La société Zend propose aussi un produit non open source mais gratuit : Zend Optimiser. Il s'agit d'une librairie qui permet d'optimiser à la volée votre code au moment où celui-ci est exécuté.

Argent et open source

Les dollars et l'open source n'ont jamais fait bon ménage. La réussite de cette société est donc d'autant plus remarquable. En vendant des solutions à des sociétés (qui en ont souvent largement les moyens), Zend est en mesure d'employer des développeurs qui travaillent 100 % de leur temps sur PHP. Toute la communauté PHP profite ainsi directement du succès et de la bonne santé financière de la société Zend.

Le rouleau compresseur PHP est désormais en marche et nul ne sait où il s'arrêtera. Des conventions et des conférences ont lieu désormais tous les mois autour de ce langage ; des milliers de sites, des centaines d'ouvrages, et même des magazines vendus en kiosques se consacrent maintenant exclusivement à PHP. Même le gouvernement français s'intéresse au phénomène et donne des instructions auprès de ses ministères afin que leurs sites soient conçus sur des solutions Lamp (par exemple SPIP Agora). Le Web est devenu central, aussi bien dans notre vie de tous les jours que dans la vie des entreprises, et PHP est en passe de devenir l'un des vecteurs déterminants de son expansion.

Technologies Lamp

Toute personne s'intéressant au PHP a dû croiser dans la littérature le terme *Lamp*. Ce sigle signifie Linux/Apache/MySQL/PHP. Lamp est un environnement complet permettant de faire fonctionner une application web. Il dispose d'un système d'exploitation (Linux), d'un serveur web (Apache), d'un système de gestion de bases de données (MySQL) et d'un langage de programmation (PHP). Il s'agit aujourd'hui, et de loin, de l'environnement le plus performant, le plus sûr et le plus abordable du marché.

Finissons ce paragraphe avec deux chiffres qui devraient marquer les esprits : plus de 1 million de serveurs web et presque 20 millions de sites exploitent aujourd'hui PHP.

1.3. Internet, comment ça marche ?

Dans cette partie, nous allons essayer de comprendre comment fonctionnent le Web et Internet en général. Une bonne compréhension de cette couche réseau vous permettra de mieux envisager le fonctionnement et l'importance de PHP.

Web et autres protocoles

Le Web est un réseau mondial de machines parlant la même langue. En informatique, cette langue est appelée « un protocole ». Le protocole du Web est l'HTTP (Hypertext Transfer Protocol).

Le Web n'est qu'un réseau parmi tant d'autres, HTTP a en effet de nombreux « cousins ».

Tableau 1.2 : *Quelques protocoles de haut niveau et leur fonction*

Protocole	Signification	Fonction
FTP	File Transfer Protocol	Transfert de fichiers
IRC	Internet Relay Chat	Dialogue en direct
NNTP	Network News Transfer Protocol	Envoi, lecture de news
POP	Post Office Protocol	Récupération des courriels
SMTP	Simple Mail Transfer Protocol	Envoi des courriels

Chacun a donc un rôle qui lui est propre et de nouveaux réseaux se créent tous les jours pour répondre aux nouveaux besoins des internautes. Nous avons ainsi vu depuis quelques années l'émergence des réseaux P2P de type BitTorrent qui permettent la recherche et l'échange de fichiers.

Créer un nouveau protocole ne consiste en fait qu'à définir une nouvelle langue compréhensible à la fois par un client et par un serveur.

Tableau 1.3 : *Exemple simplifié d'une définition de protocole*

Requête client	Réponse serveur
HELLO	HELLO

Tableau 1.3 : *Exemple simplifié d'une définition de protocole*	
Requête client	**Réponse serveur**
RECEVOIR *toto.txt*	Envoyer le fichier *toto.txt*
MESSAGE PAUL bonjour	Envoyer le message « bonjour » à l'utilisateur PAUL

En réalité, l'acceptation d'un nouveau protocole en tant que standard est extrêmement compliquée. Tout doit être parfaitement « ficelé ». Les cas les plus bizarres doivent avoir été considérés. La page **http://sunsite.dk/RFC/rfc/rfc2616.html** présente le protocole HTTP version 1.1 dans toute sa richesse et sa complexité.

Chaque protocole nécessite une application cliente qui lui est propre. Pour le HTTP, c'est un navigateur web ; pour le NNTP, c'est un lecteur de news ; pour le FTP, il s'agit d'un client FTP.

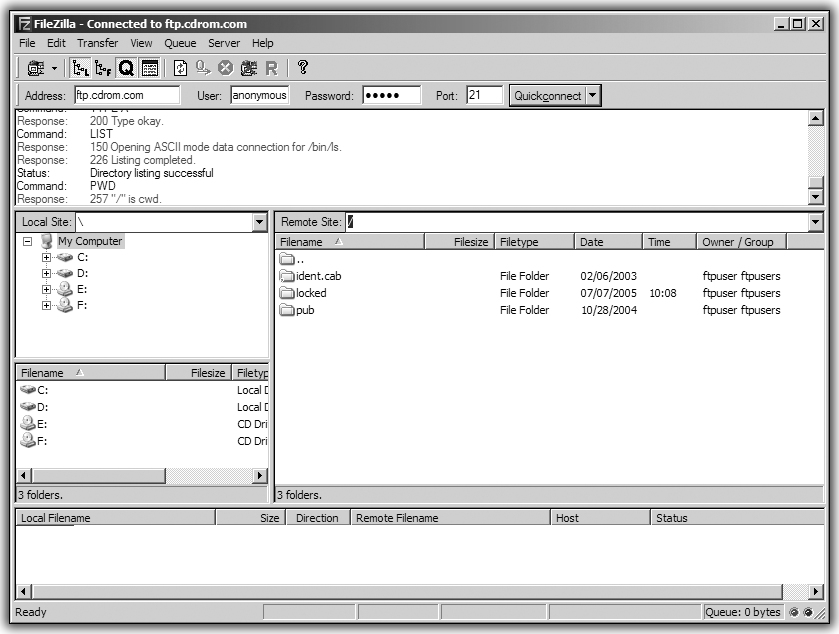

Figure 1.4 : *Client FTP listant les fichiers disponibles à l'adresse ftp://ftp.cdrom.com*

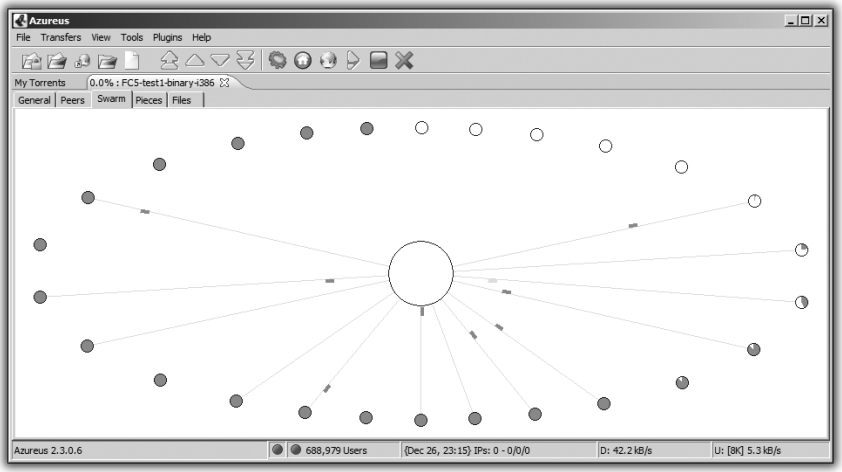

Figure 1.5 : *Azureus, logiciel permettant de télécharger sur le réseau BitTorrent*

Cependant, de plus en plus d'applicatifs permettent d'avoir accès à plusieurs protocoles. Ainsi, le logiciel Outlook Express, de Microsoft, permet de lire des courriels (POP) et des news (NNTP). Les navigateurs, quant à eux, permettent souvent d'avoir accès au protocole FTP. Si nous voulons avoir accès au serveur FTP ayant pour adresse **ftp.cdrom.com**, il suffit de taper l'URL (Uniform Resource Locator) `ftp://ftp.cdrom.com`.

Figure 1.6 : *Un navigateur web accédant au serveur FTP ftp.cdrom.com*

TCP/IP et Internet

Malgré cette profusion de protocoles, une chose ne change pas ; ces réseaux sont tous fondés sur un protocole sous-jacent unique : TCP/IP (Transmission Control Protocol/Internet Protocol). C'est cette combinaison de réseaux basés sur TCP/IP que l'on appelle Internet. Le réseau Internet est vraiment l'élément fondateur qui a permis l'émergence du Web, du mail et de tous ces services dont nous ne pourrions plus nous passer aujourd'hui. Comme pour beaucoup d'avancées scientifiques, cette invention est d'origine militaire. Dans les années 1960, une équipe de chercheurs plancha sur un système permettant d'assurer la continuité des échanges d'informations sensibles (entre des postes stratégiques), et cela même si certains postes (et donc certaines liaisons) étaient détruits. De cet impératif naquit TCP/IP. Ce protocole permet de découper les informations (pages web, courriels, images) en petits paquets et de les acheminer (de les « router ») d'un point à un autre. L'idée originale est la suivante : pour arriver à la même destination, tous ces paquets ne sont pas obligés de passer par la même route. Il est ainsi possible qu'un paquet passe par l'Asie pour aller de la France vers l'Angleterre si aucune autre route n'est à ce moment disponible. Il faut donc imaginer Internet comme un maillage mondial de serveurs interconnectés. C'est du fait de cette architecture que l'on parle, à propos du Web, de « toile d'araignée ».

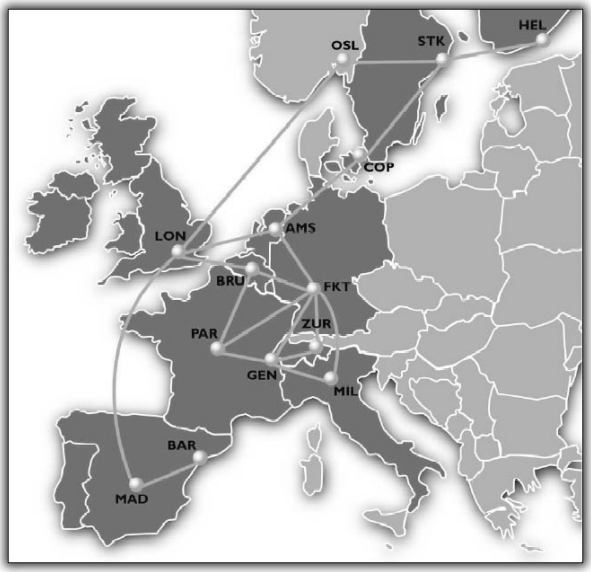

Figure 1.7 : *Quelques interconnexions européennes de la société Cable and Wireless*

Et le fournisseur d'accès dans tout ça ?

Qu'il s'agisse de Free, de Wanadoo, d'AOL, etc., son rôle est de relier votre ordinateur aux autres machines présentes sur Internet. La ligne téléphonique assume alors la fonction de lien. Votre modem sert à faire transiter les données informatiques entre Internet et votre ordinateur. Comme les modems classiques sont très lents, d'autres méthodes sont maintenant proposées pour créer un canal entre Internet et vous : le câble (le même qui vous permet d'accéder aux chaînes de télévision), les lignes téléphoniques « boostées » (l'ADSL), le satellite (pour les plus fortunés), les ondes radio et bientôt vos prises électriques, le WiMax.

Le serveur web

Le Web est donc un protocole applicatif fonctionnant sur un mode client-serveur.

Quand l'internaute souhaite voir la page *information* du site monsite.com, située à l'adresse (URL) **www.monsite.com/info.html**, il utilise un navigateur de type Internet Explorer, Netscape, Firefox, Opera, Lynx, Konqueror…

Figure 1.8 : *Le site kernix.com vu depuis le navigateur ELinks (mode texte) sous Unix*

Le navigateur va ensuite demander au serveur ayant l'adresse **www.monsite.com** de lui transmettre la page ayant pour nom *info.html*.

Essayons de comprendre plus en détail comment cet échange se déroule.

Étape 1 : le navigateur envoie une requête

Vous commencez par écrire l'URL `http://www.google.fr/index.html` dans votre navigateur.

Figure 1.9 : *Affichage d'une URL dans un navigateur*

Une fois cette adresse validée, le navigateur va enchaîner différentes actions.

Il commence par regarder quel type de protocole va être utilisé. Si l'URL commence par `http://`, c'est le protocole HTTP. En revanche, s'il s'agit de `ftp://`, le navigateur devra alors utiliser le protocole FTP.

Il doit ensuite découvrir où se trouve le serveur web **www.google.fr**. Pour cela, il va utiliser un autre service du Net : les DNS (*Domain Name Server* ; des adresses DNS sont systématiquement fournies par votre fournisseur d'accès). Les DNS sont des serveurs qui permettent d'associer un nom de domaine (**www.google.fr**) à une IP (`216.239.39.101`), un peu comme les pages blanches associent M. Dupont à son numéro de téléphone 01 02 03 04 05. Cette adresse IP est unique et identifie par conséquent de manière tout aussi unique le serveur web. C'est donc grâce à cette adresse IP que votre navigateur va pouvoir repérer le bon serveur sur Internet et entrer en contact avec lui. Comme il s'agit, dans ce cas, du protocole HTTP, le navigateur va communiquer avec le serveur HTTP (web) du serveur `216.239.39.101`. La requête qui sera faite vise à obtenir la page ayant pour nom *index.html*.

IP de Google

Si vous tapez l'URL `http://216.239.39.101`, vous arrivez sur la même page, ce qui est d'ailleurs plutôt rassurant.

Étape 2 : le serveur web retourne le fichier

Le serveur web a donc reçu une requête d'une machine qui souhaite obtenir le fichier *index.html*. Il va donc chercher sur son disque dur le fichier *index.html*, récupérer son contenu, puis envoyer ce flux de données au navigateur. Il sait où le renvoyer car, dès que vous êtes sur Internet, vous disposez vous aussi d'une adresse IP visible de l'extérieur.

Votre adresse IP

Si vous êtes relié à Internet et que votre système d'exploitation est Windows, vous pouvez connaître votre adresse IP en tapant la commande `IPCONFIG` (la commande peut être exécutée sous DOS ou directement depuis le menu **Démarrer/Exécuter**). Si, ensuite, vous installez un serveur HTTP (apache, IIS) ou FTP (WS_FTP Serveur) sur votre machine, vous serez en mesure d'être « vu » depuis l'extérieur. Si vous transmettez votre adresse IP à un ami, celui-ci pourra se connecter directement à votre machine avec un navigateur ou un client FTP. Votre machine joue alors le même rôle qu'un serveur d'hébergement, à la différence près que votre liaison n'est, dans tous les cas, pas très rapide. Sachez, par contre, que votre IP n'est pas fixe. À chaque fois que vous vous connectez ou déconnectez, vous changez d'IP.

Figure 1.10 : *L'adresse IP est ici 192.168.0.203*

En revanche, si vous aviez demandé l'URL **www.google.fr/intl/fr/about.html**, le serveur serait rentré dans le répertoire *intl*, puis dans le répertoire *fr*, et il aurait trouvé à cet endroit le fichier *about.html*.

Vous comprenez donc qu'une URL est composée de plusieurs éléments :

- le protocole **http://** ;
- l'adresse Internet du serveur, **www.google.fr** (adresse qui peut donc être une adresse IP) ;
- le chemin du fichier `/intl/fr/about.html`.

Le chemin est similaire à un chemin sur votre disque dur (par exemple *C:\tmp\message.txt*), sauf qu'il est noté sous la norme des chemins Unix : */tmp/message.txt*, où la première barre oblique (/) correspond à la racine.

Majuscules ou minuscules dans les URL ?

Pour répondre à cette question, il est nécessaire de diviser le problème en deux. Le nom de domaine (par exemple **www.google.fr**) d'un côté et le chemin d'accès au fichier (par exemple */index.html*) de l'autre. Les DNS n'étant pas sensibles à la casse (à la différence majuscule/minuscule), les URL **http://google.fr** et **http://wWW.gOOglE.fR** sont donc équivalentes.

En ce qui concerne les chemins, le problème est plus compliqué. Si le serveur HTTP fonctionne sous Unix, il sera sensible à la casse. Les chemins */index.html* et */index.htmL* seront donc différents. S'il fonctionne, par contre, sous Windows, le serveur ne sera pas sensible à cette différence. Les serveurs HTTP fonctionnant essentiellement sous Unix/Linux, il est donc préférable de faire attention à la façon d'écrire le chemin d'accès à une page ou à un script.

Étape 3 : le navigateur traite le fichier

Le navigateur reçoit donc le contenu du fichier *index.html*. Comme l'extension du fichier est *.html*, il sera traité comme un fichier HTML (voir Figure 1.11).

Le navigateur ne reçoit dans un premier temps que le contenu textuel de la page. Il doit donc, avant d'afficher la page, récupérer toutes les images contenues dans celle-ci. Il regarde dans le code, trouve toutes les adresses des images et fait des requêtes au serveur web pour les obtenir une par une. Bien évidemment, ces sous-requêtes sont totalement transparentes pour vous. Vous êtes cependant en mesure de les réaliser vous-même : quand le navigateur a besoin de l'image de titre, la requête est la suivante : **http://www.google.fr/images/title_homepage4.gif**. Si vous tapez cette URL dans votre navigateur, vous n'obtenez alors, comme prévu, que l'image de titre (voir Figure 1.12).

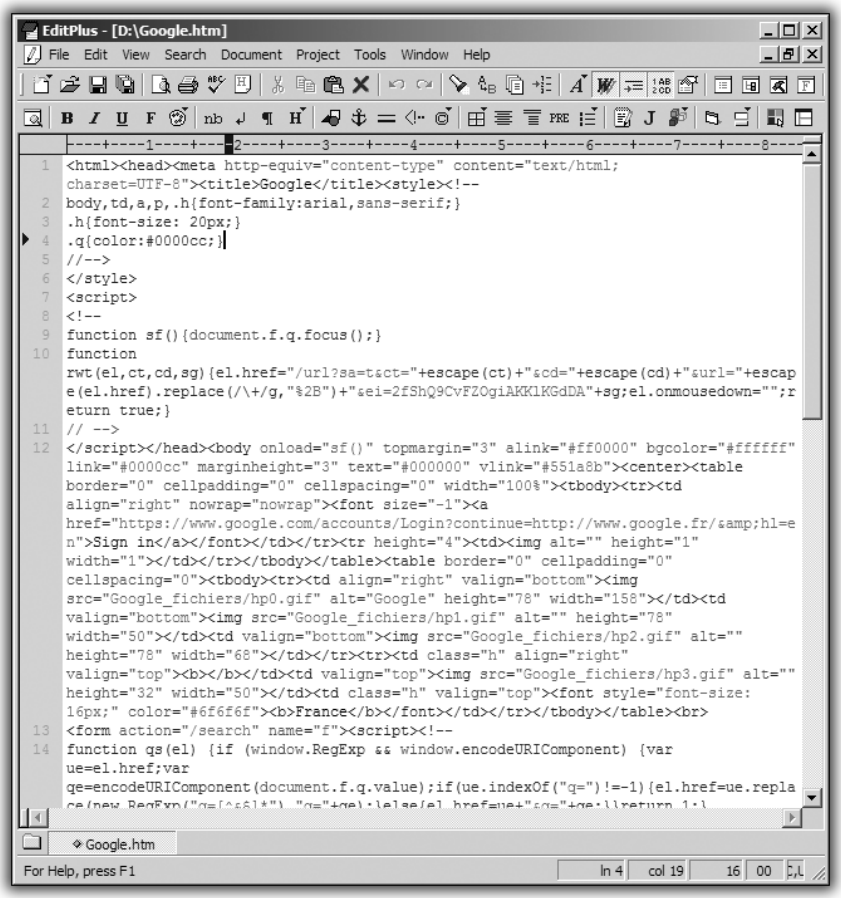

Figure 1.11 : *Voici le contenu du fichier index.html que reçoit le navigateur*

Figure 1.12 : *Requête permettant de n'obtenir que l'image de titre*

Une fois tous les éléments constitutifs de la page récupérés par le navigateur, celle-ci peut être affichée.

PHP

Vous avez vu que lorsque le navigateur fait une requête sur un fichier HTML ou sur une image le serveur lui retourne le contenu du fichier tel quel, sans lui apporter la moindre modification.

Quand, par contre, la requête est faite sur un fichier disposant d'une extension *.php*, tout se passe différemment. Si le serveur web est compatible PHP, celui-ci va traiter tous les fichiers *.php* comme des scripts et transmettra par conséquent le flux de données (le contenu du fichier) à l'interpréteur PHP avant de l'envoyer à l'internaute. Cet interpréteur aura donc pour charge d'évaluer (d'interpréter) le code source PHP et de remplacer les lignes de code par leurs résultats.

Prenons l'exemple d'un fichier *test.php* contenant le code suivant :

```
<?php
  print("bonjour monde");
?>
```

Avant d'être transmis à l'interpréteur, le flux de données contient encore le code ci-dessus. Une fois le travail de l'interpréteur terminé, le flux interprété ne contient plus que `"bonjour monde"`. C'est précisément cette phrase `"bonjour monde"` qui sera transmise et qui apparaîtra dans votre navigateur.

Au niveau du serveur web, PHP tel un « filtre » modifie donc à la volée le flux de données.

Il devient évident qu'un script écrit en PHP qui n'aurait pas d'extension serait considéré comme un simple fichier texte (ou HTML) et ne serait pas traité en tant que script PHP (son code ne serait pas interprété).

Figure 1.13 : *Sans extension, le script est considéré comme du simple texte*

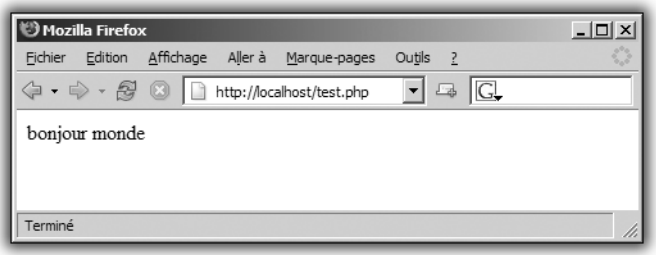

Figure 1.14 : *Avec une extension .php, le script est reconnu comme un script PHP ; il est donc interprété*

Le script doit avoir une extension particulière pour être reconnu en tant que script PHP. L'extension la plus répandue est *.php* (par exemple *test.php*). Il est néanmoins possible de rencontrer d'autres extensions : *.php3* ou *.phtml*. L'hébergeur précise généralement quelle extension doit être utilisée. Si le choix vous est offert, utiliser l'extension *.php* semble plus logique car nous en sommes aujourd'hui à la version 5 de PHP. De plus, cette extension risque fort de devenir la norme.

Les autres langages du Web

Les personnes intéressées par les technologies liées au Web auront pu s'apercevoir de la grande quantité de langages qui fourmillent sur la Toile.

Nous nous sommes déjà arrêtés sur les langages interprétés au niveau serveur tels que PHP, Perl, CFM, ou ASPX. Il existe une autre catégorie de programmes qui, eux, sont exécutés au niveau du client (dans le navigateur) : les Javascripts, les applets Java et les animations Flash.

Javascript

Expliquons rapidement la différence de fonctionnement entre ces deux modes.

- Niveau serveur : le script est exécuté sur le serveur à la suite d'une requête d'un navigateur. Le client reçoit ainsi une page prête à être affichée.
- Niveau client : la page retournée par le serveur web contient du code. C'est au niveau du navigateur que ce code va être exécuté.

Étudions deux exemples :

Listing 1-8 : test.php

```
<html>
<body>
<?php
print("test");
?>
</body>
</html>
```

Listing 1-9 : test.html

```
<html>
<body>
<script language=javascript>
document.write("test");
</script>
</body>
</html>
```

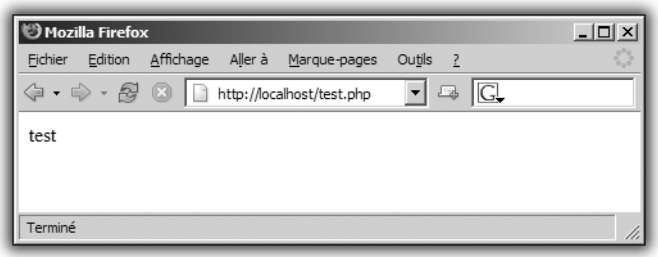

Figure 1.15 : *Résultat obtenu avec test.php ou test.html*

Ces deux programmes donnent un résultat identique, mais sont fondamentalement différents.

Dans le premier cas, le navigateur reçoit une page qui contient déjà le mot `test`, il peut donc l'afficher instantanément. Dans le deuxième cas, en revanche, il reçoit une page qui contient du code Javascript. Il doit par conséquent exécuter ce code avant d'afficher la page. Un navigateur ne gérant pas le Javascript n'aurait rien affiché à l'écran.

Chaque mode a bien évidemment ses inconvénients.

- Au niveau serveur : le fait de devoir interpréter le script dès qu'une requête lui parvient est très lourd à gérer pour le serveur et peut conduire à des ralentissements au niveau de la livraison des pages. Dans le cas de l'exemple ci-dessus, si 10 000 personnes demandent la page *test.php*, le serveur devra interpréter 10 000 fois le script. Si en revanche 10 000 personnes demandent

la page *test.html*, le serveur se contente d'envoyer 10 000 fois la page et les 10 000 exécutions se feront chez les clients. Ce mode permet de répartir la charge de travail et de n'avoir aucun ralentissement. Dans le cas ci-dessus, les deux versions se valent, mais imaginez un script censé chercher et afficher la 100 000e décimale du chiffre pi !

- Au niveau client : le Javascript est un langage certes normalisé, mais qui est géré de façon plus ou moins performante, avec plus ou moins de fonctionnalités selon les navigateurs. Il est ainsi très difficile de rendre un applicatif Javascript exécutable sur tout type de système.

Incompatibilités entre navigateurs

Bien que des normes soient édictées régulièrement par le W3C pour faire évoluer le Web, nous pouvons hélas ! déplorer le fait que les différents navigateurs ne les suivent pas scrupuleusement. Microsoft notamment, avec Internet Explorer, a particulièrement compliqué la vie des développeurs web en leur imposant d'écrire des codes sortant de la norme afin de rester compatible avec son navigateur vedette. L'émergence de Firefox et plus généralement du monde du libre (free software) commence néanmoins à marquer les esprits et Microsoft devrait avec son futur Internet Explorer 7 gagner en compatibilité (Javascript, CSS, DOM).

Java

Il est important, avant de clore ce chapitre, de s'intéresser quelques instants à Java. Développé par la société Sun Microsystem, initialement pour des composants embarqués, ce langage a fait l'effet d'une bombe lors de sa sortie. Le principe était relativement simple et révolutionnaire : créer un langage qui permette de tirer profit des avantages du langage interprété et du langage compilé. Au lieu de compiler des sources Java directement en un binaire propre à un type de processeur, la compilation se fait dans un langage intermédiaire (*bytecode*), qui doit être interprété, par la suite, par une machine virtuelle. L'interprétation est alors facilitée et correspond plus à une conversion.

Il fut très vite évident que Java aurait son rôle à jouer sur le Web, où foisonnaient une multitude de systèmes (Mac, PC, Windows, Unix, etc.), et ce fut par l'intermédiaire des applets qu'il se fit connaître du grand public. Une applet java est un programme Java qui s'exécute dans un navigateur. À la différence du Javascript, il est possible, en Java, de

construire des applications disposant d'interface graphique complexe. Des applets de tableurs, de dessins, de chats firent ainsi leur apparition sur le Web à la stupéfaction générale.

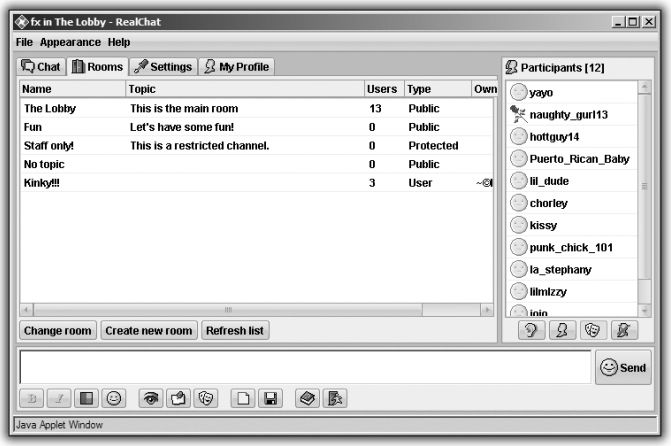

Figure 1.16 : *Un logiciel de chat dans un navigateur web*

À la plus grande joie des internautes, les applets Java permirent aussi l'arrivée des jeux sur le Web.

Figure 1.17 : *Spaceball : jeu écrit en Java, fonctionnant sur tout type de plateforme*

Hélas, les applets sont des programmes qui demandent beaucoup de ressources machine (processeur, mémoire) ! Elles nécessitent en outre la présence d'une machine virtuelle Java et sont généralement lentes à charger. Toute la politique de Sun fut donc de transférer la technologie Java vers le niveau serveur avec les servlets. Avec certaines extensions, le serveur Apache (via son cousin Tomcat) est désormais en mesure d'interpréter du Java aussi facilement que du code PHP.

Animation Flash

Le Flash est un format développé par la société Macromedia qui permet de créer des animations sur le Web. Alors qu'il s'agissait au départ d'un outil essentiellement graphique destiné aux designers web, Macromedia a vite compris qu'il disposait d'une véritable bombe et qu'il pouvait en faire un véritable environnement de développement pour le Web. L'environnement Flash MX permet désormais de réaliser des interfaces graphiques complètes, d'interagir avec des services web, de manipuler les fichiers XML et de développer des applicatifs en utilisant le langage interne aussi puissant que complet qu'est ActionScript.

FLA et SWF

Deux formats de fichiers sont liés au Flash : le FLA qui peut être comparé au fichier source et le SWF qui correspond à l'exécutable. Si vous souhaitez apporter des modifications, vous devez donc disposer du FLA pour l'ouvrir dans Flash MX. Au contraire, pour exécuter l'animation au sein d'un navigateur, le SWF vous sera nécessaire.

Flash a l'avantage d'avoir un rendu identique sur tous les navigateurs à la condition, certes très restrictive, que l'ordinateur dispose du plug-in Flash. Lorsque l'on sait que Macromedia est un concurrent de Microsoft (éditeur de l'incontournable Windows) et que les distributions Linux rechignent à installer des logiciels non libres, nous ne sommes pas près de disposer du plug-in Flash préinstallé sur nos machines. Le caractère propriétaire de Flash à l'heure où les formats de fichiers tendent tous à s'ouvrir risque également de freiner l'adoption de cette technologie parmi les sociétés éditrices de logiciels en ligne.

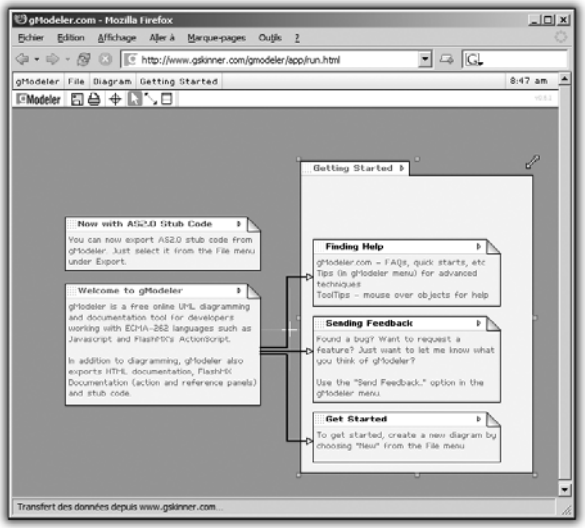

Figure 1.18 : *Outil de création de diagrammes intégralement réalisée en Flash*

1.4. Check-list

- PHP n'est qu'un langage de programmation parmi d'autre.
- PHP fait partie des langages interprétés.
- L'interprétation des scripts PHP est réalisée le plus souvent au niveau du serveur web dont le meilleur représentant est Apache.
- PHP est particulièrement adapté aux développements web.
- PHP fonctionne sur une multitude de plateformes.
- PHP est libre et gratuit.
- Le Web est une des dimensions d'Internet au même titre que les courriels ou le P2P.

Chapitre 2

L'environnement de travail

L'objectif est ici de mettre en place sur votre machine un environnement de travail permettant de tester les exemples présentés dans la suite de l'ouvrage. À l'issue de ce chapitre, vous disposerez d'un serveur web capable d'interpréter des scripts PHP, d'un système de gestion de bases de données (MySQL) et d'un outil permettant d'interagir avec ce dernier : phpMyAdmin.

2.1. WampServer

Bien qu'Apache, MySQL, et PHP puissent être installés séparément sous Windows, le choix se portera ici sur un outil capable d'automatiser l'intégralité de ce processus.

Installation

Wamp Server peut être téléchargé sur le site **www.wampserver.com** sous la rubrique *Downloads*.

1. Double-cliquez sur l'archive que vous venez de télécharger

De nombreuses modifications ont été apportées à la version 2 de Wamp Server. Les créateurs conseillent par conséquent aux personnes disposant déjà d'une version de Wamp Server sur leur machine, de sauvegarder leurs développements, de désinstaller.l'ancienne version et enfin d'installer la toute nouvelle.

Compatibilité du code

Les développements (PHP, MySQL) réalisés sur une version antérieure de Wamp Server ont 99 chances sur 100 de fonctionner sur une version plus récente de Wamp Server. Les projets PHP et MySQL apportent en effet une importance énorme au fait de maintenir la compatibilité des développements d'une version à l'autre.

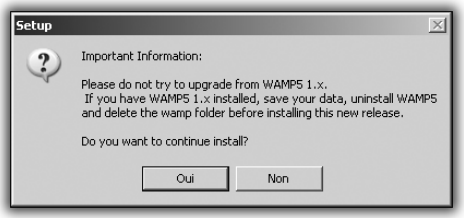

Figure 2.1 : *Avertissement*

2 La fenêtre d'installation se lance et vous présente votre version de Wamp.

Cette version est sans aucun rapport avec celle des outils qu'elle contient (que ce soit Apache, PHP ou MySQL).

Figure 2.2 : *l'installation de Wamp Server 2*

L'écran suivant permet d'accepter la licence.

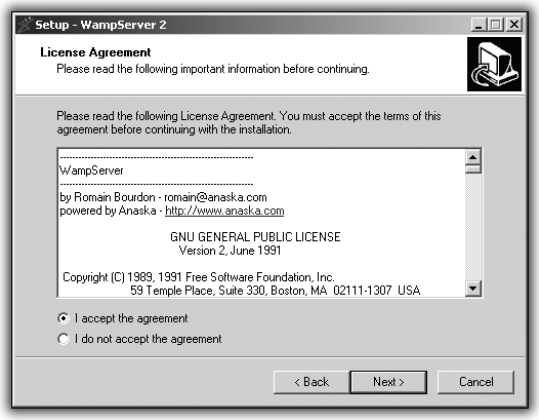

Figure 2.3 : *Acceptez…*

3 Précisez le répertoire qui contiendra l'ensemble des composants ainsi que vos sources.

Ce répertoire représentera approximativement 100 Mo de données. N'hésitez pas à sélectionner un autre disque si vous sentez que l'espace libre de votre partition *C:* est trop juste. Un changement d'emplacement dans un deuxième temps serait beaucoup plus compliqué.

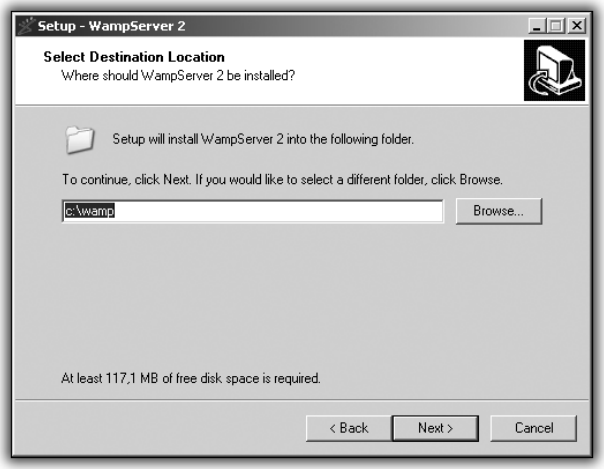

Figure 2.4 : *Indiquez le répertoire*

L'écran suivant permet d'obtenir des raccourcis sur le bureau pour lancer Wamp Server. Il n'est pas nécessaire de cocher ces options dans la mesure où il restera possible de démarrer Wamp Server en passant par le menu **Démarrer**.

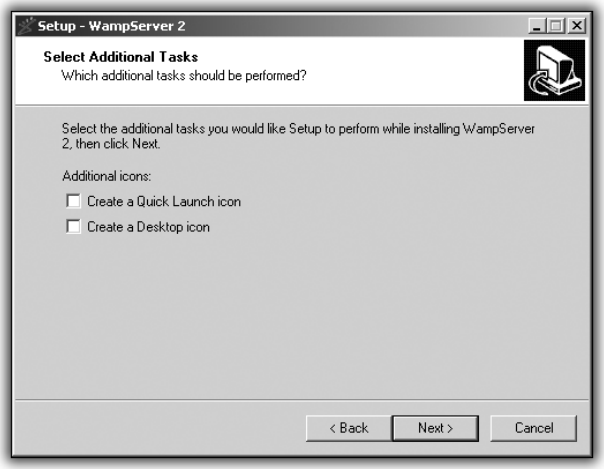

Figure 2.5 : *Raccourcis*

L'étape suivante résume vos différents choix d'installation.

4 Il est encore temps de les modifier en revenant en arrière par le bouton < **Back**.

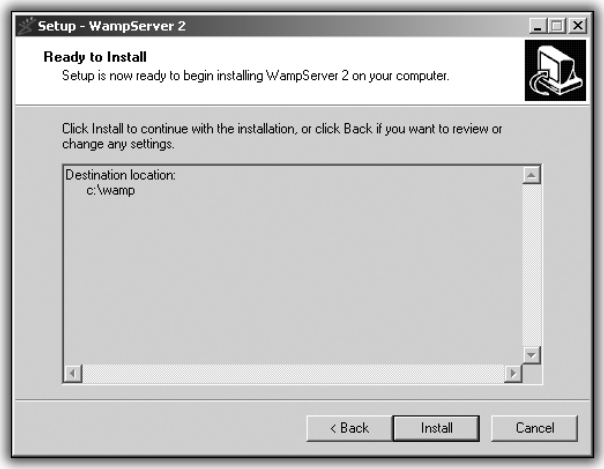

Figure 2.6 : *Résumé de l'installation*

La prochaine étape correspond à l'installation physique des composants sur votre machine. Pas loin de 2000 fichiers sont installés dans le répertoire *C:\wamp*.

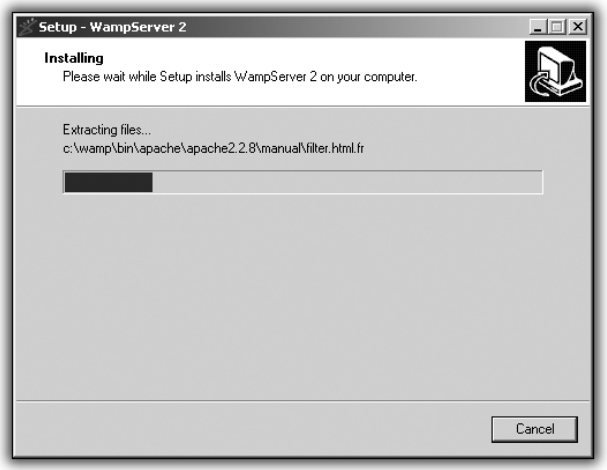

Figure 2.7 : *En cours…*

5 Définissez le navigateur qui sera utilisé pour ouvrir ces pages.

Nous vous conseillons d'utiliser le navigateur Firefox.

Figure 2.8 : *Emplacement de Firefox*

6 Indiquez votre adresse email ainsi que le nom de votre serveur d'envoi de mails. Ce serveur, dit SMTP, est différent pour chaque fournisseur d'accès à Internet. Il peut par exemple prendre les valeurs suivantes : `smtp.free.fr`, `smtp.wanadoo.fr`, `smtp.noos.fr`, `smtp.club-internet.fr`, etc.

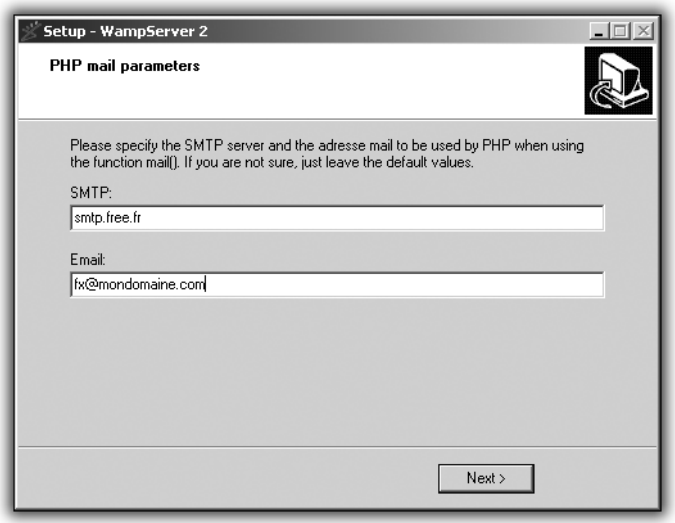

Figure 2.9 : *Serveur SMTP*

L'étape finale vous confirme que l'installation s'est bien déroulée et vous propose de démarrer sans plus attendre le Wamp Server.

Figure 2.10 : *Installation terminée*

Premiers pas

À l'issue de ces étapes, Apache, MySQL et PHP sont installés sur votre machine. Vous pouvez le vérifier en réalisant une première requête sur le domaine associé à votre machine : **http://localhost**.

Figure 2.11 : *tout fonctionne merveilleusement, Apache retourne la première page web*

La page affichée correspond au fichier *index.php* situé dans *C:\wamp\www*. Vous pouvez de la même manière faire appel à votre propre page en plaçant le fichier *test.html* dans ce même répertoire et en y faisant appel de la manière suivante : `http://localhost/test.html`.

Listing 2-1 : c:\wamp\www\test.html

```
<hr/>bonjour monde<hr/>
```

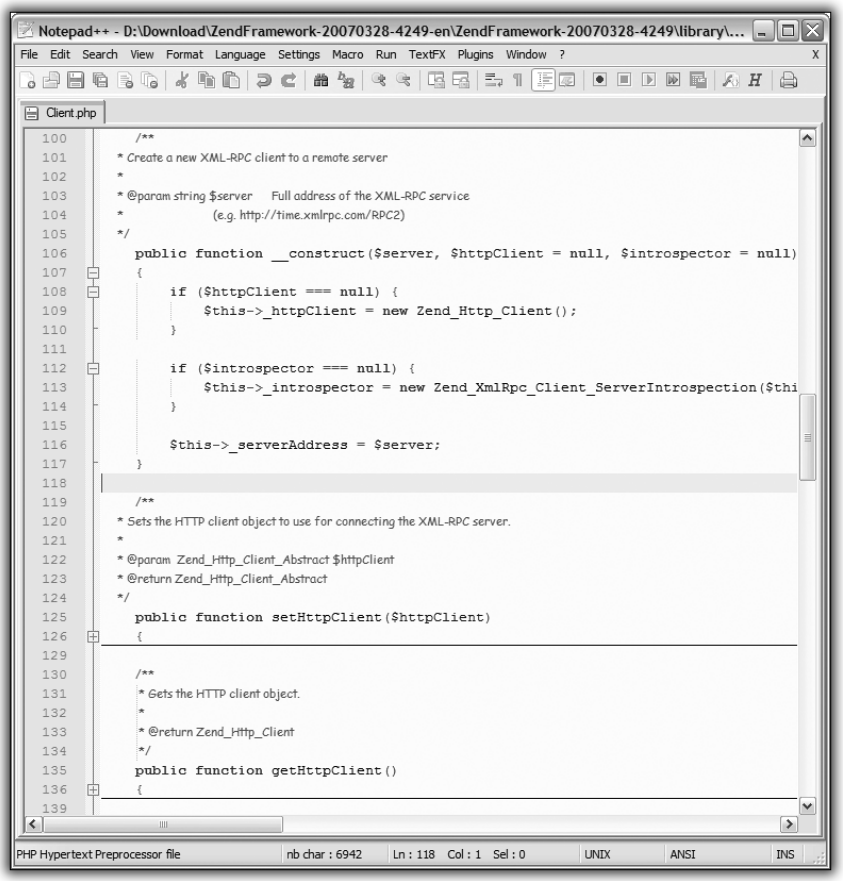

Figure 2.12 : *Les pages sont également accessibles*

Le démarrage de Wamp a enrichi votre System Tray d'une petite icône évoquant un compteur de vitesse.

Cette icône peut prendre différentes couleurs en fonction de l'état des serveurs.

- **Marron** : aucun service n'est démarré.
- **Jaune** : un seul service est démarré.
- **Blanche** : les deux services sont démarrés.

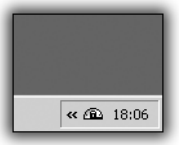

Figure 2.13 :
Wamp est démarré, tout est OK

Le gestionnaire de tâches de Windows confirme également que les programmes Apache (*httpd.exe*) et MySQL (*mysqld-net.exe*) résident bien en mémoire. La colonne *Util. mémoire* permet de quantifier la mémoire utilisée par ces services :

- *2 x Apache.exe* : 25 Mo.
- *1 x mysqld-nt.exe* : 12 Mo.
- *1 x wampmanager.exe* : 4 Mo.

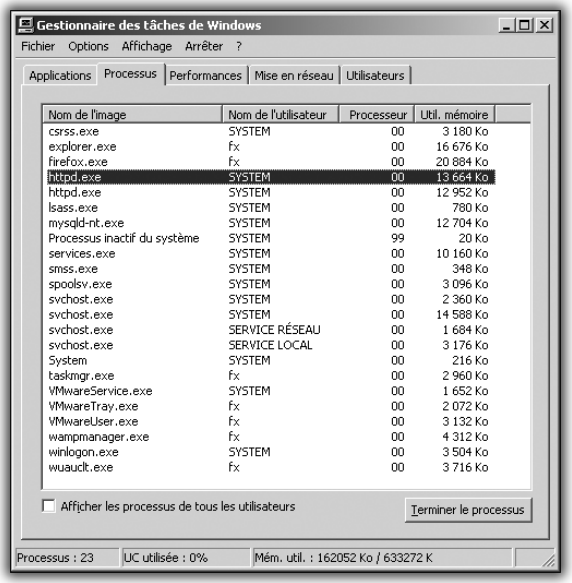

Figure 2.14 : *Gestionnaire de tâches Windows*

Si vous avez choisi de ne pas lancer Wamp Server automatiquement au démarrage, vous pouvez à tout moment le faire par le menu **Démarrer/Tous les programmes/WampServer/Start WampServer**.

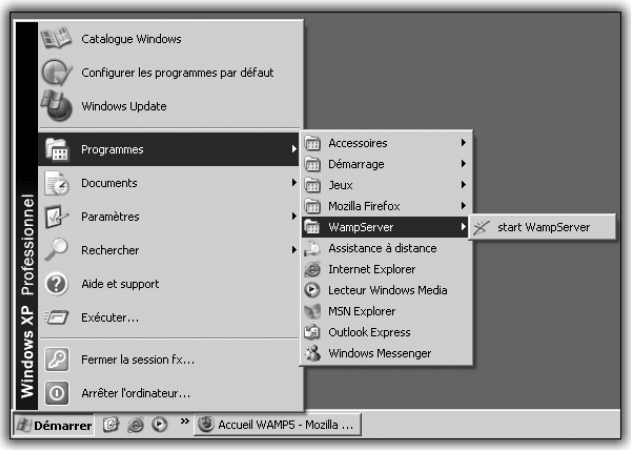

Figure 2.15 : *Menu d'accès*

Le menu de Wamp

En cliquant avec le bouton gauche de la souris sur l'icône de Wamp, vous affichez un petit menu déroulant.

Figure 2.16 :
Actions accessibles depuis le menu Wamp

Ce menu permet :

- d'ouvrir directement des pages, notamment la page d'accueil et l'application phpMyAdmin ;

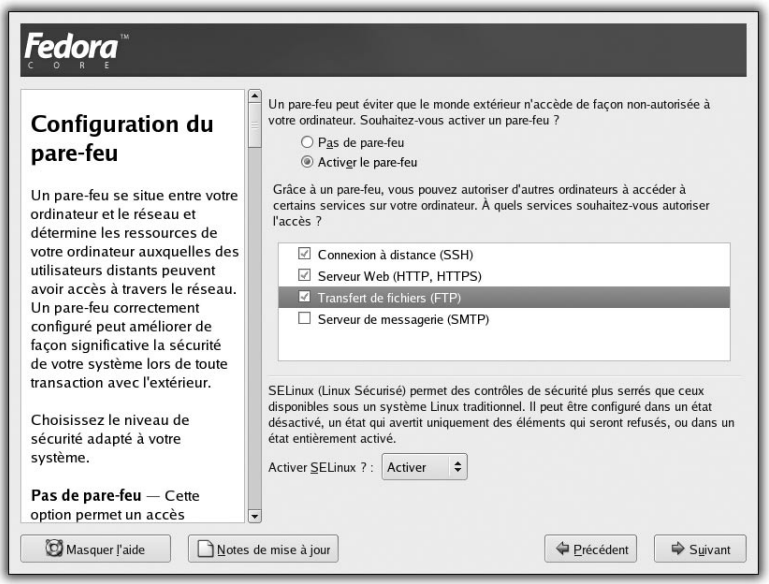

Figure 2.17 : *phpMyAdmin*

- d'ouvrir le répertoire *www* qui va contenir vos futurs scripts ;

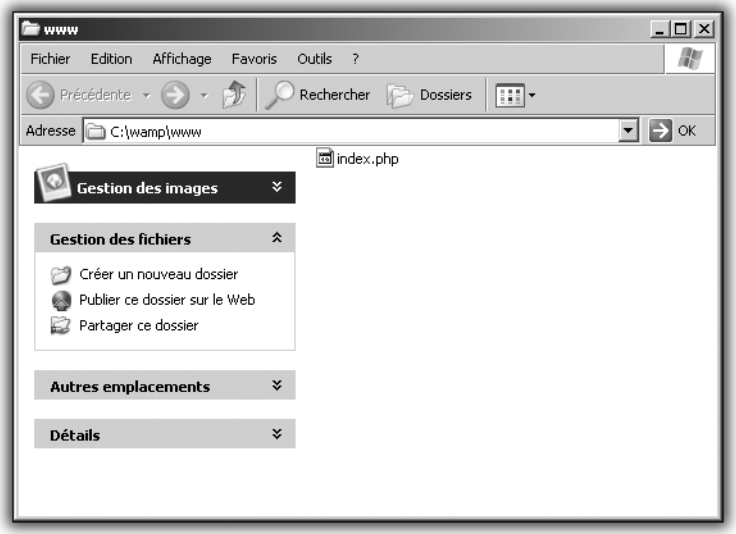

Figure 2.18 : *Le répertoire racine de du serveur web*

- de lire les fichiers LOG d'Apache, de MySQL et de PHP ;

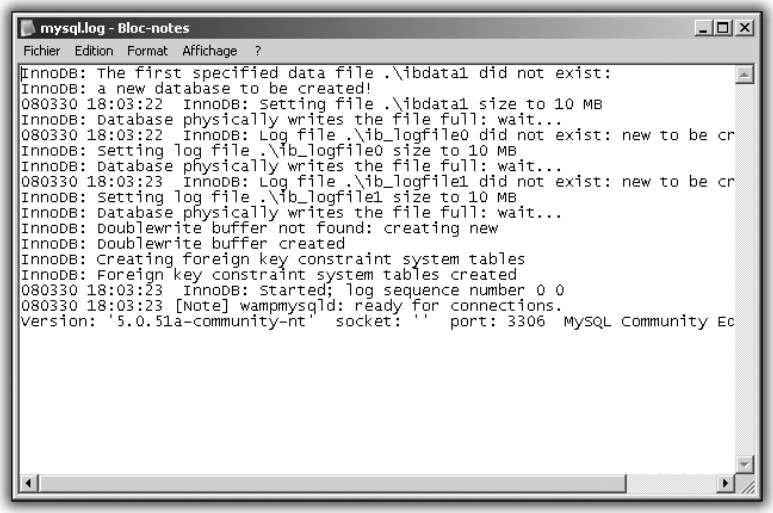

Figure 2.19 : *Un exemple de fichier LOG : mysql_error.log*

- d'éditer leur fichier de configuration ;

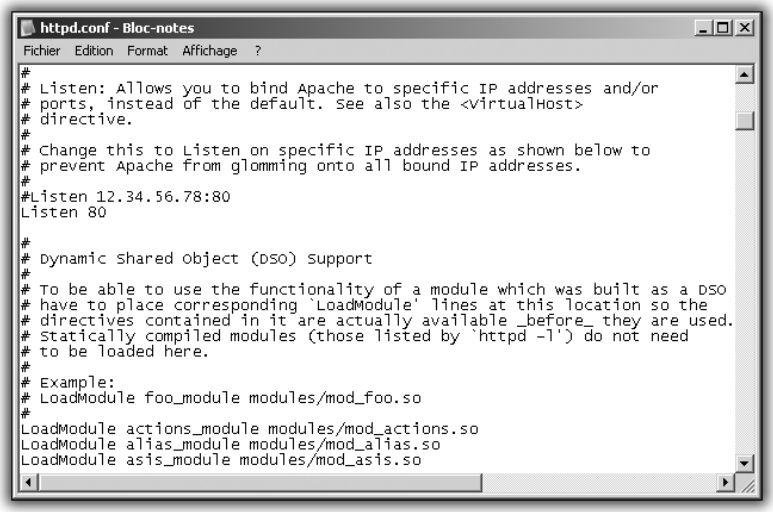

Figure 2.20 : *Un exemple de fichier de configuration : httpd.conf*

- de gérer les extensions PHP ;

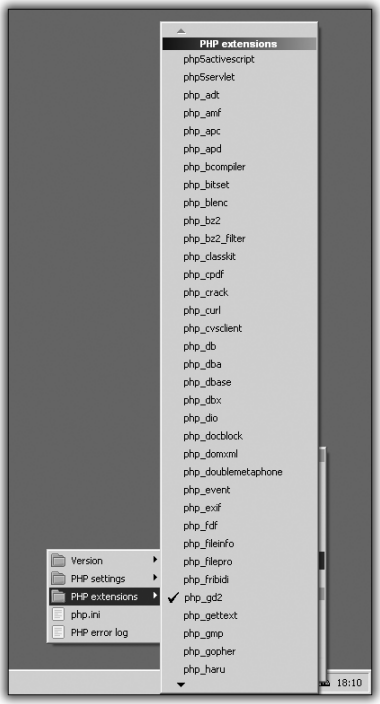

Figure 2.21 : *Chargement, suppression d'extensions PHP*

- d'arrêter, de démarrer et de redémarrer Apache et MySQL.

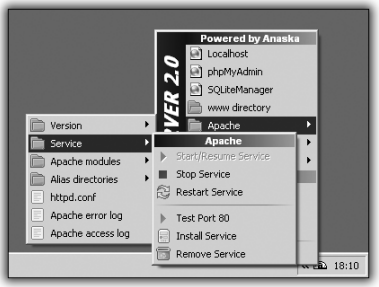

Figure 2.22 : *Menu d'Apache*

L'éditeur Notepad++

Les éditeurs de code PHP pullulent sur le Net. Un site web leur est même consacré : **www.php-editors.com/review**.

Chacun dispose bien évidemment de ses propres avantages et inconvénients. Notre choix se portera sur Notepad++ qui apparaît particulièrement adapté aux besoins du développeur PHP.

- Il est gratuit.
- Il est rapide.
- Il colorise le code.
- Il réduit les blocs de code (fonctions, class, structures de contrôle).

Figure 2.23 : *édition d'un script PHP avec Notepad++*

Cet éditeur peut être téléchargé à l'adresse **http://notepad-plus.sourceforge.net/fr/site.htm**.

2.2. Paramétrage de PHP

Au moment de sa mise en œuvre, PHP prend en compte un certain nombre de directives de configuration. Ces dernières, placées dans le fichier *php.ini*, permettent de paramétrer :

- le parseur PHP ;
- la sécurité ;

- les rapports d'erreur et les fichiers LOG ;
- la compatibilité avec les versions précédentes ;
- les extensions.

Sous Windows, ce fichier peut être ouvert via l'icône *Wamp Server* au sein du menu **PHP**.

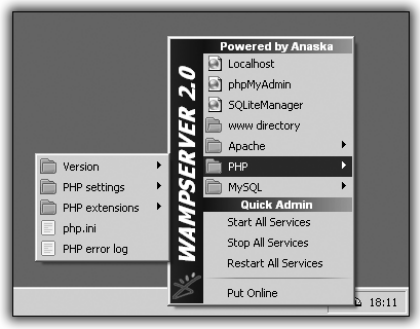

Figure 2.24 : *L'accès aux fichiers de configuration*

Le fichier se situe physiquement dans le répertoire *C:\wamp\bin\apache\apacheX.X.X\bin*.

Tableau 2.1 : *Paramètres de configuration de PHP dans php.ini*

Paramètre	Valeur	Signification
`allow_url_fopen`	Booléen : `On` ou `Off`	Autorise l'ouverture de fichiers distants
`asp_tags`	Booléen	Autorise les balises ASP de type `<% %>`
`date.timezone`	String	Précise la zone géographique de la machine (par exemple `"Europe/Paris"`)
`default_mimetype`	String	Permet de préciser le type de fichier retourné par défaut par PHP (par exemple `"text/html"`)
`default_charset`	String	Permet de préciser l'encodage par défaut des caractères (par exemple `"iso-8859-1"`)
`display_errors`	Booléen	Autorise l'affichage des erreurs
`doc_root`	String	Définit le dossier racine

Tableau 2.1 : *Paramètres de configuration de PHP dans php.ini*

Paramètre	Valeur	Signification
`engine`	Booléen	Permet d'interdire l'usage de l'interpréteur PHP
`extension_dir`	String	Permet d'indiquer le répertoire contenant les extensions (par exemple `"c:/wamp/php/ext"`)
`error_log`	String	Définit le fichier dans lequel les erreurs seront « loggées »
`error_reporting`	Entier	Définit le niveau d'affichage des erreurs
`file_uploads`	Booléen	Autorise ou non le transfert (*upload*) de fichiers
`include_path`	String	Définit les répertoires dans lesquels les fonctions `require()` et `include()` vont chercher les fichiers (par exemple, sous Windows, `.;c:\wamp\www\boutique` ; sous Linux, `.:/var/web/html/boutique`)
`log_errors`	Booléen	Indique si les erreurs des scripts doivent être enregistrées dans le fichier LOG du serveur
`magic_quotes_gpc`	Booléen	Autorise l'échappement automatique des caractères `'`, `"`, `\` et `NUL`
`max_execution_time`	Entier	Définit le temps maximal d'exécution d'un script (en secondes)
`memory_limit`	Entier	Définit la mémoire maximale que peut utiliser un script (par exemple `8M`)
`open_basedir`	String	Restreint l'ouverture des fichiers à un répertoire spécifique
`register_globals`	Booléen	Indique si les variables EGPCS doivent être initialisées en tant que variables globales

Tableau 2.1 : *Paramètres de configuration de PHP dans php.ini*

Paramètre	Valeur	Signification
`safe_mode`	Booléen	Permet de passer en mode « sécurisé » et de vérifier que le propriétaire d'un script est le même que celui du fichier accédé
`safe_mode_exec_dir`	String	Répertoire contenant les binaires pouvant être exécutés en mode sécurisé
`safe_mode_gid`	Booléen	Permet de vérifier que le groupe du propriétaire d'un script est le même que celui du fichier accédé
`session.save_path`	String	Répertoire de stockage des sessions
`session.name`	String	Nom du cookie de session (par exemple `PHPSESSID`)
`session.cookie_lifetime`	Entier	Durée de vie du cookie de la session ; la valeur `0` (par défaut) signifie que la session sera détruite avec la fermeture du navigateur
`short_open_tag`	Booléen	Autorise les balises raccourcies `<? ?>`
`upload_tmp_dir`	String	Précise le répertoire qui sera utilisé pour stocker temporairement les fichiers « uploadés »
`upload_max_filesize`	Entier	Précise la taille maximale des fichiers uploadés (par exemple `4M`)
`variables_order`	String	Définit l'ordre dans lequel PHP va initialiser ses variables globales ; la valeur `"EGPCS"` correspond à l'ordre suivant : `$_ENV`, `$_GET`, `$_POST`, `$_COOKIE`, `$_SESSION`
`zend.ze1_compatibility_mode`	Booléen	Permet à PHP 5 de se comporter comme PHP 4 au cœur du système PHP
`zlib.output_compression`	Booléen	Permet de compresser à la volée l'ensemble des données affichées par PHP

Toute modification apportée au fichier *php.ini* doit être accompagnée du redémarrage du serveur HTTP.

Le fichier httpd.conf

Ces valeurs peuvent également être précisées au sein du fichier de configuration d'Apache : *httpd.conf*. Il est même possible, en utilisant la directive `Virtualhost`, d'initialiser spécifiquement les différents sites présents sur la machine. Il pourrait ainsi être possible d'autoriser l'upload de fichiers pour un site, et de l'interdire pour un autre.

```
<VirtualHost *:80>

  ServerName       www.site.com
  DocumentRoot     /var/web/site

  php_admin_flag   engine               on
  php_admin_flag   file_uploads        off

</VirtualHost>
```

2.3. Check-list

- Wamp Server est un logiciel permettant d'installer très facilement sous Windows les logiciels Apache, MySQL et PHP.
- L'outil phpMyAdmin, lui-même écrit en PHP, permet de créer et de gérer les bases de données.
- Le fichier *php.ini* contient les directives de configuration de PHP.
- Une modification de ce fichier impose le redémarrage du serveur Apache.

Chapitre 3

Les fondamentaux

Nous nous intéresserons dans ce chapitre à la syntaxe du PHP ainsi qu'aux différents éléments qui composent ce langage : les variables, les types de données, les fonctions et enfin les structures de contrôle. Ces éléments correspondent aux briques fondamentales qui permettront par la suite de réaliser des applications de plus grande envergure. Une bonne compréhension de ce chapitre est donc indispensable pour pouvoir passer sereinement aux chapitres suivants.

Avant d'entrer dans le vif du sujet, il est important de fixer un certain nombre de conventions. Vous avez lu précédemment que PHP est un langage de programmation interprété. En ce sens, un script écrit en PHP nécessite un autre composant pour être exécuté : l'interpréteur PHP. Par défaut, les systèmes d'exploitation les plus répandus (Windows, Mac OS) ne disposent pas de tels interpréteurs. Il existe donc deux possibilités pour faire fonctionner un programme écrit en PHP :

- utiliser l'environnement mis en place dans le chapitre précédent ;
- le placer chez un hébergeur prenant en charge PHP.

Nous allons considérer dans les prochains chapitres que vous travaillez sur votre propre machine. Comme nous l'avons vu dans le premier chapitre, toute machine dispose d'un nom par défaut : ici, *localhost*. Ce serveur aura donc pour adresse **http://localhost**. Ainsi, quand une mention sera faite d'un script *test1.php*, cela impliquera que vous aurez créé un fichier portant le nom *test1.php*, que vous l'aurez placé dans le répertoire prévu à cet effet et que vous aurez renseigné l'adresse dans votre navigateur **http://localhost/test1.php** pour l'exécuter. Une adresse de type **http://localhost/chap03/test2.php** signifierait quant à elle que vous avez créé un répertoire *chap03* dans le répertoire principal et que vous y avez placé le fichier *test2.php*.

Vous constaterez également le choix du navigateur web Firefox pour illustrer les exemples. En l'espace de quelques années, ce navigateur a pris jusqu'à 30 % de parts de marché à Internet Explorer. Ses avantages sont nombreux et variés :

- respect des standard (passés, présents et futurs) ;
- avancées technologiques (XUL, CANVAS, SVG) ;
- outils utiles aux développeurs web (affichage du code source, console Javascript) ;
- gratuité, sécurité, et rapidité ;

- mises à jour fréquentes et automatiques ;
- nombreuses extensions.

Firefox en français
Firefox peut être téléchargé en langue française sur le site www.mozilla-europe.org/fr/

3.1. Structure d'un programme

Un script écrit en PHP correspond à un fichier texte contenant des lignes de code (instructions). Vous savez depuis le premier chapitre que ce fichier texte doit avoir une extension de type *.php* pour pouvoir être évalué.

Les lignes de codes contenues dans un script PHP doivent être englobées entre les balises `<?php` et `?>` : elles forment alors un bloc de code.

Les balises `<%` et `%>`
Il est possible de rencontrer les balises d'ouverture et de fermeture `<%` et `%>`. Cette notation n'est cependant pas très répandue et devrait être abandonnée avec PHP6.

Écrivons, en suivant ce principe, notre premier programme PHP : *test.php*, un classique du genre.

Listing 3-1 : Votre premier programme PHP

```
<?php
print("bonjour monde");
?>
```

Il s'agit du programme le plus simple que l'on puisse imaginer. La première ligne, `<?php`, indique à l'interpréteur que les lignes qui vont suivre doivent être traitées comme du code PHP et qu'elles doivent donc être interprétées.

La deuxième ligne, `print("bonjour monde");`, est une instruction qui commande à l'interpréteur d'afficher à l'écran la phrase `"bonjour monde"`. En PHP, chaque instruction doit être ponctuée d'un point-virgule (`;`). Au sein du bloc de code, les instructions sont exécutées les unes après les autres.

L'expression `print()` correspond à ce que l'on appelle en programmation une « fonction ». Le rôle de cette fonction est d'afficher la donnée qui lui est adressée en paramètre.

***Par défaut, PHP est livré avec un grand nombre de fonctions qui sont décrites pour la plupart dans le chapitre** « Les fonctions PHP ».*

Ce sont ces fonctions qui permettent, notamment, de travailler sur les nombres, les chaînes de caractères (mots, phrases), les tableaux, d'accéder aux bases de données, de générer des images, etc.

Enfin, la dernière ligne (`?>`) de ce script indique à l'interpréteur que les lignes qui suivent ne sont plus à interpréter. Elles peuvent donc être affichées telles quelles.

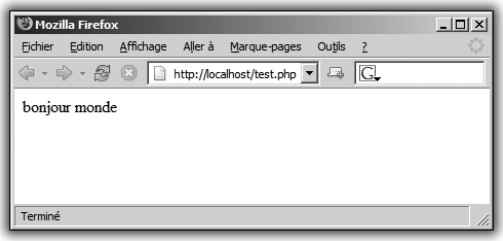

Figure 3.1 :
Le résultat de votre premier programme

Si vous voulez maintenant afficher deux messages, placez deux instructions dans le bloc de code :

Listing 3-2 : Votre script contient deux instructions

```
<?php
print("instruction 1");
print("instruction 2");
?>
```

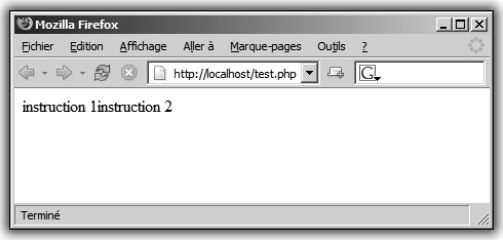

Figure 3.2 :
Résultat

Le résultat peut vous surprendre, les deux phrases se trouvant en effet sur la même ligne. En y réfléchissant bien, c'est cependant tout à fait normal. La première instruction affiche la phrase `"instruction 1"` et la deuxième affiche `"instruction 2"`. Votre navigateur va donc recevoir un fichier contenant `"instruction 1instruction 2"` et l'afficher tel quel. Si vous souhaitez avoir deux lignes différentes, rien de plus simple : ajoutez la balise HTML `<br/>` qui permet de réaliser un saut de ligne :

Listing 3-3 : Deux instructions

```
<?php
print("instruction 1<br/>");
print("instruction 2");
?>
```

La page que le navigateur reçoit contient alors la ligne `"instruction 1<br/>instruction 2"`, ce qui vous permet de séparer les deux lignes.

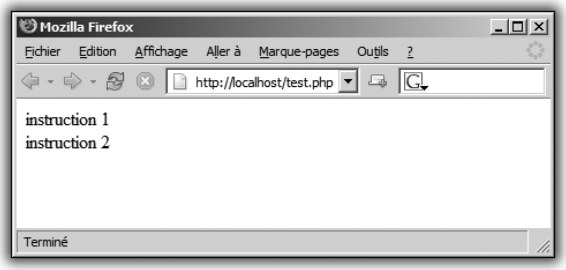

Figure 3.3 : *Le résultat correspond aux attentes*

L'expression `phpinfo()` est une autre fonction interne du PHP. Elle a pour rôle de donner des informations sur le PHP utilisé pour faire fonctionner votre script :

- version du PHP ;
- version des différents modules installés ;
- type de serveur web ;
- données système disponibles…

Listing 3-4 : Informations sur la version de PHP

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

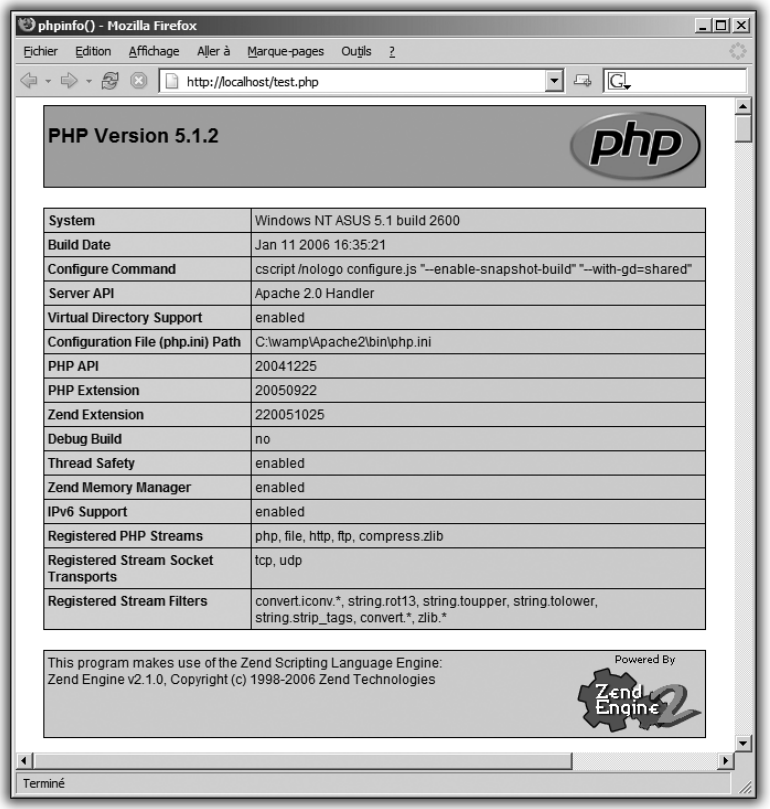

Figure 3.4 : *Informations retournées par la fonction phpinfo()*

Comme vous le voyez, les fonctions ont des rôles très variés. Une simple fonction peut réaliser des tâches plus ou moins complexes.

Le fichier *test.php*

Plutôt que de créer pour chacun des petits exemples un nouveau fichier (*test1.php*, *test2.php*, etc.), le fichier *test.php* sera réutilisé à chaque fois.

Au sein d'un même script, les blocs de code peuvent être multiples et du « code » HTML peut s'y intercaler. Toute zone non comprise entre les balises `<?php` et `?>` est considérée comme du HTML (ou tout du moins comme du texte brut). Elle est par conséquent directement retournée :

Listing 3-5 : PHP + HTML

```
<?php
print("premier bloc PHP");
?>

<hr>
partie <b>HTML</b>
<hr>

<?php
print("deuxième bloc <i>PHP</i>");
?>
```

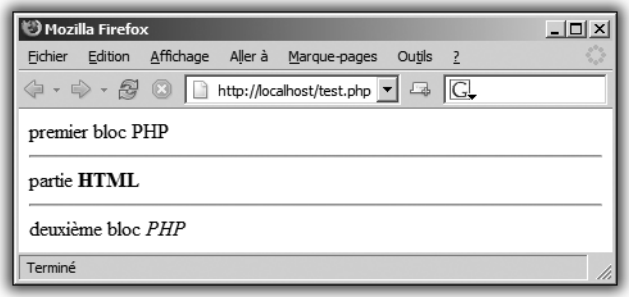

Figure 3.5 : *Du PHP et du HTML au sein du même script*

Au sein de ce script, les deux blocs suivants correspondent à du code PHP :

```
<?php
print("premier bloc PHP");
?>
<?php
print("deuxième bloc <i>PHP</i>");
?>
```

Ces blocs seront remplacés par le résultat de leur interprétation.

La partie suivante correspond quant à elle à du simple code HTML qui est « retourné » tel quel par le serveur.

```
<hr>
partie <b>HTML</b>
<hr>
```

Vous pouvez remarquer que dans le deuxième bloc de code sont affichées des données qui contiennent des balises HTML permettant la mise en italique d'un texte (`<i>PHP</i>`). C'est non seulement tout à fait autorisé, mais de plus très courant en PHP.

HTML et PHP

Il serait tout à fait envisageable qu'un fichier *.php* ne contienne que du HTML. La page serait retournée sans aucune modification. L'intérêt est cependant bien mince car vous ajoutez, dans ce cas, une étape entre le serveur web et vous. En effet, bien que la page ne contienne aucun bloc d'instructions, l'interpréteur sera quand même mis à contribution.

En affichant les sources de la page, vous obtenez la preuve que le navigateur a bien reçu le résultat de l'interprétation du code par PHP et que la partie HTML n'a pas été modifiée.

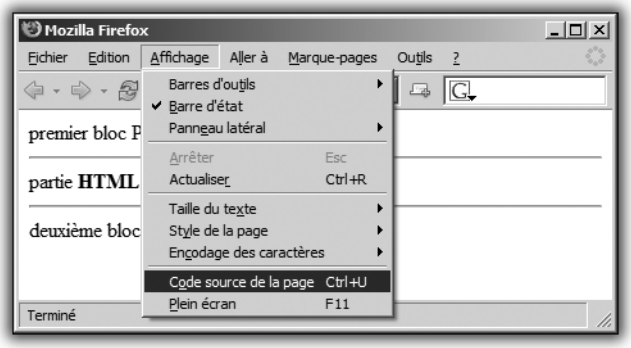

Figure 3.6 : *Sources*

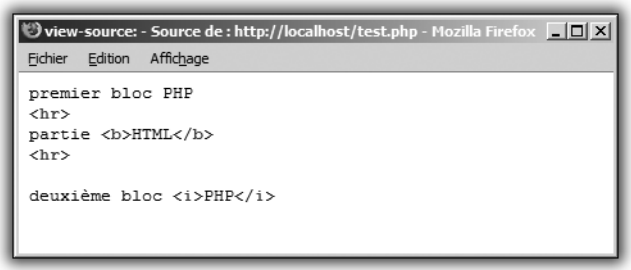

Figure 3.7 : *Les instructions PHP ont bien été interprétées*

3.2. Les commentaires

Un bloc de code peut contenir des commentaires. Ces derniers peuvent être ajoutés au code de différentes manières :

Listing 3-6 : Différentes syntaxes permettant d'insérer des commentaires

```
<?php

/* commentaires hérités
du C */

// commentaires hérités
// du C++

# commentaires hérités
# du SHELL

print("code commenté"); // commentaire de fin de ligne

?>
```

Ajouter des commentaires ne ralentit en rien l'exécution de votre code. Au moment de l'interprétation, PHP va tout simplement supprimer ces zones commentées.

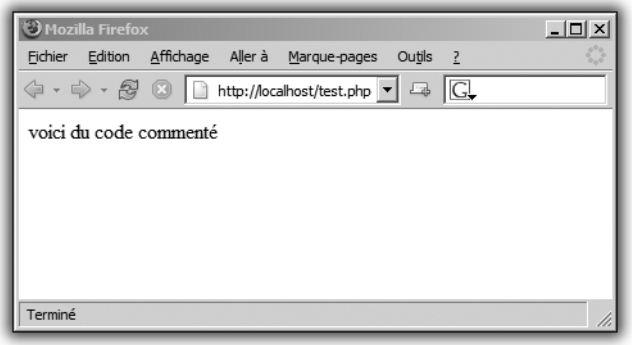

Figure 3.8 : *Les parties commentées n'apparaissent pas*

Poussons le vice un peu plus loin en générant un commentaire HTML à l'aide d'un script PHP. Les commentaires HTML doivent être placés entre <!— et —> :

Listing 3-7 : Commentaire HTML

```
<?php

print("voici un commentaire HTML : ");
print("<!-- message commenté -->");

?>
```

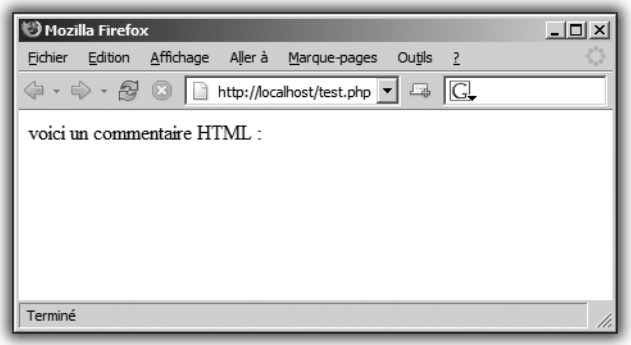

Figure 3.9 : *Code avec commentaires*

Bien qu'invisible dans le navigateur, le commentaire HTML est cependant bien présent dans les sources de la page.

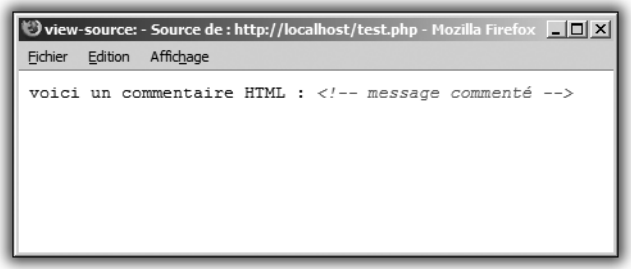

Figure 3.10 : *Sources*

3.3. Les variables

Quel que soit le langage de programmation, l'élément principal d'un programme est la variable. Une variable est un élément qui peut prendre différentes valeurs au cours de l'exécution d'un programme.

En PHP, les noms des variables sont précédés du caractère `$` : `$abc` correspond à la variable `abc`.

`$abc` est ici le nom de la variable, à ne pas confondre avec ce que contient la variable (`$abc` peut par exemple contenir la valeur `3`).

Précisément, pour donner la valeur `3` à la variable `$abc`, écrivez l'affectation suivante :

Listing 3-8 : Affectation de la valeur 3 à la variable $abc

```
$abc = 3;
```

Notation des exemples

Pour de tout petits exemples, comme celui-ci, ne correspondant en fait qu'à une sous-partie de script, les balises `<?php` et `?>` ne sont pas notées. Si vous voulez tester ce morceau de code, il convient bien évidemment de les ajouter.

Le contenu d'une variable peut être obtenu en y faisant simplement référence :

Listing 3-9 : Affectation de différentes valeurs

```
valeur de la variable abc =
<?php
print($abc);
?>

<br/>valeur de la variable abc =
<?php
$abc = 3;
print($abc)
?>

<br/>valeur de la variable abc =
<?php
$abc = 12;
print($abc)
?>
```

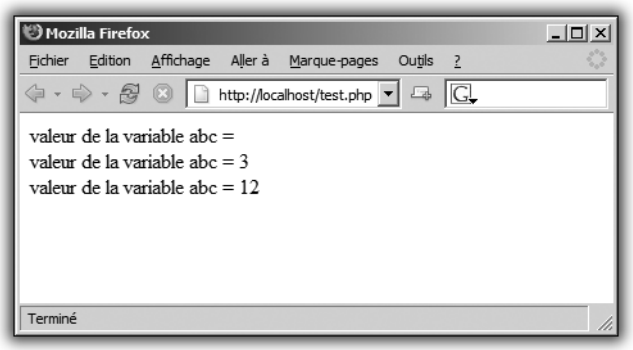

Figure 3.11 : *Différentes affectations de variables*

Vous constatez qu'avant qu'une valeur ne soit affectée à la variable, celle-ci n'existe pas. Elle peut ensuite prendre différentes valeurs lors de l'exécution du script.

Il faut aussi noter que la valeur d'une variable est conservée d'un bloc de code à l'autre.

La valeur d'une variable peut être affectée tout naturellement à une autre variable avec l'opérateur d'affectation = :

Listing 3-10 : La variable $abc contient la valeur de $d (2) à l'issue de l'affectation

```
<?php
$abc = 1;
$d = 2;
$abc = $d;
?>
```

Il existe différents moyens permettant d'affecter une valeur à une variable :

- soit directement : `$abc = 2;`;
- soit en lui faisant recevoir le résultat d'une opération : `$abc = 1 + 3;`;
- soit en lui faisant recevoir le résultat d'une fonction : `$abc = pow(2,4);` (c'est-à-dire la puissance 4 de 2).

Listing 3-11 : Différents types d'affectations

```
1- abc =
<?php
$abc = 2;
print($abc);
?>

<br/>2- abc =
<?php
$abc = 3 * 2;
print($abc)
?>

<br/>3- abc =
<?php
$abc = $abc + 1;
print($abc)
?>

<br/>4- abc =
<?php
```

```
$abc = pow(2,4);
print($abc)
?>

<br/>5- abc =
<?php
$abc = pow(2,4) - 7;
print($abc)
?>
```

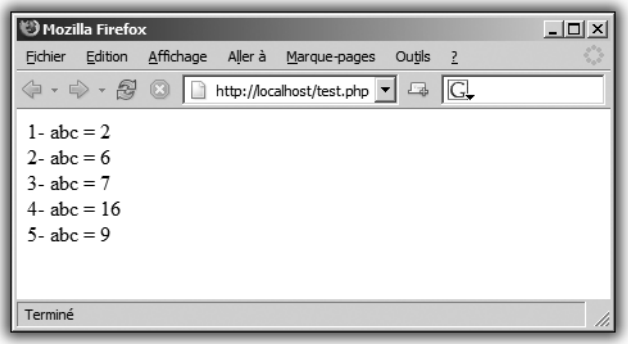

Figure 3.12 : *Différents types d'affectations*

L'instruction `$abc = $abc + 1` indique que la variable `$abc` est égale à la valeur de la variable `$abc` plus 1. Il est possible d'écrire plus succinctement cette affectation : `$abc++`.

L'opérateur ++ derrière une variable incrémente cette variable et inversement, l'opérateur –– décrémente la valeur.

```
abc =
<?php
$abc = 4;
$abc--;
print($abc);
?>
```

Certains noms de variables sont interdits, reportez-vous aux Annexes pour obtenir plus de précisions sur ce sujet.

Il existe une notation alternative et plus complexe qui permet d'avoir accès aux variables : `${'nom_de_la_variable'}`.

```
$x = 0;
${'x'} = 10; // assigne la valeur 10 à la variable x
print(${'x'}); // affiche : 10
```

```
print($x); // affiche : 10
$x = 5; // assigne 5 à la variable x
print(${"x"}); // affiche : 5

$y = "x";
print(${$y}); // affiche 5 car $x vaut 5
```

Ces deux notations sont donc complètement équivalentes ; la deuxième peut cependant se révéler très utile lorsque vous devez créer ou récupérer des variables à l'orthographe non standard ou des variables dont le nom est lui-même « variable ». L'instruction suivante est tout à fait valable, malgré la présence d'un espace dans le nom de la variable :

```
${"ma variable"} = "bonjour monde";
```

Majuscules et minuscules

Le nom des variables est sensible à la casse (*case sensitive*). Cela signifie que PHP fait une différence entre majuscules et minuscules. Une variable `$a` sera donc différente de la variable `$A` et pourra donc cohabiter avec la première en disposant de sa propre valeur.

3.4. Les constantes

Les constantes peuvent être assimilées à des variables dont le contenu ne peut être pas modifié durant l'exécution du programme. La déclaration d'une constante fait appel à la fonction `define()` dont le premier argument correspond au nom de la constante, et le second, à sa valeur.

Listing 3-12 : Création de la constante VERSION_SITE

```
define("VERSION_SITE","v3.1");
```

À la différence des variables, les constantes n'ont pas à être précédées du caractère `$`.

Listing 3-13 : Utilisation de constantes au sein d'un site

```
<?php

define("NOM_SITE","Cool Site");
print(NOM_SITE);

define("VERSION_SITE",3.1);
$a = VERSION_SITE + 1;

?>
```

PHP dispose d'un certain nombre de constantes définies par défaut. La constante `PHP_VERSION` contient par exemple la version de l'interpréteur PHP.

Listing 3-14 : Utilisation d'une constante par défaut

```
<?php

print("Version de PHP : ");
print(PHP_VERSION);

?>
```

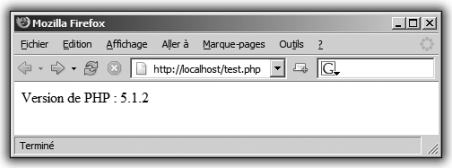

Figure 3.13 :
Version de PHP

L'exécution du script suivant vous permet d'obtenir la liste de toutes les constantes définies par défaut par PHP.

Listing 3-15 : Plus de 700 constantes définies par PHP

```
<pre>
<?php
print_r(get_defined_constants());
?>
</pre>
```

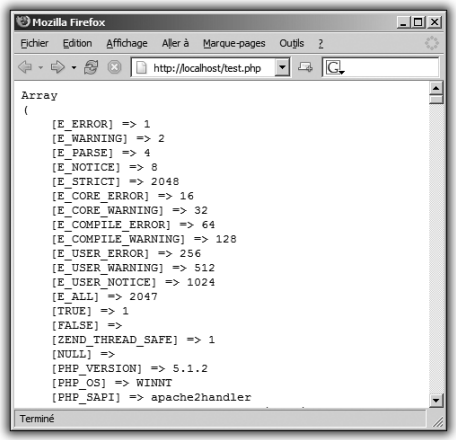

Figure 3.14 :
Plus de 700 constantes

Norme d'écriture

Les constantes sont généralement écrites en majuscule afin de les différencier des variables. Ce standard est partagé par une grande majorité des langages de programmation.

3.5. Les types de données

En informatique, les données sont souvent typées, en ce sens qu'elles contiennent une certaine catégorie d'information. Il existe globalement deux principaux types de données : les variables contenant des chiffres et les variables contenant des lettres.

Les tableaux, qui correspondent également à un type de donnée, sont présentés dans un chapitre suivant.

PHP rend les choses extrêmement simples au niveau des types car il n'impose pas d'associer explicitement un type à une variable (dans la plupart des autres langages, les variables doivent être associées à un type dès leur initialisation au début du programme). Si vous affectez `3` à `$abc`, celle-ci devient *de facto* de type numérique. Si en revanche vous lui affectez la phrase `"bonjour monde"`, elle devient ce que l'on appelle une « chaîne de caractères » (souvent appelée *string*).

Arrêtons-nous quelque temps sur ces deux principaux types de données.

Les données numériques

Vous trouvez, parmi les données numériques, les nombres entiers (`2`, `4`, `233`, `-12`) et les nombres flottants (`0.5`, `23.8`, `-123.4`).

Écriture des nombres flottants

Notez que le caractère séparant la partie entière de la partie décimale est le point. En utilisant une virgule, vous générez une erreur. Il s'agit de la norme anglo-saxonne qui est utilisée par tous les langages de programmation.

Il existe un certain nombre d'opérateurs mathématiques qui peuvent être utilisés avec les numériques :

Tableau 3.1 : *Les différents opérateurs mathématiques*	
Opérateur	**Rôle**
`+`	L'addition
`–`	La soustraction
`*`	La multiplication
`/`	La division
`%`	Le modulo

Voyez l'utilisation de tous ces opérateurs dans un même exemple :

Listing 3-16 : Différents opérateurs mathématiques

```
<?php
$a = 10 ;
$b = 5 ;
$c = 2;

$x = $a + $b; // la variable $x contient 15
$x = $x - $c ; // $x vaut 13
$x = $x * $x ; // $x vaut 169
$x = ($a / $b) + $c; // $x vaut 4
$x = 11 % $a ;
// $x vaut 1,
// le modulo correspond au reste de la division 11 / 10
?>
```

L'ordre de priorité des opérateurs doit être respecté : les opérations `* / %` sont traitées avant les opérations `+ –`.

L'expression `10 – 2 * 4` vaut ainsi `2` et non `32`. La multiplication est en effet réalisée avant la soustraction.

Quand PHP doit interpréter la ligne `$abc = 16 – 12 / 6 * 5 + 1;`, il évalue l'expression de la manière suivante :

```
16 - 2 * 5 + 1
16 - 10 + 1
6 + 1
7
```

Il peut donc être préférable d'utiliser des parenthèses pour limiter les risques :

```
$abc = 16 - ( ( 12 / 6 ) * 5 ) + 1 ;
```

Afin de gagner du temps, des formes compactes existent pour certaines opérations. Ainsi, `$abc = $abc + 2 ;` est équivalent à `$abc += 2 ;`

De la même manière, les raccourcis suivants sont tout à fait valides : `-=`, `*=`, `/=` et `%=`.

Il existe un type de numérique à part : les booléens. Seules deux valeurs booléennes existent : `true` (vrai), `false` (faux).

```
<?php
$var = true ; // $var contient la valeur booléenne true
?>
```

Il est courant que la valeur `0` soit équivalente à `false` et que la valeur `1` soit équivalente à `true`. Cet abus peut cependant se révéler dangereux lors de certains tests. Nous y ferons d'ailleurs mention plus loin.

Les chaînes de caractères

Une chaîne de caractères (souvent appelée *string*) doit être délimitée par des guillemets (`"`) ou des primes (`'`).

Les guillemets et les primes

Lorsque les caractères de délimitation sont des guillemets (`"`), la chaîne de caractères peut contenir des caractères ainsi que des variables. La variable est alors remplacée par son contenu.

Listing 3-17 : Une variable dans une chaîne de caractères

```
<?php
$n = 2;
$s = "valeur de la variable n = $n";
print($s); // affiche : "valeur de la variable n = 2"
?>
```

Quand les primes (`'`) sont utilisées comme caractères de délimitation, les variables ne sont plus remplacées par leur contenu :

```
<?php
$n = 2;
$s = 'valeur de la variable n = $n';
```

```
print($s); // affiche : 'valeur de la variable n = $n'
?>
```

Bien utiliser les signes

En utilisant la prime (`'`), PHP sait qu'il n'a pas à remplacer la présence d'une éventuelle variable par son contenu. Il affiche directement et sans traitement la chaîne de caractères. Il est donc préférable, pour optimiser votre code, d'utiliser les guillemets quand une chaîne est vierge de toute variable. Par exemple, `'bonjour monde'` est préférable à `"bonjour monde"`.

L'affichage d'une chaîne de caractères peut également être réalisé avec la fonction `echo()`. Les syntaxes suivantes pourront être trouvées indifféremment dans la littérature :

- `echo "bonjour";`
- `echo("bonjour");`
- `print("bonjour");`
- `print "bonjour";`

Syntaxe spéciale de `echo`

`echo` permet l'utilisation d'une syntaxe spéciale : `echo $var1, $var2;` pour afficher la `$var1` puis `$var2`. Aucune limitation n'existe au niveau du nombre de variables à afficher.

Échappement de caractères

Une question peut alors se poser : comment une chaîne de caractères peut contenir le guillemet (") quand ses délimiteurs sont précisément des guillemets ?

Il est dans ce cas nécessaire d'utiliser un caractère dit d'échappement : `\`. Ce caractère est nommé « barre oblique inversée » ou *antislash* (également *backslash*). Pour obtenir la phrase `bonjour " monde`, il est nécessaire d'écrire `"bonjour \" monde"`. Le principe est le même pour la prime (`'`) avec des strings délimitées par des primes :

Listing 3-18 : Échappement de caractères

```
print("bonjour \" monde"); // affiche : bonjour " monde
print('bonjour \' monde'); // affiche : bonjour ' monde
print('bonjour \" monde'); // affiche : bonjour \" monde
```

Les caractères (\) et ($) ont aussi besoin d'être précédés d'une barre oblique inversée pour être affichés dans une chaîne entre guillemets :

```
$i = 2 ;
print("\\ \$i vaut $i"); // affiche : "\ $i vaut 2"
```

L'opérateur de concaténation

Il existe un opérateur pour les chaînes de caractères : l'opérateur de concaténation. Il permet de réunir deux chaînes : `$str = $str1 . $str2 ;` et signifie que la variable `$str` contient la variable `$str1` suivie de `$str2`.

Listing 3-19 : L'opérateur de concaténation

```
<?php
$var1 = "ui";
$var2 = "le temps est " .  "beau aujourd'h" . $var1;
print($var2); // affiche : "le temps est beau aujourd'hui"
print("cou" . 'cou'); // affiche : "coucou"
?>
```

La version raccourcie existe aussi : `.=`.

`$abc = $abc . " coucou" ;` est l'équivalent de `$abc .= " coucou";`.

L'opérateur de concaténation est également très utile pour améliorer la lisibilité d'une chaîne de caractères très longue.

Listing 3-20 : $var1 et $var2 sont rigoureusement identiques

```
<?php
$var1 = "Les 7 péchés capitaux sont : la paresse,
✂ l'orgueil, la gourmandise, la luxure, l'avarice, la
✂ colère, l'envie.";
$var2 = "Les 7 péchés capitaux sont : ".
        "la paresse, l'orgueil, la gourmandise, la luxure, ".
        "l'avarice, la colère, l'envie.";
?>
```

En informatique, les chaînes de caractères sont souvent appelées *string*. Vous trouverez donc souvent dans les exemples suivants la variable `$str` pour définir une variable de type chaîne de caractères.

Appellation des variables

Il est important, en programmation, de donner des noms pertinents aux variables. Quand vous voulez qu'une variable contienne le nom d'une ville, il est préférable de nommer la variable `$ville` plutôt que `$x`. N'hésitez donc pas à donner des noms « longs » : `$portefeuille_client` est préférable à `$prtfcl`, même s'il peut paraître rébarbatif d'écrire une variable aussi longue. Comme nous l'avons précédemment dit, votre code devra peut-être être repris après plusieurs mois, et pas obligatoirement par vous !

Le type NULL

Ce type, composé d'un seul élément (`NULL`), permet d'indiquer qu'une variable n'a pas de valeur.

Listing 3-21 : Une fois utilisée, la variable $str est passée à NULL

```
<?php
$str = "coucou";
print($str);
$str = NULL;
?>
```

Changement de type

PHP a ceci de sympathique qu'il permet aux variables de changer de type si vous leur affectez des données de types différents au cours du programme :

Listing 3-22 : Changement de type

```
<?php
$abc = "1"; // $abc est une string contenant la chaîne "1"
$abc += 2; // $abc est maintenant un numérique contenant
✂ la valeur 3
?>
```

Dans cet exemple, la variable est d'abord une chaîne de caractères contenant la chaîne `"1"`. Après l'opération d'incrémentation, elle devient de type numérique et contient la valeur `3`.

PHP propose également une opération de transtypage pour convertir les types de données. Cette opération, qui porte également le nom de « *cast* », utilise la syntaxe suivante :

```
$s = (string) 1; // $s contient maintenant la chaîne "1"
$n = (int) $s; // $n contient 1
$b = (bool) $n; // $b contient TRUE
$n = (int) 1.8; // $n contient 1
$b = (bool) -3; // $b contient TRUE
$b = (bool) 0; // $b contient FALSE
$f = (float) " 1.45 "; // $f contient 1.45
```

Autres types

Le transtypage fonctionne également avec les autres types de données étudiés dans la suite de l'ouvrage : les tableaux (`array`) et les objets (`object`).

Au final, PHP permet de ne pas avoir à se tracasser avec les problèmes de typage.

3.6. Les structures de contrôle

Tous ces exemples s'exécutaient jusqu'à maintenant linéairement, ligne après ligne. Cependant, comme dans la vie courante, il est rare que les choses se passent aussi simplement : nous réalisons certaines choses uniquement dans certains cas et, dans d'autres circonstances, nous répétons les mêmes choses plusieurs fois. L'équivalent informatique de ces événements est la structure de contrôle.

Vous disposez, avec PHP, de toutes les structures de contrôle standard. Celles-ci sont généralement regroupées en deux catégories…

Les conditions :

```
SI test est vrai ALORS FAIRE action1 SINON FAIRE action2

FAIRE DANS LE CAS 1 action1, DANS LE CAS 2 action 2 etc.
```

Les boucles :

```
FAIRE action TANT QUE test est vrai

TANT QUE test est vrai ALORS FAIRE action

POUR TOUS LES CAS SUIVANTS FAIRE action
```

Les conditions

Nous présenterons dans cette partie les expressions `if… else`, `if… elseif… else` et `switch… case`.

IF ELSE

L'équivalent de `SI test est vrai ALORS FAIRE action1 SINON FAIRE action2` est l'instruction `if… else` (*if* signifie « si » et *else* signifie « ou bien »).

Écrivez un petit programme qui affiche `"i est plus grand que 5"` si la variable `$i` est plus grande que 5 et la phrase `"i est plus petit que 5"` dans le cas contraire.

Listing 3-23 : Le premier test

```
<?php

$i = 4;

if ($i > 5)
  print("i est plus grand que 5");
else
  print("i est plus petit que 5");

?>
```

Dans ce cas, `"i est plus petit que 5"` apparaît à l'écran car vous avez initialisé la variable `$i` à `4`.

Les tests sont réalisés avec des opérateurs de comparaison.

Tableau 3.2 : *Les opérateurs de comparaison*

Opérateur	Résultat
`$a == $b`	Vrai si `$a` est égal à `$b`
`$a === $b`	Vrai si `$a` est égal à `$b` et si ces deux variables sont de même type
`$a != $b`	Vrai si `$a` est différent de `$b`
`$a !== $b`	Vrai si `$a` est différent de `$b` (en type ou en valeur)
`$a > $b`	Vrai si `$a` est supérieur à `$b`

Tableau 3.2 : *Les opérateurs de comparaison*	
Opérateur	**Résultat**
`$a < $b`	Vrai si `$a` est inférieur à `$b`
`$a >= $b`	Vrai si `$a` est supérieur ou égal à `$b`
`$a <= $b`	Vrai si `$a` est inférieur ou égal à `$b`

Il est important de comprendre que ces opérateurs renvoient une valeur booléenne, `false` ou `true`, suivant le résultat de la comparaison. Le test est en fait réalisé sur cette valeur de retour.

Écrire `if ($a > $b)` est en fait la même chose qu'écrire `if (($a > $b) == true)`.

`if ($i)`

L'expression `if ($i)` est identique à `if ($i != NULL)`. Cette version condensée est très souvent utilisée en informatique.

Dans cet exemple, une seule action est réalisée, dans un cas (`if`) comme dans l'autre (`else`). Si plusieurs actions doivent être réalisées, celles-ci sont groupées entre des balises `{ }` pour former un groupe d'instructions.

Modifiez votre script afin qu'il affiche deux lignes :

```
<?php

$i = 4;

if ($i > 5)
{
  print("nous sommes dans le IF ... ");
  print("i est plus grand que 5");
}
else
{
print("nous sommes dans le ELSE ... ");
  print("i est plus petit que 5");
}
?>
```

Les balises `{ }`
Alors qu'elles sont facultatives quand il n'y a qu'une instruction, elles deviennent obligatoires dès qu'il y en a au moins deux.

Les tests peuvent se faire aussi bien sur des numériques :

```
if ($prix == 3.5)
```

… que sur des chaînes de caractères :

```
if ($prenom == "paul")
```

Attention, cependant, à ne pas se méprendre sur les opérateurs de supériorité et d'infériorité avec des chaînes de caractères !

Ainsi, le test `if ($code > "9AEXB3")` n'est pas un test sur les longueurs respectives des deux chaînes. Il s'agit plutôt de comparer une à une les valeurs ASCII des caractères composant les deux chaînes. Chaque caractère possède en informatique un code ASCII. Celui du caractère `'a'` est par exemple 97 et celui de `'9'` est 57 (la fonction `ord()` peut être utilisée pour trouver la valeur ASCII d'un caractère : `ord("A")` retourne 98). De ce fait, `"abc"` est supérieur à `"9AEXB3"` (car `ord('a')` est supérieur à `ord('9')`) et `"def"` est supérieur à `"dc"` (car `ord('e')` est supérieur à `ord('c')`).

Pour l'instant, les tests étudiés sont simples car vous ne testez qu'une seule valeur. Si vous devez écrire le test suivant :

```
SI la personne a plus de 18 ans ALORS écrire "homme
majeur" SI elle est de sexe masculin
```

… vous écrirez, dans l'état de vos connaissances actuelles :

```
<?php

$age = 22;
$sexe = "masculin";

if ($age > 18) {
  if ($sexe == "masculin")
    print("homme majeur");
}

?>
```

La façon logique d'exprimer ce test est :

```
SI la personne a plus de 18 ans ET qu'elle est de sexe
masculin ALORS écrire "homme majeur"
```

Il s'agit là d'une double condition. L'opérateur (`&&`) peut être utilisé pour signifier l'opérateur ET :

```
<?php

$age = 22;
$sexe = "masculin";

if (($age > 18) && ($sexe == "masculin"))
  print("homme majeur");

// affiche bien "homme majeur"
// car : (22 > 18) ET (masculin == masculin)

?>
```

L'opérateur (`&&`) est appelé un « opérateur logique ». Il en existe d'autres qui vont vous permettre de réaliser des tests plus ardus.

Tableau 3.3 : *Opérateurs logiques*

Opérateur	Résultat
`$a and $b`	Vrai si `$a` ET `$b` sont vraies
`$a && $b`	Vrai si `$a` ET `$b` sont vraies (identique à and)
`$a or $b`	Vrai si `$a` OU `$b` sont vraies
`$a || $b`	Vrai si `$a` OU `$b` sont vraies (identique à or)
`$a xor $b`	Vrai si `$a` OU `$b` sont vraies, mais pas les deux
`! $a`	Vrai si `$a` est fausse

À partir de ces opérateurs logiques, il est possible d'écrire le test suivant :

```
SI la variable $a OU la variable $b est supérieure à 4,
ET que la variable $c n'est pas inférieure ou égale à 2
ALORS écrire OK
```

… qui devient :

```
if (($a > 4) or ($b > 4)) and !($c <= 2)) print("OK");
```

Il est souvent possible de simplifier une expression : `!($c <= 2)` équivaut à `($c > 2)`. En effet, dire que `$c` n'est pas inférieur ou égal à `2` est une façon complexe de dire que `$c` est supérieur à `2`.

Il est néanmoins préférable généralement, au début, de transcrire terme à terme une condition plutôt que d'essayer de la simplifier à outrance.

Un bon réflexe consiste aussi à insister sur les parenthèses afin de regrouper ensemble les conditions interdépendantes. Les priorités, comme pour les opérateurs mathématiques, peuvent faire des ravages. Le test suivant :

```
($b > 3) || ($c > 3) && ($c < 10)
```

… ne signifie pas :

```
$b OU $c supérieures à 3 ET $c inférieure à 10
```

… mais :

```
$b supérieure à 3 OU $c comprise entre 3 ET 10
```

Dans cette mesure, il est préférable d'écrire en utilisant une paire de parenthèses supplémentaires pour ne rien risquer :

```
($b > 3) || (($c > 3) && ($c < 10))
```

Les annexes contiennent un tableau présentant l'ordre de priorité des différents opérateurs.

IF… ELSEIF…

Essayez maintenant d'écrire un autre test :

```
si la couleur est rouge, jaune ou bleue, écrire "primaire"
si la couleur est noire, écrire "noire"
si la couleur est blanche, écrire "blanche"
sinon écrire "mélange"
```

Votre première proposition pourrait être celle-ci :

```
if (($couleur=="rouge") || ($couleur=="jaune") ||
    ($couleur=="bleue"))
  print("primaire");
```

```
if ($couleur == "noire") print("noire");

if ($couleur == "blanche") print("blanche");
else print("mélange");

}
```

Celle-ci serait évidemment complètement erronée car, si vous supposez que la « couleur » est `rouge`, deux messages seraient alors affichés : `"primaire"` et `"mélange"`.

Vous êtes donc obligé d'emboîter les `if… else` :

```
if (($couleur=="rouge") || ($couleur=="jaune") ||
   ($couleur=="bleue"))
  print("primaire");
else
{
  if ($couleur == "noire") print("noire");
    else
  if ($couleur == "blanche") print("blanche");
    else print("mélange");
}
```

Il s'agit en fait d'un SI… SINON SI… SINON SI… SINON. PHP propose, pour ce genre de cas, la méthode suivante : `if… elseif… elseif… else`.

```
<?php

if (($couleur=="rouge") || ($couleur=="jaune") ||
    ($couleur=="bleue"))
  print("primaire");
elseif ($couleur == "noire")
  print("noire");
elseif ($couleur == "blanche")
  print("blanche");
else
  print("mélange");

?>
```

SWITCH

Supposez désormais que vous deviez écrire le test suivant :

```
si la couleur est rouge, écrire "R"
```

```
si la couleur est bleue, écrire "B"

si la couleur est jaune, écrire "J"

sinon écrire "?"
```

Dans ce cas, il est dommage d'utiliser un `if… elseif…`. Imaginez, en effet, que la couleur soit jaune : il faudrait alors tester si la couleur est rouge, puis tester si la couleur est bleue et enfin tester si la couleur est jaune. Dans ce type de cas, où l'action à réaliser ne dépend que de la valeur d'une variable, il est préférable d'utiliser le `switch` :

```
switch ($couleur)
{
  case "rouge":
    print("R");
    break;
  case "bleue":
    print("B");
    break;
  case "jaune":
    print("J");
    break;
  default:
    print("?");
    break;
}
```

Il peut y avoir autant de `case` que nécessaire, mais il ne peut y avoir qu'un seul `default` (il n'est cependant pas obligatoire).

Un `case` peut contenir plusieurs instructions ; il faut alors les placer entre un `case` et un `break` :

```
switch ($couleur)
{
  case "rouge":
    print("couleur : ");
    print("R");
    break;
  case "bleue":
etc.
```

Les boucles

Un des principaux avantages de l'informatique, en général, est de permettre d'automatiser des tâches fastidieuses. Imaginez que vous vouliez écrire tous les entiers inférieurs à 5.

Il serait dommage de devoir écrire :

```
print("0<br/>");
print("1<br/>");
print("2<br/>");
print("3<br/>");
print("4<br/>");
```

L'informatique a créé, pour ce genre de besoin, la notion de boucle.

WHILE

`WHILE` est la boucle qui permet d'écrire ceci :

```
TANT QUE test vrai FAIRE action
```

À partir de là, il devient évident pour cet exemple que le test va être réalisé sur une variable (`$i < 5`) et que l'action va consister à l'afficher et à l'incrémenter :

```
$i = 0;
while ($i < 5)
{
  print("$i<br/>");
  $i++;
}
print("<br/>fin de la boucle");
```

Que se passe t-il au niveau de l'interpréteur PHP :

- la variable `$i` prend la valeur `0`
- nous entrons dans la boucle
- est-ce que `$i < 5` ? oui : le bloc d'expressions du `while` est alors exécuté
- affichage de `$i` (`0`) et augmentation de `1` de la valeur de `$i`
- nous remontons alors au niveau du test du `while`
- `$i` est-elle inférieure à `5` ? oui (elle vaut `1`), alors on exécute le bloc
- affichage de `$i` (`1`) et incrémentation de `$i`…
- nous continuons ainsi jusqu'au moment où nous passons la valeur de `$i` à `5`
- nous remontons alors au niveau du test, `$i` n'est plus strictement inférieure à `5`

— nous sortons alors de la boucle et passons à l'instruction suivant le bloc du `while`, c'est-à-dire l'affichage de "*fin de la boucle*"

Le danger du `while`

Faites attention, quand vous travaillez avec une boucle `while`, à ne pas vous retrouver dans une boucle infinie. Cela aurait été le cas, si vous aviez oublié la ligne `$i++`. En effet, la variable aurait gardé la valeur `0` et la boucle `while` aurait affiché `"0"` indéfiniment.

Comme pour les conditions, le test peut être complexe :

```
$i = 1;
while (($n != 121) && ($i < 100))
{
  $n = $i * $i;
  print("i = $i , n= $n<br/>");
  $i++;
}
```

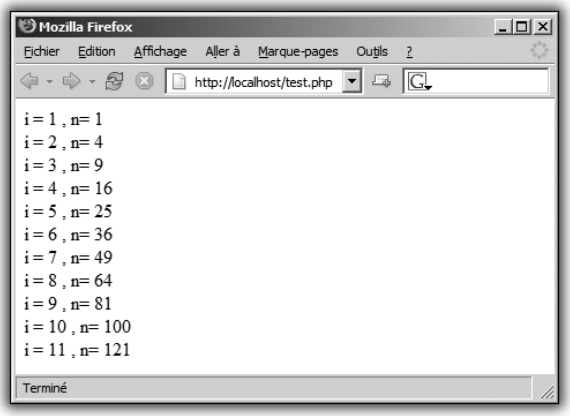

Figure 3.15 : *Boucle while*

Dans cet exemple, la condition d'arrêt de la boucle est double : la variable `$n` doit être différente de `121` et la variable `$i` doit être inférieure à `100`.

DO... WHILE

Une variante existe au `while` : le `do... while`. Il ne s'agit plus d'un `tant que... faire`, mais d'un `faire... tant que`. Dans ce cas, le code de la boucle est donc au moins exécuté une fois :

```
$i = 6;
do
{
  print($i);
}
while ($i < 5);
```

Bien que `$i` contienne la valeur `6` et que le test soit `$i < 5`, le code est bien exécuté une fois.

FOR

La syntaxe de la boucle `for` est la suivante :

```
for (expression 1; expression 2; expression 3)
```

- `expression 1` est l'instruction initiale exécutée avant la première itération ;
- `expression 2` est le test réalisé au début pour chaque itération ;
- `expression 3` est l'instruction exécutée à la fin de chaque itération.

Ainsi, l'équivalent de cette boucle `while` :

```
$i = 0;
while ($i <= 10)
{
  print("$i <br/>");
  $i++;
}
```

… est la boucle `for` suivante :

```
for ($i = 0; $i <= 10; $i++)
{
  print("$i <br/>");
}
```

Vous souhaitez maintenant écrire les nombres pairs inférieurs à 50 :

```
for ($i = 0; $i <= 50; $i = $i + 2)
{
  print("$i <br/>");
}
```

La boucle `for` permet donc de condenser les principaux éléments d'une boucle sur une seule et même ligne.

Toutes les parties de la boucle `for` peuvent être vides. Ainsi, les deux boucles suivantes sont équivalentes :

Listing 3-24 : Version 1

```
for ($i = 0; $i <= 10; $i++)
{
  print("$i <br/>");
}
```

Listing 3-25 : Version 2

```
$i = 0;
for (;$i <= 10;)
{
  print("$i <br/>");
  $i++;
}
```

L'intérêt de cette notation alambiquée est cependant assez mince.

BREAK et CONTINUE

Il est parfois nécessaire de terminer une boucle en plein milieu de son exécution ; l'instruction `break` est alors utilisée.

La boucle suivante permet d'arrêter l'exécution du `for` dès que la variable d'incrémentation passe à la valeur 7.

Listing 3-26 : Le break nous permet de sortir de la boucle

```
for ($i=0;$i<=10;$i++) {
  if ($i==7) {
    print("<i>i contient 7, nous sortons de la
    ✂ boucle.</i>");
    print("<br/><br/>");
    break;
  }
  print("$i<br/>");
}
print("<b>fin de la boucle.</b>");
```

(voir Figure 3.16)

Le cousin du `break` est le `continue`. Il permet de forcer le passage à l'itération suivante à n'importe quel niveau d'une boucle.

L'instruction `continue` permet dans l'exemple suivant de sauter l'affichage de la valeur 7 :

Figure 3.16 : *Sortie en plein milieu de boucle*

Listing 3-27 : Nous sautons le traitement de la valeur 7

```
for ($i=0;$i<=10;$i++) {
  if ($i==7) {
    print("<i>i contient 7, passage à l'itération
    ✂ suivante.</i>");
    print("<br/>");
    continue;
  }
  print("$i<br/>");
}
print("<br/><b>fin de la boucle.</b>");
```

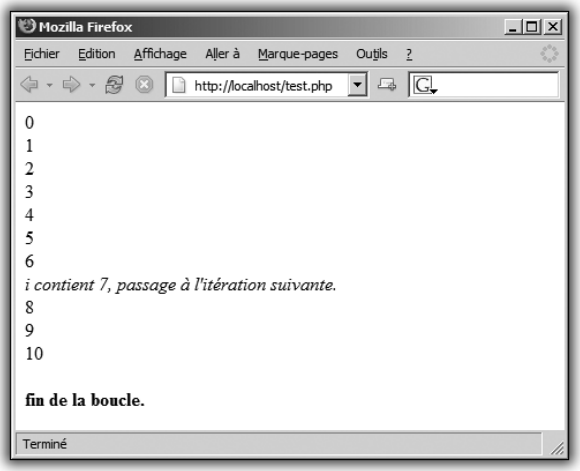

Figure 3.17 : *Instruction continue*

Ces deux instructions peuvent aussi bien être utilisées dans le cadre d'un `for`, d'un `while` ou d'un `switch`.

3.7. Organisation du code

PHP propose différentes méthodes pour organiser vos développements : la définition de fonctions utilisateur et l'inclusion de fichiers. Ces méthodes visent avant à tout à :

- améliorer la lisibilité de vos sources ;
- éviter la duplication de code.

Les objets, qui permettent également de mieux architecturer votre code, sont présentés dans un chapitre à part.

Les fonctions

Vous savez désormais que PHP permet de faire appel à des fonctions. Les exemples précédents vous ont ainsi permis d'illustrer l'utilisation de la fonction `print()` dont le rôle est d'afficher les données qui lui sont transmises.

La plupart des fonctions reçoivent des paramètres et retournent une valeur qui en dépend : on parle d'« arguments d'une fonction ». Par exemple, la fonction `print()` prend comme paramètre une chaîne de caractères ou un chiffre. Sa signature est donc celle-ci : `print($str)`.

La fonction de puissance prend quant à elle, deux arguments, à savoir le nombre et la puissance à laquelle il doit être élevé : `pow($n,$p)`.

```
$resultat = pow(2,4);
// $resultat contient 16
// qui correspond bien à 2 exposant 4
```

Ces deux fonctions, comme plusieurs centaines d'autres, sont incluses par défaut dans PHP. Vous pouvez y faire appel à n'importe quel moment dans votre code.

En plus de cette « bibliothèque » de fonctions, PHP vous permet également de définir vos propres fonctions.

Définir une fonction

Supposez que vous vouliez qu'une fonction renvoie le double d'une valeur. Il faut tout d'abord lui donner un nom, `double()`, et décider de combien d'arguments elle a besoin ; a priori, un seul. La syntaxe pour déclarer une fonction en PHP est la suivante :

```
function double($n)
{
  $resultat = $n * 2;
  return $resultat;
}
```

Nommer l'argument `$n` n'est pas du tout obligatoire. Vous auriez tout à fait pu écrire la fonction ainsi :

```
function double($nimportequoi)
{
  $resultat = $nimportequoi * 2;
  return $resultat;
}
```

Il est cependant souvent intéressant de donner un nom d'argument en rapport avec le type de l'argument. Ainsi, si la fonction nécessite comme paramètre un booléen, il peut être judicieux de choisir `$bool` ; et s'il s'agit une chaîne de caractères, `$str` ou `$ch` peuvent être des bons choix.

Dans la majorité des cas, votre fonction retournera une valeur. Ceci est réalisé avec l'instruction `return` :

```
return $resultat;
```

L'instruction `return`

L'instruction `return` met fin à la fonction en renvoyant un résultat relatif à la donnée qui lui est passée en paramètre. Une fonction contenant des conditions peut tout à fait avoir différentes instructions `return`.

Une fois votre fonction déclarée dans votre script PHP, vous pouvez l'appeler de n'importe quel endroit et autant de fois que vous le voulez :

```
function double($n)
{
  $resultat = $n * 2;
  return $resultat;
}
```