PHP – Les bases – Partie 2

Les variables de type tableau

PHP propose une autre façon de grouper et de manipuler les données: les tableaux. Il y a deux types de tableaux: les tableaux à index numériques et les tableaux associatifs. Ces deux types de tableau ont chacun leur syntaxe de déclaration.

Un tableau peut être créé en utilisant la structure de langage array(). Il prend un nombre illimité de paramètres, chacun séparé par une virgule, sous la forme d'une paire *key => value*.

```
array(
    key  => value,
    key2 => value2,
    key3 => value3,
    ...
)
```

La virgule après le dernier élément d'un tableau est optionnelle et peut ne pas être ajoutée. C'est généralement ce qui est fait pour les tableaux sur une seule ligne, i.e. *array(1, 2)* est préféré à *array(1, 2,)*.

Pour les tableaux sur plusieurs lignes, la virgule finale est généralement utilisée, car elle permet d'ajouter plus facilement de nouveaux éléments à la fin.

Depuis PHP 5.4, vous pouvez également utiliser la syntaxe courte, qui remplace *array()* par *[]*.

```
<?php
$array = array(
    "foo" => "bar",
    "bar" => "foo",
);

// depuis PHP 5.4
$array = [
    "foo" => "bar",
    "bar" => "foo",
];
?>
```

La clé key peut être soit un integer, soit une chaîne de caractères. La valeur value peut être de n'importe quel type.

On peut créer un tableau à index automatiques (clef entière numérique) de cette manière:

```
<?php
$tableau = array("un","deux","trois");
echo $tableau[0]; // affiche un
echo $tableau[1]; // deux
?>
```

La clef est générée en commançant par 0 et incrémentée de 1 pour chaque nouvelle entrée sans clef.
Il est cependant possible de mélanger les clefs automatiques et les clef associatives:

```
<?php
$tableau = array("un","deux","a"=>"trois",5=>"quatre","cinq");
var_dump($tableau);
// affichera
array (size=5)
  0 => string 'un' (length=2)
  1 => string 'deux' (length=4)
  'a' => string 'trois' (length=5)
  5 => string 'quatre' (length=6)
  6 => string 'cinq' (length=4)
?>
```

Vous remarquerez que lorsque l'on rentre une clef numérique manuellement, l'auto-incrémentation continuera l'entier supérieure à celui-ci.

Un tableau existant peut être modifié en y assignant explicitement des valeurs.

L'assignation d'une valeur dans un tableau est effectué en spécifiant la clé, entre crochets. La clé peut également ne pas être renseignée, sous la forme : *[]*.

```
<?php
$arr['clé'] = valeur;
$arr[] = valeur;
?>
```

Un tableau peut être **multidimensionnel**, c'est à dire que des tableaux peuvent contenir d'autres tableaux, qui a leurs tours peuvent contenir d'autres tableaux etc...

```
<?php
$tableau = array("niveau1",
                 "niveau1",
                 array("niveau2",
                       "niveau2",
                       array("niveau3",
                             "niveau3")
                       ),
                 "niveau1"
           );
```

```
echo $tableau[0]; // niveau1
echo $tableau[2][0]; // niveau2
echo $tableau[2][2][0]; //niveau3

// autre manière de générer un tableau multidimensionnel

$a = array();
$a[0][0] = "a";
$a[0][1] = "b";
$a[1][0] = "y";
$a[1][1] = "z";

?>
```

Traitement des tableaux : la boucle foreach

La structure de langage *foreach* fournit une façon simple de parcourir des tableaux.

foreach ne fonctionne que pour les tableaux et les objets, et émettra une erreur si vous tentez de l'utiliser sur une variable de type différent ou une variable non initialisée. Il existe deux syntaxes :

```
<?php

foreach ($tab as $value){
    //commandes
}
foreach ($tab as $key => $value){
    //commandes
}
?>
```

La première forme passe en revue le tableau *$tab*. À chaque itération, la valeur de l'élément courant est assignée à *$value* et le pointeur interne de tableau est avancé d'un élément (ce qui fait qu'à la prochaine itération, on accédera à l'élément suivant).

La seconde forme assignera en plus la clé de l'élément courant à la variable *$key* à chaque itération.

Le foreach remplace avantageusement depuis php 4 la méthode while (list each) utilisée auparavant :

```
<?php
$arr = array("un", "deux", "trois");
while (list(, $value) = each($arr)) {
    echo "Valeur : $value<br />";
}
```

```
foreach ($arr as $value) {
    echo "Valeur : $value<br />";
}
?>
```

! Les fonctions list() ou each() gardent une utilité mais dans des cas différents que nous verrons plus tard.

On peut imaginer utiliser une boucle for() ou while() pour afficher les éléments d'un tableau à index automatique, si et seulement si les valeurs n'ont pas été altérées et qu'il n'y a pas de clef manquantes risquant de générer des erreurs.

On utilisera count() pour connaître le nombre d'éléments dans le tableau.

```
<?php
$a = array("un","deux","trois","quatre","cinq","six","sept");

$nb = count($a); // nombre d'éléments du tableau

// les {} ne sont pas obligatoires sur une ligne
for($i=0;$i<$nb;$i++) echo $a[$i]." ";

// idem avec la boucle while
$i=0;
while($i<$nb){
    echo $a[$i]." ";
    $i++;
}
?>
```