Chap 2 : Syntaxe de base PHP

1. Le fichier

- Extension des fichiers
  Les fichiers PHP doivent obligatoirement se terminer par l'extension .php pour une question d'interprétation du serveur et pour des raisons de sécurité.
  Si vous ne mettez pas .php le fichier pourrait être non interprété et donc le code deviendrait visible (imaginez si il contient des mots de passe) dévoilant votre logique de fonctionnement.

- Nom des fichiers
  Seuls les caractères alphanumériques, tirets bas et tirets demi-cadratin('-') sont autorisés. Les espaces et les caractères spéciaux sont interdits. Le nom du fichier doit être en minuscule.

- Format des fichiers
  Utiliser le format ASCII, utiliser le jeu de caractères UTF-8, être formaté Dos

2. Insertion de code PHP dans le document HTML

Le code php commence par **<?php** et se termine par **?>**
Exemple 1:

```
<?php
      echo 'Hello World !';
?>
```

Important : une instruction se termine par **;**
echo : permet l'affichage

Exemple 2 :

```
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title></title>
    </head>
    <body>
        <?php
        // put your code here
        ?>
    </body>
</html>
```

Un fichier php contient souvent HTML et PHP. C'est la balise **<?php** qui indique au serveur que ce code doit être interprété comme du PHP.

Les // indique que ce qui suit ne doit pas être interprété (commentaire). On peut aussi utiliser /* */ si l'on doit commenter plusieurs lignes.

La partie traitement de PHP peut être inséré n'importe où dans la page. Par contre la partie affichage de PHP doit suivre la logique d'affichage du code HTML.

Exemple :

```
<!DOCTYPE html>
<?php
function paragraphe($message, $couleur='blue')
    {
        return '<p style="color:'.$couleur.'">'.$message.'</p>';
    }
?>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title></title>
    </head>
    <body>
        <?php
            echo paragraphe("Super ce BTS SIO !");
        ?>
    </body>
</html>
```

PHP peut aussi retourner du code HTML via sa commande echo.

Exemple : `echo ‘<p style=«color:blue»>Super ce BTS SIO !</p> ‘`

3. Les variables

Les variables en PHP commencent par un $ et sont formés d'une suites de lettres, de chiffres et de caractères de soulignements. Le premier caractère du nom d'une variable ne peut être un chiffre.

Exemple :

```
<?php
            $message = "Super ce BTS SIO";
            echo $message;
?>
```

L'affectation d'une variable se fait grâce au signe **=**

Si vous voulez vous servir d'une variable non définie vous aurez suivant le paramétrage une notice (avertissement).
Remarques : Les navigateurs n'affichent pas les messages d'erreur par défaut. Pour des raisons de sécurité et aussi car les utilisateurs finaux ne sauraient pas les interpréter. Mais en phase de développement il est important d'avoir les notices et warning.
Pour avoir ces messages il faut paramétrer php.ini ou ajouter ces lignes dans le code PHP.

```
1  <!DOCTYPE html>
2  <html>
3      <head>
4          <meta charset="UTF-8">
5          <title></title>
6      </head>
7      <body>
8          <?php
9              ini_set('display_errors', 1);
10             error_reporting(E_ALL);
11             echo $toto;
12         ?>
```

Notice: Undefined variable: toto in **/Applications/MAMP/htdocs/chap2/variables.php** on line **11**

```
13      <body>
14          <?php
                $message="Super ce BTS SIO";
                echo $messages;
17          ?>
18      </body>
```

Notice: Undefined variable: messages in **/Applications/MAMP/htdocs/Test0/index.php** on line **16**

(avec modification de php.ini)

On peut définir en PHP des variables de manière dynamique (variable qui dépend d'une autre variable).

Exemple

```php
<?php
    $nom="INION";
    $prof='nom';
    echo $$prof;
?>
```

Ici echo remplace $prof par sa valeur cad nom. Comme $nom contient INION.
Echo affiche le contenu de $nom.(ici : INION)
C'est les $$ qui demande à PHP d'aller jusqu'à ce niveau. Avec un seul $ on obtient : nom.

Qu'affiche ce code ?

```php
<?php
    $nom="INION";
    $prenom="Sebastien";
    $prof='prenom';
    echo $$prof;
?>
```

En PHP le typage est dynamique (typage faible). C'est à dire que PHP détermine le type en fonction de la valeur que vous affectez à la variable son type.

Les types possibles sont :

La famille des scalaires (simple): booléen, chaines de caractère, entier, décimal
Les autres : Tableaux, Objets, Ressources

Exemples :

```php
<?php
    $mavariable=false;
    echo $mavariable;
    $mavariable=1;
    echo $mavariable;
    $mavariable=13.2;
    echo $mavariable;
    $mavariable="Salut le monde !";
    echo $mavariable;
    $mavariable=array('ok','pasok');
    echo $mavariable[0];
?>
```

Pour connaitre le type d'une variable on va utiliser une fonction php **gettype()**.

Remarque : Une fonction porte un nom et prend aucun ou plusieurs paramètres (ces paramètres se place dans une parenthèse) et retourne une valeur. On peut utiliser des fonctions déjà écrite ou en réaliser soit même. Si elle retourne pas de valeur on parle de procédure.

```php
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title></title>
    </head>
    <body>
        <?php
        $prenom="lucie";
        echo gettype($prenom);
        ?>
    </body>
</html>
```

On a comme affichage `string`.

Par défaut l'affectation = est une affectation par copie (les deux variables sont par la suite indépendantes).

On peut faire une copie par référence en utilisant le signe **=&**. Dans ce cas les deux variables désignent une même référence.

Exemple :
$a=2; $b=&$a; // $b vaut 2
$a=4; // $b vaut 4 automatiquement

4.Déclaration et utilisation des tableaux.

Il existe plusieurs sortes de tableau en PHP.
Commençons par le plus simple : **le tableau indicé**

1.le tableau indicé

1.Indice explicite

Exemple de déclaration et affectation avec un indice explicite :

```
$notes[0]=4;
$notes[1]=12;
$notes[3]=19;
```

Un tableau est une variable qui possède un indice entre crochets. Les indices commencent à 0.
Pour afficher le contenu d'un tableau il faut utiliser la fonction **print_r()** cette fonction parcourt automatiquement tout le tableau.

Ce qui donne `print_r($notes);`
Résultat à l'écran : `Array ( [0] => 4 [1] => 12 [3] => 19 );`

On peut utiliser **echo** pour afficher un élément précis d'un tableau.
On pourrait aussi (si on est torturé… Mais cela reste toujours possible ;-)) faire une boucle avec un echo et ainsi réinventer la roue ;-)

2.Incrémentation automatique

On peut initialiser aussi un tableau sans indiquer les indices. Dans ce cas PHP part de l'indice 0 et incrémente automatiquement l'indice.

Ce qui donne :

```
$notes[] = 14;
$notes[] = 12;
$notes[] = 19;
```

Avantage -> On n'a pas à gérer les indices (ce principe est utiliser par exemple sur les sites marchands pour les caddies)

3.Initialisation avec Array

Il existe une autre méthode :

```
$notes=array(13,16,18,14);
```

Avantage -> plus compacte

2.**le tableau associatif**

Avec un tableau associatif, on ne parle plus d'indice mais de clef.

Exemple :

```
 $notes['lucie'] = 15;
 $notes['franck'] = 13;
 $notes['anne'] = 19;
 $notes['claire'] = 19;
```

ou encore

```
$notes=array('lucie'=>15,'alain'=>16,'anne'=>8);
print_r($notes);
```

ce qui donne comme résultat :

```
Array ( [lucie] => 15 [alain] => 16 [anne] => 8 )
```

Pour afficher la note d'alain on fera :

```
echo $notes['alain'];
```

Avantage -> plus parlant et intuitif.

3.**le tableau matriciel**

C'est un tableau qui a plus d'une colonne. (tableau à deux entrées par exemple).
Il n'y a pas de syntaxe particulière. L'idée est de considérer qu'un tableau peut contenir comme élément ... un tableau !

Exemple :

```
// Notes par matière et par élève
// Tableau à 2 dimensions
$notes['paul']['anglais']=12;
$notes['paul']['math']=5;
$notes['lucie']['math']=12;

print_r($notes);
```

Résultat

```
Array
(
    [paul] => Array
        (
            [anglais] => 12
            [math] => 5
        )

    [lucie] => Array
        (
            [math] => 12
        )

)
```

On voit bien que le tableau notes avec la clef **paul** contient lui même un tableau ayant deux éléments dont les clefs sont : anglais et math.
Remarque : une table dans une base de données est au final un tableau à 2 dimensions.

4.**Autres tableaux**

Il existe des tableaux déjà remplit par le système.
Exemple $_SERVER

Ces tableaux sont en générale préfixe d'un _ et sont en majuscule.
On peut afficher ce tableau comme un tableau classique avec **print_r**.

```
<?php
echo "<pre>";
print_r($_SERVER);
echo "</pre>";
?>
```

Ce qui donne (chez moi):

```
Array
(
    [HTTP_HOST] => localhost
    [HTTP_USER_AGENT] => Mozilla/5.0 (Macintosh; Intel Mac OS X 10_7_4) AppleWebKit/
536.25 (KHTML, like Gecko) Version/6.0 Safari/536.25
    [HTTP_ACCEPT] => text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
    [HTTP_ACCEPT_LANGUAGE] => fr-fr
    [HTTP_ACCEPT_ENCODING] => gzip, deflate
    [HTTP_COOKIE] => SQLiteManager_currentLangue=1
    [HTTP_CONNECTION] => keep-alive
    [PATH] => /usr/bin:/bin:/usr/sbin:/sbin
    [SERVER_SIGNATURE] =>
    [SERVER_SOFTWARE] => Apache/2.2.21 (Unix) mod_ssl/2.2.21 OpenSSL/0.9.8r DAV/2 PHP/
5.3.6
    [SERVER_NAME] => localhost
    [SERVER_ADDR] => 127.0.0.1
    [SERVER_PORT] => 80
    [REMOTE_ADDR] => 127.0.0.1
    [DOCUMENT_ROOT] => /Applications/MAMP/htdocs
    [SERVER_ADMIN] => you@example.com
    [SCRIPT_FILENAME] => /Applications/MAMP/htdocs/PhpProject1/action2.php
    [REMOTE_PORT] => 53601
    [GATEWAY_INTERFACE] => CGI/1.1
    [SERVER_PROTOCOL] => HTTP/1.1
    [REQUEST_METHOD] => GET
    [QUERY_STRING] =>
    [REQUEST_URI] => /PhpProject1/action2.php
    [SCRIPT_NAME] => /PhpProject1/action2.php
    [PHP_SELF] => /PhpProject1/action2.php
    [REQUEST_TIME] => 1346417762
    [argv] => Array
        (
        )

    [argc] => 0
)
```

5. **Recherche dans un tableau**
    1. **array_key_exists(nomDeLaClef, nomDuTableau)**

Permet de savoir si une ‘clef’ existe dans un tableau associatif.
Cette fonction retourne vrai ou faux.

```php
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <title></title>
    </head>
    <body>
        <?php
           // Tableau et recherche
           $livres = array (
                'auteurNom' => 'PELT',
                'auteurPrenom' => 'Jean-Marie',
                'nomLivre' => 'Mes plus belles Histoires de Plantes');

            if (array_key_exists('auteurNom', $livres))
            {
                echo 'La clé "auteurNom" est une clef valide !';
            }

            if (array_key_exists('titreLivre', $livres))
            {
                echo 'La clé "titreLivre" est une clef valide !';
            }
        ?>
    </body>
</html>
```

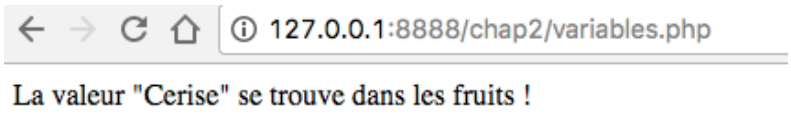
2.**in_array(nomElement, nomDuTableau)**

Permet de savoir si une ‘valeur’ existe dans un tableau.
Cette fonction retourne vrai ou faux.

```php
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <title></title>
    </head>
    <body>
      <?php
            $fruits = array ('Banane', 'Pomme', 'Poire', 'Cerise', 'Fraise', 'Framboise');

            if (in_array('Myrtille', $fruits))
            {
                echo 'La valeur "Myrtille" se trouve dans les fruits !';
            }

            if (in_array('Cerise', $fruits))
            {
                echo 'La valeur "Cerise" se trouve dans les fruits !';
            }
        ?>
    </body>
</html>
```



3.**array_search(nomElement, nomDuTableau)**

Permet de savoir si une ‘clef’ existe dans un tableau.
Cette fonction retourne l’indice de l’élément recherché ou la clef suivant le type tableau.
Si la fonction ne trouve pas l’élément elle renvoie false.

```php
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <title></title>
    </head>
    <body>
      <?php
          $fruits = array ('Banane', 'Pomme', 'Poire', 'Cerise', 'Fraise', 'Framboise');

        $position = array_search('Fraise', $fruits);
        echo '"Fraise" se trouve en position ' . $position . '<br />';

        $position = array_search('Banane', $fruits);
        echo '"Banane" se trouve en position ' . $position;

        $position = array_search('Bananes', $fruits);
        if ($position) echo '"Banane" se trouve en position ' . $position;
        ?>
    </body>
</html>

```

```php
<?php
$array = array("voiture" => 'blue', "roue" => 'red', "siege" => 'green', "plaque" => 'red');
 $key = array_search('red', $array);   // $key = 1;
 print($key);
?>
```

Vous remarquerez qu’il s’agit d’un tableau associatif.
>> Voir diaporama formulaire <<

5.Les constantes

Elles ne sont pas précédés du signe $ et comme leur nom l'indique ne peuvent changer de valeur.

2 domaines d'utilisation:
-> configuration
-> configuration linguistique

Syntaxe :
```
<?php
define ("CHEMIN_IMAGES","/images/");
echo CHEMIN_IMAGES;
?>
```

6.Les opérateurs arithmétiques

Symbole	Définition
+	Addition
-	Soustraction
/	Division
*	Multiplication
%	Modulo:reste d'une division
++	Incrément
--	Décrément

Exemple :
```
$a=4; $b=8; $c=$a+$b;
$c++; $reste=$b%$a;
$f=$a*($b-2); Attention aux priorités des opérateurs
$g=9; $h=3; $g+=$f; $g*=$f;
```

7.Les opérateurs de comparaison

Symbole	Définition
==	Egal
<	Inférieur strict
>	Supérieur strict
<=	Inférieur ou égal
>=	Supérieur ou égal
!=	Différent

8.Les opérateurs logiques

Symbole	Définition
&& AND	Fonction et
ll OR	Fonction OU
XOR	Fonction OU exclusif
!	Fonction Négation (inverse)

9.Les opérateurs de concaténation

Il existe deux formes une forme normale et une forme compacte.

Exemple de forme normale:

```
$nom='Dupond';
$message='Bonjour '. $nom;
```

Exemple de forme compacte:

```
$nom='Dupond';
$bon='Bonjour ';
$bon.=$nom;
```