Présentation et utilisation basique

Qu’est ce que PHP ?

PHP (ou Hypertext Preprocessor) est un langage de scripts généraliste et Open Source, spécialement conçu pour le développement d'applications web. Il peut être intégré facilement au HTML.

Installation de l’environnement

Pré-requis :

- PHP
- Un editeur de texte

Installation Wamp

- Téléchargez la dernière version de Wamp Server à partir du site Web officiel : http://www.wampserver.com/
- Exécutez le fichier d'installation de Wamp Server en double-cliquant dessus. Suivez les instructions à l'écran pour installer le programme.
- Une fois l'installation terminée, vous devriez voir une icône Wamp Server apparaître dans la barre des tâches de Windows. Si vous ne la voyez pas, vous pouvez la trouver dans le menu Démarrer.
- Cliquez sur l'icône Wamp Server dans la barre des tâches et sélectionnez "Start All Services". Cela démarrera Apache et MySQL, qui sont les deux composants principaux de Wamp Server.
- Ouvrez votre navigateur Web et accédez à l'adresse http://localhost/. Vous devriez voir la page d'accueil de Wamp Server s'afficher. Si c'est le cas, cela signifie que Wamp Server est bien installé et fonctionne correctement.

Vous pouvez maintenant utiliser PHP en créant des fichiers PHP dans le répertoire "www" de Wamp Server et en les ouvrant dans votre navigateur à l'aide de l'adresse http://localhost/nom-du-fichier.php.

IDE

Pour votre editeur de texte, prenez ce que vous voulez :

- VisualStudioCode
- SublimeText
- nano
- etc...

Choisissez l'IDE sur lequel vous êtes le plus à l'aise.

Pourquoi PHP 8 ?

PHP 8 a plusieurs avantage par rapport à ses anciennes versions :

- Gain de performance (20 à 30% de requêtes en plus pas seconde)
- Renforcement du typage

Ces avantages sont non négligeables car ils permettent une meilleure performance et une meilleure lisibilité du code.

Conditions

En PHP, il est possible d'exécuter du code seulement dans une ou plusieurs conditions précises, définies arbitrairement. Pour coder ces conditions, plusieurs structures existent :

- La structure if, elseif, else
- La condition ternaire
- La structure switch case

Utilisation de if

```php
$age = 24;
if ($age > 18) // Après le mot clé "if", dans les parenthèses,
//nous écrivons la condition à respecter.
{
    // Si la condition est respectée, le corps de la condition s'exécutera.
    // Dans notre cas, le programme affichera "Vous êtes majeur !"
    echo "Vous êtes majeur !";
}
```

Utilisation de elseif et else

```php
$age = 24;
if ($age <0)
{
    echo"Vous ne pouvez pas avoir un âge négatif";
}
elseif ($age > 18)
{
    echo "Vous êtes majeur !";
}
else
{
    echo "Vous êtes mineur !";
}
```

Condition ternaire

La condition ternaire est une façon plus concise décrire la structure “else if” vue précédemment, elle s'utilise avec l'opérateur ? de la manière suivante :

```
$age = 24 ;

$majeur = $age > 18 ? True : False ;
```

Dans ce cas, la variable “majeur” vaut True car la variable age est supérieur à 18, rendant ainsi la condition vraie

Le switch case

```php
$departement =75;
switch($departement) {
    case 75:
        echo "Paris";
        break
    case 93:
        echo "Seine Saint Denis";
        break
    default:
        echo "Autre departement";
        break
}
```

Boucle

Les structures de répétitions, ou boucles, permettent d'exécuter une instruction ou un groupe d’instruction de manière répétée. Il existe plusieurs type de boucles en PHP :

- Les boucles for
- Les boucles while et do while
- Les boucles foreach

Les boucles répètent une instruction, ou un groupe d’instructions, tant qu’une condition est vraie ; cette condition est définie par le développeur en amont.

Boucles for

```
for ($expression1 ; $expression2 ; $expression3)
{
    //Mes instructions a repeter
```

- $expression1 est l'initialisation d'une ou plusieurs variables servant de compteur pour la boucle.
- $expression2 est ensuite évaluée avec une valeur booléenne : si elle vaut True, la boucle continue, sinon la boucle s'arrête
- $expression3 n'est exécutée qu'à la fin de chaque itération. Il s'agit le plus souvent d'une instruction d'incrémentation de la variable compteur ($expression1)

Exemple

```php
for ($compteur = 0; $compteur < 10; $compteur = $compteur +1)
{
    echo "Bonjour";
}
// le code affichera "Bonjour" dix fois
```

Boucle while et do ... while

```php
while ($condition) {
    //Mes instruction a repeter tant que la condition est vrai
}
```

$condition est toujours évaluée avec une valeur booléenne : si elle vaut True, la boucle continue et les instructions comprises entre accolades sont exécutées, sinon la boucle s'arrête.

Si elle est toujours vraie on obtient une boucle infinie (comme pour la boucle “for”).

Exemple

```php
$n = 1;
while ($n % 7 !=0)
{
    $n = rand(1,100);
    //cette instruction genere un nombre aleatoire entre 1 et 100,
    //via la fonction rand
    echo "$n /";
}
```

do ... while

```
do {
    //Mes instructions
} while ($condition);
```

Ici les instructions sont exécutées avant l'évaluation de $condition qui est toujours évaluée avec une valeur booléenne.

Les instructions sont donc exécutées au moins une fois.

Exemples

Voici un exemple d'utilisation de do...while

```php
do {
$n = rand(1,100);
//cette instruction genere un nombre aleatoire
//entre 1 et 100, via la fonction rand
echo "$n /";
} while ($n % 7 != 0);
//on obtient le même type de résultat que l’exemple précédent avec la boucle while.
//à noter que l'on a pas eu besoin d'initialiser la variable n
// dans cette exemple. En effet dans l'exemple précédent c’était nécessaire
//pour entrer dans la boucle, ce n'est pas nécessaire dans une boucle de type do...while
```

Boucle foreach

La boucle foreach permet de parcourir rapidement l'ensemble des éléments d'un tableau, ce que peut aussi faire la boucle for mais foreach se révèle plus efficace.

```php
$tableau = ["valeur1" => 10, "valeur2" => 20, "valeur3" => 30];

foreach($tableau as $cle=>$valeur) {
    echo "$cle:$valeur/";
}
```

Concatenation

Il est possible de concaténer plusieurs chaînes de caractères en les séparant par un point ou en écrivant directement dans des guillemets doubles :

```php
$salut = "bonjour";
$prenom = "Pierre";

echo $salut." ".$prenom;
echo "$salut $prenom";
```

Les fonctions

Une fonction est un bloc d'instructions réutilisable à volonté, une fois défini. Ce bloc d'instructions peut prendre des paramètres en entrée ou non et peut renvoyer une valeur en sortie ou non

```
function nomdelafonction($param1,$param2,...){
    //instruction a executer
return $unevaleur;
}


nomdelafonction($param1,$param2,...);
```