Chapitre 3
Les bases du langage PHP

1. Les balises

1.1 Syntaxe de base

Le XHTML (*eXtensible HyperText Markup Language*) est un langage sous forme de balises. Successeur du HTML, le XHTML se fonde sur la syntaxe du XML et est donc plus exigeant afin d'assurer la compatibilité aussi bien sur les ordinateurs classiques que sur les smartphones.

Vous connaissez les balises `<html>`, `<body>`, `<head>`…

Le PHP doit aussi s'écrire entre deux balises et elles sont définies comme telles :

`<?php` : marque le début du code PHP.

`?>` : marque la fin du code PHP.

Ces balises seront affichées en rouge si vous utilisez Notepad++.

Si l'on reprend l'exemple du chapitre Utilisation de WAMP, le code PHP s'écrit donc :

```
<?php
    echo '<p>Hello !</p>';
?>
```

Vous pouvez aussi l'écrire sur une seule ligne :

```
<?php echo '<p>Hello !</p>'; ?>
```

Il existait d'autres façons d'écrire ces balises.

À la place de `echo`, vous pouviez écrire :

```
<?= '<p>Hello !</p>' ?>
```

Enfin, à la place de `<?php` et `?>`, il était possible d'écrire :

```
<script language="php"> </script>
```

ou bien :

```
<% %>
```

ou bien encore :

```
<? ?>
```

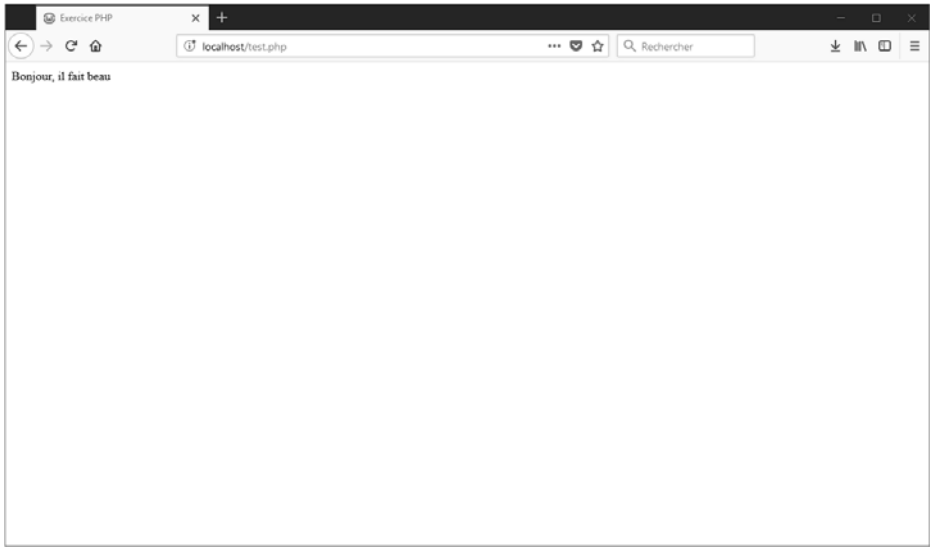
Mais depuis PHP 7, ces écritures sont devenues obsolètes.

1.2 Insertion des balises PHP dans du code XHTML

Ce qu'il faut retenir, c'est que vous pouvez insérer du code PHP à n'importe quel endroit dans le code XHTML.

```
<!DOCTYPE html>
<html lang="fr">
   <head>
     <title>Exercice PHP</title>
     <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
    </head>
  <body>
    Bonjour, il fait <?php echo 'beau'; ?>
  </body>
</html>
```

Ce code affiche ceci dans Firefox :



Si vous faites un clic droit et que vous choisissez l'option **Afficher le code source de la page**, vous obtenez le code suivant :

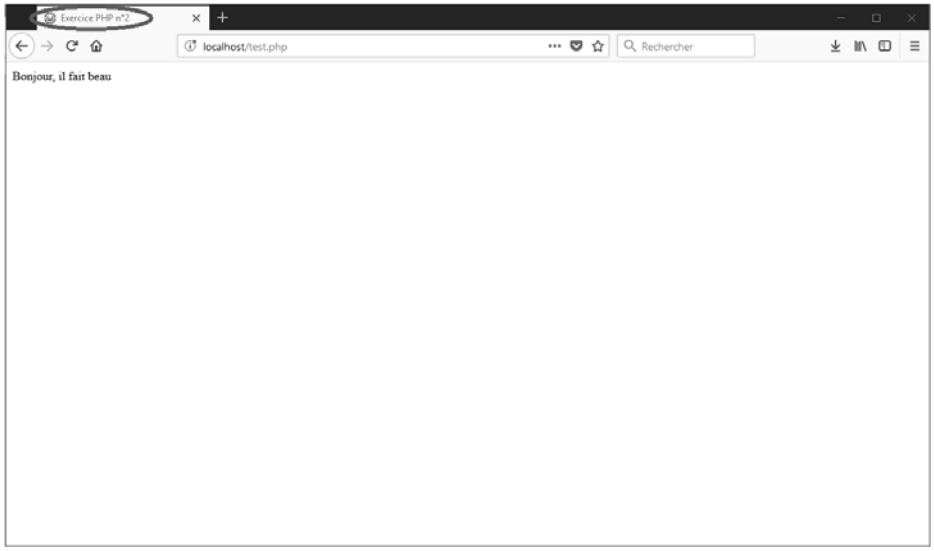
```
<!DOCTYPE html>
<html lang="fr">
    <head>
      <title>Exercice PHP</title>
      <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
    </head>
  <body>
    Bonjour, il fait beau
  </body>
</html>
```

Ceci est le code HTML reçu et interprété par le navigateur. Vous constatez que vous ne voyez pas le code serveur entre les balises PHP.

De la même façon, il est tout à fait possible de générer dynamiquement le titre de la page HTML, c'est-à-dire le contenu de la balise `<title>`.

```
<!DOCTYPE html>
<html lang="fr">
    <head>
      <title>Exercice <?php echo 'PHP n°2'; ?></title>
      <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
    </head>
   <body>
    Bonjour, il fait <?php echo 'beau'; ?>
   </body>
</html>
```

Ce code affiche ceci dans Firefox :



1.3 Envoi des données au serveur web

Il existe plusieurs instructions pour envoyer des données au serveur, c'est-à-dire pour insérer du code HTML dans une page web.

La première instruction est `echo`.

Elle s'écrit de cette façon :

```
<?php echo 'texte'; ?>
```

Il est aussi possible de l'écrire de cette façon :

```
<?php echo "texte"; ?>
```

ou bien :

```
<?php echo('texte'); ?>
```

Attention : si vous utilisez la première syntaxe, les variables ne sont pas interprétées (voir le prochain exemple).

La deuxième instruction est `print`.

Elle s'écrit de cette façon :

```
<?php print('texte'); ?>
```

Elle est équivalente à `echo`.

Il existe aussi des variantes à `print` qui sont :

– `printf()` : affiche une chaîne de caractères formatée.

– `sprintf()` : retourne une chaîne formatée.

– `vprintf()` : affiche une chaîne formatée.

– `sscanf()` : analyse une chaîne à l'aide d'un format.

– `fscanf()` : analyse un fichier en fonction d'un format.

– `flush()` : vide les tampons de sortie.

Vous remarquez qu'une instruction se termine toujours par un point-virgule.

Il est aussi possible d'écrire plusieurs instructions sur la même ligne si elles sont séparées par des points-virgules.

```
<?php echo 'texte'; ?>
```

est équivalent à :

```
<?php echo 'tex'; echo 'te'; ?>
```

et à :

```
<?php     echo 'tex';
          echo 'te';
?>
```

1.4 Insertion de code XHTML avec l'instruction echo

La fonction `echo` permettant d'insérer n'importe quel code HTML, vous pouvez écrire :

```
<?php echo '<table border="1"><tr><td>texte</td></tr></table>'; ?>
```

Ce qui a pour effet d'insérer un tableau HTML.

Vous pouvez aussi insérer une image de cette façon :

```
<?php echo '<img src="feuille.jpg" name="feuille" />'; ?>
```

Et donc vous pouvez écrire une page web complète avec l'instruction `echo` :

```
<?php
$nom="Jean";
echo '<!DOCTYPE html>';
echo '<html>';
echo '<head>';
echo '<title>PHP ENI</title>';
echo '</head>';
echo '<body>';
echo '<p>';
echo "Hello $nom !<br />";
echo 'La date du jour est le 7 janvier 2013 .';
echo '</p>';
echo '</body>';
echo '</html>';
?>
```

L'utilisation de variables permet de rendre le site dynamique, c'est-à-dire que les informations affichées peuvent provenir d'une base de données dont le contenu change régulièrement. Dans cet exemple, la variable est `$nom` et vous remarquez que la chaîne de caractères est entourée de guillemets.

1.5 Les commentaires

Les commentaires sont très utiles en PHP car ils permettent d'ignorer le code à exécuter par le serveur web. Le texte en commentaire ne sera visible que par vous et sert à donner des explications sur des lignes de code en PHP.

Il existe deux types de commentaires :

– Monoligne :

```
<?php
// commentaire monoligne
// echo 'on ne verra rien';
?>
```

– Multiligne :

```
<?php
/* commentaire multiligne
   echo 'on ne verra';
   secho 'rien';
*/
?>
```

Dans les deux cas, l'instruction `echo` n'est pas exécutée.

2. Les variables

2.1 Affectation

Une variable est une information stockée en mémoire temporairement. C'est-à-dire une zone de mémoire qui permet de stocker une information dans une page PHP et qui est automatiquement détruite lorsque la page a fini d'être exécutée.

Une variable PHP commence toujours par un `$` suivi d'une lettre puis d'une suite de lettres ou de chiffres ou du signe _, par exemple `$age`.

Attention : PHP est sensible à la casse, c'est-à-dire qu'il fait la différence entre les minuscules et les majuscules, donc `$nom` est différent de `$Nom`.

Une variable possède toujours un nom et une valeur. Par exemple, `$age = 25` affecte la valeur `25` à la variable `$age` grâce au signe =.

Il n'est pas nécessaire de définir et de typer une variable, cela se fait automatiquement.

Ainsi vous pouvez écrire :

```
<?php
$age = 25; //variable de type numérique
//puis
$age = 'toto'; //variable de type texte
?>
```

Cela ne provoque pas d'erreur. Une variable peut donc changer de type. Ici, elle passe du type integer au type string (voir section suivante).

2.2 Les types de variables

Il y a deux catégories de types de variable :

– Scalaire :
  – Les nombres entiers appelés **integer** : ce sont les nombres entiers (1, 2, 3…) ainsi que les nombres négatifs (-1, -2, -3…).
  – Les nombres décimaux appelés **float** : ce sont les nombres avec des virgules (1.35665) positifs ou négatifs. Attention, le point est utilisé comme séparateur.
  – Les chaînes de caractères appelées **string** : tout texte encadré par des guillemets (`"bonjour"`) ou par des quotes (`'bonjour'`).
  – Les **booléens** : ce type n'a que deux valeurs possibles : vrai ou faux. Ils sont notés `true` ou `false`.
– Composé :
  – Les tableaux : ce type permet de stocker plusieurs variables. Vous le verrez dans le chapitre Les fonctions et structures de contrôle - Les tableaux.
  – Les objets : type complexe décrit dans le chapitre L'objet.