PHP - Les boucles

Date de dernière mise à jour : 27/06/2007 à 19:36

Définition et présentation

Les boucles vont vous permettre d'effectuer un certain nombre de fois les mêmes opérations, par exemple afficher une suite de nombres que l'on incrémentera à chaque tour de boucle. Vous allez pouvoir répéter autant d'instructions que vous souhaitez et vous arrêter lorsque vous l'aurez décidé, selon une condition bien précise par exemple (si vous ne vous rappelez plus de ce qu'est une condition, je vous engage à relire le chapitre les concernant). Voyons maintenant différents types de boucles que nous expliquerons au fur et à mesure. Commençons par la boucle la plus simple, la boucle while.

La boucle while

Cette boucle qui est une des plus utilisées en PHP va vous permettre d'effectuer une ou plusieurs actions **tant que** la condition que vous placerez à l'intérieur des parenthèses sera vérifiée. Voyons de plus près sa syntaxe :

```
<?php
while(
condition
)    {
instruction 1
;
instruction 2
;          ...     }
?>
```

Voyons maintenant un exemple. Imaginiez que vous souhaitez afficher la valeur d'un nombre tant que celui-ci est inférieur à 7. Dès que le nombre sera supérieur à 7, vous quitterez la boucle. Le nombre sera incrémenté d'une unité à chaque tour de boucle. Voilà ce que ça donne avec une boucle while en pratique :

```
<?php     $i
=

;
//on définit la variable $i qui sera notre nombre que l'on incrémentera. Ici $i va commencer à 0
while(
$i
7
)    {          echo
$i
.
"
;
$i
++;     }
?>
```

On traduit donc la boucle while **tant que**. Tant que la condition est vérifiée (ici $i <?php $i

```
=

;
//on définit la variable $i qui sera notre nombre que l'on incrémentera. Ici $i va commencer à 0
```

```
$j
=
4
;    while(
$i
7
AND
$j
5
)    {          echo
$i
.
"
;
$i
++;
$j
++;    }
?>
```

Vous allez constater que la variable $i ne s'affiche qu'une seule fois. Pourquoi ? car au prochain tour de boucle, $j vaudra 5, or on n'effectue la boucle que si $i La boucle for
Cette boucle est très utile lorsque, comme pour notre premier exemple, vous souhaitez incrémenter une variable sur une certaine plage de valeurs connue à l'avance.

Etudions sa syntaxe :

```
<?php
for(
initialisation
;
condition de continuité
;
expression 3
)    {
instruction 1
;
instruction 2
;          ...    }
?>
```

La première expression (**initialisation**) est interprétée au premier démarrage de la boucle. Elle ne sera exécutée qu'**une seule fois** (et non à chaque tour de boucle). On y place généralement l'initialisation des variables. La deuxième condition est appelée **condition de continuité**, on la place pour dire **"si cette condition est respectée, alors la boucle peut continuer"**. La troisième expression sera elle interprétée à la fin de chaque tour de boucle. On y place généralement l'incrémentation d'une variable, mais vous pouvez y mettre n'importe quoi, faites attention cependant aux boucles infinies. On entend par boucle infinie une boucle qui ne s'arrête jamais. Outre le fait de monopoliser le processeur, votre script ne s'arrêtera pas en fonction de la configuration de PHP.

Voici notre premier exemple qui utilise cette fois-ci une boucle for :

```
<?php
for(
$i
```

=

;
$i
7
;
$i
++) { echo
$i
.
"
; }
?>

Vous pouvez constater que pour cet exemple précis, le code a été légèrement réduit. Ces deux codes sont pourtant équivalents. Reprenons notre deuxième exemple, cette fois-ci à l'aide d'une boucle for. Comme vous allez pouvoir le constater, on peut également placer plusieurs conditions :

<?php $i
=

;
//on définit la variable $i qui sera notre nombre que l'on incrémentera. Ici $i va commencer à 0
for(
$j
=
4
;
$i
7
AND
$j
5
;
$j
++) { echo
$i
.
"
;
$i
++; }
?>

Voyons maintenant une dernière boucle, qui n'est pas plus compliquée qu'une boucle while. Il s'agit de la boucle do-while.

La boucle do-while

Rien qu'au nom de la boucle, vous pouvez vous douter qu'elle a des similitudes avec la boucle while. Effectivement, c'est le cas ! Néanmoins, une différence de taille les séparent. Reprenons notre premier exemple avec la boucle while :

<?php $i
=

;

```
//on définit la variable $i qui sera notre nombre que l'on incrémentera. Ici $i va commencer à 0
while(
$i
7
)    {          echo
$i
.
"
;
$i
++;    }
?>
```

Vous constatez que si on initialise $i à 8, la boucle ne sera jamais exécutée, car dès l'analyse de la condition, 8 Vous aimeriez peut-être effectuer au moins une fois une opération, et si une condition est vraie, reboucler autant de fois que vous le souhaitez ? la boucle do-while permet de le faire en PHP. Voici sa syntaxe :

```
<?php
do    {
instruction 1
;
instruction 2
;          ...    }    while(
condition
)
?>
```

Dans ce cas, les instructions seront exécutées au moins une fois, même si la condition de continuité est fausse. Par contre, si la condition de continuité est vraie, PHP rebouclera. Voyons notre exemple du 8 <?php $i
=
8
;
$j
=
7
;   do   {   echo
'la boucle a bouclé'
;   }   while(
$i
$j
)
?>
```

Chapitre suivant

| ID | Donnée |
|---|---|
| 0 | Blabla |
| 1 | Xydsqk |
| 2 | makexs |
| 3 | Antho |

Les tableaux
Définition et présentation
Les tableaux à indices numériques (indexés)
```

Les tableaux à indices chaînés (associatifs)

Parcourir un tableau