REPUBLIQUE DEMOCRATIQUE DU CONGO

INSTITUT UNIVERSITAIRE D'ETUDES ET DE FORMATION EN DEVELOPPEMENT

MBANZA-NGUNGU

PHP

LANGAGE DE PROGRAMMATION ORIENTE WEB

SD Soft – 2018

Introduction

Ce qui fait le succès du Web aujourd'hui, c'est à la fois sa simplicité et sa facilité d'accès. Un internaute lambda n'a pas besoin de savoir « comment ça fonctionne derrière ». Et heureusement pour lui.

En revanche, un apprenti webmaster tel que vous doit, avant toute chose, connaître les bases du fonctionnement d'un site web.

- Qu'est-ce qu'un serveur et un client ?
- Comment rend-on son site dynamique ?
- Que signifient PHP ?

Ce premier chapitre est là pour répondre à toutes ces questions et vous montrer que vous êtes capables d'apprendre à créer des sites web dynamiques. Tous les lecteurs seront à la fin rassurés de savoir qu'ils commencent au même niveau !

Les sites statiques et dynamiques

On considère qu'il existe deux types de sites web : les sites statiques et les sites dynamiques.

Les sites statiques : ce sont des sites réalisés uniquement à l'aide des langages HTML et CSS. Ils fonctionnent très bien mais leur contenu ne peut pas être mis à jour automatiquement : il faut que le propriétaire du site (le webmaster) modifie le code source pour y ajouter des nouveautés. Ce n'est pas très pratique quand on doit mettre à jour son site plusieurs fois dans la même journée ! Les sites statiques sont donc bien adaptés pour réaliser des sites « vitrine », pour présenter par exemple son entreprise, mais sans aller plus loin. Ce type de site se fait de plus en plus rare aujourd'hui, car dès que l'on rajoute un élément d'interaction (comme un formulaire de contact), on ne parle plus de site statique mais de site dynamique.

Les sites dynamiques : plus complexes, ils utilisent d'autres langages en plus de HTML et CSS, tels que PHP et MySQL. Le contenu de ces sites web est dit « dynamique » parce qu'il peut changer sans l'intervention du webmaster ! La plupart des sites web que vous visitez aujourd'hui, y compris le Site du Zéro, sont des sites dynamiques. Le seul pré requis pour apprendre à créer ce type de sites est de déjà savoir réaliser des sites statiques en HTML et CSS.

L'objectif de ce cours est de vous rendre capables de réaliser des sites web dynamiques entièrement par vous-mêmes, pas à pas.

En effet, ceux-ci peuvent proposer des fonctionnalités bien plus excitantes que les sites statiques. Voici quelques éléments que vous serez en mesure de réaliser :

- **un espace membres** : vos visiteurs peuvent s'inscrire sur votre site et avoir accès à des sections qui leur sont réservées ;
- **un forum** : il est courant aujourd'hui de voir les sites web proposer un forum de discussion pour s'entraider ou simplement passer le temps ;
- **un compteur de visiteurs** : vous pouvez facilement compter le nombre de visiteurs qui se sont connectés dans la journée sur votre site, ou même connaître le nombre de visiteurs en train d'y naviguer !

- **des actualités** : vous pouvez automatiser l'écriture d'actualités, en offrant à vos visiteurs la possibilité d'en rédiger, de les commenter, etc. ;
- **une newsletter** : vous pouvez envoyer un e-mail à tous vos membres régulièrement pour leur présenter les nouveautés et les inciter ainsi à revenir sur votre site.

Bien entendu, ce ne sont là que des exemples. Il est possible d'aller encore plus loin, tout dépend de vos besoins. Sachez par exemple que la quasi-totalité des sites de jeux en ligne sont dynamiques. On retrouve notamment des sites d'élevage virtuel d'animaux, des jeux de conquête spatiale, etc.

Mais... ne nous emportons pas. Avant de pouvoir en arriver là, vous avez de la lecture et bien des choses à apprendre !

Commençons par la base : savez-vous ce qui se passe lorsque vous consultez une page web ?

Comment fonctionne un site web ?

Lorsque vous voulez visiter un site web, vous tapez son adresse dans votre navigateur web, que ce soit Mozilla Firefox, Internet Explorer, Opera, Safari ou un autre. Mais ne vous êtes-vous jamais demandé comment faisait la page web pour arriver jusqu'à vous ?

Il faut savoir qu'Internet est un réseau composé d'ordinateurs. Ceux-ci peuvent être classés en deux catégories.

Les clients : ce sont les ordinateurs des internautes comme vous. Votre ordinateur fait donc partie de la catégorie des clients. Chaque client représente un visiteur d'un site web.

Les serveurs : ce sont des ordinateurs puissants qui stockent et délivrent des sites web aux internautes, c'est-à-dire aux clients. La plupart des internautes n'ont jamais vu un serveur de leur vie. Pourtant, les serveurs sont indispensables au bon fonctionnement du Web.

PHP peut fonctionner seul et suffit à créer un site dynamique, mais les choses deviennent réellement intéressantes lorsqu'on le combine à un SGBD tel que MySQL. Cependant pour simplifier, oublions pour le moment MySQL et concentrons-nous sur PHP. Les clients sont incapables de comprendre le code PHP : ils ne connaissent que le HTML et le CSS. Seul le serveur est capable de lire du PHP.

Les concurrents de PHP

Parmi les concurrents de PHP, on peut citer les suivants :

ASP .NET : conçu par Microsoft, il exploite le framework (c'est-à-dire un ensemble de bibliothèques qui fournissent des services pour les développeurs) .NET bien connu des développeurs C#. Ce langage peut être intéressant si vous avez l'habitude de développer en C# .NET et que vous ne voulez pas être dépaysés.

Ruby on Rails : très actif, ce framework s'utilise avec le langage Ruby et permet de réaliser des sites dynamiques rapidement en suivant certaines conventions.

Django : il est similaire à Ruby on Rails, mais il s'utilise en langage Python.

Java et les JSP (Java Server Pages) : plus couramment appelé « JEE », il est particulièrement utilisé dans le monde professionnel. Il demande une certaine rigueur. La mise en place d'un projet JEE est traditionnellement un peu plus longue et plus lourde mais le système est apprécié des professionnels et des institutions.

Quant à PHP, il se démarque de ses concurrents par une importante communauté qui peut vous aider rapidement sur Internet si vous avez des problèmes. C'est un langage facile à utiliser, idéal pour les débutants comme pour les professionnels.

Préparer son ordinateur pour l'utilisation de PHP

Nous savons désormais que PHP s'exécute sur le serveur et que son rôle est de générer des pages web. Cependant, seul un serveur peut lire du PHP ; or votre ordinateur n'est pas un serveur.

Avec un site statique

Les webmasters qui créent des sites statiques avec HTML et CSS ont de la chance, ils ont en général déjà tous les programmes dont ils ont besoin.

- Un éditeur de texte : en théorie, un programme tel que le Bloc-notes livré avec Windows suffit, bien qu'il soit recommandé d'utiliser un outil un peu plus évolué comme Notepad++. Nous reparlerons du choix de l'éditeur à la fin de ce chapitre.
- Un navigateur web : il permet de tester la page web. Vous pouvez utiliser par exemple Mozilla Firefox, Internet Explorer, Google Chrome, Opera, Safari, ou tout autre navigateur auquel vous êtes habitués pour aller sur le web. Il est conseillé de tester son site régulièrement sur différents navigateurs.

Avec un site dynamique

Pour que votre ordinateur puisse lire du PHP, il faut qu'il se comporte comme un serveur. Rassurez-vous, vous n'avez pas besoin d'acheter une machine spéciale pour cela : il suffit simplement d'installer les mêmes programmes que ceux que l'on trouve sur les serveurs qui délivrent les sites web aux internautes.

Voici les programmes dont nous allons avoir besoin :

Apache : c'est ce qu'on appelle un serveur web. Il s'agit du plus important de tous les programmes, car c'est lui qui est chargé de délivrer les pages web aux visiteurs. Cependant, Apache ne gère que les sites web statiques (il ne peut traiter que des pages HTML). Il faut donc le compléter avec d'autres programmes.

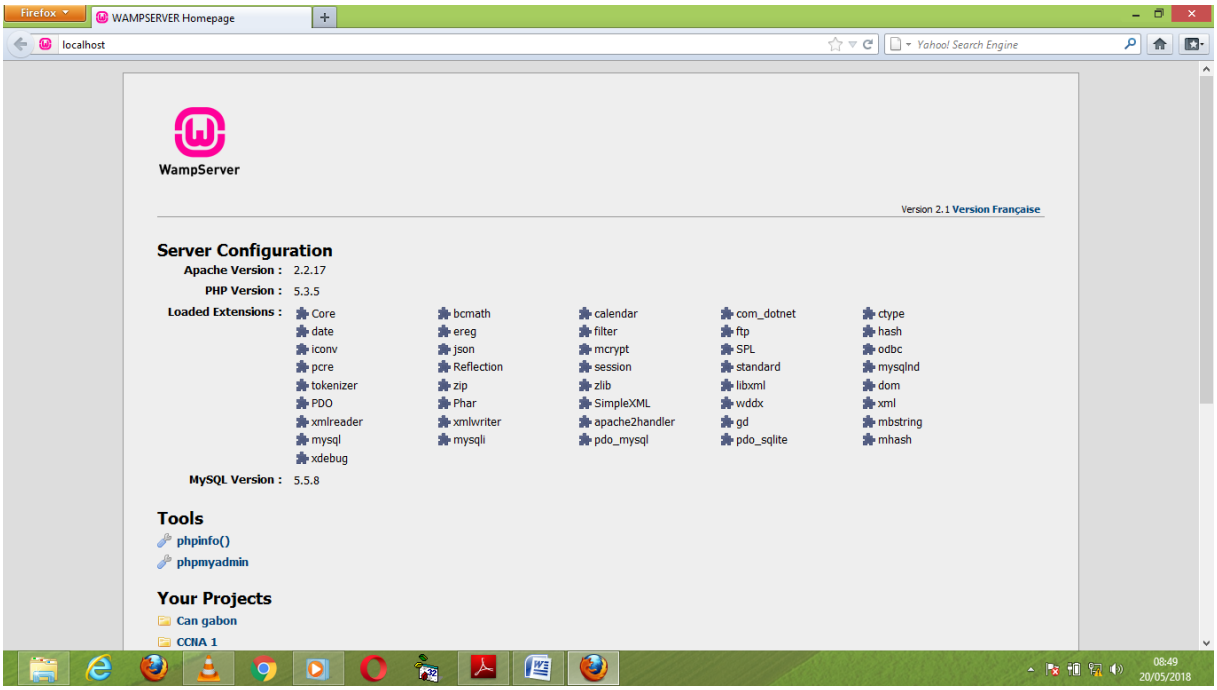
PHP : c'est un plug-in pour Apache qui le rend capable de traiter des pages web dynamiques en PHP. En clair, en combinant Apache et PHP, notre ordinateur sera capable de lire des pages web en PHP.

MySQL : c'est le logiciel de gestion de bases de données dont je vous ai parlé en introduction. Il permet d'enregistrer des données de manière organisée (comme la liste des membres de votre site). Nous n'en aurons pas besoin immédiatement, mais autant l'installer de suite.

Tous ces éléments qui vont nous aider à créer notre site dynamique sont libres et gratuits. Certes, il en existe d'autres (parfois payants), mais la combinaison Apache

+ PHP + MySQL est la plus courante sur les serveurs web, à tel point qu'on a créé des « packs » tous prêts qui contiennent tous ces éléments.

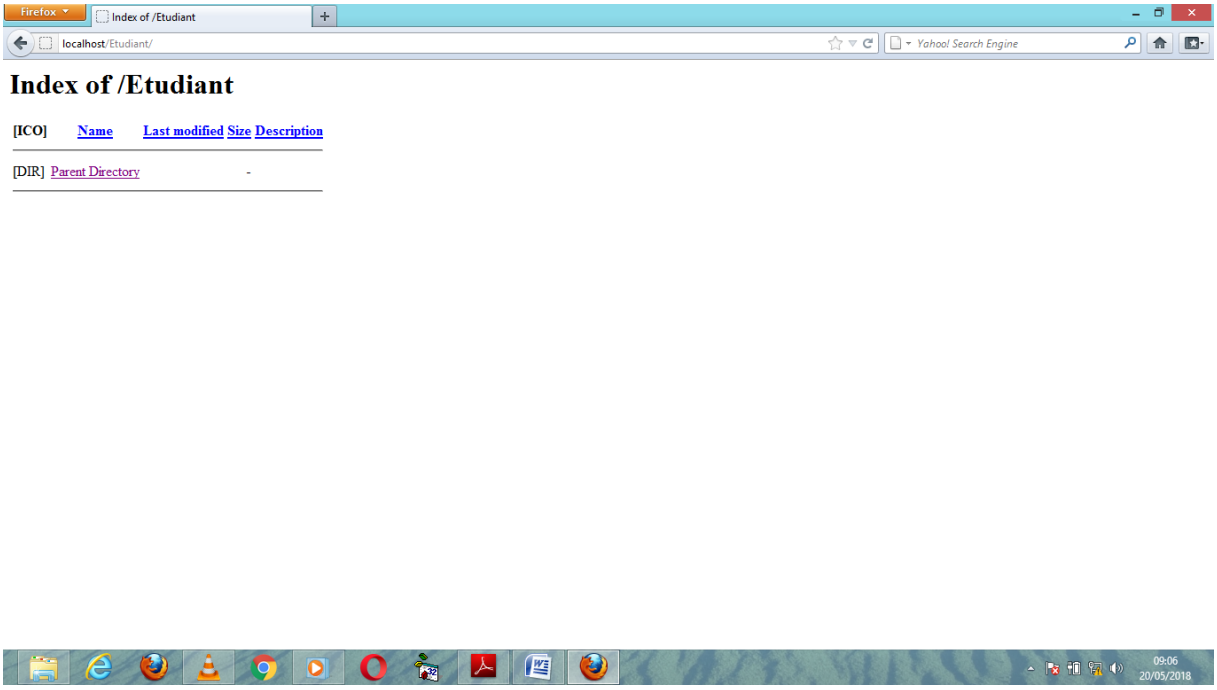
Nous allons utiliser le logiciel **WAMPSERVER** qui contient tous ces éléments cités ci haut.



Après l'installation de **Wampserver**, un dossier se crée dans la racine nommé **Wamp**. Nous devons ouvrir ce dossier et ouvrir ensuite le dossier **www**.

Une fois dans ce dossier, nous allons créer un dossier nommé « **Etudiant** ».

Maintenant, nous pouvons créer les pages web à l'intérieur de ce dossier. Mais si nous cliquons sur le lien « Etudiant » nous n'aurons qu'une page web vide.



Choix d'un éditeur

Pour coder en PHP, nous avons besoin d'un éditeur de texte.

En réalité, il existe plusieurs éditeurs pour écrire du texte en PHP.

Parmi ces éditeurs, nous pouvons citer :

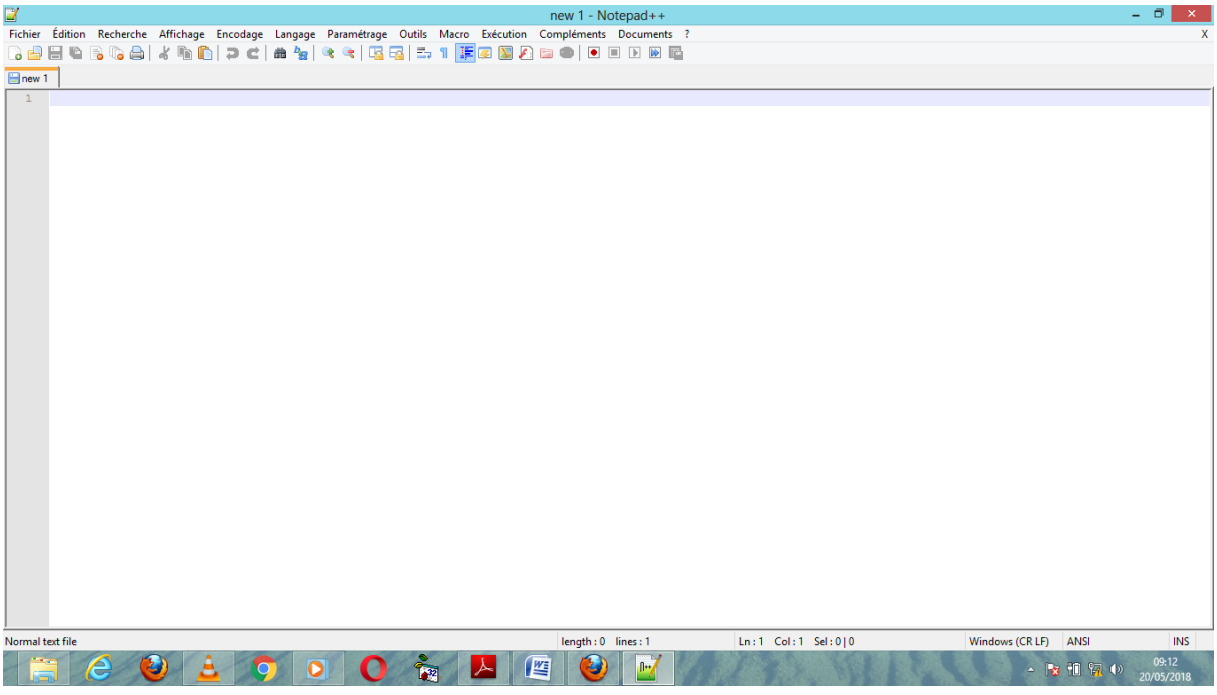
- Komodo
- Notepad++
- gEdit
- textWrangler

<u>Editeur orienté Système d'exploitation</u>

Windows : Komodo, Notepad++
Mac : Komodo, TextWrangler
Linux : Komodo, gEdit

Nous allons utiliser l'éditeur Notepad++

Il ne suffit pas "simplement" d'écrire du texte dans l'éditeur, il faut aussi donner des instructions à l'ordinateur.



Nous allons mettre les codes HTML, dans cet espace en respectant cette structure :

```
<!DOCTYPE html>
<html>
	<head>
		<title>SD SOFT</title>
		<meta charset="utf-8"/>
	</head>
```

```
        <body>
            <header>
                <h1>CAN GABON TOTAL 2017</h1>
            </header>
            <ul>
                <li><a href="index.html">Accueil</a></li>
                <li><a href="palmares.html">Palmares</a></li>
                <li><a href="joueurs.html">Liste des joueurs</a></li>
            </ul>
        </body>
    </html>
```

Et après introduction de ces codes, nous allons enregistrer ce fichier sous **index.html**

Chapitre 1. Les balises PHP

Vous savez donc que le code source d'une page HTML est constitué de balises (aussi appelées tags). Par exemple, <ul> est une balise.

Le code PHP vient s'insérer au milieu du code HTML. On va progressivement placer dans nos pages web des morceaux de code

PHP à l'intérieur du HTML. Ces bouts de code PHP seront les parties dynamiques de la page, c'est-à-dire les parties qui peuvent changer toutes seules (c'est pour cela qu'on dit qu'elles sont dynamiques).

Si je vous parle de cela, ce n'est pas par hasard. Pour utiliser du PHP, on va devoir introduire une nouvelle balise... et celle-ci est un peu spéciale. Elle commence par **<?php** et se termine par **?>** ; c'est à l'intérieur que l'on mettra du code PHP, ce que je vais vous apprendre tout au long de ce cours.

Insérer une balise PHP au milieu du code HTML

La balise PHP que nous venons de découvrir s'insère au milieu du code HTML comme je vous l'ai dit plus tôt.

Le code PHP peut être insérer partout dans les balises HTML voir l'exemple ci-après :

```
    <!DOCTYPE html>
    <html>
        <head>
            <title>SD SOFT<?php /* insérer du code PHP ici */ ?> </title>
            <meta<?php /* inserer du code PHP ici */ ?>charset="utf-8"/>
        </head>
        <body>
            <header>
                <h1>CAN GABON TOTAL 2017</h1>
            </header>
            <?php /* insérer du code PHP ici */ ?>
            <ul>
                <li><a href="index.html">Accueil</a></li>
                <li><a href="palmares.html">Palmares</a></li>
```

```
                <li><a href="joueurs.html">Liste des joueurs</a></li>
            </ul>
            <?php /* inserer du code PHP ici */ ?>
        </body>
    </html>
```

Affichage du texte en PHP

Vous allez cependant un peu mieux comprendre comment le PHP fonctionne, c'est-à-dire comment il génère du code HTML. Il est indispensable de bien comprendre cela, soyez donc attentifs !

L'instruction **echo**

Le PHP est un langage de programmation, ce qui n'était pas le cas du HTML (on parle plutôt de langage de description, car il permet de décrire une page web). Si vous avez déjà programmé dans d'autres langages comme le C ou le Java, cela ne devrait pas vous surprendre.

Tout langage de programmation contient ce qu'on appelle des instructions. On en écrit une par ligne en général, et elles se terminent toutes par un point-virgule. Une instruction commande à l'ordinateur d'effectuer une action précise.

Ici, la première instruction que nous allons découvrir permet d'insérer du texte dans la page web. Il s'agit de l'instruction **echo**, la plus simple et la plus basique de toutes les instructions que vous devez connaître.

Voici un exemple d'utilisation de cette instruction :

```
<?php echo "Une personne sans conscience est un animal." ?>
```

Si nous voulons afficher le guillemet dans l'affichage du texte d'une page web en php, il faut mettre le texte entre l'antislash.

Ex. **<?php** echo "Une personne sans \"conscience\" est un animal." **?>**

Enregistrer une page PHP

Je vous ai expliqué comment faire dans le chapitre précédent mais un petit rappel ne peut pas faire de mal.

Enregistrez la page avec l'extension **.php**, par exemple **etudiant.php**, dans le dossier tests que je vous ai fait créer.

Les commentaires en PHP

Il existe deux types de commentaires :

- les commentaires monolignes ;
- les commentaires multilignes.

Tout dépend de la longueur de votre commentaire. Je vais vous présenter les deux.

Les commentaires monolignes

Pour indiquer que vous écrivez un commentaire sur une seule ligne, vous devez taper deux slashs : « // ». Tapez ensuite votre commentaire.

Un exemple ?

Code : PHP

```
<?php
    echo "J'habite en RDC."; // Cette ligne indique où j'habite
    // La ligne suivante indique mon âge
    echo "J'ai 26 ans.";
?>
```

Les commentaires multilignes

Ce sont les plus pratiques si vous pensez écrire un commentaire sur plusieurs lignes, mais on peut aussi s'en servir pour écrire des commentaires d'une seule ligne. Il faut commencer par écrire /* puis refermer par */ :

Code : PHP

```
<?php
    /* La ligne suivante indique mon âge
    Si vous ne me croyez pas...
    ... vous avez raison ;o) */
    echo "J'ai 26 ans.";
?>
```

Chapitre 2. Inclure des portions de page

Une des fonctionnalités les plus simples et les plus utiles de PHP est l'inclusion de pages. On peut très facilement inclure toute une page ou un bout de page à l'intérieur d'une autre. Cela va grandement vous faciliter la tâche en vous évitant d'avoir à copier le même code HTML plusieurs fois.

Exemple code HTML :

```
<!DOCTYPE html>
<html>
    <head>
        <title> CAN GABON TOTAL 2017</title>
        <meta charset="utf-8"/>
        <link rel="stylesheet" href="canstyle.css"/>
    </head>
    <body>
```

```
        <!-- L'en-tête -->
            <header>
                <h1 class="sdf">CAN GABON TOTAL 2017</h1>
            </header>
        <!-- Le menu -->
            <nav>
                <ul>
                    <li><a href="index.html">Accueil</a></li>
                    <li><a href="palmares.html">Palmares</a></li>
                    <li><a href="joueurs.html">Liste des joueurs</a></li>
                </ul>
            </nav>
        <!-- Le corps -->
            <section>
                <p>
                    <h1 class="jen"><em>Coupe d'Afrique des Nations</em></h1>
                    <h3>Du 14 Janvier au 05 Février</h3>
                    <h3>31 ème Edition</h3>
                    <img src="drapeau.png" alt="Drapeau de la Rdc"/>
                    <h2>Gabon 2017</h2>
                </p>
            </section>
            <br/><br/><br/>
        <!-- Le pied de page -->
            <footer>
                <p>Copyright SD SOFT 2017 - Tous droits réservés</p>
            </footer>
        </body>
</html>
```

D'une page à l'autre, ce site contiendra à chaque fois le même code pour l'en-tête, le menu et le pied de page ! En effet, seul le contenu du corps change en temps normal.

La solution d'insertion d'une page web

En PHP, nous pouvons facilement insérer d'autres pages (on peut aussi insérer seulement des morceaux de pages) à l'intérieur d'une page.

Le principe de fonctionnement des inclusions en PHP est plutôt simple à comprendre. Vous avez un site web composé de disons vingt pages. Sur chaque page, il y a un menu, toujours le même. Pourquoi ne pas écrire ce menu (et seulement lui) une seule fois dans une page menus.php ?

En PHP, vous allez pouvoir inclure votre menu sur toutes vos pages. Lorsque vous voudrez modifier votre menu, vous n'aurez qu'à modifier menus.php et l'ensemble des pages de votre site web sera automatiquement mis à jour !

Cas pratique :

Copier coller les codes du menu dans un nouveau fichier puis enregistrer le sous le nom menu.php ou menu.html

Pour faire appel à ce nouveau fichier, il faut écrire :

<?php include ("menu.php"); **?>**

Ce code suppose que votre page index.php et celles qui sont incluses (comme menu.php) sont dans le même dossier. Si le menu était dans un sous-dossier appelé connections, il aurait fallu écrire :

<?php include ("connections/pied.html"); **?>**

Chapitre 3. Les variables

Les variables sont un élément indispensable dans tout langage de programmation, et en PHP on n'y échappe pas. Ce n'est pas un truc de programmeurs tordus, c'est au contraire quelque chose qui va nous simplifier la vie. Sans les variables, vous n'irez pas bien loin.

Les variables nous permettent de retenir temporairement des informations en mémoire. Avec elles, nous allons pouvoir par exemple retenir le pseudonyme du visiteur, effectuer des calculs et bien d'autres choses !

Qu'est-ce qu'une variable ?

Rien qu'avec leur nom, vous devez vous dire que c'est quelque chose qui change tout le temps. En effet, le propre d'une variable c'est de pouvoir varier (belle lapalissade, n'est-ce pas ? ;-)). Mais qu'est-ce que c'est concrètement ?

Une variable, c'est une petite information stockée en mémoire temporairement. Elle n'a pas une grande durée de vie. En PHP, la variable (l'information) existe tant que la page est en cours de génération. Dès que la page PHP est générée, toutes les variables sont supprimées de la mémoire car elles ne servent plus à rien. Ce n'est donc pas un fichier qui reste stocké sur le disque dur mais une petite information temporaire présente en mémoire vive.

C'est à vous de créer des variables. Vous en créez quand vous en avez besoin pour retenir des informations.

Un nom et une valeur

Une variable est toujours constituée de deux éléments :

- **son nom** : pour pouvoir la reconnaître, vous devez donner un nom à votre variable. Par exemple age_du_visiteur ;
- **sa valeur** : c'est l'information qu'elle contient, et qui peut changer. Par exemple : 26.

Ici, je vous ai donné l'exemple d'une variable appelée age_du_visiteur qui a pour valeur 26.

On peut modifier quand on veut la valeur de cette variable, faire des opérations dessus, etc. Et quand on en a besoin, on l'appelle (par son nom), et elle nous dit gentiment la valeur qu'elle contient.

Par exemple, on peut imaginer l'échange suivant :

- « Hep ! Toi, la variable age_du_visiteur, que contiens-tu ? »
- « 26 »
- « Merci ! »

Vous allez voir que ces petites bébêtes, même si elles peuvent vous sembler encore un peu floues, seront vraiment indispensables pour votre site en PHP.

Par exemple, vous pourrez retenir temporairement le nom du visiteur. Dans une variable nom_du_visiteur, vous stockerez son pseudo, mettons « Rayth19 ». Dès que vous en aurez besoin vous pourrez l'utiliser, notamment pour afficher un message de bienvenue personnalisé : « Salut Rayth19 ! Bienvenue sur mon site ! ».

Les différents types de variables

Les variables sont capables de stocker différents types d'informations. On parle de types de données. Voici les principaux types à connaître.

- Les chaînes de caractères (string) : les chaînes de caractères sont le nom informatique qu'on donne au texte. Tout texte est appelé chaîne de caractères. En PHP, ce type de données a un nom : string. On peut stocker des textes courts comme très longs au besoin.

  Exemple : « Je suis un texte ». Une chaîne de caractères est habituellement écrite entre guillemets ou entre apostrophes (on parle de guillemets simples) : 'Je suis un texte'. Les deux fonctionnent mais il y a une petite différence que l'on va découvrir plus loin.

- Les nombres entiers (int) : ce sont les nombres du type 1, 2, 3, 4, 5, 6 etc. On compte aussi parmi eux les entiers relatifs : -1, -2, -3, -4, -5...

  Exemple : 26.

- Les nombres décimaux (float) : ce sont les nombres à virgule, comme 13,261. On peut stocker de nombreux chiffres après la virgule, ce qui devrait convenir pour la plupart des usages que vous en ferez. Attention, les nombres doivent être écrits avec un point au lieu de la virgule (c'est la notation anglaise).

  Exemple : 13.261.

- Les booléens (bool) : c'est un type très important qui permet de stocker soit vrai soit faux. Cela permet de retenir si une information est vraie ou fausse. On les utilise très fréquemment. On écrit true pour vrai, et false pour faux.

  Exemple : true.

- Rien (NULL) : aussi bizarre que cela puisse paraître, on a parfois besoin de dire qu'une variable ne contient rien. Rien du tout. On indique donc qu'elle vaut NULL. Ce n'est pas vraiment un type de données, mais plutôt l'absence de type.

Affectation d'une valeur à une variable

Code PHP :

<?php $age_du_visiteur = 26 ; **?>**

Avec ce code PHP, on vient en fait de créer une variable :

- son nom est age_du_visiteur ;
- sa valeur est 26.

Nota : *Pour le nom, évitez aussi les accents, les cédilles et tout autre symbole : PHP ne les apprécie pas trop*.

Le type string (chaîne de caractères)

Ce type permet de stocker du texte. Pour cela, vous devez entourer votre texte de guillemets doubles " " ou de guillemets simples ' ' (attention, ce sont des apostrophes).

Voici deux exemples, l'un avec des guillemets simples et l'autre avec des guillemets doubles :

Code : PHP

```
<?php
$nom_du_visiteur = "Rayth19";
$nom_du_visiteur = 'Rayth19';
?>
```

Attention, petit piège : si vous voulez insérer un guillemet simple alors que le texte est entouré de guillemets simples, il faut l'échapper comme on l'a vu précédemment en insérant un antislash devant. Il en va de même pour les guillemets doubles. Voici un exemple pour bien comprendre :

Code : PHP

```
<?php
$jen = "Mon \"nom\" est Rayth19";
$jen = 'Je m\'appelle Rayth19';
?>
```

En effet, si vous oubliez de mettre un antislash, PHP va croire que c'est la fin de la chaîne et il ne comprendra pas le texte qui suivra (vous aurez en fait un message Parse error).

Vous pouvez en revanche insérer sans problème des guillemets simples au milieu de guillemets doubles et inversement :

Code : PHP

```php
<?php
$jen = 'Mon "nom" est Rayth19';
$jen = "Je m'appelle Rayth19";
?>
```

Le type int (nombre entier)

On vient de l'utiliser pour nos exemples précédents. Il suffit tout simplement d'écrire le nombre que vous voulez stocker, sans guillemets.

Code : PHP

```php
<?php
$age_du_visiteur = 26;
?>
```

Le type float (nombre décimal)

Vous devez écrire votre nombre avec un point au lieu d'une virgule. C'est la notation anglaise.

Code : PHP

```php
<?php
$poids = 87.1;
?>
```

Le type bool (booléen)

Pour dire si une variable vaut vrai ou faux, vous devez écrire le mot true ou false sans guillemets autour (ce n'est pas une chaîne de caractères !). Je vous conseille de bien choisir le nom de votre variable pour que l'on comprenne ce que ça signifie.

Voyez vous-mêmes :

Code : PHP

```php
<?php
$je_suis_un_zero = true;
$je_suis_bon_en_php = false;
?>
```

Une variable vide avec NULL

Si vous voulez créer une variable qui ne contient rien, vous devez lui passer le mot-clé NULL (vous pouvez aussi l'écrire en minuscules : null).

Code : PHP

```
<?php
$pas_de_valeur = NULL;
?>
```

Afficher le contenu d'une variable

Vous vous souvenez que l'on peut afficher du texte avec **echo** ? On peut aussi s'en servir pour afficher la valeur d'une variable !

Code : PHP

```
<?php
$age_du_visiteur = 26;
echo $age_du_visiteur;
?>
```

Comme vous le voyez, il suffit d'écrire le nom de la variable que vous voulez afficher.

La concaténation

En fait, écrire 26 tout seul comme on l'a fait n'est pas très parlant. On aimerait écrire du texte autour pour dire : « Le visiteur a 26 ans ». La concaténation est justement un moyen d'assembler du texte et des variables.

Comment faire cela ? Les petits malins auront l'idée d'écrire trois instructions **echo** :

Code : PHP

```
<?php
$age_du_visiteur = 26;
echo "Le visiteur a ";
echo $age_du_visiteur;
echo " ans";
?>
```

Vous pouvez tester, ça fonctionne, comme vous le montre la figure suivante.

Le visiteur a 26 ans

Mais il y a plus malin. On peut tout faire sur une ligne. Pour cela, il y a deux méthodes et c'est justement maintenant que le fait d'utiliser des guillemets simples ou doubles va faire la différence.

Concaténer avec des guillemets doubles

Avec des guillemets doubles, c'est le plus simple. Vous pouvez écrire le nom de la variable au milieu du texte et il sera remplacé par sa valeur.

Concrètement, essayez ce code :

Code : PHP

```
<?php
$age_du_visiteur = 26;
echo "Le visiteur a $age_du_visiteur ans";
?>
```

Ça affiche : Le visiteur a 26 ans.

En effet, lorsque vous utilisez des guillemets doubles, les variables qui se trouvent à l'intérieur sont analysées et remplacées par leur vraie valeur. Ça a le mérite d'être une solution facile à utiliser, mais je vous recommande plutôt celle des guillemets simples, que nous allons voir dès à présent.

Concaténer avec des guillemets simples

Si vous écrivez le code précédent entre guillemets simples, vous allez avoir une drôle de surprise :

Code : PHP

```
<?php
$age_du_visiteur = 26;
echo 'Le visiteur a $age_du_visiteur ans'; // Ne marche pas
?>
```

Ça affiche : Le visiteur a $age_du_visiteur ans.

Miséricorde ! On ne peut pas concaténer du texte avec des guillemets simples ?

Eh bien si ! Mais cette fois, il va falloir écrire la variable en dehors des guillemets et séparer les éléments les uns des autres à l'aide d'un point. Regardez :

Code : PHP

```
<?php
$age_du_visiteur = 26;
echo 'Le visiteur a' .$age_du_visiteur. 'ans'; // Ne marche pas
?>
```

Ça affiche : Le visiteur a 26 ans.

Faire des calculs simples

On va maintenant faire travailler votre ordinateur, et vous allez voir qu'il encaisse les calculs sans broncher. Eh oui, PHP sait aussi faire des calculs !

Oh je vous rassure, on ne va pas faire des calculs tordus, juste des additions, des soustractions, des multiplications et des divisions. Ici comme vous vous en doutez, on ne va travailler que sur des variables qui contiennent des nombres.

Les opérations de base : addition, soustraction, multiplication et division

Les signes à connaître pour faire les quatre opérations de base (vous les trouverez sur votre pavé numérique, à droite du clavier en principe) sont représentés par le tableau suivant.

Opérateurs de base :

+ Addition

- Soustraction

***** Multiplication

/ Division

Après, pour vous en servir, ça coule de source. Voici quelques exemples :

Code : PHP

```
<?php
$nombre = 3 + 4; // $nombre prend la valeur 7
$nombre = 4 - 1; // $nombre prend la valeur 3
$nombre = 3 * 5; // $nombre prend la valeur 15
$nombre = 52 / 2; // $nombre prend la valeur 26
// Allez on rajoute un peu de difficulté
$nombre = 3 * 5 + 1; // $nombre prend la valeur 16
$nombre = (1 + 2) * 3; // $nombre prend la valeur 9
?>
```

Voici des calculs avec plusieurs variables :

Code : PHP

```
<?php
$nombre = 10;
$resultat = ($nombre + 5) * $nombre; // $resultat prend la valeur 150
?>
```

C'est de la pure logique, je ne peux rien vous dire de plus. Si vous avez compris ces bouts de code, vous avez tout compris.

Le modulo

Il est possible de faire un autre type d'opération un peu moins connu : le modulo. Cela représente le reste de la division entière.

Par exemple, 6 / 3 = 2 et il n'y a pas de reste. En revanche, 7 / 3 = 2 (car le nombre 3 « rentre » 2 fois dans le nombre 7) et il reste 1.

Vous avez fait ce type de calculs à l'école primaire, souvenez-vous !

Le modulo permet justement de récupérer ce « reste ». Pour faire un calcul avec un modulo, on utilise le symbole **%**.

Code : PHP

```
<?php
$nombre = 10 % 5; // $nombre prend la valeur 0 car la division tombe juste
$nombre = 10 % 3; // $nombre prend la valeur 1 car il reste 1
?>
```

Et les autres opérations ?

Je passe sous silence les opérations plus complexes telles que la racine carrée, l'exponentielle, la factorielle, etc. Toutes ces opérations peuvent être réalisées en PHP mais il faudra passer par ce qu'on appelle des fonctions, une notion que l'on découvrira plus tard. Les opérations basiques que l'on vient de voir sont amplement suffisantes pour la programmation PHP de tous les jours.

Chapitre 4. Les Conditions

Ce chapitre est d'une importance capitale. En effet, vous serez très souvent amenés à employer des conditions dans vos pages web PHP.

À quoi servent les conditions ? On a parfois besoin d'afficher des choses différentes en fonction de certaines données. Par exemple, si c'est le matin, vous voudrez dire « Bonjour » à votre visiteur ; si c'est le soir, il vaudra mieux dire « Bonsoir ».

C'est là qu'interviennent les conditions. Elles permettent de donner des ordres différents à PHP selon le cas. Pour notre exemple, on lui dirait : Si c'est le matin, affiche « Bonjour ». Sinon, si c'est le soir, affiche « Bonsoir ». Vous allez voir que les conditions constituent vraiment la base pour rendre votre site dynamique, c'est-à-dire pour afficher des choses différentes en fonction du visiteur, de la date, de l'heure de la journée, etc.

Voilà pourquoi ce chapitre est si important !

La structure de base : if... else

Une condition peut être écrite en PHP sous différentes formes. On parle de structures conditionnelles. Celle que je vais vous apprendre à utiliser maintenant est la principale à connaître. Nous en verrons d'autres un peu plus loin.

Pour étudier la structure **if... else**, nous allons suivre le plan qui suit.

1. Les symboles à connaître : il va d'abord falloir retenir quelques symboles qui permettent de faire des comparaisons. Soyez attentifs car ils vous seront utiles pour les conditions.

2. La structure if... else : c'est le gros morceau. Là vous allez voir comment fonctionne une condition avec if... else.

Inutile de vous dire qu'il est indispensable de bien comprendre cela.

3. Des conditions multiples : on compliquera un peu nos conditions. Vous allez voir en effet qu'on peut utiliser plusieurs conditions à la fois.

4. Le cas des booléens : nous verrons ensuite qu'il existe une façon particulière d'utiliser les conditions quand on travaille sur des booléens. Si vous ne savez pas ce que sont les booléens, revoyez le chapitre sur les variables.

5. L'astuce bonus : parce qu'il y a toujours un bonus pour récompenser ceux qui ont bien suivi jusqu'au bout !

Les symboles à connaître

Juste avant de commencer, je dois vous montrer les symboles que nous serons amenés à utiliser. Je vais vous faire un petit tableau avec ces symboles et leur signification. Essayez de bien les retenir, ils vous seront utiles !

Symbole	Signification
==	Est égal à
>	Est supérieur à
<	Est inférieur à
>=	Est supérieur ou égal à
<=	Est inférieur ou égal à
!=	Est différent de

Il y a deux symboles « égal » (==) sur la première ligne, et il ne faut pas confondre ça avec le simple = que je vous ai appris dans le chapitre sur les variables. Ici, le double égal sert à tester l'égalité, à dire « Si c'est égal à... ».

Dans les conditions, on utilisera toujours le double égal (==).

Voici ce qu'on doit écrire, dans l'ordre, pour utiliser une condition.

- Pour introduire une condition, on utilise le mot if, qui en anglais signifie « si ».
- On ajoute à la suite entre parenthèses la condition en elle-même (vous allez voir que vous pouvez inventer une infinité de conditions).
- Enfin, on ouvre des accolades à l'intérieur desquelles on placera les instructions à exécuter si la condition est remplie.

Puisqu'un exemple vaut toujours mieux qu'un long discours :

Code : PHP

```
<?php
$age = 8;
if ($age <= 12)
{
echo "Salut petit frère !";
}
?>
```

Ici, on demande à PHP : si la variable $age est inférieure ou égale à 12, affiche « Salut petit frère ! ».

Vous remarquerez que dans la quasi-totalité des cas, c'est sur une variable qu'on fait la condition.

Dans notre exemple, on travaille sur la variable $age. Ce qui compte ici, c'est qu'il y a deux possibilités : soit la condition est remplie (l'âge est inférieur ou égal à 12 ans) et alors on affiche quelque chose ; sinon, eh bien on saute les instructions entre accolades, on ne fait rien.

Bon, on peut quand même améliorer notre exemple. On va afficher un autre message si l'âge est supérieur à 12 ans :

Code : PHP

```
<?php
$age = 8;
if ($age <= 12) // SI l'âge est inférieur ou égal à 12
{
echo "Salut gamin ! Bienvenue sur mon site !<br />";
$autorisation_entrer = "Oui";
}else // SINON
{
echo "Ceci est un site pour enfants, vous êtes trop vieux pour pouvoir entrer. Au revoir !<br />";
$autorisation_entrer = "Non";
}
echo "Avez-vous l'autorisation d'entrer ? La réponse est : $autorisation_entrer";
?>
```

Bon : comment marche ce code ? Tout d'abord, j'ai mis plusieurs instructions entre accolades.

Ensuite, vous avez remarqué que j'ai ajouté le mot else (« sinon »). En clair, on demande : Si l'âge est inférieur ou égal à 12 ans, fais ceci, sinon fais cela.

Essayez ce bout de code chez vous, en vous amusant à modifier la valeur de $age (sur la première ligne). Vous allez voir que le message qui s'affiche change en fonction de l'âge que vous indiquez !

Bien entendu, vous mettez les instructions que vous voulez entre accolades. Ici par exemple, j'ai donné une valeur différente à la variable $autorisation_entrer après avoir affiché un message, valeur qui pourrait nous servir par la suite :

Code : PHP

```
<?php
if ($autorisation_entrer == "Oui") // SI on a l'autorisation d'entrer
{
// instructions à exécuter quand on est autorisé à entrer
}elseif ($autorisation_entrer == "Non") // SINON SI on n'a pas l'autorisation d'entrer
{
// instructions à exécuter quand on n'est pas autorisé à entrer
}
else // SINON (la variable ne contient ni Oui ni Non, on ne peut pas agir)
{
echo "Euh, je ne connais pas ton âge, tu peux me le rappeler s'il te plaît ?";
}
?>
```

Le cas des booléens

Si vous regardez bien le dernier code source (avec $autorisation_entrer), vous ne trouvez pas qu'il serait plus adapté d'utiliser des booléens ?

On a parlé des booléens dans le chapitre sur les variables. Vous vous souvenez ?

Ce sont ces variables qui valent soit true (vrai) soit false (faux). Eh bien, les booléens sont particulièrement utiles avec les conditions ! Voici comment on teste une variable booléenne :

Code : PHP

```
<?php
if ($autorisation_entrer == true)
{
echo "Bienvenue petit Zéro. :o)";
}elseif ($autorisation_entrer == false)
{
echo "T'as pas le droit d'entrer !";
}
?>
```

Voilà, jusque-là rien d'extraordinaire. Vous avez vu que je n'ai pas mis de guillemets pour **true** et **false**, comme je vous l'ai dit dans le chapitre sur les variables.

Mais un des avantages des booléens, c'est qu'ils sont particulièrement adaptés aux conditions.

Pourquoi ? Parce qu'en fait vous n'êtes pas obligés d'ajouter le == true. Quand vous travaillez sur une variable booléenne,

PHP comprend très bien ce que vous avez voulu dire :

Code : PHP

```
<?php
if ($autorisation_entrer)
{
echo "Bienvenue petit Zéro. :o)";
}else
{
echo "T'as pas le droit d'entrer !";
}
?>
```

PHP comprend qu'il faut qu'il vérifie si $autorisation_entrer vaut true.

En effet, si vous « lisez » la première ligne, ça donne : « SI on a l'autorisation d'entrer... ».

C'est donc un raccourci à connaître quand on travaille sur des booléens.

Des conditions multiples

Ce qu'on va essayer de faire, c'est de poser plusieurs conditions à la fois. Pour cela, on aura besoin de nouveaux mots-clés. Voici les principaux à connaître :

Mot clé	Signification	Symbole équivalent
AND	Et	&&
OR	Ou	||

La première colonne contient le mot-clé en anglais, le troisième son équivalent en symbole. Les deux fonctionnent aussi bien, mais je vous recommande d'utiliser le mot-clé de préférence, c'est plus « facile » à lire (j'espère que vous connaissez un peu l'anglais, quand même). Servez-vous de ces mots-clés pour mettre plusieurs conditions entre les parenthèses. Voici un premier exemple :

Code : PHP

```
<?php
if ($age <= 12 AND $sexe == "garçon")
{
echo "Bienvenue sur le site de SD Soft !";
}elseif ($age <= 12 AND $sexe == "fille")
{
echo "Ce n'est pas un site pour les experts, retourne jouer à la cour !";
```

```
}
?>
```

C'est tout simple en fait et ça se comprend très bien : si l'âge est inférieur ou égal à 12 ans et que c'est un garçon, on lui permet d'accéder au site de son super-héros préféré. Sinon, si c'est une fille dont l'âge est inférieur ou égal à 12 ans, on l'envoie gentiment balader (hum, hum, m'accusez pas de sexisme hein, c'était juste pour l'exemple).

Bon allez, un dernier exemple avec OR pour que vous l'ayez vu au moins une fois, et on arrête là.

Code : PHP

```
<?php
if ($sexe == "fille" OR $sexe == "garçon")
{
echo "Salut chers camarades !";
}else
{
echo "Euh, si t'es ni une fille ni un garçon, t'es qui alors ?";
}
?>
```

Swift

Permet de faciliter les conditions dans une structure longue de if....elseif... else.

Prenons l'exemple de ces codes PHP :

```
<?php
if ($note == 0)
{
echo "Tu es vraiment un gros Zéro !!!";
}
elseif ($note == 5)
{
echo "Tu es très mauvais";
}
elseif ($note == 7)
{
echo "Tu es mauvais";
}
elseif ($note == 10)
```

```
{
echo "Tu as pile poil la moyenne, c'est un peu juste...";
}
elseif ($note == 12)
{
echo "Tu es assez bon";
}
elseif ($note == 16)
{
echo "Tu te débrouilles très bien !";
}
elseif ($note == 20)
{
echo "Excellent travail, c'est parfait !";
}
else
{
echo "Désolé, je n'ai pas de message à afficher pour cette note";
}
?>
```

Comme vous le voyez, c'est lourd, long, et répétitif. Dans ce cas, on peut utiliser une autre structure plus souple : c'est switch.

Voici le même exemple avec switch (le résultat est le même, mais le code est plus adapté) :

Code : PHP

```
<?php
$note = 10;
switch ($note) // on indique sur quelle variable on travaille
{
case 0: // dans le cas où $note vaut 0
echo "Tu es vraiment un gros Zéro !!!";
break;
case 5: // dans le cas où $note vaut 5
echo "Tu es très mauvais";
break;
```

```
case 7: // dans le cas où $note vaut 7
echo "Tu es mauvais";
break;
case 10: // etc. etc.
echo "Tu as pile poil la moyenne, c'est un peu juste...";
break;
case 12:
echo "Tu es assez bon";
break;
case 16:
echo "Tu te débrouilles très bien !";
break;
case 20:
echo "Excellent travail, c'est parfait !";
break;
default:
echo "Désolé, je n'ai pas de message à afficher pour cette note";
}
?>
```

Les ternaires : des conditions condensées

Il existe une autre forme de condition, beaucoup moins fréquente, mais que je vous présente quand même car vous pourriez un jour ou l'autre tomber dessus. Il s'agit de ce qu'on appelle les ternaires.

Un ternaire est une condition condensée qui fait deux choses sur une seule ligne :

- on teste la valeur d'une variable dans une condition ;
- on affecte une valeur à une variable selon que la condition est vraie ou non.

Prenons cet exemple à base de if... else qui met un booléen $majeur à vrai ou faux selon l'âge du visiteur :

Code : PHP

```
<?php
$age = 24;
if ($age >= 18)
{
$majeur = true;
}else
```

```
{
$majeur = false;
}
?>
```

On peut faire la même chose en une seule ligne grâce à une structure ternaire :

Code : PHP

```
<?php
$age = 24;
$majeur = ($age >= 18) ? true : false;
?>
```

Chapitre 5. Les Boucles

Dans la série des éléments de base de PHP à connaître absolument, voici les boucles! Répéter des instructions, ça, l'ordinateur sait faire (et en plus, il ne bronche jamais) !

Imaginez que vous êtes en train de créer le forum de votre site. Sur une page, on affiche par exemple une trentaine de messages.

Il serait bien trop long et répétitif de dire « Affiche le message 1 et le nom de son auteur », « Affiche le message 2 et le nom de son auteur », « Affiche le message 3 et le nom de son auteur », etc. Pour éviter d'avoir à faire cela, on peut utiliser un système de boucle qui nous permettra de dire une seule fois : « Affiche 30 messages et le nom de leur auteur à chaque fois ».

Bien entendu, nous n'allons pas pouvoir apprendre à créer le forum de votre site dans ce chapitre, il est encore trop tôt.

Néanmoins, prenez bien le temps de comprendre le fonctionnement des boucles car nous en aurons besoin tout au long de ce cours. Ce n'est pas bien compliqué, vous allez voir !

Une boucle simple : while

Qu'est-ce qu'une boucle ? C'est une structure qui fonctionne sur le même principe que les conditions (if... else). D'ailleurs, vous allez voir qu'il y a beaucoup de similitudes avec le chapitre sur les conditions.

Concrètement, une boucle permet de répéter des instructions plusieurs fois. En clair, c'est un gain de temps, c'est très pratique, et bien souvent indispensable. On peut si vous voulez présenter le principe dans le schéma suivant.

Voilà ce qui se passe dans une boucle :

1. comme d'habitude, les instructions sont d'abord exécutées dans l'ordre, de haut en bas (flèche rouge) ;

2. à la fin des instructions, on retourne à la première (flèche verte) ;
3. on recommence à lire les instructions dans l'ordre (flèche rouge) ;
4. et on retourne à la première (flèche verte) ;
5. etc., etc.

Le seul hic dans ce schéma, c'est que ça ne s'arrête jamais ! Les instructions seraient réexécutées à l'infini !

C'est pour cela que, quel que soit le type de boucle (while ou for), il faut indiquer une condition. Tant que la condition est remplie, les instructions sont réexécutées. Dès que la condition n'est plus remplie, on sort enfin de la boucle (ouf !).

Voici comment faire avec une boucle simple : while.

Code : PHP

```
<?php
while ($continuer_boucle == true)
{
// instructions à exécuter dans la boucle
}
?>
```

while peut se traduire par « tant que ». Ici, on demande à PHP : TANT QUE $continuer_boucle est vrai, exécuter ces instructions.

Les instructions qui sont répétées en boucle se trouvent entre les accolades { et }. Mais bon là je ne vous apprends rien, vous commencez à avoir l'habitude de voir des accolades de partout. ;-)

Ce n'est pas beaucoup plus compliqué que ça, il n'y a guère plus de choses à savoir. Cependant, je vais quand même vous montrer un ou deux exemples d'utilisation de boucles, pour que vous voyiez à quoi ça peut servir...

Pour notre premier exemple, on va supposer que vous avez été punis et que vous devez recopier 26 fois « Je ne dois pas regarder les mouches voler quand j'apprends le PHP. ».

Avant, il fallait prendre son mal en patience et ça prenait des heures... Maintenant, avec PHP, on va faire ça en un clin d'oeil !

Regardez ce code :

Code : PHP

```
<?php
$nombre_de_lignes = 1;
while ($nombre_de_lignes <= 26)
{
echo 'Je ne dois pas regarder les matchs quand j\'apprends le PHP.<br />';
$nombre_de_lignes++; // $nombre_de_lignes = $nombre_de_lignes +1
```

```
}
?>
```

La boucle pose la condition : TANT QUE $nombre_de_lignes est inférieur ou égal à 26.

Dans cette boucle, il y a deux instructions :

- le echo, qui permet d'afficher du texte en PHP. À noter qu'il y a une balise HTML
 à la fin : cela permet d'aller à la ligne. Vu que vous connaissez le HTML, ça n'a rien de surprenant : chaque phrase sera écrite sur une seule ligne ;
- ensuite, une instruction bizarre : $nombre_de_lignes++; Quésaco ? Regardez mon commentaire : c'est exactement la même chose. En fait, c'est une façon plus courte d'ajouter 1 à la variable. On appelle cela l'incrémentation (ce nom barbare signifie tout simplement que l'on a ajouté 1 à la variable).

Chaque fois qu'on fait une boucle, la valeur de la variable augmente : 1, 2, 3, 4... 25, 26... Dès que la variable atteint 27, on arrête la boucle. Et voilà, on a écrit 26 lignes en un clin d'oeil.

Si la punition avait été plus grosse, pas de problème ! Il aurait suffi de changer la condition (par exemple, mettre « TANT QUE c'est inférieur ou égal à 500 » pour l'écrire 500 fois).

Nota : *Il faut TOUJOURS s'assurer que la condition sera fausse au moins une fois. Si elle ne l'est jamais, alors la boucle s'exécutera à l'infini !*

PHP refuse normalement de travailler plus d'une quinzaine de secondes. Il s'arrêtera tout seul s'il voit que son travail dure trop longtemps et affichera un message d'erreur.

Code : PHP

```
<?php
$nombre_de_lignes = 1;
while ($nombre_de_lignes <= 26)
{
echo 'Ceci est la ligne n°' . $nombre_de_lignes . '<br />';
$nombre_de_lignes++;
}
?>
```

Voilà, c'est tout bête, et cet exemple ressemble beaucoup au précédent. La particularité, là, c'est qu'on affiche à chaque fois la valeur de $nombre_de_lignes (ça vous permet de voir que sa valeur augmente petit à petit).

Une boucle plus complexe : for

Mais non, n'ayez pas peur voyons.

Il ne vous arrivera rien de mal : ici le mot « complexe » ne veut pas dire « compliqué ».

for est un autre type de boucle, dans une forme un peu plus condensée et plus commode à écrire, ce qui fait que for est assez fréquemment utilisé.

Cependant, sachez que for et while donnent le même résultat et servent à la même chose : répéter des instructions en boucle.

L'une peut paraître plus adaptée que l'autre dans certains cas, cela dépend aussi des goûts.

Alors, comment ça marche un for ? Ça ressemble beaucoup au while, mais c'est la première ligne qui est un peu particulière.

Pour que vous voyiez bien la différence avec le while, je reprends exactement l'exemple précédent, mais cette fois avec un for :

Code : PHP

```
<?php
for ($nombre_de_lignes = 1; $nombre_de_lignes <= 26;
$nombre_de_lignes++)
{
echo 'Ceci est la ligne n°' . $nombre_de_lignes . '<br />';
}
?>
```

Que de choses dans une même ligne !

Bon, vous vous en doutez, je ne vais vous expliquer que la ligne du for, le reste n'ayant pas changé.

Après le mot for, il y a des parenthèses qui contiennent trois éléments, séparés par des points-virgules ;.

Décrivons chacun de ces éléments.

- Le premier sert à l'initialisation. C'est la valeur que l'on donne au départ à la variable (ici, elle vaut 1).
- Le second, c'est la condition. Comme pour le while, tant que la condition est remplie, la boucle est réexécutée. Dès que la condition ne l'est plus, on en sort.
- Enfin, le troisième c'est l'incrémentation, qui vous permet d'ajouter 1 à la variable à chaque tour de boucle.

Les deux derniers codes donnent donc exactement le même résultat. Le for fait la même chose que le while, mais rassemble sur une seule ligne tout ce qu'il faut savoir sur le fonctionnement de la boucle.

La boucle while est plus simple et plus flexible : on peut faire tous les types de boucles avec mais on peut oublier de faire certaines étapes comme l'incrémentation de la variable.

En revanche, for est bien adapté quand on doit compter le nombre de fois que l'on répète les instructions et il permet de ne pas oublier de faire l'incrémentation pour augmenter la valeur de la variable !

Chapitre 6. Les Fonctions

En PHP, on n'aime pas avoir à répéter le même code. Pour pallier ce problème, nous avons découvert les boucles, qui permettent d'exécuter des instructions un certain nombre de fois. Ici nous allons découvrir un autre type de structure indispensable pour la suite : les fonctions.

Comme les boucles, les fonctions permettent d'éviter d'avoir à répéter du code PHP que l'on utilise souvent. Mais alors que les boucles sont de bêtes machines tout juste capables de répéter deux cents fois la même chose, les fonctions sont des robots « intelligents » qui s'adaptent en fonction de ce que vous voulez faire et qui automatisent grandement la plupart des tâches courantes.

Qu'est-ce qu'une fonction ?

Une fonction est une série d'instructions qui effectue des actions et qui retourne une valeur. En général, dès que vous avez besoin d'effectuer des opérations un peu longues dont vous aurez à nouveau besoin plus tard, il est conseillé de vérifier s'il n'existe pas déjà une fonction qui fait cela pour vous. Et si la fonction n'existe pas, vous avez la possibilité de la créer.

Dialogue avec une fonction

Voici le genre de dialogue qu'on peut avoir avec une fonction :

Toi, la fonction calculCube, donne-moi le volume d'un cube dont l'arête mesure 4 cm.

La fonction effectue les calculs demandés puis répond :

Ce cube a un volume de 64 cm^3

On donne en entrée à la fonction un paramètre sur lequel elle va faire des calculs (ici, la longueur de l'arête : 4, cela veut dire nous aurons 4 * 4 * 4) et la fonction nous retourne en sortie le résultat : 64.

Les fonctions en PHP

Nous avons jusqu'ici imaginé le dialogue avec une fonction représentée par un robot, ce qui n'est pas très intéressant. Revenons aux choses sérieuses et concrètes.

Appeler une fonction

En PHP, comment appelle-t-on une fonction ? Par son nom, JenSB ! Par exemple :

Code : PHP

```
<?php
calculCube();
```

```
?>
```

Comme vous le voyez, j'ai simplement écrit le nom de la fonction, suivi de parenthèses vides, puis de l'inévitable point-virgule. En faisant cela, j'appelle la fonction calculCube mais je ne lui envoie aucune information, aucun paramètre.

Certaines fonctions peuvent fonctionner sans paramètres, mais elles sont assez rares. Dans le cas de calculCube, ça n'a pas de sens de l'appeler sans lui donner la longueur de l'arête du cube pour faire le calcul !

Si on veut lui envoyer un paramètre (un nombre, une chaîne de caractères, un booléen...), il faut l'écrire entre les parenthèses :

Code : PHP

```
<?php
calculCube(4);
?>
```

Ainsi, calculCube saura qu'elle doit travailler avec le nombre 4.

Récupérer la valeur de retour de la fonction

Maintenant que nous savons appeler une fonction et même lui envoyer plusieurs paramètres, il faut récupérer ce qu'elle nous retourne si toutefois elle retourne quelque chose. Il y a en effet deux types de fonctions :

- celles qui ne retournent aucune valeur (ça ne les empêche pas d'effectuer des actions) ;
- celles qui retournent une valeur.

Si la fonction ne retourne aucune valeur, il n'y a rien de plus à faire que dans les codes précédents. La fonction est appelée, fait son travail, et on ne lui demande rien de plus.

En revanche, si la fonction retourne une valeur (comme ça devrait être le cas pour calculCube), on la récupère dans une variable, comme ceci :

Code : PHP

```
<?php
$volume = calculCube(4);
?>
```

Sur une ligne comme celle-ci, il se passe en fait les deux choses suivantes (dans l'ordre, et de droite à gauche) :

1. la fonction calculCube est appelée avec le paramètre 4 ;
2. le résultat renvoyé par la fonction (lorsqu'elle a terminé) est stocké dans la variable $volume.

La variable $volume aura donc pour valeur 64 après l'exécution de cette ligne de code !

Les fonctions prêtes à l'emploi de PHP

PHP propose des centaines et des centaines de fonctions prêtes à l'emploi. Sur le site officiel, la documentation PHP les répertorie toutes, classées par catégories.

Ces fonctions sont très pratiques et très nombreuses. En fait, c'est en partie là que réside la force de PHP : ses fonctions sont vraiment excellentes car elles couvrent la quasi-totalité de nos besoins. J'ai en fait remarqué que, pratiquement à chaque fois que je m'apprêtais à écrire une fonction, celle-ci existait déjà.

Voici un petit aperçu des fonctions qui existent pour vous mettre l'eau à la bouche :

- Une fonction qui permet de rechercher et de remplacer des mots dans une variable.
- Une fonction qui envoie un fichier sur un serveur.
- Une fonction qui permet de créer des images miniatures (aussi appelées thumbnails).
- Une fonction qui envoie un mail avec PHP (très pratique pour faire une newsletter !).
- Une fonction qui permet de modifier des images, y écrire du texte, tracer des lignes, des rectangles, etc.
- Une fonction qui crypte des mots de passe.
- Une fonction qui renvoie l'heure, la date...
- Etc.

Dans la plupart des cas, il faudra indiquer des paramètres à la fonction pour qu'elle sache sur quoi travailler.

Nous allons ici découvrir rapidement quelques fonctions pour vous habituer à les utiliser. Nous ne pourrons jamais toutes les passer en revue (j'ai dit qu'il y en avait des centaines et des centaines !) mais avec l'expérience de ces premières fonctions et la documentation de PHP, vous n'aurez aucun mal à aller plus loin tous seuls.

Nous allons voir quelques fonctions qui effectuent des modifications sur des chaînes de caractères et une qui permet de récupérer la date. Ce sont seulement des exemples destinés à vous habituer à utiliser des fonctions.

Traitement des chaînes de caractères

De nombreuses fonctions permettent de manipuler le texte. En voici quelques-unes qui vont vous montrer leur intérêt.

strlen : longueur d'une chaîne

Cette fonction retourne la longueur d'une chaîne de caractères, c'est-à-dire le nombre de lettres et de chiffres dont elle est constituée (espaces compris).

Code : PHP

```
<?php
$phrase = 'Bonjour les Zéros ! Je suis une phrase !';
$longueur = strlen($phrase);
echo 'La phrase ci-dessous comporte ' . $longueur . ' caractères :<br />' . $phrase;
```

```
?>
```

str_replace : rechercher et remplacer

str_replace remplace une chaîne de caractères par une autre. Exemple :

Code : PHP

```
<?php
$ma_variable = str_replace('b', 'p', 'bim bam boum');
echo $ma_variable;
?>
```

On a besoin d'indiquer trois paramètres :

1. la chaîne qu'on recherche (ici, les « b » (on aurait pu rechercher un mot aussi)) ;
2. la chaîne qu'on veut mettre à la place (ici, on met des « p » à la place des « b ») ;
3. la chaîne dans laquelle on doit faire la recherche.

str_shuffle : mélanger les lettres

Pour vous amuser à mélanger aléatoirement les caractères de votre chaîne !

Code : PHP

```
<?php
$chaine = 'Cette chaîne va être mélangée !';
$chaine = str_shuffle($chaine);
echo $chaine;
?>
```

strtolower : écrire en minuscules

strtolower met tous les caractères d'une chaîne en minuscules.

Code : PHP

```
<?php
$chaine = 'COMMENT CA JE CRIE TROP FORT ???';
$chaine = strtolower($chaine);
echo $chaine;
?>
```

À noter qu'il existe strtoupper qui fait la même chose en sens inverse : minuscules → majuscules.

Récupérer la date

Nous allons découvrir la fonction qui renvoie l'heure et la date. Il s'agit de date (un nom facile à retenir, avouez !). Cette fonction peut donner beaucoup d'informations. Voici les principaux paramètres à connaître :

Paramètre	Description
H	Heure
i	Minute
d	Jour
m	Mois
Y	Année

Nota : *attention, respectez les majuscules/minuscules pour avoir un résultat escompté.*

Si vous voulez afficher l'année, il faut donc envoyer le paramètre Y à la fonction :

Code : PHP

```
<?php
$annee = date('Y');
echo $annee;
?>
```

On peut bien entendu faire mieux, voici la date complète et l'heure :

Code : PHP

```
<?php
// Enregistrons les informations de date dans des variables
$jour = date('d');
$mois = date('m');
$annee = date('Y');
$heure = date('H');
$minute = date('i');
// Maintenant on peut afficher ce qu'on a recueilli
echo 'Bonjour ! Nous sommes le ' . $jour . '/' . $mois . '/' .
$annee . 'et il est ' .$heure. ' h '. $minute;
?>
```