône clignotante de EasyPHP, dans le coin inférieur droit de votre écran, puis cliquez du bouton gauche sur l'entrée de menu Quitter. Pour arrêter temporairement le serveur personnel, cliquez sur Arrêter. Pour reprendre, cliquez sur Redémarrer.



Il est conseillé de créer un sous-répertoire de `C:\Program Files\EasyPHP\www` par projet. Ainsi, pour tester l'application de l'animalerie, vous pouvez créer un sous-répertoire `C:\Program Files\EasyPHP\www\animaux` dans lequel viendront prendre place les scripts de cette application. Pour leur exécution, vous devrez alors taper, par exemple : `localhost/animaux/AjouterAnimal.php`. La liste des répertoires locaux est affichée en choisissant dans le menu de l'icône EasyPHP l'option Web local.

PHPMyAdmin

PHPMyAdmin est un outil d'administration de base de données qui n'est pas spécifique de EasyPHP, mais qu'on peut utiliser quelle que soit la façon dont a été installé MySQL.

Grâce à PHPMyAdmin, plus besoin de recourir à `mysqld.exe` dans une fenêtre MS-DOS, vous restez sous Windows. Il s'agit d'une suite de scripts PHP dont vous appelez la tête de pont en tapant tout simplement dans la fenêtre de votre navigateur :



`localhost/home/`

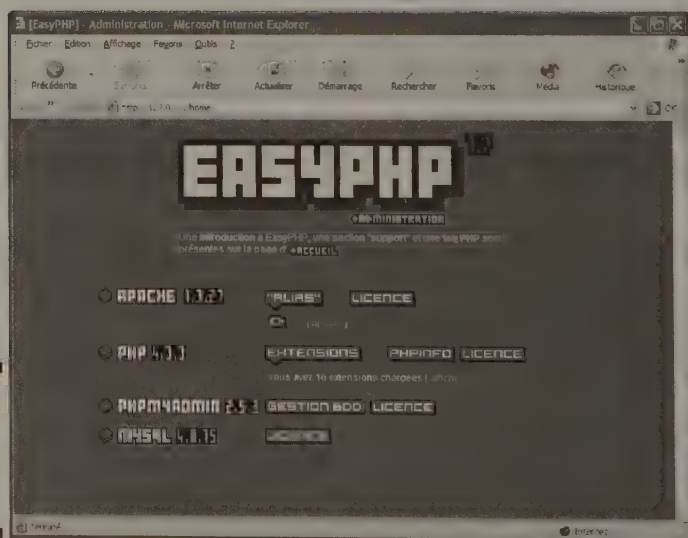
Surtout, n'oubliez pas le slash terminal afin que cette commande puisse charger le script `index.php`.



Vous pouvez aussi cliquer du bouton droit sur l'icône de EasyPHP dans le coin inférieur droit de votre écran, puis cliquer du bouton gauche sur l'entrée de menu Administration.

Vous allez voir s'afficher l'écran reproduit sur la Figure C.5.

Figure C.5 :
Ecran
d'accueil
d'administra-
tion.



Cliquez alors sur le bouton **Gestion BDD**, à droite de PHPMyAdmin. S'affiche alors l'écran d'accueil dans lequel vous choisirez votre base de données en cliquant sur la petite flèche de la boîte à liste

déroulante dans la colonne de gauche, comme vous le voyez sur la Figure C.6.

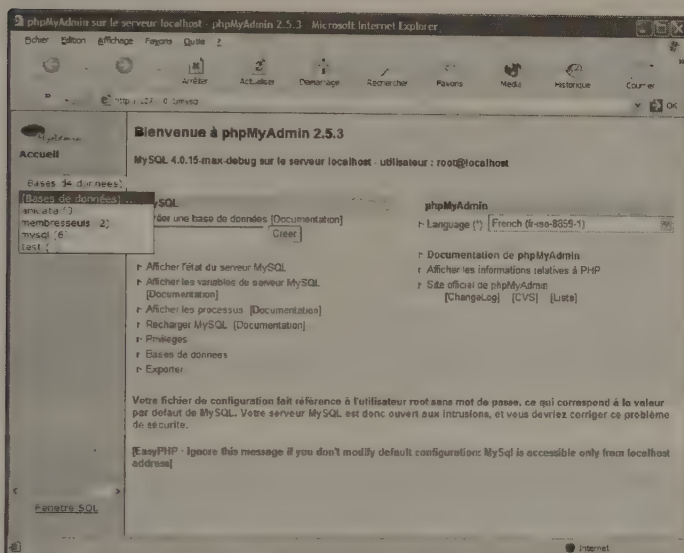


Figure C.6 :
Ecran
d'accueil de
PHPMyAdmin.

En cliquant sur la petite flèche à droite de la boîte à liste déroulante marquée Language, vous allez faire apparaître une (longue) liste de langues. Si nécessaire, choisissez évidemment French de façon que le texte qui sera affiché dans les écrans successifs soit rédigé dans votre langue habituelle.

Attention, pour que PHPMyAdmin puisse fonctionner, vous devez accepter les cookies !

Sélectionnez la base de données sur laquelle vous voulez travailler. Dans le cas de l'application de l'animalerie, cette base s'appelle AniCata. Vous voyez alors la structure de cette base avec la liste des tables qu'elle contient (Figure C.7). Dans cet écran, cliquez, par exemple, sur la ligne animal, puis sur le quatrième bouton (*propriétés*) dans la colonne Action. Vous voyez alors s'afficher la structure de la table animal telle que la montre la Figure C.8.

Pour voir le contenu actuel de cette table, vous pouvez simplement cliquer sur le lien Afficher, tout en haut de la fenêtre. Sinon, déroulez l'écran au moyen de l'ascenseur droit jusqu'à trouver le bouton Exécuter sous un champ de saisie de requête SQL (voir la Figure C.9)



Figure C.7 :
Liste des
tables de la
base de
données
AniCata.

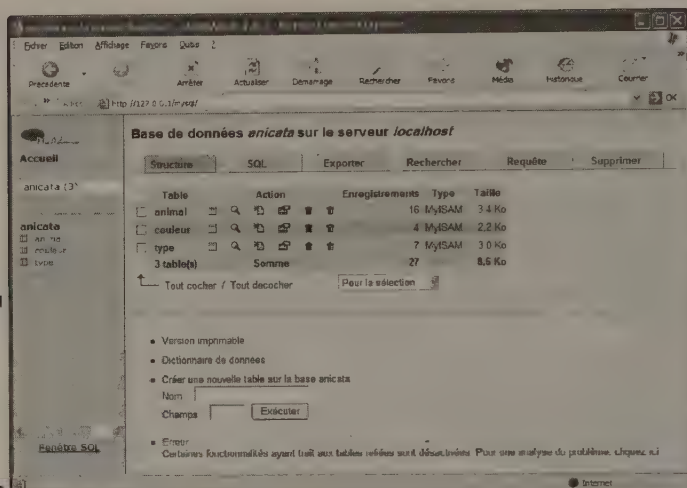
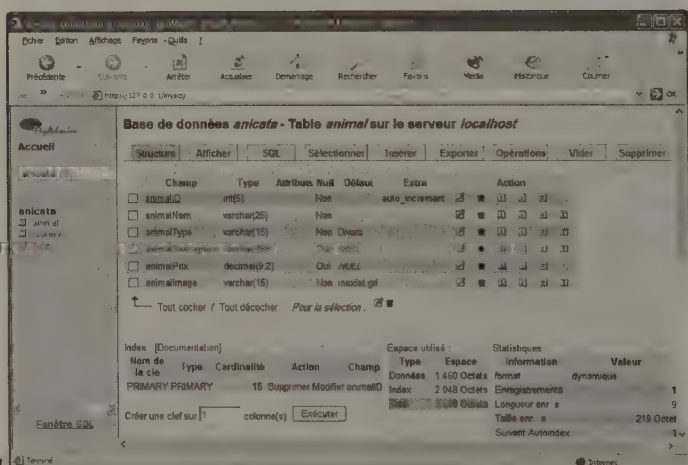


Figure C.8 :
Structure de
la table
animal de la
base de
données
AniCata.



et cliquez dessus. Vous voyez alors s'afficher toutes les lignes de la table (Figure C.10).

En cliquant sur la case Effacer (illustrée par une corbeille) devant une ligne quelconque, vous pourrez supprimer cette ligne. En cliquant sur la case Modifier (la première), une nouvelle fenêtre s'affichera dans laquelle vous pourrez modifier chacun des champs de la ligne (voir la Figure C.11). Vous pourrez ensuite sauvegarder

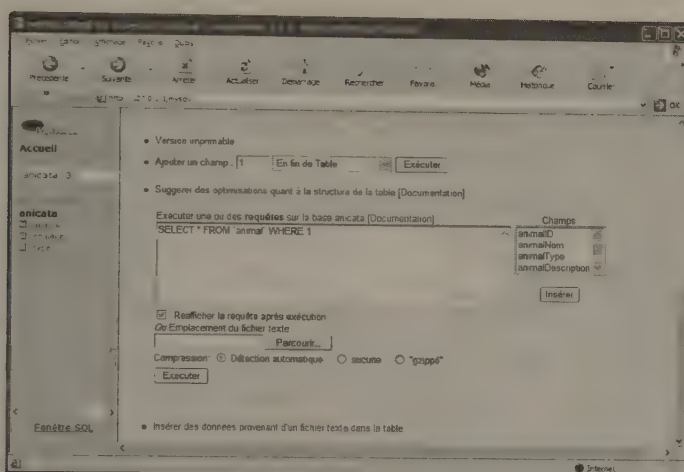


Figure C.9 :
Structure de
la table
animal.

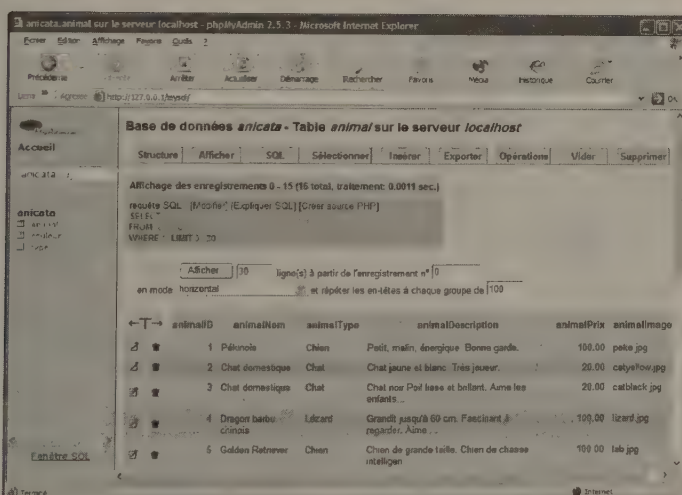
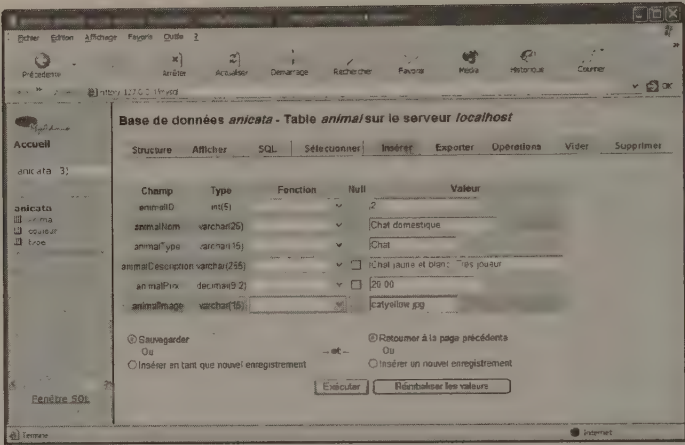


Figure C.10 :
Contenu
courant de la
table animal.

vos modifications ou les ignorer. En cas de modification, une confirmation vous sera demandée.

PHPMyAdmin possède encore bien d'autres fonctionnalités que nous vous laissons le plaisir de découvrir à l'occasion. Avec ce que nous venons de vous présenter, vous en savez déjà suffisamment pour apporter des corrections de détail aux données qui se trouvent dans vos bases de données.

Figure C.11 :
Editez les
lignes d'une
table avec
PHPMyAdmin.



Index

A

Accès
 à une page Web 38
 réservé (exemple d') 60
Accolades 127
Account 368, 373, 424
 méthodes 424
action (attribut) 248
addCart 554
addItem 547
Admin-OO.php 649
Admin.php 608
Adresse IP 16
Affectation 163
 opérateur d' 131
AFNIC 16
ALTER TABLE 79
AniCata (base de données) 49
 tables 59
Animal (table) 59
Apache 12, 22, 347, 351
 configuration 348
 configurer
 le serveur 702
 pour PHP 699
 et Linux 23
 installation 690, 693
Apostrophes 336
Appels système 326
Application 305
 modification 308
 Web 3
apxs (utilitaire) 690, 694
Arguments 166
Arobase 133
Assistance technique 18, 34

Association 154
Attaque SQL 335
Attributs 43
 définir 44
Auth-OO 377
Authentification HTTP 344
 Apache 347, 351
 PHP 352
 démarche orientée objet 360
 démarche procédurale 356
Autorisation 332
 d'accès 347
Avertissements 128, 133

B

Balises 11
Base de données 3, 5
 conception 40
 connexion 208
 construire 61, 75
 créer 75
 une table 76
 de sécurité 106
 d'utilisateurs 354
 et fonctions 222
 extraire des informations 214
 identifier les objets 44
 mise à jour 80
 modifier 79
 nommer 43
 planifier 34
 rechercher des informations 86
 système de gestion 5
 trier 89

Bloc (d'instructions) 127

Boîte

de saisie 235

noire 618

Boucles 185

infinies 193

quitter 194

Boutons

submit 261, 284

break 185, 194

C

Calendrier 314

Caractères

accentués 67

de contrôle 139

de substitution 150

littéraux 151, 323

spéciaux 323

Caractéristiques

ajout 443

panier d'achat 566

Catalog, classe 470, 539

Catalogue en ligne 447

approche orientée objet 469

base de données

accès 451

ajout de données 452, 489

conception 449

création 448, 451

catalog_index_page.inc 457

catalog_product_page.inc 460, 463

démarche procédurale 465

fields_index_page.inc 456

fields_product_page.inc 459

pages Web

affichage 464

conception 453

création 452

index 453, 455

page produits 454, 459

script 476

Chaînes de caractères 55, 137

capitalisation 67

code ASCII 148

et recherche 150

Champs vides 46

champs 42

CHAR 264

checkAddress 423

checkEmail 423

checkForBlanks 420

checkName 423

checkPhone 423

checkState 423

checkZip 423

Chemin critique 315

Classes de données élémentaire 619

classes

parent 624

super classe 624

Clé primaire 45, 77

Clients 34

se préoccuper des 37

CMS 571, 653

Admin-OO.php 649

Admin.php 608

application 571

approche orientée objet 616

base de données

accès 579

conception 573

création 572, 577

table Content 575

table Content_Download 576

table Content_Type 575

table Department 574

table Dept_User 573

code

d'affichage 639

d'identification 583

company-OO.inc 646

company.inc 601

CompanyHome-OO.php 639

CompanyHome.php 590

content_form.inc 606

démarche procédurale 583

détail de contenu 604

fields_content.inc 605, 638, 649

modèle objet 616

pages Web

conception 579

identification 637

Code

existant 660

PHP 307

réutilisable 659

- source
 - exemples 305
 - modification 308
- Commande en ligne 40
- Commentaires 156, 659, 661
- Commerce électronique
 - panier d'achat 481
- CompanyHome-OO.php 639
- CompanyHome.php 590
- Comparaison 147
- comparePassword 372
- Compte MySQL 329
 - configuration 331
- Concaténation 140
- Conception
 - décisions élémentaires 482
 - fonctionnalité 484
- Conditionnel 127
- Configuration, méthodes de 619
- Constantes 659
 - PHP 134
- Consultation, méthodes de 619
- ContentDownload, classe 633
- ContentItem, classe 626
- ContentType, classe 624
- Contenu, gestion de 571, 653
- continue 194
- Cookies 286, 290, 320, 717
 - refuser 297
- Couleur (table) 59
- CREATE TABLE 77
- createNewAccount 426
- createOrder 553
- Cryptage 339

D

- Database 364, 368, 424
 - classe 548, 636
- DATE 265
- date() (fonction) 142
- Date/heure (valeurs) 56
- Dates 141
 - et listes 241
 - et MySQL 145
 - mise en forme 142
- DATETIME 145
- DDL 578

- DECIMAL 265
- DELETE 98
- delete 632, 636
- Department, classe 622
- Développement
 - environnement de 311
 - planification 312
- die() 210
- displayAllofType 476
- displayCart 547
- displayCategories 475
- displayForm 419
- displayOrderSummary 555
- displayPage 376
- displayPrompt 363
- DISTINCT 240
- do ... while 186, 192
- Documentation 661
- Domaine 346
 - noms de 16, 19
- Données
 - externes
 - vérifier le type 321
 - filtrage 337
 - format des 264
 - nettoyer 266
 - préparer 263
 - stockage 325
 - types de 55, 57
 - vérification 337
- double_form.inc 393
- DROP DATABASE 76

E

- EasyPHP 23, 26, 710
 - installation 712
 - téléchargement 712
 - utilisation 713
- Échappement 336
- echo 160
 - instruction 126
- Éditeurs de code 309
- else 182
- elseif 182
- Email (classe) 432, 557
- En-têtes HTTP 345
- entype (attribut) 277

- Entité 43
- ENUM 265
- Énumération (valeurs) 57
- Environnement
 - de développement intégré 311
 - sécurisé 23
- ereg() (fonction) 153
- ereg_replace() 266
- Erreurs
 - gestion des 328
 - MySQL 210
- Espaces 67
 - blancs 126
- Euro 179
- Exemples de programmes 305
 - modification 308
- exit 165
- Expressions
 - rationnelles 257
 - régulières 323
- Extensions (de l'application) 39
- extract() 173, 218

F

- Factorisation 618
- Factoriser 619
- Favoris 287
- Fichier
 - de mots de passe 350
 - délimité 84
- fields_login.inc 391
- fields_products_page.inc 496
- fields_ship_info.inc 508, 511
- fields_summary.inc 515, 517
- Filtrage des données 337
- findAll 624, 626
- findByContentId 635
- findByDeptType 632
- findById 624, 626, 632, 635
- Fonctionnalité 484
- Fonctions 166
 - arguments 200
 - déclarer 197
 - noms de 166
 - renvoi de valeur 203
- for 186, 188

- foreach 174, 176, 179
- Formulaire 35
 - boutons radio 245
 - cases à cocher 247
 - champs 234, 237
 - cachés 293
 - vides 252
 - contrôler 252
 - d'expédition 493, 508
 - dynamiques 234
 - et transfert
 - de fichier 277, 278
 - d'informations 286, 292
 - listes de sélection 237
 - traiter 248
- FreeBSD 22
- FTP 15

G

- Généraliser 619
- Gestion
 - de contenu 575
 - système de 571, 653
 - des erreurs 328
- get et post (choix entre) 250
- getAllFields 420
- getAllItems 547
- getAllofType 475
- getCatalogNumber 544
- getCategoryAndTypes 475
- getConnection 367, 621
- getContentDate 632
- getContentItem 633
- getCreationDate 632
- getDescription 624
- getId 621
- getItemInfo 554
- getLastUpdate 632
- getLastUpdBy 632
- getMessage 373, 431, 547
- getName 541, 544, 621
- getOrderInfo 554
- getPrice 541, 544
- getQuantity 544
- getVariable 430
- GNU 6

GPL (Licence) 6
 GRANT 104, 109, 332
 Graphisme 38
 Guillemets 67, 138

H

Hackers 104
 header() 284
 Hébergeur 17
 Heures 141
 Homogénéité 658
 HTML 11
 htmlspecialchars() 266
 HTTP 339
 httpd.conf 699, 703
 HTTPS 339

I

IDE 311
 Identifiant 449
 Identification 384
 affichage de la page Web 400
 démarche orientée objet 412
 démarche procédurale 400
 page Web 388
 if 181
 include 306
 Incrémentation 188
 Indices 168
 Informations
 ajouter 268
 mettre à jour 273
 transférer 277
 INSERT INTO 82, 268
 Instructions
 conditionnelles 147, 181
 PHP (écriture) 126
 Intranet 572
 IP (adresse) 16
 Item, classe 542

J

JavaScript 4
 Jointures 94

L

Langage de définition de données 578
 Lerdorf (Rasmus) 9
 Liens HTML 283
 Linux 12, 21, 689
 et MySQL 679, 681
 PHP pour 690
 Listes
 de discussion 660
 de sélection 239
 Littéraux, caractères 323
 LOAD 82, 84
 localhost 31, 71
 Login 15, 31, 71
 table 60
 login 431

M

Mac OS 12
 Mac OS X
 et MySQL 683
 Macintosh 22
 mail (instruction) 18
 Membre (table) 60, 78
 MembresSeuls (base de données) 52
 tables 60
 Messages d'erreur 128
 method (attribut) 250
 Méthodes
 de configuration 619
 de consultation 619
 de factorisation 618
 de recherche 618
 statiques de recherche 618
 Microsoft IIS 12, 23
 mktime() 292

Modèle objet 616
 Mot de passe 15, 31, 71
 et sécurité 103
 MySQL 329
 my.ini 687
 MySQL 5
 accès par script PHP 334
 autorisation 332
 avantages 7
 client 73
 communiquer avec 65
 compte 17, 31, 32, 102, 329
 configuration 331
 créer un 105
 configurer 686
 et fichiers texte 82
 installa-
 tion 24, 671, 672, 680, 681, 684
 liste des comptes 107
 mot de passe 329
 nouveaux comptes 108
 permission 332
 privilège 332
 sauvegarde des bases 338
 sécurité 329, 330
 serveur 8
 téléchargement 672
 test de 30
 MySQL AB 7
 mysql_close() 209
 mysql_connect() 208
 mysql_envoi.php 68
 mysql_fetch_array() 218
 mysql_query() 213
 mysql_select_db() 212
 mysqldump 113

N

Navigateurs 38
 Navigation 38
 Nombres
 formater 137
 Noms
 de comptes 102
 d'hôte 102
 HTML par défaut 17
 validité des 257

Numériques (valeurs) 56
 Numéros de téléphone 265

O

Objectifs (d'une application) 34
 OCR 82
 Opérateurs
 de comparaison 148
 priorité 135
 ORDER 241
 Order, classe 549
 Ordinateur (installation d'un) 20
 Orientée objet, programmation 307

P

Pac OS X
 PHP pour 693
 Pages Web dynamiques 4
 Panier d'achat 481
 ajout de caractéristiques 566
 approche orientée objet 538
 base de données
 accès 489
 conception 485
 création 485, 487
 table Customer_Order 485
 table Food 487
 table Order_Item 487
 démarche procédurale 520
 fields_products_page.inc 496
 fields_ship_info.inc 508, 511
 fields_summary.inc 515, 517
 pages Web
 conception 490
 confirmation 495
 création 490
 formulaire d'expédition 493, 508
 informations produits 491, 495
 méthode orientée objet 505
 méthode procédurale 500
 panier 492, 500
 récapitulatif 494, 515
 ProcessOrder.php 530

script 557
 ShopCatalog.php 521
 shopping_product_page.inc 497, 499
 ShoppingCart.php 526
 single_form.inc 512
 summary_page.inc 518
 parent, classe 624
 PasswordPrompter 362, 363
 Permission 332
 PHP 9, 307
 avantages 10
 avec MySQL 710
 configurer pour MySQL 705
 écriture des programmes 61
 erreurs 128
 et casse des caractères 128
 et HTML 123
 et nombres 135
 fichiers 17
 fonctions 196
 formulaires HTML 228
 indenter les instructions 127
 installation 26, 689, 692, 695
 instructions 11, 159
 interpréteur 124
 opérateurs 135
 options d'installation 697
 téléchargement 691, 694, 700
 test de 28
 version 18, 233
 php.ini 129, 233, 294, 705
 php.ini-dist 705
 PHPMyAdmin 716
 PHPSESSID 294
 PKG 683
 Planification 34, 312
 écrire sur papier 39
 planification 657
 print_r 169
 privilège 332
 Procédurale, programmation 307
 ProcessOrder.php 530
 Profils de recherche 258
 Programmation
 orientée objet 307
 procédurale 307
 protected 620
 Protégées, propriétés 620

R

Récapitulatif 494, 515
 Recherche, méthodes de 618
 Relations (entre tables) 46
 Remarques 128
 Répertoire include 306
 Requêtes 336
 attaque SQL 335
 fichier protégé 346
 GRANT 332
 SQL 213
 Ressources 315
 Restaurations 115
 return 197, 203
 Risques de sécurité 318
 root 71
 RPM 671, 679, 690

S

Sauvegardes 19, 112
 bases de données 338
 save 632, 636
 Script (langage de) 9, 11
 Sécurité 317, 318
 authentification HTTP 344
 Apache 347, 351
 cryptage 339
 MySQL 329, 330
 protection de page Web 411, 442
 risques 318
 scripts PHP 319
 serveur Web sécurisé 339
 source externe 319
 sécurité 317
 SELECT 83, 86, 87, 108, 215
 et fonctions 88
 selectAccount 372
 selectCatalog 474
 selectOrder 553
 sendEmail 434
 Séparation
 mise en page et fonctions 659
 Serveur
 et code PHP 124
 MySQL 676, 680, 683, 685

- Web 13 702
 - Apache 347, 351
 - sécurisé 339
- Session 286, 293, 413, 428
 - fermer 301
 - ouvrir 295
 - privées 301
 - sans cookies 298
- session.save_path 294
- session_destroy() 302
- session_id() 300
- session_start() 295
- setAddr 434
- setContentDate 632
- setcookie() 291
- setCreationDate 632
- setDescriptionn 632
- setFieldsNotRequired 419
- setLastUpdBy 632
- setMessage 434
- setName 632
- setSubj 434
- SGBD 5
- SGBDR 5
- ShopCatalog.php 521
- shopping_product_page.inc 497, 499
- ShoppingCart, classe 544
- ShoppingCart.php 526
- SHOW DATABASES 76
- SID 300
- single_form.inc 512
- Site Web 14
- Solaris 22
- sort() 170
- Source externe 319
 - identifier 319
 - nettoyage des données 322
 - source des données 320
 - vérifier le type 321
- Sous-programmes 166
- Spam 18
- SQL 9
 - requêtes 66
- SSL 23
- Statiques, méthodes 618
- storeVariable 430
- strip_tags() 266
- summary_page.inc 518
- Super classe 624
- switch 184

- Système
 - appels 326
 - de gestion de contenu 571, 653

T

- Tableau 167
 - afficher 168
 - dimensions 177
 - extraire des valeurs 172
 - fonctions de tri 171
 - parcourir 174
 - suppression d'éléments 169
 - trier 170
- Tables 42
 - ajouter une ligne 82
 - colonnes 45
 - combinaison de 92
 - définir 44
 - mettre à jour 98
 - modifier 79
 - supprimer des informations 98
- Tabulations 139
- tarball 681
- test.php 29
- Tests 658
- time() 292
- timestamp (UNIX) 141, 144
- track-vars 294
- trans-sid 294, 298, 299, 300
- trim() 267
- Type (table) 59

U

- Union 93
- UNIX 7, 12, 22
- unset() 302
- UPDATE 98, 273
- updateCart 547
- updateOrderInfo 554
- URL
 - et transfert d'informations 285, 286
 - relative 299
- useDatabase 367
- Utilisateurs malveillants 256

V

Valeurs par défaut 45
Variables
 de session 295
 et données 264
 et fonctions 198
 et formulaires 235
 globales 199
 incrémenter et décrémenter 164
 noms de 130
 PHP 130
Vente en ligne (exemple de) 59
Vérification des données 337
verifyData 421

W

WebForm 413
 classe 555, 636
WebPage 374, 376
 classe 556
WHERE (expressions utilisables
 dans) 89
while 186, 189
whois 16
Windows 12, 21
 et MySQL 671
 PHP pour 700
WinMySQLAdmin 671

Mégapoche c'est 2 fois plus d'infos à prix serré !



Pour comprendre enfin quelque chose à la micro-informatique !

Un nouveau matériel ou un nouveau logiciel vient de débarquer dans votre vie et vous n'avez pas de temps à perdre pour en apprendre l'utilisation. Deux solutions s'offrent à vous, attendre un miracle, solution peu probable, ou faire confiance à votre Mégapoche qui vous donnera toutes les informations essentielles pour démarrer un apprentissage efficace dans la joie et la bonne humeur !

Motorisez tous vos sites Web avec PHP et MySQL !

Ce livre de la collection « Mégapoche Pour les Nuls » va vous donner tous les outils qui vous permettront de développer des sites Web à base de PHP. Vous apprendrez à manipuler tous les outils de gestion de sessions, les cookies, gérer le code XML, mettre en place des systèmes de sécurité, et bien d'autres choses encore. Une partie intitulée « PHP au quotidien » vous proposera des applications prêtes à l'emploi qui vous éviteront des heures de développement.

L'informatique en français dans le texte.

Tout et seulement tout ce que vous devez savoir.

Un accès rapide à l'information grâce à un système d'icônes d'aide à la navigation.

Les dix commandements.

Une bonne dose d'humour.

**L'ESPRIT
DES NULS**

FIRST
Interactive
www.efirst.com

Rayon librairie : Informatique / PHP & MySQL

Public : Débutant / Initié

DÉCOUVREZ

Création d'un environnement de développement

Développer une base de données MySQL

Gestion des données et mise à jour des pages Web avec PHP

Utiliser un serveur Web sécurisé

Créer une application de vente en ligne

SUIVEZ LE GUIDE



Cette icône signale une manipulation vous simplifie la tâche.



Cette icône indique un danger, indique généralement ce qu'il ne faut pas faire.



P9-BAG-323

65 1496 2 V-07 16,
ISBN-978-2-7540-0411



9 782754 004114