plusieurs lignes (tous les garçons) si vous aviez choisi une clause du type `WHERE sexe = 'masculin'`.

Écrivez le code permettant la mise à jour :

```
$sql = "UPDATE eleve SET nom = '".$_REQUEST['nom']."',".
       "prenom = '".$_REQUEST['prenom']."',".
       "adresse = '".$_REQUEST['adresse']."',".
       "ville = '".$_REQUEST['ville']."',".
       "cp = '".$_REQUEST['codepostal']."',".
       "pays = '".$_REQUEST['pays']."',".
       "sexe = '".$_REQUEST['sexe']."',".
       "naissance = '".$_REQUEST['naissance']."',".
       "taille = '".$_REQUEST['taille']."',".
       "email = '".$_REQUEST['email']."',".
       "telephone = '".$_REQUEST['telephone']."',".
       "lv = '".$_REQUEST['lv']."'".
       " WHERE ideleve = '".$_REQUEST['id']."'";
```

La mise à jour se fait directement avec les valeurs transmises. Ce n'est pas très judicieux dans la mesure où vous aviez mis en place toute une politique de vérification de champs pour la création.

Rajoutez donc tous ces tests :

```
if ($_REQUEST['enregistre'] == "oui")
{
  if (empty($_REQUEST['nom']) || empty($_REQUEST['prenom']) ||
      empty($_REQUEST['adresse']) ||
      ✂ empty($_REQUEST['ville']) ||
      empty($_REQUEST['codepostal']) ||
      ✂ empty($_REQUEST['pays']) ||
      empty($_REQUEST['naissance']) ||
      ✂ empty($_REQUEST['telephone']) ||
      empty($_REQUEST['lv']))
      die("ERREUR : tous les champs doivent être remplis.");

  if ($_REQUEST['sexe']!="masculin" &&
      $_REQUEST['sexe']!="feminin")
      die("ERREUR : choisissez votre sexe.");

  $reg_email = "/^[\w\.-]+@[\w\.-]+\.[a-z]{2,3}$/i";
  if (preg_match($reg_email,$_REQUEST['email']) == 0)
      die("ERREUR : adresse email non valide.");

  if ($_REQUEST['taille']<=100 || $_REQUEST['taille']>=220)
      die("ERREUR : la taille n'est pas valide.");
```

```
  $sql = "UPDATE eleve SET nom = '".$_REQUEST['nom']."',".
         "prenom = '".$_REQUEST['prenom']."',".
         "adresse = '".$_REQUEST['adresse']."',".
         "ville = '".$_REQUEST['ville']."',".
         "cp = '".$_REQUEST['codepostal']."',".
         "pays = '".$_REQUEST['pays']."',".
         "sexe = '".$_REQUEST['sexe']."',".
         "naissance = '".$_REQUEST['naissance']."',".
         "taille = '".$_REQUEST['taille']."',".
         "email = '".$_REQUEST['email']."',".
         "telephone = '".$_REQUEST['telephone']."',".
         "lv = '".$_REQUEST['lv']."'".
         " WHERE ideleve = '".$_REQUEST['id']."'";

  mysql_query ($sql);
}
```

En rassemblant toutes les parties, votre script *eleve_edite.php* prend la forme suivante :

```
<?php

echo "<html>";
echo "<head>";
echo "<title>Admin Ecole</title>";
echo "</head>";
echo "<body>";

$liendb = mysql_connect("localhost", "root", "");
mysql_select_db ("test");

if  ($_REQUEST['enregistre'] == "oui")
{
  if (empty($_REQUEST['nom']) || empty($_REQUEST['prenom']) ||
      empty($_REQUEST['adresse']) ||
      ✂ empty($_REQUEST['ville']) ||
      empty($_REQUEST['codepostal']) ||
      ✂ empty($_REQUEST['pays']) ||
      empty($_REQUEST['naissance']) ||
      ✂ empty($_REQUEST['telephone']) ||
      empty($_REQUEST['lv']))
      die("ERREUR : tous les champs doivent être
      ✂ remplis.");

  if ($_REQUEST['sexe']!="masculin" &&
      $_REQUEST['sexe']!="feminin")
      die("ERREUR : choisissez votre sexe.");

  $reg_email = "/^[\w\.-]+@[\w\.-]+\.[a-z]{2,3}$/i";
  if (preg_match($reg_email,$_REQUEST['email']) == 0)
      die("ERREUR : adresse email non valide.");
```

```
  if ($_REQUEST['taille']<=100 || $_REQUEST['taille']>=220)
      die("ERREUR : la taille n'est pas valide.");

  $sql = "UPDATE eleve SET nom = '".$_REQUEST['nom']."',".
         "prenom = '".$_REQUEST['prenom']."',".
         "adresse = '".$_REQUEST['adresse']."',".
         "ville = '".$_REQUEST['ville']."',".
         "cp = '".$_REQUEST['codepostal']."',".
         "pays = '".$_REQUEST['pays']."',".
         "sexe = '".$_REQUEST['sexe']."',".
         "naissance = '".$_REQUEST['naissance']."',".
         "taille = '".$_REQUEST['taille']."',".
         "email = '".$_REQUEST['email']."',".
         "telephone = '".$_REQUEST['telephone']."',".
         "lv = '".$_REQUEST['lv']."'".
         " WHERE ideleve = '".$_REQUEST['id']."'";

  mysql_query ($sql);
}

echo "<h1>admin - ecole</h1>";
echo "<p align=left> :: fiche d'élève
✂ [".$_REQUEST['id']."]</p>";

$sql = "SELECT * FROM eleve WHERE ideleve =
✂ '".$_REQUEST['id']."'";
$resultat = mysql_query ($sql);
$eleve = mysql_fetch_array ($resultat);

?>
<form action="eleve_edite.php" method="post">
<input type="hidden" name="enregistre" value="oui" />
<input type="hidden" name="id"
       value="<?php echo $_REQUEST['id']; ?>" />

<table>
<tr>
 <td>nom</td>
 <td><input type="text" name="nom"
            value="<?php echo $eleve['nom']; ?>" /></td>
</tr>
<tr>
 <td>prénom</td>
 <td><input type="text" name="prenom"
            value="<?php echo $eleve['prenom']; ?>" /></td>
</tr>
<tr>
 <td>adresse</td>
 <td><textarea name="adresse"><?php echo $eleve['adresse']; ?>
     </textarea></td>
```

```
</tr>
<tr>
 <td>ville</td>
 <td><input type="text" name="ville"
            value="<?php echo $eleve['ville']; ?>" /></td>
</tr>
<tr>
 <td>code postal</td>
 <td><input type="text" name="codepostal"
            value="<?php echo $eleve['cp']; ?>" /></td>
</tr>
<tr>
 <td>pays</td>
 <td><input type="text" name="pays"
            value="<?php echo $eleve['pays']; ?>" /></td>
</tr>
<tr>
 <td>sexe</td>
 <td>
  M <input type="radio" name="sexe" value="masculin"
  <?php if ($eleve['sexe'] == "masculin") echo "CHECKED"; ?>> -
  F <input type="radio" name="sexe" value="feminin"
  <?php if ($eleve['sexe'] == "feminin") echo "CHECKED"; ?>>
 </td>
</tr>
<tr>
 <td>date naissance</td>
 <td><input type="text" name="naissance"
            value="<?php echo $eleve['naissance']; ?>"
            ✂ /></td>
</tr>
<tr>
 <td>taille (cm)</td>
 <td><input type="text" name="taille"
            value="<?php echo $eleve['taille']; ?>" /></td>
</tr>
<tr>
 <td>email</td>
 <td><input type="text" name="email"
            value="<?php echo $eleve['email']; ?>" /></td>
</tr>
<tr>
 <td>téléphone</td>
 <td><input type="text" name="telephone"
            value="<?php echo $eleve['telephone']; ?>" /></td>
</tr>
<tr>
 <td>langue vivante</td>
 <td>
   <select name="lv">
     <option value="anglais"> anglais </option>
```

```
     <option value="espagnol" <?php if ($eleve['lv'] ==
     ✂ "espagnol")
echo "SELECTED"; ?>> espagnol </option>
     <option value="allemand" <?php if ($eleve['v'] ==
     ✂ "allemand")
echo "SELECTED"; ?>> allemand </option>
   </select>
 </td>
</tr>

</table>

<br/>

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

</form>

</body>
</html>

<?php mysql_close($liendb); ?>
```

Le fichier *eleve_edite.php* contient donc à la fois le formulaire et le script. Cette solution peut être intéressante pour diminuer le nombre de fichiers de votre applicatif. Le côté négatif est l'augmentation de la taille du script qui devient ainsi moins lisible.

Appliquez cette technique à la création de profil, en fusionnant les fichiers *ajout_eleve.html* et *eleve_enregistre.php* en un fichier *eleve_ajoute.php* :

```
<?php

if  ($_REQUEST['enregistre']=="oui" && $_REQUEST['id']>=1)
{
  $liendb = mysql_connect("localhost", "root", "");
  mysql_select_db ("test");

  if (empty($_REQUEST['nom']) || empty($_REQUEST['prenom']) ||
      empty($_REQUEST['adresse']) ||
      ✂ empty($_REQUEST['ville']) ||
      empty($_REQUEST['codepostal']) ||
      ✂ empty($_REQUEST['pays']) ||
      empty($_REQUEST['naissance']) ||
      ✂ empty($_REQUEST['telephone']) ||
      empty($_REQUEST['lv']))
      die("ERREUR : tous les champs doivent être
      ✂ remplis.");
```

```
  if ($_REQUEST['sexe']!="masculin" &&
      $_REQUEST['sexe']!="feminin")
      die("ERREUR : choisissez votre sexe.");

  $reg_email = "/^[\w\.-]+@[\w\.-]+\.[a-z]{2,3}$/i";
  if (preg_match($reg_email,$_REQUEST['email']) == 0)
      die("ERREUR : adresse email non valide.");

  if ($_REQUEST['taille']<=100 || $_REQUEST['taille']>=220)
      die("ERREUR : la taille n'est pas valide.");

  $sql = "INSERT INTO eleve (nom, prenom, adresse, ville,
  ✂ cp, pays, sexe, naissance, taille, email, telephone,
  ✂ lv) VALUES (".
          "'".$_REQUEST['nom']."',".
          "'".$_REQUEST['prenom']."',".
          "'".$_REQUEST['adresse']."',".
          "'".$_REQUEST['ville']."',".
          "'".$_REQUEST['codepostal']."',".
          "'".$_REQUEST['pays']."',".
          "'".$_REQUEST['sexe']."',".
          "'".$_REQUEST['naissance']."',".
          "'".$_REQUEST['taille']."',".
          "'".$_REQUEST['email']."',".
          "'".$_REQUEST['telephone']."',".
          "'".$_REQUEST['lv']."')";

  mysql_query ($sql);
  mysql_close($liendb);
  header("location:  http://localhost/admin.php");

}

echo "<html>";
echo "<head>";
echo "<title>Admin Ecole</title>";
echo "</head>";
echo "<body>";

echo "<h1>admin - ecole</h1>";
echo "<p align=left> :: ajouter un élève
✂ [".$_REQUEST['id']."]</p>";

?>

<form action="eleve_ajoute.php" method="post">
<input type="hidden" name="enregistre" value="oui" />

<table>
<tr>
 <td>nom</td>
```

```
 <td><input type="text" name="nom" /></td>
</tr>
<tr>
 <td>prénom</td>
 <td><input type="text" name="prenom" /></td>
</tr>
<tr>
 <td>adresse</td>
 <td><textarea name="adresse"></textarea></td>
</tr>
<tr>
 <td>ville</td>
 <td><input type="text" name="ville" /></td>
</tr>
<tr>
 <td>code postal</td>
 <td><input type="text" name="codepostal" /></td>
</tr>
<tr>
 <td>pays</td>
 <td><input type="text" name="pays" /></td>
</tr>
<tr>
 <td>sexe</td>
 <td>
  M <input type="radio" name="sexe" value="masculin" /> -
  F <input type="radio" name="sexe" value="feminin" />
 </td>
</tr>
<tr>
 <td>date naissance</td>
 <td><input type="text" name="naissance" /></td>
</tr>
<tr>
 <td>taille (cm)</td>
 <td><input type="text" name="taille" /></td>
</tr>
<tr>
 <td>email</td>
 <td><input type="text" name="email" /></td>
</tr>
<tr>
 <td>téléphone</td>
 <td><input type="text" name="telephone" /></td>
</tr>
<tr>
 <td>langue vivante</td>
 <td>
   <select name="lv">
     <option value="anglais"> anglais </option>
     <option value="espagnol"> espagnol </option>
```

```
      <option value="allemand"> allemand </option>
    </select>
  </td>
</tr>

</table>

<br/>

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

</form>

</body>
</html>
```

Après l'ajout à la base, vous basculez sur la liste des élèves. Pour cela, vous transmettez au navigateur la directive HTTP `location` grâce à la fonction PHP `header()`. L'instruction est la suivante :

```
header("location:  http://localhost/admin.php");
```

11.3. La suppression : DELETE

Le fichier *eleve_edite.php* permet déjà de mettre à jour les informations du profil `"élève"`. Vous allez maintenant faire en sorte de pouvoir supprimer un profil d'élève depuis ce fichier.

La commande SQL utilisée pour une suppression d'enregistrement est `DELETE`. La syntaxe est la suivante :

```
DELETE FROM nom_table WHERE clause
```

Si vous souhaitez effacer la ligne dont l'`ideleve` est `4`, écrivez :

```
DELETE  FROM  eleve  WHERE  ideleve  =  '4'
```

Pour pouvoir inclure le code de mise à jour, vous aviez ajouté un `input hidden` qui indiquait au script de réaliser l'action de mise à jour. Deux actions sont désormais possibles : la mise à jour et l'effacement. La technique n'est donc plus valide.

Pour contourner le problème, remplacez le bouton **enregistre** par un menu déroulant et un bouton. Le menu déroulant permettra de choisir l'action à réaliser sur la fiche ; il portera le nom `"action"`. Ainsi, plutôt

que tester la variable `$_REQUEST['enregistre']`, vous allez tester la variable `$_REQUEST['action']` et agir en fonction de sa valeur.

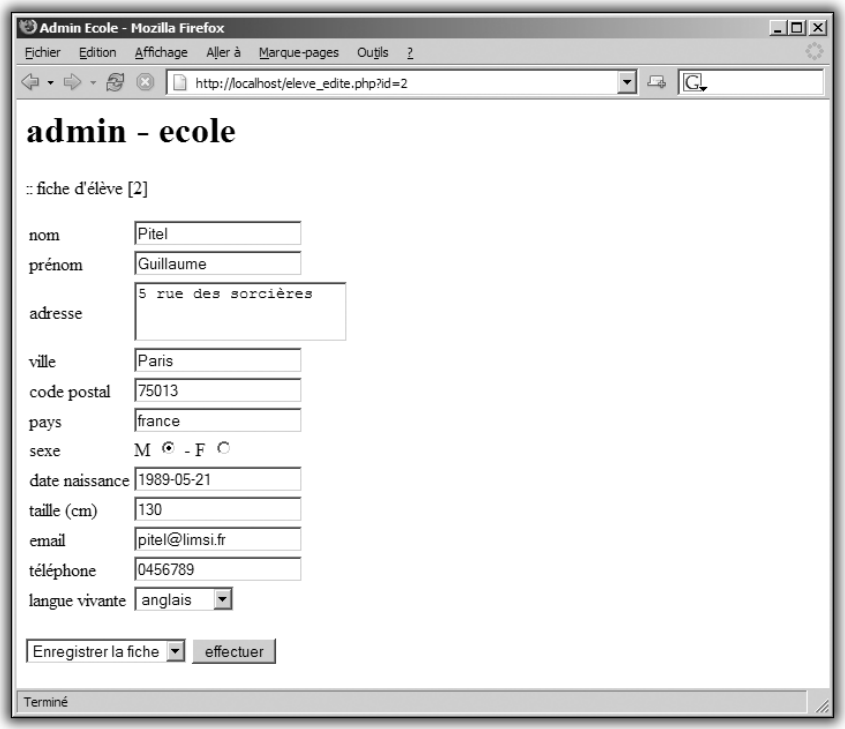

Figure 11.5 : *Mise à jour d'une fiche élève*

Le menu déroulant prend la forme suivante :

```
<select  name="action">
 <option  value="maj">  Enregistrer  la  fiche  </option>
 <option  value="suppr">  Supprimer  la  fiche  </option>
</select>

<input  type="submit"  value="effectuer">
```

Et le test devient :

```
if  ($_REQUEST['action']=="maj")
{
  if (empty($_REQUEST['nom']) || empty($_REQUEST['prenom']) ||
      empty($_REQUEST['adresse']) ||
      ✂ empty($_REQUEST['ville']) ||
      empty($_REQUEST['codepostal']) ||
      ✂ empty($_REQUEST['pays']) ||
```

```
        empty($_REQUEST['naissance']) ||
        ✂ empty($_REQUEST['telephone']) ||
        empty($_REQUEST['lv']))
        die("ERREUR : tous les champs doivent être
        ✂ remplis.");

    if ($_REQUEST['sexe']!="masculin" &&
        $_REQUEST['sexe']!="feminin")
        die("ERREUR : choisissez votre sexe.");

    $reg_email = "/^[\w\.-]+@[\w\.-]+\.[a-z]{2,3}$/i";
    if (preg_match($reg_email,$_REQUEST['email']) == 0)
        die("ERREUR : adresse email non valide.");

    if ($_REQUEST['taille']<=100 || $_REQUEST['taille']>=220)
        die("ERREUR : la taille n'est pas valide.");

    $sql = "UPDATE eleve SET nom = '".$_REQUEST['nom']."',".
           "prenom = '".$_REQUEST['prenom']."',".
           "adresse = '".$_REQUEST['adresse']."',".
           "ville = '".$_REQUEST['ville']."',".
           "cp = '".$_REQUEST['codepostal']."',".
           "pays = '".$_REQUEST['pays']."',".
           "sexe = '".$_REQUEST['sexe']."',".
           "naissance = '".$_REQUEST['naissance']."',".
           "taille = '".$_REQUEST['taille']."',".
           "email = '".$_REQUEST['email']."',".
           "telephone = '".$_REQUEST['telephone']."',".
           "lv = '".$_REQUEST['lv']."'".
           " WHERE ideleve = '".$_REQUEST['id']."'";

    mysql_query ($sql);
}
elseif  ($_REQUEST['action']=="suppr" &&
✂ $_REQUEST['id']>=1)
{
    $sql = "DELETE FROM eleve WHERE
    ✂ ideleve='".$_REQUEST['id']."'";
    mysql_query ($sql);
}
```

Si vous supprimez le profil, vous ne devez surtout pas continuer à exécuter le script, sous peine de générer une erreur. Il est en effet impossible d'afficher un profil d'élève qui n'est plus présent dans la table.

Vous allez donc réutiliser l'instruction suivante :

```
header("location:  http://localhost/admin.php");
```

Avant d'utiliser cette ligne, il est cependant nécessaire de modifier légèrement le script. En effet, vous avez vu dans un chapitre précédent que la fonction `header()` ne pouvait être appelée que si aucun affichage n'avait encore eu lieu. Or, votre script présente les cinq lignes suivantes :

```
echo  "<html>";
echo  "<head>";
echo  "<title>Admin  Ecole</title>";
echo  "</head>";
echo  "<body>";
```

Vous êtes donc contraint de les déplacer plus bas dans le script, au moins en dessous de la fonction `header()`.

Le script final devient donc :

```
<?php

$liendb = mysql_connect("localhost", "root", "");
mysql_select_db ("test");

if  ($_REQUEST['action']=="maj")
{
  if (empty($_REQUEST['nom']) || empty($_REQUEST['prenom']) ||
      empty($_REQUEST['adresse']) ||
      ✂ empty($_REQUEST['ville']) ||
      empty($_REQUEST['codepostal']) ||
      ✂ empty($_REQUEST['pays']) ||
      empty($_REQUEST['naissance']) ||
      ✂ empty($_REQUEST['telephone']) ||
      empty($_REQUEST['lv']))
      die("ERREUR : tous les champs doivent être
      ✂ remplis.");

  if ($_REQUEST['sexe']!="masculin" &&
      $_REQUEST['sexe']!="feminin")
      die("ERREUR : choisissez votre sexe.");

  $reg_email = "/^[\w\.-]+@[\w\.-]+\.[a-z]{2,3}$/i";
  if (preg_match($reg_email,$_REQUEST['email']) == 0)
      die("ERREUR : adresse email non valide.");

  if ($_REQUEST['taille']<=100 || $_REQUEST['taille']>=220)
      die("ERREUR : la taille n'est pas valide.");

  $sql = "UPDATE eleve SET nom = '".$_REQUEST['nom']."',".
         "prenom = '".$_REQUEST['prenom']."',".
         "adresse = '".$_REQUEST['adresse']."',".
         "ville = '".$_REQUEST['ville']."',".
```

```
            "cp = '".$_REQUEST['codepostal']."',".
            "pays = '".$_REQUEST['pays']."',".
            "sexe = '".$_REQUEST['sexe']."',".
            "naissance = '".$_REQUEST['naissance']."',".
            "taille = '".$_REQUEST['taille']."',".
            "email = '".$_REQUEST['email']."',".
            "telephone = '".$_REQUEST['telephone']."',".
            "lv = '".$_REQUEST['lv']."'".
            " WHERE ideleve = '".$_REQUEST['id']."'";

  mysql_query ($sql);
}
elseif  ($_REQUEST['action']=="suppr" &&
✂ $_REQUEST['id']>=1)
{
  $sql = "DELETE FROM eleve WHERE
  ✂ ideleve='".$_REQUEST['id']."'";
  mysql_query ($sql);
  header("Location: http://localhost/admin.php");
}

echo "<html>";
echo "<head>";
echo "<title>Admin Ecole</title>";
echo "</head>";
echo "<body>";

echo "<h1>admin - ecole</h1>";
echo "<p align=left> :: fiche d'élève
✂ [".$_REQUEST['id']."]</p>";

$sql = "SELECT * FROM eleve WHERE ideleve =
✂ '".$_REQUEST['id']."'";
$resultat = mysql_query ($sql);
$eleve = mysql_fetch_array ($resultat);

?>

<form action="eleve_edite.php" method="post">
<input type="hidden" name="enregistre" value="oui" />
<input type="hidden" name="id" value="<?php echo
✂ $_REQUEST['id']; ?>" />

<table>
<tr>
 <td>nom</td>
 <td><input type="text" name="nom" value="<?php echo
 ✂ $eleve['nom']; ?>" /></td>
</tr>
<tr>
 <td>prénom</td>
```

```
 <td><input type="text" name="prenom" value="<?php echo
 ✂ $eleve['prenom']; ?>" /></td>
</tr>
<tr>
 <td>adresse</td>
 <td><textarea name="adresse"><?php echo $eleve['adresse'];
 ✂ ?></textarea></td>
</tr>
<tr>
 <td>ville</td>
 <td><input type="text" name="ville" value="<?php echo
 ✂ $eleve['ville']; ?>" /></td>
</tr>
<tr>
 <td>code postal</td>
 <td><input type="text" name="codepostal" value="<?php echo
 ✂ $eleve['cp']; ?>" /></td>
</tr>
<tr>
 <td>pays</td>
 <td><input type="text" name="pays" value="<?php echo
 ✂ $eleve['pays']; ?>" /></td>
</tr>
<tr>
 <td>sexe</td>
 <td>
  M <input type="radio" name="sexe" value="masculin"
  <?php if ($eleve['sexe'] == "masculin") echo "CHECKED"; ?>> -
  F <input type="radio" name="sexe" value="feminin"
  <?php if ($eleve['sexe'] == "feminin") echo "CHECKED"; ?>>
 </td>
</tr>
<tr>
 <td>date naissance</td>
 <td><input type="text" name="naissance" value="<?php echo
 ✂ $eleve['naissance']; ?>" /></td>
</tr>
<tr>
 <td>taille (cm)</td>
 <td><input type="text" name="taille" value="<?php echo
 ✂ $eleve['taille']; ?>" /></td>
</tr>
<tr>
 <td>email</td>
 <td><input type="text" name="email" value="<?php echo
 ✂ $eleve['email']; ?>" /></td>
</tr>
<tr>
 <td>téléphone</td>
 <td><input type="text" name="telephone" value="<?php echo
 ✂ $eleve['telephone']; ?>" /></td>
```

```
</tr>
<tr>
 <td>langue vivante</td>
 <td>
   <select name="lv">
     <option value="anglais"> anglais </option>
     <option value="espagnol" <?php if ($eleve['lv'] ==
     ✂ "espagnol")
echo "SELECTED"; ?>> espagnol </option>
     <option value="allemand" <?php if ($eleve['v'] ==
     ✂ "allemand")
echo "SELECTED"; ?>> allemand </option>
   </select>
 </td>
</tr>

</table>

<br/>

<select  name="action">
 <option  value="maj">  Enregistrer  la  fiche  </option>
 <option  value="suppr">  Supprimer  la  fiche  </option>
</select>

<input  type="submit"  value="effectuer">

</form>

</body>
</html>

<?php mysql_close($liendb); ?>
```

Fonction mysql_close()

Il est préférable de toujours fermer la connexion à la base. Vous quittez le script prématurément si l'action correspond à une suppression. Vous en profitez pour fermer la connexion juste avant de réaliser la redirection (`header()`).

11.4. La factorisation du code

Si vous laissez les scripts *eleve_edite.php* et *eleve_ajoute.php* comme ils sont, vous pouvez avoir des craintes quant à la validité de vos données. Il est en effet nécessaire de s'identifier pour avoir accès à la page d'accueil contenant la liste des élèves, mais aucune vérification n'est

faite sur l'ajout ou la modification (suppression). Toute personne tapant directement `eleve_edite.php?id=6` peut modifier sans problème le contenu de la page.

Il est donc nécessaire de forcer l'identification en haut de chaque script :

```
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("verification : ERREUR");
    exit(0);
}
```

Cette solution est cependant loin d'être optimale. Écrire ce code à plusieurs endroits entraîne une perte de temps et un risque accru de laisser des erreurs. De plus, si vous voulez modifier le code d'accès en `essai2/essai2`, vous serez obligé d'aller modifier tous les fichiers, ce qui compliquerait la mise à jour.

La fonction include

Pour éviter cet écueil, PHP vous propose la fonction `include()`. Elle permet, depuis un script *A*, d'appeler un fichier *B*. Vous pouvez donc créer un fichier *identification.inc.php*, et l'appeler au début de chaque script devant être protégé :

```
<?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("verification : ERREUR");
    exit(0);
}

?>
```

Cette technique permet de modifier la gestion de l'identification en un instant, qu'il y ait 5 ou 5000 scripts dans votre applicatif.

Dans la même veine, vous pouvez aussi remarquer l'utilisation dans plusieurs scripts d'une connexion à une base de données. En ayant à l'esprit la fonction `include()` et en étant suffisamment paresseux, il vous viendra tout de suite à l'idée de créer le fichier *variables.inc.php* :

Listing 11-4 : variables.inc.php

```
<?php

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

?>
```

Ce fichier est aussi inclus en haut de tous les scripts et vous permet de déplacer votre applicatif d'un hébergeur vers un autre sans le moindre souci. Tous les paramètres susceptibles de changer sont isolés dans un fichier unique. Stocker de telles variables dans un fichier extérieur est donc une bonne habitude à prendre.

Ne nous arrêtons pas là dans cette quête « factorisatrice » et regardons du côté du visuel… Chaque page dispose du même en-tête et du même pied de page. Comme il y a fort à parier qu'à un moment ou à un autre vous vouliez améliorer visuellement l'ensemble et placer sur chaque page une référence à un copyright, il pourrait donc être tout à fait judicieux de créer les fichiers *haut.inc.php* et *bas.inc.php* :

Listing 11-5 : haut.inc.php

```
<html>
<head>
 <title>Admin Ecole</title>
</head>
<body>

<h1>Admin - Ecole</h1>
```

Listing 11-6 : bas.inc.php

```
</body>
</html>
```

Chaque script du back-office sera donc construit de la façon suivante :

- appel à *var.inc.php* ;

- appel à *identification.inc.php* ;
- appel à *haut.inc.php* ;
- le contenu proprement dit du script ;
- *bas.inc.php*.

Modifiez en ce sens les fichiers *admin.php, eleve_ajoute.php*, et *eleve_edite.php* :

Listing 11-7 : admin.php

```
<?php

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

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

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

echo "<p align=left> :: accueil</p>";

echo "<table width=90% align=center border=1>";
echo "<tr><td>id</td><td>nom</td><td>prenom</td>
<td>naissance</td><td> </td></tr>";

while ($eleve = mysql_fetch_array ($resultat))
{
  $id = $eleve['ideleve'];
  $nom = $eleve['nom'];
  $prenom = $eleve['prenom'];
  $date = $eleve['naissance'];
  echo "<tr>";
  echo "<td>$id</td>";
  echo "<td>$nom</td>";
  echo "<td>$prenom</td>";
  echo "<td>$date</td>";
  echo "<td>";
  echo "<a href='eleve_edite.php?id=$id'>voir</a>";
  echo "</td>";
  echo "</tr>";
}

echo "</table>";

mysql_close($liendb);
```

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

?>
```

Listing 11-8 : eleve_ajoute.inc.php

```
<?php

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

if  ($_REQUEST['enregistre'] == "oui")
{
  $liendb = mysql_connect ($bddserver, $bddlogin,
  ✂ $bddpassword);
  mysql_select_db ($bdd);

  if (empty($_REQUEST['nom']) || empty($_REQUEST['prenom']) ||
      empty($_REQUEST['adresse']) ||
      ✂ empty($_REQUEST['ville']) ||
      empty($_REQUEST['codepostal']) ||
      ✂ empty($_REQUEST['pays']) ||
      empty($_REQUEST['naissance']) ||
      ✂ empty($_REQUEST['telephone']) ||
      empty($_REQUEST['lv']))
      die("ERREUR : tous les champs doivent être remplis.");

  if ($_REQUEST['sexe']!="masculin" &&
      $_REQUEST['sexe']!="feminin")
      die("ERREUR : choisissez votre sexe.");

  $reg_email = "/^[\w\.-]+@[\w\.-]+\.[a-z]{2,3}$/i";
  if (preg_match($reg_email,$_REQUEST['email']) == 0)
      die("ERREUR : adresse email non valide.");

  if ($_REQUEST['taille']<=100 || $_REQUEST['taille']>=220)
      die("ERREUR : la taille n'est pas valide.");

  $sql = "INSERT INTO eleve (nom, prenom, adresse, ville,
  ✂ cp, pays, sexe, naissance, taille, email, telephone,
  ✂ lv) VALUES (".
          "'".$_REQUEST['nom']."',".
          "'".$_REQUEST['prenom']."',".
          "'".$_REQUEST['adresse']."',".
          "'".$_REQUEST['ville']."',".
          "'".$_REQUEST['codepostal']."',".
          "'".$_REQUEST['pays']."',".
          "'".$_REQUEST['sexe']."',".
          "'".$_REQUEST['naissance']."',".
          "'".$_REQUEST['taille']."',".
```

```
            "'".$_REQUEST['email']."',".
            "'".$_REQUEST['telephone']."',".
            "'".$_REQUEST['lv']."')";

  mysql_query ($sql);
  mysql_close($liendb);
  header("location:  http://localhost/admin.php");

}

include("haut.inc.php");

echo "<p align=left> :: ajouter un élève
✂ [".$_REQUEST['id']."]</p>";

?>

<form action="eleve_ajoute.php" method="post">
<input type="hidden" name="enregistre" value="oui" />

<table>
<tr>
 <td>nom</td>
 <td><input type="text" name="nom" /></td>
</tr>
<tr>
 <td>prénom</td>
 <td><input type="text" name="prenom" /></td>
</tr>
<tr>
 <td>adresse</td>
 <td><textarea name="adresse"></textarea></td>
</tr>
<tr>
 <td>ville</td>
 <td><input type="text" name="ville" /></td>
</tr>
<tr>
 <td>code postal</td>
 <td><input type="text" name="codepostal" /></td>
</tr>
<tr>
 <td>pays</td>
 <td><input type="text" name="pays" /></td>
</tr>
<tr>
 <td>sexe</td>
 <td>
  M <input type="radio" name="sexe" value="masculin" /> -
  F <input type="radio" name="sexe" value="feminin" />
 </td>
```

```
</tr>
<tr>
 <td>date naissance</td>
 <td><input type="text" name="naissance" /></td>
</tr>
<tr>
 <td>taille (cm)</td>
 <td><input type="text" name="taille" /></td>
</tr>
<tr>
 <td>email</td>
 <td><input type="text" name="email" /></td>
</tr>
<tr>
 <td>téléphone</td>
 <td><input type="text" name="telephone" /></td>
</tr>
<tr>
 <td>langue vivante</td>
 <td>
   <select name="lv">
     <option value="anglais"> anglais </option>
     <option value="espagnol"> espagnol </option>
     <option value="allemand"> allemand </option>
   </select>
 </td>
</tr>

</table>

<br/>

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

</form>

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

Listing 11-9 : eleve_edite.php

```
<?php

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

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

if  ($_REQUEST['action']=="maj")
{
  if (empty($_REQUEST['nom']) || empty($_REQUEST['prenom']) ||
```

```
        empty($_REQUEST['adresse']) ||
        ✂ empty($_REQUEST['ville']) ||
        empty($_REQUEST['codepostal']) ||
        ✂ empty($_REQUEST['pays']) ||
        empty($_REQUEST['naissance']) ||
        ✂ empty($_REQUEST['telephone']) ||
        empty($_REQUEST['lv']))
        die("ERREUR : tous les champs doivent être
        ✂ remplis.");

    if ($_REQUEST['sexe']!="masculin" &&
        $_REQUEST['sexe']!="feminin")
        die("ERREUR : choisissez votre sexe.");

    $reg_email = "/^[\w\.-]+@[\w\.-]+\.[a-z]{2,3}$/i";
    if (preg_match($reg_email,$_REQUEST['email']) == 0)
        die("ERREUR : adresse email non valide.");

    if ($_REQUEST['taille']<=100 || $_REQUEST['taille']>=220)
        die("ERREUR : la taille n'est pas valide.");

    $sql = "UPDATE eleve SET nom = '".$_REQUEST['nom']."',".
            "prenom = '".$_REQUEST['prenom']."',".
            "adresse = '".$_REQUEST['adresse']."',".
            "ville = '".$_REQUEST['ville']."',".
            "cp = '".$_REQUEST['codepostal']."',".
            "pays = '".$_REQUEST['pays']."',".
            "sexe = '".$_REQUEST['sexe']."',".
            "naissance = '".$_REQUEST['naissance']."',".
            "taille = '".$_REQUEST['taille']."',".
            "email = '".$_REQUEST['email']."',".
            "telephone = '".$_REQUEST['telephone']."',".
            "lv = '".$_REQUEST['lv']."'".
            " WHERE ideleve = '".$_REQUEST['id']."'";

    mysql_query ($sql);
}
elseif  ($_REQUEST['action']=="suppr" &&
✂ $_REQUEST['id']>=1)
{
    $sql = "DELETE FROM eleve WHERE
    ✂ ideleve='".$_REQUEST['id']."'";
    mysql_query ($sql);
    header("Location: http://localhost/admin.php");
}

include("haut.inc.php");

echo "<p align=left> :: fiche d'élève
✂ [".$_REQUEST['id']."]</p>";
```

```
$sql = "SELECT * FROM eleve WHERE ideleve =
✂ '".$_REQUEST['id']."'";
$resultat = mysql_query ($sql);
$eleve = mysql_fetch_array ($resultat);

?>

<form action="eleve_edite.php" method="post">
<input type="hidden" name="enregistre" value="oui" />
<input type="hidden" name="id" value="<?php echo
✂ $_REQUEST['id']; ?>" />

<table>
<tr>
 <td>nom</td>
 <td><input type="text" name="nom" value="<?php echo
 ✂ $eleve['nom']; ?>" /></td>
</tr>
<tr>
 <td>prénom</td>
 <td><input type="text" name="prenom" value="<?php echo
 ✂ $eleve['prenom']; ?>" /></td>
</tr>
<tr>
 <td>adresse</td>
 <td><textarea name="adresse"><?php echo $eleve['adresse'];
 ✂ ?></textarea></td>
</tr>
<tr>
 <td>ville</td>
 <td><input type="text" name="ville" value="<?php echo
 ✂ $eleve['ville']; ?>" /></td>
</tr>
<tr>
 <td>code postal</td>
 <td><input type="text" name="codepostal" value="<?php echo
 ✂ $eleve['cp']; ?>" /></td>
</tr>
<tr>
 <td>pays</td>
 <td><input type="text" name="pays" value="<?php echo
 ✂ $eleve['pays']; ?>" /></td>
</tr>
<tr>
 <td>sexe</td>
 <td>
  M <input type="radio" name="sexe" value="masculin"
  <?php if ($eleve['sexe'] == "masculin") echo "CHECKED"; ?>> -
  F <input type="radio" name="sexe" value="feminin"
  <?php if ($eleve['sexe'] == "feminin") echo "CHECKED"; ?>>
 </td>
```

```
</tr>
<tr>
 <td>date naissance</td>
 <td><input type="text" name="naissance" value="<?php echo
 $eleve['naissance']; ?>" /></td>
</tr>
<tr>
 <td>taille (cm)</td>
 <td><input type="text" name="taille" value="<?php echo
 $eleve['taille']; ?>" /></td>
</tr>
<tr>
 <td>email</td>
 <td><input type="text" name="email" value="<?php echo
 $eleve['email']; ?>" /></td>
</tr>
<tr>
 <td>téléphone</td>
 <td><input type="text" name="telephone" value="<?php echo
 $eleve['telephone']; ?>" /></td>
</tr>
<tr>
 <td>langue vivante</td>
 <td>
   <select name="lv">
     <option value="anglais"> anglais </option>
     <option value="espagnol" <?php if ($eleve['lv'] ==
     "espagnol")
echo "SELECTED"; ?>> espagnol </option>
     <option value="allemand" <?php if ($eleve['v'] ==
     "allemand")
echo "SELECTED"; ?>> allemand </option>
   </select>
 </td>
</tr>

</table>

<br/>

<select  name="action">
 <option  value="maj">  Enregistrer  la  fiche  </option>
 <option  value="suppr">  Supprimer  la  fiche  </option>
</select>

<input  type="submit"  value="effectuer">

</form>

</body>
</html>
```

```
<?php

mysql_close ($liendb);
include ("bas.inc.php");

?>
```

Modifiez, tout de suite, les fichiers *haut.inc.php* et *bas.inc.php* pour vous rendre compte de l'intérêt de cette action :

```
<html>
<head>
 <title>Admin Ecole</title>
</head>
<body>

<h1>Admin - Ecole</h1>

<center>

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

<a href="admin.php">accueil</a> -
<a href="eleve_ajoute.php">ajouter un  élève</a>

<hr>

© libre

</center>

</body>
</html>
```

Instantanément, tous vos scripts disposent de la nouvelle mise en page. (voir Figure 9.6)

En conclusion, il est très important de toujours avoir à l'esprit la factorisation de votre code :

- Cela le rend plus lisible, plus facile à maintenir, à mettre à jour, à faire évoluer.
- Cela vous fait gagner du temps.
- Cela réduit la taille du code et ainsi la présence de bugs.

Cette technique est aussi de plus en plus utilisée en front-office, et porte le nom de *template*. Ces templates permettent de modifier très rapidement l'intégralité du site.

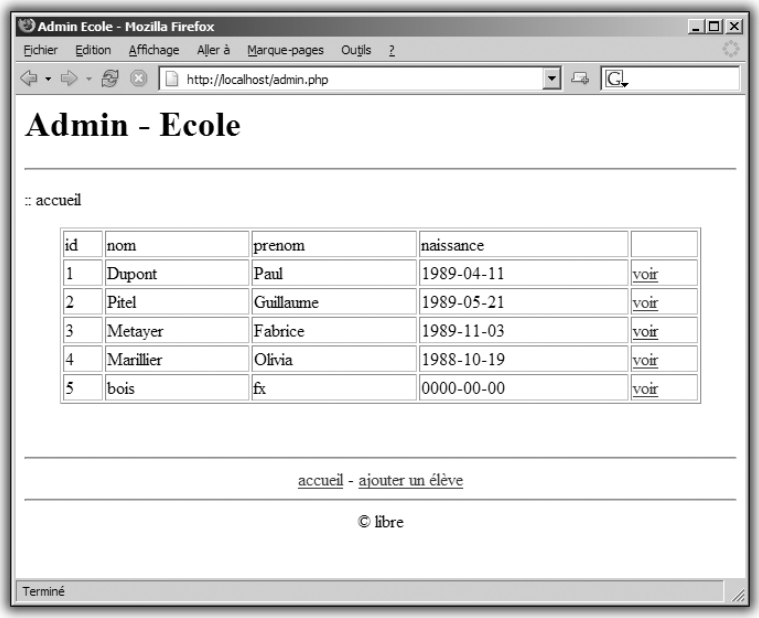

Figure 11.6 : *Nouvelle mise en page*

L'amélioration visuelle : les CSS

Avant d'enrichir votre applicatif d'autres fonctionnalités, nous allons nous arrêter quelques instants sur son aspect visuel. Bien que cela semble souvent secondaire, bien présenter une page, en jouant sur les couleurs, les polices, etc., peut faciliter considérablement la lisibilité et la compréhension de votre page.

Le HTML donne la possibilité de jouer sur les polices, les couleurs, les fonds… La première solution qui vient donc à l'esprit est de modifier directement le code afin d'inclure des indications visuelles. Pour avoir un tableau avec un fond bleu, vous pouvez écrire :

```
<table  bgcolor="blue">
```

Cependant, c'est encore une fois une mauvaise approche car elle n'a pas l'avantage de factoriser le travail. En effet, si vous souhaitez modifier la couleur du titre des rubriques, il faut revenir sur tous les scripts et les modifier un à un.

Pour faciliter la gestion de l'aspect visuel de votre applicatif, vous allez tirer partie des CSS (*Cascading Style Sheets*). Ce sont des commandes qui permettent, là encore, de modifier l'aspect visuel des pages, et qui ont un double avantage :

- Vous pouvez apporter des modifications plus fines (taille, espacement de lettres, etc.) et plus nombreuses (boutons, menus, etc.).
- Vous pouvez gérer, en un seul endroit, les attributs visuels de toutes les pages.

Les instructions CSS peuvent être placées :

- directement dans les balises ;

```
<p  style="background:  #CCCCCC;  color:  black;">
```

- dans l'en-tête de la page.

```
<html>
<head>
<style type="text/css">
<!--
P {background: #CCCCCC; color: black;}
-->
</style>
</head>
</html>
```

Grâce à votre factorisation PHP et à l'existence de votre fichier *haut.inc.php*, vous allez être en mesure de modifier l'intégralité des scripts en n'intervenant que sur un fichier :

```
<html>
<head>
<title>Admin Ecole</title>
<style type="text/css">
<!--
BODY {background: #D0D8D5;}
-->
</style>
</head>
<body>
<h1>Admin - Ecole</h1>
<center>
<hr>
```

Avec cette modification, toutes vos pages ont désormais un fond vert très clair.

Figure 11.7 : *Modification du fond de la page*

Sur le même modèle, il est possible de modifier l'intégralité des éléments HTML de vos pages :

```
<html>
<head>
<title>Admin Ecole</title>
<style type="text/css">
<!--

BODY
{background:#D0D8D5; color:#01291A; font-family:Verdana;
✂ font-size:10px;}

H1
{background:#0B3E2B; color:white; font-family:Arial;
✂ font-size:20px; font-weight:bold;}

P
{background:#B6CCC4; color:#0B3E2B; font-family:Arial;
✂ font-size:14px; font-weight:bold;}

TD
```

```
{background:white; color:#0B3E2B; font-family:Arial;
✂ font-size:14px; font-weight:normal;}

INPUT
{background:#EBF0EE; color:#01291A; font-family:Verdana;
✂ font-size:10px;}

TEXTAREA
{background:#EBF0EE; color:#01291A; font-family:Verdana;
✂ font-size:10px;}

SELECT
{background:#EBF0EE; color:#01291A; font-family:Verdana;
✂ font-size:10px;}

HR {color:#0B3E2B;}

A {color:#AF6114; font-family:verdana; font-size:10px;}

-->
</style>

</head>
<body>
<h1>Admin - Ecole</h1>
<center>
<hr/>
```

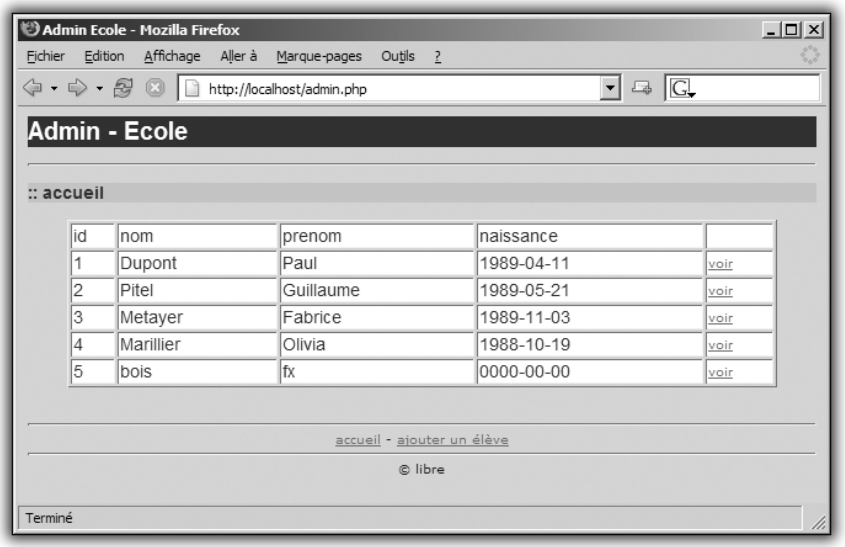

Figure 11.8 : *Encore une nouvelle présentation*

L'aspect devient maintenant plus agréable, mais reste encore légèrement monotone. Toutes les cases de vos tableaux ont un fond blanc. Il serait donc préférable, dans le fichier *admin.php*, de démarquer la première ligne, qui contient les intitulés des colonnes. Pour cela, il est nécessaire d'intervenir dans le fichier et d'ajouter une propriété à tous les `<td>` en question. La propriété est `class='nom_de_la_class'`. Si vous choisissez de l'appeler `intitule`, vous modifiez *admin.php* de cette manière :

```
echo "<table width=90% align=center border=1>";
echo "<tr>";
echo "<td class='intitule'>id</td>";
echo "<td class='intitule'>nom</td>";
echo "<td class='intitule'>prenom</td>";
echo "<td class='intitule'>naissance</td>";
echo "<td class='intitule'> </td>";
echo "</tr>";
```

La présence de cette classe va vous permettre d'affiner votre CSS et de faire en sorte que les cellules de tableaux ayant la classe `intitule` aient un aspect visuel différent. Précisez-le avec la notation `TD.intitule` :

```
<html>
<head>
<title>Admin Ecole</title>
<style type="text/css">
<!--

BODY
{background:#D0D8D5; color:#01291A; font-family:Verdana;
✂ font-size:10px;}

H1
{background:#0B3E2B; color:white; font-family:Arial;
✂ font-size:20px; font-weight:bold;}

P
{background:#B6CCC4; color:#0B3E2B; font-family:Arial;
✂ font-size:14px; font-weight:bold;}

TD
{background:white; color:#0B3E2B; font-family:Arial;
✂ font-size:14px; font-weight:normal;}

TD.intitule
{background:#777; color:white; font-family:Arial;
✂ font-size:14px; font-weight:bold;}

INPUT
```

```
{background:#EBF0EE; color:#01291A; font-family:Verdana;
✂ font-size:10px;}

TEXTAREA
{background:#EBF0EE; color:#01291A; font-family:Verdana;
✂ font-size:10px;}

SELECT
{background:#EBF0EE; color:#01291A; font-family:Verdana;
✂ font-size:10px;}

HR {color:#0B3E2B;}

A {color:#AF6114; font-family:verdana; font-size:10px;}

-->
</style>

</head>
<body>
<h1>Admin - Ecole</h1>
<center>
<hr/>
```

Vous auriez aussi pu écrire :

```
.intitule
{background:#0B3E2B; color:white; font-family:Arial;
✂ font-size:14px; font-weight:bold;}
```

Dans ce cas, toutes les balises ayant la classe `intitule` auraient partagé le même style, par exemple `<p class='intitule'>`.

Une des dernières propriétés intéressantes des CSS est de permettre de modifier le style d'un élément de la page en fonction d'un événement. Ainsi, il est possible de changer l'aspect visuel des liens lorsque vous passez le pointeur de la souris au-dessus :

```
A:hover
{background:#AF6114; color:white; font-family:Verdana;
✂ font-size:10px;}
```

Avant de finir cette partie, il convient de revenir sur quelques inconvénients des CSS :

- Certains navigateurs anciens ne gèrent pas du tout les CSS.
- Selon les navigateurs, certains styles sont absents ou mal traités.

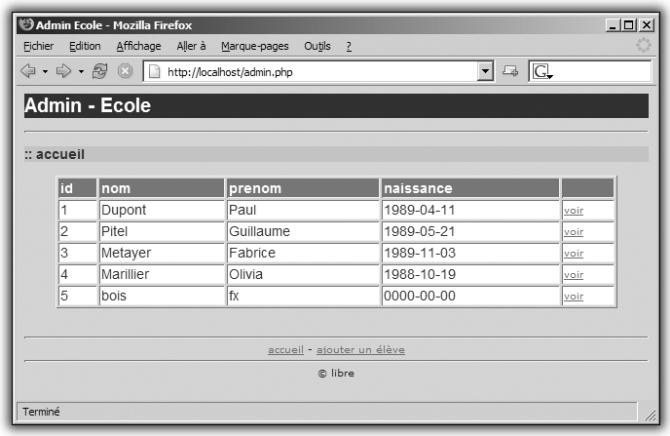

Figure 11.9 : *Nouvelle présentation avec changement de l'aspect visuel des liens*

11.5. Recherche et tri au sein d'une base

La fonction de recherche et de tri d'éléments de la base est souvent utile au sein d'un back-office. Vous allez donc modifier la page d'accueil en lui ajoutant, tout d'abord, un petit formulaire permettant de taper le nom ou le prénom d'un élève, puis un menu qui offrira la possibilité de lister les élèves suivant un critère donné.

Définir la fonction de recherche

La première idée qui peut venir à l'esprit est de créer un nouveau script nommé par exemple *eleve_recherche.php*, qui listerait les élèves correspondant à un mot-clé. Cependant, en y regardant de plus près, vous pouvez vous apercevoir qu'il s'agit en fait du fichier *admin.php*, auquel il suffit d'ajouter une clause à la fin de la requête `SELECT`.

L'idée est donc de générer une clause dès qu'un mot-clé est passé en paramètre. Rien de bien sorcier en perspective ! Le moteur de recherche doit par contre être suffisamment permissif et vous renvoyer `Paul Dupont` même si vous ne tapez que `Dup`. L'utilisation de l'opérateur d'égalité (=) dans la clause n'est donc pas valide. Ce qu'il faut employer, dans ce cas, c'est la fonction MySQL `INSTR()`. Cette

fonction prend deux arguments : la donnée dans laquelle on cherche un motif et le motif en question. Le premier argument est généralement le nom d'une colonne.

La requête est donc construite ainsi :

```
SELECT * FROM $table_eleve WHERE INSTR(nom,$motclef) OR
✂ INSTR(prenom,$motclef)
```

Fonction MySQL

Il existe une multitude de fonctions qui peuvent être incluses directement dans des requêtes SQL. Vous en trouverez un certain nombre dans les annexes.

Listing 11-10 : Moteur de recherche intégré

```
<?php

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

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

?>

<p align='left'>:: accueil</p>

<form action="admin.php" method="post">
  <input type="text" name="motclef"
         value="<?php echo $_REQUEST['motclef'];?>" />
  <input type="submit" value="rechercher" />
</form>


<table width='90%' border='1'>

<tr>
 <td class='intitule'>id</td>
 <td class='intitule'>nom</td>
 <td class='intitule'>prenom</td>
 <td class='intitule'>naissance</td>
 <td class='intitule'> </td>
</tr>

<?php
```

```
$clause = '';
if  (isset($_REQUEST['motclef'])) {
  $clause .= " WHERE
  ✂ INSTR(nom,'".$_REQUEST['motclef']."')";
  $clause .= " OR
  ✂ INSTR(prenom,'".$_REQUEST['motclef']."')";
}

$sql = "SELECT * FROM eleve ".$clause;
$resultat = mysql_query ($sql);

while ($eleve = mysql_fetch_array ($resultat))
{
  $id = $eleve['ideleve'];
  $nom = $eleve['nom'];
  $prenom = $eleve['prenom'];
  $date = $eleve['naissance'];
  echo "<tr>";
  echo "<td>$id</td>";
  echo "<td>$nom</td>";
  echo "<td>$prenom</td>";
  echo "<td>$date</td>";
  echo "<td>";
  echo "<a href='eleve_edite.php?id=$id'>voir</a>";
  echo "</td>";
  echo "</tr>";
}

echo "</table>";

mysql_close($liendb);

include("bas.inc.php");

?>
```

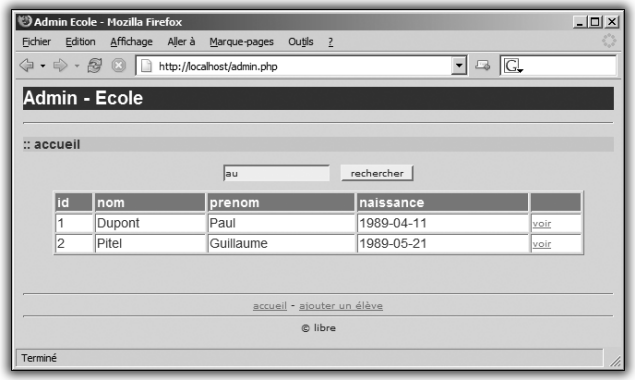

Figure 11.10 : *Résultat de la recherche sur le mot 'au'*

Dans le cas où un mot-clé a été tapé, vous entrez dans la condition `if`, et la variable `$clause` est initialisée, par exemple :

```
" WHERE INSTR(nom,'au') OR INSTR(prenom,'au')")
```

La variable `$sql` est ensuite initialisée. Si la variable `$clause` n'a pas été créée, la requête reste `"SELECT * FROM $table_eleve"`, sinon la clause est ajoutée :

```
"SELECT * FROM $table_eleve WHERE INSTR(nom,'au') OR
✂ INSTR(prenom,'au')"
```

Si vous tapez à présent le mot-clé `dup`, vous n'obtenez aucun résultat car `INSTR` est sensible à la casse.

Pour lister tous les élèves, sans vous préoccuper de la différence minuscules/majuscules, préférez l'opérateur de comparaison `LIKE` :

Listing 11-11 : Test non sensible à la casse

```
if  (isset($motclef))
{
  $clause = "WHERE nom LIKE '".$_REQUEST['motclef']."' OR
  ✂ prenom LIKE '".$_REQUEST['motclef']."'";
}
```

Il est important de ne jamais hésiter à composer vos requêtes à partir d'éléments disparates. Cela permet d'aboutir à des fonctionnalités très intéressantes sans trop d'efforts.

Définir la fonction de tri

L'autre fonctionnalité très utile en SGBD est la possibilité d'obtenir les données triées. Pour illustrer cette fonctionnalité, vous allez ajouter au script *admin.php* la possibilité de trier les élèves par noms, prénoms ou dates de naissance.

Pour préciser un ordre, il suffit d'ajouter la commande `ORDER BY` à la fin de la requête. Ainsi, si vous souhaitez classer les élèves par noms, écrivez :

```
SELECT * FROM eleve ORDER BY nom
```

Il est possible de réaliser un classement multiple en définissant un `ORDER BY` sur plusieurs colonnes. Si vous souhaitez classer les élèves par noms et, pour un même nom, par dates de naissance, écrivez :

```
ORDER BY nom, naissance
```

ORDER BY et WHERE

Les commandes `ORDER BY` et `WHERE` ne sont pas exclusives. Les deux peuvent cohabiter. Il est cependant nécessaire de mettre le `WHERE` avant l'`ORDER BY`.

Tout comme pour la recherche, vous voyez qu'il suffit d'enrichir la requête si une variable `$ordre` est transmise :

```
<?php

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

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

?>

<p align="left"> :: accueil</p>

<form action="admin.php" method="post">
<input type="text" name="motclef" value="<?php echo
✂ $motclef; ?>"> <input type="submit"
✂ value="rechercher">
</form>

<table width="90%" align="center" border="1">
  <tr>
    <td class="intitule">id</td>
    <td class="intitule">nom</td>
    <td class="intitule">prenom</td>
    <td class="intitule">naissance</td>
    <td class="intitule"> </td>
  </tr>


<?php

if (isset($motclef))
{
    $clause = " WHERE nom LIKE '%$motclef%' OR
    ✂ prenom LIKE '%$motclef%'";
}

if (isset($ordre))
{
    $orderby = " ORDER BY $ordre";
}
```

```
$sql = "SELECT * FROM $table_eleve" . $clause .
✂ $orderby;
$resultat = mysql_query ($sql);

while ($eleve = mysql_fetch_array ($resultat))
{
    $id = $eleve['ideleve'];
    $nom = $eleve['nom'];
    $prenom = $eleve['prenom'];
    $date = $eleve['naissance'];
    echo "<tr>";
    echo "<td>$id</td>";
    echo "<td>$nom</td>";
    echo "<td>$prenom</td>";
    echo "<td>$date</td>";
    echo "<td>";
    echo "<a href=eleve_edite.php?id=$id>voir</a>";
    echo "</td>";
    echo "</tr>";
}

echo "</table>";

?>

<form action="admin.php" method="post">
  <select name="ordre">
      <option value="nom">classement par nom</option>
      <option value="prenom">classement par
      ✂ prénom</option>
      <option value="naissance">classement par
      ✂ date</option>
  </select>
     <input type="submit" value="trier">
</form>

<?php
mysql_close($liendb);
include("bas.inc.php");
?>
```

LIMIT

Il est possible de limiter le nombre de résultats retournés par un `SELECT` avec `LIMIT`. Ainsi, `SELECT * FROM eleve LIMIT 10` retourne 10 élèves. La commande `LIMIT` peut être complétée par un `WHERE` et un `ORDER BY`, mais doit cependant être placée à la fin de la requête.

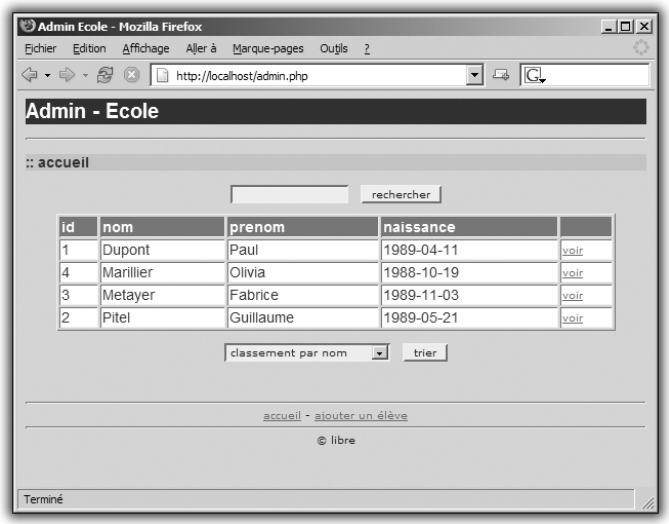

Figure 11.11 : *Classement par prénoms*

11.6. Check-list

- Une authentification HTML nécessite de travailler au niveau de l'en-tête HTTP et d'utiliser de la fonction `header()`.
- Les requêtes de type `UPDATE` permettent de mettre à jour des informations de la base.
- Les requêtes de type `DELETE` permettent de supprimer un enregistrement de la base.
- Il est toujours possible de limiter la portée d'une requête SQL en l'enrichissant d'une clause `WHERE`.
- La factorisation est une notion essentielle en informatique. Il est ainsi possible, pour un code PHP utilisé en différents endroits, de le placer dans un fichier extérieur et de l'inclure via la fonction `include()`.
- Au niveau de la présentation de la page, il est préférable de passer par un fichier CSS plutôt que placer des consignes visuelles au sein même des éléments qui la composent. L'évolution sera en effet largement facilitée.

Chapitre 12

La gestion des fichiers

Dans le cadre de ce chapitre, nous nous intéresserons principalement à la manipulation des fichiers en PHP. Vous étudierez dans un premier temps les différentes actions qui peuvent être réalisées sur un fichier, puis la présentation de différents formats de fichiers qui peuvent être générés avec PHP : GZIP (archives), CSV (Excel), PNG (image), PDF, SWF (Flash).

12.1. Manipuler des fichiers

Comme beaucoup de langages de programmation, PHP permet de manipuler les fichiers. Toutes les opérations habituelles sont disponibles :

- lire
- écrire
- renommer
- déplacer
- copier
- supprimer

Prenez un premier exemple : quand vous envoyez un fichier *hello.html* sur votre serveur d'hébergement (par FTP ou via le navigateur), une commande PHP vous permet en une ligne de le renommer :

Listing 12-1 : Renomme le fichier hello.html en hello.htm

```
rename("hello.html","hello.htm");
```

Ces fonctions permettent donc de manipuler les fichiers distants (présents sur le serveur d'hébergement).

Les fichiers de cache

Ces fonctions rendent possible le stockage de données dans des fichiers. Cela va a priori à l'encontre de ce qui a été dit précédemment au sujet des bases de données, à savoir qu'il est souvent préférable d'utiliser une base de données pour stocker des informations.

Il faut cependant garder à l'esprit que les requêtes SQL utilisent beaucoup de ressources et qu'il est judicieux, pour accroître les performances d'un script, de les réduire le plus possible.

La mise en œuvre d'une base de données est en effet très lourde à gérer pour le système : la couche réseau, le système de fichiers, ainsi que le SGBD sont mis à contribution.

Impact des requêtes SQL

Pour ceux qui souhaitent optimiser leur code, sachez que la diminution du nombre de requêtes SQL est de loin ce qui peut améliorer le plus les performances de vos scripts. Les optimisations sur les boucles et les diverses finesses du langage sont, en comparaison, négligeables.

Pour alléger les ressources du serveur et accélérer vos scripts, il peut donc être intéressant de mettre en place un système de cache.

Le principe d'un fichier de cache consiste à contenir de manière statique et pendant une durée déterminée des informations qui ne sont pas susceptibles d'être modifiées régulièrement, mais qui ont de fortes chances d'être réclamées fréquemment.

Pour créer ce fichier, il suffit d'extraire les informations de la base de données et de les écrire dans un fichier selon une certaine norme. L'avantage réside dans le fait que ce fichier devient aussi facile à traiter pour le serveur qu'un simple fichier *.html*. Dès lors, vous n'avez pas à craindre que l'afflux soudain d'un grand nombre d'internautes freine l'ensemble du site.

Imaginez qu'un site web partenaire ait besoin d'accéder aux noms et aux prénoms des élèves de l'école. Ce site n'a pas accès à la base de données et ne peut donc pas extraire directement les informations dont il a besoin. Vous allez par conséquent mettre à sa disposition un fichier de cache contenant la liste des noms et des prénoms. Ce fichier sera disponible à l'adresse **http://localhost/eleves.cache**. Votre politique de cache sera alors de mettre en œuvre deux scripts :

- Le script *ecrit-cache.php* va permettre d'écrire les informations dans le fichier.
- Le script *lit-cache.php*, sera utilisé par le site partenaire pour accéder à votre fichier via le Web (en sachant évidemment que votre partenaire dispose du PHP).

L'écriture

Commencez par le stockage de vos données dans le fichier *eleves.cache* :

Listing 12-2 : Le script ecrit-cache.php

```
<?php

$fichier = fopen ("eleves.cache", "w+");
$liendb = mysql_connect("localhost", "root", "");
mysql_select_db ("test");
$sql = "SELECT * FROM eleve";
$resultat = mysql_query ($sql);
while ($eleve = mysql_fetch_array ($resultat)) {
  $nom = $eleve['nom'];
  $prenom = $eleve['prenom'];
  fwrite($fichier, "$nom,$prenom\n");
}
mysql_close($liendb);
fclose($fichier);
echo "fichier enregistré";

?>
```

Comme vous pouvez vous en rendre compte, le principe de fonctionnement est assez similaire à celui des bases de données.

Il convient, dans un premier temps, de créer un « descripteur de fichier » avec `fopen()` : `$fichier`.

Le premier argument de `fopen()` correspond au nom du fichier avec lequel vous allez travailler. Dans cet exemple, le nom est simple : `eleves.cache`. Celui-ci peut cependant être moins évident. Il peut ainsi être précédé d'un chemin (*path*). Si vous écrivez `../eleves.cache`, cela signifie que vous ouvrez le fichier *eleves.cache* dans le répertoire au-dessus du répertoire courant.

Répertoire courant

Le répertoire courant est le répertoire où se trouve le script PHP qui a été appelé. En écrivant `eleves.cache`, vous ouvrez donc un fichier qui sera situé dans le même répertoire que le script d'appel.

Le nom du fichier peut aussi être une URL : il est ainsi possible d'ouvrir un fichier présent sur le Web (`http://localhost/eleves.cache`) ou sur un serveur FTP (`ftp://ftp.kernix/eleves.cache`).

Le deuxième argument transmis à `fopen()` correspond au mode d'ouverture du fichier. Plusieurs modes existent :

Tableau 12.1 : *Les modes d'ouverture d'un fichier avec fopen()*

Mode d'ouverture	Signification
`r`	Ouverture en lecture seule (*read*)
`w`	Ouverture en écriture seule (*write*) ; le fichier est créé s'il n'existe pas
`r+`	Ouverture en lecture/écriture
`w+`	Ouverture en lecture/écriture ; le fichier est créé s'il n'existe pas et, s'il existe, il est préalablement mis à zéro
`a`	Ouverture en écriture (*append*) ; le fichier est créé s'il n'existe pas et, s'il existe, le pointeur est placé à la fin du fichier
`a+`	Ouverture en lecture/écriture ; le fichier est créé s'il n'existe pas et, s'il existe, le pointeur est placé à la fin du fichier

Dans ce cas, choisissez le mode `"w+"` car vous devez écrire dans le fichier *eleves.cache*.

Lors du premier appel au script *ecrit-cache.php*, le fichier est créé ; lors des appels suivants, vous n'ajouterez pas de données aux fichiers (`"a+"`), mais remplacerez le contenu par de nouvelles données.

Une fois le fichier ouvert, il est possible d'y écrire avec la fonction `fwrite()`. Cette fonction prend, elle aussi, deux arguments : le premier argument est le descripteur de fichier et le second correspond à la chaîne de caractères que vous souhaitez stocker dans le fichier. Il est important de placer un retour chariot (`\n`) à la fin de la chaîne si l'on ne veut pas avoir toutes ces données sur la même ligne.

Il convient de noter qu'une manipulation de fichier se conclut toujours par la fermeture du fichier avec `fclose()`.

La lecture

Passons maintenant à la lecture du fichier . Nous supposons dans un premier temps que le script *lit-cache.php* est sur le même compte que le script *ecrit-cache.php*.

La lecture locale

Vous pouvez ouvrir le fichier en lecture (`"r"`) en l'appelant directement par son nom :

Listing 12-3 : Le script lit-cache.php

```
$fichier = fopen ("eleves.cache", "r");
$eleves = fread ($fichier, filesize("eleves.cache"));
fclose ($fichier);
echo $eleves;
```

Vous faites dans ce cas appel à la fonction `fread()`. Cette fonction prend deux arguments : le premier est le descripteur de fichier et le second correspond à la quantité de données (en octets) que vous souhaitez lire dans le fichier. Faites appel à la fonction `filesize()` pour trouver cette valeur. Cette fonction retourne la taille du fichier, dont le nom lui est passé en paramètre. Grâce à cette fonction, vous lisez l'intégralité du fichier avec un seul appel à `fread()`.

La lecture distante

Supposons maintenant que ce script soit hébergé sur un autre serveur. Le fichier doit cette fois être ouvert via le Web. Passez à `fread()` le nom `"http://localhost/eleves.cache"`.

Ouverture de fichier web

Il est impossible d'ouvrir des fichiers web en écriture car cela reviendrait à permettre à n'importe qui de modifier le contenu de vos pages !

À la différence de la version locale, vous ne pouvez pas utiliser la fonction `filesize()` avec un fichier distant. Vous êtes donc obligé de lire le fichier morceau par morceau avec une boucle `while` :

```
$eleves = "";
while ($str = fread($fichier,16))
{
  $eleves .= $str;
}
```

À chaque itération de la boucle `while`, vous complétez la variable `$eleves` avec 16 octets du fichier *eleves.cache*. C'est le descripteur de fichier qui garde en mémoire l'endroit où vous vous trouvez dans le fichier. La boucle s'arrête naturellement dès qu'il n'y a plus rien à lire dans le fichier.

Si vous souhaitez revenir au début d'un fichier, vous avez (au moins) deux possibilités : la première, peu élégante, consiste à fermer le fichier et à le rouvrir ; la seconde consiste à utiliser la fonction `rewind()`.

`rewind()` prend en paramètre un descripteur de fichier. Cette deuxième solution est beaucoup plus intéressante car moins consommatrice en ressources.

Écrivez maintenant le script *lit-cache.php* qui va lire le fichier *eleves.cache* via le Web :

Listing 12-4 : ouverture d'un fichier web

```
<?php

$fichier = fopen ("http://localhost/eleves.cache", "r");
$eleves = "";
while ($str = fread($fichier,16))
{
  $eleves .= $str;
}
fclose($fichier);
echo $eleves;

?>
```

Vous prenez conscience, avec cet exemple, de l'extrême simplicité du langage PHP, même pour réaliser des tâches a priori complexes sur les fichiers. Voyez la fonction `fread()` avec un autre exemple :

```
<?php

$url  = "http://www.google.com";

$fichier = fopen ($url, "r");
while ($str = fread ($fichier, 16))
{
```

```
  $src .= $str;
}
fclose($fichier);

echo "<textarea rows=10 cols=80>$src</textarea>";

?>
```

Ce script récupère les sources de la page `http://www.google.com` et les affiche dans une zone de texte.

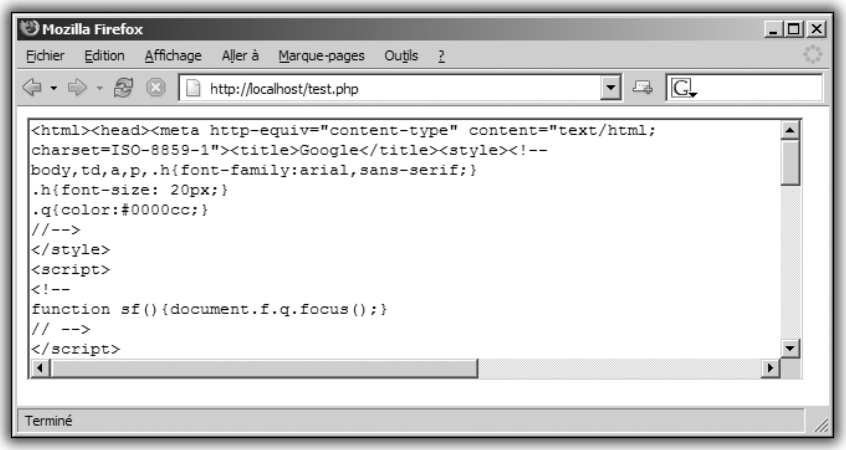

Figure 12.1 : *Affichage des sources de www.google.com*

Il est bien évidemment possible d'ouvrir plusieurs fichiers au même moment. Il suffit pour cela d'utiliser des noms de descripteurs différents.

Écrivez un petit script qui lit le fichier *eleves.cache* (via le Web) et qui génère un fichier *eleves2.cache*, où, cette fois, le nom apparaît après le prénom, avec un deux-points comme caractère de séparation. Utilisez les fonctions `file_get_contents()` (qui retourne le contenu d'un fichier) et `strlen()` pour trouver la taille du fichier distant.

Listing 12-5 : Le script lit-cache.php

```
<?php

$fichier1 = fopen ("http://localhost/eleves.cache", "r");
$fichier2 = fopen ("eleves2.cache", "w+");
$taille = strlen (file_get_contents("http://localhost
✂ /eleves.cache"));
$eleves = fread ($fichier1,$taille);
$tab_eleves = split ("\n",$eleves);
```

```
foreach ($tab_eleves as $ligne) {
  list ($nom,$prenom) = split(",",$ligne);
  fwrite($fichier2,"$prenom:$nom");
}
fclose ($fichier2);
fclose ($fichier1);

?>
```

Ne laissez pas traîner ce fichier *eleves2.cache*. La fonction `unlink()` permet de l'effacer :

```
unlink("eleves2.cache");
```

Vous vous apercevez, avec cette dernière fonction (ainsi qu'avec la fonction `rename()` vue plus haut), qu'il est inutile d'utiliser `fopen()` quand vous ne cherchez pas à travailler sur le contenu du fichier.

Les fichiers modèles : templates

Étudions enfin un dernier exemple qui illustre la lecture de fichier avec PHP. Vous avez vu dans les chapitres précédents l'importance, en programmation web, de la séparation entre le contenu et le visuel. L'utilisation de fichiers templates (modèles) est une bonne façon d'y parvenir. Le principe est très simple : plutôt que de mettre dans le même fichier le code PHP et la page HTML, stockez séparément ces deux parties. Le fichier *.html* comporte des balises indiquant ainsi où les informations issues de la base doivent être placées. Le script PHP se charge quant à lui de lire le fichier et de compléter les manques avec les données qu'il est allé récupérer dans la base (ou n'importe où ailleurs).

Dans cet exemple, le fichier *modele1.html* comporte deux balises qui seront remplacées :

- `%%NBELEVES%%` correspond au nombre d'élèves.
- `%%TABELEVES%%` sera remplacé par la liste des élèves.

Listing 12-6 : Le contenu du fichier modèle modele1.html

```
<html>
<body bgcolor="white">

<p>%%NBELEVES%% élèves</p>

<hr/>
```

```
<table align="center" border="1">
 <tr>
  <td>
   <font size="small">%%LISTELEVES%%</font>
  </td>
 </tr>
</table>

<hr/>

</body>
</html>
```

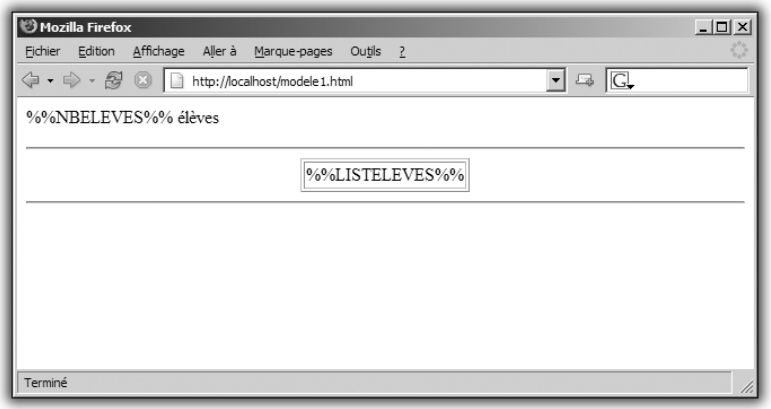

Figure 12.2 : *Le premier modèle*

Écrivez maintenant ce que l'on qualifie généralement de *wrapper*, c'est-à-dire le script qui va construire la page finale à partir du modèle :

```
<?php

$fichier = fopen ("modele1.html", "r");
$html = fread ($fichier, filesize("modele1.html"));
fclose ($fichier);

$liendb = mysql_connect("localhost", "root", "");
mysql_select_db ("test");
$sql = "SELECT * FROM eleve";
$resultat = mysql_query ($sql);
$nb = mysql_num_rows($resultat);
$eleves = "";
while ($eleve = mysql_fetch_array ($resultat)) {
  $nom = $eleve['nom'];
  $prenom = $eleve['prenom'];
  $eleves .= "$nom,$prenom<br>";
}
```

```
mysql_close ($liendb);

$html = str_replace ("%%NBELEVES%%","$nb",$html);
$html = str_replace ("%%LISTELEVES%%",$eleves,$html);

echo $html;

?>
```

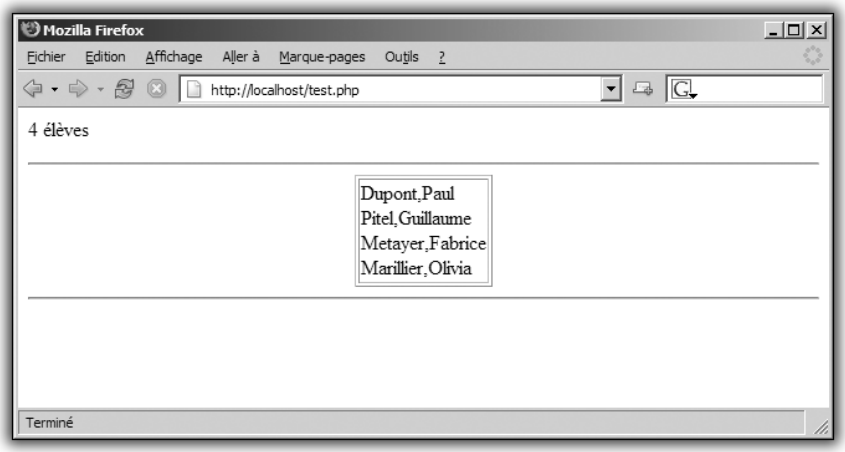

Figure 12.3 : *Les balises ont été remplacées*

Le grand avantage de ce mode de fonctionnement est de permettre de changer l'aspect visuel sans modifier le code. Cela rend la mise à jour de la page extrêmement aisée. Si vous travaillez avec un client, celui-ci peut réaliser un nouveau design très simplement sous Dreamweaver et vous transmettre le fichier en tant que modèle.

Imaginez maintenant que plusieurs partenaires veuillent pointer sur cette page, mais que chacun désire que la page soit habillée à sa façon. Rien de plus simple : vous prévoyez dans le script que le partenaire transmet en paramètre l'adresse du template, puis le script ouvre ce template via le Web pour générer la page.

Écrivez un deuxième template :

Listing 12-7 : autre modèle

```
<html>
<body bgcolor="#C7D0D9">

<table align="center" width="100%">
<tr><td bgcolor="white">%%NBELEVES%%</td></tr>
```

```
<tr><td bgcolor="#CCCCCC">%%LISTELEVES%%</td></tr>
</table>

</body>
</html>
```

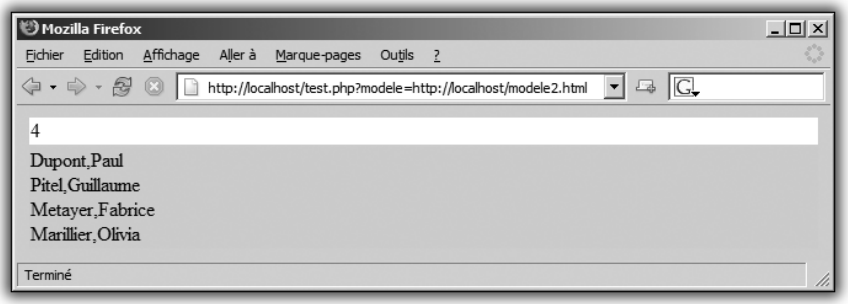

Figure 12.4 : *Le deuxième modèle*

Modifiez maintenant le script pour qu'il accepte en paramètre l'adresse du fichier modèle :

Listing 12-8 : le nom du modèle est transmis en paramètre

```
<?php

$fichier = fopen ($_REQUEST['modele'], "r");
while ($str = fread ($fichier, 16)) {
  $html .= $str;
}
fclose ($fichier);

$liendb = mysql_connect("localhost", "root", "");
mysql_select_db ("test");
$sql = "SELECT * FROM eleve";
$resultat = mysql_query ($sql);
$nb = mysql_num_rows($resultat);
$eleves = "";
while ($eleve = mysql_fetch_array ($resultat)) {
  $nom = $eleve['nom'];
  $prenom = $eleve['prenom'];
  $eleves .= "$nom,$prenom<br>";
}
mysql_close ($liendb);

$html = str_replace ("%%NBELEVES%%","$nb",$html);
$html = str_replace ("%%LISTELEVES%%",$eleves,$html);

echo $html;
```

```
?>
```

Il devient alors possible de spécifier et d'héberger son propre modèle.

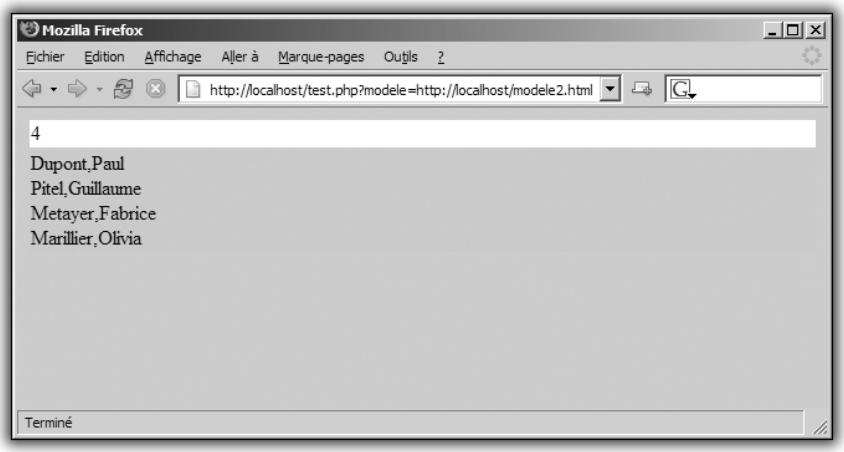

Figure 12.5 : *Page obtenue en tapant l'URL http://localhost/test.php?modele=http://localhost/modele2.html*

Cette solution sera de plus en plus courante car les applicatifs en ligne vont devenir peu à peu des services et il sera important que les clients de ces services puissent les adapter.

12.2. Créer des fichiers spéciaux

Vous avez vu qu'il était très simple, en PHP, de manipuler des fichiers. Vous allez voir, dans cette partie, qu'il est possible, aussi simplement, de générer des fichiers spéciaux : fichiers compressés, fichiers exploitables sous Excel, fichiers image.

Les fichiers compressés

Le PHP dispose de fonctions permettant de compresser des données. En les compressant, ces données prennent moins de place et sont donc beaucoup plus rapides à transférer.

Le format de compression par défaut en PHP est GZIP. Ce format est peu connu en dehors du monde Unix, mais il dispose de nombreux avantages :

- L'algorithme de compression est excellent.
- Des outils de décompression existent sur tous les systèmes d'exploitation (Stuffit Expander sous Mac, WinZip sous Windows).

Ce premier exemple aura pour objet de récupérer les noms et les prénoms des élèves, d'en faire une liste et de retourner la liste compressée afin de la stocker sur votre disque :

```
<?php
$liendb = mysql_connect("localhost", "root", "");
mysql_select_db ("test");
$sql = "SELECT * FROM eleve";
$resultat = mysql_query ($sql);
$liste = "";
while ($eleve = mysql_fetch_array ($resultat))
{
  $nom = $eleve['nom'];
  $prenom = $eleve['prenom'];
  $liste .= "$nom,$prenom\n";
}
mysql_close($liendb);
echo gzencode($liste);
?>
```

En appelant le fichier directement, vous obtenez un contenu illisible.

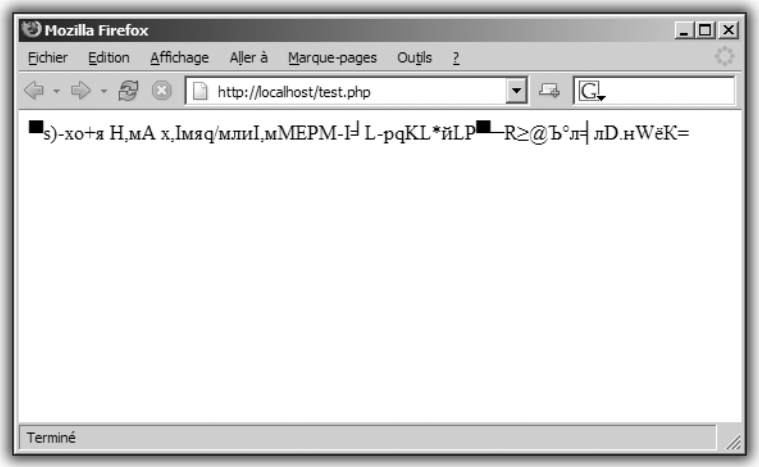

Figure 12.6 : *Contenu compressé illisible*

Le problème vient du fait que le fichier que vous appelez est considéré par le navigateur comme un simple fichier *.html*. Son contenu brut est

donc affiché. Il est cependant possible de donner des directives au navigateur pour lui indiquer que le contenu qu'il reçoit est compressé et qu'il doit par conséquent le sauvegarder plutôt que l'afficher.

La commande utilisée pour y parvenir est `header()`. Cette fonction permet de modifier l'en-tête d'un fichier transmis à un navigateur. Avec cette fonction, il est possible d'intervenir sur :

- les directives de cache (durée de conservation d'un fichier sur le disque) ;
- les dates de création ;
- le type de contenu du fichier.

Le rôle de la fonction `header()` se limite en fait à passer des commandes HTTP. La commande permettant de préciser le contenu du fichier envoyé est `Content-type`. Par défaut, le contenu est de type `text/html`. Si vous souhaitez indiquer que le fichier est compressé au format GZIP, il suffit d'écrire :

```
header("Content-type: application/x-gzip");
```

Il faut aussi ajouter une autre directive afin d'indiquer quel nom devra prendre le fichier à sauvegarder :

```
header('Content-Disposition:  attachement;
✂ filename="eleves.txt.gz"');
```

En-tête HTTP et courriel

La modification de l'en-tête HTTP d'un fichier est à rapprocher des ajouts que vous aviez faits dans l'en-tête d'un courriel pour préciser qu'il était en HTML. L'avantage de PHP est de permettre d'accéder à ces zones cachées et ainsi d'aller plus loin dans le code et les fonctionnalités. Il est par contre nécessaire de connaître les différents protocoles (HTTP, SMTP) pour pouvoir en tirer véritablement parti.

Complétez votre script avec ces deux nouvelles lignes :

```
<?php
header("Content-type: application/x-gzip");
header('Content-Disposition:  attachement;
✂ filename="eleves.txt.gz"');
$liendb = mysql_connect("localhost", "root", "");
mysql_select_db ("test");
$sql = "SELECT * FROM eleve";
$resultat = mysql_query ($sql);
```

```
$liste = "";
while ($eleve = mysql_fetch_array ($resultat))
{
  $nom = $eleve['nom'];
  $prenom = $eleve['prenom'];
  $liste .= "$nom,$prenom\r\n";
}
mysql_close($liendb);
echo gzencode($liste);
?>
```

Désormais, vous obtenez bien le comportement désiré. Le navigateur vous propose de sauvegarder le fichier sur le disque.

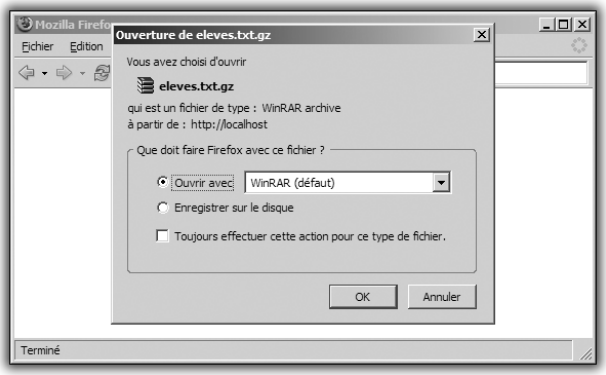

Figure 12.7 : *Le fichier compressé peut être sauvé sur le disque*

Une fois que le fichier se trouve sur le disque, il est possible de le décompresser :

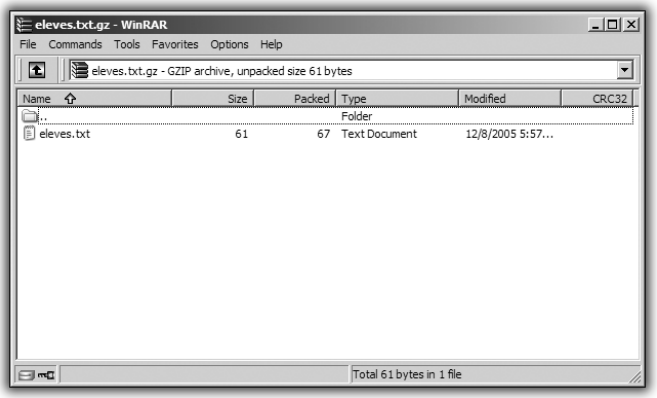

Figure 12.8 : *Ouverture du fichier compressé avec WinRAR*

Le format BZIP2

Un autre format de compression est disponible : BZIP2. L'avantage de ce format est d'offrir un meilleur taux de compression que le GZIP. Du fait de la jeunesse de ce format, les décompresseurs de fichiers GZIP ne sont pas très répandus. Cela interdit donc son usage si l'on souhaite conserver une certaine forme de compatibilité avec le plus grand nombre d'internautes.

Grâce à la commande `header()`, vous avez pu forcer le navigateur à sauver les données qui lui ont été transmises. Cette technique peut se révéler très intéressante si vous souhaitez associer un événement au téléchargement d'un fichier.

Plutôt que donner un lien fixe sur le fichier (tel `http://localhost/fichier.exe`), vous avez la possibilité de passer par un script qui pourra par exemple, dans un premier temps, vérifier un mot de passe, puis logger le téléchargement dans une base avant de transmettre les données au navigateur.

La fonction header()

La commande `header()` doit être utilisée avant que toute donnée soit affichée. Si vous laissez une simple ligne vide avant l'ouverture `<?php`, vous obtiendrez une erreur. En effet, le caractère de saut de ligne aura été affiché avant la commande `header()`.

L'exemple suivant retourne le fichier *prog.exe* présent sur le serveur si l'internaute s'est identifié avec le login `"test"` :

```
<?php

if ($login == "test")
{
  header("Content-type: application/octetstream");
  header('Content-Disposition:  attachement;
  ✂ filename="prog.exe"');
  // possibilité ici d'enregistrer l'événement dans une
  ✂ base de données
  $fichier = fopen ("prog.exe", "r");
  while ($str = fread ($fichier, 1024))
  {
     echo $str;
  }
  fclose($fichier);
```

```
  return(1);
}

?>

<form>
<input name='login' /><input type='submit'
✂ value='telecharger' />
</form>
```

Les fichiers Excel

Il est très courant, lorsque vous développez un applicatif en ligne, d'avoir besoin de générer un fichier d'export des données stockées dans votre base. Un tel fichier peut être utilisé par un logiciel tiers pour réaliser certaines opérations spécifiques. Vous pourriez ainsi imaginer d'exporter les courriels de tous les élèves et utiliser ce fichier avec un logiciel de publipostage afin d'envoyer un bulletin d'information à l'ensemble de la classe.

Il s'avère que la quasi-totalité du marché de l'informatique grand public est occupée par Windows et qu'Excel est l'outil le plus utilisé pour traiter des listings de données. Vous allez donc présenter, dans cette partie, une manière de générer des fichiers compatibles avec Excel.

La méthode se révèle relativement simple : il suffit de générer un fichier CSV (Comma Separated Value), c'est-à-dire un fichier où les enregistrements sont organisés ligne par ligne et où les données sont séparées par des virgules :

- `"Dupont","Paul"`
- `"Pitel","Guillaume"`
- `"Metayer","Fabrice"`
- `"Marillier","Olivia"`

Grâce à la fonction `header()` et à la directive `Content-type`, vous êtes en mesure d'ouvrir vos données directement sous Excel sans avoir besoin de passer par un fichier temporaire :

```
<?php
header("Content-type: text/x-csv");
header('Content-Disposition:  attachement;
✂ filename="list.csv"');
$liendb = mysql_connect("localhost", "root", "");
```

```
mysql_select_db ("test");
$sql = "SELECT * FROM eleve";
$resultat = mysql_query ($sql);
$liste = "";
while ($eleve = mysql_fetch_array ($resultat))
{
  $nom = $eleve['nom'];
  $prenom = $eleve['prenom'];
  $liste .= "\"$nom\",\"$prenom\"\r\n";
}
mysql_close($liendb);
echo $liste;
?>
```

Figure 12.9 : *Le navigateur vous propose bien d'ouvrir le fichier avec Excel*

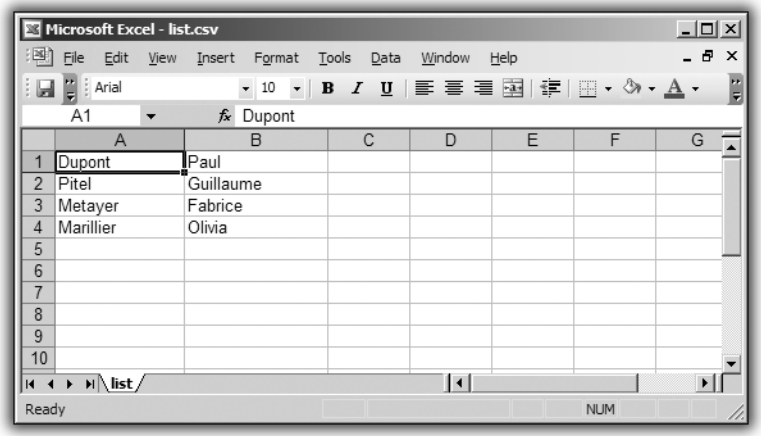

Figure 12.10 : *Les données sont directement lisibles dans Excel*

Dans Excel, il est possible de traiter vos données de manière très précise, d'utiliser des filtres, d'exécuter des opérations complexes. Il est souvent préférable de fonctionner avec une exportation de fichier vers un logiciel évolué plutôt que de recoder laborieusement une multitude de fonctionnalités qui ne serviront peut-être jamais. Il vaut mieux limiter l'applicatif en ligne à sa fonction première. Lorsque vous développez une boutique en ligne, il est sans doute préférable d'exporter les commandes vers un logiciel de comptabilité plutôt que de créer un module comptable.

Les fichiers Flash

PHP est un logiciel développé par des développeurs et pour des développeurs. Dans cette communauté, la notion de défi technique et de jeu prend souvent le pas sur les considérations purement économiques. Très rapidement, des modules plus ou moins farfelus sont ainsi venus enrichir la librairie d'extensions du PHP. Parmi ceux-ci, on peut trouver un module permettant de générer des fichiers SWF. Un fichier SWF est normalement généré avec le logiciel de Macromedia Flash. Avec PHP, il est cependant possible de créer dynamiquement des animations Flash.

Parmi tous les projets consacrés à ce sujet sur Internet, le projet Ming semble être le plus actif et le plus abouti. Grâce à cette extension, vous êtes en mesure, en quelques lignes de PHP, de réaliser de véritables animations qui peuvent interagir avec la souris, le clavier, être accompagnées de musique, etc.

Bien que Wamp Server soit livré par défaut avec l'extension permettant de générer des SWF, il ne l'autorise pas par défaut. L'étape préalable à la manipulation des SWF consiste donc à faire en sorte que PHP charge cette extension au démarrage. La première solution consiste à modifier le fichier *php.ini* en supprimant le point-virgule devant l'instruction suivante :

```
extension=php_ming.dll;
```

Le serveur Apache doit ensuite être redémarré pour prendre en compte cette modification.

La seconde solution consiste à passer par l'icône de Wamp Server et le menu **PHP extensions** pour sélectionner `php_ming`.

Extensions sous Linux

L'ajout d'extensions sous Linux se révèle beaucoup plus complexe. Vous êtes en effet obligé de récupérer les sources de l'extension ainsi que celles de PHP. Les deux devront être recompilées pour pouvoir disposer de cette nouvelle extension. Cette situation est essentiellement due au fait qu'il existe une multitude de distributions Linux et que ces dernières fonctionnent le plus souvent sur une grande variété de plateformes (PC, Mac, etc.). Cette diversité rend ainsi la distribution en mode binaire extrêmement fastidieuse pour les développeurs.

Le code suivant permet ainsi de jouer le fichier *test.mp3* (placé dans le même répertoire que le script) :

```
<?php
$m = new SWFMovie();
$m->setRate(12.0);
$m->streamMp3(fopen("test.mp3", "r"));
$m->setFrames(141);
header('Content-type: application/x-shockwave-flash');
$m->output();
?>
```

Notez le `Content-type : application/x-shockwave-flash`. En six lignes, vous êtes en mesure d'ajouter une dimension musicale à votre site. De plus, il est avantageux d'utiliser Flash dans ce cas car il s'agit de la technologie la plus portable : Flash existe en effet pour tous les navigateurs et les systèmes d'exploitation.

Le code suivant permet, quant à lui, de faire tourner sur lui-même un carré rouge :

```
<?php
$s = new SWFShape();
$s->setRightFill($s->addFill(0xff, 0, 0));
$s->movePenTo(-50,-50);
$s->drawLineTo(50,-50);
$s->drawLineTo(50,50);
$s->drawLineTo(-50,50);
$s->drawLineTo(-50,-50);
$p = new SWFSprite();
$i = $p->add($s);
for($j=0; $j<17; ++$j) {
  $p->nextFrame();
  $i->rotate(5);
}
$p->nextFrame();
$m = new SWFMovie();
```

```
$i = $m->add($p);
$i->moveTo(160,120);
$i->setName("blah");
$m->setBackground(0xff, 0xff, 0xff);
$m->setDimension(320,240);
header('Content-type: application/x-shockwave-flash');
$m->output();
?>
```

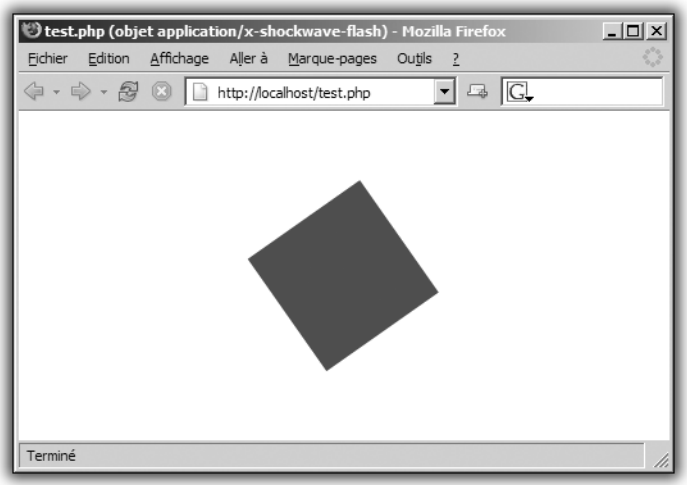

Figure 12.11 : *Flash généré par du code PHP*

Un des grands avantages de Flash est qu'il permet d'obtenir une interactivité en temps réel avec l'internaute. L'exemple suivant permet de déplacer un cercle vert sur un carré rouge et d'appeler la fonction Javascript extérieure `externAction()` lorsque le cercle est placé sur le carré :

Listing 12-9 : Script qui génère le FLASH

```
<?
ming_useswfversion(6);

$movie = new SWFMovie();
$movie->setRate(30.000000);
$movie->setDimension(500, 500);
$movie->setBackground(0xcc,0xcc,0xcc);

$square = new SWFShape();
$square->setLine(5,0,0,0);
$square->setRightFill(255,0,0);
$square->movePenTo(-75,-75);
$square->drawLine(150,0);
```

```
$square->drawLine(0,150);
$square->drawLine(-150,0);
$square->drawLine(0,-150);

$sprite = new SWFSprite();
$sprite ->add($square);
$sprite ->nextFrame();

$f = $movie->add($sprite);
$f->setName("square");
$f->moveTo(250,350);

$circle = new SWFShape();
$circle->setLine(5,0,0,0);
$circle->setRightFill(0,255,0);
$circle->drawCircle(50);

$sprite = new SWFSprite();
$sprite ->add($circle);
$sprite ->nextFrame();

$f = $movie->add($sprite);
$f->setName("circle");
$f->moveTo(250,100);

$movie->add(new SWFAction("
circle.onPress = function () {
  this.startDrag(1);
};
circle.onRelease = function () {
  stopDrag();
  if ((circle._x > (square._x - square._width/2)) &&
      (circle._x < (square._x + square._width/2)) &&
      (circle._y > (square._y - square._height/2)) &&
      (circle._y < (square._y + square._height/2))) {
    circle._x = square._x;
    circle._y = square._y;
    _root.getUrl('javascript:externAction()');
  }
};
"));

$movie->output();
?>
```

Listing 12-10 : page contenant la function javascript appelée et l'appel au FLASH

```
<html>
<head>
<title>Exemple Ming</title>
</head>
```

```
<script>
function externAction () {
  alert("Bravo !");
}
</script>
<body>
<p>Faire un drag&drop du cercle vert dans la carré rouge</p>
<object classid="clsid:D27CDB6E-AE6D-11cf-96B8-444553540000"
codebase="http://download.macromedia.com/pub/shockwave/cabs/
✂ flash/swflash.cab#version=6,0,29,0" width="300" height="300">
  <param name="movie" value="test.php">
  <param name="quality" value="high">
  <param name="wmode" value="transparent">
  <embed src="test.php" width="300" height="300"
  ✂ wmode="transparent" quality="high"
  pluginspage="http://www.macromedia.com/go/getflashplayer"
  ✂ type="application/x-shockwave-flash"></embed>
</object>
</body>
</html>
```

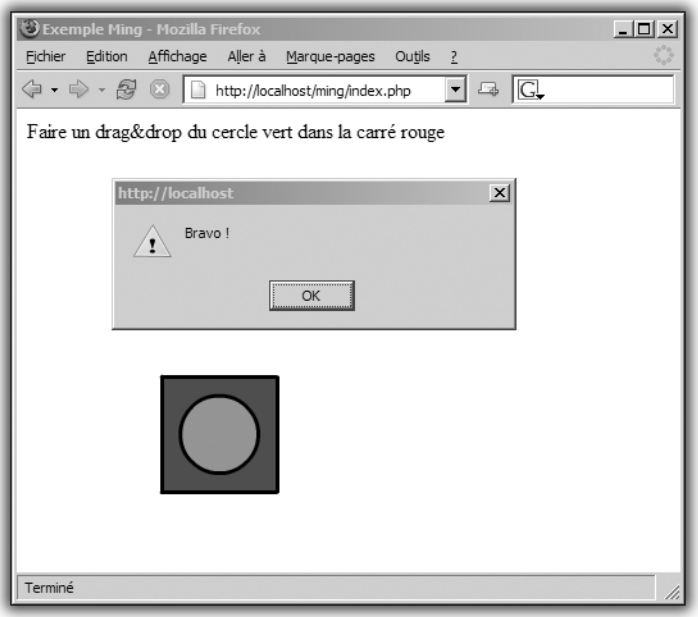

Figure 12.12 : *Déclenchement d'une action lorsque le cercle est placé sur le carré*

Si cette extension n'est pas présente chez votre hébergeur, n'hésitez pas à lui en parler : il y a de fortes chances pour qu'il vous l'installe

rapidement. Le projet Ming est hébergé à l'adresse **http://ming.sourceforge.net/**. Vous trouverez sur le site l'extension à télécharger ainsi qu'une documentation complète.

Les fichiers PDF

Le format de fichiers PDF est devenu depuis quelques années un standard pour les échanges de documents sur Internet. Le logiciel Acrobat Reader, de la société Adobe, est en effet présent sur toutes les plateformes (Windows, Mac, Unix). PHP donne la possibilité de créer à la volée des documents PDF. Cette technique peut être intéressante si vous souhaitez créer des documents téléchargeables à partir de données présentes dans votre base.

Comme pour le Flash, l'extension de gestion des PDF doit également être autorisée dans le fichier *php.ini*. La ligne à décommenter est cette fois : `extension=php_pdf.dll`. L'alternative consistant à sélectionner l'extension `php_pdf` dans le menu d'extensions de Wamp est également possible.

Écrivez maintenant un script qui va générer un document PDF proposant autant de pages que d'élèves, avec dans chaque page le nom et le prénom de l'élève :

Listing 12-11 : Script qui génère un document PDF

```
<?php

$pdf = pdf_new();

if (!pdf_open_file($pdf, "")) {
    print("error");
    exit(0);
};

pdf_set_info($pdf, "Author", "fx bois");
pdf_set_info($pdf, "Title", "exemple");
pdf_set_info($pdf, "Creator", "fx bois");
pdf_set_info($pdf, "Subject", "exemple");

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

```
{
  $nom = $eleve['nom'];
  $prenom = $eleve['prenom'];
  pdf_begin_page($pdf, 595, 842);
  pdf_add_bookmark($pdf, "fiche $nom",0,0);
  $font = pdf_findfont($pdf, "Helvetica", "host", 0);
  if ($font) pdf_setfont($pdf, $font, 12);
  pdf_set_value($pdf, "textrendering", 1);
  pdf_show_xy($pdf, "$nom $prenom", 50, 750);
  pdf_moveto($pdf, 50, 740);
  pdf_stroke($pdf);
  pdf_end_page($pdf);
}

mysql_close($liendb);

pdf_close($pdf);

$buf = pdf_get_buffer($pdf);
$len = strlen($buf);

header("Content-type: application/pdf");
header("Content-Length: $len");
header("Content-Disposition: inline; filename=fichier
✂ -eleve.pdf");
print($buf);
pdf_delete($pdf);

?>
```

Figure 12.13 : *Document PDF généré par du PHP ouvert dans Internet Explorer*

Plutôt que de l'afficher directement, il est également possible de sauver le fichier PDF sur le disque en remplaçant les lignes suivantes :

```
header("Content-type: application/pdf");
header("Content-Length: $len");
header("Content-Disposition: inline; filename=fichier
✂ -eleve.pdf");
print($buf);
```

… par :

```
$fichier = fopen ("fiche-eleve.pdf", "w+");
fwrite($fichier, $buf);
fclose($fichier);
```

Les fichiers image

À la différence des extensions vues précédemment, le module de génération d'images est un des modules les plus populaires de PHP, et il y a de très fortes chances pour que vous puissiez en disposer chez votre hébergeur.

La directive à décommenter dans `php.ini` est : `extension=php_gd2.dll`.

Les fondamentaux

Ce module est basé sur la librairie GD qui contient tout le code permettant de réaliser les nombreuses manipulations graphiques. De nombreux autres langages de programmation tels que Perl ou Python permettent de l'interfacer.

Développée prioritairement pour les environnements Unix/Linux, cette librairie existe aussi sous Windows et peut donc permettre la génération d'image sur ce système.

Comme pour PDF ou Flash, vous utilisez des fonctions du module pour générer le contenu de l'image ainsi que la fonction `header()` pour faire comprendre au navigateur que le fichier est de type image.

Le module est en mesure de générer différents formats d'images, chacun disposant de ses points forts.

- JPEG : très bonne compression pour des images complexes (photos), très répandu.

- GIF : très bonne compression pour des images simples, très répandu, gère la transparence.
- PNG : très bonne compression générale, gestion avancée de la transparence.
- WBMP : il s'agit d'un format utilisé pour le WAP.

Aujourd'hui, le format le plus complet est le PNG. Son seul inconvénient est sa jeunesse. De ce fait, certains navigateurs anciens ne sont pas en mesure de le lire.

Format GIF

Les versions les plus récentes de la librairie GD autorisent de nouveau la création d'images au format GIF. L'algorithme de compression (LZW) est en effet tombé dans le domaine public il y a peu de temps.

Votre premier script va générer une image noire contenant un rectangle blanc :

Listing 12-12 : Première création d'image

```
<?php

header ("Content-type: image/png");
$image = @Imagecreate (150, 150);
$noir = ImageColorAllocate ($image, 0, 0, 0);
$blanc = ImageColorAllocate ($image, 255, 255, 255);
ImageFilledRectangle ($image, 10, 10, 20, 20, $blanc);
ImagePng ($image);

?>
```

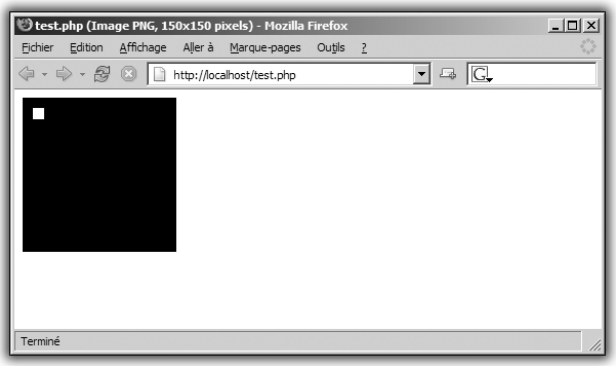

Figure 12.14 : *Génération d'une image noire contenant un rectangle blanc*

La création d'images est donc un processus très simple qui peut être divisé en plusieurs étapes :

1. Précisez que le fichier est de type image avec la commande `header()`. Dans ce cas, vous générez une image PNG en utilisant la directive `Content-type: image/png`. Si vous choisissez de générer une image JPEG, vous écrirez `Content-type: image/jpeg`, et `Content-type: image/gif` dans le cas d'une image GIF.
2. Créez l'image avec la fonction `Imagecreate()`. Vous transmettez à cette fonction la largeur et la hauteur et elle retourne un identifiant d'image : `$image`.
3. Définissez les couleurs utilisées dans l'image avec la fonction `ImageColorAllocate()`. Le premier argument est l'image elle-même (`$image`), les trois derniers arguments sont les composantes RVB (rouge, vert, bleu) de la couleur. La couleur noire est représentée par le triplet (0, 0, 0), la couleur blanche par (255, 255, 255) et la couleur rouge par (255, 0, 0). Cette manière de coder les couleurs est la même qu'en HTML, à la différence qu'ici les valeurs sont décimales alors qu'en HTML les couleurs sont hexadécimales : le rouge s'écrit par exemple `#FF0000` en HTML. La manière la plus simple de trouver les codes des couleurs est d'utiliser les palettes des outils de création graphique comme Photoshop, Paint Shop Pro ou Gimp. Il faut savoir que la première couleur définie servira de couleur de fond pour l'image. Dans le cas présent, il s'agit de la couleur noire (`$noir`).

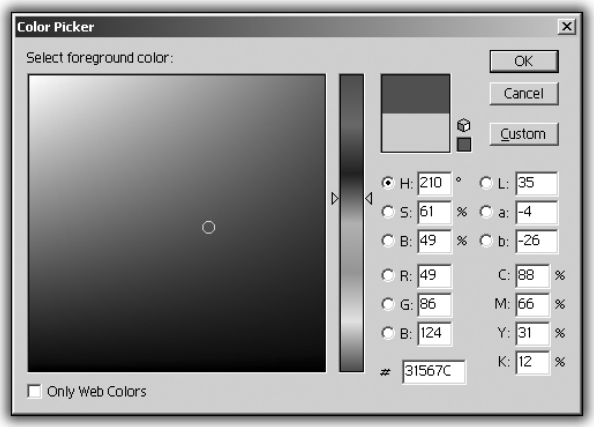

Figure 12.15 : *La palette de Photoshop donne le codage RVB (ou RGB : Red, Green, Blue)*

4 Composez l'image avec tous ses éléments : formes géométriques, textes, morceaux d'autres images. Dans ce script, choisissez simplement un carré blanc placé dans l'angle gauche de l'image. Utilisez, pour cela, la fonction `ImageFilledRectangle()` qui prend six arguments : l'identifiant de l'image, les deux coordonnées de l'angle supérieur gauche, les deux coordonnées de l'angle inférieur droit et enfin la couleur du rectangle.

Une liste complète des fonctions est fournie dans le chapitre consacré « Les fonctions PHP ».

5 La dernière étape consiste à générer le contenu de l'image avec la fonction `ImagePng()`. Les fonctions `ImageGIF`, `ImageJPEG`, `ImageWBMP` sont aussi disponibles pour les autres formats.

Dans l'exemple précédent, vous avez fait le choix d'appeler le script directement. Le navigateur n'affiche donc que l'image. Si vous souhaitez inclure l'image au sein d'une page web, il suffit de faire appel au script depuis la balise `<img>`. Écrivez la page HTML suivante dans ce sens :

Listing 12-13 : Inclusion d'image créée dynamiquement au sein d'une page HTML

```
<html>
<body>
<hr/>
<center>
 <img src="test.php" />
</center>
<hr/>
</body>
</html>
```

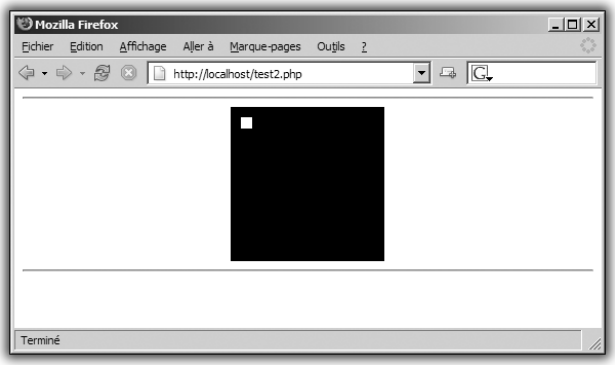

Figure 12.16 : *Image dans une page HTML*

Écrivez maintenant un deuxième exemple, où vous allez exploiter d'autres fonctionnalités de ce module (traçage de ligne, écriture) :

```
<?php

header ("Content-type: image/png");

$image = @Imagecreate (250, 200);

$couleur1 = ImageColorAllocate ($image, 212, 208, 200);
$couleur2 = ImageColorAllocate ($image, 198, 193, 182);
$couleur3 = ImageColorAllocate ($image, 79, 78, 74);


for ($i = 0; $i <= 200; $i += 10)
{
  ImageLine ($image, 0, $i, 250, $i, $couleur2);
}

ImageString ($image,5,10,20, "B o n j o u r",$couleur3);
ImageStringUp ($image,4,150,60, "Monde",$couleur3);

ImagePng ($image);

?>
```

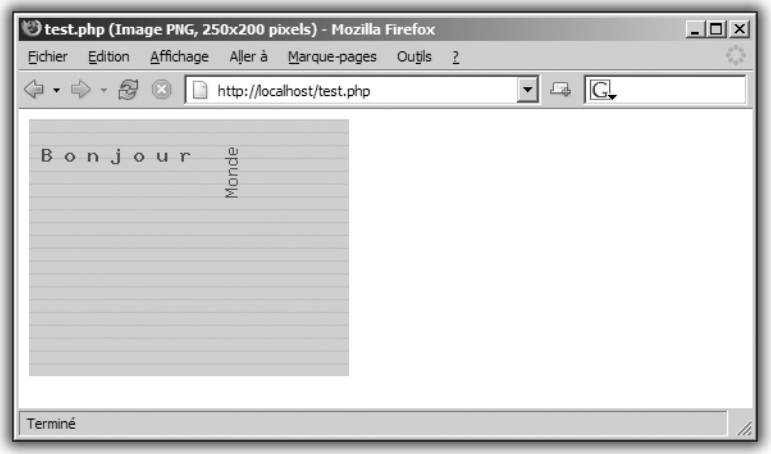

Figure 12.17 : *Autres fonctionnalités du module image*

Vous retrouvez dans cet exemple les principes vus plus haut.

La boucle `for()` permet de tracer des lignes tous les 10 pixels grâce à la fonction `ImageLine()`. Les arguments de cette fonction sont

l'identifiant de l'image, les coordonnées du point de départ, les coordonnées du point d'arrivée et la couleur.

Les textes sont, quant à eux, tracés avec les fonctions `ImageString()` et `ImageStringUp()` dont les arguments sont l'identifiant de l'image, la taille de la fonte, les coordonnées du point de départ, le texte et enfin la couleur. Le suffixe `Up` dans `ImageStringUp()` signifie que le texte est tracé verticalement.

Comme vous pouvez vous en apercevoir, la fonte est très sommaire. Heureusement, d'autres fonctions permettent de spécifier la fonte du texte : `ImageTTFText()`.

ImageTTFText sous Linux

Pour pouvoir utiliser cette fonction, PHP doit avoir été compilé avec le support de la librairie Freetype. Il n'est donc pas assuré que vous puissiez l'utiliser.

Cette fonction prend huit arguments : l'identifiant de l'image, la taille de la fonte, l'angle, les coordonnées de l'origine, la couleur, le nom de la fonte et le texte. Une fonte doit donc avoir été transférée sur le compte pour pouvoir l'utiliser. De nombreux sites web proposent le téléchargement gratuit de fontes : **www.fontfreak.com** ou **www.1001freefonts.com** en sont deux bons exemples.

```
<?php

header ("Content-type: image/png");

$image = @Imagecreate (250, 200);

$couleur1 = ImageColorAllocate ($image, 212, 208, 200);
$couleur2 = ImageColorAllocate ($image, 198, 193, 182);
$couleur3 = ImageColorAllocate ($image, 79, 78, 74);
$blanc    = ImageColorAllocate ($image, 255, 255, 255);

for ($i = 0; $i <= 200; $i += 10)
{
  ImageLine ($image, 0, $i, 250, $i, $couleur2);
}

ImageTTFText ($image,20,0,5,50,$couleur3,"font1.ttf","B o n
✂ j o u r");
ImageTTFText ($image,70,30,60,170,$blanc,"font2.ttf","Monde");
```

```
ImagePng ($image);

?>
```

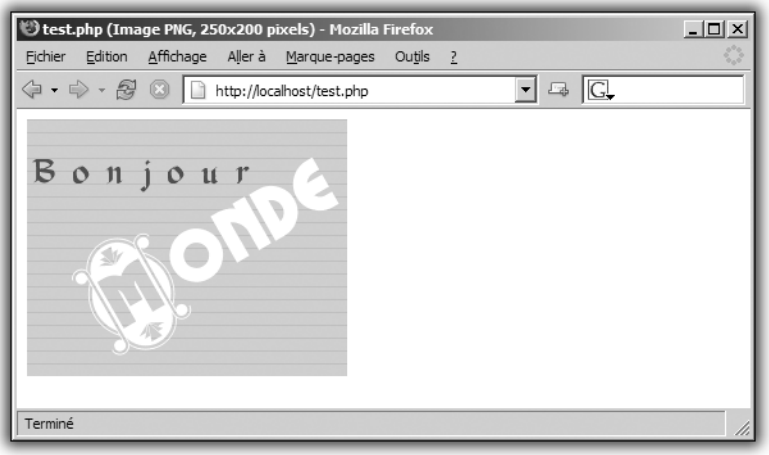

Figure 12.18 : *Des polices TrueType sont utilisées pour dessiner les textes*

Si la fonte se trouve dans un autre répertoire, par exemple *datas*, il faut alors écrire `datas/font1.ttf`.

Vous remarquez que l'écriture verticale est ici réalisée avec la même fonction que l'écriture horizontale. Vous vous contentez en fait de mettre un angle de 30 degrés.

PHP, en permettant d'accéder aux bases de données d'une part et de générer des images dynamiques d'autre part, est devenu un langage de choix pour la création d'outils de statistiques et de *reporting*.

Vous allez dans la suite de ce chapitre réaliser des graphiques reflétant les notes de vos élèves. Pour cela, il nous faut ajouter la notion de notes et de classement à votre base de données.

Les bases de données relationnelles

Comme vous l'avez vu précédemment, une base de données peut contenir plusieurs tables. L'avantage des BDD relationnelles telles que MySQL ou PostgreSQL est de permettre de créer des liens entre ces tables. On parle à propos de ces bases de SGBDR (système de gestion de bases de données relationnelles).

Supposons que vous vouliez enregistrer dans votre base de données des informations concernant l'élève, son professeur et ses notes. Spontanément, un utilisateur non expérimenté pourrait proposer d'organiser ses données dans une seule et grande table comportant les champs suivants : `nomeleve`, `prenomeleve`, `classement`, `moyenne`, `nomprofesseur`.

Cette manière de faire est à proscrire tout de suite !

Vous vous apercevez très vite, en remplissant la table, qu'il faut enregistrer le nom d'un même professeur pour tous les élèves de sa classe :

- `dupont, eric, 3, 16, mortier`
- `durand, michel, 12, 11, mortier`

Cela vous fait donc perdre à la fois du temps et de l'espace disque inutilement.

Un autre inconvénient de cette organisation est de ne pas permettre de conserver un historique des classements et des moyennes sur plusieurs trimestres.

Les BDD relationnelles viennent alors à votre secours. Si vous essayez de séparer intelligemment les données en plusieurs tables liées entre elles, ces inconvénients disparaissent instantanément. Vous pouvez en effet envisager l'organisation de vos données de cette manière : un professeur possède plusieurs élèves et un élève possède plusieurs moyennes et classements. En terme de BDD, on peut dire que la table des professeurs est liée à celle des élèves et que cette dernière est liée à celle des notes. Les liaisons correspondent en fait à la présence de l'identifiant (`idprof`) du professeur dans la table *eleve* et à l'identifiant (`ideleve`) de l'élève dans la table moyenne.

Vos trois tables sont donc :

- *prof* : `idprof`, `nom`, `prenom`
- *eleve* : `ideleve`, `idprof`, `nom`, `prenom`
- *exam* : `idexam`, `ideleve`, `moyenne`, `classement`, `trimestre`

Pour sélectionner le nom des élèves qui ont, d'une part, une moyenne supérieure à 10 au premier trimestre et qui, d'autre part, ont comme professeur M. Paul, vous pouvez utiliser la requête suivante :

```
SELECT eleve.nom AS nomeleve, prof.nom AS nomprof, exam
✂ .classement FROM prof, eleve, exam WHERE prof.nom =
✂ 'Paul' AND eleve.idprof = prof.idprof AND moyenne
✂ .ideleve = eleve.ideleve AND exam.classement >= '10' AND
✂ exam.trimestre = '1'
```

Analysons minutieusement cette requête. Comme vous avez besoin des trois tables, vous écrivez `FROM prof, eleve, exam`.

Les tables *prof* et *eleve* contiennent des colonnes intitulées de la même manière : *nom* et *prenom*. Il devient donc impossible, en travaillant sur ces deux tables en même temps, d'écrire `WHERE nom = 'dupont'` car le serveur de BDD ne peut savoir s'il doit faire une vérification sur la table *prof* ou sur la table *eleve*. Pour préciser de quel nom il s'agit, utilisez la notation `table.colonne`, dès que vous voulez spécifier une colonne dans une table en particulier :

```
WHERE prof.nom = 'Paul'
```

De la même manière, un obstacle apparaît si vous voulez sélectionner à la fois le nom de l'élève et le nom du professeur dans la même requête. La précédente méthode ne servirait ici à rien :

```
SELECT eleve.nom, prof.nom, exam.classement
```

Vous n'avez en effet aucun moyen, dans le résultat de la requête, de préciser si vous voulez le nom du professeur ou celui de l'élève. La technique consiste donc à renommer les colonnes au niveau de `SELECT : eleve.nom AS nomeleve`. De cette manière, en PHP, `$tab['nomeleve']` contient le nom de l'élève et `$tab['nomprof']` celui du professeur.

SELECT dans plusieurs tables

Il est donc très dangereux d'utiliser `SELECT table1.*, table2.*`, si *table1* et *table2* contiennent des colonnes de même intitulé.

Dans cet exemple, contentez-vous de créer une table *exam* :

```
CREATE TABLE exam (
  idexam int(10) unsigned NOT NULL auto_increment,
  ideleve int(10) unsigned NOT NULL default '0',
  moyenne float(5,2) unsigned NOT NULL default '0.0',
  trimestre tinyint(3) unsigned NOT NULL default '0',
  PRIMARY KEY  (idexam),
  KEY ideleve (ideleve)
)
```

Quelques remarques sur la création de cette table :

- La colonne des moyennes présente des nombres à virgule et est donc de type `FLOAT`. Il est possible avec les `FLOAT` de déterminer la précision. `(5,2)` signifie que le nombre peut s'étendre sur 5 caractères, avec 2 caractères après la virgule (`19,25` s'étend bien sur 5 caractères virgule comprise). Il est pertinent de bien fixer la précision d'un nombre à virgule directement dans MySQL, cela vous évite de faire des conversions en PHP.
- Vous avez placé un index sur la colonne *ideleve* car il y a de fortes chances pour que les requêtes sur cette table fassent intervenir précisément cette colonne.
- La présence du champ `ideleve` lie la table *eleve* à la table *exam* et crée une relation entre ces deux tables.

Ajoutez la variable `$table_exam` à votre fichier *variables.inc.php* et réalisez un petit script qui devra :

- créer la table ;
- associer une note par trimestre à chaque élève de la table.

Listing 12-14 : Création et initialisation de la table exam

```
<?php

include("variables.inc.php");

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

$sql = "CREATE TABLE $table_exam (
  idexam int(10) unsigned NOT NULL auto_increment,
  ideleve int(10) unsigned NOT NULL default '0',
  moyenne float(5,2) unsigned NOT NULL default '0.0',
  trimestre tinyint(3) unsigned NOT NULL default '0',
  PRIMARY KEY  (idexam),
  KEY ideleve (ideleve)
)";
mysql_query ($sql);

echo "création de la table effectuée<hr>";

$sql = "SELECT ideleve FROM $table_eleve";
$resultat = mysql_query ($sql);
$i = 0;
while ($eleve = mysql_fetch_array ($resultat))
```

```
{
  $tab_eleves[$i] = $eleve['ideleve'];
  $i++;
}

$i = 0;
srand ((double) microtime() * 1000000);
while ($ideleve = $tab_eleves[$i])
{
  echo "eleve [$ideleve] :";
  for ($j = 1; $j <= 3; $j++)
 {
    $moyenne = rand(0,20);
    echo " $moyenne";
    $sql = "INSERT INTO $table_exam
    ✂ (ideleve,moyenne,trimestre) VALUES
    ✂ ($ideleve,$moyenne,$j)";
    mysql_query ($sql);
 }
 echo "<br>";
 $i++;
}

echo "<hr>moyennes enregistrées<hr>";

mysql_close($liendb);

?>
```

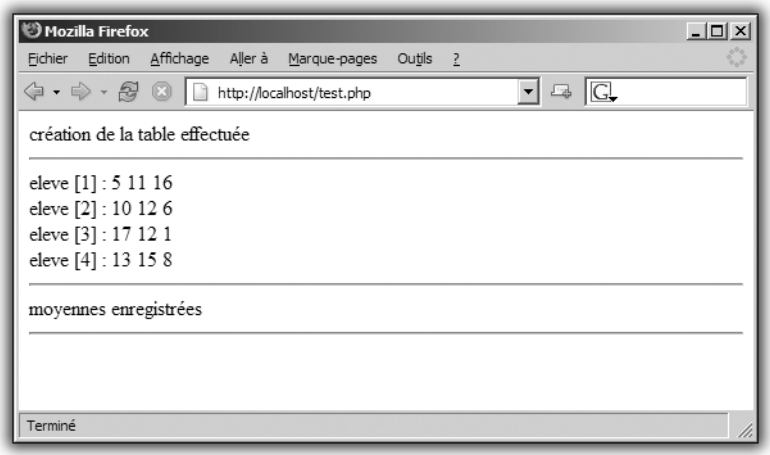

Figure 12.19 : *Initialisation de la table et des moyennes*

Créer des graphiques

Vous allez, dans cette partie, réaliser un script qui permettra de voir l'évolution de la moyenne de la classe sur les trois trimestres. Le graphique va être du type `bar chart`. Suivez pas à pas la façon de le construire.

L'image est de type PNG :

```
header ("Content-type: image/png");
```

Elle a pour dimensions 340 x 200 :

```
$image = @Imagecreate (340, 220);
```

La couleur de fond est le même vert que vous avez utilisé dans le back-office :

```
$fond = ImageColorAllocate ($image, 208, 216, 213);
```

D'autres couleurs sont utilisées pour les axes, les légendes, etc. Déclarez-les dès maintenant :

```
$coul_axes = ImageColorAllocate ($image, 11, 62, 43);
$coul_lignes = ImageColorAllocate ($image, 227, 235, 232);
$coul_legendes = ImageColorAllocate ($image, 11, 62, 43);
$coul_barres = ImageColorAllocate ($image, 42, 124, 94);
$coul_orange = ImageColorAllocate ($image, 207, 140, 53);
```

Deux axes, vertical et horizontal, sont présents :

```
imageline ($image,30,30,30,190,$coul_axes);
imageline ($image,30,190,320,190,$coul_axes);
```

Terminez le script avec la génération de l'image :

```
ImagePng ($image);
```

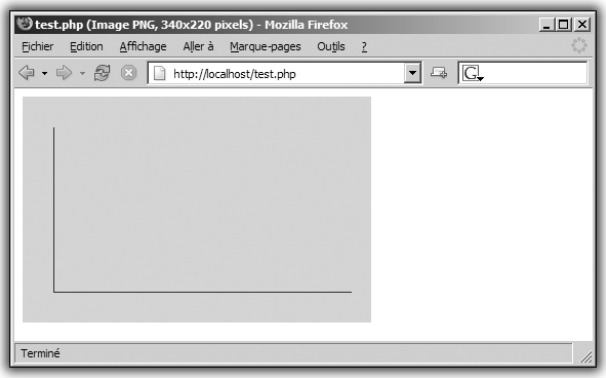

Figure 12.20 : *Génération d'un graphique*

Origine d'une image

L'angle supérieur gauche de l'image a pour coordonnées (0,0). Les coordonnées vont donc de gauche à droite et de haut en bas.

Dessinez les flèches à l'extrémité des axes. Pour cela, vous devez utiliser la fonction `imagefilledpolygon()` qui permet de dessiner un polygone (une forme géométrique à n côtés). Cette fonction prend quatre arguments :

- l'identifiant ;
- un tableau contenant les coordonnées des différents points formant le polygone (x0, y0, x1, y1, etc.) ;
- le nombre de sommets du polygone ;
- la couleur du polygone.

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

imagefilledpolygon ($image, $tab_fleche_ord, 3, $coul_axes);
imagefilledpolygon ($image, $tab_fleche_abs, 3, $coul_axes);
```

Pour vous rendre compte de l'avancée, définissez des affichages intermédiaires. Le code contient pour l'instant les instructions suivantes :

```
<?php

header ("Content-type: image/png");

// création de l'image
$image = @Imagecreate (340, 220);

// définition des couleurs
$fond      = ImageColorAllocate ($image, 208, 216, 213);
$coul_axes = ImageColorAllocate ($image, 11, 62, 43);
$coul_lignes = ImageColorAllocate ($image, 227, 235, 232);
$coul_legendes = ImageColorAllocate ($image, 11, 62, 43);
$coul_barres = ImageColorAllocate ($image, 42, 124, 94);
$coul_orange = ImageColorAllocate ($image, 207, 140, 53);

// les axes
imageline ($image,30,30,30,190,$coul_axes);
imageline ($image,30,190,320,190,$coul_axes);

// les flèches au bout des axes
$tab_fleche_ord = array (30, 30, 26, 34, 34, 34);
```

```
$tab_fleche_abs = array (320, 190, 316, 186, 316, 194);

imagefilledpolygon ($image, $tab_fleche_ord, 3, $coul_axes);
imagefilledpolygon ($image, $tab_fleche_abs, 3, $coul_axes);

// génération de l'image
ImagePng ($image);

?>
```

Passez maintenant à l'affichage des légendes des axes :

```
ImageTTFText
✂ ($image,10,0,5,20,$coul_legendes,"arial.ttf","moyenne");
ImageTTFText ($image,10,0,280,180,$coul_legendes,"arial
✂ .ttf", "trimestre");
```

L'axe des ordonnées dispose d'une graduation de 0 à 20 :

```
imageline ($image,26,190,30,190,$coul_axes);
imageline ($image,26,155,30,155,$coul_axes);
imageline ($image,26,120,30,120,$coul_axes);
imageline ($image,26,85,30,85,$coul_axes);
imageline ($image,26,50,30,50,$coul_axes);
```

Les ordonnées sont indiquées au niveau de la graduation :

```
ImageTTFText ($image,8,0,6,190,$coul_legendes,"arial.ttf", "0");
ImageTTFText ($image,8,0,6,155,$coul_legendes,"arial.ttf", "5");
ImageTTFText ($image,8,0,6,120,$coul_legendes,"arial.ttf", "10");
ImageTTFText ($image,8,0,6,85,$coul_legendes,"arial.ttf", "15");
ImageTTFText ($image,8,0,6,50,$coul_legendes,"arial.ttf", "20");
```

Pour faciliter la lecture des valeurs, affichez une trame horizontale légère :

```
imageline ($image,31,155,320,155,$coul_lignes);
imageline ($image,31,120,320,120,$coul_lignes);
imageline ($image,31,85,320,85,$coul_lignes);
imageline ($image,31,50,320,50,$coul_lignes);
```

Affichez maintenant les barres avec des valeurs fixes.

Moyenne de la classe :

- Premier trimestre : 13,5.
- Deuxième trimestre : 12.
- Troisième trimestre : 16.

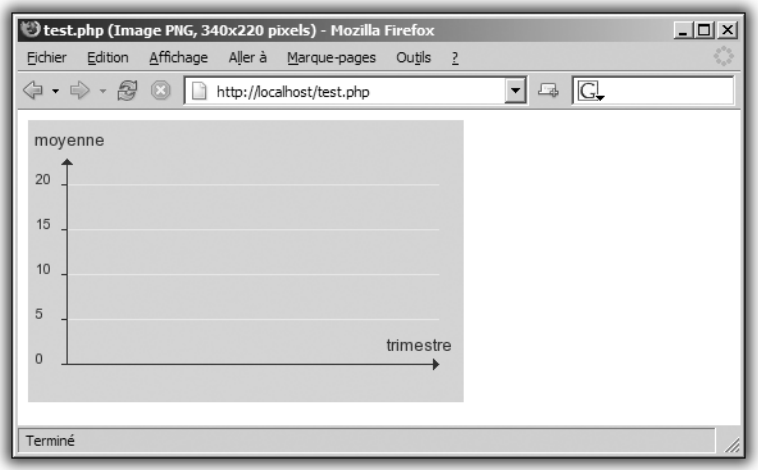

Figure 12.22 : *Ajout de la graduation et des légendes*

La difficulté est de trouver la hauteur des barres. Vous disposez de 140 pixels (190-50) pour afficher des données qui vont au maximum jusqu'à 20. Comme les coordonnées vont de haut en bas, c'est en fait la différence entre la note et la note maximale (20) qui doit vous intéresser. Une simple règle de trois permet ensuite de trouver l'ordonnée du sommet des barres : `(20 – note) * 7 + 50`.

```
imagefilledrectangle ($image, 40, (20-13.5)*7+50, 110,
189, $coul_barres);
imagefilledrectangle ($image, 120, (20-12)*7+50, 190, 189,
$coul_barres);
imagefilledrectangle ($image, 200, (20-16)*7+50, 270, 189,
$coul_barres);
```

Affichez dans les barres la valeur de la moyenne :

```
ImageTTFText ($image,10,0,50,180,$coul_orange,"arial.ttf",
✂ "13.5");
ImageTTFText ($image,10,0,130,180,$coul_orange,"arial.ttf",
✂ "12");
ImageTTFText ($image,10,0,210,180,$coul_orange,"arial.ttf",
✂ "16");
```

Regroupez maintenant toutes les parties au sein d'un script *graph.php* et allez chercher les véritables valeurs dans la table *exam* :

Listing 12-15 : Script graph.php

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

```
// récupération de la moyenne de la classe sur les 3 trimestres
$liendb = mysql_connect($bddserver, $bddlogin, $bddpassword);
mysql_select_db ($bdd);
$sql = "SELECT * FROM $table_exam";
$resultat = mysql_query ($sql);

// le tableau $tab contient 3 cases pour les 3 trimestres
// (de 0 à 2)
// chaque case contient la somme de toutes les moyennes
// des élèves pour le trimestre donné
$i = 0;
while ($tmp = mysql_fetch_array ($resultat))
{
  $tab[$tmp['trimestre'] - 1] += $tmp['moyenne'];
  $i++;
}
mysql_close($liendb);

// la variable $i contient le nombre de notes totales dans
// la table exam elle permet d'obtenir le nombre d'élèves
$nb_eleves = $i / 3;

// utilisez la fonction number_format pour n'avoir que
// 2 chiffres après la virgule
for ($i = 0; $i < 3; $i++)
{
  $tab[$i] = number_format($tab[$i] / $nb_eleves, 2);
}

//-------------------------------------------------------------
header ("Content-type: image/png");

// création de l'image
$image = @Imagecreate (340, 220);

// définition des couleurs
$fond      = ImageColorAllocate ($image, 208, 216, 213);
$coul_axes = ImageColorAllocate ($image, 11, 62, 43);
$coul_lignes = ImageColorAllocate ($image, 227, 235, 232);
$coul_legendes = ImageColorAllocate ($image, 11, 62, 43);
$coul_barres = ImageColorAllocate ($image, 42, 124, 94);
$coul_orange = ImageColorAllocate ($image, 207, 140, 53);

// les axes
imageline ($image,30,30,30,190,$coul_axes);
imageline ($image,30,190,320,190,$coul_axes);

// les flèches au bout des axes
$tab_fleche_ord = array (30, 30, 26, 34, 34, 34);
$tab_fleche_abs = array (320, 190, 316, 186, 316, 194);
imagefilledpolygon ($image, $tab_fleche_ord, 3, $coul_axes);
```

```
imagefilledpolygon ($image, $tab_fleche_abs, 3, $coul_axes);

// légendes
ImageTTFText ($image,10,0,5,20,$coul_legendes,"arial
✂ .ttf","moyenne");
ImageTTFText ($image,10,0,280,180,$coul_legendes,"arial
✂ .ttf","trimestre");

// graduations
imageline ($image,26,190,30,190,$coul_axes);
imageline ($image,26,155,30,155,$coul_axes);
imageline ($image,26,120,30,120,$coul_axes);
imageline ($image,26,85,30,85,$coul_axes);
imageline ($image,26,50,30,50,$coul_axes);

// ordonnées
ImageTTFText ($image,8,0,6,190,$coul_legendes,"arial .ttf","0");
ImageTTFText ($image,8,0,6,155,$coul_legendes,"arial .ttf","5");
ImageTTFText ($image,8,0,6,120,$coul_legendes,"arial .ttf","10");
ImageTTFText ($image,8,0,6,85,$coul_legendes,"arial .ttf","15");
ImageTTFText ($image,8,0,6,50,$coul_legendes,"arial .ttf","20");
// lignes légères
imageline ($image,31,155,320,155,$coul_lignes);
imageline ($image,31,120,320,120,$coul_lignes);
imageline ($image,31,85,320,85,$coul_lignes);
imageline ($image,31,50,320,50,$coul_lignes);

// affichage des barres
imagefilledrectangle ($image, 40, (20 - $tab[0]) * 7 + 50,
✂ 110, 189, $coul_barres);
imagefilledrectangle ($image, 120, (20 - $tab[1]) * 7 +
✂ 50, 190, 189, $coul_barres);
imagefilledrectangle ($image, 200, (20 - $tab[2]) * 7 +
✂ 50, 270, 189, $coul_barres);

// affichage de la valeur de la barre
ImageTTFText ($image,10,0,50,180,$coul_orange,"arial .ttf",$tab[0]);
ImageTTFText ($image,10,0,130,180,$coul_orange,"arial .ttf",$tab[1]);
ImageTTFText ($image,10,0,210,180,$coul_orange,"arial .ttf",$tab[2]);
ImagePng ($image);

?>
```

Ajoutez finalement la ligne suivante en bas du script *admin.php* afin d'avoir un aperçu de la moyenne de la classe :

```
<img src="graph.php" />
```

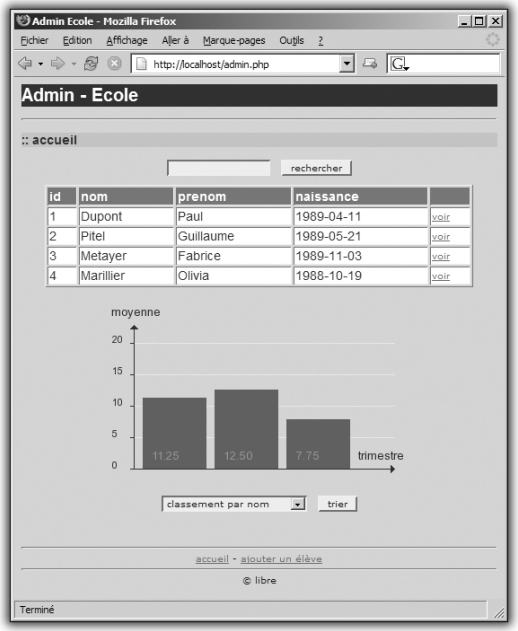

Figure 12.23 : *Un graphique inclus dans la page*

12.3. Check-list

- PHP dispose de toutes les fonctions permettant de travailler sur les fichiers texte : ouverture, modification, suppression, déplacement, etc.
- La technique dite des fichiers de cache est particulièrement intéressante pour alléger les ressources utilisées par un script.
- Des extensions peuvent être chargées afin de permettre à PHP de générer des fichiers plus complexes : PDF, Flash, image, etc.
- PHP, via la librairie GD, permet de travailler très finement sur les images. Cette extension est compatible avec la plupart des formats de fichiers image actuels : GIF, JPEG, PNG, etc.
- La génération d'un fichier spécial nécessite l'utilisation de la fonction `header()`.
- Le format XML permet de stocker des informations et se révèle être le meilleur choix pour les échanges de fichiers structurés.

Chapitre 13

La programmation objet

Les exemples étudiés jusqu'à maintenant étaient développés de façon procédurale. Fonctions et variables étaient mélangées au sein de vos sources, sans réelle cohérence ni logique.

La programmation orientée objet (POO) propose un nouveau paradigme en vous permettant de manipuler des entités disposant chacune de leurs propres informations et traitements. L'applicatif final consiste ensuite à faire fonctionner ces différents objets de façon collaborative et intelligente.

Cette façon d'opérer facilite à la fois

- la modélisation ;
- la maintenance ;
- la relecture ;
- l'évolutivité.

Longtemps décriée, la dimension objet de PHP a gagné ses lettres de noblesse avec l'arrivée de la version 5 du langage. Cette évolution devenait indispensable dans une industrie informatique où tout applicatif d'envergure se doit d'être écrit en objet.

Parmi les évolutions majeures, la version 5 a notamment apporté :

- une distinction claire entre la copie et la duplication d'objets ;
- la gestion des interfaces ;
- l'opérateur `instanceof` ;
- le mot-clé `final` ;
- les constantes de classe ;
- les méthodes abstraites ;
- les méthodes et les attributs statiques ;
- les constructeurs et destructeurs ;
- la syntaxe `$monObjet->bonjour()->monde()` ;
- les exceptions ;
- les itérateurs ;
- la possibilité de définir une fonction `__autoload()` ;
- l'apparition de différents niveaux de visibilité : `public`, `protected`, `private`.

Plus généralement, les concepteurs du langage tendent à rendre la plateforme PHP réellement objet en fournissant la plupart des nouvelles extensions sous forme de classes (ex: PDO, XML, SOAP, etc.).

Cette « objectisation » correspond précisément au défi majeur que PHP devra relever dans les années à venir :

1. Devenir un véritable langage objet où tout élément est lui-même objet (types de données, I/O, etc.) afin d'attirer les développeurs chevronnés et les acteurs majeurs de l'industrie informatique (chasse gardée actuelle de Java et C#).
2. Ne pas perdre cette facilité d'accès inégalable qui lui a permis de devenir le premier langage de programmation web mondial.

13.1. Classes et objets

Deux notions fondamentales interviennent en POO : les classes et les objets.

Classes

Une classe peut être assimilée à un moule qui permet de créer des objets. Cette opération de création est appelée une instanciation.

Déclaration d'une classe

Le mot-clé `class` est utilisé pour définir une classe.

Listing 13-1 : Définition de la classe Rectangle

```
class Rectangle {

}
```

Cette classe doit maintenant être complétée de fonctions et de données qui lui sont propres. En POO, les fonctions sont appelées méthodes et les données, attributs. La classe Rectangle peut disposer par exemple :

- des méthodes : `surface()`, `perimetre()` ;
- des attributs : `longueur`, `largeur` et `couleur`.

Les méthodes `surface()` et `perimetre()` ont la possibilité d'accéder aux attributs de la classe par l'intermédiaire de la variable `$this` qui peut être assimilée à une référence à l'objet lui-même.

Listing 13-2 : Déclaration de la classe Rectangle

```
<?php

class Rectangle {

  public $longueur = null;
  public $largeur = null;
  public $couleur = "rouge";

  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);
    }
  }

}
```

Nom des méthodes

Il est vivement déconseillé de commencer le nom des méthodes par deux caractères __ (caractère espace souligné). PHP utilise en effet cette norme pour nommer ses méthodes, dites « magiques ».

Initialisation des attributs

Un attribut ne peut être initialisé qu'avec une valeur constante. Les initialisations suivantes ne sont par conséquent pas valides :

Listing 13-3 : Initialisation interdites

```
public $date = date();
public $id = "Paul"."Dupont";
```

La fonction `array()` peut cependant être utilisée si tous les éléments du tableau correspondent à des données statiques.

Listing 13-4 : Initialisation autorisée

```
public $tab = array("abc",123);
```

Objets

L'instanciation d'un objet utilise le mot-clé `new`. Un objet `$rectangle` peut être créé avec la syntaxe suivante :

Listing 13-5 : Instanciation d'un objet

```
$rect = new Rectangle();
```

Une fois l'objet créé, vous pouvez accéder à ses attributs et ses méthodes avec le séparateur –>.

Listing 13-6 : Récupération de la valeur d'un attribut et utilisation d'une méthode

```
$rect = new Rectangle();
echo $rect->couleur;

$rect->longueur = 3;
$rect->largeur = 5;
$tmp = $rect->perimetre();
```

PHP autorise également l'utilisation d'une variable pour spécifier un attribut derrière le séparateur –>.

Listing 13-7 : L'attribut est spécifié à l'aide d'une variable

```
$rect = new Rectangle(3,5);
$attr = "longueur";
$rect->$attr = 4;
```

Création dynamique d'attributs

PHP permet de créer des attributs à la volée :

```
$monrect = new Rectangle();

$monrect->toto = 1;
```

L'objet `$monrect` contient désormais l'attribut `$toto`.

La fonction `var_dump()` permet de visualiser les attributs d'une classe.

Listing 13-8 : Utilisation de var_dump()

```
$rect = new Rectangle();
print("<pre>");
var_dump($rect);
```

```
print("</pre>");
```

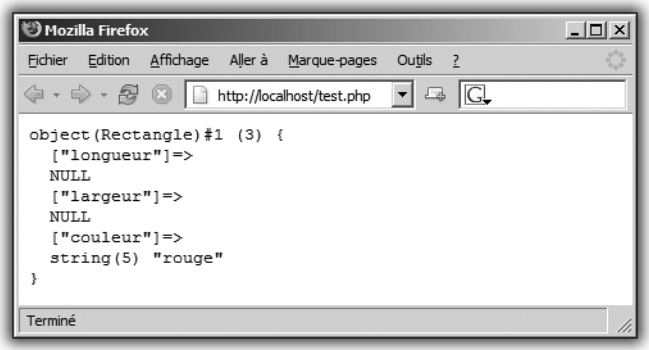

Figure 13.1 : *Contenu de l'objet $rect*

La vérification de la correspondance d'un objet à l'instanciation d'une classe donnée peut être réalisée avec l'opérateur `instanceof`.

Listing 13-9 : Vérification qu'un objet est bien l'instance d'une classe donnée

```
$rect = new Rectangle(3,5);

if ($rect instanceof Rectangle) {
  print("L'objet est un rectangle");
}
```

Variables globales et classes

Le mot-clé `global` et la « super globale » `$GLOBALS` peuvent être utilisés au sein d'une classe pour faire référence aux variables globales du script.

Fonction __autoload()

PHP doit disposer, pour instancier un objet, de la définition de la classe associée. Les sources de la classe peuvent être placées au sein même du script ou dans un fichier extérieur inclus via la fonction `require_once()`. Cette seconde possibilité a l'avantage d'autoriser le partage de la classe avec d'autres scripts de l'applicatif.

L'inclusion d'un fichier de définition de classe est moins fastidieuse depuis PHP5. Si votre script dispose d'une fonction `__autoload()`, PHP y fera appel pour chaque instanciation d'objet de la classe duquel il ne dispose pas.

Listing 13-10 : Définition de la fonction __autoload() qui ira chercher toutes les définitions de classe dans le répertoire class

```
function __autoload($nom_classe) {
  require_once("class/".$nom_classe.".php");
}
```

__autoload()

La fonction `__autoload()` ne doit être définie qu'une fois dans votre script !

Mot-clé static et constantes de classe

Les attributs et les méthodes dont les définitions sont complétées du mot-clé `static` peuvent être utilisés sans nécessiter l'instanciation d'un objet. Le séparateur `::` remplace alors –> pour y accéder.

Listing 13-11 : Utilisation d'un attribut et d'une méthode static

```
class Rectangle {
  public static $nom = 'rectangle';
  static function presentation() {
    print("Bonjour je suis un rectangle");
  }
}

echo Rectangle::$nom;
Rectangle::presentation();
```

Une méthode (`static` ou non) ne peut utiliser la variable `$this` pour accéder à un élément `static` (attribut ou méthode) de la classe. En effet, vous n'êtes pas au niveau objet mais au niveau classe. Le mot-clé `self` est alors utilisé pour faire référence à la classe elle-même.

Listing 13-12 : Utilisation d'un attribut static au sein de la classe

```
class Rectangle {
  public static $nom = 'rectangle';
  static function presentation() {
    print("Bonjour je suis un ".self::$nom);
  }
}
```

Les constantes de classe ne diffèrent des attributs `static` que du point de vue de leur syntaxe : le mot-clé `const` est utilisé pour leur déclaration et aucun `$` ne les précède.

Listing 13-13 : Utilisation d'une constante de classe

```
class Rectangle {
  const NOM = 'rectangle';
  static function presentation() {
    print("Bonjour je suis un ".self::NOM);
  }
}

echo Rectangle::NOM;
```

Comme les attributs, ces constantes ne peuvent être initialisées qu'avec des valeurs statiques.

Conversion

PHP donne la possibilité de convertir un objet en tableau et inversement. Le moyen pour y parvenir consiste à utiliser une opération de *cast*.

Listing 13-14 : Conversion d'un objet en tableau.

```
class Test {
  public $a = 1;
  public $b = 2;
  function hello {}
}
$test = new Test();
$tab = (array) $test;
```

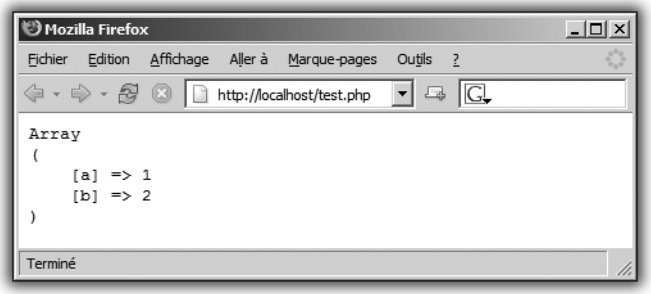

Figure 13.2 : *Tous les attributs sont conservés*

Listing 13-15 : Conversion d'un tableau en objet

```
$tab = array("a"=>1, "b"=>2);
$obj = (object) $tab;
var_dump($obj);
```

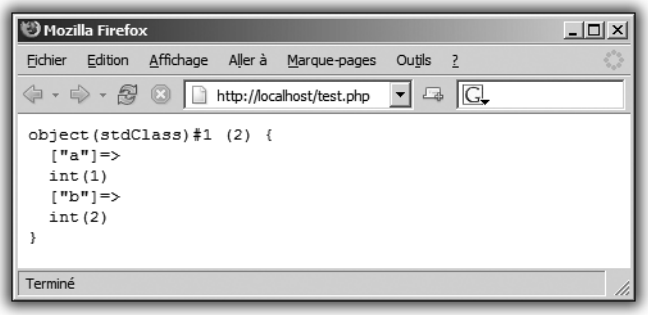

Figure 13.3 : *Conversion du tableau*

Constructeur et destructeur

La classe Rectangle reposant sur deux attributs essentiels (`$longueur` et `$largeur`), la grande majorité des instanciations sera suivie de leur initialisation. Pour éviter cette situation, la POO prévoit l'utilisation d'une fonction spécifique : le constructeur. En PHP, le constructeur correspond à une méthode portant le nom `__contruct()`. Cette fonction est appelée au moment de l'initialisation de l'objet et reçoit les arguments transmis à la classe lors de l'instanciation de l'objet. L'utilisation principale d'un constructeur reste l'initialisation des attributs de la classe.

Listing 13-16 : Initialisation automatique de l'objet grâce au constructeur

```
class Rectangle {

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

  function __construct($longueur,$largeur) {
    $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);
    }
  }

}
$rect = new Rectangle (3,5);
```

PHP prévoit également la possibilité de définir une méthode `__destruct()` qui, si elle existe, sera appelée juste avant la suppression de l'objet. Ce destructeur est extrêmement utile pour les objets utilisant des ressources extérieures telles qu'une base de données, un fichier ou un service web. Au moment de la destruction de l'objet, le destructeur se charge alors de nettoyer son environnement en fermant les ressources ouvertes.

Listing 13-17 : Appels au constructeur et au destructeur

```
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);
    }
  }

  function __destruct() {
    print("Bye bye ".self::NOM."<br/>");
  }
```

```
}

$rect = new Rectangle(3,5);
unset($rect);
```

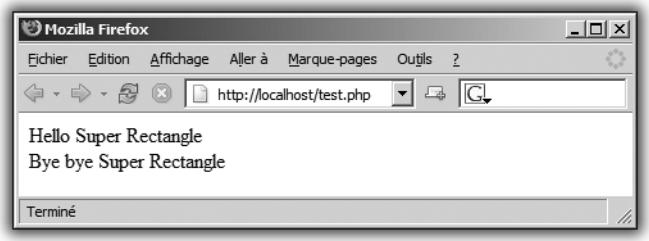

Figure 13.4 : *Les deux fonctions sont bien appelées*

Valeur `null`
Le destructeur est également appelé lorsque la variable `$rect` prend la valeur `null`.

13.2. Les méthodes magiques

Il s'agit de méthodes optionnelles normalisées par PHP et appelées suite à certains événements sur les objets. Les méthodes `__construct()` et `__destruct()` appelées lors de la création et la destruction d'un objet sont de bons exemples de méthodes magiques.

__sleep() et __wakeup()

Les fonctions `serialize()` et `unserialize()` présentées dans le chapitre consacré aux tableaux peuvent également être utilisées sur des objets. Dans un tel cas, PHP appelle, si elles existent dans la classe, les méthodes `__sleep()` avant la sérialisation et `__wakeup()` après la désérialisation.

Cette fonctionnalité est particulièrement utile lorsque votre objet repose sur une connexion ou un fichier extérieur. La fonction `__sleep()` peut alors être utilisée pour fermer les connections et `__wakeup()` pour les rouvrir.

__toString()

Cette méthode est appelée lorsque PHP se retrouve à devoir afficher un objet avec `print()` ou `echo()` (l'objet doit être seul entre parenthèses !). Si cette méthode existe, PHP affichera la valeur de retour de `__toString()` plutôt que le message `Object id #X`. Cette fonctionnalité est particulièrement utile pour debugger vos scripts.

Listing 13-18 : Utilisation de la méthode __toString()

```
class Rectangle {

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

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

  function __toString() {
    return (self::NOM." [".$this->longueur."x".
                          $this->largeur."]");
  }
}

$rect = new Rectangle(3,5);
print($rect);
```

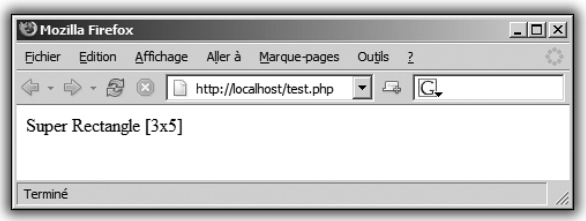

Figure 13.5 : *La fonction print()affiche la valeur retournée par la méthode __toString()*

Surcharge des accesseurs

Lorsque la méthode `__call()` est incluse au sein d'une classe, PHP l'utilise pour chaque appel à une méthode non définie de l'objet. Le

premier argument de `__call()` correspond au nom de la méthode appelée et le second argument, à un tableau des différents arguments de la méthode appelée.

Listing 13-19 : Utilisation de la méthode __call()

```
class Rectangle {

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

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

  function __call($methode,$arguments) {
    return ("Appel à la methode ".$methode."() de
    ✂ ".self::NOM );
  }

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

}

$rect = new Rectangle(3,5);
echo $rect->coucou()."<br/>";
echo $rect->surface();
```

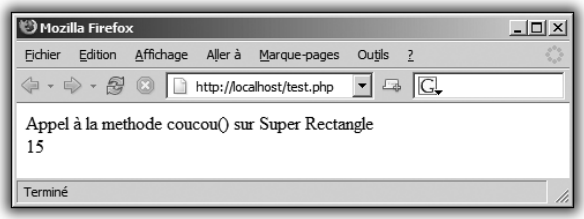

Figure 13.6 : *L'appel de la méthode coucou()conduit à un appel de __call()*

Associée à la fonction `call_user_func()`, la méthode `__call()` peut être utilisée pour tracer puis déléguer de façon générique les appels à des méthodes internes.

Listing 13-20 : Utilisation de __call() pour déléguer les appels de méthodes

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

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

  function __call($methode,$arguments) {
    print ("Méthode [ ".$methode." ]<br/>");
    $tmp = '_'.$methode;
    return call_user_func(array($this,$tmp),$arguments);
  }

  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);
    }
  }

}

$rect = new Rectangle(3,5);
echo $rect->surface();
echo "<br/>";
echo $rect->perimetre();
```

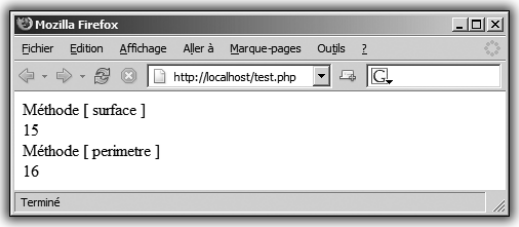

Figure 13.7 : *Chaque appel à une méthode est tracé*

Les méthodes `__set()` et `__get()` sont appelées lorsque vous souhaitez modifier ou récupérer des attributs qui n'existent pas.

L'exemple suivant montre comment utiliser ces deux méthodes pour travailler avec l'attribut virtuel `dim`.

Listing 13-21 : Utilisation des méthodes __get() et __set()

```
class Rectangle {

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

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

  function __set($nom,$val) {
    if ($nom=="dim" && count($val)==2) {
      list ($this->longueur, $this->largeur) = $val;
    }
  }

  function __get($nom) {
    if ($nom=="dim") {
      return array($this->longueur,$this->largeur);
    }
  }

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

}

$rect = new Rectangle(3,5);
$rect->dim = array(2,3);
$dim = join("x",$rect->dim);
echo "Surface d'un rectangle de $dim = ";
echo $rect->surface();
```

Figure 13.8 : *Affichage du résultat*

Sur le même principe, les méthodes `__isset()` et `__unset()` peuvent être définies.

13.3. Polymorphisme

Le polymorphisme est un des principes les plus importants de la programmation objet. L'idée principale consiste à créer un lien d'héritage entre deux classes.

Principe général

En dérivant d'une classe mère, la classe fille hérite de ses méthodes et de ses attributs qui viendront s'ajouter à celles et ceux déjà définis. Encore plus intéressant, la classe fille peut redéfinir les attributs et les méthodes de la classe mère : on dit alors qu'elle les surcharge. En favorisant l'héritage entre vos classes vous augmentez la généricité de vos classes et favorisez leur réutilisation. Tout le code générique est placé dans les classes parentes, et les codes spécifiques dans les classes filles.

Au niveau syntaxique, une classe B hérite de la classe A en utilisant le mot-clé `extends` :

Listing 13-22 : La classe B étend la classe A

```
class B extends A {

}
```

En considérant qu'un carré est un rectangle dont les côtés sont égaux, vous pouvez modéliser cette relation en déclarant une classe Carré qui étend la classe Rectangle.

Le mot-clé `parent` permet d'accéder aux méthodes et attributs de la classe mère. Il est particulièrement utile pour faire « remonter » les paramètres d'initialisation à la classe mère.

Listing 13-23 : La classe Carre hérite de la méthode surface() de la classe Rectangle

```
class Rectangle {

  public $longueur = null;
  public $largeur = null;
  const NOM = "Rectangle";
```

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

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

}

class Carre extends Rectangle {

  public $cote = null;
  const NOM = "Carré";

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

}

$carre = new Carre(5);
echo $carre->surface();
```

Mot-clé `final`

Le mot-clé `final` accolé à la définition d'une classe interdit qu'elle soit étendue. Accolé à la définition d'une méthode, il interdit que cette méthode soit surchargée.

Visibilité

Une notion de visibilité peut être associée à la définition des méthodes et des attributs. La visibilité par défaut correspond à `public` et autorise un accès sans restriction. Les modes `private` et `protected` limitent quant à eux les accès de la façon suivante :

- `private` signifie que seule la classe associée peut accéder à la méthode (ou l'attribut) ;

- `protected` signifie que seules les classes disposant d'un lien d'héritage peuvent accéder à la méthode (ou l'attribut).

La visibilité doit précéder la définition de l'attribut ou de la méthode :

Listing 13-24 : Exemple de définition d'un attribut en mode protected et d'une méthode en mode private

```
protected $str = "";
private function maMethode() {}
```

PHP émet une erreur en cas de non-respect de la règle de visibilité :

Listing 13-25 : L'accès à une méthode protégée déclenche l'affichage d'une erreur

```
class Rectangle {

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

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

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

}

$rect = new Rectangle(3,2);
echo $rect->surface();
```

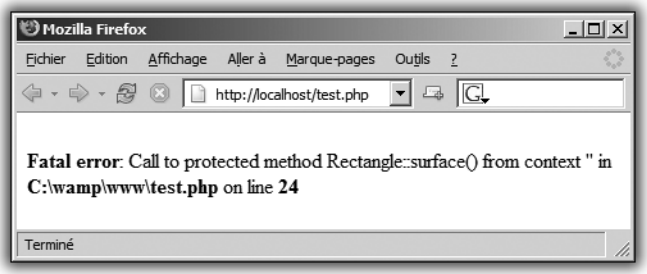

Figure 13.9 : *Erreur correspondant à l'accès à une méthode protégée*