Sommaire

Généralités 1
Tests et mises au point localement 2
Pour tester le code PHP il est possible d'installer EasyPHP sur votre machine 2
Création de code PHP 2
Envoi au serveur 2
Lancement du code PHP 2
Syntaxe d'une page PHP 2
Exemple 2
Syntaxe des commandes 2
Variables 2
Exemple 2
Commandes de bases 3
Echo ou Print 3
Exemple 3
Instructions conditionnelles PHP 3
if 3
if … else 3
if … else if … else 3
Boucle 4
La boucle while 4
La boucle do ... while 4
La boucle for 5
Formulaire PHP 5
Exemple 5
Exemple d'application PHP 5
Afficher la table de multiplication (table de 8) 5
Variables utilisées 6
Algorithme 6
Le programme en PHP 6

Généralités

(Source Wikipedia)

PHP: Hypertext Preprocessor, plus connu sous son sigle PHP, est un langage de programmation compilé à la volée libre principalement utilisé pour produire des pages Web dynamiques via un serveur HTTP, mais pouvant également fonctionner comme n'importe quel langage interprété de façon locale. PHP est un disposant depuis la version 5 de fonctionnalités de POO (programmation orientée objet).

Le langage PHP est utilisé principalement en tant que langage de script côté serveur, ce qui veut dire que c'est le serveur (la machine qui héberge la page Web en question) qui va interpréter le code PHP et générer du code (constitué généralement d'XHTML ou d'HTML, de CSS, et parfois de JavaScript) qui pourra être interprété par un navigateur. PHP peut également générer d'autres formats en rapport avec le Web, comme le WML, le SVG, le format PDF, ou encore des images bitmap telles que JPEG, GIF ou PNG.

Il a été conçu pour permettre la création d'applications dynamiques, le plus souvent développées pour le Web. PHP est très majoritairement installé sur un serveur Apache, mais peut être installé sur les autres principaux serveurs HTTP du marché. Ce couplage permet de récupérer des informations issues d'une base de données, d'un système de fichiers (contenu de fichiers et de l'arborescence) ou plus simplement des données envoyées par le navigateur afin d'être interprétées ou stockées pour une utilisation ultérieure.

Tests et mises au point localement.

Pour tester le code PHP il est possible d'installer EasyPHP sur votre machine.

Ou tests en ligne

http://phptester.net/

Création de code PHP

Envoi au serveur

Le code php est stocké sur le serveur sous forme de pages " .php " créées avec un éditeur de texte.

Lancement du code PHP

Lorsque le navigateur demande le chargement d'une page avec l'extension PHP le serveur interprète le code et crée une page au format HTML qui sera lue par celui-ci.

Syntaxe d'une page PHP

Le serveur interprète le contenu de la page comme une page HTML et interprète le code PHP contenu entre la balise ouvrante " <?php " et la balise fermante "?> ". Le code PHP peut cohabiter avec du HTML en plusieurs parties. Le fichier de code PHP doit avoir une extension reconnue comme du code PHP (en général extension .php)

Exemple

```
<!DOCTYPE html>
<html>
<body>

<?php
echo "Petit script PHP";
?>

</body>
</html>
```

Syntaxe des commandes

Variables

Les variables sont des «conteneurs» pour stocker des informations. Le type de variable est implicite lors de leur déclaration.

- Une variable commence par le signe $
- Un nom de variable ne doit contenir que des caractères alphanumériques
- Un nom de variable ne doit pas commencer par un chiffre
- Un nom de variable ne doit pas contenir d'espace.
- Les noms de variable sont sensible à la casse (majuscule/minuscule) ($x \neq X$)

Exemple

```
<?php
$x=5;
$y=6;
$z=$x+$y;
echo $z;
$a= "Bonjour " ;
```

```
        echo $a ;
?>
```

Dans l'exemple ci-dessus les variables x, y et z sont du type entier et a est du type chaîne de caractères.

Commandes de bases

Echo ou Print

Les instructions echo ou print permettent d'afficher. Il semble qu'il est plus rapide d'utiliser echo.

Exemple

```
<?php
$a= "Bonjour" ;
$b= "à tous " ;
        echo $a . " " .$b ;
        print $a. " " .$b ;
?>
```

Instructions conditionnelles PHP

Très souvent, lorsque vous écrivez du code, vous souhaitez effectuer des actions différentes pour différentes décisions. Vous pouvez utiliser des instructions conditionnelles dans votre code pour ce faire.

En PHP, nous avons les instructions conditionnelles suivantes:

- **if** - exécute un code uniquement si une condition spécifiée est vraie
- **if ... else** - exécute un code si une condition est vraie et un autre code si la condition est fausse
- **if ... else if .... else** - sélectionne l'un des nombreux blocs de code à exécuter

if

```
<?php
$t=date("H");
if ($t<"20")
        {
        echo "Bonjour!";
        }
?>
```

if … else

```
<?php
$t=date("H");
if ($t<"20")
        {
         echo "Bonjour!";
        }
          else
                {
                echo "Bonsoir!";
                }
?>
```

if … else if … else

```
<?php
$t=date("H");
if ($t<"12")
        {
        echo "Bonjour!";
        }
else if ($t<"20")
```

```
        {
        echo "Bon après-midi!";
        }
        else
        {
        echo "Bonsoir!";
        }
?>
```

Boucle

Lorsque l'on veux répéter une partie de code plusieurs fois nous pouvons utiliser des boucles.

En PHP, nous avons les instructions de boucle suivantes:

- **tandis que** - une boucle à travers un bloc de code tant qu'une condition spécifiée est vraie
- **do ... while** - boucles à travers un bloc de code une fois, puis répète la boucle tant que la condition spécifiée est vraie
- **for -** boucles à travers un bloc de code un nombre de fois spécifié
- **foreach** - boucle à travers un bloc de code pour chaque élément dans un tableau

La boucle while

La boucle while exécute un bloc de code tant qu'une condition est vraie.

```
        <?php
$i=1;
while($i<=5)
        {
        echo "La valeur de i = " . $i . "<br>";
        $i++;
        }
?>
```

Donne :

```
La valeur de i = 1
La valeur de i = 2
La valeur de i = 3
La valeur de i = 4
La valeur de i = 5
```

La boucle do ... while

La boucle do ... while exécute toujours le bloc de code une fois, la condition sera alors vérifiée, et la boucle sera répétée tant que la condition est vraie.

```
        <?php
$i=1;
do
        {
        $i++;
        echo "La valeur de i= " . $i . "<br>";
        }
while ($i<=5);
?>
```

Donne :

```
La valeur de i = 2
La valeur de i = 3
```

La valeur de i = 4

La valeur de i = 5

La valeur de i = 6

Nota : dans les boucles " while " et " do....while " il est nécessaire que la condition de sortie évolue dans la boucle, sinon c'est une boucle sans fin (bug).

La boucle for

La boucle " for " est utilisée lorsque l'on sait à l'avance combien de fois le script doit être exécuté.

```
<?php
for ($i=1; $i<=5; $i++)
        {
        echo "La valeur de i = " . $i . "<br>";
        }
?>
```

Formulaire PHP

Tout élément de formulaire dans une page HTML sera **automatiquement** disponible dans les scripts PHP.

Exemple

L'exemple ci-dessous contient un formulaire HTML avec deux champs de saisie et un bouton submit (envoyer).

```
<html>
<body>

<form action="bienvenue.php" method="post">
Nom: <input type="text" name="nom">
Age: <input type="text" name="age">
<input type="submit">
</form>

</body>
</html>
```

Lorsqu'un utilisateur remplit le formulaire ci-dessus et clique sur le bouton submit, les données du formulaire sont envoyées à un fichier PHP, appelé "bienvenue.php":

"bienvenue.php" ressemble à ceci:

```
<html>
<body>

Bonjour <?php echo $_POST["nom"]; ?>!<br>
tu as <?php echo $_POST["age"]; ?> ans.

</body>
</html>
```

La sortie peux ressembler à:

Bonjour Toto!
Tu as 18 ans.

Exemple d'application PHP

Afficher la table de multiplication (table de 8)

Variables utilisées

$table (entier)
$resultat (entier)
$nombre (entier de 0 à 9)

Algorithme

Pour les nombres de 0 à 9 ($nombre) avec un incrément de 1
    calculer $resultat=$table*$nombre
Afficher ($table x $nombre = $resultat)

Le programme en PHP.

```php
<?php
/*
Variables

$table (entier)
$resultat (entier)
$nombre (entier de 0 à 9)

Programme
*/
$table=8;
for($nombre=0; $nombre <= 9; $nombre+=1){
    $resultat=$table*$nombre;
    print($table."x".$nombre."=".$resultat);
    print("<br>");
    }

?>
```