

Langage PHP : Hypertext Preprocessor

Une des différences fondamentales entre les langages PHP et JavaScript est que le PHP est exécuté depuis un serveur distant, alors que le JavaScript est exécuté par le navigateur. Il faudra donc essayer de solliciter le moins possible le serveur PHP en répartissant judicieusement certaines actions vers le JavaScript, car le serveur gère des dizaines de sites alors que votre navigateur ne s'occupe que de l'internaute.

Pour faire fonctionner le langage PHP en local, il faudra installer MAMP, WAMP ou XAMPP. Pour les sites distants, pas d'inquiétude, la plupart des hébergeurs proposent le PHP dans leurs packs. La formule *Kilsufi* d'OVH à 21 € / an convient très bien. Même des hébergeurs gratuits comme *Free page perso* possèdent un serveur PHP. Il suffira donc de mettre vos pages PHP en ligne à l'aide d'un client FTP, puis de les lire avec votre navigateur.

Les navigateurs ne sachant pas interpréter du code PHP, toute page contenant le moindre morceau de code PHP doit avoir l'extension *.php* afin d'indiquer au serveur PHP qu'il faut traduire les blocs de code PHP en HTML, CSS ou JavaScript, c'est à dire en langages compréhensibles par les navigateurs.

« Hello world ! » en PHP

```
<!doctype html>
<html>
  <body>
    <?php
    echo "Hello world!";
    ?>
  </body>
</html>
```

Notez que :

- Le code PHP est placé directement dans le code source HTML, il est encadré par deux balises spéciales qui sont destinées à être reconnues par l'interpréteur PHP : **<?php** et **?>**
- Le code PHP peut être positionné n'importe où : à l'intérieur d'une balise, avant le doctype, dans la zone `<head>` ou `<body>`

Quand on affiche, depuis un navigateur, le code source d'une page Web générée par PHP, on ne voit que le code HTML résultant :

```
<!doctype html>
<html>
  <body>
    Hello world!
  </body>
</html>
```

La commande *echo* permet donc d'écrire dans une page web. Elle génère du code HTML précisément à l'endroit où se trouve cette commande.

Syntaxe

La syntaxe de PHP est proche de celle du langage C. On place traditionnellement une instruction par ligne terminée par un point-virgule. Optez pour l'indentation qui augmente la lisibilité du code. Ayez à l'esprit que dans une page PHP, on peut trouver mélangés, du HTML, du CSS, du PHP et du JavaScript. Le code doit donc être lisible du code, surtout si l'on doit le transmettre à d'autres développeurs.

Variables

Les variables en PHP ont plusieurs types :

- integer : variable numérique entière

- double : variable numérique en double précision
- string : variable de type chaîne de caractères
- array : variable de type tableau

Une variable peut changer de type selon ses affectations au cours du code :

```
$glop = "0"; // $glop est une chaîne de caractères (ASCII 48)
$glop++; // $glop devient le caractère "1" (ASCII 49)
$glop = $glop + 1.3; // $glop devient variable numérique double précision (2.3)
$glop = 5 + "10 petits choux"; // $glop devient un entier (15)
```

Ces choses là, parfois bien pratiques, sont impossibles à faire dans des langages rigoureux comme le C.

Opérateurs et expressions

Opérateurs arithmétiques

Ces opérateurs sont très traditionnels en informatique. Même un tableur les utilise !

```
$toto + $titi Addition      Somme de $toto et $titi
$toto - $titi Soustraction  Reste de la différence de $titi et $toto
$toto * $titi Multiplication Produit de $toto par $titi
$toto / $titi Division      Quotient de $toto par $titi
```

Opérateurs sur les chaînes : concaténation

La concaténation, c'est la juxtaposition de chaînes de caractères. Par exemple, si je concatène « 2 » et « 2 » ça donne « 22 ». En PHP, l'opérateur de concaténation est le point.

```
$toto = "Bonjour ";
$titi = $toto . "les amis !"; // maintenant $titi contient "Bonjour les amis !"
```

Il est à noter que vous pouvez spécifier une chaîne en l'encadrant entre des apostrophes (quotes) ou des doubles guillemets. Dans le second cas, PHP tentera d'interpréter le contenu de la chaîne ce qui diminuera la performance. La troisième solution qui utilise la concaténation permet de lire le contenu de la variable \$toto tout en conservant la performance.

```
$toto = 'les amis';
echo 'Coucou $toto'; // Affiche Coucou $toto
echo "Coucou $toto"; // Affiche Coucou les amis
echo 'Coucou ' . $toto; // Affiche Coucou les amis
```

Opérateurs logiques

Chose originale avec PHP, il existe deux versions des opérateurs logiques ET et OU, avec des précédences différentes.

```
$toto and $titi Et          Résultat vrai si $toto ET $titi sont vrais
$toto or $titi Ou          Résultat vrai si $toto OU $titi est vrai (ou les deux)
$toto && $titi Et          Résultat vrai si $toto ET $titi sont vrais
$toto || $titi Ou          Résultat vrai si $toto OU $titi est vrai (ou les deux)
```

Opérateurs de comparaison

Encore du classique. Sans surprise pour les habitués de la syntaxe C.

```
$toto == $titi Egal        Résultat vrai si $toto est égal à $titi
$toto != $titi Différent   Résultat vrai si $toto est différent de $titi
$toto < $titi Inférieur    Résultat vrai si $toto est strictement inférieur à $titi
$toto > $titi Supérieur    Résultat vrai si $toto est strictement supérieur à $titi
$toto <= $titi Inf ou égal Résultat vrai si $toto est inférieur ou égal à $titi
$toto >= $titi Sup ou égal Résultat vrai si $toto est supérieur ou égal à $titi
```

Instructions conditionnelles

If

If est une instruction conditionnelle fondamentale pour tous les langages. Elle permet d'exécuter une ou plusieurs instructions si une condition est remplie.

```
if ($toto > $titi)
    echo "toto est supérieur à titi"; // ne s'affichera que si $toto est supérieur à $titi
if (($toto > 4) and ($titi > 4))
    echo "toto et titi sont supérieurs à 4"; // ne s'affiche que si $toto et $titi sont supérieurs à 4
```

```
if ($toto > $titi) {
    echo "toto est supérieur à titi"; // plusieurs instructions conditionnées
    echo "toto est inférieur à titi";
}
```

If else

Le *if else* permet de spécifier un ensemble d'instructions à effectuer si la condition est remplie ou pas.

```
if ($toto > $titi) {
    echo "toto est supérieur à titi";
}
else {
    echo "toto n'est pas supérieur à titi";
}
```

Else if

Le *elseif* permet d'imbriquer plusieurs conditions. Le *elseif* en un seul mot peut être utilisé tout aussi bien que l'expression *else if* séparée par un espace.

```
if ($toto > $titi) {
    echo "toto supérieur à titi";
} elseif ($toto == $titi) {
    echo "toto égal à titi";
} else {
    echo "toto inférieur à titi";
}
```

La boucle for

La boucle *for* permet d'exécuter un certain nombre de fois une portion de code. Par conséquent, elle boucle autour d'un bloc d'instructions :

```
for ( initialisation; condition; incrémentation ) {
    bloc d'instructions à boucler;
}
```

Remarques :

- Pas de point-virgule sur la première ligne.
- Comme pour les *if*, les accolades sont facultatives si le bloc d'instruction ne contient qu'une seule ligne.

```
<?php
for( $i = 1; $i < 4; $i++ )
    echo $i . '<br>';
?>
```

Affiche :

```
1  
2  
3
```

Remarque : `$i++` est une manière raccourcie d'écrire `$i=$i+1`

L'instruction `switch`

L'instruction `switch` équivaut à une série d'instructions `if`. Elle permet clairement de comparer la même variable avec un grand nombre de valeurs différentes, et d'exécuter différentes parties de code suivant la valeur à laquelle elle est égale. C'est exactement à cela que sert l'instruction `switch`.

L'exemple suivant représente deux manières différentes d'écrire la même chose, l'une en utilisant une série de **if**, et l'autre en utilisant l'instruction **switch** :

```
<?php  
if ($i == 0) {  
    echo "i égal 0";  
} elseif ($i == 1) {  
    echo "i égal 1";  
} elseif ($i == 2) {  
    echo "i égal 2";  
}  
?>
```

Une variante avec l'instruction **switch** qui fait exactement la même chose avec parfois un affichage plus clair :

```
<?php  
switch ($i) {  
    case 0:  
        echo "i égal 0";  
        break;  
    case 1:  
        echo "i égal 1";  
        break;  
    case 2:  
        echo "i égal 2";  
        break;  
}  
?>
```