PHP

Cours d'initiation à PHP
Niveau : BTS
Prérequis : HTML
Algorthmie
notions de SQL

1 Introduction

1.1 L'histoire de PHP

Aujourd'hui (début 2009), 2 millions de serveurs utilisent PHP pour abriter 50 millions de sites web. Il doit y avoir une raison ...

L'histoire commence en 1994 :

Un étudiant danois, Rasmis Lerdorf, qui venait de mettre son CV en ligne sur son site, s'est demandé comment compter le nombre de visites ...

Il à écrit une série de scripts CGI mis sur le serveur de son hébergeur, qui incrémentaient une base SQL.

Comme de nombreux internautes lui demandèrent ce programme, Rasmus Lerdorf mit en ligne en 1995 la première version de ce programme qu'il baptisa ***P****ersonal* ***H****ome* ***P****age v1.0.*

Devant le succès de PHP 1.0, Rasmus Lerdorf décida d'améliorer ce langage en y intégrant des structures plus avancées telles que des boucles, des structures conditionnelles, et y intégra un package permettant d'interpréter les formulaires qu'il avait développé (*FI*, Form Interpreter) ainsi que le support de mSQL. C'est de cette façon que la version 2 du langage, baptisée pour l'occasion *PHP/FI v 2*, vit le jour en 1995. Le projet s'internationalisa et de nombreux développeurs prirent part au projet. PHP fut rapidement utilisé sur de nombreux sites (15.000 fin 1996, puis 50.000 en mi 1997).

En 1997, le projet passa sous la direction de deux etudiants Israéliens : Zeev Suraski et Andi Gurmans, qui décidèrent, pour mettre au point PHP 3, de tout réécrire. C'est PHP 3.0 en 1998. Le sens de l'acronyme PHP à d'ailleurs changé à cette date. Il signifie maintenant ***P****HP* ***H****ypertext* ***P****rocessor*.

Suraski et Gurmans choisirent d'accélérer encore PHP pour la V4 en changeant le moteur du systéme : Plutôt que d'interpréter chaque instruction une à une, il fut décidé de « compiler » l'ensemble du script. C'est le Zend Engine, cœur de la version PHP 4 sortie en 2000.

La version 5 (2004) met l'accent sur la programmation objet et les services Web.

Une version 6 est, bien sur, en développement.

Zeev Suraski à fondé la société Zend Technologies qui fourni des services payant autour du langage PHP.

Zend Encoder (Rend les scripts illisibles, mais toujours portables)

Zend accelerator (jusqu'à + 300% de performance)

Zend IDE (Environnement de développement)

Il est à noter que la société Zend, à but commercial, emploie une dizaine de développeurs qui travaillent à plein temps pour améliorer PHP qui reste lui, bien sur, gratuit.

1.2 vocabulaire

1.2.1 Interprété / compilé

Les lignes de programmation que l'on écrit ne sont pas compréhensibles par la machine. Il faut passer par une traduction « Langage de programmation / Langage ordinateur ». (De même que si vous vouliez expliquer votre programme à votre grand-père, il vous faudrait passer par une traduction « Français / Langage de programmation »).

Soit cette traduction est faite « à l'avance » par un programme appelé compilateur, soit elle est faite « au moment de l'exécution » par un programme appelé interpréteur.

Le PHP est dans la seconde catégorie.

Avantage du compilé :

Traduction faite « une fois pour toute »

Inconveignant du compilé :

Traduction faite à destination d'un seul systéme (Windows 32 bits, Mac OS X, Sun, ...)

Il faut donc une compilation par plateforme.

Avantage de l'interprété :

C'est le même fichier qui va fonctionner sur chaque plateforme. C'est la portabilitée.

Inconveignant de l'interprété

Le système doit faire 2 choses : Traduire le script puis l'exécuter. Il est donc forcément plus lent.

Le programme est fourni sous forme de sources en clair... Le savoir faire du concepteur n'est pas protégé.

1.2.2 Script

On appelle Script un fichier écrit dans un langage interprété.

Comme il n'y a pas d'un coté un un fichier source et de l'autre un fichier exécutable, on utilise le terme script pour laisser entendre que ce fichier est aussi un programme exécutable.

1.2.3 LAMP

Ce sont les initiales de **L**inux **A**pache **M**ySQL **P**HP

Ces 4 produits forment un environnement complet pour faire fonctionner un site web :

- Le système d'exploitation : Linux
- Le serveur HTTP : Apache
- Le système de gestion des bases de donnée : MySQL
- Le langage de programmation : PHP

Ces 4 produits sont Open Source, donc gratuits. Ils disposent d'une très large communauté de développeurs (Forums d'aide, Scripts réutilisables, ...).

De plus ils forment aujourd'hui, et de loin, l'environnement le plus performant et le plus sur (les rares failles de sécurité on toujours été corrigées dans la journée).

1.2.4 Open Source

L'open source est un mouvement planétaire qui regroupe ... ceux qui veulent y participer.

Il à pour principe fondateur la mise à disposition des sources de logiciels. Ainsi, alors que l'ASP

n'est développé que par Microsoft, PHP est un projet sur lequel travaille des centaines d'étudiants, de chercheurs, d'ingénieurs et de passionnés à travers le monde. En disposant des sources, tout un chacun peut étudier la manière avec laquelle le langage est conçu et peut ainsi corriger d'éventuels bugs. Cela explique directement que PHP soit extrêmement stable et ne souffre que de très rares bugs ou failles.

Appartenant à tout le monde en même temps, les logiciels Open Source ont un énorme avantage sur les logiciels propriétaires : Ils ne peuvent pas disparaître.

Un autre avantage à travailler avec des logiciels Open Source tel que PHP est d'avoir à sa disposition une énorme bibliothèque de scripts dont les sources peuvent être récupérés gratuitement sur le Web. Il est quand même très rare de ne pas trouver sur le web un morceau de code qui répond exactement à votre besoin ...

1.2.5 Sécurité

PHP est sur. C'est à dire que les très rares failles de sécurité (Buffer overflow, Injection SQL, ...) ont été corrigées.

Ce qui ne veut pas dire que le serveur qui héberge votre site web est correctement mis à jour.

Ce qui veut encore moins dire que le site web que vous allez développer est sur ...

1.3 PHP A quoi ça sert ?

Essayez d'afficher l'heure sur une page HTML, et vous aurez la réponse ...

PHP sert à faire des pages dynamiques. C'est à dire à avoir un affichage personalisé.

Imaginez un forum de discutions ...

Il est impensable que le concepteur du site prévoie une page HTML par message posté. Il s'agit bien sur d'un programme qui va concevoir la page en fonction de la demande de l'utilisateur.

Il est fort probable que ce programme soit écrit en PHP...

PHP n'est pas non plus un langage universel. Il est par exemple quasiment impossible de gérer une interface avec PHP.

Il est conçu pour le développement Web.

1.3.1 Logiciels en ligne

Une tendance lourde du marché est le retour à des applications fonctionnant de manière centralisée.

C'est à dire avec des codes sources et une base de donnée se trouvant à un seul endroit, mais accessible facilement depuis n'importe ou.

L'architecture Web se prête à merveille à ce cahier des charges.

Les langages comme PHP servent donc à développer des sites Web, mais de plus en plus à réaliser des applications professionnelles « en ligne ». Que ce soit sur Internet ou sur un réseau d'entreprise.

Avantages :

Applicatif centralisé : Maintenance facilitée (Installation, mise à jour, sauvegardes, sécurité)

Installation aisée : Un navigateur suffit

Données centralisées : Travail collaboratif

1.4 PHP et la concurrence

PHP n'est pas le seul langage de programmation « web ». C'est aujourd'hui le plus rapide et l'un des plus puissants. Mais en informatique les choses évoluent vite...

La syntaxe de PHP est proche de celle du C, (et donc du C++) : C'est ce qui à permis à de nombreux développeurs de venir au PHP, et qui à en partie, contribué à son sucés.

Il y a donc des chances que son successeur reprenne tout ou partie de sa syntaxe.

Parmi les concurrents de PHP citons :

1.4.1 Ruby

Ruby est un langage de programmation interprété orienté objet.

Avantages :

Ruby est Open Source et gratuit.

Ruby est fortement orienté objet :

- toute donnée est un objet, y compris les types ;
- toute fonction est une méthode ;
- toute variable est une référence à un objet.

Cette structuration du langage offre une énorme puissance de développement.

La plateforme RAILS permet de faire fonctionner Ruby en AJAX. Cette technologie est peut-être la prochaine évolution des développements Web.

Inconvénients :

Assez récent, encore peu répandu chez les hébergeurs.

1.4.2 Python

Python est un langage de programmation interprété orienté objet.

Avantages :

Python est sous licence GPL

Python est extrêmement bien pensé et lisible (Les blocs sont défini par l'indentation) ce qui permet une qualité de développement objet supérieure à celle du PHP.

Inconvénients :

Le web est loin d'être la priorité des concepteurs ...

1.4.3 ASP

ASP (Active Server Pages) est un langage de programmation WEB propriété de Microsoft

Avantages :

Compatible avec les outils Microsoft (Front Page, Visual Studio, SQL Server, ...)

Partie client très ergonomique

Inconvénients :

Faible performances

Obligatoirement exécute avec IIS, le serveur HTTP de microsoft. Hors IIS (Internet Information Services) est une des plus grosses passoires de tout les temps ! (Cf Nimda, Red Code, ...)

Cher ! On est chez Microsoft, donc tout est payant : (ASP, Front Page, Visual Studio, SQL Server, ...) Mais aussi l'hébergement (Soit via les licences IIS, soit via l'investissement dans une

machine assez musclée pour supporter la bête en cas hébergement interne)

1.4.4 ColdFusion

ColdFusion est un langage propriétaire (Macromédia) utilisé pour développer des applications Web

Avantages :

Environnement de développement très ergonomique et complet (Code source, pages web, sereur FTP, ...)

Simplicité d'utilisation, niveau grand public

Inconvénients :

Solution propriétaire, donc payante

Peu d'hébergeurs le proposent, donc payant à nouveau

Performances moyenne

Langage lourd, et très mal conçu dés que l'on sort des cadres prévu.

1.5 Quelques rappels sur le fonctionnement d'internet

1.5.1 Protocoles

Internet, c'est des ordinateurs reliés entre eux et qui parlent un langage commun (ou protocole). Tout ces protocoles ont en commun un langage de bas niveau (une sorte de grammaire commune) : TCP/IP (Transmission Control Protocol / Internet Protocol)

Internet est une invention d'origine militaire. Dans le années 60, pendant la guerre froide, les militaires voulaient pouvoir faire communiquer entre eux des ordinateurs, même si une partie des postes qui composent le réseau se trouvent détruits.

D'où l'idée directrice de TCP/IP : L'information à faire circuler est découpée en paquets, et chacun de ces paquets est acheminée de façon autonome sans être obligé de passer par la même route. D'où le nom de Web (Toile d'araignée).

Quelques exemples de protocoles Internet :

Protocole	Signification	Fonction
HTTP	Hypertext Transfer Protocol	Le web
FTP	File Transfert Protocol	Transfert de fichiers
IRC	Internet Relay Chat	Dialogue en direct
NNTP	Network News Transfert Protocol	Envoi & lecture de news
POP	Post Office Protocol	Récupération des mails
SMTP	Simple Mail Transfert Protocol	Envoi des mails
BitTorrent		Transfert de données Poste à poste

1.5.2 URL

Une URL (Uniform Resource Locator, littéralement « localisateur uniforme de ressource ») est une chaine de caractère qui adresse de façon unique une ressource d'Internet

http://Jojo:lApIn@wWw.exEmplE.cOm:8888/chemin/d/acc%C3%A8s.php?q=req&q2=req2#signet

1 Schéma de représentation (Facultatif, les navigateurs ajoutent http://)

1.1 **http:** - protocole de communication + :

1.2 **//** - Séparateur, pour certains protocoles (http, ftp, edk, ...)

2 Localisation de l'hébergeur

2.1 **Jojo:lApIn@** Données d'authentification (Facultative et déconseillées, car non sécurisé) sous la forme : nom d'utilisateur + : + Mot de passe + @

2.2 **wWw.exEmplE.cOm** - nom de domaine : Obligatoire sous sa forme de nom (Non sensible à la case, ou sous forme d'adresse IP directement)
Le www est une convention, pas une obligation. Il ne correspond qu'a un sous répertoire chez votre hébergeur.

2.3 **:8888** Numéro de port TCP/IP (Facultatif, 80 par défaut pour http, 21 pour ftp, ...)

3 Localisation de la ressource (Sensible à la case, selon hébergement)

3.1 **/chemin/d/** - Obligatoire pour les services à chemin d'accès (/ par défaut : /).

3.2 **acc%C3%A8s.php** - nom de la page Web, optionnel (index.php ou index.htm par défaut); on remarque qu'un caractère non ASCII comme « è » est codé en « %C3%A8 » (Unicode UTF-8). L'extension n'à de signification que pour le serveur (interprétation php, ...)

4 Données supplémentaires

4.1 **?** - caractère de séparation : indique que des données suivent.

4.2 **q=req&q2=req2** - chaîne de requête, traitée par la page Web sur le serveur

4.3 **#** - caractère de séparation : indiquer une balise.

4.4 **signet** – identifie la balise, il s'agit d'un emplacement à l'intérieur de la page Web, cette donnée sera traitée par le navigateur Web.

Quelques exemples :

- URL de Wikipédia :
  - http://fr.wikipedia.org/
- La même, version courte :
  - fr.wikipedia.org
- URL d'une page sur Wikipédia :
  - http://fr.wikipedia.org/wiki/Rep%C3%A8re_uniforme_de_ressource
- URL d'un fichier sur un site FTP :
  - ftp://ftp.rfc-editor.org/in-notes/rfc2396.txt
- URL d'un lien (mailto) vers une adresse courriel :
  - mailto:Quidam.no-spam@exemple.com
- URL d'un forum de discussion de Usenet :
  - news:fr.comp.infosystemes.www.auteurs
- URL d'un site Gopher :
  - gopher://gopher.quux.org/
- URL d'un fichier partagé sur eMule :
  - ed2k://|file|Planet.Terror.FRENCH.DVDRiP.XviD.avi|731607040|b50b55bc0bfa1f4de1c6e9cd858d7b5c|/

1.6 Principe de fonctionnement Client / Serveur

Que se passe t-il une fois arrivé sur le serveur.

Celui ci trouve le fichier demandé par l'utilisateur.(acces.php) dans l'exemple précédent.

Si l'extension du fichier l'indique, le fichier est interprété par un programme dit Middleware (Traduction : Truc qu'on sait pas exactement ou ça se trouve) Exemple : Zend

C'est le résultat de l'exécution de acces.php qui sera envoyé à l'utilisateur et non le script originel. (Bien que le résultat soit envoyé sous le même nom : acces.php)

Seront envoyé avec tout les fichiers liés (comme les images présentes dans la page, les éventuels fichiers css, ...)

C'est le navigateur du poste client qui se charge de mettre en forme le code reçu et de l'afficher.

Ce mode de fonctionnement à plusieurs conséquences :

1.6.1 : On dispose du code de ce que l'on affiche

Tout ce qu'un navigateur affiche c'est qu'il l'a reçu par http. Dit autrement, si vous pouvez l'afficher, vous pouvez le récupérer (sauf la partie PHP).

C'est le bonheur des développeurs, et le malheur des éditeurs...

1.6.2 : Le script PHP est sécurisé

Le script php n'est jamais envoyé à l'utilisateur. Ce n'est que le résultat de son exécution qui est affiché par les navigateurs.

On peut donc écrire en dur des mots de passe dans du code PHP : ce n'est pas une faille de sécurité.

De même, les commentaires PHP restent visibles des seuls développeurs.

1.6.3 : PHP n'agit qu'au niveau du serveur

Ce que vous envoyez au serveur, c'est du HTML (Ou du XHTML, ou du CSS, ...) mais dans tout les cas, c'est du texte (et des images) avec des attributs de présentation. Pas un programme.

Vu depuis PHP, vous n'avez la main sur ce que fait l'utilisateur que quand il clic. Pas avant.

Exemple : L'utilisateur choisi une valeur dans une combo box, vous ne pouvez pas changer l'affichage avant qu'il ait cliqué sur « Afficher »

1.6.4 : Langages clients

Ce dernier point rebute beaucoup de développeurs. C'est pourquoi on voit se multiplier les appels à les langages tels que Java, JavaScript (Ne pas confondre ...) ou Flash.

Java

C'est un langage (développé par Sun) à mi chemin entre l'interprètè et le compilé.

Il est présent dans les applications Web sous forme d'applets (petit programmes) java.

Ces programmes disposent de puissantes interfaces graphique et s'exécutent dans n'importe quel navigateur disposant d'une machine virtuelle Java.

Une applet Java reste lente à charger et consomme beaucoup de ressources (processeur & mémoire). Il faut donc limiter leur usage à des cas précis (Jeux, Chat, ...)

Flash

Un peu à l'instar de Java, mais en version propriétaire), Flash (Macromédia) permet de réaliser des applications graphiques complexes ... mais lourde à installer.

D'autant plus que ces dernières ne fonctionneront que si le navigateur contient le programme additionnel adéquat (Plug-in).

Quand on sait que Macromédia est le concurrent de Microsoft, et que Linux & le monde libre rechigne à installer les logiciels propriétaires ... c'est pas demain que les navigateurs auront le plug-in Flash installé de série.

Javascript

C'est un langage Open Source, inspiré de Java (d'où le nom) qui permet d'exécuter des instructions sur le poste client. On peut parler de HTML dynamique.

Le problème principal, est que comme JS est interprété par le navigateur ... il faut souvent une version du code par navigateur. Merci Internet Explorer ...

Différence entre JS et PHP :

Test.php	Test.html
<html> <body> <?php echo «Bonjour le monde»; ?> </body> </html>	<html> <body> <script language=javascript> document.write(«Bonjour le monde»); </script> </body> </html>

Ces deux fichiers afficheront la même chose.

Mais l'un fera travailler le serveur (Imaginez que 10.000 personnes fassent la demande en même temps)

L'autre fera travailler le client (Mais l'utilisateur dispose de l'intégralité du code source)

Utilisé avec parcimonie, JS est un excellent complément à PHP.

1.6.5 : Ajax

Ajax n'est pas un langage (C'est du Javascript et du PHP).

C'est une technique de programmation qui permet d'exécuter des petits scripts php en les liant à des événements javascript. Par exemple, dans la Google bar, quand vous tapez 3 caractères, Google vous propose les recherches les plus fréquentes commençant par ces caractères.

C'est du Ajax.

Nous y reviendrons...

2 Environnement

Il existe des plateformes de développement dites WAMP (par analogie à LAMP)

Ces outils sont très pratiques pour développer, mais en aucun cas ne doivent servir de plateformes de production (Performances dégradées, sécurité non garantie)

2.1 WampServer

Site en français : http://www.wampserver.com/

La version 2.0 (Novembre 2007) Intègre :

- Apache 2.2.6
- MySQL 5.0.45
- PHP 5.2.5

Vous pouvez aussi installer EasyPhp.

2.2 Notepad++

Il est indispensable d'utiliser une éditeur à colloration syntaxique.

Le plus ergonomique et le plus puissant etant aussi gratuit, donc pas d'hésitations : Installez Notepad++

http://notepad-plus.sourceforge.net/

2.3 PhpMyAdmin

PhpMyAdmin est un outil d'administration des bases de données (une série d'écrans pour voir, modifier, sauvegarder ...) vos bases de données.

Il est installé avec WampServer ainsi qu'avec EasyPHP.

Pour le démarrer : Passez par le WAMP ou tappez l'url suivante :

http://localhost/mysql/ si vous avez EasyPHP

http://localhost/phpmyadmin/ si vous avez Wampserver

2.4 Le navigateur

Indispensable pour développer des applications web !

Chacun réagit différemment.

Je recommande, pour développer, de prendre le plus rapide et celui qui respecte le mieux les CSS : Firefox.

Télécharger ici : http://www.mozilla-europe.org/fr/products/firefox/

Pour ceux qui développent des sites à destination du public, il est indispensable de tester son site sur différents navigateurs.

Pour info voici les parts de marché en France, en janvier 2007

Produit	Janvier 2007
Internet Explorer	79,7%
Mozilla/Firefox	15,8%
Safari	2,6%
Opera	0,2%
Netscape	0,2%
Autres (Consoles, téléphonnes, ...)	1,5%

2.5 L'écran

De même, n'oubliez pas que tout le monde n'as pas, comme vous, un écran 1900 * 1440 en 25 pouces !

Testez vos réalisations dans différentes configurations.

Rappel des configurations d'écran les plus fréquemment rencontrées :

Résolution	Janvier 2007
1024 x 768	55,34 %

1280 x 1024	17,23 %
1280 x 800	8,23 %
800 x 600	8,18 %
1152 x 864	3,67 %

3 Et c'est partit !

Rappel : Pour que votre fichier soit interprété par le moteur Zend, il doit :

- Se trouver dans le répertoire www de l'environnement WAMP
- Être appelé par le navigateur (et non ouvert directement)
- Se terminer par une extension reconnue (.php)

3.1 Cohabitation PHP / HTML

Dans un script PHP le PHP cohabite avec le HTML.

Par ex : Script_1.php

```
1    <html>
2         <body>
3              <?php
4                   echo «Bonjour le monde»;
5              ?>
6         </body>
7    </html>
```

Les lignes 1, 2, 6 et 7 contiennent du HTML : Rien de nouveau.

La ligne 3 contient une instruction qui peut se lire comme suit : « Eh, Zend, commence à travailler à partir d'ici »

La ligne 5 signale la fin du code PHP

La ligne 4, enfin, contient une instruction PHP. (Ici Echo, qui correspond à « Afficher »)

3.1.1 Exercice

- Installez un environnement WAMP
- Recopiez le script Script_1.php
- Exécutez-le
- Affichez le code source généré.

Il peut y avoir autant de balises <?php que vous voulez dans un script

Il peut **n**'y avoir **que** du PHP dans un script PHP

Ex : Script_2.php

```
<?php
     echo «<html> »;
     echo «<body> »;
     echo «Bonjour le monde»;
     echo «</body> »;
     echo «</html> »;
?>
```

Il peut n'y avoir aucune instruction PHP dans un script PHP ... mais c'est dommage de ralentir le traitement en faisant lire le fichier par Zend avant de l'envoyer !

Un balise PHP peut être dans une instruction HTML :

```
Bonjour monsieur <strong><?php echo $nom; ?> <strong>
```

Ici, l'instruction PHP affiche le contenu de la variable $nom.

Plus fort encore :

Une directive PHP reste valide même si elle est fermée :

```
<html>
     <body>
          <?php
               if(1 == 2) {
          ?>
          Partie jamais affichée
          <?php
               } else {
          ?>
          Bonjour le monde
          <?php
               }
          ?>
     </body>
</html>
```

Ce programme affichera 'seulement' :
Bonjour le monde

3.3 Les commentaires

Commentaire de bloc :

/* : Début de commentaire

*/ : Fin de commentaire

Tout ce qui est entre deux est ignoré. Et non converti en HTML.

Commentaire de ligne

// : Tout ce qui est à droite est ignoré.

3.4 Les variables

Une variable est un objet repéré par son nom, pouvant contenir des données, qui pourront être modifiées lors de l'exécution du programme.

3.4.1 allocation

Quelque soit le type de variable, son nom doit obligatoirement être précédé du caractère dollar ($). Contrairement à de nombreux langages de programmation, comme le langage C, les variables en PHP n'ont pas besoin d'être déclarées, c'est-à-dire que l'on peut commencer à les utiliser sans en avoir averti l'interpréteur précédemment, ainsi si la variable existait précédemment, son contenu est utilisé, sinon l'interpréteur lui affectera la valeur en lui assignant 0 par défaut. De cette façon si vous ajoutez 3 à une nouvelle variable (non définie plus haut dans le code), sa valeur sera 3...

3.4.2 Nommage des variables

Avec PHP, les noms de variables doivent répondre à certains critères :

un nom de variable doit commencer par une lettre (majuscule ou minuscule) ou un "_" (pas par un chiffre)

un nom de variable peut comporter des lettres, des chiffres et le caractère _ (les espaces ne sont pas autorisés!)

Les noms de variables sont sensibles à la casse.

3.4.3 Typage des variables

Comme la déclaration est implicite, le typage l'est aussi

Instruction	Type de la variable
$Variable = 0;	type entier
$Variable = 12;	type entier
$Variable = 0.0;	type réel
$Variable = 12.0;	type réel
$Variable = "0.0";	type chaîne
$Variable = "Bonjour tout le monde";	type chaîne

3.4.4. Les tableaux

Les tableaux stockent des données sous forme de liste. Les données contenues dans la liste sont accessibles grâce à un index (un numéro représentant l'élément de la liste). Contrairement à des langages tels que le langage C, il est possible de stocker des éléments de types différents dans un même tableau.

Ainsi, pour désigner un élément de tableau, il suffit de faire suivre au nom du tableau l'indice de l'élément entre crochets :

```
$Tableau[0] = 12;
$Tableau[1] = "Coucou";
```

Avec PHP, il n'est pas nécessaire de préciser la valeur de l'index lorsque l'on veut remplir un tableau, car il assigne la valeur 0 au premier élément (si le tableau est vide) et incrémente les indices suivants. De cette façon, il est facile de remplir un tableau avec des valeurs. Le code précédent est équivalent à:

```
$Tableau[] = 12;
$Tableau[] = "Coucou";
```

Lorsqu'un tableau contient d'autres tableaux, on parle de tableaux multidimensionnels. Il est possible de créer directement des tableaux multidimensionnels en utilisant plusieurs paires de crochets pour les index (autant de paires de crochets que la dimension voulue). Par exemple, un tableau à deux dimensions pourra être déclaré comme suit :

```
$Tableau[0][0] = 12;
$Tableau[0][1] = "Coucou";
$Tableau[1][0] = 1245.652;
$Tableau[1][1] = "Au revoir";
```

PHP permet l'utilisation de chaînes de caractères au lieu de simples entiers pour définir les indices d'un tableau, on parle alors de *tableaux associatifs*. Cette façon de nommer les indices peut parfois être plus agréable à utiliser :

```
$Toto['Age'] = 12;
$Toto['Adresse'] = "22 rue des bois fleuris";
$Toto['Nom'] = "Ah, vous auriez bien aimé connaître le nom de famille de Toto...";
```

3.4.5 Constantes

Une constante est une variable dont la valeur est inchangeable lors de l'exécution d'un programme. Avec PHP, les constantes sont définies grâce à la fonction define(). la syntaxe de la fonction define() est la suivante :

```
define("Nom_de_la_variable", Valeur);
```

3.5 Opérateurs, structures et boucles

PHP est proche des autres langages.

3.5.1 If, Else, ElseIf

```
if (expression)
     commande1;
[ else
     commande2;]
```

Si l'expression dans la parenthèse est vrai, la commande1 est exécutée.

Si else est présent, commande2 n'est exécuté que si expression est faux.

Expression doit être de la forme booléenne.

Elle peut être une comparaison ($i < 5), ($toto == 'Oui')

Le résultat d'une fonction, si celle çi est de type booléen isNull($toto), toutFormater()

Une variable booléenne $toto = true; if($toto)

3.5.2 Bloc d'instructions : { }

Le If, par exemple n'admet qu'une seule commande.

Cette commande peut être un bloc.

```
$toto = 1;
if($toto == 1) {
     echo «toto vaut 1»;
     echo $toto;
}
```

3.5.3 While

```
while (expression)
      commandes;
```

Tant que expression est vrai, effectue commande

Attention :

expression est évaluée AVANT d'effectuer commande : Il est donc possible de ne jamais passer dans la boucle.

C'est vous qui gérez la condition de fin de boucle : Attention aux boucles infinies.

```
/* exemple 1 */

$i = 1;
while ($i <= 10) {
    echo $i++;
}

/* exemple 2 */
```

```php
$i = 1;
while (true) {
    echo $i++;
    if ($i > 10) {
        break;
    }
}
```

3.5.4 For

```
for (commande1; expression; commande2)
      commande;
```

Commande1 est exécutée un seule fois (avant d'entrer dans la boucle)

Expression est évaluée, et si elle est vrai, on boucle

Commande 2 est exécutée en fin de boucle, juste avant de revenir à Expression.

```php
/* exemple 1 */

for ($i = 1; $i <= 10; $i++) {
    echo $i;
}

/* exemple 2 */

for ($i = 1; ; $i++) {
    if ($i > 10) {
        break;
    }
    echo $i;
}

/* exemple 3 */

$i = 1;
for (; ; ) {
    if ($i > 10) {
        break;
    }
    echo $i;
    $i++;
}

/* exemple 4 */

for ($i = 1, $j = 0; $i <= 10; $j += $i, print $i, $i++);
```

Boucle For ou boucle While ?

Si, à l'instant ou la boucle s'exécute, vous savez combien de 'tour' vous devez faire, choisissez une boucle For (plus facile à gérer). Sinon (Si l'événement de fin est non maitrisé) choisissez une boucle while (ou do, c'est pareil).

3.5.5 Switch

L'instruction *switch* équivaut à une série d'instructions *if* sur une même variable.

```php
switch ($i) {
case "tarte":
    echo "i est une tarte";
    break;
case "barre":
```

```
    echo "i est une barre";
    break;
case "gateau":
    echo "i est un gateau";
    break;
else:
    echo "i est un aliment quelconque";
}
```

Attention : PHP ne sort pas tout seul d'un switch, il faut le faire explicitement avec un Break.

3.5.6 Foreach

Foreach est une boucle très pratique pour passer en revue un tableau.

```
$annee = array(1=>''janvier'', 2=>''fevrier'', 3=>''mars'', 4=>''avril'',
     5=>''mai'', 6=>''juin'', 7=>''juillet'', 8=>''aout'', 9=>''septembre'',
     10=>''octobre'', 11=>'novembre'', 12=>''decembre'')

foreach($annee as $numero => $mois) {
     echo $mois . 'est le mois N°' . $numero;
}
```

3.5.7 Les ruptures (Continue, Break, Return, Exit)

L'instruction ***continue*** permet d'aller à l'occurrence suivante d'une boucle.

L'instruction ***break*** permet de sortir d'une structure *for*, *foreach*, *while*, *do-while* ou *switch*.

break accepte un argument numérique optionnel qui vous indiquera combien de structures emboîtées ont été interrompues.

Appelée depuis une fonction, la commande ***return*** termine immédiatement la fonction, et retourne l'argument qui lui est passé.

Depuis le script, ***return*** termine le script.

Exit() ou ***Die()*** (synonymes) affichent le message passé en paramétre et termine le script.

3.5.8. les fonctions

Une fonction est un outil informatique que l'on crée sois même.

Il sert généralement à factoriser une opération que l'on répére souvent.

Exemple : Je crée une fonction carre qui calcul le carre du nombre passé en parametre :

```
// creation de l'outil
function carre($nombre) {
     $reponse = $nombre * $nombre;
     retrun $reponse;
}

// utilisation de l'outil
echo ''5 ^ 2 = '' . carre(5);
```

3.5.9. Tout le reste

Plutôt qu'un long exposé théorique, mieux vaut partir sur une série d'exercices.

Pour toute question : la référence c'est http://fr.php.net

4 Exercices

4.1 Afficher la date et l'heure

Ecrire un script Date_V1.php qui affiche

> Nous sommes le **04/12/2007** il est **18h01**

Utiliser :

- la fonction echo (http://fr.php.net/manual/fr/function.echo.php)
- La fonction date (http://fr.php.net/manual/fr/function.date.php)

4.2 Écrire des fonctions

Écrire un script Date_V2.php qui affiche

> Nous sommes le **04/12/2007** il est **18h01**

Ce script doit contenir les fonctions debut_html() et fin_html()

debut_html() :

Avec un paramètre (le titre de la page).

Le paramètre doit être facultatif.

Cette fonction doit générer tout le « haut » d'une page HTML (Balise <html>, <head>, <title>, ... jusqu'à <Body>)

fin_html() :

Pas de paramètres

Cette fonction doit générer tout le « bas » d'une page HTML

4.3 Utiliser des inclusions

Ecrire deux scripts Date_V3.php et fonction.php qui affichent

> Nous sommes le **04/12/2007** il est **18h01**

Le script fonction.php doit contenir les fonctions debut_html() et fin_html()

Le script Date_V3.php doit appeler fonction.php

Utiliser :

- l'instruction require (http://fr.php.net/manual/fr/function.require.php)

4.4 Un formulaire

Ecrire deux scripts Date_V4.php et Date_V4.html qui affichent

Pour Date_V4.html :

Un formulaire (method=post, action= Date_V4.php)

Un champ input de type text (name = format)

Un champ input de type submit (name = afficher, value = Afficher)

Pour Date_V4.php :

Un affichage :

> La date affichée avec la chaine de données «**\l\e d/m/Y à H\hi**» est la suivante :
> **le 04/12/2007 à 18h01**

Pour récupérer la valeur saisie dans le formulaire dans le second script :

utiliser $_POST['format']

4.5 Programmation Client / Serveur

Ecrire un script Date_V5.php qui affiche à la fois le formulaire et la date.

La difficulté de cet exercice est de bien gérer le « 1er affichage »

On est alors dans un cas ou on affiche la date alors que la variable $_POST['format'] n'existe pas encore ...

Utiliser :

- la structure if (http://fr.php.net/manual/fr/language.control-structures.php)
- la fonction isSet (http://fr.php.net/manual/fr/function.isset.php)

4.6 Formulaire complexe

Écrire un scirpt Exercice6.php qui réalise l'affichage suivant :

La combo couleur est remplie avec les valeurs suivantes :

- Rouge
- Vert
- Noir
- Bleu
- Orange
- Jaune

Quand on clique sur le bouton OK, la chaine de caractéres saisie dans le champ « Texte » s'affiche autant de fois que précisé dans le champ « Nombre » et dans la couleur choisie dans la champ « Couleur »

Attention : La combo box couleur doit étre repositionnée sur la couleur saisie
Et les champ Texte et Nombre doivent conserver les valeurs saisies.

Vous utiliserez :

- la structure for (http://fr.php.net/manual/fr/control-structures.for.php)

5 Les variables, règles avancés

5.1 Portée des variables

Une variable, ou une constante est visible (est définie) dans l'entitée ou elle est crée.

Ex :

Fichier include.php

```
$a = 1;
```

Fichier exemple.php

```
$b = 1;
function toto() {
     $c = 1;
     echo a;   // Affiche une erreur
     echo b;   // Affiche une erreur
     echo c;   // Affiche 1
}
echo a;   // Affiche 1
echo b;   // Affiche 1
echo c;   // Affiche une erreur
```

Note (Pour les geeks hardcore) on peut faire des trucs comme ça :

```
$a = ''toto'';
$toto = ''coucou'';
echo $$a;// affiche coucou
```

5.2 Les variables pré-définies

PHP fourni au développeur une série d'outils sous formes de variables pré-définies. Voici quelques expemles parmis les plus utilisés :

5.2.1 $GLOBALS

Contient une référence sur chaque variable qui est en fait disponible dans l'environnement d'exécution global. Les clés de ce tableau sont les noms des variables globales. $GLOBALS existe depuis PHP 3.

Exemple

```
$a = 1;
$GLOBALS['b'] = 2;
function toto() {
     echo $GLOBALS['b'];     // affiche 2
     echo $GLOBALS['a'];     // affiche 1
}
```

5.2.2 $_SERVER

Les variables fournies par le serveur web, ou bien directement liées à l'environnement d'exécution du script courant. C'est la nouvelle version de l'ancienne variable $HTTP_SERVER_VARS, qui est maintenant obsolète.

Pour en connaître le contenu faire un script qui apelle la fonction phpinfo();

5.2.3 $_GET

Les variables fournies au script via la chaîne de requête URL.

Exemple : Si on apelle le script toto.php par cette URL : toto.php?page=1&tri=nom

on dispose automatiquement des variables $_GET['page'] (qui vaut 1) et $_GET['tri'] qui vaut 'nom'

5.2.4 $_POST

Les variables fournies par le protocole HTTP en méthode POST.

Si le script est appelé par un formulaire on reçoit une occurrence du tableau $_POST pour chaque champ renseigné du formulaire.

Ex : titi.html

```
<form target=toto.php method=post>
Votre nom : <input type=text name=nom><br>
Votre sexe :<input type=radio name=sexe value=m>monsieur
<input type=radio name=sexe value=f>madame
</form>
```

toto.php

```
echo $_POST['nom'];     // affiche la valeur saisie dans le champ « Votre nom : »
echo $_POST['sexe'];    // affiche m ou f, selon la saisie.
```

5.3 les variables de session

Le principe de la programation client serveur fait qu'entre deux appels d'un script, toutes les variables sont perdues.

... Sauf la variable de session.

En début de script (Avant l'envoi de l'en-tête http) faire : `session_start();`

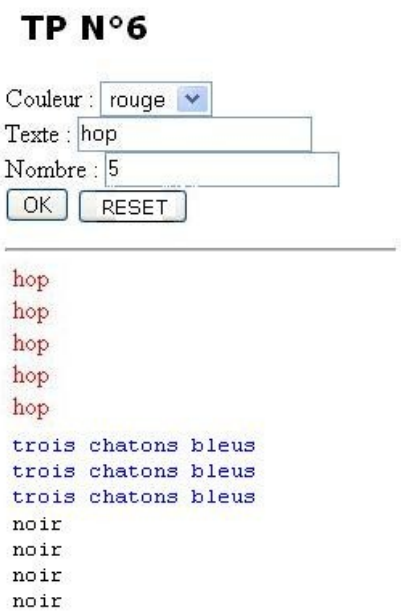
Cette fonction crée ou restaure une session. On dispose alors d'un tableau assocatif ($_SESSION) qui est persistant d'un appel sur l'autre.

Ex :

```
session_start();
if(isset($_SESSION['compteur']))
     $_SESSION['compteur']++;
else
     $_SESSION['compteur'] = 1;
Echo ''ça fait '' . $_SESSION['compteur'] . '' fois que vous appuyez sur F5'';
```

5.4 Exercice

Ecrire un scirpt Exercice7.php qui réalise l'affichage suivant :



Modification par rapport à l'exercice précédent : Le programme garde en mémoire les saisies antérieures de l'utilisateur.
Le bouton Reset vide la mémoire et l'affichage.

Pour faire cet exercice, il faut d'un coté ajouter des lignes à un tableau qui reste en mémoire.
Et de l'autre coté afficher tout le contenu de ce tableau.

Le tableau à ceci de particulier qu'il n'est pas perdu d'une execution du script à l'autre.
La seulle variable PHP à avoir cette particularitée est la variable $_SESSION
Vous devez appeler session_start() en début de programme
et stocker votre résultat dans $_SESSION['affichage']

- Voir : http://fr.php.net/manual/fr/function.session-start.php

La parite du bas est obtenu en affichant tout ce qui se trouve dans $_SESSION['affichage']

Pour 'nettoyer' l'affichage, il suffit alors de désallouer le tableau $_SESSION['affichage']

- Voir la fonction unset (http://fr.php.net/manual/fr/function.unset.php)

7 Les bases de donnée

7.1 Principe

Ceci n'est pas un cours d'initiation à la gestion des bases de données relationelles. Juste de leur utilisation avec Php

Ce cours considére comme préalable que vous savez créer une base de donnée avec PhpMyAdmin et l'administrer.

7.2 Connexion

Un script php qui veut executer une requette doit d'abord se connecter à la base de donnée.

En fait, il doit, dans un premier temps, se connecter à un serveur (dans notre cas, un serveur MySQL), et dans un second temps, choisir une base sur ce serveur.

```
/* Connexion à un serveur MySQL */
$p_base = mysql_pconnect("127.0.0.1", "root", "") or die("Echec de connexion");
/* Ouverture d'une base */
mysql_select_db("MaBase", $p_base) or die("Echec de connexion");
```

La fonction mysql_pconnect :

mysql_pconnect() se comporte exactement comme mysql_connect(), mais avec deux différences majeures :

Premièrement, lors de la connexion, la fonction essaie de trouver une connexion permanente déjà ouverte sur cet hôte, avec le même nom d'utilisateur et de mot de passe. Si une telle connexion est trouvée, son identifiant est retourné, sans ouvrir de nouvelle connexion.

Deuxièmement, la connexion au serveur MySQL ne sera pas terminée avec la fin du script. Au lieu de cela, le lien sera conservé pour un prochain accès.

C'est pourquoi ce type de connexion est dite 'persistante'.

- Le premier paramétre indique le serveur MySQL. Il peut également comprendre un numéro de port, "hostname:port". On aurait pu écire ''Localhost''.
- Le second indique l'utilisateur, et détermine les droits d'accés.
- Le troiséme contient le mot de passe.
  Une base locale à par défaut le user ''root'' sans mot de passe.

La fonction retourne un identifiant de lien persistant MySQL en cas de succès, ou FALSE si une erreur survient.

La syntaxe « or die (''message'')» permet de terminer le script sur un message d'erreur en cas de probléme.

La deuxiéme phase de la connexion consiste à choisir une base sur le serveur : c'est **mysql_select_db()**

7.3 Select

L'éxécution d'une requette se passe également en deux temps :

1 – on éxécute la requette (et on teste le code retour)

2 – on parcours le résultat retourné

Etape 1 :

```
$result = mysql_query("SELECT * WHERE 1=1");
if (!$result) {
      die("Requête invalide : " . mysql_error());
}
```

La fonction **mysql_query()** execute la requete qui lui est passé en paramétre (via une chaine ou via une variable)

Pour les requêtes du type SELECT, SHOW, DESCRIBE, EXPLAIN et les autres requêtes retournant un jeu de résultats, mysql_query() retournera une ressource en cas de succès, ou FALSE en cas d'erreur.

Pour les autres types de requêtes, UPDATE, DELETE, DROP, etc., mysql_query() retourne TRUE en cas de succès ou FALSE en cas d'erreur.

L'étape 2 consiste à exploiter la resosurce retournée pour lire le resultat :

Exemple complet :

```
01<?php
02   /* Test : En local ou en prod ? */
03   if($_SERVER['HTTP_HOST'] == "www.monsite.com") {
```

```
04        $local         = false;
05        $serveur       = "SQL5";
06        $utilisateur   = "NoBrAiN";
07        $password      = "NoPaIn";
08        $base          = "basedeprod";
09    } else {
10        $local         = true;
11        $serveur       = "127.0.0.1";
12        $utilisateur   = "root";
13        $password      = "";
14        $base          = "basedetest";
15    }
16
17    /* Connexion à MySQL */
18    $p_base = mysql_pconnect($serveur, $utilisateur, $password)
19         or die("Echec de connexion au serveur");
20    /* Ouverture d'un base */
21    mysql_select_db($base, $p_base) or die("Echec de connexion à la Base");
22?>
23
24<table>
25    <caption>Liste des fournisseurs</caption>
26    <tr>
27         <th>Code</th>
28         <th>Libelle</th>
29    </tr>
30
31<?php
32    /* Ecriture d'une requete dans une chaine de caractére */
33    $requete  = "Select code, libelle from fournisseur";
34    /* Execution de la requete */
35    $p_resultat = mysql_query($requette);
36    /* Test du code retour de l'execution */
37    if(!$p_resultat) {
38         if($local) {
39              echo "<HR>Requete = " . $requete . "\n";
40              echo "<BR>Erreur N°" .  mysql_errno() . ": " . mysql_error() . "\n";
41              echo "<BR>Script : " . $_SERVER['PHP_SELF'] . "\n";
42         }
43         die("Echec d'execution de la requête");
44    }
45
46    /* Boucle sur le resultat de la requete */
47    while($ligne = mysql_fetch_array($p_resultat)) {
48         echo "<tr>";
49         /* On récupére le résultat sous la forme d'un tableau scalaire */
50         echo "<td>" . $ligne[0] . "</td>";
51         /* ... ou sous la forme d'un tableau associatif */
52         echo "<td>" . $ligne['libelle'] . "</td>";
53         echo "</tr>\n";
54    }
55    /* On libére le pointeur ramené par la requete */
56    mysql_free_result($p_resultat);
57?>
58</table>
```

Explications :

Le test de la ligne 2, avec le nom du serveur, me dit si je suis en local ou en production. Comme cela un même script fonctionne sur les deux environnements sans changer le code.

Dans les deux cas les mêmes variables sont remplies, mais avec des valeurs différentes.

Je peut mettre mon mot de passe en clair, car il ne sera jamais transmis au navigateur.

Ligne 18, je me connecte à un serveur.

Ligne 20 à une base.

La partie situé entre les lignes 1 et 21 doit étre executée au début de chaque script qui fait un acces à la base, il est évident que ce code doit étre factorisé dans un include.

Les lignes 23 à 28 générent le débur d'un tableau HTML

Ligne 33 : je met une requête dans une variable, que j'exécute ligne 35.

A ce stade je ne dispose pas du résultat? Seuleument d'une « resource », c'est à dire d'un machin inexploitable qui pointe sur mon résultat.

Ligne 37 : Dans un premier temps, je teste le bon déroullement de ma requete

Ligne 38 : Si j'ai une erreur et que je suis en local, j'affiche des informations pour le déboggage.

(surtout pas en production, car donner la structure de sa base de donnée est une faille de sécurité)

Ligne 43 : Le local ou en production, je plante si j'ai eu une erreur sur la requete.

Ligne 47 : La fonction **mysql_fetch_array()** exploite la fameuse ressource (de la ligne 35).

Je met le resultat de cette ressource dans une variable $ligne.

Attention c'est une affectation (=) pas une comparaison (==)...

La ligne 47 put se lire comme suit : Tant que tu trouve quelque chose dans la variable $p_resultat, tu le met, sous forme de tableau, dans $ligne.

De la ligne 48 à la ligne 53, pour chaque ligne ramené par la requette SQL, j'ajoute une ligne au tableau HTML avec le code dans la premiére case et le libellé dans la seconde.

Ligne 56 : Prennez l'habitude de libérer la table (en libérabt la ressource) dés que possible, cela évite les bloquages en accés concurents pour le SGBD.

7.4 Update, Insert & Delete

Pour toutes les requetes modificatives, il n'y a pas de phase d'affichage à gérer.

Cela ne dispense pas pour autant de tester de code retour !

```
/* Ceci devrait retourner le nombre correct de lignes effacées */
mysql_query("DELETE FROM mytable WHERE id < 10");
echo "Lignes effacées : " . mysql_affected_rows();
```

8 Les fichiers

8.1 – Les fichiers texte

Pourquoi s'embeter avec des fichiers alors qu'on vient de voir les bases de données, beaucoup plus puissantes ?

Plus puissantes, oui, mais beaucoup plus lente aussi ...

De plus en plus de sites à fort trafic utilisent un systéme de fichier en cache pour fluidifier le trafic.

Si, par exemple, votre application utilise des données à faible taux de rafraichissement (CAD rarement mises à jour), vous pouvez les lire à partir d'un fichier plutôt que d'accéder à la base de donnée.

Si on reprend l'exemple de la liste des villes du TP N°4 : Il est rare que l'on crée une nouvelle ville. On peut imaginer un systéme ou, plutot que chaque utilisateur fasse un select sur la table

des villes, il pioche cette information dans un fichier. Et ce fichier est recrée chaque nuit par le serveur.

Tiens, on vas même le faire ...

8.1.1 - L'ecriture

On ecrit un script «ecrit-cache.php» qui crée un fichier ville.cache à partir de la table Ville.

```
<?php
$p_fichier = fopen("ville.cache", "w+");
$p_base = mysql_connect("localhost", "root", "");
mysql_select_db("TP4", $p_base);
$p_resultat = mysql_query("select * from ville");
while ($resultat = mysql_fetch_array($p_resultat)) {
     $zip = $resultat['codepostal'];
     $ville = $resultat['nom'];
     fwrite($p_fichier, $zip . ";" . $ville . "\n");
}
mysql_free_result($p_resultat);
mysql_close($p_base);
fclose($p_fichier);
echo "Fichier ville.cache (re)cr&eacute;e";
?>
```

Quelques explication :

Ligne 1 : J'ouvre un fichier (**fopen**) et je récupére « en échange » un pointeur sur mon fichier que je vais utiliser ensuite dans toutes les autres fonctions de manipulation de ce fichier.

Le premier paramétre est le nom du fichier, éventuellement précédé d'un chemin.

Voici quelques exemples qui donnent une idée de la puissance de la fonction :

```
$handle = fopen("../file.txt", "r");
$handle = fopen("/home/rasmus/file.gif", "wb");
$handle = fopen("http://www.example.com/", "r");
$handle = fopen("ftp://user:password@example.com/somefile.txt", "w");
```

Le second paramétre donne le mode d'ouverture du fichier :

mode	Description
'r'	Ouvre en lecture seule
'r+'	Ouvre en lecture et écriture
'w'	Ouvre en écriture seule. Si le fichier n'existe pas, il est crée
'w+'	Ouvre en lecture et écriture. Si le fichier n'existe pas, il est crée. Si il existe, il est remis à zéro.
'a'	Ouvre en écriture (Ajout); Si le fichier n'existe pas, il est crée. Si il existe, on place le pointeur à la fin du fichier.
'a+'	Ouvre en lecture et écriture; Si le fichier n'existe pas, il est crée. Si il existe, on place le pointeur à la fin du fichier.

Ligne 8, en fin de boucle While (**fwrite**): j'écrit dans le fichier. Deux paramétres.

Le 1er : Le pointeur qui précise sur quel fichier je travaille.

Le 2nd : La chaine de caractére à écrire dans le fichier (Terminée par /n pour générer un saut de ligne)

Ligne 12 (**fclose**) : Je ferme le fichier.

Cette opération est indispensable, car c'est souvent elle qui conditionne l'écriture physique du fichier par le systéme d'opération.

8.1.2 – La lecture

Pour une lecture locale (sur le même serveur que le script) :

```php
<?php
$p_fichier = fopen("ville.cache", "r");
$villes = fread($p_fichier, filesize("ville.cache"));
fclose($p_fichier);
echo $villes;
?>
```

Fread prend 2 arguments :

Le 1er : Le pointeur qui précise sur quel fichier je travaille.

Le 2nd : le nombre de caractéres que je veut lire (Ici : tout le fichier d'un coup)

Dans le cas d'un lecture distante, on ne peut pas utiliser **filesize**.

Il faut intégrer la lecture dans une boucle while :

```php
<?php
$p_fichier = fopen("http://127.0.0.1/ville.cache", "r");
$villes = "";
while ($unBoutDeFichier = fread($p_fichier, 64)) {
     $villes .= $unBoutDeFichier;
}
fclose($p_fichier);
echo $villes;
?>
```

8.1.3 – Exercice

Ecrivez un script exercice8.php qui affiche le code source de l'url www.google.com.

Et affichez le dans un <textarea> de 80 colonnes sur 15 lignes.

Pour cela vous devez concidérer cette url comme un simple fichier distant à lire.

8.2 – Les templates

8.2.1 - Exemple

Un des grand principes de la programation web, c'est de séparrer le contenu de la présentation.

Toute une nouvelle générations de logiciels sont basés sur ce principe : Ce sont les CMS (Content Management Systems ou Systémes de Gestion de Contenu). Les CMS sont les logiciels qui permettent de générer très facilement des sites internet interactifs (Ou il est facile de publier).

Les plus connus sont Joomla, Spip, ...

Ces logiciels fonctionnent suivant le principe des Templates (ou squelettes dans Spip) :

L'utilisateur s'occupe de la présentation, de la mise en page et le CMS s'occupe de remplir la présentation de l'utilisateur avec les données (c'est donc lui qui gére l'accés à la base, ...)

Voici un exemple simple :

Un utilisateur, même non informaticien, crée une page html de présentation (avec Dreamweaver, OpenOffice, ...)

Et à la place des données, il utilse une liste de mots clé convenu à l'avance.

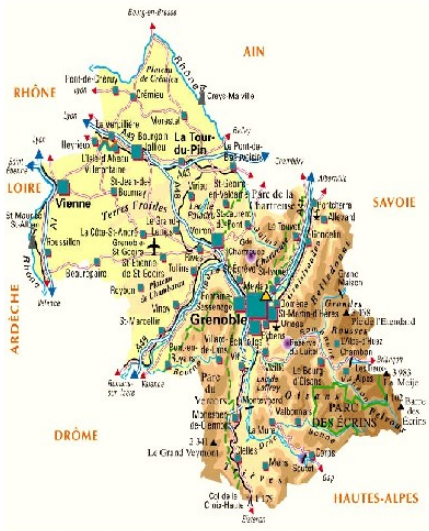
Par example, pour la liste des villes, voici une page modele1.html réalisée avec OpenOffice (Enregistrer sous ... html) :

Code source	Apparence

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0
Transitional//EN">
<HTML>
<HEAD>
	<META HTTP-EQUIV="CONTENT-TYPE"
CONTENT="text/html; charset=windows-1252">
	<TITLE></TITLE>
	<META NAME="GENERATOR"
CONTENT="OpenOffice.org 2.3  (Win32)">
	<META NAME="AUTHOR" CONTENT="Hugues
Levasseur">
	<META NAME="CREATED"
CONTENT="20080306;11345912">
	<META NAME="CHANGEDBY" CONTENT="Hugues
Levasseur">
	<META NAME="CHANGED"
CONTENT="20080306;11391243">
	<STYLE TYPE="text/css">
	<!--
		@page { size: 21cm 29.7cm; margin:
2cm }
		P { margin-bottom: 0.21cm }
		TD P { margin-bottom: 0cm }
		H1 { margin-bottom: 0.21cm }
		H1.western { font-family: "Arial",
sans-serif; font-size: 16pt }
		H1.cjk { font-family: "Lucida Sans
Unicode"; font-size: 16pt }
		H1.ctl { font-family: "Tahoma";
font-size: 16pt }
	-->
	</STYLE>
</HEAD>
<BODY LANG="fr-FR" DIR="LTR">
<H1 CLASS="western" ALIGN=CENTER><FONT
FACE="Verdana, sans-serif">Liste
des %%NbVilles%% villes de l'Is&eacute;re</FONT></
H1>
<TABLE WIDTH=100% BORDER=0 CELLPADDING=4
CELLSPACING=0 STYLE="page-break-before: auto">
	<COL WIDTH=95*>
	<COL WIDTH=161*>
	<TR VALIGN=TOP>
		<TD WIDTH=37%>
			<P ALIGN=CENTER
STYLE="border: 3.00pt double #000000; padding:
0.05cm">
			<FONT FACE="Arial, sans-
serif"><FONT SIZE=2>%%ListeVille%%</FONT></FONT></
P>
		</TD>
		<TD WIDTH=63%>
			<P><IMG
SRC="http://www.1france.fr/images/departements/38-
isere.jpg" NAME="Image1" ALIGN=MIDDLE WIDTH=386
HEIGHT=482 BORDER=0></P>
		</TD>
	</TR>
</TABLE>
<P STYLE="margin-bottom: 0cm"><BR>
</P>
</BODY>
</HTML>
```

Liste des %%NbVilles%% villes de l'Isére

%%ListeVille%%



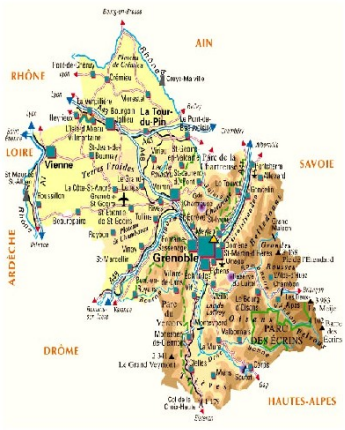
Et voici un script php (dit wrapper) qui va lire modele1.html et replacer les mits clé par les valeurs lues dans la base :

Code Source	Resultat

```php
<?php
/* Transfert du fichier modele1.html dans la
chaine $html */
$nom_fichier = "modele1.html";
$p_fichier = fopen($nom_fichier, "r");
$html = fread($p_fichier, filesize($nom_fichier));
fclose($p_fichier);

$p_base = mysql_connect("localhost", "root", "");
mysql_select_db("TP4", $p_base);
$p_resultat = mysql_query("select * from ville");
/* Recuperation du nombre de villes */
$nbVille = mysql_num_rows($p_resultat);
/* recuperation de la liste des villes */
$listeVille = "";
while ($resultat = mysql_fetch_array($p_resultat))
{
        $listeVille .= $resultat['codepostal'] .
"\t" . $resultat['nom'] . "<br>\n";
}
mysql_free_result($p_resultat);
mysql_close($p_base);

/* Remplacement des mots cle par leur valeurs */
$html = str_replace("%%NbVilles%%", $nbVille,
$html);
$html = str_replace("%%ListeVille%%", $listeVille,
$html);

/* Affichage */
echo $html;
?>
```

Liste des 634 villes de l'Isére

38000 GRENOBLE
38038 SECT-DEPT-RATTACHE2
38070 SAINT-QUENTIN-FALLAVIER
38080 FOUR
38080 L'ISLE-D'ABEAU
38080 SAINT-ALBAN-DE-ROCHE
38080 SAINT-MARCEL-BEL-ACCUEIL
38090 BONNEFAMILLE
38090 ROCHE
38090 SAINT-QUENTIN-FALLAVIE
38090 VAULX-MILIEU
38090 VILLEFONTAINE
38100 GRENOBLE
38110 CESSIEU
38110 CHATANAY
38110 DOLOMIEU
38110 EVRIEU
38110 FAVERGES-DE-LA-TOUR
38110 LA-BATIE-MONTGASCON
38110 LA-CHAPELLE-DE-LA-TOUR
38110 LA-TOUR-DU-PIN
38110 MONTAGNIEU
38110 ROCHETOIRIN
38110 SAINT-CLAIR-DE-LA-TOUR
38110 SAINT-DIDIER-DE-LA-TOUR
38110 SAINT-JEAN-DE-SOUDAIN
38110 SAINT-VICTOR-DE-CESSIEU
38110 SAINTE-BLANDINE
38112 MEAUDRE
38113 VEUREY-VOROIZE
38114 ALLEMOND
38114 OZ
38114 RIVIER-D'ALLEMONT
38114 VAUJANY
38114 VILLARD-RECULAS
38118 HIERES-SUR-AMBY
38118 SAINT-BAUDILLE-DE-LA-TOUR
38119 PIERRE-CHATEL
38119 SAINT-THEOFFREY
38119 VILLARD-SAINT-CHRISTOPHE
38120 FONTANIL-CORNILLON
38120 MONT-SAINT-MARTIN
38120 PROVEYSIEUX



Vous comprenez l'intéret de ce fonctionnement : Chaque utilisateur peut personnaliser sa présentation, sans même avoir de notions de programation, et l'accés aux données reste sécurisé.

8.2.2 – Exercice

Ecrivez un wrapper exercice9.php affin qu'il interpréte ces mêmes mots clé :

%%NbVille%%

%%ListeVille%%

non pas à partir de la base de donnée, mais à partir du fichier « ville.cache » précédement construit.

Vous vous arrangerez pour que votre script accepte un paramétre (passé par l'URL) modele qui indique le nom du template à utiliser. Exemple : exercice9.php?modele=modele1.html

Pour les plus rapides :

Ajoutez en début de script :

```
$debut = time() + microtime();
```

Et en fin de script :

```
$fin = time() + microtime();
echo "<br>Page générée en " . round($fin - $debut, 3) . " secondes";
```

Procédez de même pour la version « Sur base de donnée » et comparez les temps d'execution.

8.3 : Les fichiers compressés

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

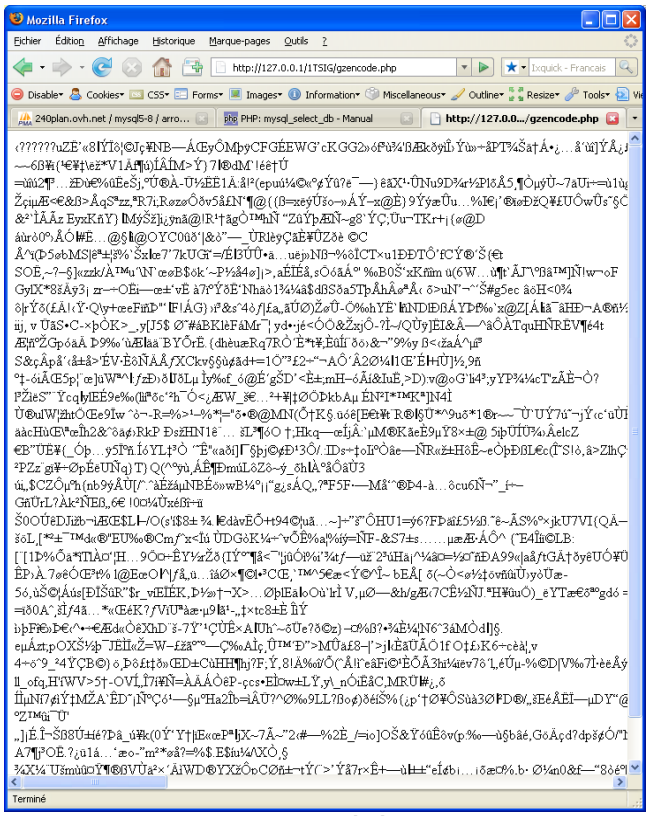
- L'algorithme de compression est excellent

- Des outils de décompression open source existent sur tout les OS (Stuffit Expander sous Mac, 7Zip sous Windows)

Dans un premier temps, on récupére la liste des villes (... pour changer) et on l'affiche compressée (Fonction **gzencode**) :

```
<?php
$p_base = mysql_connect("localhost", "root", "");
mysql_select_db("TP4", $p_base);
$p_resultat = mysql_query("select * from ville");
$liste = "";
while ($resultat = mysql_fetch_array($p_resultat)) {
     $liste .= $resultat['codepostal'] . "," . $resultat['nom'] . "\n");
}
mysql_free_result($p_resultat);
mysql_close($p_base);

echo gzencode($liste);
?>
```

Le resultat est ... décevant :



Le probléme vient du fait que le fichier est conssidéré par le navigateur comme un fichier html : Il est donc affiché tel quel.

Le truc consiste à modifier l'entéte HTTP du fichier généré par le script php, et à préciser que ce n'est par un flux html mais un flux qui contient un fichier Gzippé.

Pour cela on utilise la fonction **header**

Recopiez le script précédent et ajoutez au début les lignes suivantes :

```
header("Content-type: application/x-gzip");
header("Content-disposition: attachement; filename=\"listeville.txt.gz\"");
```

... et testez.

La classe !

Attention :

Le header d'un fichier généré ne peut étre mis qu'une fois.

Et le naigateur le fixe, par défaut à HTML.

Il faut donc utiliser la fonction header avant que le moindre caractére ne soit émis par le script.

Or une simple ligne blanche avant le prompt <?php génére le header html par défaut.

8.4 : Fichier CSV

CSV signifie Comma Separated Value, ou Valeur Séparés par une Virgule.

C'est un format ouvert reconnu par tout les tableurs (Excel, ...)

Pour générer un fichier csv : On modifie légérement le script précédent :

- Content-type devient « text/x-csv »
- filename=\"listeville.txt.gz\" devient filename=\"listeville.csv\"
- Dans la boucle, il faut finir chaque ligne par « \r\n » au lieu de « \n »
- le echo final ne passe plus par la fonction gzencode, mais affiche directement $liste

C'est tout.

... vous pouvez tester.

8.5 : Fichiers Flash

PHP est écrit par des développeurs pour des développeurs. Dit autrement la notion de défi technique prend souvent le pas sur des considérations purement économiques...

Parmis les modules farfelus qui sont venus se gréffer, on trouve le projet Ming, qui permet de générer des fichiers SWF.

SWF c'est le format ShockWave Flasf, utilisé par Macromédia.

Préalable :

Il faut ajouter la dll au serveur apache.

Ouvrez le fichier Fichier php.ini et enlevez le ; devant

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

Redémarez le serveur Apache.

Exemple de fichier SWF : Un player mp3.

Le script suivant joue le fichier tadam.mp3, placé dans le même répertoir que le script :

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

Vous trouverez plein d'exemples sur la page sourceforge du projet Ming :
http://ming.sourceforge.net/examples/index.html

8.6 : Fichiers PDF

Comme pour le chapitre précédent, vous devez autorisez l'extension php_pdf.dll dans le fichier php.ini

Voici un script qui génére un fichier pdf avec une ville par page :

```
<?php
$pdf = pdf_new();
pdf_set_info($pdf, "Author", "Moi même");
pdf_set_info($pdf, "Title", "Liste des villes");
pdf_set_info($pdf, "Creator", "Encore moi");
pdf_set_info($pdf, "Subject", "exemple");

$p_base = mysql_connect("localhost", "root", "");
mysql_select_db("TP4", $p_base);
$p_resultat = mysql_query("select * from ville");
$liste = "";
while ($resultat = mysql_fetch_array($p_resultat)) {
     pdf_begin_page($pdf, 595, 842);
     pdf_add_bookmark($pdf, "Fiche " . $resultat['nom'], 0, 0);
     $font = pdf_findfont($pdf, "Helvetica", "host", 0);
     if($font)
          pdf_setfont($pdf, $font, 12);
     pdf_set_value($pdf, "textrendering", 1);
     pdf_show_xy($pdf, $resultat['codepostal'] . " : " . $resultat['nom'], 50,
750);
     pdf_moveto($pdf, 50, 740);
     pdf_stroke($pdf);
     pdf_end_page($pdf);
}
mysql_free_result($p_resultat);
mysql_close($p_base);
pdf_close($pdf);

$chaine = pdf_get_buffer($pdf);
$taille = strlen($chaine);

/* Version pour un affichage en ligne */
header("Content-type: application/pdf");
header("Content-length: " . $taille);
header("Content-disposition: inline; filename=\"listeville.pdf\"");
echo $chaine;

/* Version pour enregistrer sur le disque
$p_fichier = fopen("listeville.pdf", "w+");
fwrite($p_fichier, $chaine);
fclose($p_fichier);
*/

pdf_delete($pdf);
?>
```

8.7 : Fichiers image

Le projet PHP qui permet de créer dynamiquement – presque – tout les formats d'images s'apelle la librairie GD (Graphic device).

Elle est intégré à PHP depuis la version 4.0.2 : Pas de modif à faire dans le fichier php.ini

Elle permet de générer des images au format jpeg, png wbmp (format wap) ... et depui peu gif puisque l'algorithme LZW viens de tomber dans le domaine public.

Exemple simple :

Un script bouton.php qui crée un bouton contenant un texte passé en paramétre

```php
<?php

header("Content-type: image/png");
$string = $_GET['text'];
$im     = imagecreatefrompng("images/button1.png");
$orange = imagecolorallocate($im, 220, 210, 60);
$px     = (imagesx($im) - 7.5 * strlen($string)) / 2;
imagestring($im, 3, $px, 9, $string, $orange);
imagepng($im);
imagedestroy($im);

?>
```

Et Appel.html, un script qui apelle bouton.php

```
<html>
     <head>
          <title>GD c'est top</title>
     </head>
     <body>
          <img src="bouton.php?text=coucou" />
          <br />
          <img src="bouton.php?text=hop" />
     </body>
</html>
```

Pour en savoir plus : Lire l'un des tutoriels suivant (ou les 4 ...) :

- http://www.lephpfacile.com/cours/22-la-librairie-gd
- http://mtodorovic.developpez.com/php/gd/
- http://www.phpdebutant.org/article111.php
- http://www.siteduzero.com/tuto-3-166-1-creer-des-images-en-php.html

8.7.1 - Exercice

Vous devez générer une image de graphique à barre.

Le graphique doit avoir 26 barres.

Chaqune correspond à une lettre de l'alphabet

La hauteur de chaque barre est proportionelle au nombre de villes (dans la table ville) dont le nom commence par cette lettre.

Le graphique doit tenir en une image de 800 * 600 pixels et etre de format png.

Les barres sont verticales

Les lettres correspondantes doivent etre ecrite en dessous de chaque barre

8.8 : Fichiers XML

XML est **le** format d'échange de données sur internet.

Bien sur, PHP offre plein de facilitées pour lire et ecrire des fichiers XML.

8.8.1 – Petite présentation d'XML

Si je veut décrire une classe et des éléves j'obtient un fichier du genre :

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

```
     <eleve>Pierre Durand</eleve>
</classe>
```

On peut détailler encore plus :

```
<classe>
     <eleve>
          <nom>Dupont</nom>
          <prenom>Paul</prenom>
     </eleve>
     <eleve>
          <nom>Durand</nom>
          <prenom>Pierre</prenom>
     </eleve>
</classe>
```

On peut même ajouter des attributs :

```
<classe niveau="cp">
     <eleve>
          <nom>Dupont</nom>
          <prenom>Paul</prenom>
     </eleve>
     <eleve>
          <nom>Durand</nom>
          <prenom>Pierre</prenom>
     </eleve>
</classe>
```

Les régles à respecter :

- Une seulle racine (Ici c'est <classe>
- Les balises orphelines doivent se finir par /> ex : <prof />
- Les attributs doivent etre entre guillemets
- La case doit etre respectée entre la balise ouvrante et la balise fermante
- Le fichier doit commencer par une ligne qui donne la version XML utilisé et l'encodage des données. Ex :

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

8.8.2 – La liste des villes en fichier XML :

```php
<?php
header("Content-Type: text/xml");
$xml = "<?xml version=\"1.0\" encoding=\"ISO-8859-1\" ?>\n";
$xml .= "<ListeVille>\n";

$p_base = mysql_connect("localhost", "root", "");
mysql_select_db("TP4", $p_base);
$p_resultat = mysql_query("select * from ville");
while ($resultat = mysql_fetch_array($p_resultat)) {
     $xml .= "<ville code=\"" . $resultat['code'] . "\">\n";
     $xml .= "<codepostal>" . $resultat['codepostal'] . "</codepostal>\n";
     $xml .= "<nom>" . $resultat['nom'] . "</nom>\n";
     $xml .= "<pays>" . $resultat['pays'] . "</pays>\n";
     $xml .= "</ville>\n";
}
mysql_free_result($p_resultat);
mysql_close($p_base);
$xml .= "</ListeVille>\n";
```

```php
echo $xml;

$p_fichier = fopen("ville.xml", "w+");
fwrite($p_fichier, $xml);
fclose($p_fichier);
?>
```

Recopiez le script ci dessus, et testez l'affichage de données xml dans votre navigateur.

8.8.3 – 3 exemples de lecture de fichier xml

1 : Tout afficher

```php
<?php
$classe = simplexml_load_file("ville.xml");
var_dump($classe);
?>
```

2 : Afficher uniquement les noms des villes

```php
<?php
$classe = simplexml_load_file("ville.xml");
foreach($classe as $ville) echo $ville->nom . "<br>";
?>
```

3 : Tout afficher de la premiére ville

```php
<?php
$classe = simplexml_load_file("ville.xml");
$uneVille = $classe->ville[0]->attributes();
foreach($uneVille as $cle => $valeur) {
     echo "Cle = " . $cle . "<br>";
     echo "Valeur = " . $valeur . "<br>";
}
?>
```

9 la programation objet

9.1 – Introduction

9.1.1 – Considérations philosophyques

Depuis 65 ans que les ortinateurs existent (eh oui, Colossus, c'etait en 1943) la puissance de calcul a vertigineusement progressé

La « loi » de Moore parle d'une puissance qui double tout les 18 mois.

Soit $2^{(65/1,5)}$ ou 2^{43} fois ou 8.796.093.022.208 ou 8 mille milliards de fois.

Cool non ?

Et les logiciels, quels progrés ont ils fait durant la même période ?

Certes, on est pasé des fiches électriques à brancher dans des prises aux cartes à perforér

Puis à l'assembleur, puis au COBOL puis au C++ pour aboutir aujourd'hui à des langages aussi perfectionnés que le PHP.

...

Mais quand on écrit un programme, on part toujours d'une page blanche !

Heureusement que les électroniciens sont plus doués pour la réutilisation.

9.1.2 – L'idée directrice de la POO

POO pour Programation Orientée Objet.

L'idée c'est de créer des outils, des briques de programmes qui peuvent étre réutilisées de façon autonome.

L'avantage d'une organisation en objets est que chaque morceau de code aura des régles d'accés qui permetent à la fois de ne rien oublier et de n'avoir aucune redondance du code.

Par exemple, je crée un Agenda en php.

Ce qui inclut des tables, des fonctions d'affichage, de recherche de créneaux libres, ...

Le programmeur qui récupérera mon code n'a pas besoin de savoir comment je me suis débrouillé. Il à « seulement » besoin du projet comme d'une boite noire, et d'un ensemble d'outils pour le manipuler :

- Un outil pour l'installer
- Un outil pour l'afficher
- Un outil pour récupérer les RDV d'un utilisateur à une date donnée
- Un outil pour changer la couleur d'affichage des cases occupées entre 25 et 50 %
- etc

Bref, la programation objet apporte la réutilisation du code.

Couplée avec les téléchargements de modules open source, c'est une vraie évolution dans la maniére d'écrire des applications.

9.1.3 – Attention

Ce chapitre traite de la POO appliquée au PHP

Ce n'est en aucun cas un cours théorique de programation orientée objet.

Si les concepts abordés ici vous aident à mieux comprendre la POO : tant mieux

Si les racourcis que j'utilise sont en opposition avec votre cours d'algorithmie : C'est l'autre prof qui à raison !

9.2 – Vocabulaire

9.2.1 – Objet & Classe

Le mot objet est utilisé car c'est le plus générique que l'on ai en programation (Y'avait bien Machins ou bidules, mais c'etait limite sérieux ...)

Donc Un **Objet**, c'est un truc...

Un **Classe** c'est le moule qui à servit à créer le truc et qui lui donne des fonctions et des données.

Je met en rouges les nouveaux mots clé PHP au furs et à mesure.

Exemple :

```
class Rectangle {
}
```

On à fabriqué un moule pour créer des objets.

```
$unRectangle = new Rectangle();
$champ_de_fleurs = new Rectangle();
```

Là on à utilisé le moule 2 fois pour créer deux objets

Rectange est une classe

$unRectangle et $champ_de_fleurs sont des objets

Créer un objet à partir d'une classe see dit aussi instancier une classe.

9.2.2 – Attribut & Méthode

On vas perfectionner un peu la classe Rectangle en lui ajoutant des propriétées :

- Longueur
- Largeur
- Couleur

et des fonctions :

- Surface
- Perimetre

Les propriétées s'apellent des **attributs**

Les fonctions s'apellent des **méthodes**

```
<?php
class Rectangle {
     public $longueur = 0;
     public $largeur = 0;
     public $couleur = "rouge";

     function perimetre() {
          return (2 * ($this->longueur + $this->largeur));
     }

     function surface() {
          return ($this->longueur * $this->largeur);
     }
}
?>
```

9.2.3 – Visibilitée

Trois nouveaux concepts :

- ->
- $this
- public

-> est la notation pour adresser un morceau d'un objet (attribut ou méthode)

Par exemple, si j'instancie $toto = new Rectangle(); je peut récdupérer l'attribut couleur de $toto en faisant $toto->couleur

-> (moins suivit de supérieur) s'applique à une objet, pas à une classe.

Le $ à disparu de l'attribut.

Un méthode s'apelle pareillement : $toto->perimetre();

$this est un mot clé PHP pour adresser un attribut ou une méthode depuis la classe. Il remplace le nom de l'objet.

Public est l'un des trois niveaux de visibilitée des attributs d'une classe (Private, Protected &

Public) Ils définissent la visibilitée d'un attribut (Voir plus loin)

Exemple d'utilisation :

```
$toto = new Rectangle();
$toto->largeur = 5;
$toto->longueur = 8;
echo $toto->perimetre();
```

9.3 – Méthodes magiques

9.3.1 – Le constructeur (__construct)

Cette méthode est automatiquement appelée quand on instancie l'objet.

Elle sert généralement à initialiser les attributs de l'objet.

9.3.2 – Le destructeur (__destruct)

Cette méthode est automatiquement appelée quand on desalloue l'objet.

Mais pas quand le programme se termine.

Exemple :

```
<?php
class Rectangle {
     private $longueur = null;
     private $largeur = null;
     private $couleur = "rouge";

     function __construct($long, $larg) {
          $this->$longueur = $long;
          $this->$largeur = $larg;
          if($this->$longueur == null or $this->$largeur == null)
               echo "Ce rectangle est mal taillé !<br>";
     }

     function perimetre() {
          return (2 * ($this->longueur + $this->largeur));
     }

     function surface() {
          return ($this->longueur * $this->largeur);
     }

     function __destruct() {
          echo "Tchô le beau rectangle<br>";
     }
}

$toto = new Rectangle(6, 2);
echo "Surface de toto : " . $toto->surface . " m²<br>";
unset($toto);
?>
```

Ce programme affichera :

```
Surface de toto : 12 m²
Tchô le beau rectangle
```

9.4 – Conventions d'utilisation

9.4.1 GetToto et SetToto

Affin d'aboutir à des objets « autonomes » on protége les attributs.

Pas pour faire suer le monde, mais pour factoriser le code, et s'assurer que tout les contrôles seront bien fait.

Généralement, chaque attribut privé est « caché » derriére les méthodes Get et Set.

Exemple :

```
<?php
class Rectangle {
     private $longueur = null;
     private $largeur = null;
     private $couleur = "rouge";

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

     function GetLongueur() {
          return $this->$longueur;
     }

     function SetLongueur($long) {
          if(empty($long))
               return false;

          if(!is_numeric($long))
               return false;

          $this->$longueur = $long;
          return true;
     }

     function GetLargeur() {
          return $this->$largeur;
     }

     function SetLargeur($larg) {
          if(empty($larg))
               return false;

          if(!is_numeric($larg))
               return false;

          $this->$largeur = $larg;
          return true;
     }

     function perimetre() {
          return (2 * ($this->longueur + $this->largeur));
     }

     function surface() {
          return ($this->longueur * $this->largeur);
     }

}
```

```
$toto = new Rectangle(6, 2);
$titi = 5;

echo "Largeur du rectangle : " . $toto->GetLargeur();
if(!$toto->SetLargeur($titi))
     echo "titi pas numérique";
echo "nouvelle largeur : " . $toto->GetLargeur();
?>
```

9.5 – Allocation dynamique

9.5.1 Allocation dynamique

On peut faire ça :

```
<?php
class Truc {
}
$toto = new Truc();
$toto->machin = 2;
$toto->bidule = "Hop";
?>
```

C'est l'allocation dynamique d'attributs.

9.5.2 Encore des méthodes magiques : __get et __set

En cas d'allocation dynamique la méthode __set est appelée (Si elle existe).

Elle admet deux paramétres : Le nom et la valeur de la nouvelle variable.

On peut aussi prévoir la méthode __get().

Elle admetera alors un paramétre : Le nom de la nouvelle variable.

9.6 – Polymorphisme

Parmis les 3 types de polymorphisme ... seul l'héritage est utilisé.

9.6.1 Héritage

Le principe c'est de créer des classes génériques et des classes spécialisées.

Ex :

```
<?php
class Rectangle {
     public $longueur = 0;
     public $largeur = 0;
     public $couleur = "rouge";

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

     function perimetre() {
          return (2 * ($this->longueur + $this->largeur));
     }
```

```
     function surface() {
          return ($this->longueur * $this->largeur);
     }
}

class Carre extends Rectangle {
     public $cote = 0;

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

$toto = new Carre(5);
echo "Surface du carré : " . $toto->surface();
?>
```

Le mot clé Extends sert à définir l'héritage

Le mot clé parent fait référence à une méthode d'une des classes dont Carre descend.

Les :: qui précédent l'appel au constructeur du parent est un appel dit statique : On utilise une méthode d'une classe sans l'avoir préalablement instancié.

Le niveau de visibilité Protected s'applique aux héritages : Seuls les héritiers peuvent utiliser l'attribut.

9.6.2 Surcharge

Si un fils à une méthode qui porte le même nom que celle de l'un de ses ancétres, c'es tla méthode du fils qui est prioritaire.

Sauf si la méthode de l'ancêtre est précédée du mot Final.

Pareil pour les attributs

Voilà c'est tout ...

10 Les cookies

Bientôt

11 Architecture client-serveur, régles avancées & Quelques mots sur JavaScript et Ajax

6.1 Javascript

6.1.1 Historique

6.1.2 Principe

6.1.3 Différence avec PHP

... voir chap 6 du livre