Après avoir vu un premier aperçu du langage PHP, nous allons maintenant étudier les différentes structures de contrôles du langage. Les structures de contrôles nous permettrons de faire des tests entre les variables et d'exécuter diverses boucles.

Voici un petit récapitulatif des principales structures de contrôles :

Instruction	Signification
if	Si
else	Sinon
elseif	Sinon si
switch	Selon
for	Pour chaque (boucle)
while	Tant que (boucle)
==	Strictement égal
!=	Différent
<	Strictement inférieur
>	Strictement supérieur
<=	Inférieur ou égal
>=	Supérieur ou égal
and ou &&	ET logique
or ou ||	OU logique

Nous allons maintenant illustrer toutes ces structures de contrôles.

- **if**, **else**, **elesif** :

Nous allons initialiser une variable numérique $nombre à la valeur 11 par exemple, et faire différent tests dessus.

exemple1

```php
<?php
$nombre = 11;
if ($nombre >= 0 && $nombre < 10) {
   // on teste si la valeur de notre variable est comprise entre 0 et 9
   echo $nombre.' est compris entre 0 et 9';
}
elseif ($nombre >= 10 && $nombre < 20) {
   // on teste si la valeur de notre variable est comprise entre 10 et 19
   echo $nombre.' est compris entre 10 et 19';
}
else {
   // si les deux tests précédents n'ont pas aboutis, alors on tombe dans ce cas
   echo $nombre.' est plus grand que 19';
}
?>
```

A l'affichage on aura :

```
11 est compris entre 10 et 19
```

(Remarquons déjà que les instructions qui doivent être exécutées lorsqu'un test est validé sont systématiquement comprises entre des crochets **{ }**).

En effet, résumons ce qui vient de se passer.

Dans un premier temps, on teste si $nombre est supérieur ou égal à 0 et strictement inférieur à 10, et dans ce cas, et seulement dans ce cas, on écrira alors sur l'écran, $nombre est compris entre 0 et 9.
Or vu que $nombre est égal à 11, on se rend compte que ce test ne sera pas satisfait.
On a alors deux solutions. Soit on écrit directement le cas Sinon (**else**), soit on peut faire un autres test, ce qui correspond à un SinonSi (**elseif**).
Nous avons choisis de faire un second test (**elseif**).
La, on teste si $nombre est supérieur ou égal à 10 et strictement inférieur à 20 (ce qui est notre cas car $nombre est égal à 11).
Le test est donc validé, et l'on exécute alors les instructions présentes entre les crochets **{ }** du **elseif**.
On affichera donc à l'écran : 11 est compris entre 10 et 19
Viens ensuite le cas **else** qui est exécuté seulement si aucunes des conditions définies par les **if** et les **elseif** n'est vérifiées.

- **switch** :

Le **switch** représente exactement la même chose qu'une succession d'un **if** et de plusieurs **elseif**. En revanche, utiliser un **switch** à un certain avantage comparé à un **if** et à plusieurs **elesif**, c'est que sa structure est beaucoup moins lourde et nettement plus agréable à lire.

Prenons un exemple simple. Nous allons déclarer une variable contenant une chaîne de caractères, puis nous allons tester cette chaîne grâce au **switch**.
On aura alors le code suivant :

exemple2

```php
<?php
$nom = "LA GLOBULE";

switch ($nom) {
   case 'Jean' :
      echo 'Votre nom est Jean.';
      break;
   case 'Benoît' :
      echo 'Votre nom est Benoît.';
      break;
   case 'LA GLOBULE' :
      echo 'Votre nom est LA GLOBULE.';
      break;
   default :
      echo 'Je ne sais pas qui vous êtes !!!';
}
?>
```

Dans notre cas, vu que $nom contient la chaîne de caractère LA GLOBULE, on verra alors s'afficher à l'écran la phrase suivante :

```
Votre nom est LA GLOBULE.
```

En revanche, si la variable $nom avait contenu la chaîne de caractère "toto", ce même code aurait affiché à l'écran :

```
Je ne sais pas qui vous êtes !!!
```

En utilisant un **if** puis une succession de **elseif**, le code suivant aurait exactement eu le même affichage sur l'écran :

exemple3

```php
<?php
$nom = "LA GLOBULE";

if ($nom == "Jean") {
```

```
   echo 'Votre nom est Jean.';
}
elseif ($nom == "Benoît") {
   echo 'Votre nom est Benoît.';
}
elseif ($nom == "LA GLOBULE") {
   echo 'Votre nom est LA GLOBULE.';
}
else {
   echo 'Je ne sais pas qui vous êtes !!!';
}
?>
```

Attention !

Notez bien l'utilisation de **break** dans chaque cas de votre switch. Si celui-ci est omis, tous les messages s'afficheront.

- **for** (pour chaque) :

La structure de contrôle **for** nous permet d'écrire des boucles. En clair, cela veut dire que nous allons exécuter une série d'instructions un nombre de fois bien déterminé.

Prenons l'exemple suivant :

exemple4

```
<?php
$chiffre = 5;

// Début de la boucle
for ($i=0; $i < $chiffre; $i++) {
   echo 'Notre chiffre est différent de '.$i.'<br />';
}
// Fin de la boucle

echo 'Notre chiffre est égal à '.$i;
?>
```

Ce qui affichera à l'écran :

```
Notre chiffre est différent de 0
Notre chiffre est différent de 1
Notre chiffre est différent de 2
Notre chiffre est différent de 3
Notre chiffre est différent de 4
Notre chiffre est égal à 5
```

En effet, on initialise notre variable $chiffre à 5. On démarre la boucle **for** qui dit que l'on va exécuter les instructions situées entre les crochets de la boucle (**{ }**) pour i variant de 0 à $chiffre-1 (donc jusqu'à 4), i étant incrémenter à chaque passage de boucle ($i++).
($i varie de 0 à $hiffre-1 car on impose que $i soit strictement inférieur à $chiffre).
On exécute alors 4 fois les instructions présentes dans la boucle, et à chaque passage, $i verra sa valeur augmentée de 1.

Aparté :

L'utilisation des boucles est extrêmement importante (et indispensable) en programmation. La compréhension de ce passage est capital.

- **while** (tant que)

Voyons maintenant l'autre boucle, la boucle **while** (dite boucle tant que). Il faut déjà savoir que la boucle **while** n'est pas

vraiment d'une nécessité absolue (elle est absente dans certains langages de programmation) vu qu'elle est toujours remplaçable par une boucle **for**.

Reprenons l'exemple précédent, et écrivons le à l'aide de la boucle **while**, on a :

exemple5

```php
<?php
$chiffre = 5;
$i = 0;

// Début de la boucle
while ($i < $chiffre) {
   echo 'Notre chiffre est différent de '.$i.'<br />';
   $i = $i + 1;
}
// Fin de la boucle

echo 'Notre chiffre est égal à '.$i;
?>
```

Ce qui affichera à l'écran exactement la même chose que ce qu'affiche le code que l'on a utilise pour la boucle **for**.

Ici, on initialise notre variable $chiffre à 5, puis la variable $i à 0.
Ensuite, nous faisons le test suivant : "tant que $i et augmenter la valeur de $i de 1"
Puis dès que la condition $i < $chiffre n'est plus vérifiée, nous sortons de la boucle pour finir l'exécution des instructions qui suivent.

Auteur : LA GLOBULE
Dernière révision du cours : le 29/05/2007 à 21:37