Les structures de contrôle

Pour commencer voici un petit tableau bien utile sur les instructions les plus utilisées

if	Si
else	Autrement
elseif	Autrement Si
switch	selon
while	Chaque fois que *(boucle)*
for	Tant que *(boucle)*
==	Strictement égal à
!=	Différent de
<	Plus petit que
>	Plus grand que
<=	Plus petit ou égal
>=	Plus grand ou égal
and ou >	Et
or ou ||	Ou

Pour illustrer les if, else et elseif, voici un exemple très simple à lire, nous définissons une variable à la valeur ' 512 ' puis nous allons tester si celle−ci est comprise entre 0 et 499 puis entre 500 et 999 et enfin supérieure à 999, ce qui nous donne :

Le code PHP	Ce qui donne à l'écran

```php
<?
$toto = 512;

// on enchaîne les contrôles ci−dessous ----
if($toto>=0 && $toto<500)         //1er
    {
    echo $toto.' est compris entre 0 et 499';
    }
elseif($toto>=500 && $toto<1000) //2eme
    {
    echo $toto.' est compris entre 500 et 999';
    }
else                                  //3eme
    {
    echo $toto.' est plus grand que 999';
    }
?>
```

512 est compris entre 500 est 999

1er contrôle : Si (512 est plus grand ou égal à 0 et que 512 est plus petit que 500**) {** on affiche le 1er message **; }**
2e contrôle : Si (512 est plus grand ou égal à 500 et que 512 est plus petit que 1000**) {** on affiche le 2e message **; }**
3e contrôle : Dans tous les autres cas { on affiche le 3e message **; }**

Je vous invite à faire plusieurs tests en changeant à chaque fois la valeur de **$toto** pour vérifier que les messages respectifs s'affichent bien quand les conditions sont remplies.

Voici un autre exemple pour mieux comprendre, et qui nous permettra de voir la structure switch par la suite.

Le code PHP	Ce qui donne à l'écran

```php
<?
$medor = 'chien';

// on enchaîne les contrôles ci-dessous ----
if($medor == 'girafe')
    {
    echo 'Medor est une girafe !';
    }
elseif($medor == 'elephant')
    {
    echo 'Medor est un éléphant';
    }
elseif($medor == 'souris')
    {
    echo 'Medor est une souris';
    }
elseif($medor == 'chien')
    {
    echo 'Medor est un chien';
    }
elseif($medor == 'chat')
    {
    echo 'Medor est un chat';
    }
else
    {
    echo 'Peut être un hippopotame ? Qui sait ...';
    }
?>
```

Medor est un chien

Cependant vous vous apercevez que cette structure est un peu lourde, et pas forcement facile à lire. Utiliser un switch permet de résoudre ce probleme. Le code suivant est exactement le même que le précédent, mais ecrit avec un switch

Le code PHP	Ce qui donne à l'écran

```php
<?
$medor = 'chien';

switch($medor)
    {
    case 'girafe':
        echo 'Medor est un girafe !';
        break;
    case 'elephant':
        echo 'Medor est un éléphant';
        break;
    case 'souris':
        echo 'Medor est une souris';
        break;
    case 'chien':
        echo 'Medor est un chien';
        break;
    case 'chat':
        echo 'Medor est un chat';
        break;
    default:
        echo 'Peut être un hippopotame ? Qui sait ...';
```

Medor est un chien

```
    }
?>
```

Notez bien l'utilisation de break; . Cela permet de sortir de la boucle et donc de gagner en efficacité.

Passons maintenant à la fameuse boucle while, je dis "fameuse" car elle est sujette à de petites galères quand on l'utilise pour la première fois. Nous allons reprendre notre variable $toto à laquelle nous allons donner la valeur 6, puis à l'aide de la boucle nous allons nous mettre dans le cas où nous ne connaissons pas la valeur de $toto et allons la rechercher. Ce qui donne :

Par convention la variable $i fait toujours office de compteur dans une boucle elle a toujours la valeur "0" au début, vous notez que cette valeur prend +1 à la fin de la boucle ($i++) et chaque fois que la condition a été respectée *(en l'occurrence que $i est différent de $toto)* on retourne à WHILE et l'on fait un nouveau test, etc., ce qui donne en français :

1. $i = 0 , on teste si 0 est différent de toto = Oui, on affiche le message (echo) puis on ajoute 1 à $i ($i++) et on retourne à while.
2. $i = 1 , on teste si 1 est différent de toto = Oui, on affiche le message (echo) puis on ajoute 1 à $i ($i++) et on retourne à while.
3. $i = 2 , on teste si 2 est différent de toto = Oui, on affiche le message (echo) puis on ajoute 1 à $i ($i++) et on retourne à while.
4. etc...
5. $i = 6 , on teste si 6 est différent de toto = Non, on sort de la boucle (accolades), et on poursuit le code. En l'occurrence on affiche le message (Toto est égal à 6).

Nous allons maintenant voir les boucle for. Elles permettent de realiser la même chose que les boucles while, encore une fois c'est la syntaxe qui change. Au lieu de déclarer le compteur avant le debut de la boucle ($i=0;) et a chaque fin de tour d'incrementer le compteur ($i++), on le fait directement dans la déclaration de la boucle. Voici le meme code que précédemment avec une boucle for :

Le code PHP	Ce qui donne à l'écran
`<?` `$toto = 6;` `//----------[DEBUT BOUCLE]------------` `for($i=0; $i != $toto ; $i++)` `    {` `    echo 'Toto est different de '.$i.' ';` `    }` `//------------[FIN BOUCLE]------------` `echo 'Toto est égal à '.$i;` `?>`	Toto est different de 0 Toto est different de 1 Toto est different de 2 Toto est different de 3 Toto est different de 4 Toto est different de 5 **Toto est égal à 6**

Notez : Il est vraiment très important de maîtriser les boucles car c'est là un élément systématiquement utilisé pour afficher des résultats venant d'une base de données et nous l'utiliserons beaucoup.