Chapitre 3

Concepts de base de PHP

3.1 Le langage PHP

Inventé par Rasmus Lerdorf en 1994, le langage PHP s'est imposé comme un standard pour la création de pages dynamiques sur Internet. Ce chapitre tente d'en donner une première approche. Elle sera suivie dans les chapitres suivants par une présentation de la gestion des formulaires (qui permettent à l'internaute de communiquer avec le serveur) puis de l'interaction avec les bases de données.

Les notions présentées ici ne permettront pas de résoudre tous les problèmes rencontrés. Je peux donner quelques conseils pour trouver les immenses ressources disponibles sur Internet. En outre, de nombreux manuels ont été publiés sur le sujet. Les éditions O'Reilly (uniquement en anglais depuis la faillite de leur branche française), et les éditions Eyrolles en proposent plusieurs d'excellente qualité.

1. pour trouver une description d'une commande ou d'une fonction particulière, taper « php *nomdefonction* » dans Google. On aboutit généralement sur *php.net* (entièrement traduit en français). Ce site est rempli de liens internes qui renvoient à des tables de matières. Chaque page est suivie de dizaines de commentaires avec des remarques et des conseils.
2. le site *http://php.developpez.com* offre également des centaines d'articles, de tutoriels, de forums et de ressources sur PHP et des notions annexes, comme ses extensions et ses frameworks.

3.2 Interaction de PHP et HTML

La différence fondamentale entre une page HTML et une page PHP, marquée par l'extension du nom de fichier, réside dans le fait qu'une page PHP sera toujours confiée au module PHP afin qu'il l'interprète. À la différence d'une page HTML, qu'on peut visualiser sans peine sur son PC en tapant le nom complet du fichier dans la zone d'URL, un fichier PHP doit passer par un serveur web pour pouvoir être affiché valablement.

En principe et malgré l'extension du nom de fichier, l'interpréteur considère que la page est composée de code HTML, qui sera intégralement transmis au client. Si on veut lancer l'interpréteur, il faut utiliser une balise de script. Plusieurs écritures sont possibles, mais le format suivant ne posera jamais de problème :

```
<?php
// programme d'une seule instruction
echo 'Bonjour le monde';
?>
```

La notation **<?** peut donner lieu à de mauvaises interprétations si le serveur dispose d'interpréteurs pour d'autres langages [1]. Très subtil aussi, tout texte situé avant la balise de script est considéré comme faisant partie de la page HTML. C'est notamment vrai d'un simple espace ou d'une ligne vide, qui seront envoyé au navigateur avant l'exécution du script. Cette émission déclenche l'envoi du header HTTP, qui précède le texte. Certaines opérations ne sont plus possibles quand le header a été envoyé. Cela peut provoquer des erreurs irritantes quand on veut utiliser des sessions. Par contre, les blancs à l'intérieur du code PHP n'ont pas d'importance (sauf évidemment dans les chaînes de caractères).

Comment mélanger HTML et PHP

Trois techniques sont disponibles pour créer la page d'un site :

1. on écrit une page HTML normale et on y dispose des morceaux de scripts PHP pour gérer certaines parties variables.
2. on écrit un programme en PHP (la balise `<?php` se trouve alors sur la première ligne du fichier) qui génère la page HTML uniquement à l'aide de commandes d'affichage `print` ou `echo`.
3. on utilise des templates (généralement fournis en Open Source) : cela suppose l'installation du template sur le site (une série de pages en PHP). Le concepteur écrit ensuite deux types de pages : des pages contenant l'essentiel du code HTML avec quelques marqueurs spéciaux (propres au template et souvent entourés d'accolades) et des pages de programmation qui utilisent les fonctions spécifiques du template [2].

La première et la dernière techniques doivent s'employer quand on travaille en collaboration avec des graphistes. La technique du tout PHP éclate le code HTML au sein de petites sous-routines, ce qui rend la collaboration avec les graphistes plus difficile. Notons aussi que l'utilisation d'un template passe par la lecture et l'apprentissage de son copieux manuel.

On peut encore utiliser la solution du *framework*, un ensemble de routines et d'outils qui permettent de générer rapidement des sites en proposant des classes déjà préconfigurées. Parmi ceux-ci, on citera **CakePHP**, qui constitue une transposition assez réussie de *Ruby On Rails* sur PHP, **CodeIgniter,** son clone **Kohana**, **Symphony** et **Zend Framework**. Personnellement, je juge Zend intéressant, par son utilisation de PHP5, par son respect du modèle MVC et par une structuration assez géniale de l'arborescence des répertoires. Tous ces frameworks nécessitent un investissement important en apprentissage avant de pouvoir être utilisés. Notons enfin que certains entretiennent des relations amicales avec les templates : soit en intégrant un modèle de templates, soit en autorisant l'intégration d'un gestionnaire de templates, soit enfin en proposant des services qui s'apparentent aux templates. Il n'en sera plus question dans cette petite introduction.

3.3 Syntaxe de base

Héritage du C et de Perl

PHP est un langage de scripts qui dérive de Perl et de C. Quelques différences fondamentales existent :

– PHP n'exige pas la déclaration préalable des variables.

1. Elle suppose aussi que le serveur est configuré pour la comprendre (paramètre `short_open_tag=on`.
2. Les étudiants intéressés trouveront une explication détaillées de ce principe sur *http ://phpcodeur.net/articles/php/templates*. Il existe de nombreux systèmes, par exemple **Smarty PHP**.

- les variables ne sont pas typées, seule la valeur qu'on y met permet de déterminer le type de la variable. En outre, une même variable peut changer de type en fonction des valeurs successives qu'on lui attribue.
- les définitions de fonctions et de procédures commencent par le mot `function` pour remplacer le type ou `void`. Seule la présence de l'instruction `return` permet de distinguer une fonction d'une procédure.

Pour le reste, on retrouve les mêmes structures de base dans le langage :

- l'affectation au moyen du signe = et ses variantes ++,--, +=...
- les instructions `if()`... `else` et `switch()` pour les alternatives
- les instructions `while`, `do` ... `while` et `for` pour les boucles (`break` et `continue` existent également). Il existe une autre boucle, nommée `foreach`, que nous verrons dans la section 3.7.3.
- les opérateurs sont les mêmes qu'en C avec trois variantes :
  - il existe un opérateur pour concaténer les chaînes de caractères (`.`)
  - un nouveau type de comparateurs d'égalité/inégalité : `===` et `!==` teste des égalités strictes en tenant compte des types (`1==='1'` renvoie faux), tandis que les opérateurs du C ne tiennent pas compte des types (`1=='1'` renvoie vrai).
  - outre `&&` et `||` il existe deux connecteurs logiques `and` et `or` , qui n'ont pas même priorité, mais là, je conseille de toutes façons l'emploi des parenthèses pour éviter les ennuis.

Remarquons que, puisque PHP n'est pas typé, il autorise l'emploi de la structure `switch` avec tout type de données. C'est particulièrement bienvenu pour les chaînes de caractères.

Syntaxe alternative

Les alternatives et les boucles connaissent une syntaxe alternative qui n'emploie pas d'accolades. Cette syntaxe peut se révéler utile dans une interaction avec HTML où l'accolade fermante peut parfois passer inaperçue. Cette syntaxe s'applique également à `foreach`, mais pas aux boucles `do...while`.

```
if($i==1):
        $a=1;
        $b=2;
elseif($i==3):
        $a=10;
        $b=20;
else:
        $a=100;
endif;

while(i<10):
        echo $i;
        $i++;
endwhile;
for($i=1;$i<10;$i++):
        echo $i;
        echo $i+1;
endfor;
```

3.4 Les variables

Le nom des variables commencent toujours par le signe `$`. PHP est sensible à la casse du nom des variables.

Il existe de nombreuses portées pour les variables.

3.4.1 Variables locales

Les variables utilisées dans le programme principal, dans une fonction ou une méthode de classe ont un statut local. Leur valeur n'est pas accessible en dehors de ce bloc.

3.4.2 Variables globales

Les variables globales doivent être déclarées au moyen de la directive `global` préalablement à leur initialisation. Il est ainsi possible de partager des variables entre plusieurs fonctions et/ou le programme principal. Cette déclaration doit se faire partout où on désire utiliser la variable. Un petit exemple :

```
function testglob(){
global $glob;
$glob=140;
}
testglob();
global $glob;
echo $glob; // affiche 140
```

3.4.3 Variables superglobales

Elles sont définies et gérées par le système. Il s'agit en fait de tableaux qui contiennent des valeurs importantes pour les systèmes : les variables de session, les variables récupérées dans les formulaires remplis par le client et les variables de cookies. À la différence des variables globales normales, il n'est pas nécessaire de les déclarer.

3.4.4 Variables paramètres

Les paramètres d'une fonction sont considérés comme des variables locales de la fonction. Ils sont toujours passés par valeur. Il est néanmoins possible de forcer un passage par référence, au moment de la définition de la fonction [3].

```
<?php
function BadInc($var){
$var++;
}
function Inc(&$var){
$var++;
}
$variable=1;
BadInc($variable);
```

3. La spécification du passage par référence au moment de l'appel est obsolète et déconseillée depuis la version 5.3 du langage.

```
echo $variable.'<br/>'; // 1
Inc($variable);
echo $variable.'<br/>'; // 2
?>
```

PHP permet de donner une valeur par défaut aux paramètres

```
function testPara($un,$deux=2,$trois=3){
echo "$un - $deux - $trois";
}
testPara(10,20,30); // affiche 10 - 20 - 30
testPara(10,20);    // affiche 10 - 20 - 3
testPara(10);       // affiche 10 - 2 - 3
testPara();         // erreur
```

3.4.5 Variables d'instances et de classes

Lors de la définition d'une classe, on peut définir des variables d'instances qui seront contenues dans les instances de la classes. On peut aussi définir des variables communes liées à la classe. Nous reviendrons sur ce sujet plus loin. Il faut noter que ces variables sont généralement précédées du nom de l'objet, de la classe ou de `this` (bien que ce dernier soit parfois facultatif). Ces variables doivent être déclarées (au moyen de `public`, `protected` ou `private`)

3.4.6 Types de variable

Comme il a été dit, les variables n'ont pas de type prédéfini et le type peut changer en cours de programme.

```
$maVar=1;
echo $maVar;
$maVar++;
echo $maVar;
$maVar='Brol';
echo $maVar;
```

On peut tester le type d'une variable avec une série de fonctions prédéfinies :

`gettype($Var)`	renvoie le type de la variable
`is_int($Var)`	renvoie 1 si la variable est de type entier
`is_long($Var)`	renvoie 1 si la variable est de type entier long
`is_double($Var)`	renvoie 1 si la variable est de type flottant
`is_string($Var)`	renvoie 1 si la variable est une chaîne de caractères
`is_array($Var)`	renvoie 1 si la variable est un tableau
`is_object($Var)`	renvoie 1 si la variable est un objet
`empty($Var)`	renvoie 1 si la variable est nulle (0 ou chaîne vide)
`isset($Var)`	renvoie 1 si la variable existe

3.5 Les constantes

On peut définir des constantes en PHP. Voici un exemple :

```
define('BASE','Etudiants');
```

On notera la présence des apostrophes. Ne pas placer d'apostrophes dans une instruction `define` peut donner des résultats parfois surprenant. Par convention, le nom des constantes s'écrit en majuscules.

3.6 Les chaînes de caractères

3.6.1 Définition d'une chaîne constante

Avec les tableaux, les chaînes de caractères jouent un rôle fondamental : effectivement le but de la plupart des applications est de générer un fichier source en HTML. À la différence du C, PHP implémente de vraies chaînes de caractères et surtout un opérateur spécifique, nommé de concaténation `'.'`.

Il existe trois manières de créer une chaîne constante :

– une série de caractères délimités par des apostrophes simples : le contenu de la chaîne est reproduit tel quel et sans aucune possibilité d'interprétation, à l'exception du groupe `\'` qui représente l'apostrophe et de `\\` qui se représente lui-même. C'est la manière la plus rapide d'utiliser des chaînes et on doit essayer d'utiliser cette syntaxe le plus souvent possible.

```
echo 'Il n'y a qu\'une apostrophe dans ce texte';
```

– une série de caractères délimités par des guillements doubles : le contenu de la chaîne est toujours interprété avant envoi, ce qui suppose une charge de traitement supplémentaire. L'interprétation porte sur des groupes commençant par un anti-slash (ce sont les mêmes qu'en C) ou sur des variables, dont le nom commence par `$`. Certains noms de variables complexes (notamment des tableaux avec des indices variables et des variables d'instance précédées d'un nom d'objet) peuvent parfois poser des problèmes. Il vaut mieux alors construire la chaîne en concaténant ce qui précède et suit la variable.

```
$Verbe='servent';
echo "En HTML\n les retours \n à la ligne ne $Verbe à rien.";
```

– une syntaxe reprise d'autres langages de scripts permet d'écrire un texte disposé sur plusieurs lignes. On notera que le mot servant de marque finale peut être choisi librement (ici LAFIN), mais qu'il doit **toujours figurer au début d'une ligne**, sans espace le précédant. On choisira un mot qui ne doit pas figurer dans le texte (en général FIN ou END).

```
$SUBSTANTIF='lignes'
echo <<<LAFIN
J'ai vraiment beaucoup de choses à dire et
il me faut plusieurs $SUBSTANTIF.
LAFIN;
```

Comme dans le cas des apostrophes, les caractères spéciaux et les variables sont pris en compte.

Une des difficultés des chaînes de caractères réside dans le mélange des différents langages : HTML, Javascript, SQL et PHP. Personnellement, j'ai pris la résolution d'utiliser essentiellement des chaînes délimitées par des apostrophes, ce qui met permet d'insérer les guillemets fréquents en HTML. Par contre, il est plus facile d'utiliser les guillemets doubles pour une création de requête SQL, puisqu'on y emploie des apostrophes pour délimiter les chaînes et qu'on utilise souvent des variables PHP pour les constantes de SQL. Dès que la syntaxe devient un peu complexe, j'ai recours à la concaténation :

```
echo '<img␣source="brol.png">';
$query="SELECT␣*␣FROM␣Table␣WHERE␣ID='$Valeur'";
$query2="SELECT␣*␣FROM␣Table2␣WHERE␣ID='".$obj[5]->Val."'";
```

3.6.2 Caractères spéciaux

Un problème sérieux concerne l'encodage des caractères spéciaux. Les caractères de l'anglais ne posent aucun problème, mais les caractères du français doivent figurer sous une forme appropriée. À titre d'exemple, j'ai pris le mot « Été ».

Type	Codage	Remarque
Utiliser l'éditeur	`Été`	Risque de ne pas être lisible sur tous les navigateurs.
Fonction `chr()`	`chr(201).'t'.chr(233)`	Risque de ne pas être lisible sur tous les navigateurs.
Notation hexa	`"\xc9t\xe9"`	Risque de ne pas être lisible sur tous les navigateurs.
Entités html	`"&Eacute;t&eacute;"`	Compatibilité maximale, mais ne fonctionne qu'avec les constantes
Fonction	`htmlentities("Été")`	Possibilité d'appliquer la fonction au contenu d'une variable.

On voit que la fonction `htmlentities()` constitue la solution la plus facile. Attention néanmoins qu'il faut toujours l'appliquer une seule fois, parce qu'à la deuxième application le signe `&` sera converti à son tour. On peut spécifier d'autres arguments pour cette fonction : le second permet de signaler le traitement des guillemets simples ou doubles, le troisième de spécifier un encodage différent de iso8859-1. On consultera la page de la documentation en ligne pour des détails. La fonction `htmlspecialchar_decode()` fait le travail inverse, mais est moins utile parce que les réponses des utilisateurs ne contiennent pas d'entités html.

Que se passe-t-il avec les guillemets simples ou doubles contenus dans les données ? On ne peut évidemment pas placer d'antislash manuellement dans les données. On dispose donc des fonctions `addslashes()` et `stripslashes()` qui ajoutent et retirent automatiquement les antislashes nécessaires. Faute d'être un peu soigneux, on peut facilement voir se transformer le titre du roman « L'étranger » en une horreur : `L\\\'étranger`. Il faut équilibrer les `add` et `strip`.

3.6.3 Fonctions et opérateur .

L'opérateur . sert à concaténer les chaînes de caractères. Par exemple, les deux écritures suivantes sont équivalents :

```
$SQL = "SELECT␣*␣FROM␣$Table";
$SQL = "SELECT␣*␣FROM␣".$Table;
```

PHP dispose en outre de nombreuses autres fonctions pour manipuler les chaînes de caractères. Cela n'a pas de sens de vouloir les citer toutes. Il existe plusieurs familles de fonctions :

- fonctions *informatives* qui donnent des indications sur la longueur d'une chaîne (`strlen()`), la position d'une sous-chaîne (`strpos()`)
- fonctions de *comparaison* (famille `strcmp`)
- fonctions de *modifications* : extractions d'un caractère (`[]`), d'une sous-chaîne (`substr()`), conversion de la casse (`strtolower()`, `strtoupper()`), suppression d'espaces de la famille `trim()`).
- fonctions de *conversion* : surtout utiles pour les dates, puisque les nombres se convertissent spontanément en chaîne lorsque le contexte l'exige. De nombreuses fonctions relient les chaînes et les tableaux.

Un manuel d'introduction au langage ou la documentation en ligne fournissent la liste et des exemples d'emploi.

3.7 Tableaux en PHP

C'est l'une des grandes richesses de PHP. On distingue deux types de tableaux : les tableaux traditionnels ou indicés, et les tableaux associatifs. Notons que les variables tableaux sont des simples variables et qu'il est facile d'effacer tout un tableau en affectant une valeur simple à sa variable.

3.7.1 Tableaux indexés

Un tableau se définit sur le tas, par l'affectation d'une variable au moyen d'un indice. Rien n'oblige à définir les indices séquentiellement, une liberté qui peut se révéler source de surprises.

```
$Test[1] = 1;
$Test[2] = 4;
```

Ou encore à l'aide de la fonction `array()`. Les commandes suivantes

```
$Test = array(1,"deux", 1.5);
print_r ($Test);
```

provoquent l'affichage de ceci :

```
Array
(
    [0] => 1
    [1] => deux
    [2] => 1.5
)
```

On notera l'hétérogénéité des données. La fonction `print_r()` est en principe destinée à la mise au point des programmes, vu qu'elle ne produit aucune mise en page utilisable dans un contexte web.

La notion de tableau à plusieurs dimensions n'existe pas plus qu'en C. On peut cependant placer des tableaux dans un tableau. Mais l'affectation se fait au niveau de chaque élément. L'exemple suivant serait impossible à imaginer en C :

```
$Binome[0]=1;
$Binome[1]=array(1,1);
$Binome[2]=array(1,2,1);
$Binome[3]=array(1,3,3,1);
$Binome[4]=array(1,4,6,4,1);
$Binome[5]=array(1,5,10,10,5,1);
```

La fonction `sizeof()` renvoie le nombre d'élément d'un tableau. Appliquée à `$Binome`, elle renverrait 6. Pour connaître la taille de chacun des sous-tableaux, on devrait programmer une boucle :

```
for($i=0;$i<sizeof($Binome);$i++)
        echo "Taille de \$binome[$i] :".sizeof($Binome[$i])."<br/>";
```

3.7.2 Tableaux associatifs

Dans un tableau associatif, les indices sont remplacés par des clés alphabétiques. Quand ces clés sont constituées d'un seul mot, on peut se passer des guillemets, mais c'est une pratique à déconseiller. L'exemple suivant affiche « one ».

```
$Test = array("un"=>"one","deux"=>"two");
echo $Test["un"];
```

Il est déconseillé de mélanger les indices et les associations, car cela risque de poser certains problèmes desquels seules des explications plus détaillées permettraient de nous libérer.

Les tableaux associatifs seront particulièrement utiles pour la manipulation du résultat des requêtes.

3.7.3 Boucle `foreach`

On a vu plus haut une méthode reprise au langage C pour parcourir un tableau. Il existe une commande plus simple qu'on emploie plus fréquemment : `foreach`.

```
$Musicien = array("Marais","Vivaldi","Bach","Mozart");
foreach($Musiciens as $M)
   echo "$M <br/>";
```

Avec des tableaux associatifs, il est possible d'utiliser une deuxième variable pour récupérer la valeur de la clé :

```
$Nombres = array(un=>"one",deux=>"deux");
foreach($Nombres as $Cle=>$Valeur)
   echo "$Cle : $Valeur<br/>";
```

Attention cependant, les variables manipulées par ces boucles sont des copies des éléments du tableau et ne peuvent donc pas servir à modifier le contenu du tableau. Les versions récentes de PHP permettent de résoudre ce problème en passant des références de variables à la commande `foreach`.

3.7.4 Fonctions concernant les tableaux

Il existe des dizaines de fonctions pour manipuler les tableaux (pour les trier, effectuer des recherches, ajouter et retrancher des éléments...). Voici deux exemples de fonctions utiles :

```
// Analyse en mots
$Phrase = "Bonjour␣le␣monde";
$TPhrase = explode("␣",$Phrase);
print_r($TPhrase); // chaque mot est placé dans une case
// Gestion de multiples conditions
$Conditions[0]='Brol<5';
$Conditions[1]='Nom="Dupond"';
//...
$ClauseWhere='WHERE␣'.implode('␣AND␣',$Conditions);
// WHERE Brol<5 AND Nom="Dupond" AND ...
```