13.4. Les interfaces

Certains langages de programmation permettent à une classe d'hériter de plusieurs classes mères. Les concepteurs de PHP n'ont pas souhaité mettre œuvre cet héritage multiple et ont préféré travailler sur la notion d'interface.

Une interface correspond à un ensemble de définitions de méthodes qu'une classe devra surcharger si elle souhaite l'implémenter. Une classe peut implémenter plusieurs interfaces.

La définition d'une interface utilise le mot-clé `interface` suivi d'un bloc contenant la définition des méthodes.

Listing 13-26 : Définition d'une interface

```
interface MonInterface {
  public function methode1 ();
  public function methode2 ();
  // etc.
}
```

Une classe implémente une interface en utilisant le mot-clé `implements`.

Listing 13-27 : Chaque classe doit implémenter la méthode aCotesEgaux()

```
interface ObjetGeom {
  public function aCotesEgaux ();
}

class Rectangle implements ObjetGeom{

  private $longueur = null;
  private $largeur = null;

  function __construct($longueur,$largeur) {
    $this->longueur = $longueur;
    $this->largeur = $largeur;
  }

  function aCotesEgaux () {
    if ($this->largeur==$this->longueur) {
      return true;
    }
    return false;
  }

}
```

```
class Carre extends Rectangle {

  public $cote = null;

  function __construct($cote) {
    parent::__construct($cote,$cote);
    $this->cote = $cote;
  }

  function aCotesEgaux () {
    return true;
  }

}
```

Toutes les méthodes des interfaces doivent être implémentées par la classe. Une erreur est déclenchée dans le cas contraire.

Listing 13-28 : La classe Carre n'implémente pas getNbCotes()

```
interface ObjetGeom {
  public function aCotesEgaux ();
  public function getNbCotes ();
}

class Carre implements ObjetGeom {

  public $cote = null;

  function __construct($cote) {
    parent::__construct($cote,$cote);
    $this->cote = $cote;
  }

  function aCotesEgaux () {
    return true;
  }

}
```

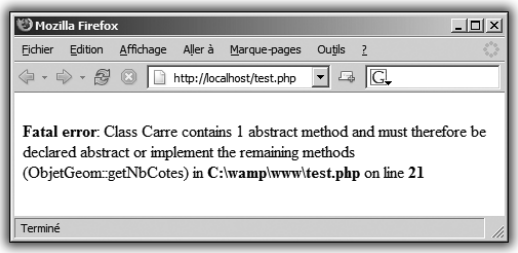

Figure 13.10 : *Affichage du résultat*

Une interface peut être assimilée à un contrat que doit respecter la classe qui l'implémente.

13.5. Itérateurs

La boucle `foreach()` peut être utilisée sur un objet afin d'afficher ses différents attributs.

```
$rect = new Rectangle(3,5);
foreach ($rect as $attr => $val) {
  print("$attr = $val<br/>");
}
```

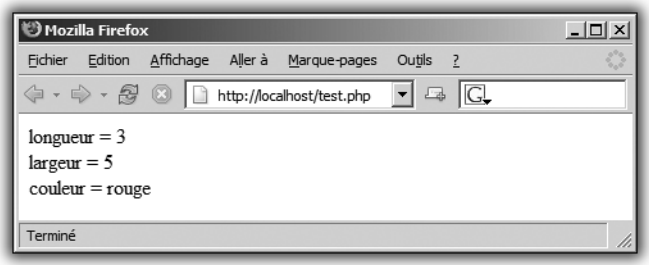

Figure 13.11 : *Affichage des différents attributs de l'objet $rect avec la boucle foreach()*

Le comportement par défaut consiste à passer, à chaque itération de la boucle, d'un attribut `visible` à l'autre.

PHP vous permet d'implémenter l'interface `Iterator` afin de définir vous-même le comportement de la boucle `foreach()` :

Tableau 13.1 : *Différentes fonctions devant être implémentées pour définir votre propre itérateur.*

Méthode	Rôle
`rewind()`	Revient au début de la liste
`current()`	Valeur de l'élément en cours
`key()`	Valeur de la clé de l'élément en cours
`next()`	Passer à l'élément suivant
`valid()`	Retourne `false` lorsque vous êtes à la fin de la ligne (`true` sinon)

L'exemple suivant permet d'itérer parmi les différentes valeurs de l'attribut `$tab` plutôt que parmi les attributs de l'objet.

Listing 13-29 : Définition de votre propre itérateur

```
class Rectangle implements Iterator{

  public $longueur = null;
  public $largeur = null;
  public $couleur = "rouge";
  public $tab = array("mm","cm","m");
  public $i = 0;

  function __construct($longueur,$largeur) {
    $this->longueur = $longueur;
    $this->largeur = $largeur;
  }

  public function rewind() {
    print("rewind<br/>");
    $this->i = 0;
  }

  public function current() {
    print("current<br/>");
    return ($this->tab[$this->i]);
  }

  public function key() {
    print("key<br/>");
    return ($this->i);
  }

  public function next() {
    print("next<br/>");
    $this->i++;
  }

  public function valid() {
    print("valid<br/>");
    if ($this->i<count($this->tab)) return true;
    return false;
  }

}

$rect = new Rectangle(3,5);

foreach ($rect as $attr => $val) {
  print("<b>$attr = $val</b><br/>");
}
```

Figure 13.12 : *Trace des différents appels de PHP lors de la mise en œuvre d'un itérateur avec la boucle foreach*

13.6. Exceptions

La gestion d'erreurs en POO fait intervenir une notion supplémentaire : les exceptions. Ces dernières correspondent à des signaux pouvant être émis par une méthode et capturés ailleurs dans le code.

Principe général

L'émission d'une exception utilise le mot-clé `throw`.

Listing 13-30 : La méthode test() émet une exception lorsque $n est null

```
class MonObjet {
  function test($n=null) {
    if ($n==null) throw new Exception("La valeur ne
    ✂ convient pas");
    return true;
  }
}
```

Mot clé `throw`
L'émission d'une exception termine la méthode à la manière d'un `return ()`.

La capture d'une exception fait intervenir la structure `try { … } catch () { … }`.

Le bloc `try` permet d'englober l'ensemble des appels aux méthodes susceptibles d'émettre des exceptions. Le bloc `catch()` se charge quant à lui de recevoir l'ensemble des exceptions émises par ces méthodes. Un bloc `try` doit obligatoirement être suivi d'un bloc `catch()`.

Listing 13-31 : Contrôle d'un code susceptible d'émettre une exception

```
$obj = new MonObjet();
try {
  // utilisation d'une méthode susceptible
  // d'émettre une exception
  $obj->test();
}
catch (Exception $e) {
  // traitement de l'exception
  print($e->getMessage());
}
```

De la bonne utilisation des exceptions
Les exceptions ne doivent en aucun cas remplacer une gestion standard d'erreur. Elles ne doivent être utilisées que pour signaler des situations exceptionnelles.

L'exemple suivant montre comment émettre une exception quand la méthode `surface()` est appelée sur un rectangle invalide.

Listing 13-32 : Gestion d'une exception dans la méthode surface()

```
class Rectangle {

  public $longueur = null;
  public $largeur = null;

  function __construct($longueur,$largeur) {
    $this->longueur = $longueur;
    $this->largeur = $largeur;
  }
```

```
  function surface() {
    if ($this->longueur<1 ||
        $this->largeur<1) {
      throw new Exception("Valeurs incohérentes");
    }
    return ($this->longueur*$this->largeur);
  }

}

$rect = new Rectangle(-1,4);
try {
  echo $rect->surface();
}
catch (Exception $e) {
  print("<hr/>".$e->getMessage()."<hr/>");
}
```

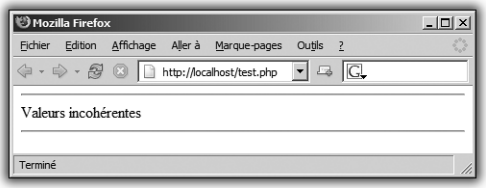

Figure 13.13 : *Affichage de l'exception*

La classe Exception

Une exception correspond à une instance de la classe `Exception`. Chaque nouvelle instance peut être initialisée avec un ou deux arguments. Le premier argument correspond au texte du message d'erreur et le second, optionnel, au code d'erreur.

Cette classe dispose d'un certain nombre de méthodes pouvant être très utiles dans une phase de débogage.

Tableau 13.2 : *Méthodes proposées par la classe Exception*

Méthode	Rôle
`getMessage()`	Retourne la valeur du message
`getCode()`	Retourne la valeur du code d'erreur
`getFile()`	Retourne le nom du fichier d'où l'exception a été émise

Tableau 13.2 : *Méthodes proposées par la classe Exception*	
Méthode	**Rôle**
`getLine()`	Retourne le numéro de la ligne où l'exception a été émise
`getTrace()`	Retourne un tableau de la trace d'erreur
`getTraceAsString()`	Retourne une chaîne de caractères correspondant à la trace de l'erreur

Listing 13-33 : Utilisation de toutes les méthodes proposées par la classe Exception

```
$rect = new Rectangle(-1,4);
try {
  echo $rect->surface();
}
catch (Exception $e) {
  print("<hr/>".
    "<i>Message</i> : ".$e->getMessage()."<br/>".
    "<i>Code</i> : ".$e->getCode()."<br/>".
    "<i>File</i> : ".$e->getFile()."<br/>".
    "<i>Line</i> : ".$e->getLine()."<br/>".
    "<i>TraceAsString</i> : <br/><pre>"
    .$e->getTraceAsString()."</pre>".
    "<hr/>");
}
```

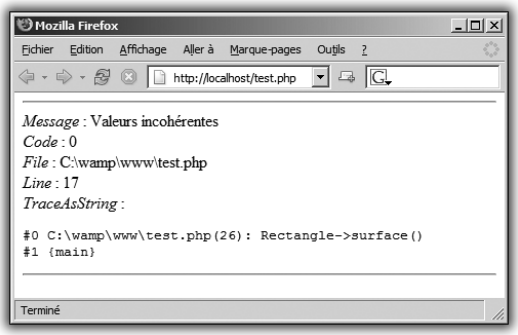

Figure 13.14 : *Affichage des valeurs retournées par les différentes méthodes*

Cette classe peut tout à fait être étendue de façon à surcharger ses méthodes et attributs ou à les compléter avec les vôtres.

En définissant vos propres exceptions, vous vous donnez également la possibilité d'en capturer plusieurs émises dans un même bloc `try`.

```
try {
  // code susceptible d'émettre plusieurs exceptions
} catch (MonException $monexception) {
  …
} catch (Exception $exception) {
  …
}
```

Constructeur et exceptions

Un constructeur ne pouvant retourner de valeur, les exceptions sont la seule solution pour avertir d'une erreur.

13.7. Réflexion

PHP vous donne désormais la possibilité d'obtenir des informations sur la structure interne d'une classe. La classe mise en œuvre porte le nom `Reflection`.

```
<?php
class Rectangle {

  public $longueur = null;
  public $largeur = null;
  public $couleur = "rouge";
  const NOM = "Super Rectangle";

  function __construct($longueur,$largeur) {
    print("Hello ".self::NOM."<br/>");
    $this->longueur = $longueur;
    $this->largeur = $largeur;
  }

  function perimetre() {
    if ($this->longueur!=null &&
        $this->largeur!=null) {
      return (2*$this->longueur+2*$this->largeur);
    }
  }

  function surface() {
    if ($this->longueur!=null &&
        $this->largeur!=null) {
      return ($this->longueur*$this->largeur);
    }
  }

}
```

```
print("<pre>");
Reflection::export(new ReflectionClass('Rectangle'));
```

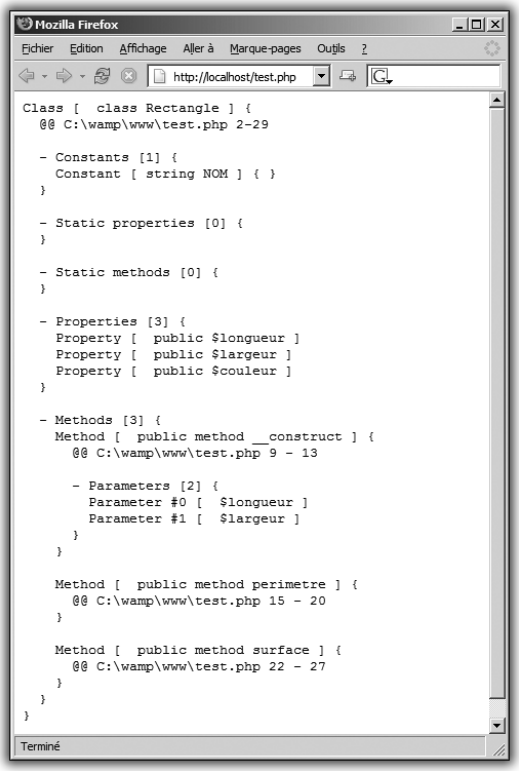

Figure 13.15 : *Affichage de la structure interne de la classe Rectangle*

Cette fonctionnalité est particulièrement intéressante pour la création automatique de documentation et pour obtenir des informations sur une classe des sources et de l'API de laquelle vous ne disposez pas.

13.8. Version objet de la génération de graphique

Le script de génération de graphique étudié dans le chapitre précédent souffre de nombreux inconvénients :

- la récupération des données n'est pas clairement séparée de l'affichage graphique ;

- les données liées à la présentation (couleurs, polices) ne sautent pas aux yeux ;
- le script n'est pas réutilisable sans un changement important du code.

Utilisez le concept de classe pour remédier à ces faiblesses. Voyez les attributs et les méthodes dont vous avez besoin.

- les attributs : toutes les couleurs, le nom de la fonte, le tableau contenant les moyennes, l'identifiant de l'image ;
- les méthodes : `__construct()`, le constructeur pour initialiser la taille de l'image et les couleurs, `enregistre_donnees()` qui permet de transmettre les données, `cree_image()` qui contient tout le code, `genere_image()` qui génère le contenu de l'image.

Il suffit ensuite de placer le code déjà écrit dans les bonnes méthodes et de passer par les attributs pour accéder aux variables :

Listing 13-34 : Le fichier class_graph.inc.php

```
<?php

class Graph {

  public $coul_fond;
  public $coul_axes;
  public $coul_lignes;
  public $coul_legendes;
  public $coul_barres;
  public $coul_orange;
  public $image;
  public $font = "verdana.ttf";
  public $abs = "trimestre";
  public $ord = "moyenne";
  public $donnees;

  public function __construct ($largeur, $hauteur,
                               $font = "arial.ttf") {
    $this->image =  Imagecreate ($largeur, $hauteur);
    if ($this->image===false) throw new Exception ();
    $this->coul_fond = ImageColorAllocate ($this->image,
                                           208, 216, 213);
    $this->coul_axes = ImageColorAllocate ($this->image,
                                           11, 62, 43);
    $this->coul_lignes = ImageColorAllocate ($this->image,
                                            227, 235, 232);
    $this->coul_legendes = ImageColorAllocate ($this->image,
                                              11, 62, 43);
```

```
    $this->coul_barres = ImageColorAllocate ($this->image,
                                             42, 124, 94);
    $this->coul_orange = ImageColorAllocate ($this->image,
                                             207, 140, 53);
    $this->font = $font;
  }

  public function enregistre_donnees ($datas) {
    $this->donnees = $datas;
  }

  public function cree_image () {
    imageline ($this->image,30,30,30,190,$this->coul_axes);
    imageline ($this->image,30,190,320,190,$this->coul _axes);

    $tab_fleche_ord = array (30, 30, 26, 34, 34, 34);
    $tab_fleche_abs = array (320, 190, 316, 186, 316, 194);

    imagefilledpolygon ($this->image, $tab_fleche_ord, 3,
                        $this->coul_axes);
    imagefilledpolygon ($this->image, $tab_fleche_abs, 3,
                        $this->coul_axes);

    ImageTTFText ($this->image,10,0,5,20,
                  $this->coul_legendes,
                  $this->font,$this->ord);
    ImageTTFText ($this->image,10,0,280,180,
                  $this->coul_legendes,
                  $this->font,$this->abs);

    imageline ($this->image,26,190,30,190,$this->coul_axes);
    imageline ($this->image,26,155,30,155,$this->coul_axes);
    imageline ($this->image,26,120,30,120,$this->coul_axes);
    imageline ($this->image,26,85,30,85,$this->coul_axes);
    imageline ($this->image,26,50,30,50,$this->coul_axes);

    ImageTTFText ($this->image,8,0,6,190,
                  $this->coul_legendes, $this->font,"0");
    ImageTTFText ($this->image,8,0,6,155,
                  $this->coul_legendes, $this->font,"5");
    ImageTTFText ($this->image,8,0,6,120,
                  $this->coul_legendes, $this->font,"10");
    ImageTTFText ($this->image,8,0,6,85,
                  $this->coul_legendes, $this->font,"15");
    ImageTTFText ($this->image,8,0,6,50,
                  $this->coul_legendes, $this->font,"20");

    imageline ($this->image,31,155,320,155,
               $this->coul_lignes);
    imageline ($this->image,31,120,320,120,
               $this->coul_lignes);
```

```
    imageline ($this->image,31,85,320,85,
               $this->coul_lignes);
    imageline ($this->image,31,50,320,50,
               $this->coul_lignes);

    imagefilledrectangle ($this->image, 40,
                          (20-$this->donnees[0])*7+50, 110,
                          189, $this->coul_barres);
    imagefilledrectangle ($this->image, 120,
                          (20-$this->donnees[1])*7+50, 190,
                          189, $this->coul_barres);
    imagefilledrectangle ($this->image, 200,
                          (20-$this->donnees[2])*7+50, 270,
                          189, $this->coul_barres);

    ImageTTFText ($this->image,10,0,50,180,$this->coul_orange,
                  $this->font,$this->donnees[0]);
    ImageTTFText ($this->image,10,0,130,180,$this->coul_orange,
                  $this->font,$this->donnees[1]);
    ImageTTFText ($this->image,10,0,210,180,$this->coul_orange,
                  $this->font,$this->donnees[2]);
  }

  public function genere_image () {
    header ("Content-type: image/png");
    ImagePng ($this->image);
  }
}

?>
```

Deux remarques peuvent être faites sur le code…

- observez la signature assez étrange du constructeur : `function __construct($largeur,$hauteur,$font = "arial.ttf")`. Cela signifie que l'on peut créer un objet `class` de deux manières :

```
$mongraph = new Graph (340,220);
```

Dans ce cas, la variable `font` prend la valeur `"arial.ttf"`.

```
$mongraph = new Graph (340,220,"verdana.ttf");
```

Dans ce cas, l'attribut `font` est initialisé à `"verdana.ttf"`.

Pour obtenir ce résultat, vous pouvez aussi écrire :

```
$mongraph = new Graph (340,220);
$mongraph->font = "verdana.ttf";
```

- il n'est pas nécessaire que les variables `$largeur` et `$hauteur` existent en tant qu'attributs car vous n'en avez besoin que dans le constructeur.

Grâce à cette classe, le script *graph.php* se simplifie énormément :

Listing 13-35 : Le script graph.php

```
include("variables.inc.php");

include("class_graph.inc.php");

$liendb = mysql_connect($bddserver, $bddlogin,
                        $bddpassword);
mysql_select_db ($bdd);
$sql = "SELECT * FROM $table_exam";
$resultat = mysql_query ($sql);
$i = 0;
while ($tmp = mysql_fetch_array ($resultat)) {
  $tab[$tmp['trimestre'] - 1] += $tmp['moyenne'];
  $i++;
}
mysql_close($liendb);

$nb_eleves = $i / 3;

for ($i = 0; $i < 3; $i++) {
  $tab[$i] = number_format($tab[$i] / $nb_eleves,2);
}


try {
  $mongraph = new Graph(340,220);
  $mongraph->enregistre_donnees($tab);
  $mongraph->cree_image();
  $mongraph->genere_image();
}
catch (Exception $e) { die ('erreur'); }
```

Le script *class_graph.inc.php*

N'oubliez pas d'inclure le fichier *class_graph.inc.php* sous peine d'erreur !

Vous voyez que ce nouveau code présente l'avantage d'être beaucoup plus clair. Les parties acquisition des données et affichage de l'image sont désormais clairement séparées et le code est parfaitement factorisé.

Créez maintenant un fichier *graph_eleve.php* qui permettra d'afficher l'évolution des notes d'un élève :

Listing 13-36 : Le fichier graph_eleve.php

```
include("variables.inc.php");
include("class_graph.inc.php");

$liendb = mysql_connect($bddserver, $bddlogin,
                        $bddpassword);
mysql_select_db ($bdd);

$sql = "SELECT * FROM $table_exam ".
       "WHERE ideleve = '".$_REQUEST['id']."'";

$resultat = mysql_query ($sql);

while ($tmp = mysql_fetch_array ($resultat)) {
  $tab[$tmp['trimestre'] - 1] = $tmp['moyenne'];
}

mysql_close($liendb);

try {
  $mongraph = new Graph (340,220,"verdana.ttf");
  $mongraph->enregistre_donnees($tab);
  $mongraph->ord = "moyenne de l'élève";
  $mongraph->cree_image();
  $mongraph->genere_image();
}
catch (Exception $e) { die ('erreur'); }
```

Dans ce script, vous utilisez la fonte `verdana.ttf` plutôt que la fonte `arial.ttf`. De plus, vous changez l'intitulé de l'ordonnée en `"moyenne de l'élève"`. C'est possible car vous avez fait en sorte de placer de nombreuses données en attributs. Vous voyez également l'intérêt de diviser la génération d'images en plusieurs méthodes et de ne pas avoir l'intégralité du code dans le constructeur.

Envoi des fontes

N'oubliez pas d'envoyer sur votre compte les fontes que vous utilisez dans vos scripts.

Incluez la balise suivante en bas du script *eleve_edite.php* :

```
<img src="graph_eleve.php?id=<?php echo $id; ?>" />
```

Vous obtenez le résultat suivant :

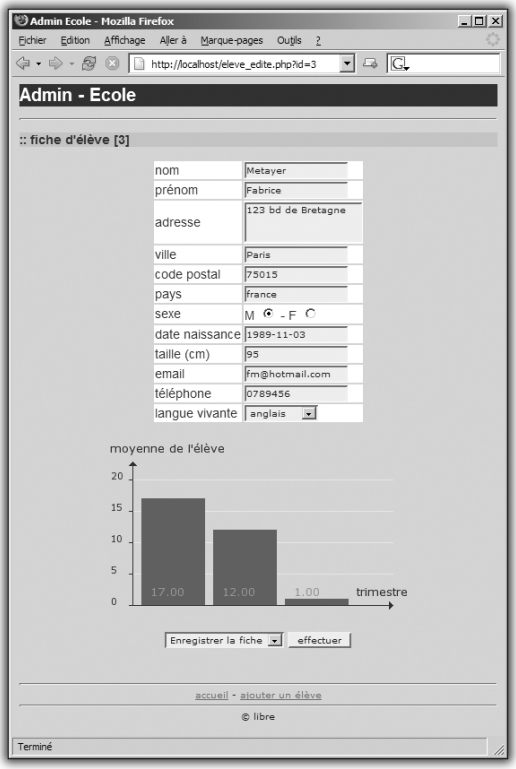

Figure 13.16 :
Une nouvelle version de la fiche élève

13.9. Check-list

- La dimension objet de PHP a largement évolué avec la version 5.
- Une classe peut être considérée comme une entité regroupant les données et les fonctions qui lui sont propres.
- Les classes, par leur nature, sont particulièrement faciles à inclure dans une application. C'est aujourd'hui principalement sous cette forme que l'on trouve le code PHP sur le Web.
- Les techniques permettant de mieux organiser votre code sont au nombre de trois : les `include()`, les fonctions et les classes.

Chapitre 14

XML

Peu de formats de documents ont autant influencé l'industrie informatique que le XML (*Extensible Markup Language*). Datant du milieu des années 1990, ce format est aujourd'hui la référence pour les échanges de données notamment sur Internet.

Les avantages de ce format sont nombreux :

- Un document XML est lisible, structuré et compréhensible.
- Le format respecte un standard et assure, par là même, une véritable portabilité et pérennité des données.
- Tous les langages de programmation proposent désormais des bibliothèques permettant un accès facile à ce format.

14.1. Le format

À la manière du HTML, qui comme lui est issu du SGML, le format XML s'organise avec des balises. Cependant, à la différence de HTML, les balises ne sont pas prédéfinies et peuvent être choisies en fonction des besoins de l'application. Le document suivant par exemple représente une classe :

```
<classe>
 <eleve>Paul Dupont</eleve>
 <eleve>Eric Durant</eleve>
</classe>
```

La hiérarchie saute aux yeux : une classe est composée d'élèves. Il est également possible de dire que l'élément `<classe>` dispose de plusieurs enfants : les éléments `<eleve>`.

Cette hiérarchie peut être affinée si le besoin se fait ressentir. Les nom et prénom d'un élève peuvent faire l'objet d'éléments spécifiques :

```
<classe>
 <eleve>
  <nom>Dupont</nom>
  <prenom>Paul</prenom>
 </eleve>
 <eleve>
  <nom>Durant</nom>
  <prenom>Eric</prenom>
 </eleve>
</classe>
```

Commentaires
Tout comme pour l'HTML les commentaires sont entourés des balises <!— et —>.

Tout comme les balises HTML, les balises XML peuvent également disposer d'attributs. Ces derniers sont, eux aussi, librement définissables. Le niveau de la classe peut être précisé de la manière suivante :

```
<classe niveau="cp">
 <eleve>
  <nom>Dupont</nom>
  <prenom>Paul</prenom>
 </eleve>
 <eleve>
  <nom>Durant</nom>
  <prenom>Eric</prenom>
 </eleve>
</classe>
```

Les documents XML précédents sont pour l'instant incorrects. La première ligne d'un document XML doit en effet préciser deux informations :

- la version de la norme XML ;
- l'encodage des données.

Cette ligne prend la forme suivante :

```
<?xml version="1.0" encoding="UTF-8"?>
```

Cette gestion de l'encodage permet à XML de contenir des données écrites dans toutes les langues. L'encodage UTF-8 notamment pourrait être assimilé à un alphabet universel des caractères de toutes les langues mondiales. Pour des langues européennes, l'encodage ISO-8859-1 peut suffire. Cet encodage permet en effet à un document XML de contenir des données accentuées.

D'autres contraintes doivent être respectées pour obtenir un document valide :

- Le document ne doit contenir qu'une racine (dans cet exemple, il s'agit de `<classe>`). Un document XML dispose d'une véritable structure d'arbre composée d'éléments (nœuds) pouvant avoir des enfants.
- Les balises orphelines doivent contenir une barre oblique (/) avant le chevron final > (par exemple `<voyages />`).

- Tous les attributs doivent être entourés de guillemets ou de primes (", ').
- La casse doit être respectée entre une ouverture et une fermeture de balise.
- Certains caractères doivent être convertis au sein du contenu d'un élément (<, >, ', ", & sont remplacés respectivement par <, >, ', ", &). La fonction PHP `htmlspecialchars()` permet de réaliser cette conversion.
- Les balises d'ouverture et de fermeture ne peuvent s'entremêler (exemple qui ne fonctionne pas : `<eleve><nom>Dupont</eleve></nom>`).

La fonction `header()` est une nouvelle fois nécessaire afin d'indiquer au navigateur que le document reçu est de type XML et qu'il doit par conséquent l'afficher dans une « vue » adaptée. Le script suivant permet d'obtenir les élèves dans le cadre d'un document XML.

Listing 14-1 : Génération d'un document XML

```
include("variables.inc.php");

$liendb = mysql_connect($bddserver, $bddlogin,
✂ $bddpassword);
mysql_select_db ($bdd);
$sql = "SELECT * FROM $table_eleve";
$resultat = mysql_query ($sql);

header('Content-Type: application/xml');
$xml = '<?xml version="1.0" encoding="ISO-8859-1"?>';
$xml .= "\n<classe>\n";
while ($eleve = mysql_fetch_array ($resultat)) {
  foreach ($eleve as &$valeur) {
    $value = htmlspecialchars($value);
  }
  $xml .= " <eleve id='".$eleve['ideleve']."'>\n";
  $xml .= "  <nom>".$eleve['nom']."</nom>\n";
  $xml .= "  <prenom>".$eleve['prenom']."</prenom>\n";
  $xml .= "  <adresse>".$eleve['adresse']."</adresse>\n";
  $xml .= "  <ville>".$eleve['ville']."</ville>\n";
  $xml .= "  <cp>".$eleve['cp']."</cp>\n";
  $xml .= "  <pays>".$eleve['pays']."</pays>\n";
  $xml .= " </eleve>\n";
}
mysql_close($liendb);
$xml .= "</classe>\n";

print($xml);
```

Figure 14.1 : *Affichage des données en mode XML dans Firefox*

CDATA

Il est possible de définir un contenu d'élément sans se préoccuper des contraintes décrites plus haut. La section d'échappement spéciale suivante `<![CDATA[ ]]>` permet d'indiquer que tout ce qui est placé entre `<![CDATA[` et `]]>` ne doit pas être interprété mais considéré comme du texte brut.

L'écriture suivante devient donc possible

```
<test><![CDATA[pas de problème de balise orpheline :
<img>]]></test>
```

14.2. SimpleXML

Au lieu de manipuler directement les balises comme dans les exemples précédents, PHP propose l'extension SimpleXML pour construire un

document XML mais aussi et surtout le lire et accéder directement à n'importe quel nœud du document.

Autre avantage non négligeable de SimpleXML, cette extension est incluse par défaut dans PHP5.

Création

La création d'un document XML avec SimpleXML est très simple. Le point de départ consiste à créer l'élément racine ; un élément est représenté par la classe `SimpleXMLElement`. Le constructeur de cette classe prend en argument une chaîne de caractères représentant une donnée XML valide.

```
$racine = new SimpleXMLElement('<classe/>');
```

L'étape suivante consiste à attacher des éléments enfants à cette racine à l'aide de la méthode `addChild()` qui prend en premier argument le nom de l'élément et en second argument la valeur de l'élément.

```
$racine->addChild('eleve', 'Paul Dupont');
$racine->addChild('eleve', 'Eric Durant');
```

La représentation textuelle peut ensuite être obtenue en faisant appel à la méthode `asXML()` de l'élément racine (`$classe`).

```
echo $racine->asXML();
```

Cette méthode retourne une chaîne de caractères et ne prend en aucun cas soin de modifier le `Content-Type` pour indiquer au navigateur que le contenu est de type XML. La fonction `header()` doit par conséquent être utilisée.

Listing 14-2 : Création d'un document XML à l'aide de l'extension SimpleXML

```
$racine = new SimpleXMLElement('<classe/>');
$racine->addChild('eleve', 'Paul Dupont');
$racine->addChild('eleve', 'Eric Durant');
header('Content-Type: application/xml');
echo $racine->asXML();
```

SimpleXML permet également d'ajouter des attributs aux éléments à l'aide de la méthode `addAttribute()`.

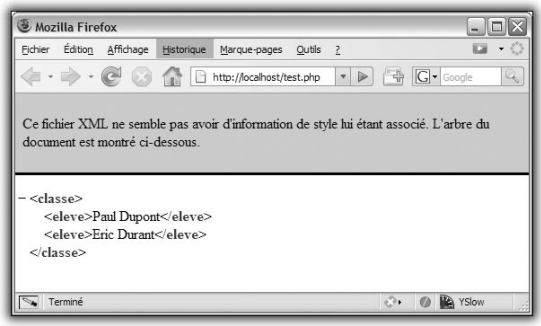

Figure 14.2 : *La méthode asXML() retourne bien une chaîne de caractères correspondant à la représentation XML*

Listing 14-3 : Utilisation de la méthode addAttribute()

```
$eleves = array(array('id' => 1,
                      'nom' => 'gueudet',
                      'prenom' => 'edouard'),
                array('id' => 2,
                      'nom' => 'henry',
                      'prenom' => 'thomas'));
$racine = new SimpleXMLElement('<classe/>');
foreach ($eleves as $eleve) {
  $noeud = $racine->addChild('eleve');
  $noeud->addChild('nom', $eleve['nom']);
  $noeud->addChild('prenom', $eleve['prenom']);
  $noeud->addAttribute('id', $eleve['id']);
}
header('Content-Type: application/xml');
echo $racine->asXML();
```

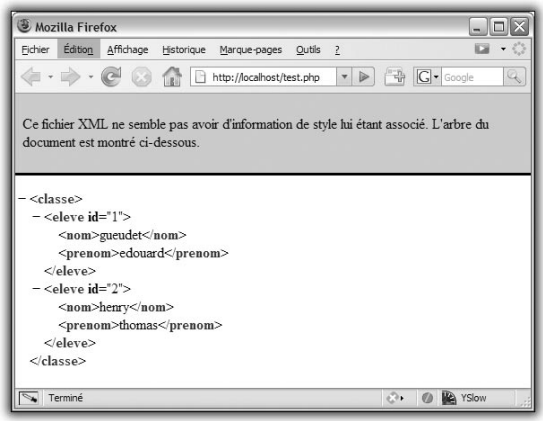

Figure 14.3 : *L'attribut id a bien été ajouté aux éléments <eleve>*

SimpleXML (comme les autres extensions PHP relatives à XML) manipule en interne des chaînes de caractères au format UTF-8. Il convient donc de vérifier que les données transmises sont au bon format.

Listing 14-4 : Utilisation de caractères non ASCII

```
$racine = new SimpleXMLElement('<tests/>');
header('Content-Type: application/xml');
$racine->addChild('test', 'hélène');
echo $racine->asXML();
```

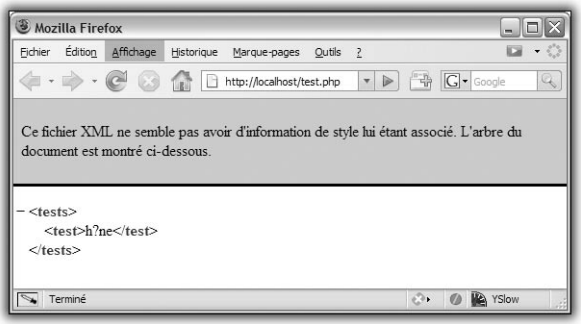

Figure 14.4 : *La chaîne "hélène", non convertie, s'affiche mal*

Deux solutions permettent de résoudre cette problématique. La première consiste à utiliser la fonction `utf8_encode()` pour transmettre toutes les chaînes dans le bon encodage.

```
$racine->addChild('test', utf8_encode('hélène'));
```

La seconde repose sur l'éditeur utilisé pour écrire les sources du script. Si le format d'encodage sélectionné est bien UTF-8, la donnée apparaîtra alors sans erreur.

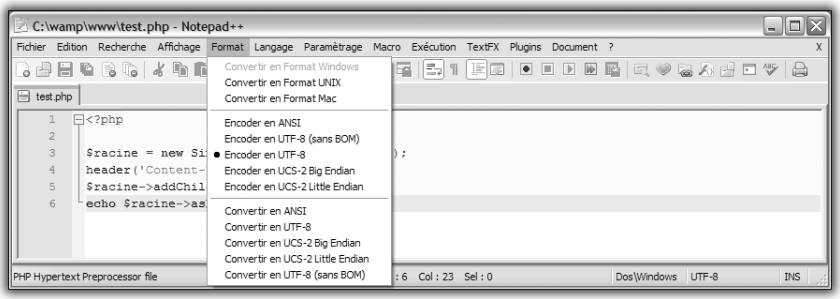

Figure 14.5 : *Notepad++ permet de forcer l'encodage du script en UTF8*

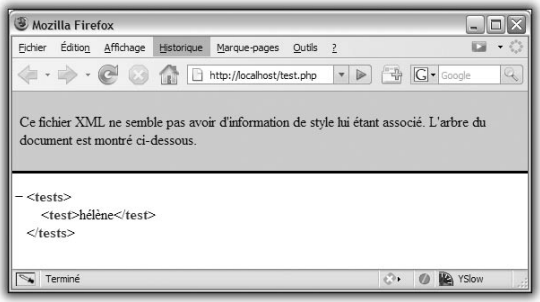

Figure 14.6 : *Le prénom "hélène" apparaît sans erreur quand l'encodage UTF8 est utilisé*

Lecture

L'opération de lecture repose en grande partie sur la méthode `children()`. Cette dernière retourne l'ensemble des enfants d'un élément donné. Fonctionnalité extrêmement pratique, un élément (`SimpleXMLElement`) donne accès aux valeurs de ses enfants directement via ses attributs. L'exemple suivant montre comment un élément (`coords`) peut accéder à la valeur de ses deux enfants : `x` et `y`.

```
$coords = new SimpleXMLElement('<coords><x>2</x><y>3</y>
✂ </coords>');
echo "x=".$coords->x." y=".$coords->y;
```

L'affichage des noms et prénoms des élèves d'une classe devient donc un jeu d'enfant.

Listing 14-5 : Parcours des élèves de la classe

```
$str = '<classe>
  <eleve id="1">
    <nom>gueudet</nom>
    <prenom>edouard</prenom>
  </eleve>
  <eleve id="2">
    <nom>henry</nom>
    <prenom>thomas</prenom>
  </eleve>
</classe>';

$classe = new SimpleXMLElement($str);

$eleves = $classe->children();
```

```
foreach ($eleves as $eleve) {
  echo $eleve->prenom." ".$eleve->nom."<br/>";
}
```

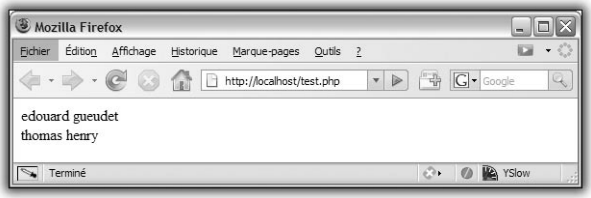

Figure 14.7 : *Les 2 élèves ont bien été trouvés*

La méthode `attributes()` donne accès à l'ensemble des attributs d'un élément.

L'exemple précédent peut être modifié de la façon suivante pour afficher l'`id` de l'élève.

```
foreach ($eleves as $eleve) {
  $attributs = $eleve->attributes();
  echo "[".$attributs["id"]."] ";
  echo $eleve->prenom." ".$eleve->nom."<br/>";
}
```

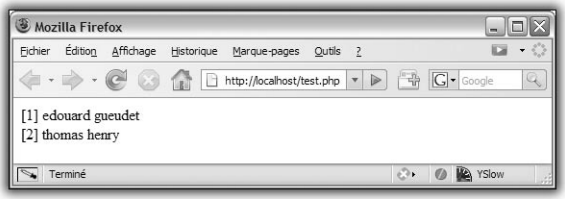

Figure 14.8 : *Affichage de l'id*

14.3. Formats spéciaux

Le format XML permet d'organiser les données selon son bon vouloir tout en étant certain qu'un partenaire sera en mesure de le manipuler. Il conviendra cependant, en même temps que la transmission des données, de fournir une documentation décrivant comment sont agencés les attributs et les balises, à quoi ils correspondent, etc.

Pour éviter cette fastidieuse tâche de description, l'industrie s'est très vite organisée pour faire émerger des structures de documents XML prédéfinies (spécifications).

RSS

Le format RSS est un cas typique de standardisation d'une structure XML pour répondre à un besoin partagé par de nombreux intervenants du monde du web. L'idée est ici de permettre à un site de contenu de pouvoir fournir à des partenaires un document contenant un résumé des derniers articles mis en ligne. Ces documents sont généralement qualifiés de flux (*feed*) RSS.

Figure 14.9 : *Icône utilisée par les différents navigateurs pour indiquer la présence d'un flux RSS*

En établissant ce standard, l'industrie informatique a pu miser et investir dessus. Les navigateurs web peuvent directement les afficher, des appareils électroniques tels que des iPods sont en mesure d'importer leur contenu, des sites web tels que Google Reader ne servent qu'à les agréger pour pouvoir les consulter en un point central.

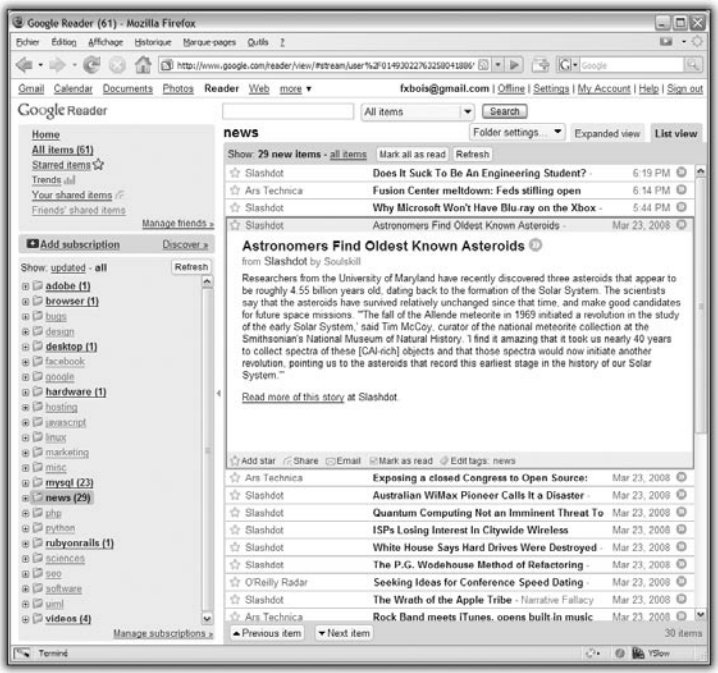

Figure 14.10 : *Google Reader, lecteur de flux RSS*

Figure 14.11 : *Affichage du flux RSS High Tech du Figaro*

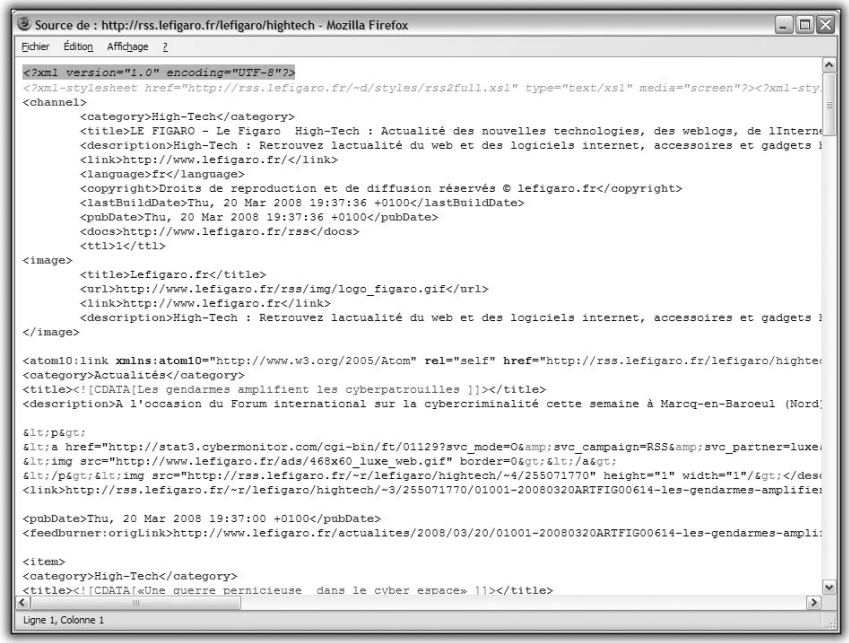

Figure 14.12 : *L'affichage des sources prouve bien qu'il s'agit d'un document XML de type RSS*

Structure

L'élément de plus haut niveau est la balise `<rss>` incluant en attribut la version du format. La version la plus récente et la plus répandue est la 2.0 : `<rss version= "2.0">`.

Le seul enfant de l'élément `<rss>` est l'élément `<channel>`.

Tableau 14.1 : *Enfants de l'élément* `channel`

Elément	Usage	Remarque
`title`	Titre du flux	Obligatoire
`link`	Url permettant d'accéder à ce flux	Obligatoire
`description`	Description	Obligatoire
`language`	Langue	

Tableau 14.1 : *Enfants de l'élément* `channel`		
Elément	**Usage**	**Remarque**
`Image`	Image	
`copyright`	Copyright	
`managingEditor`	Adresse email de la personne responsable du flux	
`webMaster`	Adresse email du webmaster	
`pubDate`	Date de publication	Format RFC 822
`lastBuildDate`	Date de dernière génération	Format RFC 822
`rating`		
`category`	Catégories	
`generator`	Nom du logiciel qui a permis de construire le flux	
`ttl`	Durée de vie	en minutes

L'élément <channel> peut disposer également de plusieurs éléments enfants <item> qui correspondent véritablement aux histoires/articles associés au flux.

Un <channel> contient donc plusieurs <item>. Une analogie possible consisterait à comparer un <channel> à un journal et les <item> à des articles. Le journal dispose bien d'un titre et d'un rédacteur en chef, comme les articles ont un titre et un auteur.

Chaque <item> dispose de plusieurs éléments enfants.

Tableau 14.2 : *Enfants de l'élément* `item`		
Elément	**Usage**	**Remarque**
`title`	Titre	Obligatoire
`link`	Lien direct vers l'article	Obligatoire
`description`	Description	Obligatoire
`author`	Auteur	Adresse email

Tableau 14.2 : *Enfants de l'élément* `item`		
Elément	**Usage**	**Remarque**
`category`	Catégories	
`comments`	Lien vers la page de commentaires	
`guid`	Numéro/Code unique de l'article	
`pubDate`	Date de publication	RFC 822

Le script suivant liste les enfants d'une classe mais cette fois sous la forme d'un flux RSS.

Listing 14-6 : Génération d'un flux RSS

```
$classe = array(array('id' => 1,
                      'nom' => 'gueudet',
                      'prenom' => 'edouard'),
                array('id' => 2,
                      'nom' => 'henry',
                      'prenom' => 'thomas'));

$rss = new SimpleXMLElement('<rss/>');
$rss->addAttribute('version', '2.0');

$channel = $rss->addChild('channel');
$channel->addChild('title', 'la classe');
$channel->addChild('link', 'http://localhost/test.php');
$channel->addChild('description', 'élèves de la classe');
$channel->addChild('generator', 'a la mano');

$lien_edit = 'http://localhost/eleve_edite.php?id=';
foreach ($classe as $eleve) {
  $item = $channel->addChild('item');
  $item->addChild('title', $eleve['prenom'].'
  ✂ '.$eleve['nom']);
  $item->addChild('link', $lien_edit.$eleve['id']);
  $item->addChild('guid', 'eleve-'.$eleve['id']);
  $item->addChild('description', 'id='.$eleve['id']);
}

header('Content-Type: application/xml');
echo $rss->asXML();
```

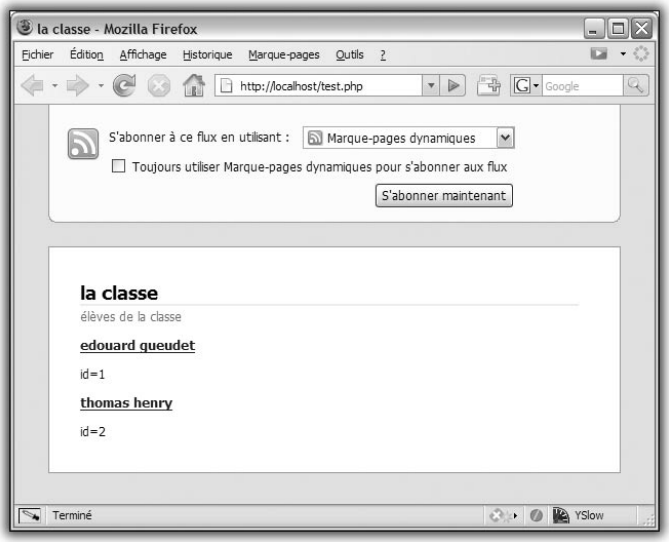

Figure 14.13 : *L'affichage d'un document XML est spécial dans le cadre d'un flux RSS*

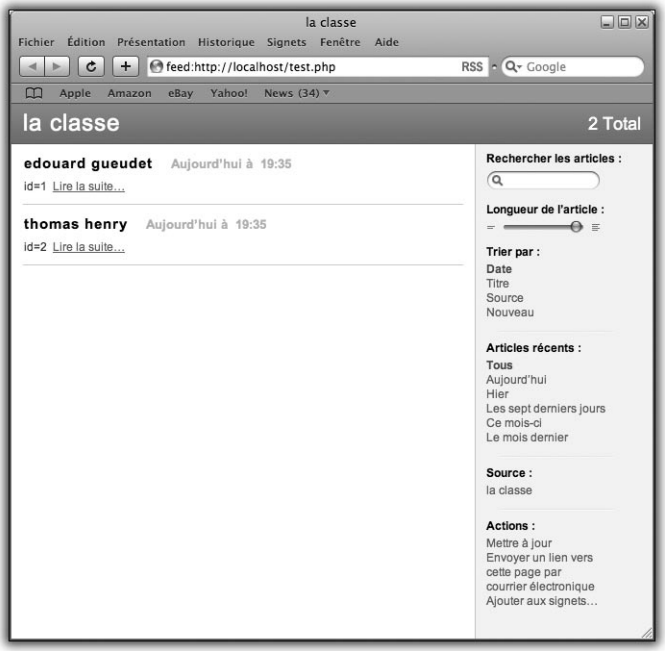

Figure 14.14 : *Chaque navigateur a un mode d'affichage des flux XML qui lui est propre*

Inclusion dans un site

Tous les navigateurs sont aujourd'hui en mesure d'indiquer la présence d'un flux RSS sur un site web. Pour Firefox, il s'agit d'une petite icône orange située à droite de l'adresse du site.

Figure 14.15 : *Le site kernix.com dispose bien d'un flux RSS associé*

Le site indique la présence d'un flux en utilisant une balise `<LINK>` spécifique dans l'en-tête (`<HEADER>`) de la page.

Listing 14-7 : Balise à ajouter pour indiquer au navigateur que le site dispose d'un flux RSS

```
<link rel="alternate" type="application/rss+xml"
✂ title="RSS" href="/test.php" />
```

L'attribut `href` permet de préciser l'emplacement du flux.

Flux multiples

Il est possible d'associer plusieurs flux à une page en multipliant les balises `<LINK>` dans l'en-tête. En cliquant sur l'icône de flux, le navigateur propose alors de sélectionner le flux souhaité. L'attribut `title` est dans ce cas très utile.

XHTML

L'XHTML est une version de l'HTML pour laquelle les contraintes de l'XML sont respectées. En développant des pages à la norme XHTML, ces dernières deviennent directement manipulables comme tout autre document XML.

L'exemple suivant permet de récupérer tous les liens (<a href="">) de la page du W3C consacrée à l'XHTML : **www.w3.org/TR/xhtml1**.

Le constructeur de la classe `SimpleXMLElement` peut prendre en premier paramètre une URL correspondant à un document XML. Pour cela, le troisième paramètre vaut `true`, et le second, `null`.

```
$liens = array();

function extraitLiens($noeud) {
  if (strtoupper($noeud->getName()) == 'A') {
    $attributs = $noeud->attributes();
    $href = (string) $attributs['href'];
    if (strlen($href) > 1) {
      $GLOBALS['liens'][] = $href;
    }
  }
  foreach ($noeud->children() as $enfant) {
    extraitLiens($enfant);
  }
}

$xml = new SimpleXMLElement("http://www.w3.org/TR/xhtml1/",
                            null,
                            true);

extraitLiens($xml);

print("<pre>");
print_r($liens);
print("</pre>");
```

Le script repose sur une utilisation récursive de la fonction `extraitLiens()`, qui, pour chaque nœud reçu en paramètre, teste si ce dernier correspond à un lien (`== 'A'`) et s'appelle de nouveau pour tous les éventuels enfants.

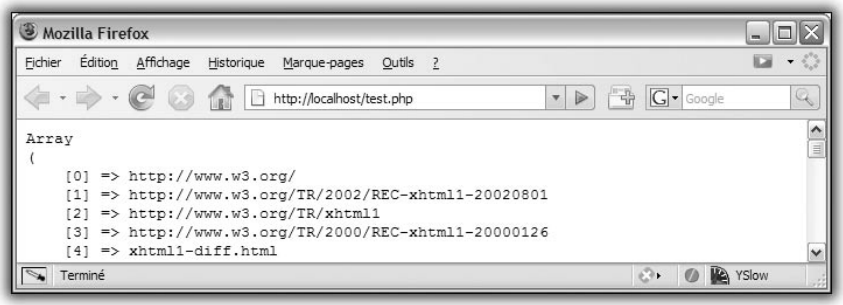

Figure 14.16 : *Liste des liens*

SVG

SVG est une spécification XML permettant de construire des images au format vectoriel. Une telle image a l'avantage de pouvoir être déformable sans perte de qualité.

Parmi les logiciels graphiques, Illustrator appartient au monde du vectoriel et Photoshop à celui des images bitmap.

L'exemple suivant permet de construire pas à pas un échiquier de 64 pièces.

Listing 14-8 : Construction d'un échiquier

```
$nb_pieces = 8;
$taille_piece = 30;
$couleurs = array('#FFFFFF', '#000000');

$svg = new SimpleXMLElement('<svg/>');
$svg->addAttribute('width', $nb_pieces * $taille_piece);
$svg->addAttribute('heigth', $nb_pieces * $taille_piece);
$svg->addAttribute('xmlns', 'http://www.w3.org/2000/svg');

for ($i = 0; $i < $nb_pieces; $i++) {
  for ($j = 0; $j < $nb_pieces; $j++) {
    $carre = $svg->addChild('rect');
    $carre->addAttribute('x', $j*$taille_piece);
    $carre->addAttribute('y', $i*$taille_piece);
    $carre->addAttribute('width', $taille_piece);
    $carre->addAttribute('height', $taille_piece);
    $carre->addAttribute('style',
    ✂ 'fill:'.$couleurs[($i+$j)%2]);
  }
```

```
}

header('Content-Type: application/xml');
echo $svg->asXML();
```

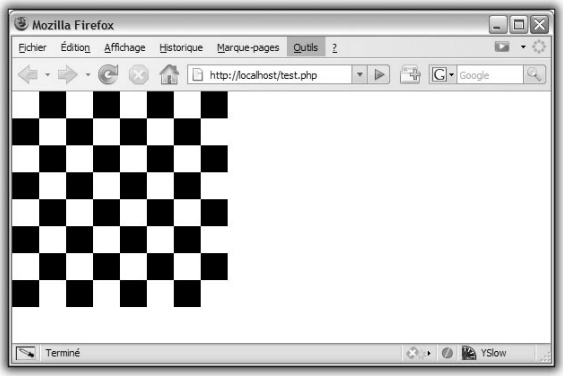

Figure 14.17 : *Affichage de l'échiquier dans un navigateur*

Ce fichier XML ne semble pas avoir d'information de style lui étant associé. L'arbre du document est montré ci-dessous.

```
− <svg width="240" heigth="240">
    <rect x="0" y="0" width="30" height="30" style="fill:#FFFFFF"/>
    <rect x="30" y="0" width="30" height="30" style="fill:#000000"/>
    <rect x="60" y="0" width="30" height="30" style="fill:#FFFFFF"/>
    <rect x="90" y="0" width="30" height="30" style="fill:#000000"/>
    <rect x="120" y="0" width="30" height="30" style="fill:#FFFFFF"/>
    <rect x="150" y="0" width="30" height="30" style="fill:#000000"/>
    <rect x="180" y="0" width="30" height="30" style="fill:#FFFFFF"/>
    <rect x="210" y="0" width="30" height="30" style="fill:#000000"/>
    <rect x="0" y="30" width="30" height="30" style="fill:#000000"/>
    <rect x="30" y="30" width="30" height="30" style="fill:#FFFFFF"/>
    <rect x="60" y="30" width="30" height="30" style="fill:#000000"/>
    <rect x="90" y="30" width="30" height="30" style="fill:#FFFFFF"/>
    <rect x="120" y="30" width="30" height="30" style="fill:#000000"/>
    <rect x="150" y="30" width="30" height="30" style="fill:#FFFFFF"/>
    <rect x="180" y="30" width="30" height="30" style="fill:#000000"/>
    <rect x="210" y="30" width="30" height="30" style="fill:#FFFFFF"/>
    <rect x="0" y="60" width="30" height="30" style="fill:#FFFFFF"/>
    <rect x="30" y="60" width="30" height="30" style="fill:#000000"/>
    <rect x="60" y="60" width="30" height="30" style="fill:#FFFFFF"/>
    <rect x="90" y="60" width="30" height="30" style="fill:#000000"/>
    <rect x="120" y="60" width="30" height="30" style="fill:#FFFFFF"/>
    <rect x="150" y="60" width="30" height="30" style="fill:#000000"/>
    <rect x="180" y="60" width="30" height="30" style="fill:#FFFFFF"/>
    <rect x="210" y="60" width="30" height="30" style="fill:#000000"/>
```

Figure 14.18 : *Les sources de la page révèlent bien qu'un document XML se cache derrière l'image*

Création d'images au format SVG

Le logiciel Inkscape est un concurrent open source d'Illustrator. Il peut être téléchargé gratuitement sur le site **www.inkscape.org**. Les créations réalisées peuvent être sauvegardées au format *SVG* et analysées avec un simple éditeur de texte pour comprendre la structure.

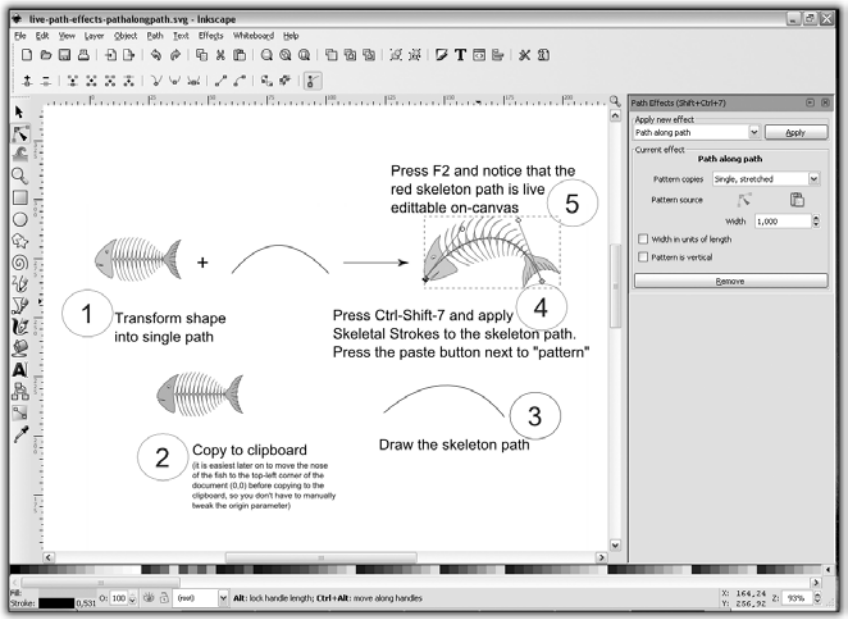

Figure 14.19 : *Inkscape permet de créer des images au format SVG*

14.4. Check-list

- Le format XML permet de stocker des informations et se révèle être le meilleur choix pour les échanges de données structurées.
- L'utilisation d'une extension de type SimpleXML est préférable à une construction "à la main" d'un fichier XML.
- Les nouveaux formats informatiques prennent de plus en plus souvent la forme de spécifications XML (*RSS*, *SVG*, *ODF*, *GDATA*, *SOAP*).

Chapitre 15

Les cookies et les sessions

Nous nous intéresserons dans ce chapitre aux différentes méthodes qui sont offertes pour associer des données à un internaute : les cookies et les sessions.

Pour illustrer ces différentes techniques, vous mettrez en place une mini-boutique.

15.1. Les cookies

En surfant sur le Web, vous avez pu vous apercevoir que certains sites étaient en mesure de vous reconnaître. En revenant sur une boutique où vous avez déjà acheté, il n'est pas rare d'y trouver des messages du genre : « Bienvenue M. Dupont » ; « Vous avez acheté le produit X, nous vous proposons cette liste de produits qui peuvent vous intéresser » ; « Voici les derniers produits apparus dans notre boutique depuis votre dernière visite », etc.

En vous connectant depuis une autre machine ou un autre compte, vous vous apercevrez cependant que tous ces messages sont absents.

Ces sites utilisent une technique qui leur permet de stocker des informations dans votre ordinateur, informations qui leur sont retransmises dès que vous naviguez sur leur site. Les données sont stockées dans de petits fichiers appelés *cookies*. Ces cookies ne servent pas que les intérêts marketing des grandes boutiques du Web, ils vous permettent aussi :

- de ne pas avoir à retaper systématiquement votre identifiant et votre mot de passe sur certains sites ;
- de pouvoir disposer d'un système de panier très pratique dans les boutiques en ligne.

Ces cookies contiennent tout type d'information et sont utilisés dans de très nombreux cas de figure. Souvent diabolisés, ils vous rendent finalement davantage service qu'ils ne vous desservent. Leur présence est d'ailleurs d'autant moins gênante qu'ils peuvent être effacés à tout moment. Sous Firefox, il suffit d'aller dans le menu **Outils/Vie privée** > **Afficher les cookies** puis d'appuyer sur le bouton **Supprimer tous les cookies** pour les faire disparaître.

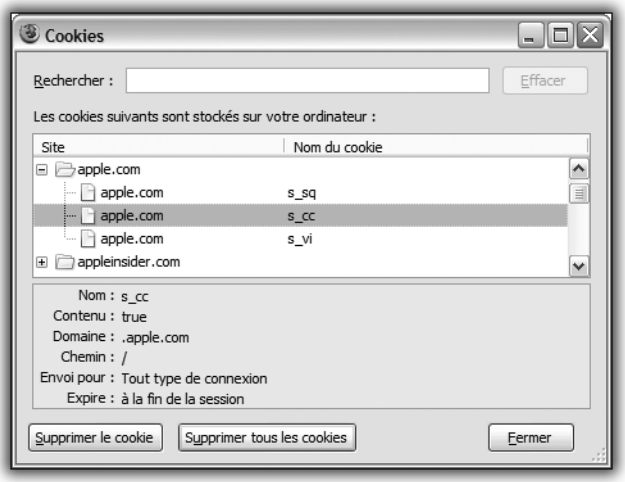

Figure 15.1 : *Suppression des cookies*

La perte de l'espace disque est aussi un reproche, très souvent exagéré, que l'on fait aux cookies. Un cookie fait en moyenne, entre 50 et 150 octets. Il vous faudrait stocker 10 millions de cookies pour perdre 1 Go d'espace disque !

Ce n'est pas gênant, mais quand même…

En identifiant de manière unique les internautes, certains services tels que **doubleclik.com** (gestion de bannières publicitaires) ont constitué une base de données mondiale afin d'étudier leur comportement. Ils savent où vous êtes allé, quand et pendant combien de temps… *Big Brother* n'est alors plus très loin !

Aspects techniques

Plusieurs fois dans ce livre, nous nous sommes intéressés à l'en-tête HTTP des pages web, notamment lors de l'utilisation de la fonction `header()`. Les cookies sont en réalité des données stockées en texte, qui sont transmises dans l'en-tête HTTP des requêtes et des réponses :

Listing 15-1 : Exemple de directive HTTP permettant de créer le cookie moncookie

```
Set-Cookie: moncookie=hello; path=/; expires Mon,
✂ 09-Dec-2002 13:46:00 GMT
```

Listing 15-2 : Directive HTTP contenue dans la requête envoyée par un navigateur

```
Cookie: moncookie=hello
```

La gestion des cookies en PHP est très simple. Elle ne nécessite que l'utilisation de la fonction `setcookie()`. Cette fonction prend par défaut deux paramètres : le nom du cookie et sa valeur.

Écrivez deux scripts, l'un pour créer le cookie et l'autre pour lire sa valeur :

Listing 15-3 : Le script cree_cookie.php

```
<?php
setcookie("moncookie","hello");
echo "cookie créé";
?>
```

Listing 15-4 : Le script lit_cookie.php

```
<?php
echo "valeur contenue dans mon cookie :
✂ ".$_COOKIE['moncookie'];
?>
```

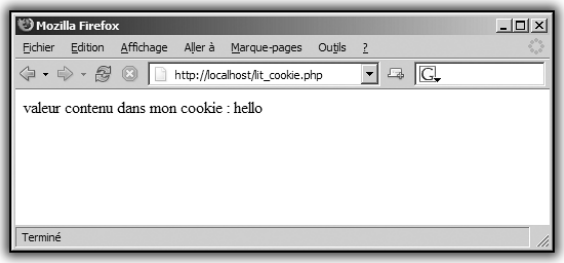

Figure 15.2 : *Lecture d'un cookie*

Comme les paramètres avec `$_REQUEST[]`, les cookies peuvent être récupérés au sein d'une variable super-globale : `$_COOKIE`. Ce tableau associatif contient autant de cellules que vous avez créé de cookies. Une fois le cookie `moncookie` créé, la variable `$_COOKIE['moncookie']` devient accessible dans vos scripts.

Suppression

Pour supprimer un cookie en PHP, il suffit de supprimer le contenu du cookie. Si vous souhaitez supprimer le cookie `moncookie`, écrivez `setcookie("moncookie")`.

Comme les cookies sont transmis dans l'en-tête HTTP, leur utilisation impose les mêmes contraintes que pour la fonction `header()` : interdiction absolue d'afficher quoi que ce soit avant d'utiliser `setcookie()`.

Limitations

Tout n'est tout de même pas permis avec les cookies. Voici une liste non exhaustive des limitations :

- Seul le site qui a créé le cookie peut y accéder.
- La taille d'un cookie doit être inférieure à 4 ko.
- Le nombre de cookies créés par un domaine donné est limité à 20.

La fonction `setcookie()` peut prendre d'autres paramètres…

- La date d'expiration : en secondes, la fonction PHP `time()` renvoyant la date actuelle en secondes.
- Le chemin : indique quelle partie du site peut avoir accès au cookie.
- Le domaine : indique quel domaine peut avoir accès au site.
- Un indicateur de sécurité : si sa valeur est `1`, il indique que la valeur du cookie ne peut être transmise que si vous êtes en HTTS (HTTP sécurisé par du SSL).

Voyez ces quelques exemples…

Premier exemple : modifiez le fichier *cree_cookie.php*.

```
<?php

setcookie("moncookie","hello",time()+5);

echo "cookie créé";

?>
```

Si vous appuyez, au bout de 5 secondes, sur le bouton **Rafraîchir** (*reload*) de la page **http://localhost/lit_cookie.php**, le cookie disparaîtra.

Heure du serveur

Il est courant que l'heure du serveur web soit décalée par rapport à l'heure de votre machine. Cela peut fausser vos tests. Pour voir la date et l'heure du serveur web, utilisez la fonction `date ("Y-m-d G:i:s")`.

Deuxième exemple :

```
setcookie("moncookie","hello",time()+3600,"/",".kernix .com",1);
```

Le cookie est, cette fois, créé pour une heure. Tous les scripts situés sur le domaine `.kernix.com` peuvent y accéder, à la condition que les transmissions soient cryptées.

Troisième exemple :

```
setcookie("moncookie","hello");
```

En ne précisant pas de date d'expiration, vous créez cette fois un cookie de session. Le cookie existera tant que le navigateur restera ouvert. Dès sa fermeture, le cookie sera automatiquement effacé.

Pour tester ce comportement, enchaînez les étapes suivantes :

1. Appelez le script *http:///localhost/cree_cookie.php* (en ayant modifié son contenu au préalable !).
2. Appelez le script *http:///localhost/lit_cookie.php*.
3. Fermez le navigateur.
4. Appelez de nouveau le script *http:///localhost/lit_cookie.php*. Le cookie a disparu.

Pour changer la valeur du cookie `moncookie`, il suffit d'appeler la fonction `setcookie()` en modifiant la valeur du cookie.

Application : la mini-boutique FoxShop

Pour appliquer les concepts que vous venez de voir, vous allez développer une boutique extrêmement simplifiée. Les cookies seront utilisés à plusieurs endroits :

- pour la gestion du panier ;
- pour enregistrer le profil du client et lui épargner une nouvelle saisie à chaque achat.

Pour ne pas mélanger les scripts de cette boutique avec votre applicatif de gestion d'école, vous allez créer le répertoire *boutique* sur votre compte distant et y placer tous vos développements. Pour accéder à la page d'accueil de la boutique, l'URL sera donc **http://localhost/boutique**.

Au niveau de la base de données, vous avez besoin de deux tables.

- *produit :* idproduit, référence produit, nom, prix, description.
- *commande :* idcommande, liste des produits, montant global, nom client, prénom client, adresse client, date.

Votre applicatif sera composé de deux parties : front-office et back-office.

Le front-office, qui permet à un internaute d'acheter en ligne, contient les fichiers suivants :

- *index.php* (page d'accueil de la boutique) ;
- *boutique.php* (affichage du catalogue) ;
- *ajout_caddie.php* (ajout d'un produit au panier) ;
- *voir_caddie.php* (résumé de la commande, identification de paiement) ;
- *enregistre_commande.php* (enregistrement de la commande).

Le back-office, qui permet de gérer les produits et les commandes, contient les fichiers suivants :

- *adm_produits.php* (script permettant l'ajout de produits) ;
- *adm_commandes.php* (script listant les commandes) ;
- *identification.inc.php* ;
- *variables.inc.php* ;
- *haut.inc.php* ;
- *bas.inc.php*.

Entraînez-vous

Ce pourrait être le bon moment de fermer le livre et d'essayer de réaliser ce petit applicatif. Si vous cherchez de votre côté, les progrès viendront en effet bien plus vite.

Le back-office

Les tables dont vous avez besoin peuvent être définies de la manière suivante :

Table *produit* :

Listing 15-5 : Table produit

```
CREATE TABLE produit (
  idproduit int(10) unsigned NOT NULL auto_increment,
  reference varchar(16) NOT NULL default '',
  nom varchar(16) NOT NULL default '',
  description tinytext NOT NULL,
  prix float(12,2) unsigned NOT NULL default '0.00',
  PRIMARY KEY  (idproduit)
)
```

Table *commande* :

Listing 15-6 : Table commande

```
CREATE TABLE commande (
  idcommande int(10) unsigned NOT NULL auto_increment,
  produits varchar(64) NOT NULL default '',
  montant float(12,2) unsigned NOT NULL default '0.00',
  nom varchar(16) NOT NULL default '',
  prenom varchar(16) NOT NULL default '',
  adresse tinytext NOT NULL,
  date datetime NOT NULL default '0000-00-00 00:00:00',
  PRIMARY KEY  (idcommande)
)
```

Listing 15-7 : Le script adm_produits.php

```
<?php

include("variables.inc.php");
include("identification.inc.php");

if ($_REQUEST['enregistre'] == "oui") {

  if (empty($_REQUEST['reference']) ||
      empty($_REQUEST['nom']) ||
      empty($_REQUEST['prix']) ||
      empty($_REQUEST['description']))
       die("ERREUR : tous les champs doivent être
       ✂ remplis.");
  if (preg_match("/^\d+(\.\d+)?$/",$_REQUEST['prix']) == false)
    die("ERREUR : prix non valide.");

  $liendb = mysql_connect($bddserver, $bddlogin,
  ✂ $bddpassword);
  mysql_select_db ($bdd);

  $sql = "INSERT INTO $table_produit (reference, nom,
  ✂ prix, description)".
         " VALUES (".
         "'".$_REQUEST['reference']."',".
         "'".$_REQUEST['nom']."',".
         "'".$_REQUEST['prix']."',".
         "'".$_REQUEST['description']."')";
  mysql_query ($sql);

  mysql_close($liendb);

}

include("haut.inc.php");

?>
```

```
<p>:: ajouter un produit au catalogue de la boutique</p>

<form action="<?php echo $_SERVER['PHP_SELF']; ?>"
➥ method="post">
<input type="hidden" name="enregistre" value="oui" />

<table>
<tr>
 <td>référence</td>
 <td><input type="text" name="reference" /></td>
 <td rowspan="3" valign="top">description</td>
 <td rowspan="3"><textarea name="description"
 ➥ rows="8"></textarea></td>
</tr>
<tr>
 <td>nom</td>
 <td><input type="text" name="nom" /></td>
</tr>
<tr>
 <td>prix</td>
 <td><input type="text" name="prix" /></td>
</tr>
</table>

<br/>

<input type="submit" value="enregistrer le produit" />

</form>

<hr>

<p>:: les produits de la boutique</p>

<table width="98%" align="center" border="1">
 <tr>
  <td class="intitule">id</td>
  <td class="intitule">référence</td>
  <td class="intitule">nom</td>
  <td class="intitule">prix  </td>
 </tr>

<?php

$liendb = mysql_connect($bddserver, $bddlogin,
➥ $bddpassword);
mysql_select_db ($bdd);

$sql = "SELECT * FROM $table_produit";
$resultat = mysql_query ($sql);

if (mysql_num_rows($resultat) > 0) {
```

```
  while ($produit = mysql_fetch_array ($resultat)) {
    $id = $produit['idproduit'];
    $reference = $produit['reference'];
    $nom = $produit['nom'];
    $prix = $produit['prix'];
    echo "<tr>";
    echo " <td>$id</td>";
    echo " <td>$reference</td>";
    echo " <td>$nom</td>";
    echo " <td>$prix</td>";
    echo "</tr>";
  }
}
else
{
  echo "<tr><td colspan='4' align='center'>aucun
  ✂ produit</td></tr>";
}

echo "</table>";

mysql_close($liendb);

include("bas.inc.php");

?>
```

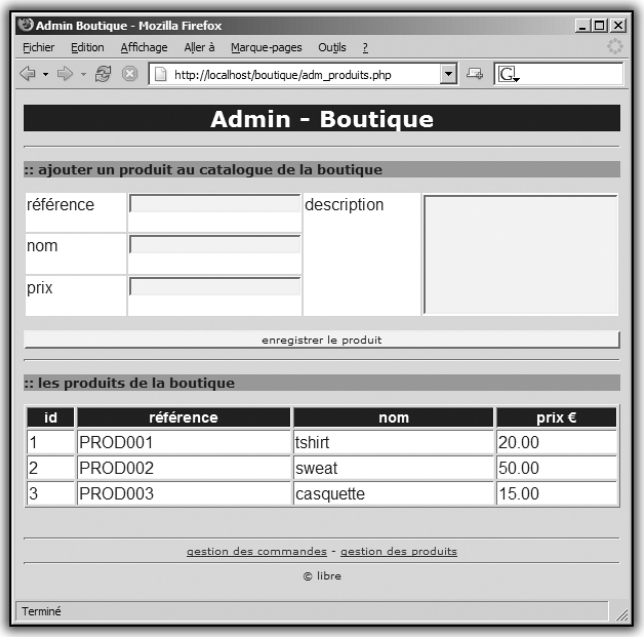

Figure 15.3 : *Page d'administration de la boutique*

Listing 15-8 : Le script adm_commandes.php

```
<?php

include("variables.inc.php");
include("identification.inc.php");
include("haut.inc.php");

?>

<p>:: les commandes</p>

<table>
 <tr>
  <td class="intitule">id</td>
  <td class="intitule">contenu</td>
  <td class="intitule">client</td>
  <td class="intitule">montant  </td>
  <td class="intitule">date</td>
 </tr>

<?php

$liendb = mysql_connect($bddserver, $bddlogin,
✂ $bddpassword);
mysql_select_db ($bdd);

$sql = "SELECT * FROM $table_commande";
$resultat = mysql_query ($sql);

if (mysql_num_rows($resultat) > 0) {

  while ($produit = mysql_fetch_array ($resultat)) {
    $id = $produit['idcommande'];
    $produits =
    ✂ str_replace(",","<br/>",$produit['produits']);
    $client = $produit['prenom']."
    ✂ ".$produit['nom']."<br/>".nl2br($produit['adresse']);
    $montant = $produit['montant'];
    $date = $produit['date'];
    echo "<tr>";
    echo "<td>$id</td>";
    echo "<td>$produits</td>";
    echo "<td>$client</td>";
    echo "<td>$montant</td>";
    echo "<td>$date</td>";
    echo "</tr>";
  }
}
else echo "<tr><td colspan='5'>aucune commande</td></tr>";
```

```
echo "</table>";

mysql_close($liendb);

include("bas.inc.php");

?>
```

Listing 15-9 : Le script haut.inc.php

```
<html>
<head>

<title>Admin Boutique</title>

<style type="text/css">
<!--

*
{ font-family:verdana; font-size:10px;}

BODY
{ background:#D1DAE3; color:#01291A; }

TABLE
{width:100%; }

H1
{ background: #091F35; color: white; font-size:22px;
✂ font-weight: bold; }

P
{ background:#8F9BB7; color:#091F35; font-size:13px;
✂ font-weight:bold; text-align:left; }

TD
{ background:white; color:#091F35; font-family:arial;
✂ font-size:16px; font-weight:normal; vertical-align:top; }

TD.intitule
{ background:#091F35; color:white; font-size:14px;
✂ font-weight:bold; text-align:center; }

INPUT
{ background:#FCF0E9; color:#01291A; width:100%; }

TEXTAREA
{ background:#FCF0E9; color:#01291A; width:100%; }
```

```
SELECT
{ background:#FCF0E9; color:#01291A; }

HR
{ color: #091F35; }

A
{ color: #6C111E; }

A:hover
{ background:#6C111E; color:white; font-weight:bold;}

-->
</style>

</head>
<body>

<center>

<h1>Admin - Boutique</h1>

<hr/>
```

Listing 15-10 : Le script bas.inc.php

```
<br/><br/><hr/>

<a href="adm_commandes.php">gestion des commandes</a> -
<a href="adm_produits.php">gestion des produits</a>

<hr/>

© libre

</center>

</body>
</html>
```

Listing 15-11 : Le script identification.inc.php

```
<?php

if ($_SERVER['PHP_AUTH_USER']!= "essai" ||
    $_SERVER['PHP_AUTH_PW']!="essai") {

  header("status: 401 Unauthorized");
  header("HTTP/1.0 401 Unauthorized");
  header("WWW-authenticate: basic realm=\"acces
  ✂ securise\"");
```

```
  print("Echec");
  exit(0);

}

?>
```

Listing 15-12 : variables.inc.php

```
<?php

$bddserver      = "localhost";
$bddlogin       = "root";
$bddpassword    = "";
$bdd            = "test";
$table_commande = "commande";
$table_produit  = "produit";
$url            = "http://localhost/boutique";

?>
```

Vous constatez la présence, dans le script *adm_produits.php*, de la ligne suivante :

```
<form action="<?php echo $PHP_SELF; ?>" method="post">
```

La variable `$_SERVER['PHP_SELF']` est une variable interne de PHP, qui contient à tout moment l'URL du script appelé.

Variables prédéfinies

PHP dispose, par défaut, d'un certain nombre de variables prédéfinies. Celles-ci contiennent des informations sur le serveur web, la page appelée, l'internaute, etc.

Une liste exhaustive de ces variables est fournie dans les annexes.

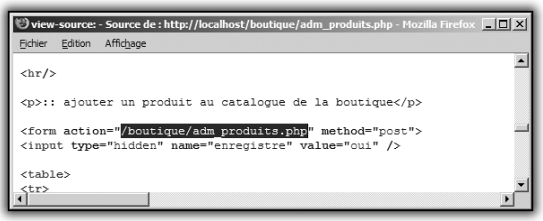

Figure 15.4 : *Contenu de la variable PHP_SELF*

Il peut être très intéressant de passer par cette variable quand vous souhaitez faire référence au script dans lequel vous vous trouvez. De cette manière, vous limitez le nombre de mises à jour si vous devez renommer le script.

`$_SERVER['PHP_SELF']` et `include()`

Si un fichier *f1.php* (**http://localhost/f1.php**) inclut un fichier *f2.php*, la variable `$_SERVER['PHP_SELF']` contiendra `/f1.php` dans `f1.php` ET dans `f2.php`. `$_SERVER['PHP_SELF']` contient l'URL appelée, et non le chemin du fichier dans lequel on se trouve !

Dans le script *adm_commandes.php*, vous remarquez l'utilisation des fonctions `nl2br()` et `str_replace()`.

- `nl2br()` : lorsque vous enregistrez dans une table, une donnée multiligne provenant d'un `textarea`, cette donnée est stockée avec des retours à la ligne (codés `\n`). Pour l'afficher dans une page web, il est donc nécessaire d'utiliser la fonction `nl2br()` pour convertir les retours à la ligne texte (`\n`) en retours à la ligne HTML (`<br/>`).
- `str_replace()` : permet de remplacer un texte par un autre dans une chaîne de caractères. Si vous souhaitez remplacer la chaîne `"toto"` par la chaîne `"titi"` dans la chaîne `$ch` et stocker le résultat dans la chaîne `$ch2`, il suffit d'écrire `$ch2 = str_replace("toto","titi",$ch);`.

Une description plus approfondie de cette fonction est fournie dans le chapitre « Fonctions PHP ». Vous y découvrirez notamment les possibilités impressionnantes qu'elle offre lorsqu'elle est utilisée en adéquation avec les expressions régulières.

Le front-office

Le front office correspond dans cet exemple à la zone du site qui permettra à l'internaute de sélectionner ses achats et de passer sa commande. Commencez par réaliser la page d'accueil et la page catalogue :

Listing 15-13 : Le fichier index.html

```
<html>
 <head>
  <title>Boutique FoxShop</title>
  <link href="look.css" rel="stylesheet" type="text/css" />
 </head>
<body>

<div class='titre'><a href='boutique.php'>Boutique
✂ <i>FoxSHOP</i></a></div>

<div class='bienvenue'>Welcome</div>

</body>
</html>
```

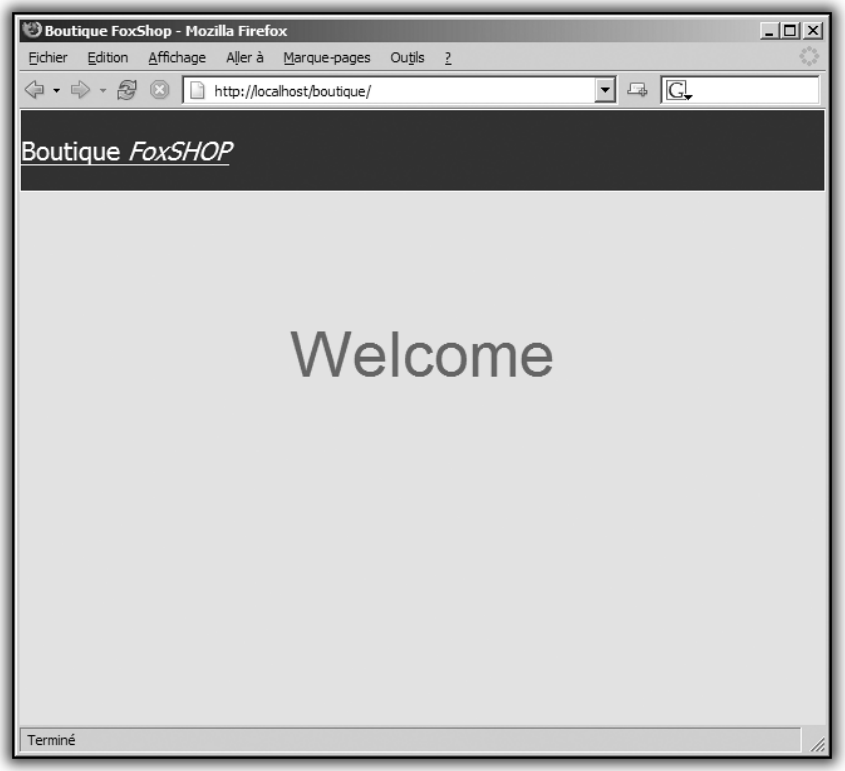

Figure 15.5 : *Page d'accueil de la boutique*

A priori, rien à signaler, si ce n'est la gestion des styles (CSS). Comme la page d'accueil est de type HTML, vous ne pouvez recourir à la

technique d'inclusion et utiliser un fichier *haut.inc.php* pour inclure les caractéristiques visuelles de la boutique. Heureusement, une autre technique permet d'isoler les CSS en un endroit unique. Les propriétés CSS peuvent être incluses dans un fichier extérieur qui sera lié à la page web de la manière suivante :

```
<link href="look.css" rel="stylesheet" type="text/css" />
```

Dans ce cas, ce fichier s'appelle *look.css*, il se situe au même niveau que le script *index.php* et contient les instructions suivantes :

```
BODY
{background: #025053;}

TD
{background: #C2DADB; color: #022324; font-family:
✂ georgia;
 font-weight: bold; font-size: 14px; letter-spacing: 1px;}

A
{color: #EB8814; font-family: georgia; font-weight: bold;
 font-size: 14px; letter-spacing: 1px;}

.titre
{color: #047F84; font-family: georgia; font-weight: bold;
 font-size: 24px; letter-spacing: 1px;}

.reference
{color: #047F84; font-family: arial; font-weight: bold;
 font-size: 12px; letter-spacing: 1px;}

.description
{background: #DAE9EA; color: #047F84; font-family:
✂ verdana;
 font-weight: normal; font-size: 12px; letter-spacing: 1px;}
```

Passez maintenant au script *boutique.php*, qui permet d'accéder à l'ensemble du catalogue :

Listing 15-14 : boutique.php

```
<?php

include("variables.inc.php");

if (!isset($_REQUEST['id'])) $id = 1;
else $id = $_REQUEST['id'];

?>

<html>
```

```
 <head>
  <title>Boutique FoxShop - catalogue</title>
  <link href="look.css" rel="stylesheet" type="text/css" />
 </head>
<body>

<div class='titre'><a href='boutique.php'>Boutique
✂ <i>FoxSHOP</i></a></div>

<table class='catalogue'>
<tr>
 <td class='liste'>
  <div class='tdTitre'>Nos produits</div>

<?php
$liendb = mysql_connect($bddserver, $bddlogin,
✂ $bddpassword);
mysql_select_db ($bdd);
$sql = "SELECT * FROM $table_produit";
$resultat = mysql_query ($sql);
while ($produit = mysql_fetch_array ($resultat)) {
  print("  - ");
  print("<a href=".$_SERVER['PHP_SELF']."?id=".$produit
  ✂ ['idproduit']."">".
        $produit['nom']."</a>");
  print("<br/>");
}
?>

 </td>
 <td class='detail'>

<?php
$sql = "SELECT * FROM $table_produit WHERE idproduit =
✂ '$id'";
$resultat = mysql_query ($sql);
$produit = mysql_fetch_array ($resultat);
print("<div class='tdTitre'>".$produit['nom'].
      " [ref#".$produit['reference']."]</div>");
?>

<div class='description'>

<?php
print(nl2br($produit['description'])."<br/><br/>");
print($produit['prix']."  <br/><br/>");
mysql_close($liendb);
?>

<form action="ajout_caddie.php" method="post">
 <input type="hidden" name="id" value="<?php echo $id; ?>" />
```

```
 <input type="submit" value="ajouter au panier" />
</form>

</div>

 </td>
</tr>
</table>

</body>
</html>
```

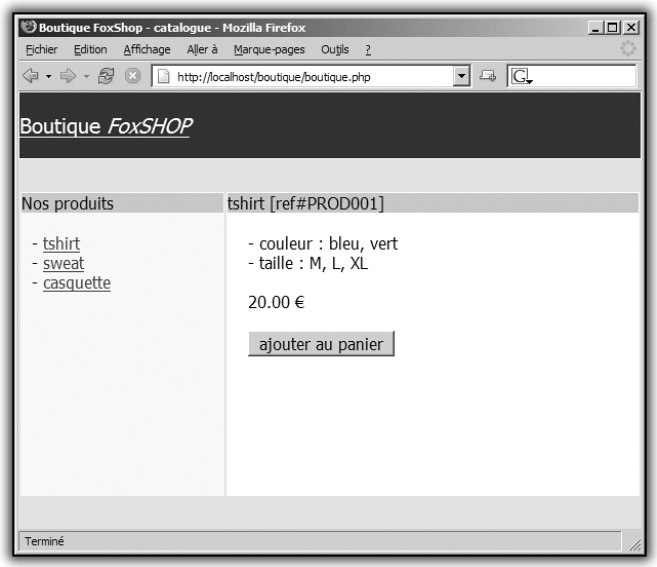

Figure 15.6 : *Catalogue de la boutique*

Quand aucun produit n'est spécifié, vous prenez la valeur par défaut `1` :

```
if (!isset($_REQUEST['id'])) $id = 1;
else $id = $_REQUEST['id'];
```

C'est par exemple le cas quand vous venez de la page d'accueil.

En revanche, lorsque vous cliquez sur le nom d'un produit, vous passez le paramètre `id` au script et lui permettez ainsi d'aller chercher les caractéristiques du produit associé. Vous pouvez donc dire que le catalogue de la boutique est entièrement dynamique.

Le bouton `"ajouter au panier"` va permettre de passer la commande en basculant sur *ajout_caddie.php*.

Venons-en justement au cœur du problème : la gestion du panier. L'idée est de construire un cookie `monpanier` qui contiendra les `id` de tous les produits qui y ont été ajoutés. Pour faire simple, vous choisirez le format de stockage `idprod1,idprod2,idprod3`, etc.

Le fichier *ajout_caddie.php* se contente donc de concaténer la chaîne `,idprodN` à la fin du cookie existant (s'il n'existe pas, il s'initialise). Une fois la manipulation sur le cookie réalisée, vous êtes redirigé sur la fiche du produit que vous venez d'acheter.

Le code du script *ajout_caddie.php* devient maintenant évident :

Listing 15-15 : Le script ajout_caddie.php

```
<?php
include("variables.inc.php");
setcookie("monpanier",$_COOKIE['monpanier'].",".$_REQUEST
✂ ['id'],time()+86400);
header("Location: $url/boutique.php?id=".$_REQUEST['id']);
?>
```

Vous disposez donc d'un panier et il devient nécessaire d'y faire référence dans la boutique.

Apportez quelques modifications à *boutique.php* de manière que les éléments suivants soient affichés (lorsque le panier existe) :

- le message `"Votre panier contient N articles"` ;
- un bouton pour valider la commande ;
- le contenu du panier pour rendre compte, *de visu*, du contenu du cookie.

Listing 15-16 : Vous disposez maintenant d'un panier dans la boutique

```
<?php

include("variables.inc.php");

if (!isset($_REQUEST['id'])) $id = 1;
else $id = $_REQUEST['id'];

?>

<html>
 <head>
  <title>Boutique FoxShop - catalogue</title>
  <link href="look.css" rel="stylesheet" type="text/
  ✂ css" />
```

```
 </head>
<body>

<div class='titre'><a href='boutique.php'>Boutique
✂ <i>FoxSHOP</i></a></div>

<table class='catalogue'>
<tr>
 <td class='liste'>
  <div class='tdTitre'>Nos produits</div>

<?php
$liendb = mysql_connect($bddserver, $bddlogin,
✂ $bddpassword);
mysql_select_db ($bdd);
$sql = "SELECT * FROM $table_produit";
$resultat = mysql_query ($sql);
while ($produit = mysql_fetch_array ($resultat)) {
  print("  - ");
  print("<a href=".$_SERVER['PHP_SELF']."?id="
  ✂ .$produit['idproduit'].">".
        $produit['nom']."</a>");
  print("<br/>");
}
?>

 </td>
 <td class='detail'>

<?php
$sql = "SELECT * FROM $table_produit WHERE idproduit
✂ = '$id'";
$resultat = mysql_query ($sql);
$produit = mysql_fetch_array ($resultat);
print("<div class='tdTitre'>".$produit['nom'].
      " [ref#".$produit['reference']."]</div>");
?>

<div class='description'>

<?php
print(nl2br($produit['description'])."<br/><br/>");
print($produit['prix']."  <br/><br/>");
mysql_close($liendb);
?>

<form action="ajout_caddie.php" method="post">
 <input type="hidden" name="id" value="<?php echo
 ✂ $id; ?>" />
 <input type="submit" value="ajouter au panier" />
</form>
```

```
<?php

if (isset($_COOKIE['monpanier']))
{
 print("<div class='panier'>");
 $tab = split(",",$_COOKIE['monpanier']);
 $nb_prod = sizeof($tab) - 1;
 print("votre panier contient ".$nb_prod."
 ✂ produit(s)<br/>");
 print("<form action='voir_caddie.php'
 ✂ method='post'>");
 print("<input type='submit' value='valider la
 ✂ commande'/></form>");
 print("cookie = {".$_COOKIE['monpanier']."}");
 print("</div>");
}

?>

</div>

 </td>
</tr>
</table>

</body>
</html>
```

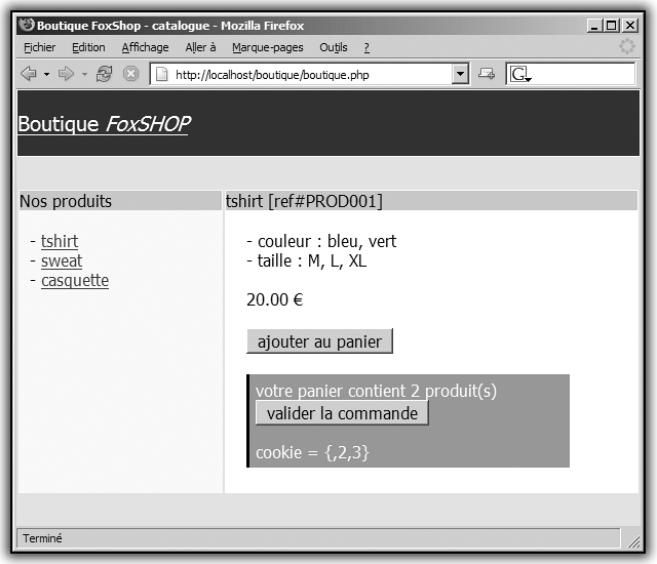

Figure 15.7 : *Vous avez acheté un sweater (id=2) et une casquette (id=3)*

Vous utilisez pour trouver le nombre d'éléments dans le panier la fonction `split()`. Utilisée sur la chaîne `",2,3"`, la fonction `split(",",",2,3")` retourne un tableau de trois éléments avec un premier élément vide. De ce fait, vous êtes obligé de décrémenter de `1` la taille du tableau pour trouver le nombre exact de produits :

```
$nb_prod = sizeof($tab) - 1;
```

Passez maintenant à la validation de la commande. Vous avez deux scripts à écrire :

- Le script *voir_caddie.php* liste les produits présents dans le panier et vous permet d'enregistrer vos informations client.
- Le script *enregistre_commande.php*, appelé par *voir_caddie.php*, enregistre la commande dans la base et supprime le caddie.

Listing 15-17 : Le script voir_caddie.php (voir Figure 12.8)

```
<?php
include("variables.inc.php");
?>

<html>
 <head>
  <title>Boutique FoxShop - validation
  ✂ commande</title>
  <link href="look.css" rel="stylesheet" type="text/
  ✂ css" />
 </head>
<body>

<div class='titre'><a href='boutique.php'>Boutique
✂ <i>FoxSHOP</i></a></div>

<div class='caddie'>

<?php
$montant = 0;
$listeproduits = " ";
$_COOKIE['monpanier'][0] = ' ';
$liendb = mysql_connect($bddserver, $bddlogin,
✂ $bddpassword);
mysql_select_db ($bdd);
$sql = "SELECT * FROM $table_produit ".
       "WHERE idproduit IN
       ✂ (".$_COOKIE['monpanier'].")";
$resultat = mysql_query ($sql);
print("<table width='100%'>");
while ($prod = mysql_fetch_array ($resultat)) {
  print("<tr>");
```

```
  print("<td class='prod'>[".$prod['reference']."]
  ✂ ".$prod['nom']."</td>");
  print("<td class='montant'>".$prod['prix']."
  ✂  </td></tr>");
  $montant += $prod['prix'];
  $listeproduits .= ',' .  $prod['reference'];
}
$listeproduits[0] = ' ';
// frais de port
$montant += 5;
print("<tr><td class='total'>MONTANT + PORT</td>");
print("<td class='total'>$montant  </td></tr>");
print("</table>");
mysql_close($liendb);
?>

<form action="enregistre_commande.php" method="post">

 <input type="hidden" name="montant" value="<?php
 ✂ echo $montant; ?>">
 <input type="hidden" name="listeproduits"
        value="<?php echo $listeproduits; ?>">

 <label>nom</label><br/><input type="text" name="nom"
 ✂ /><br/>
 <label>prénom</label><br/><input type="text"
 ✂ name="prenom" /><br/>
 <label>adresse</label><br/><input type="text"
 ✂ name="adresse" /><br/>
 <label>code postal</label><br/><input type="text"
 ✂ name="cp" /><br/>
 <label>ville</label><br/><input type="text"
 ✂ name="ville" /><br/>
 <input type="submit"  value="enregistrer ma
 ✂ commande" />

</form>

</div>

</body>
</html>
```

(voir Figure 12.8)

Pour ne pas être gêné par les virgules initiales dans `$_COOKIE['monpanier']` et `$listeproduits`, remplacez la première lettre par un caractère blanc :

```
$monpanier[0] = ' ';
```

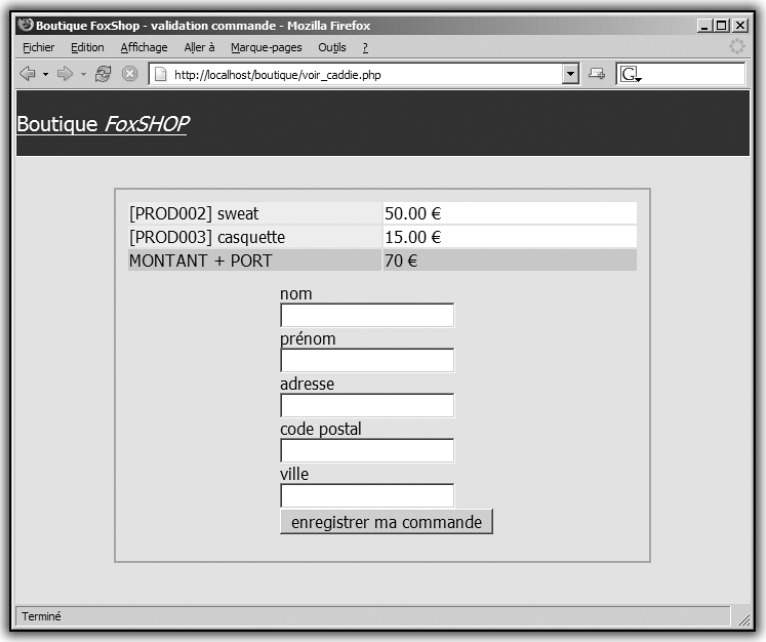

Figure 15.8 : *Visualisation du caddie*

Vous avez vu, en effet, qu'une chaîne de caractères pouvait être considérée comme un tableau de caractères. Il est donc possible de modifier un à un les caractères en y accédant par un index.

Supposons maintenant que vous ayez, dans votre panier, les produits `2` et `3`. Pour les sélectionner dans la base, la solution la plus évidente est la requête SQL suivante :

```
SELECT * FROM $table_produit WHERE idproduit = '2' OR
idproduit = '3'
```

Dans le cas présent, vous allez tirer parti d'une autre syntaxe, plus compacte, qui permet d'aboutir au même résultat.

La ligne suivante :

```
SELECT * FROM $table_produit WHERE idproduit IN (2,3)
```

… est équivalente à la requête précédente et signifie : « sélectionner tous les produits dont l'`idproduit` est contenu sur la liste suivante (2,3) ».

Ce procédé comporte cependant un sérieux défaut : si vous ajoutez plusieurs fois le même produit au panier, celui-ci n'apparaît qu'une

seule fois dans votre récapitulatif. Il faut donc préciser pour chaque produit sa quantité dans le panier. La fonction `array_count_values()`, qui dénombre le nombre d'occurrences d'une donnée dans un tableau, peut vous être utile.

Listing 15-18 : Le script voir_caddie.php

```
<?php
include("variables.inc.php");
?>

<html>
 <head>
  <title>Boutique FoxShop - validation commande</title>
  <link href="look.css" rel="stylesheet" type="text/css" />
 </head>
<body>

<div class='titre'><a href='boutique.php'>Boutique
✂ <i>FoxSHOP</i></a></div>

<div class='caddie'>

<?php
$montant = 0;
$listeproduits = " ";
$_COOKIE['monpanier'][0] = ' ';
$liendb = mysql_connect($bddserver, $bddlogin,
✂ $bddpassword);
mysql_select_db ($bdd);
$sql = "SELECT * FROM $table_produit ".
       "WHERE idproduit IN (".$_COOKIE['monpanier'].")";
$resultat = mysql_query ($sql);
print("<table width='100%'>");
$tab = array_count_values(split(",",$_COOKIE['monpanier']));
while ($prod = mysql_fetch_array ($resultat)) {
  print("<tr><td class='prod'>");
  print("[".$prod['reference']."] ".$prod['nom']);
  print(" (x".$tab[$prod['idproduit']].")");
  print("</td><td class='montant'>");
  print($prod['prix']."  ");
  print("</td></tr>");
  $montant += $prod['prix']*$tab[$prod['idproduit']];
  $listeproduits .= ',' .  $prod['reference'];
}
$listeproduits[0] = ' ';
// frais de port
$montant += 5;
print("<tr><td class='total'>MONTANT + PORT</td>");
print("<td class='total'>$montant  </td></tr>");
print("</table>");
```

```
mysql_close($liendb);
?>

<form action="enregistre_commande.php" method="post">

 <input type="hidden" name="montant" value="<?php echo
 ✂ $montant; ?>">
 <input type="hidden" name="listeproduits"
        value="<?php echo $listeproduits; ?>">

 <label>nom</label><br/><input type="text" name="nom" /><br/>
 <label>prénom</label><br/><input type="text" name="prenom"
 ✂ /><br/>
 <label>adresse</label><br/><input type="text"
 ✂ name="adresse" /><br/>
 <label>code postal</label><br/><input type="text"
 ✂ name="cp" /><br/>
 <label>ville</label><br/><input type="text" name="ville"
 ✂ /><br/>
 <input type="submit"  value="enregistrer ma commande" />

</form>

</div>

</body>
</html>
```

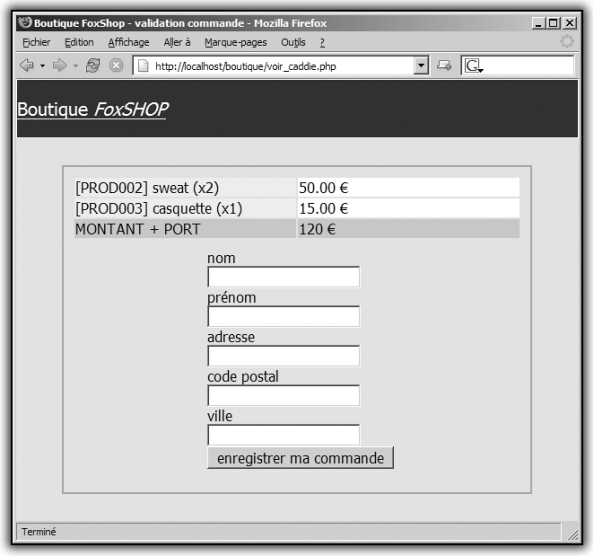

Figure 15.9 : *Le même produit est désormais listé deux fois*

Passez maintenant à l'enregistrement de la commande :

Listing 15-19 : Le script enregistre_commande.php

```
<?php

include("variables.inc.php");

if (empty($_REQUEST['nom']) || empty($_REQUEST['prenom']) ||
    empty($_REQUEST['adresse']) || empty($_REQUEST['cp']) ||
    empty($_REQUEST['ville']))
    die("ERREUR : tous les champs doivent être remplis.");

$liendb = mysql_connect($bddserver, $bddlogin,
✂ $bddpassword);
mysql_select_db ($bdd);

$date = date("Y-m-d G:i:s");

$_COOKIE['monpanier'][0] = ' ';

$tab_prod = split(",",$_COOKIE['monpanier']);

$i = 0;
while ($id = $tab_prod[$i]) {
  $sql = "SELECT * FROM $table_produit WHERE idproduit =
  ✂ '$id'";
  $resultat = mysql_query ($sql);
  $produit = mysql_fetch_array ($resultat);
  $montant += $produit['prix'];
  $listeproduits .= ',' .  $produit['reference'];
  $i++;
}
$listeproduits[0] = ' ';
$montant += 5;
$date = date("Y-m-d G:i:s");
$sql = "INSERT INTO $table_commande (produits, montant,
✂ nom, prenom, adresse, date) VALUES ('"
✂ .$_REQUEST['listeproduits']."', '".$_REQUEST['montant'
✂ ]."', '".$_REQUEST['nom']."', '".$_REQUEST['prenom']."',
'".$_REQUEST['adresse']."\n".$_REQUEST['cp']."\n"
✂ .$_REQUEST['ville']."', '$date')";

mysql_query ($sql);

mysql_close($liendb);

setcookie("monpanier","",time()-3600);

header("Location: $url/boutique.php");

?>
```

La suppression du cookie est réalisée avec l'instruction suivante :

```
setcookie("monpanier","",time()-3600);
```

Choix du procédé

Nous préférons, pour supprimer le cookie, cette dernière syntaxe à la syntaxe `setcookie("monpanier")`, car elle est plus radicale. L'idée ici est de mettre une date d'expiration dans le passé pour s'assurer que le navigateur efface bien le cookie.

La mini-boutique est donc achevée, les commandes sont bien enregistrées dans la base.

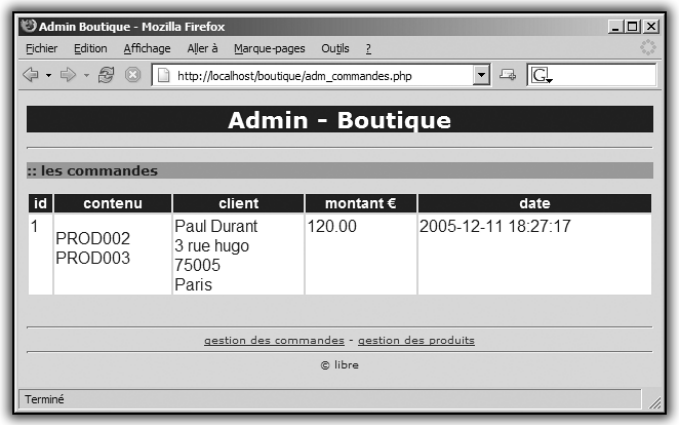

Figure 15.10 : *Administration des commandes*

Est-ce satisfaisant ? Certainement pas. Imaginez ce que pourrait entraîner le fait de taper l'URL suivante : `http://localhost/boutique/enregistre_commande.php?montant=10&listeproduits=PROD001,PROD002,PROD002&nom=Daunou&prenom=Sophie&adresse=41+rue+voltaire&cp=75002&ville=Paris`.

Cette commande permet de valider une commande de trois produits pour un montant de 10 euros ! L'exemple montre bien que, dès que vous réalisez le moindre applicatif à vocation publique, il est impératif, à un moment donné, de se mettre dans l'état d'esprit d'une personne aux intentions douteuses.

L'erreur ici est de passer directement en paramètre le montant de la commande. Le fait d'avoir la liste des produits permet cependant

d'éviter cela. C'est le script *enregistre_commande.php* qui doit calculer lui-même le montant de la commande. Comme vous allez vous baser sur le contenu du cookie, vous pouvez aussi vous débarrasser du paramètre `listeproduits`. Les paramètres `hidden`, du script *voir_caddie.php*, peuvent donc être supprimés :

Listing 15-20 : Version sécurisée de enregistre_commande.php

```
<?php

include("variables.inc.php");

if (empty($_REQUEST['nom']) || empty($_REQUEST['prenom']) ||
    empty($_REQUEST['adresse']) || empty($_REQUEST['cp']) ||
    empty($_REQUEST['ville']))
    die("ERREUR : tous les champs doivent être remplis.");

$liendb = mysql_connect($bddserver, $bddlogin, $bddpassword);
mysql_select_db ($bdd);

$date = date("Y-m-d G:i:s");

$_COOKIE['monpanier'][0] = ' ';

$tab_prod = split(",",$_COOKIE['monpanier']);

$i = 0;
while ($id = $tab_prod[$i]) {
  $sql = "SELECT * FROM $table_produit WHERE idproduit = '$id'";
  $resultat = mysql_query ($sql);
  $produit = mysql_fetch_array ($resultat);
  $montant += $produit['prix'];
  $listeproduits .= ',' .  $produit['reference'];
  $i++;
}

$listeproduits[0] = ' ';

$montant += 5;

$date = date("Y-m-d G:i:s");

$sql = "INSERT INTO $table_commande (produits, montant,
✂ nom, prenom, adresse, date) VALUES ('$listeproduits',
✂ '$montant', '".$_REQUEST['nom']."', '"
✂ .$_REQUEST['prenom']."', '".$_REQUEST['adresse']."\n"
✂ .$_REQUEST['cp']."\n".$_REQUEST['ville']."', '$date')";

mysql_query ($sql);
```

```
mysql_close($liendb);

setcookie("monpanier","",time()-3600);

header("Location: $url/boutique.php");

?>
```

Dans le cadre de cette application, les cookies vont servir également à stocker le profil client et lui préremplir son formulaire d'identification en cas de nouvel achat.

L'idée est donc de créer un nouveau cookie, `$_COOKIE['monprofil']`, qui contient toutes les informations des clients. La création doit se faire lors de l'enregistrement de la commande dans *enregistre_commande.php*. Vous allez choisir la norme `nom1=valeur1;;nom2=valeur2;;…` pour stocker les données et ajouter la ligne suivante dans *enregistre_commande.php* :

```
setcookie("monprofil","nom=".$_REQUEST['nom'].";;prenom="
✂ .$_REQUEST['prenom'].";;adresse=".$_REQUEST['adresse']
✂ .";;cp=".$_REQUEST['cp'].";;ville=".$_REQUEST['ville']
✂ ."",time()+604800);
```

Norme pour le stockage de données

Il est courant en PHP de stocker les données dans son propre format. Veillez alors à ne pas utiliser de caractères trop répandus comme caractères de séparation afin d'éviter tout risque de conflit avec l'une des valeurs. Dans l'exemple suivant, si vous choisissez la virgule comme caractère de séparation, vous serez bien ennuyé pour différencier la virgule de séparation de la virgule présente dans l'adresse : `adresse=9, rue Jean-Jaurès,CP=75015,ville=Paris`.

La deuxième étape consiste à préremplir le formulaire. La seule difficulté est de récupérer chaque caractéristique du profil. Il faut utiliser une première fois la fonction `split()` avec deux points-virgules (`";;"`) comme premier paramètre et une deuxième fois avec le signe égal (=).

Listing 15-21 : Le formulaire est désormais pré rempli

```
<?php
include("variables.inc.php");
?>

<html>
```

```
 <head>
  <title>Boutique FoxShop - validation commande</title>
  <link href="look.css" rel="stylesheet" type="text/css" />
 </head>
<body>

<div class='titre'><a href='boutique.php'>Boutique
✂ <i>FoxSHOP</i></a></div>

<div class='caddie'>

<?php
$montant = 0;
$listeproduits = " ";
$_COOKIE['monpanier'][0] = ' ';
$liendb = mysql_connect($bddserver, $bddlogin, $bddpassword);
mysql_select_db ($bdd);
$sql = "SELECT * FROM $table_produit ".
       "WHERE idproduit IN (".$_COOKIE['monpanier'].")";
$resultat = mysql_query ($sql);
print("<table width='100%'>");
$tab = array_count_values(split(",",$_COOKIE['monpanier']));
while ($prod = mysql_fetch_array ($resultat)) {
  print("<tr><td class='produit'>");
  print("[".$prod['reference']."] ".$prod['nom']);
  print(" (x".$tab[$prod['idproduit']].")");
  print("</td><td class='montant'>");
  print($prod['prix']."  ");
  print("</td></tr>");
  $montant += $prod['prix']*$tab[$prod['idproduit']];
  $listeproduits .= ','.$prod['reference'];
}
$listeproduits[0] = ' ';
// frais de port
$montant += 5;
print("<tr><td class='total'>MONTANT + PORT</td>");
print("<td class='total'>$montant  </td></tr>");
print("</table>");
mysql_close($liendb);
?>

<form action="enregistre_commande.php" method="post">

 <input type="hidden" name="montant" value="<?php echo
 ✂ $montant; ?>">
 <input type="hidden" name="listeproduits"
        value="<?php echo $listeproduits; ?>">

<?php
if (!empty($_COOKIE['monprofil'])) {
  $tab_tmp = split(";;",$_COOKIE['monprofil']);
```

```
  $i = 0;
  while ($tab_tmp[$i]) {
     list ($nom,$val) = split("=",$tab_tmp[$i]);
     $tab_profil[$nom] = $val;
     $i++;
  }
}
?>

 <label>nom</label><br/>
 <input type="text" name="nom" value="<?php echo
 $tab_profil['nom']?>" />
 <br/><label>prénom</label><br/>
 <input type="text" name="prenom"
        value="<?php echo $tab_profil['prenom']?>" />
 <br/><label>adresse</label><br/>
 <input type="text" name="adresse"
        value="<?php echo $tab_profil['adresse']?>" />
 <br/><label>code postal</label><br/>
 <input type="text" name="cp"
        value="<?php echo $tab_profil['cp']?>" />
 <br/><label>ville</label><br/>
 <input type="text" name="ville"
        value="<?php echo $tab_profil['ville']?>" />
 <br/><input type="submit"  value="enregistrer ma commande" />

</form>

</div>

</body>
</html>
```

Plutôt qu'un deuxième tableau temporaire pour recueillir les données issues du deuxième `split()`, utilisez la fonction `list` qui vous permet de placer directement les données dans deux variables.

Si l'internaute n'est pas encore client, le tableau `$tab_profil` est vide. De ce fait, afficher `$tab_profil['nom']` n'est pas gênant, car cela équivaut à ne rien afficher. Si, par contre, il est déjà client, `$tab_profil['nom']` contiendra le nom qu'il avait spécifié lors de son dernier achat. Quand c'est possible, comme ici, il est intéressant de ne pas multiplier les cas (avec des `if`…) et d'essayer d'être le plus générique possible.

15.2. Les sessions

Vous venez de voir que les cookies permettent de transmettre de l'information d'une page à l'autre. Certains inconvénients peuvent cependant être notés :

- l'obligation de les gérer en haut du script avant que des données ne soient affichées ;
- le format assez primaire de stockage de l'information (une chaîne de caractères).

Pour répondre à ces limitations, le langage PHP vous propose d'utiliser un système de session. À n'importe quel endroit de votre script, il est possible de définir certaines variables en tant que variable de session. Une fois enregistrées, ces variables sont propagées d'une page à l'autre. Elles deviennent schématiquement des variables globales à l'ensemble du site, tout en restant associées à un visiteur donné. Si vous modifiez le contenu d'une variable de session dans un script *A*, vous récupérez le contenu modifié dans un script *B*. Il est important de signaler que le nombre de variables de session n'est pas limité : vous pouvez en enregistrer autant que nécessaire.

Le mode de fonctionnement des sessions est extrêmement simple. Il s'appuie sur :

- la fonction `session_start()` qui indique au script que vous souhaitez récupérer les variables de session.
- la variable super-globale `$_SESSION` qui contient l'ensemble des variables de session.

```
$_SESSION['x'] = 2;
```

Réalisez ce petit exemple, qui permet de savoir combien de fois une personne est venue sur une page :

Listing 15-22 : Exemple simple d'utilisation des sessions

```
<?php

session_start();

if (!isset($_SESSION['visite'])) {
  echo "première visite";
  $_SESSION['visite'] = 1;
}
else {
```

```
  $_SESSION['visite']++;
  echo "vous avez visité cette page ".$_SESSION['visite'
  ✂ ]." fois";
}

?>
```

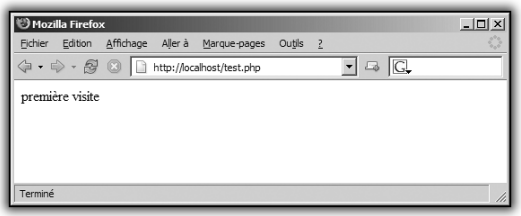

Figure 15.11 : *Message apparaissant en arrivant pour la première fois sur la page*

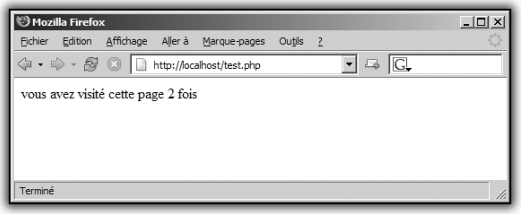

Figure 15.12 : *Message apparaissant lorsque la page est rafraîchie*

Analysons cet exemple…

Dans ce script, vous souhaitez faire usage de variables de session. Vous l'indiquez à PHP en exécutant la fonction `session_start()`. Votre seule contrainte est de faire appel à cette fonction avant le premier affichage du script.

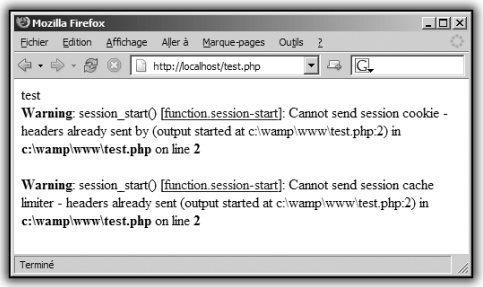

Figure 15.13 : *Erreur générée à la suite d'un affichage de la chaîne "test" avant l'appel à session_start()*

Vous testez ensuite si la variable `$_SESSION['visite']` (qui va contenir le nombre de vos visites) existe. Si ce n'est pas le cas, vous affichez le message `"première visite"` et initialisez la variable de session `"visite"` à `1`. À partir de ce moment, toute modification de sa valeur sera propagée d'une page à l'autre.

En rafraîchissant la page, vous vous retrouvez dans le cas où la variable `$_SESSION['visite']` existe. Lors de chaque accès à cette page, vous faites augmenter le compteur de visites en incrémentant directement la variable `$_SESSION['visite']`.

Bien évidemment, si vous placez ce même code dans un autre script, la variable `$_SESSION['visite']` sera aussi incrémentée. La variable de session est en effet partagée par tous les scripts d'un même site.

Un autre grand avantage des variables de session est de permettre d'enregistrer des variables de type complexe comme des tableaux. Vous pouvez voir l'extrême intérêt de cette fonctionnalité dans le cadre de votre applicatif. Plutôt que de stocker vos données en les séparant par des virgules (`valeur1,valeur2`, etc.) ou en utilisant une norme boiteuse (`nom1=valeur1;;nom2=valeur2;;…`), autant utiliser dans le premier cas un tableau scalaire, et, dans le deuxième cas, un tableau associatif.

Modifiez les fichiers *ajout_caddie.php*, *boutique.php*, *voir_caddie.php*, *enregistre_commande.php* afin d'utiliser des sessions plutôt que des cookies.

Commencez par l'ajout au panier :

Listing 15-23 : Le script ajout_caddie.php

```
<?php

include("variables.inc.php");

session_start();

if (!isset($_SESSION['monpanier'])) $_SESSION['monpanier']
✂ = array();

$_SESSION['monpanier'][] = $_REQUEST['id'];

header("Location: $url/boutique.php?id=".$_REQUEST['id']);

?>
```

Listing 15-24 : Le script boutique.php

```
<?php

include("variables.inc.php");
session_start();

if (!isset($_REQUEST['id'])) $id = 1;
else $id = $_REQUEST['id'];

?>

<html>
 <head>
  <title>Boutique FoxShop - catalogue</title>
  <link href="look.css" rel="stylesheet" type="text/css" />
 </head>
<body>

<div class='titre'><a href='boutique.php'>Boutique
✂ <i>FoxSHOP</i></a></div>

<table class='catalogue'>
<tr>
 <td class='liste'>
  <div class='tdTitre'>Nos produits</div>

<?php
$liendb = mysql_connect($bddserver, $bddlogin,
✂ $bddpassword);
mysql_select_db ($bdd);
$sql = "SELECT * FROM $table_produit";
$resultat = mysql_query ($sql);
while ($produit = mysql_fetch_array ($resultat)) {
  print("  - ");
  print("<a href=".$_SERVER['PHP_SELF']."?id=".$produit
  ✂ ['idproduit']."><". $produit['nom']."</a>");
  print("<br/>");
}
?>

 </td>
 <td class='detail'>

<?php
$sql = "SELECT * FROM $table_produit WHERE idproduit = '$id'";
$resultat = mysql_query ($sql);
$produit = mysql_fetch_array ($resultat);
print("<div class='tdTitre'>".$produit['nom'].
      " [ref#".$produit['reference']."]</div>");
?>
```

```
<div class='description'>

<?php
print(nl2br($produit['description'])."<br/><br/>");
print($produit['prix']."  <br/><br/>");
mysql_close($liendb);
?>

<form action="ajout_caddie.php" method="post">
 <input type="hidden" name="id" value="<?php echo $id; ?>" />
 <input type="submit" value="ajouter au panier" />
</form>

<?php

if (isset($_SESSION['monpanier']))
{
 print("<div class='panier'>");
 $nb_prod = count($_SESSION['monpanier']);
 print("votre panier contient ".$nb_prod." produit(s)<br/>");
 print("<form action='voir_caddie.php' method='post'>");
 print("<input type='submit' value='valider la commande'/>
 ✂ </form>");
 print("session = {".implode(",",$_SESSION['monpanier'])."}");
 print("</div>");
}

?>

</div>
 </td>
</tr>
</table>

</body>
</html>
```

Le passage aux sessions a globalement clarifié et simplifié le code. Les différents produits ajoutés au panier sont désormais stockés dans une variable de session nommée `$_SESSION['monpanier']`, de type tableau scalaire.

Passez maintenant à la validation et à l'enregistrement de la commande :

Listing 15-25 : Le script voir_caddie.php

```
 <?php

include("variables.inc.php");
```

```
session_start();

?>

<html>
 <head>
  <title>Boutique FoxShop - validation commande</title>
  <link href="look.css" rel="stylesheet" type="text/css" />
 </head>
<body>

<div class='titre'><a href='boutique.php'>Boutique
✂ <i>FoxSHOP</i></a></div>

<div class='caddie'>

<?php
$montant = 0;
$listeproduits = " ";
$liendb = mysql_connect($bddserver, $bddlogin, $bddpassword);
mysql_select_db ($bdd);
$sql = "SELECT * FROM $table_produit ".
       "WHERE idproduit IN (".implode(',',$_SESSION
       ✂ ['monpanier']).")";
$resultat = mysql_query ($sql);
print("<table width='100%'>");
$tab = array_count_values($_SESSION['monpanier']);
while ($prod = mysql_fetch_array ($resultat)) {
  print("<tr><td class='produit'>");
  print("[".$prod['reference']."] ".$prod['nom']);
  print(" (x".$tab[$prod['idproduit']].")");
  print("</td><td class='montant'>");
  print($prod['prix']."  ");
  print("</td></tr>");
  $montant += $prod['prix']*$tab[$prod['idproduit']];
  $listeproduits .= ','.$prod['reference'];
}
$listeproduits[0] = ' ';
// frais de port
$montant += 5;
print("<tr><td class='total'>MONTANT + PORT</td>");
print("<td class='total'>$montant  </td></tr>");
print("</table>");
mysql_close($liendb);
?>

<form action="enregistre_commande.php" method="post">

 <input type="hidden" name="montant" value="<?php echo
 ✂ $montant; ?>">
 <input type="hidden" name="listeproduits"
```

```
        value="<?php echo $listeproduits; ?>">

 <label>nom</label><br/>
 <input type="text" name="nom"
        value="<?php echo $_SESSION['profil']['nom']?> "/>
 <br/><label>prénom</label><br/>
 <input type="text" name="prenom"
        value="<?php echo $_SESSION['profil']['prenom']?>" />
 <br/><label>adresse</label><br/>
 <input type="text" name="adresse"
        value="<?php echo $_SESSION['profil']['adresse']?>" />
 <br/><label>code postal</label><br/>
 <input type="text" name="cp"
        value="<?php echo $_SESSION['profil']['cp']?>" />
 <br/><label>ville</label><br/>
 <input type="text" name="ville"
        value="<?php echo $_SESSION['profil']['ville']?>" />
 <br/><input type="submit"  value="enregistrer ma commande" />

</form>

</div>

</body>
</html>
```

Listing 15-26 : Le script enregistre_commande.php

```
<?php

include("variables.inc.php");

session_start();

if (empty($_REQUEST['nom']) || empty($_REQUEST['prenom']) ||
    empty($_REQUEST['adresse']) || empty($_REQUEST['cp']) ||
    empty($_REQUEST['ville']))
    die("ERREUR : tous les champs doivent être remplis.");

$liendb = mysql_connect($bddserver, $bddlogin, $bddpassword);
mysql_select_db ($bdd);

$date = date("Y-m-d G:i:s");

$i = 0;
foreach ($_SESSION['monpanier'] as $i) {
  $sql = "SELECT * FROM $table_produit WHERE idproduit = '$id'";
  $resultat = mysql_query ($sql);
  $produit = mysql_fetch_array ($resultat);
  $montant += $produit['prix'];
  $listeproduits .= ',' .  $produit['reference'];
```

```
  $i++;
}
$listeproduits[0] = ' ';
$montant += 5;

$sql = "INSERT INTO $table_commande (produits, montant,
✂ nom, prenom, adresse, date) VALUES ('"
✂ .$_REQUEST['listeproduits']."', '".$_REQUEST['montant'
✂ ]."', '".$_REQUEST['nom']."', '".$_REQUEST['prenom']."',
'".$_REQUEST['adresse']."\n".$_REQUEST['cp']."\n"
✂ .$_REQUEST['ville']."', '$date')";

mysql_query ($sql);

mysql_close($liendb);

session_unregister('monpanier');

$_SESSION['profil']['nom']     = $_REQUEST['nom'];
$_SESSION['profil']['prenom']  = $_REQUEST['prenom'];
$_SESSION['profil']['adresse'] = $_REQUEST['adresse'];
$_SESSION['profil']['cp']      = $_REQUEST['cp'];
$_SESSION['profil']['ville']   = $_REQUEST['ville'];

header("Location: $url/boutique.php");

?>
```

Dans *voir_caddie.php*, vous pouvez parcourir le contenu du panier directement avec la variable `$_SESSION['monpanier']`. Il en est de même pour le préaffichage des informations client. Tout est stocké dans la variable de session `$_SESSION['monprofil']`.

Vous remarquez l'usage d'une nouvelle fonction dans *enregistre_commande.php* : `session_unregister()`. Cette fonction permet tout simplement de supprimer la variable de session dont le nom est passé en paramètre.

La fonction `session_destroy()`
Cette fonction va plus loin que `session_unregister()` : elle vous permet de supprimer toutes les variables de session.

Les sessions sont donc un outil très puissant. Elles peuvent vous faciliter grandement la tâche pour le développement d'applications en ligne. Il ne faut cependant pas croire que les sessions pallient tous les défauts des

cookies. Une erreur courante consiste à se dire qu'avec les sessions vous n'avez plus à vous soucier des internautes qui n'acceptent pas les cookies. Ce raisonnement est complètement faux car les sessions utilisent en fait directement les cookies :

Listing 15-27 : La fonction session_id() retourne la valeur de l'identifiant de session unique qui a été alloué

```
<?php
session_start();
echo "Identifiant de session : ".session_id();
?>
```

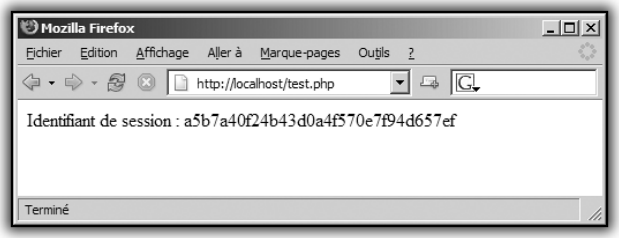

Figure 15.14 : *Valeur de l'identifiant de session*

Les variables de session sont en réalité stockées sur le serveur web et sont reconnues grâce à cet identifiant unique. Cet identifiant est stocké dans un cookie sur votre disque. Une personne qui n'accepte pas les cookies ne peut propager l'`id` de session qui lui est transmis, et ne peut donc pas profiter des sessions.

Ce mode de fonctionnement permet en revanche d'écarter les limitations des cookies concernant leur nombre et leur taille. Comme les données des sessions sont stockées sur le serveur, ces limitations sautent :

Listing 15-28 : Pour les plus curieux, la fonction session_save_path() retourne l'endroit où sont stockées les données des sessions sur le server (souvent /tmp sous Unix/Linux)

```
<?php
session_start();
echo session_save_path();
?>
```

Par défaut, une session est détruite quand l'internaute ferme son navigateur. Vous devinez donc que le cookie qui est créé pour stocker l'`id` de session est un cookie de session. Vous pouvez en avoir la preuve en utilisant la fonction `session_get_cookie_params()`, qui retourne un tableau contenant tous les paramètres du cookie de session :

Listing 15-29 : Affichage des propriétés du cookie de session

```
<?php
session_start();
print_r(session_get_cookie_params());
?>
```

Figure 15.15 : *Propriétés du cookie de session*

Il est possible de faire en sorte qu'une session subsiste à la fermeture du navigateur. Utilisez pour cela la fonction `session_set_cookie_params()` qui permet de modifier les propriétés du cookie de session. Transformez ce petit exemple de compteur de manière à ce que la session soit conservée une semaine :

Listing 15-30 : Le compteur conserve désormais la bonne valeur, même si vous quittez le navigateur

```
<?php

session_set_cookie_params(time()+604800);
include("variables.inc.php");

session_start();

if (!isset($_SESSION['monpanier'])) $_SESSION['monpanier']
✂ = array();

$_SESSION['monpanier'][] = $_REQUEST['id'];

header("Location: $url/boutique.php?id=".$_REQUEST['id']);

?>
```

La fonction `session_set_cookie_params()` devant être appelée avant chaque appel à `session_start()`, l'idéal est de placer l'appel à ces deux fonctions dans *variables.inc.php*.

Listing 15-31 : Les sessions sont maintenant démarrées dans variables.inc.php

```
<?php

$bddserver  =  "localhost";
$bddlogin  =  "root";
$bddpassword  =  "";
$bdd  =  "test";
$table_eleve  =  "eleve";
$table_exam  =  "exam";
$url  =  "http://localhost";

session_set_cookie_params(time()+604800);
include("variables.inc.php");

?>
```

Cette option peut être intéressante si vous souhaitez mettre en place un système de panier permanent dans votre boutique : le client retrouve son panier lorsqu'il se reconnecte au site marchand.

Transmission de l'identifiant de session par URL

L'identifiant de session peut également être transmis par l'URL. Cette technique a l'avantage de fonctionner avec tous les internautes (qu'ils acceptent les cookies ou pas). Elle souffre cependant des défauts suivants :

- Elle est beaucoup plus lourde à mettre en place.
- Elle demande plus de ressources au niveau serveur.
- Elle nécessite que l'interpréteur PHP soit compilé avec certaines options.

15.3. Check-list

- Les cookies et les sessions permettent de propager des données liées à un internaute durant toute la durée de sa visite.
- Les sessions ont l'avantage d'être plus simples à gérer que les cookies. Une session est composée d'un cookie contenant son identifiant et d'un fichier sur le serveur contenant les données.
- Les variables `$_SESSION` et `$_COOKIE` sont utilisées pour y avoir accès.
- Longtemps décriés, les cookies sont désormais entrés dans les mœurs du Web.

Chapitre 16

La gestion de la sécurité

La popularité des applications web a non seulement attiré les utilisateurs et les développeurs, mais également les pirates informatiques (souvent appelés *hackers*). Les sites spécialisés en sécurité tels que **phpsec.org**, **secunia.com** ou **www.securityfocus.com** signalent désormais quotidiennement des failles dans des outils tels que le forum phpBB (**www.phpbb.com**), l'outil de gestion de contenu (CMS) PHP-Nuke (**phpnuke.org**), la plateforme d'e-commerce osCommerce (**www.oscommerce.com**), le gestionnaire de blog Wordpress (**wordpress.org**) et plusieurs centaines d'autres.

PHP ne souffre en lui-même d'aucun problème de sécurité. Le souci vient plutôt du fait qu'il est extrêmement facile de laisser des failles de sécurité au cœur d'applications écrites d'une part pour le Web et d'autre part en PHP. Listons quelques-uns des aspects les plus problématiques :

- PHP est un langage extrêmement facile à prendre en main. Un débutant sans connaissance préliminaire en sécurité peut réaliser et mettre en ligne un applicatif en quelques jours (voir en quelques heures).
- La sécurité d'une application web impose des connaissances informatiques relativement larges : serveur web, PHP, Javascript, authentification, base de données, sessions, cookies, protocoles, système d'exploitation (droits, chemins), etc.
- Pour une application en ligne, le nombre potentiel d'utilisateurs malintentionnés est aussi gigantesque qu'incontrôlable.
- Les techniques mises en œuvre pour exploiter les failles sont souvent triviales et, par conséquent, accessibles à de jeunes gens en mal d'amusement (les *scripts kiddies*). Nous sommes loin de la décompilation et de l'assembleur qui ne permettaient qu'à un tout petit nombre de pirates de « cracker » des logiciels.
- Certains défauts du langage PHP ont des implications directes dans la nature faillible des applications dont il est à l'origine : les nombreuses façons de transmettre des données (`input`, session, cookies), l'absence de « composants », l'inconsistance des différents noms (variables, objets, fonctions, etc.), la présence de variables globales rendant la relecture du code ardue et les tests d'unité pour le moins complexes.

Ce chapitre vise à vous présenter les failles potentielles les plus répandues qui peuvent s'immiscer au sein d'une application. Plus que

cela, il doit vous faire prendre conscience que la sécurisation d'un code est une étape indispensable et essentielle lors du développement d'un applicatif.

16.1. La sécurité avec PHP

Commençons par étudier les différents points devant être surveillés au sein de vos scripts PHP.

Le b-a ba

Certains développeurs ont tendance à concevoir leurs applicatifs en pensant qu'un certain nombre de tâches pourront être réalisées à la fin du projet. Cette pratique est généralement à proscrire car, comme dans beaucoup d'autres domaines, la fin d'un projet se déroule souvent dans l'urgence et le stress.

Il est donc conseillé de ne pas attendre l'issue du projet pour initialiser de manière cohérente vos différents mots de passe et ne pas laisser des accès de type `"admin/admin"`, `"test/test"`. Pour rappel, un mot de passe doit contenir au minimum huit caractères et inclure des caractères numériques.

Dictionnaires et brute force

N'allez surtout pas croire qu'un hacker souhaitant trouver le mot de passe de votre site essaiera successivement et péniblement plusieurs dizaines de possibilités. Il existe aujourd'hui des dictionnaires de mots et des scripts permettant en quelques minutes de tester plusieurs centaines de milliers de possibilités. Ces dictionnaires disposent également de tous les prénoms, marques d'alcools, noms de planètes et divers termes souvent utilisés en informatique pour initialiser un mot de passe.

L'échec d'une attaque lancée à l'aide d'un dictionnaire ne rebutera cependant pas un pirate motivé. L'attaque dite *brute force* consiste à tester toutes les combinaisons de caractères possibles les unes après les autres. Cette attaque peut durer des heures, voire des journées, mais n'oubliez pas que le pirate n'a pas à attendre devant sa machine, qu'aujourd'hui les machines sont extrêmement puissantes et que les liaisons Internet sont quasi gratuites.

Il peut donc être malin d'interdire un trop grand nombre de tentatives d'accès successives à une zone sécurisée. Il convient pour cela d'enregistrer chaque tentative (accompagnée de son adresse IP) dans une base de données et de vérifier, avant d'autoriser l'accès, qu'il n'y a pas eu plus de n tentatives depuis cette même adresse auparavant.

Mise à jour de PHP

Disposer d'un interpréteur PHP en permanence à jour est crucial pour votre applicatif. Chaque nouvelle version de PHP apporte en effet son lot de nouvelles fonctionnalités, de corrections de bugs et surtout de corrections de failles de sécurité. Bien que ces failles soient la plupart du temps mineures, il peut arriver que certaines soient exploitables de façon distante. Vous ne pourrez alors rien y faire, quelles que soient vos compétences de programmeur.

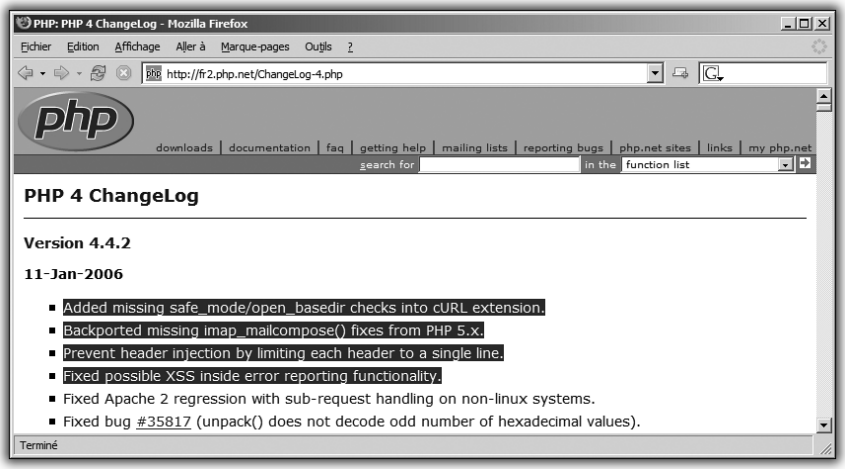

Figure 16.1 : *Exemple de lignes à prendre en compte dans le rapport de mises à jour d'une nouvelle version de PHP*

La fonction `phpversion()` ou la constante `PHP_VERSION` peuvent être utilisées pour obtenir la version de l'interpréteur.

Initialiser toutes les variables

Cette pratique consiste à donner une valeur à toutes les variables que vous utilisez au sein de votre script.

```
if ($_POST['identifiant']=='sys' && $_POST['pass']=='sys') {
  $status_admin = true;
}
```

Ce code n'est pas correct dans la mesure où la variable `$status_admin` n'a pas été initialisée.

La version correcte est la suivante :

```
$status_admin = false;
if ($_POST['identifiant']=='sys' && $_POST['pass']=='sys') {
  $status_admin = true;
}
```

Le statut d'administrateur est faux si l'authentification n'a pas été réalisée ou si elle échoue. Dans le premier code, son statut ne serait pas défini.

CURL

Les programmeurs débutants ont souvent tendance à penser que les tentatives de piratage ne passent que par la méthode `GET`. Cette supposition est complètement fausse. Des outils tels que CURL (**curl.haxx.se**) permettent ainsi de réaliser des scripts qui simulent l'envoi de paramètres en mode `POST`. CURL permet même de simuler la transmission de cookies. La page de documentation est disponible à l'adresse **http://curl.haxx.se/docs/manpage.html**.

De la même manière, le changement des mots de passe par défaut est essentiel. Ainsi n'oubliez surtout pas de modifier le mot de passe `root` de MySQL.

Utiliser les constantes

La plupart des attaques visent à exploiter une faille qui permettra de modifier une variable utilisée dans un endroit sensible du script (ex: inclusion, exécution etc.). L'utilisation des constantes, dont le contenu n'est pas modifiable par définition, permet de rendre la tâche du pirate plus ardue.

Se méfier de la puissance de certaines fonctions

La fonction `extract()` permet d'importer dans la table générale des variables le tableau qui lui est passé en paramètre. Voyons tout de suite un exemple qui devrait clarifier les choses :

Listing 16-1 : utilisation de la fonction extract()

```
print("bonjour : <br/>");
print("hello : <hr/>");
$tab = array('bonjour'=>'monde', 'hello'=>'world');
print_r($tab);
print("<hr/>");
extract($tab);
print("bonjour : $bonjour<br/>");
print("hello : $hello<br/>");
```

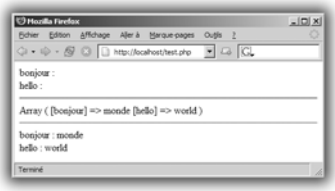

Figure 16.2 :
Les cellules du tableau deviennent accessibles en tant que variables

Vous comprenez alors l'importance d'être sûr de son tableau avant d'appeler la fonction `extract()`. Un internaute qui parviendrait à modifier le tableau passé en paramètre pourrait écraser toutes les variables de l'application (par exemple `$status_admin`).

La fonction `extract()` peut également prendre un second paramètre qui indique le comportement à avoir quand une variable est déjà définie avant l'appel à `extract()`. Nous vous conseillons d'utiliser la valeur `EXTR_SKIP` qui interdit l'écrasement d'une variable préexistante.

```
$bonjour = 'coucou';
print("bonjour : $bonjour<br/>");
print("hello : $hello<hr/>");
$tab = array('bonjour'=>'monde', 'hello'=>'world');
print_r($tab);
print("<hr/>");
extract($tab,EXTR_SKIP);
print("bonjour : $bonjour<br/>");
print("hello : $hello<br/>");
```

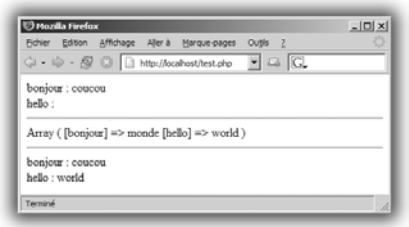

Figure 16.3 : *La variable $bonjour conserve sa valeur*

Dangers de la fonction mail

Il est courant de composer un mail à partir d'éléments provenant d'un formulaire. L'adresse email est souvent utilisée pour composer l'en-tête `From:` et permettre au destinataire du message, de répondre directement à l'internaute.

Listing 16-2 : Les paramètres du formulaire sont directement utilisés dans le mail

```
mail("fx@domain.fr","sujet","contenu","From:
✂ {$_REQUEST['email']}");
```

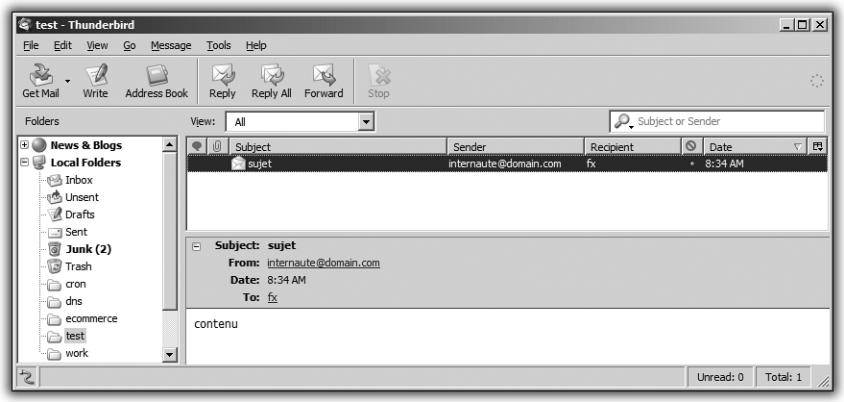

Figure 16.4 : *L'origine du destinataire apparaît bien : internaute@domain.com*

Notre erreur consiste ici à ne pas vérifier le paramètre avant de l'inclure dans le message. En effet, les spammers parviennent désormais à exploiter les millions de formulaires présents sur le web pour envoyer des messages non sollicités. Leur technique consiste à transmettre au script PHP non pas une adresse email mais une adresse email avec d'autres directives exploitables dans l'en-tête du mail.

En transmettant à notre script les paramètres suivants :

```
email=dest%40domain.com%0D%0ACc%3Adest2%40domain.com
```

le spammer initialise à la fois les en-têtes `From:` et `Cc:` du mail.

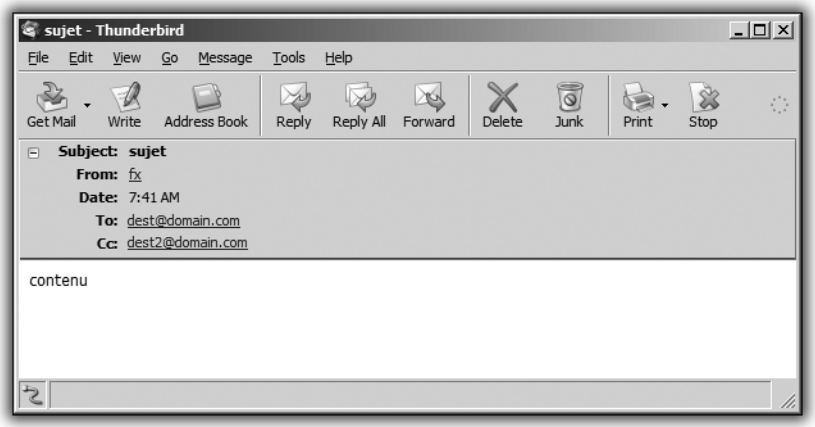

Figure 16.5 : *dest2@domain.com va recevoir un email de spam*

Pour être valide, notre script doit donc vérifier que le paramètre email ne contient pas de retour à la ligne.

Listing 16-3 : Vérification de la validité du paramètre email

```
if (strpos($_REQUEST['email'],"\n")!=false ||
    strpos($_REQUEST['email'],"\r")!=false)
  exit (1);
else
  mail("fx@domain.fr","sujet","contenu","From:
  ✂ {$_REQUEST['email']}");
```

Les cookies et les sessions

Aussi bien les cookies que les sessions prennent une place tous les jours plus importante dans les applications web en tant que point central du système d'authentification.

Il n'est donc pas étonnant que le cookie d'un internaute soit si convoité par les pirates. Il s'avère en plus que les techniques existent et qu'elles ne sont pas si complexes à mettre en place.

Le fonctionnement avec un passage de l'identifiant de session en paramètre est par exemple très dangereux. Prenez l'exemple d'un webmail fonctionnant sur ce schéma. Une URL typique au sein de

l'application pourrait être : **www.monmail.com/lire.php?PHPSESSID=1234-5678&num_msg=ABC**. Elle pourrait correspondre à la lecture du message ABC par le visiteur dont la session porte l'identifiant 1234-5678. Supposons maintenant que le courriel lu comporte dans son contenu un lien vers le site **monsite.com** et que l'internaute clique dessus. Si les développeurs du webmail ne sont pas vigilants, l'administrateur de **monsite.com** peut se retrouver dans ses logs avec une visite dont l'origine est précisément **www.monmail.com/lire.php?PHPSESSID=1234-5678&num_msg=ABC**. En cliquant sur ce lien qui dispose donc de l'identifiant de session, ce cher administrateur aura la surprise de pouvoir lire un courriel qui ne lui est pas du tout adressé. Pour parer à ce genre de faille, nous recommandons donc de passer par un script dont le rôle se limiterait à une redirection et qui ne prendrait en paramètre que l'URL de destination et en aucun cas l'identifiant de session ; par exemple : **www.monmail.com/redir.php?url=http://www.monsite.com**.

Ces techniques sont souvent évoquées en tant que *session hijacking*.

L'IP dans la session : une fausse bonne idée

Pour parer aux failles décrites précédemment, le programmeur peut avoir envie de mettre en œuvre un système permettant de vérifier si l'internaute qui réclame la session est le même que celui qui en est à l'origine. Une idée pourrait donc consister à placer dans la session l'adresse IP de l'internaute qui vient de la créer et à vérifier systématiquement si cette même IP est identique à celle de l'internaute qui réclame la session. Cette idée qui peut paraître bonne à première vue est hélas invalide. Il convient en effet de savoir qu'un visiteur ne garde pas obligatoirement la même IP durant toute la durée de sa visite. Certains routeurs permettent en effet la connexion à deux fournisseurs d'accès et autorisent ainsi un passage alterné par l'un ou l'autre des fournisseurs. Si le routeur est connecté à Free ADSL et à Noos, l'internaute, d'une page à l'autre, apparaîtra tour à tour avec une IP Free et une IP Noos. Même si cet aspect n'existe que pour 1 % des internautes, il est à prendre en compte.

Les transferts de fichiers

Le fait de demander une photo n'implique pas nécessairement que l'internaute choisira un fichier de type image. S'il envoie au contraire un fichier PHP, il pourra alors exécuter son script de manière distante au sein même de votre compte et provoquer de véritables catastrophes.

Le test suivant permet de vérifier que le fichier transmis se termine bien par *.jpeg*, *.jpg*, *.gif* ou *.png* :

Listing 16-4 : le fichier est-il de type image ?

```
if (preg_match("/\.(jpeg|jpg|gif|png)$/i",
                $_FILES['userfile']['tmp_name'])) {
    echo "ok";
}
else {
    echo "ko";
}
```

Inclusion de fichier

Soyez extrêmement prudent avec les inclusions de fichiers faisant intervenir des données passées en paramètres. L'exemple suivant est à proscrire :

```
if ($_REQUEST['action']!='accueil')
✂ include($_REQUEST['action']);
else include("index.php");
```

Un pirate peut transmettre la valeur suivante `action=../../WINDOWS/win.ini` et obtenir le contenu de n'importe quel fichier présent sur le disque dur.

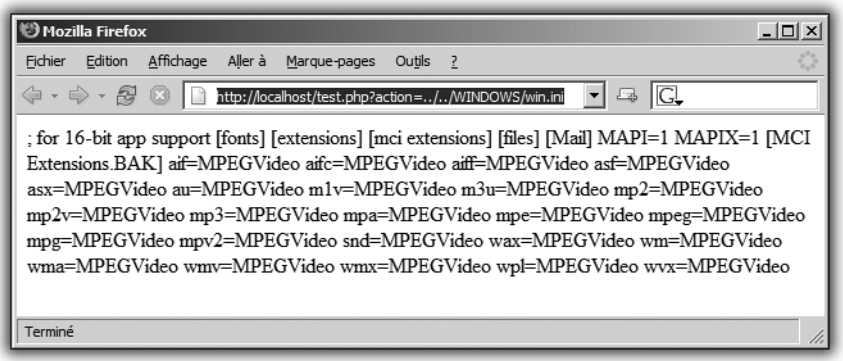

Figure 16.6 : *contenu du fichier win.ini présent dans C:\WINDOWS*

L'inclusion d'un fichier avec la fonction `include()` ne doit donc jamais exploiter directement une donnée transmise en paramètre.

16.2. Sécuriser les bases de données

Les failles liées aux bases de données sont aujourd'hui les plus répandues.

Les injections SQL

Le programmeur débutant ne réalise pas à quel point l'utilisation des bases de données impose une extrême prudence.

Étudiez le code suivant :

```
mysql_query("DELETE FROM matable WHERE ID = ".$_REQUEST
✂ ['monid']);
```

Cette ligne vise à supprimer de la table *matable* la ligne dont la clé est égale au paramètre `monid`. Si la variable `$_GET['monid']` contient la valeur `22`, la requête équivaut à ceci :

```
mysql_query("DELETE FROM matable WHERE ID = 22");
```

Supposons maintenant qu'une personne mal intentionnée transmette l'URL suivante : **http://127.0.0.1/supprime.php?monid=22+OR+ID+ %3E+0**.

Vous pouvez alors vous inquiéter car la requête devient :

```
mysql_query("DELETE FROM matable WHERE ID = 22 OR ID > 0");
```

Elle entraîne une suppression complète de la table *matable* ! Cette attaque est qualifiée d'injection SQL.

Deux habitudes importantes doivent être prises pour éviter cette situation :

- passer par la fonction `mysql_real_escape_string()` avant d'insérer des données extérieures dans la base (provenant d'un formulaire, d'un cookie etc.) ;
- utiliser les guillemets autour des données.

Listing 16-5 : version sécurisée de notre requête

```
mysql_query("DELETE FROM matable WHERE ID = '".
            mysql_real_escape_string($_REQUEST
            ['monid'])."'");
```

Les Cross Site Scripting

Ces attaques sont plus connues sous le terme générique de XSS. Le principe d'une attaque XSS est le suivant :

- Le pirate remplit un formulaire en y incluant des balises nocives. Les données du formulaire sont stockées dans une base de données sans être nettoyées préalablement.
- Un internaute accède au site (par exemple un forum), visualise l'article transmis par le pirate et subit l'assaut du pirate.

Les balises nocives dont nous parlons correspondent le plus souvent à des liens. Ces derniers sont écrits de manière à récupérer le cookie de l'internaute et à le transmettre au pirate.

Illustrons cela à l'aide d'un exemple. Le premier script enregistre les données dans une table *article* :

Listing 16-6 : ajout.php

```
<?php

if (!empty($_REQUEST['contenu'])) {
  $contenu = $_REQUEST['contenu'];
  $liendb = mysql_connect("localhost", "root", "");
  mysql_select_db("test");
  mysql_query("INSERT INTO article (contenu) VALUES
  ✂ ('$contenu')");
  print("enregistrement ok");
  return (true);
}

?>

<form method='post' action='ajout.php'>
<textarea style='width:100%' name='contenu'>
</textarea><br/>
<input type='submit' value='enregistrer'>
</form>
```

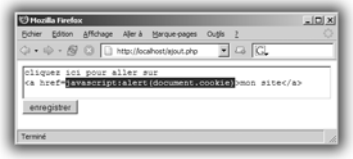

Figure 16.7 :
Texte renseigné par le pirate

Le second script permet d'afficher un article enregistré dans la base et de placer un cookie secret sur la machine de l'internaute.

Listing 16-7 : visualisation de l'article

```
<?php

setcookie('secret',rand(100,999));

$liendb = mysql_connect("localhost", "root", "");
mysql_select_db("test");
$sql = "SELECT * FROM article";
$resultat = mysql_query ($sql);
$article = mysql_fetch_array ($resultat);

?>

<div style='width:200px; border:2px solid black;
padding:6px; height:200px'>

<?php print($article['contenu']); ?>

</div>
```

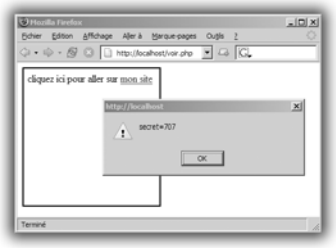

Figure 16.8 :
Apparition du cookie

En cliquant sur le lien de l'article, le code secret apparaît ! C'est grave car il va sans dire que dans la réalité les pirates font plutôt en sorte de se faire envoyer discrètement le code secret.

La parade classique pour ces attaques XSS consiste à protéger les données envoyées par les internautes à l'aide de la fonction `htmspecialchars()`. Cette fonction convertit les caractères `&`, `"`, `'`, `<` et `>` en leur équivalent HTML : `&`, `&quot;`, `&#039;`, `<` et `>`. Cela vous permet de ne pas altérer le contenu tout en vous protégeant.

Le script *ajout.php* doit donc être modifié de la façon suivante :

```
if (!empty($_REQUEST['contenu'])) {
  $contenu = htmlspecialchars($_REQUEST['contenu']);
  $liendb = mysql_connect("localhost", "root", "");
```

```
  mysql_select_db("test");
  mysql_query("INSERT INTO article (contenu) VALUES
  ✂ ('$contenu')");
  print("enregistrement ok");
  return (true);
}
```

Le même message est désormais enregistré de la manière suivante :

```
cliquez ici pour aller sur <a href=javascript:alert
✂ (document.cookie)>mon site</a>
```

Visuellement, vous obtenez un affichage correct et vous ne subissez plus cette attaque.

Figure 16.9 : *L'attaque apparaît au grand jour*

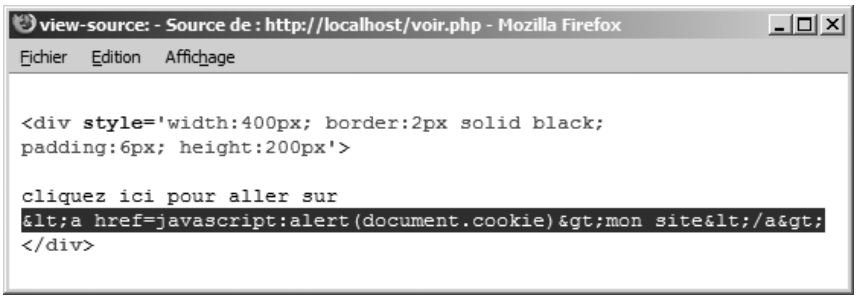

Figure 16.10 : *Les caractères dangereux ont bien été remplacés par leur équivalent HTML*

16.3. Sécuriser le serveur web

Le serveur web peut être assimilé au socle de votre application web. Un serveur web mal configuré, et c'est l'ensemble de votre outil qui peut en pâtir. Inversement, certains paramètres peuvent accroître largement la sécurité globale.

Les directives PHP

La plupart des directives de configuration de PHP peuvent être initialisées au sein du fichier *php.ini* ainsi qu'au niveau des fichiers de configuration d'Apache. Chaque site web du serveur peut ainsi disposer d'un environnement qui lui est propre.

Si certains paramètres par défaut ne vous conviennent pas, il est possible de demander un changement de configuration à votre administrateur système.

Un certain nombre de paramètres doivent être particulièrement contrôlés :

- `magic_quotes_gpc` : ce paramètre permet d'ajouter les fameux caractères d'échappement (`\'`, `\"`) pour les données transmises via les méthodes `GET`, `POST` ou via les cookies. Bien que cela soit a priori très pratique, cette fonctionnalité est risquée dans la mesure où elle ne conduit pas à un systématisme dans la protection des données. Les directives `magic_quotes_runtime` et `magic_quotes_sybase` peuvent également être forcées à `false` pour éviter les mêmes écueils.
- `register_globals` : cette directive est à `false` par défaut depuis la version 4 de PHP. Il est conseillé de la laisser ainsi afin d'éviter au maximum les problèmes d'initialisation et d'écrasement de variables rencontrés plus haut.
- `display_errors` : cette directive peut rester à `on` pendant tout le développement car elle vous permet d'afficher des informations précieuses durant la phase de correction des bugs (souvent qualifiée de phase de débogage) . Une fois le site mis en production, il est cependant conseillé de la passer à `off` afin de ne pas transmettre d'informations stratégiques à un éventuel pirate. Une erreur sur une inclusion de script donnera des informations sur l'organisation des répertoires et des scripts (il saura alors où frapper).

Figure 16.11 : *L'internaute sait désormais que test.php se situe dans C:\wamp\www*

Une erreur sur une connexion mysql permettra quant à elle d'obtenir des informations sur le serveur SGBD (caché jusqu'à présent) et d'orienter certaines attaques sur celui-ci. Encore une fois, plus votre ennemi dispose d'informations à exploiter, plus sa capacité de nuisance sera grande et rapide.

Figure 16.12 :
Erreur de connexion au SGBD

REMARQUE

De la discrétion

Dans la même lignée, veillez à ce que le fichier de log créé par le serveur web ne soit pas accessible en ligne (par exemple **www.toto.com/logs/access.log**). Un tel fichier contiendrait en effet les mêmes informations que celles affichées en mode `display_errors on`.

La directive `display_errors` est directement liée à `error_reporting` qui définit le niveau de précision des affichages d'erreur. Il est ainsi possible d'indiquer à PHP de n'afficher que les erreurs ou de les afficher accompagnées d'avertissements (notice).

Prenez le script suivant :

```
print($tmp);
$tmp = 3;
print("ici");
```

Placez-vous tout d'abord dans un environnement où `error_reporting` n'affiche pas les avertissements :

```
error_reporting = E_ALL & ~E_NOTICE
```

Figure 16.13 :
Rien à signaler a priori

Affichez maintenant les avertissements en plus des erreurs :

```
error_reporting = E_ ALL & E_NOTICE
```

Vous obtenez le résultat suivant :

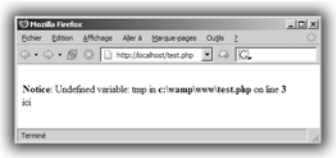

Figure 16.14 :
Vous êtes prévenu que la variable $tmp n'a pas été initialisée

En augmentant le degré d'affichage des erreurs, vous augmentez instantanément le taux de « capture » d'erreurs et la qualité globale du code.

La directive error_reporting

Cette directive peut prendre une multitude d'autres valeurs dont voici la liste :

- `E_ALL`, `E_ERROR` ;
- `E_WARNING` ;
- `E_PARSE`, `E_NOTICE` ;
- `E_CORE_ERROR` ;
- `E_CORE_WARNING` ;
- `E_COMPILE_ERROR` ;
- `E_COMPILE_WARNING` ;
- `E_USER_ERROR` ;
- `E_USER_WARNING` ;
- `E_USER_NOTICE`.

En jouant sur ces valeurs, vous obtiendrez un niveau de report en adéquation avec votre style de programmation.

Dans la même lignée, il est conseillé de logger tous les événements inhabituels qui ont lieu au sein de votre applicatif. La fonction `error_log()` vous permet d'ajouter vos propres commentaires au sein du fichier de log de PHP.

```
error_log_test("ajout de mes propres logs");
```

Vous retrouvez ces commentaires dans le fichier *C:\wamp\logs \php_error.log*.

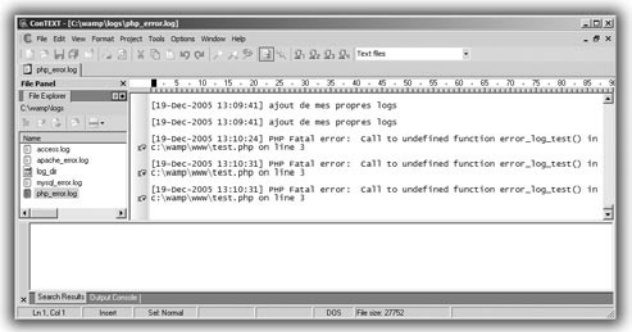

Figure 16.15 : *Vos logs apparaissent au milieu d'autres lignes d'erreur*

Cette fonction est particulièrement adaptée pour signaler des comportements anormaux. Voyez des exemples d'événements pouvant être loggés :

- un échec répété lors de l'authentification ;
- un internaute qui souhaite régler un panier qui n'existe pas ;
- une erreur dans une requête SQL.

Fonction error_log() et débogage

L'avantage de l'utilisation de la fonction `error_log()` par rapport à une simple fonction `print()` pendant une phase de débogage est de ne pas entrer en conflit avec les fonctions `setcookie()`, `header()` ou `session_start()`. En effet, l'utilisation de `print()` avant ces fonctions entraînerait l'affichage d'un message d'erreur. La fonction `error_log()` qui n'affiche rien à l'écran, mais qui écrit dans le fichier de log, ne rencontre pas ce problème.

En plus du message, il peut être intéressant d'ajouter l'adresse IP de l'internaute qui est à l'origine de l'événement.

```
error_log_test($_SERVER['REMOTE_ADDR']" : bizarre !");
```

S'il s'agit d'un applicatif où l'internaute est identifié, l'ajout de l'identifiant à la ligne de log est également pertinent.

Les directives Apache

La présence d'un répertoire contenant tous les fichiers inclus (`include()`) est une situation extrêmement courante. Nous avons