PHP - Les conditions

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

Définition et présentation

Les conditions sont les éléments les plus utilisés dans n'importe quel langage. Il est impossible de faire l'impasse dessus. Grâce aux conditions, vous allez pouvoir effectuer des actions telles que "Si le visiteur est un homme, afficher 'Bonjour Monsieur', sinon afficher 'Bonjour Madame'". Vous allez pouvoir effectuer plein de traitements en fonction de données diverses. Une condition n'est ni plus ni moins qu'un bloc de code PHP contenant des instructions à exécuter en fonction de la condition qui elle sera placée entre parenthèses (bien que ce ne soit pas toujours le cas, on verra tout en bas l'opérateur ternaire, en attendant ne soyez pas affolés par le terme, tout deviendra compréhensif plus bas). Voyons déjà quelques opérateurs nécessaires pour appliquer tout ça aux conditions :

Les opérateurs de base conditionnels

On utilise quelques opérateurs très simples en PHP pour comparer deux variables. Nous ne verrons pas certains opérateurs particuliers qui dépassent le cadre de ce chapitre, mais vous les reverrez dans la partie qui concernera l'optimisation et la rédaction de vos scripts. Voici les opérateurs de base et leur signification :

Symbole
Signification

$a == $b
 $a est égal à $b

$a $a est inférieur à $b

$a > $b
 $a est supérieur à $b

$a $a est inférieur ou égal à $b

$a >= $b
 $a est supérieur ou égal à $b

$a != $b
 $a est différent de $b

$a $b
 $a est différent de $b

La structure de base if/else

La structure if / else est la base en PHP. Grâce à elle, vous allez pouvoir effectuer deux choses : une action si la condition est vraie, et une autre action si elle est fausse. On traduit le if par "si" et le else par "sinon". Voyons un exemple :

<?php $heure
=
12
;
//On définit une heure
if(
$heure

```
12
)
//Si l'heure contenue dans la variable est inférieure à 12
{         echo
'Il n\'est pas encore midi !'
;    }
?>
```

Vous constaterez en exécutant ce code qu'en fonction de ce que vous allez attribuer à la variable $heure, le message "Il n'est pas encore midi" s'affichera ou non. Vous pouvez tout à fait effectuer plusieurs actions dans une condition. Vous pouvez également imbriquer des conditions :

```
<?php     $heure
=
12
;
//On définit une heure
if(
$heure
12
)
//Si l'heure contenue dans la variable est inférieure à 12
{         echo
'Il n\'est pas encore midi !'
;         if(
$heure
==
11
)         {               echo
'Il est bientôt midi !'
;         }     }
?>
```

Notez l'utilisation du double égal, si vous mettez un simple signe égal, ça voudra dire "si j'ai attribué 11 à la variable, alors faire l'action", le problème est qu'attribuer une valeur à une variable renvoie toujours (dans notre cas) la valeur VRAI, donc la condition sera toujours exécutée. Pensez-y lorsque vous coderez vos scripts PHP car sinon vous risquez de vous retrouver avec des erreurs insolubles !

Bon tout ça c'est bien beau, mais vous aimeriez peut-être passer à plus évolué ! Nous allons donc voir maintenant le if / else en PHP. Else signifie "sinon", nous allons prendre un exemple simple qui consiste à comparer les valeurs de deux variables. Si les deux variables sont égales, nous afficherons "Les valeurs sont égales" dans le cas contraire nous afficherons "Les valeurs ne sont pas égales".

```
<?php     $nombre1
=
12
;
$nombre2
=
13
;     if(
$nombre1
==
$nombre2
)
```

```
//Si les deux nombres sont égaux
{	echo
'Les valeurs sont égales'
;	}	else
//Sinon
{	echo
'Les valeurs ne sont pas égales'
;	}
?>
```

Nous allons maintenant voir une autre structure de base un peu plus évoluée que le if / else, il s'agit du if / elseif / else.

La structure conditionnelle if / elseif / else

En français, on appelle cette structure le "si, sinon si, sinon". Grâce à ça, vous allez pouvoir exécuter des instructions PHP en fonction de conditions plus poussées. Prenons un exemple simple consistant à dire "Si les deux nombres sont égaux, on affiche 'Les valeurs sont égales' sinon si le nombre 1 est supérieur au nombre 2 on affiche 'Le nombre 1 est supérieur' et si aucune de ces conditions n'est validée, on affiche 'Le nombre 2 est supérieur'".

Voyons ce que ça donne niveau code :

```
<?php	$nombre1
=
12
;
$nombre2
=
13
;	if(
$nombre1
==
$nombre2
)
//Si les deux nombres sont égaux
{	echo
'Les valeurs sont égales'
;	}	elseif(
$nombre1
>
$nombre2
)
//Les deux nombres ne sont pas égaux mais $nombre1 est supérieur à $nombre2
{	echo
'Le nombre 1 est supérieur'
;	}	else
//Les deux nombres ne sont pas égaux, et $nombre1 n'est pas supérieur à $nombre2
{	echo
'Le nombre 2 est supérieur'
;	}
?>
```

Au niveau purement syntaxique, le code que je viens de vous présenter est équivalent à celui-ci :

```
<?php	$nombre1
```

```
=
12
;
$nombre2
=
13
;    if(
$nombre1
==
$nombre2
)
//Si les deux nombres sont égaux
{          echo
'Les valeurs sont égales'
;    }    else
//Les deux nombres ne sont pas égaux
{         if(
$nombre1
>
$nombre2
)
//$nombre1 est supérieur à $nombre2
{              echo
'Le nombre 1 est supérieur'
;         }         else
//$nombre1 n'est pas supérieur à $nombre2
{              echo
'Le nombre 2 est supérieur'
;         }    }
?>
```

Les conditions multiples

Les conditions multiples vont vous permettre de donner plusieurs conditions pour effectuer une ou plusieurs actions. Par exemple, si vous souhaitez que les visiteurs aient plus de 13 ans pour visiter le site et qu'ils soient des hommes, vous pourrez faire une condition de ce style :

```
<?php    $homme
=
FALSE
;
//booléen ayant la valeur FALSE (faux) ici il s'agit donc d'une femme
$age
=
17
;    if(
$homme
===
TRUE
AND
$age
>
13
)
//Le visiteur est un homme et agé de plus de 13 ans
```

```
{         echo
'Vous pouvez visiter le site'
;    }    else
//Le visiteur est une femme ou alors il a moins de 13 ans
{         echo
'Vous ne pouvez pas visiter le site'
;    }
?>
```

Notez la présence ici du triple égal. Ce n'est pas une erreur. Je ne vous ai pas présenté cet opérateur auparavant pour ne pas compliquer la tâche. En fait cet opérateur, en plus de comparer les valeurs des variables, compare également leurs types (bien que la notion de type soit très faible en PHP). Cet opérateur permet de valider une condition si les variables ont même valeur et même type. En fait, un booléen peut aussi être représenté par un nombre (0 pour FALSE et 1 pour TRUE). Le problème est que lorsque vous utiliserez des fonctions qui renvoient des booléens ou des nombres, comment distinguer 0 et 1 de FALSE et TRUE ?. C'est là qu'intervient le signe ===, qui vous permettra de savoir si la fonction a renvoyé TRUE ou 1, ce que ne permet pas de faire l'opérateur ==Il existe de la même manière l'opérateur !== qui regarde si deux variables sont de valeurs ou de types différents.

Voyons maintenant la liste des opérateurs utilisables :

Opérateur
Signification

AND
"et" logique

OR
"ou" logique

XOR
"ou exclusif" logique

&&
"et" logique, avec priorité supérieure

||
"ou" logique, avec priorité supérieure

Voyons un peu les opérateurs && et || ainsi que leurs différences avec les opérateurs AND et OR. Prenons le cas d'une condition qui doit vérifier "Si il s'agit d'un homme ou d'une femme qui a dans ce cas plus de 13 ans, on autorise l'accès au site". On constate que tous les hommes peuvent accéder au site, mais les femmes doivent avoir au moins 13 ans. On peut faire ça de plusieurs façons. La première, purement logique, utilise des parenthèses (en effet, ce qui se trouve entre parenthèses sera traité en premier par PHP, comme en mathématiques) :

```
<?php
if(
$homme
===
TRUE
OR (
$hommme
===
FALSE
```

```
AND
$age
>
13
))
//On veut soit tous les hommes, soit les filles de plus de 13 ans
{	echo
'Vous pouvez visiter le site'
;	}	else	{	echo
'Vous ne pouvez pas visiter le site'
;	}
?>
```

Vous constatez qu'au bout d'un moment, si les conditions sont multiples, le nombre de parenthèses peut être important, nous allons donc utiliser les opérateurs prioritaires && et || pour supprimer les parenthèses. Voici le code obtenu :

```
<?php
if(
$homme
===
TRUE
OR
$hommme
===
FALSE
&&
$age
>
13
)
//On veut soit tous les hommes, soit les filles de plus de 13 ans
{	echo
'Vous pouvez visiter le site'
;	}	else	{	echo
'Vous ne pouvez pas visiter le site'
;	}
?>
```

Comme le && est prioritaire, PHP effectue d'abord le traitement pour savoir si il s'agit d'une fille ayant plus de treize ans. On pourrait simuler ça par ce code :

```
<?php
if(
$homme
===
TRUE
OR
$fille_de_plus_de_treize_ans
===
TRUE
)
//On veut soit tous les hommes, soit les filles de plus de 13 ans
{	echo
```

'Vous pouvez visiter le site'
; } else { echo
'Vous ne pouvez pas visiter le site'
; }
?>

Ensuite PHP utilise le OR classique pour faire une condition entre les deux variables.

Le switch
Le switch est une instruction particulière. En effet, vous allez pouvoir remplacer des quantités de if/elseif/else par un seul switch. Voyons comment cela fonctionne avec un exemple simple : Imaginions que nous souhaitons afficher un message différent en fonction d'un nombre. Si ce nombre vaut 1, on affichera "le nombre vaut 1", si il vaut 2 "le nombre vaut 2", et si il vaut 7, on affichera "c'est un très bon chiffre". Ceci est possible via le code suivant :

<?php $nombre
=
7
; if(
$nombre
==
1
) { echo
'Le nombre vaut 1'
; } elseif(
$nombre
==
2
) { echo
'Le nombre vaut 2'
; } elseif(
$nombre
==
7
) { echo
'C\'est un très bon chiffre !'
; } else { echo
'Le nombre ne vaut ni 1, ni 2, ni 7'
; }
?>

Vous constaterez que si vous voulez faire 10 cas possibles, ça va devenir très lourd au niveau de la syntaxe.

<?php $nombre
=
7
; switch(
$nombre
) { case
1
: echo
'Le nombre vaut 1'
; break; case
2

```
:  echo
'Le nombre vaut 2'
;  break;  case
7
:  echo
'C\'est un très bon chiffre !'
;  break;  default:  echo
'Le nombre ne vaut ni 1, ni 2, ni 7'
;  break;  }
?>
```

La syntaxe est quand même nettement plus réduite. Maintenant, nous allons voir à quoi correspondent ces "case", "break" et encore "default". Regardez la structure du switch(). A l'intérieur des parenthèses, on met une seule variable pour laquelle on souhaitera comparer sa valeur (ou effectuer des conditions dessus). Chaque instruction case permet ensuite de **délimiter une condition**, on peut donc voir ça comme ça : si $nombre correspond à **1**, alors on exécute le **echo 'Le nombre vaut 1';**

Le break veut dire qu'il faut que PHP s'arrête, c'est l'équivalent de la fin de condition. Imaginions que vous ayez le code suivant :

```
<?php  $nombre
=
7
;  switch(
$nombre
)  {  case
1
:case
2
:  echo
'Le nombre vaut 1 ou 2'
;  break;  case
7
:  echo
'C\'est un très bon chiffre !'
;  break;  default:  echo
'Le nombre ne vaut ni 1, ni 2, ni 7'
;  break;  }
?>
```

Je pense que vous comprenez le fonctionnement du break à présent, ainsi que de l'utilisation de conditions OU (ici, on affiche un message différent si le nombre vaut soit 1, soit 2). Voyons maintenant comment insérer des conditions plus évoluées dans le switch :

```
<?php  $nombre
=
8
;  switch(
$nombre
)  {  case
$nombre
7
:  echo
'Le nombre est inférieur à 7'
;  break;  case
7
```

```
:                echo
'C\'est un très bon chiffre !'
;            break;           case
$nombre
>
7
:               echo
'le nombre est supérieur à sept'
;           break;           default:                echo
'Le nombre ne vaut ni 1, ni 2, ni 7'
;           break;      }
?>
```

A noter que vous pouvez tout à fait utiliser des AND, OR, etc ... dans les instructions case du switch !

Chapitre suivant

Les boucles
Définition et présentation
La boucle While (tant que)
La boucle for (pour)
La boucle do while (faire tant que)