

NOTIONS DE BASE

1

PLAN	1.1 Le langage PHP
	1.2 Le modèle client/serveur
	1.3 Les outils de développement
	1.4 L'installation du paquetage XAMPP
OBJECTIFS	➤ Connaître les caractéristiques du langage.
	➤ Comprendre l'environnement technique lié au développement en PHP.
	➤ Appréhender les outils élémentaires de développement.
	➤ Savoir installer le paquetage XAMPP.

1.1 LE LANGAGE PHP

Bref historique

Le langage PHP a été créé en 1994 par Rasmus Lerdorf. Celui-ci publia son code en 1995 sous le nom de PHP/FI (*Personal Home Page Tools/Form Interpreter*).

Andi Gutmans et Zeev Suraski redéveloppèrent le cœur pour publier la version 3 de PHP en 1998 sous le nom de *PHP:Hypertext Preprocessor* (acronyme récursif). Ils développèrent ensuite un nouveau « moteur » appelé *Zend Engine* (compression de leurs initiales : ZEEv et aNDi) pour la version 4 de PHP.

Les versions récemment publiées sont : 5.4 en 2012, 5.5 en 2013, 5.6 en 2014. La version 7.0 (pas de version 6.0 finalisée) sortie en décembre 2015 est plus rapide et intègre, par exemple, un nouvel opérateur de comparaison et la possibilité de typer le retour de fonctions. Les programmes proposés dans cet ouvrage ont été développés avec la version 5.6.3 de PHP.

Caractéristiques

Le langage de programmation PHP produit des pages web dynamiques et interface l'accès à des Systèmes de gestion de bases de données relationnelles (SGBDR). Il est souvent associé au serveur web Apache et au SGBDR MySQL.

Ses principales caractéristiques sont :

- un langage interprété ;
- un langage hybride procédural/objet ;
- le développement de sites web dynamiques ;
- l'interfaçage avec les bases de données ;
- une importante bibliothèque d'outils.

Langage interprété

Les langages de programmation sont soit compilés, soit interprétés.

Avec les langages *compilés* comme le C ou le C++, un *compilateur* traduit le fichier source contenant le texte du programme en un fichier exécutable, correspondant au programme écrit dans le langage binaire du processeur. Seul le fichier binaire est exécutable. Son exécution est rapide car il contient les instructions dans le langage natif du processeur. Le principal inconvénient est sa non-portabilité. Seul un ordinateur ayant un processeur compatible avec le langage binaire utilisé lors de la compilation peut exécuter ce programme.

Les langages *interprétés* comme PHP, ne nécessitent aucune étape de compilation. Le fichier source est directement « exécuté » *via* un logiciel *interpréteur* qui lit le texte et effectue les traitements décrits. Le programme est indépendant de l'ordinateur et de son processeur, il suffit de disposer d'un interpréteur. Néanmoins, « l'exécution » est plus lente car chaque exécution interprète le texte et indique au processeur les traitements à faire.

L'interpréteur PHP peut être activé en mode commande *via* le shell (invite de commande) ou à partir d'un navigateur *via* un serveur web tel qu'Apache.

● *Au niveau du shell*

Même si ce n'est pas son usage habituel, un programme PHP est exécutable directement au niveau du shell Unix ou Windows, dans une *fenêtre Terminal* comme n'importe quel *langage de script*.

Avec ce mode d'interprétation, le programme ne doit contenir aucune balise HTML, et les instructions d'entrée/sortie (saisie et affichage) doivent saisir en mode ligne et afficher sur un terminal texte avec gestion des sauts de ligne (' \n '). Aucune présentation « graphique » n'est possible. C'est le mode privilégié des aides proposées sur Internet pour le langage PHP. Les programmes donnés en exemple se focalisent sur les seules syntaxes PHP en dehors de tout contexte web.

La figure 1.1 présente l'interprétation du programme `hello.php` dans une fenêtre Terminal Unix.

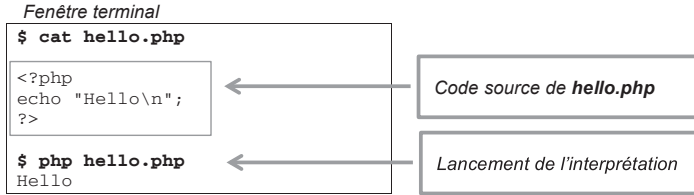


Figure 1.1 – Interprétation d'un programme PHP en shell Unix.

- *Par le serveur web*

Ce mode d'interprétation est le plus classique. Il est présenté à la figure 1.2. Le client web (navigateur) interroge un serveur web (par exemple Apache) qui se charge d'interpréter le code source PHP (stocké sur le serveur) *via* son interpréteur.

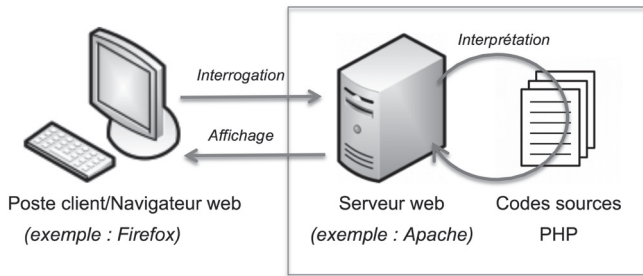


Figure 1.2 – Processus d'interprétation d'un programme PHP via le navigateur et le serveur web.

Le programme source est localisé dans un répertoire de la hiérarchie du serveur web. L'URL (*Uniform Ressource Locator*) commençant par `http://` indiqué au navigateur donne accès au programme. La figure 1.3 présente l'interprétation du programme `hello.php` *via* le navigateur.

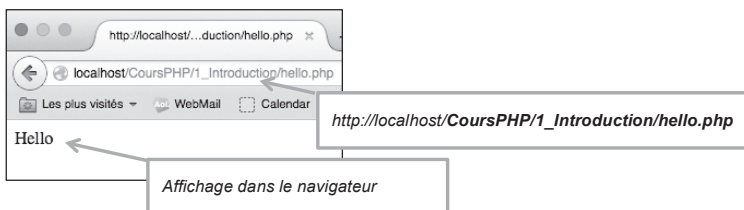


Figure 1.3 – Résultat de l'interprétation d'un programme PHP via le navigateur et le serveur web.

Remarque

Par la suite nous parlerons d'exécution du programme PHP pour indiquer son déroulement à la place d'interprétation afin de simplifier la compréhension du lecteur.

Langage hybride procédural/objet

Tout comme le langage C++, le langage PHP peut contenir des syntaxes procédurales (comme en C), ou Objet (comme en Java).

● *Forme procédurale*

Dans cette forme de programmation, *les données sont séparées des traitements*. La gestion des données consiste à les déclarer et à leur affecter des valeurs. Elles sont ensuite utilisées dans des traitements pouvant être regroupés dans des procédures ou des fonctions. Les données contiennent les informations sur lesquelles portent les traitements.

La figure 1.4 présente le programme `somme_entiers_procedural.php` et fait apparaître sa structure procédurale.

Les données sont `$resultat` et `$i`.

Les traitements correspondent à la boucle `for`, à l’affichage *via* l’instruction `echo` et à la suppression des variables *via* `unset`.

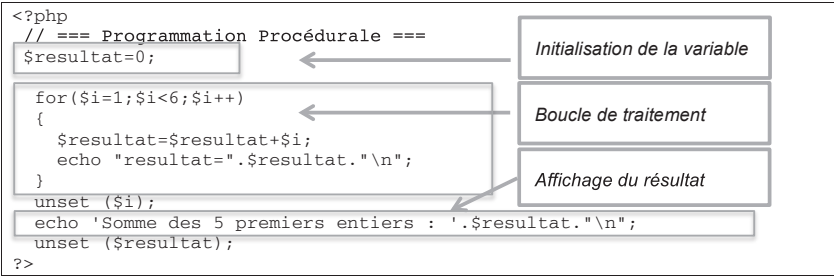


Figure 1.4 - Exemple de syntaxe procédurale.

Voici son exécution en mode shell (listing 1.1).

Listing 1.1 – Exécution de `somme_entiers_procedural.php`

```
$ php somme_entiers_procedural.php
resultat=1
resultat=3
resultat=6
resultat=10
resultat=15
Somme des 5 premiers entiers : 15
```

La figure 1.5 présente l’exécution de ce programme *via* un navigateur.

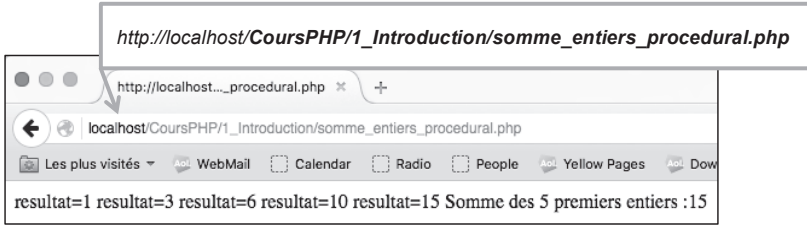


Figure 1.5 – Interprétation de somme_entiers_procedural.php via le navigateur et le serveur web.

Remarque

Aucun saut de ligne n'apparaît entre les différents affichages car la balise HTML `
` n'est pas affichée par le programme PHP et le saut de ligne «`\n`» est ignoré par le navigateur.

• Programmation Orientée Objet (POO)

En Programmation Orientée Objet (POO), un *objet* est un ensemble « autonome » contenant des données, les *propriétés*, et des traitements appelés *méthodes*.

Chaque objet est créé dynamiquement par l'instruction *new* à partir de la *classe* (structure de l'objet) dont il est l'*instance*.

La figure 1.6 présente le programme `somme_entiers_objet.php`, réécriture du programme précédent en syntaxe objet.

Voici son exécution en mode shell.

Listing 1.2 – Exécution de `somme_entiers_objet.php`

```
$ php somme_entiers_objet.php
--- Appel du constructeur ---
Variable interne resultat initialisée à : 0
--- Appel de la méthode Somme ---
resultat=1
resultat=3
resultat=6
resultat=10
resultat=15
--- Appel de la méthode Affichage ---
Somme des 5 premiers entiers : 15
--- Appel du destructeur ---
Variable interne resultat supprimée
```

La figure 1.7 présente l'exécution de ce programme *via* un navigateur.

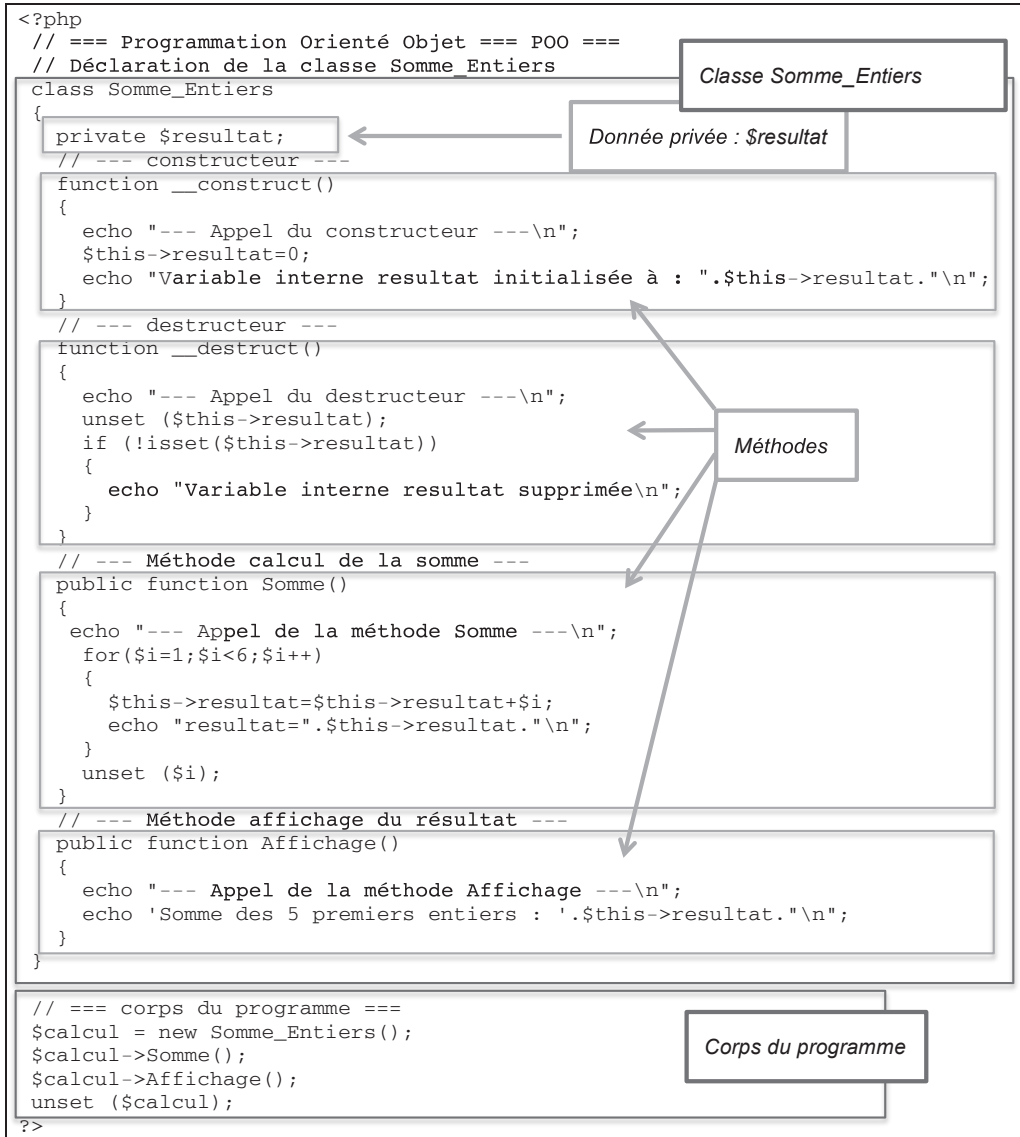


Figure 1.6 – Exemple de syntaxe objet.

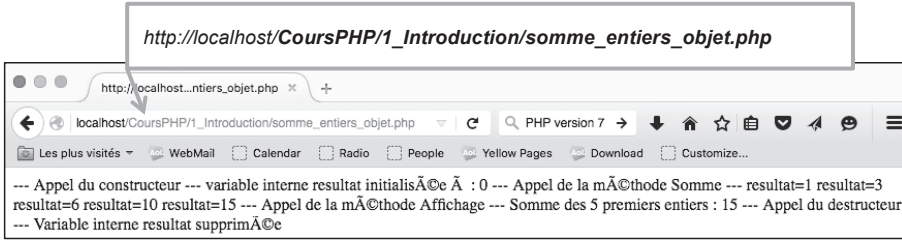


Figure 1.7 – Interprétation de somme_entiers_objet.php via le navigateur et le serveur web.

Développement de sites web dynamiques

Contrairement aux sites « statiques » qui ne contiennent que des pages HTML et qui n'effectuent aucun traitement, les sites dynamiques construits avec PHP contiennent des pages PHP (extension .php) permettant l'exécution de programmes.

Le programme PHP « dynamise » le site web, en effectuant des traitements sur des données saisies par l'utilisateur, par exemple *via* un formulaire HTML, ou en exécutant des requêtes sur des bases de données. L'exemple suivant présente cet aspect dynamique. Un premier fichier `formulaire.html` saisit le nom, le prénom et l'âge d'une personne. Le bouton « valider » transmet les données au programme PHP `formulaire.php` *via* la méthode POST.

Listing 1.3 – Fichier `formulaire.html`

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8" />
    <title>Saisie de valeurs via un formulaire</title>
  </head>
  <body>
    <div style="text-align:center; width:250px; border:solid 1px black;
padding:5px;">
      Saisie d'information
    </div>
    <br/>
    <form action="formulaire.php" method="post">
      Nom:<input type="text" name="nom" size="20"/><br/>
      Prénom:<input type="text" name="prenom" size="30"/><br/>
      Age:<input type="text" name="age" size="10"/><br/><br/>
      <input type="submit" value="Valider" />
      <input type="reset" value="Effacer le formulaire" />
    </form>
  </body>
</html>
```

La figure 1.8 présente l'affichage de ce formulaire dans le navigateur.

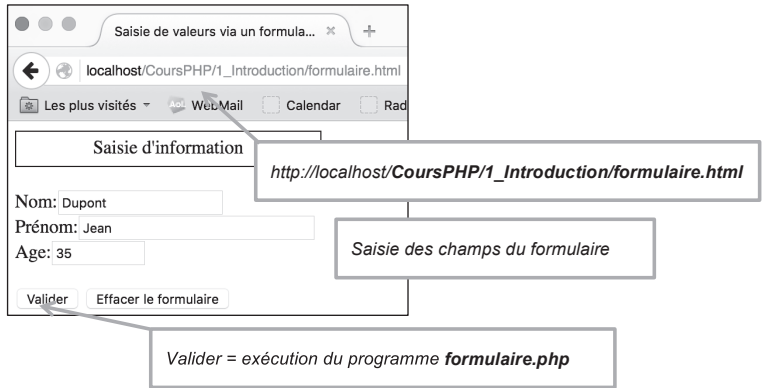


Figure 1.8 – Formulaire de saisie activant un programme PHP.

La figure 1.9 présente le programme `formulaire.php` et les différentes parties qui récupèrent, traitent et affichent les données envoyées par le formulaire HTML.

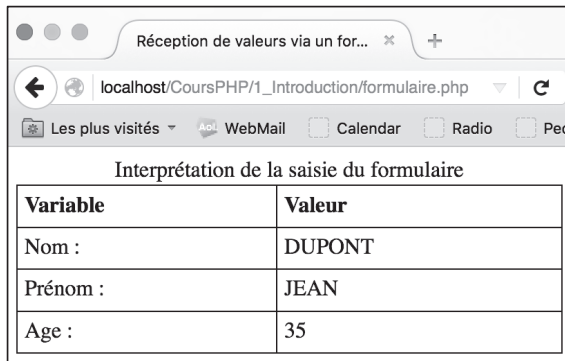


Figure 1.9 – Programme PHP traitant les données envoyées par le formulaire.

Le code PHP affiche dans un tableau HTML les données après leur traitement :

- le début du fichier est constitué de code HTML, définissant l'entête de la page, la feuille de styles et la ligne d'en-tête de la table ;
- la fin du fichier est constituée de code HTML, contenant les balises de fin de structure de la table, du corps (*body*) et de la page HTML ;
- le milieu du fichier contient du code PHP. Les traitements sont :
 - ◇ récupération des données ;
 - ◇ conversion du nom et du prénom en majuscules, et de l'âge en entier ;
 - ◇ affichage des données, encadrées par des balises HTML, *via* « *echo* ».

La figure 1.10 présente le résultat du traitement.



Interprétation de la saisie du formulaire	
Variable	Valeur
Nom :	DUPONT
Prénom :	JEAN
Age :	35

Figure 1.10 - Interprétation de la saisie du formulaire.

Remarque

Le code source de la page affichée (*via* le menu contextuel du navigateur) ne présente aucun code PHP : **le programme PHP est interprété par le serveur Apache, qui envoie uniquement du code HTML au navigateur**, comme cela est présenté sur la figure 1.11.

```
<!DOCTYPE html>
<html>
  <head> <!-- Entête HTML -->
    <meta charset="utf-8" />
    <title>Réception de valeurs via un formulaire</title>
    <!-- Feuille de style pour le tableau -->
    <style>
      table, th, td {
        border: 1px solid black;
        border-collapse: collapse;
      }
      th, td {
        padding: 5px;
        text-align: left;
      }
    </style>
  </head>
  <body>
    <table style="width:50%"> <!-- Début du tableau -->
      <caption>Interprétation de la saisie du formulaire</caption>
      <thead> <!-- En-tête du tableau -->
        <tr>
          <th>Variable</th>
          <th>Valeur</th>
        </tr>
      </thead>
      <tr>
        <!-- Début du programme PHP -->
        <tr><td>Nom : </td><td>DUPONT</td></tr><tr><td>Prénom :
</td><td>JEAN</td></tr><tr><td>Age : </td><td>35</td></tr>
        <!-- Fin du programme PHP -->
      </tr>
    </table>
  </body>
</html>
```

Syntaxe HTML provenant de
l'exécution du programme PHP

Figure 1.11 - Code source HTML résultant de l'exécution du programme PHP.

Interface avec les bases de données

La figure 1.12 présente l'architecture de l'interface de PHP avec MySQL :

- le serveur Apache interprète les pages HTML et envoie le code PHP à l'interpréteur PHP (1- Interprétation PHP) ;
- PHP traite le programme et envoie les requêtes à MySQL (2- Requêtes SQL) ;
- MySQL retourne les informations à PHP (3- Réponses MySQL) ;
- PHP retourne les informations au serveur Apache (4- Retour PHP) pour affichage dans la fenêtre du navigateur du poste client.

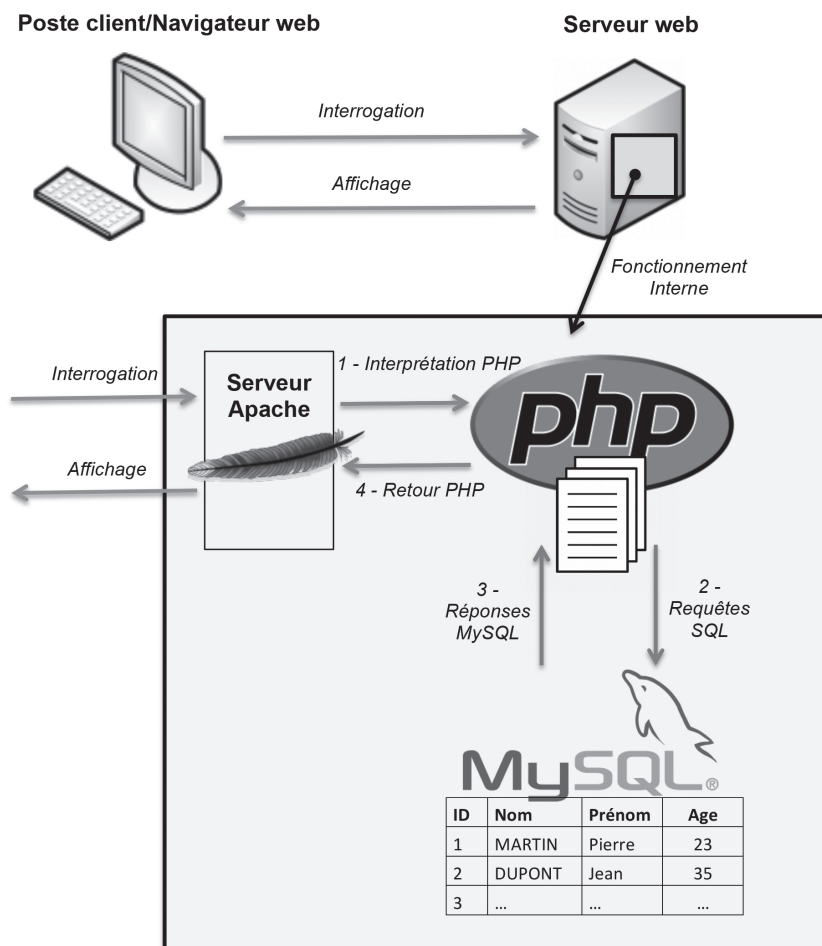


Figure 1.12 – Architecture de l'interface avec les bases de données.

Voici un exemple de ce fonctionnement. Via son navigateur et un formulaire HTML, l'utilisateur saisit le nom d'une personne à rechercher dans une base de données, le programme PHP récupère la saisie, effectue la recherche et affiche les informations qui sont contenues dans une table de la base de données.

Le fichier HTML `formulaire_SQL.html` saisit le nom de la personne à rechercher.

À la validation, le nom est transmis au programme `formulaire_SQL.php` via la méthode POST.

Listing 1.4 – Fichier formulaire_SQL.html

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8" />
    <title>Un exemple de saisie de valeurs via un formulaire</title>
  </head>
  <body>
    <div style="text-align:center; width:250px; border:solid 1px black;
padding:5px;">
      Recherche d'une personne
    </div>
    <br/>
    <form action="formulaire_SQL.php" method="post">
      Nom : <input type="text" name="nom" size="20" /><br/>
      <input type="submit" value="Valider" />
      <input type="reset" value="Effacer le formulaire" />
    </form>
  </body>
</html>
```

La figure 1.13 présente l’affichage de ce formulaire dans le navigateur :

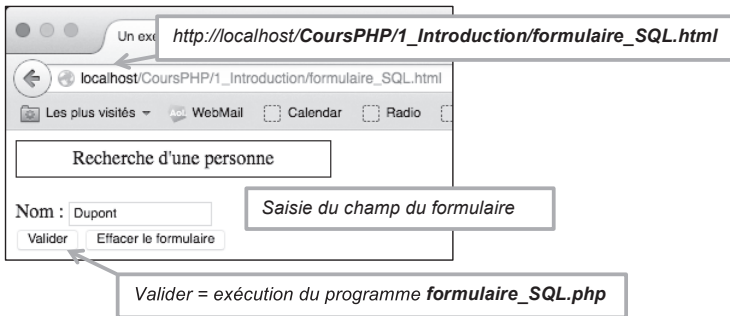


Figure 1.13 – Formulaire de saisie du nom d'une personne.

La figure 1.14 présente le programme PHP `formulaire_SQL.php`. Il possède une structure identique à `formulaire.php` (figure 1.9). Le début et la fin du fichier sont constitués de code HTML. Le milieu contient des syntaxes PHP.

Ce programme envoie une requête `SELECT` sur la table `elevés` d'une base de données MySQL nommée `CoursPHP`. Une boucle `while` récupère les données retournées dans la variable tableau `$données`, et les présente en tableau HTML.



Le mot de passe indiqué dans l'ouverture de la base (`new PDO()`) est volontairement remplacé par `'*****'`.

```

<!DOCTYPE html>
<html>
  <head> <!-- Entête HTML -->
    <meta charset="utf-8" />
    <title>Un exemple d'interrogation d'une base MySQL</title>
    <!-- Feuille de style pour le tableau -->
    <style>
      table, th, td {
        border: 1px solid black;
        border-collapse: collapse;
      }
      th, td {
        padding: 5px;
        text-align: left;
      }
    </style>
  </head>
  <body>
    <table style="width:50%"> <!-- Début du tableau -->
      <caption>Retour de la requête SQL</caption>
      <thead> <!-- En-tête du tableau -->
        <tr>
          <th>ID</th>
          <th>Nom</th>
          <th>Prénom</th>
          <th>Age</th>
        </tr>
      </thead>
      <tr>
        <!-- Début du programme PHP -->
        <?php
          // --- on récupère la donnée ---
          $nom=$_POST['nom'];
          // --- on traite la donnée ---
          $nom=strtoupper($nom);
          // --- Requêtes SQL ---
          $bdd = new PDO('mysql:host=localhost;dbname=CoursPHP','root','*****');
          $reponse = $bdd->query('SELECT ID,nom,prenom,age FROM eleves WHERE
nom='.$nom.'\'');
          // --- Mise en forme dans un tableau du retour de la requête SQL ---
          while ($donnees = $reponse->fetch())
          {
            echo '<tr>';
            echo '<td>'.$donnees['ID'].'</td>';
            echo '<td>'.$donnees['nom'].'</td>';
            echo '<td>'.$donnees['prenom'].'</td>';
            echo '<td>'.$donnees['age'].'</td>';
            echo '<tr>';
          }
          $reponse->closeCursor();
        ?>
        <!-- Fin du programme PHP -->
      </table>
    </body>
  </html>

```

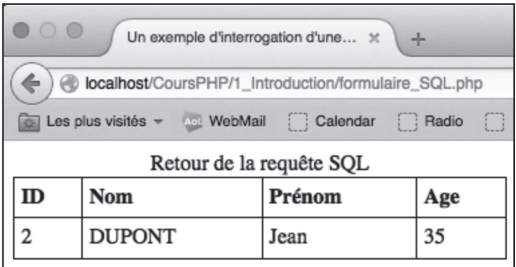
Syntaxe HTML

Syntaxe PHP

Syntaxe HTML

Figure 1.14 – Programme PHP interrogeant une base de données.

Les figures 1.15 et 1.16 présentent respectivement le résultat de l'exécution du programme PHP, et le contenu de la table `eleves` sur laquelle porte la requête.



ID	Nom	Prénom	Age
2	DUPONT	Jean	35

Figure 1.15 - Résultat de la requête SELECT.

ID	nom	prenom	age
1	MARTIN	Pierre	23
2	DUPONT	Jean	35
3	DURAND	Jacques	25

Figure 1.16 - Table élèves de la base de données coursPHP.

Une importante bibliothèque de fonctions

Le nombre très important de fonctions prédéfinies est l’un des atouts du langage PHP. Il suffit généralement de chercher dans cette bibliothèque pour trouver une fonction qui effectue les traitements souhaités. Une liste par catégorie est disponible à l’URL :

<http://fr.php.net/manual/fr/funcref.php>

On y trouve des fonctions classiques comme :

- les fonctions de traitement de chaînes de caractères ;
- les fonctions mathématiques ;
- les fonctions sur les fichiers.

Mais également des fonctions avancées comme :

- les fonctions de requêtes sur des bases de données ;
- les fonctions de cryptage ;
- les fonctions d’envoi de courriel ;
- les fonctions de gestion de la date et de l’heure ;
- les fonctions de traitement d’images ;
- les fonctions de tri ou de recherche.

1.2 LE MODÈLE CLIENT/SERVEUR

Le langage PHP fonctionne sur le modèle client/serveur comme cela est présenté à la figure 1.12. Dans ce modèle, un *client*, le navigateur du poste de travail, interroge

un *serveur* distant, le serveur web. Le langage PHP est « activé » par le serveur web, Apache le plus souvent, qui lui envoie les lignes de code à interpréter. PHP retourne le résultat au serveur web. Le client web (navigateur) ne communique qu'avec le serveur web.

Dans le cas de l'utilisation d'une base de données, celle-ci peut se trouver sur un autre ordinateur que celui qui héberge le serveur web. Ce n'est pas le cas de PHP qui est sur le même ordinateur que le serveur web.

1.3 LES OUTILS DE DÉVELOPPEMENT

Il existe de nombreux environnements intégrés de développement, appelés *Framework*, tels que *Zend Framework*. Afin de privilégier l'apprentissage de PHP indépendamment d'un environnement particulier, nous privilégions l'utilisation d'outils de base pour l'écriture du code PHP. Voici la liste des outils « de base » nécessaires à la programmation PHP dans un environnement web :

- un serveur web, par exemple *Apache* ;
- un module PHP accessible *via* le serveur web ;
- un SGBDR (Système de gestion de base de données relationnel), par exemple *MySQL* ;
- un client : un navigateur, par exemple *FireFox* ;
- un éditeur de texte : *notepad.exe*, *notepad++.exe* (Windows), *gedit* (Linux), *TextWrangler* (Mac OS X).

Les paquetages *AMP* (pour Apache, MySQL, PHP) comme *XAMPP* proposent l'installation groupée des serveurs et des modules.

1.4 L'INSTALLATION DU PAQUETAGE XAMPP

Cette section présente de manière synthétique l'installation du paquetage XAMPP. Une documentation complète, détaillant pas à pas les étapes d'installation, est proposée au téléchargement : *Installation_XAMPP.pdf*.

Le package XAMPP

Présentation

De nombreux paquetages (*Packages*) proposent d'installer Apache, MySQL et PHP. Les acronymes de ces paquetages reprennent les initiales des trois produits : **AMP**, soit **A** (Apache), **M** (MySQL), **P** (PHP).

Selon les environnements systèmes on trouve : WAMP (Windows AMP), LAMP (Linux AMP), MAMP (Mac OS X AMP). Mais il en existe un qui est disponible sur

les trois plateformes systèmes : **XAMPP** pour **X** (Cross), **A** (Apache), **M** (MySQL), **P** (PHP), **P** (Pearl).

Afin de faciliter l'apprentissage de PHP indépendamment du système d'exploitation utilisé, nous avons privilégié l'usage de ce paquetage. Nous en présentons l'installation dans ces trois environnements.

Contenu du paquetage

La version utilisée dans cet ouvrage est **XAMPP 5.6.3**. Ce paquetage contient essentiellement : Apache 2.4.10, MySQL 5.6.21, PHP 5.6.3, phpMyAdmin 4.2.1.

Fichier à télécharger

Le fichier d'installation peut être téléchargé sur le site <http://sourceforge.net/projects/xampp/files/>. Selon le système d'exploitation, il se nomme :

- Windows version 32 bits : `xampp-win32-5.6.3-0-VC11-installer.exe` ;
- Linux (version 64 bits) : `xampp-linux-x64-5.6.3-0-installer.run` ;
- Mac OS X (version 64 bits) : `xampp-osx-5.6.3-0-installer.dmg`.

Les étapes d'installation

Il n'est pas possible de détailler les étapes de l'installation dans cet ouvrage. Celles-ci sont décrites dans le fichier à télécharger : `Installation_XAMPP.pdf`.

Le répertoire de travail

Le répertoire « DocumentRoot »

Le serveur Apache regroupe l'ensemble des fichiers et des répertoires accessibles sur le web dans un répertoire racine appelé *DocumentRoot*. C'est le point de départ de la hiérarchie du site web. Il est défini dans le fichier `httpd.conf`.

Selon le système d'exploitation, et dans le cas d'une installation de XAMPP, ce fichier de configuration se trouve dans le répertoire :

- `c:\xampp\apache\conf\` pour Windows ;
- `/opt/lampp/etc/` pour Linux ;
- `/Applications/XAMPP/xamppfiles/etc/` pour Mac OS X.

Avec XAMPP, le répertoire *DocumentRoot*, racine des documents du site web dans lequel doivent se trouver tous les fichiers sources HTML et PHP est :

- `c:\xampp\htdocs\` pour Windows ;
- `/opt/lampp/htdocs/` pour Linux ;
- `/Applications/XAMPP/xamppfiles/htdocs/` pour Mac O X.

Ce répertoire contient généralement un fichier `index.html` (ou `index.php`). Cela signifie que, si on entre l'URL : `http://localhost/index.html` dans le navigateur, on affiche le texte « It works ! » qui est le contenu du fichier :

- `c:\xampp\htdocs\index.html` pour Windows ;
- `/opt/lampp/htdocs/index.html` pour Linux ;
- `/Applications/XAMPP/xamppfiles/htdocs/index.html` pour Mac OS X.

Pour vérifier si l'installation est complète et que les serveurs sont démarrés, il suffit d'entrer l'URL précédente et de constater l'affichage du message indiqué.

Le répertoire utilisé dans cet ouvrage : CoursPHP

Afin d'organiser les différents fichiers PHP et HTML proposés dans cet ouvrage, il est proposé de créer le répertoire CoursPHP (attention à la casse). Voici deux solutions différentes, la seconde est à privilégier.

- *Comme sous-répertoire directement dans « DocumentRoot »*

La première solution crée CoursPHP directement dans le répertoire correspondant à *DocumentRoot*. Par la suite tous les fichiers et répertoires devront être créés à l'intérieur du répertoire :

- `c:\xampp\htdocs\CoursPHP\` pour Windows ;
- `/opt/lampp/htdocs/CoursPHP/` pour Linux ;
- `/Applications/XAMPP/xamppfiles/htdocs/CoursPHP/` pour Mac OS X.

Cette solution est simple, mais elle pose le problème des droits d'accès au répertoire `htdocs` d'Apache. La mise en œuvre de cette solution est décrite dans la documentation détaillée *Installation_XAMPP.pdf*.

- *Comme un lien dans « DocumentRoot » vers un répertoire personnel*

La seconde solution crée CoursPHP dans *votre espace personnel*, comme sous-répertoire d'un répertoire `www`, soit :

- `c:\Users\login\Documents\www\CoursPHP\` pour Windows ;
- `/home/login/www/CoursPHP/` pour Linux ;
- `/Users/login/www/CoursPHP/` pour Mac OS X.

où `login` est votre identifiant de connexion (par exemple : `dupont`, `lery`...).

Il faut ensuite créer un lien *symbolique* vers ce répertoire dans `htdocs` d'Apache, pour le rendre *visible* sous la hiérarchie Apache.

Cette solution résout le problème des droits d'accès à CoursPHP puisqu'il est dans votre espace personnel, et non directement dans la hiérarchie d'Apache, en le rendant visible *via* le navigateur, comme s'il était dans la hiérarchie Apache.

Par contre, cela nécessite pour Windows de modifier une ligne du fichier de configuration d'Apache, `httpd.conf`, qui n'autorise pas, par défaut, le parcours des

liens vers d'autres répertoires. Sous Linux et Mac OS X, le fichier de configuration Apache l'autorise, aucune modification n'est nécessaire.

Cette solution est détaillée dans la documentation `Installation_XAMPP.pdf`.



Cette solution permet à l'utilisateur de disposer « chez lui » d'un répertoire dans lequel créer les fichiers et les répertoires de son site web. Cependant, cette solution ouvre un trou de sécurité et ne doit jamais être utilisée pour un site web en production. Dans ce cas, les fichiers doivent impérativement se trouver dans le répertoire *htdocs* du serveur Apache et le parcours des liens symboliques ne doit pas être autorisé.

L'accès à CoursPHP

Par la suite, tous les fichiers et les répertoires devront être créés à l'intérieur des répertoires :

- Pour la solution 1
 - ✧ `c:\xampp\htdocs\CoursPHP\` pour Windows ;
 - ✧ `/opt/lampp/htdocs/CoursPHP/` pour Linux ;
 - ✧ `/Applications/XAMPP/xamppfiles/htdocs/CoursPHP/` pour Mac OS X.
- Pour la solution 2
 - ✧ `c:\Users\login\Documents\www\CoursPHP\` pour Windows ;
 - ✧ `/home/login/www/CoursPHP/` pour Linux ;
 - ✧ `/Users/login/www/CoursPHP/` pour Mac OS X.

L'URL `http://localhost/CoursPHP/` montre le contenu de CoursPHP (figure 1.17).

L'URL `http://localhost/CoursPHP/1_Introduction` montre le contenu du sous-répertoire `1_Introduction`, avec les fichiers présentés dans ce chapitre (figure 1.18).

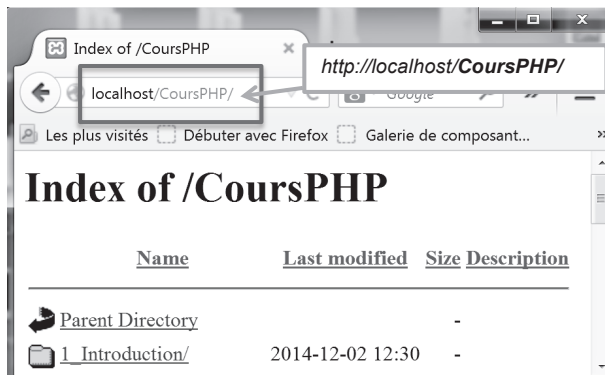


Figure 1.17 - Accès au répertoire CoursPHP.

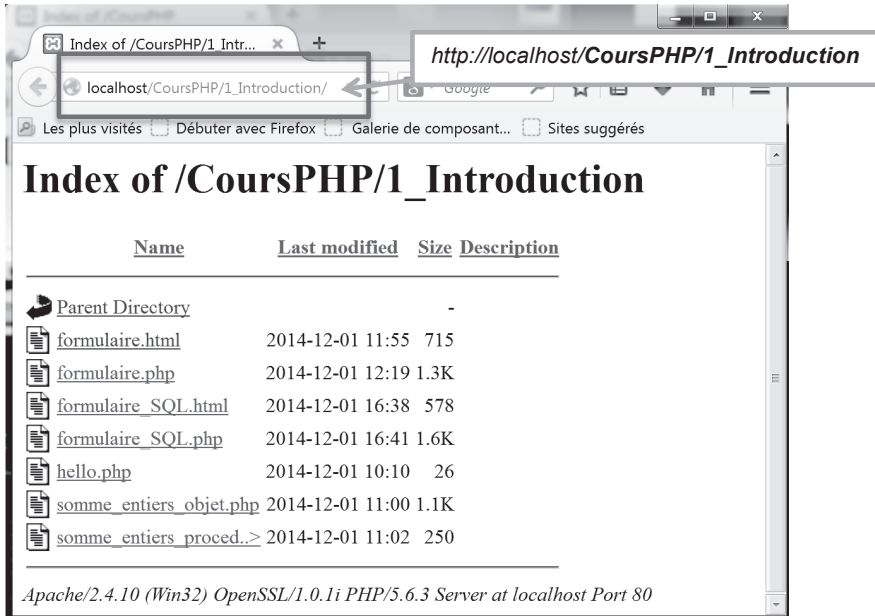
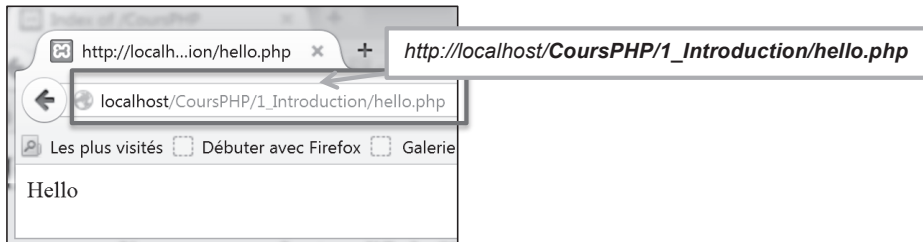


Figure 1.18 – Accès au répertoire CoursPHP/1_Introduction.

Un clic sur l'un des fichiers, par exemple `hello.php`, lance son interprétation (figure 1.19).

Figure 1.19 – Exécution de `hello.php`.

Le programme PHP `php_installation_versions.php` confirme l'installation en affichant les versions des serveurs installés.

Listing 1.5 – Programme `php_installation_versions.php`

```
<?php
echo "phpversion          = ".phpversion() . "<br/>\n";
echo "zend_version       = ".zend_version() . "<br/>\n";
echo "php_underscore      = ".php_underscore() . "<br/>\n";
echo "mysql_get_server_info = ".mysql_get_server_info() . "<br/>\n";
echo "apache_get_version   = ".apache_get_version() . "<br/>\n";
phpinfo();
?>
```

Accès à PHP en mode shell

La variable PATH

Une fois les étapes précédentes effectuées, la modification de la variable d'environnement **PATH** donne accès à l'interpréteur PHP en mode commande dans une fenêtre « Terminal ». La mise en œuvre de cette solution est décrite dans la documentation `Installation_XAMPP.pdf`.

Utilisation de la commande PHP

Dans une fenêtre Terminal, si on saisit la syntaxe « `php hello.php` » (en étant dans le bon répertoire, celui où se trouve `CoursPHP`), le programme `hello.php` est interprété par PHP et affiche le texte « Hello ».