# Les tableaux en PHP

Attention, les affichages sont prévus pour les navigateurs web (HTML).

Exemple, remplacer <br> par \n pour un saut de ligne, pour votre utilisation actuelle sous Netbeans.

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.

# Les tableaux à index numériques array(…) ;

Il s'agit d'une simple liste d'éléments. Pour les créer, on emploi le mot clé **array()** et on sépare les **éléments par des virgules**. Dans la liste, chaque élément est repéré par un **numéro unique**.

Ce numéro est appelé **index (ou indice)** et est attribué automatiquement suivant les ordres.

**Pour avoir accès à la valeur de chaque élément, on indique la variable tableaux suivi de l'indice de l'élément mis entre crochet**.

Lors de la déclaration d'un tableau, il n'est pas nécessaire d'indiquer sa taille. Ceci est géré par PHP. Dans un tableau indexé numériquement, **le premier indice est par défaut 0**.

*< ?php   
$etudiant = array("Sarah", "Sebastien", "Kevin", "Odile", "Rodrigue");  
echo $etudiant[0].'<br>';  
echo $etudiant[1].'<br>';  
echo $etudiant[2].'<br>';  
echo $etudiant[3].'<br>';  
echo $etudiant[4].'<br>';  
?>*

# Boucle, Tableau et fonction count

**La fonction count permet de compter le nombre d'éléments d’un tableau.**

*<?php   
$etudiant = array("Sarah", "Sebastien", "Kevin", "Odile", "Rodrigue");  
for($i = 0;$i < count($etudiant); $i++)  
{  
echo $etudiant[$i] . "<br>";  
}  
?>*

# Les tableaux associatifs en PHP

**Lors de la définition d'un tableau associatif, on indique nous même les indices du tableau.**

Dans la mesure où l'on est libre de les spécifier, **les indices** peuvent être non seulement des nombres, mais aussi des **chaînes de caractères**. Dans ce cas on parle de '**clé**'. Ce type de tableau est très pratique dans le cas où l'on s'intéresse à la signification des valeurs contenues.

***$tableau= array(   
cle1=>valeur1 ,  
cle2=>valeur2 ,  
...);***

On sépare donc les différentes clés valeur par des virgules. **Pour accéder à un élément du tableau, on peut faire comme précédemment en indiquant la valeur de la clé comme indice**.

*<?php   
$personne = array(   
"prenom" => "Jessy",   
"nom" => "Brown",   
"telephone" => "000011111"  
);  
//affichage de l'élément nom  
echo $personne['nom'];  
?>*

# Afficher tous les éléments d'un tableau foreach

Si on veut afficher tous les éléments du tableau, en affichant chaque clé et chaque valeur, on peut utiliser l'instruction **foreach**.

*<?php   
$personne = array(  
"prenom" => "Jessy",  
"nom" => "Brown",  
"telephone" => "000011111"*

*);  
  
foreach ($personne as $key => $value) {  
echo "Clé: ". $key . ", Valeur : ". $value . "<br>";  
  
}  
?>*

# Tableaux multidimensionnels

Dans notre exemple, on a eu à gérer un tableau avec une seule personne. Vous imaginez peut être un tableau de plusieurs personnes, dont chaque personne a son prénom, son nom et son numéro de téléphone. Comment faire ?

Il faut créer un **tableau multidimensionnel**. Pour faire plus simple, nous nous limiterons aux tableaux à **deux dimensions**, sachant qu'on peut en faire à plusieurs dimensions.

Définissons alors des personnes avec pour chacun : un numéro, un nom, un prénom et un numéro de téléphone. Ainsi, on peut accéder à leur valeur individuellement (par exemple on veut juste avoir le prénom de la première personne).

*<?php   
$personnes = array(  
1 => array('prenom' => 'Jessy', 'nom' => 'Brown', 'telephone' => '00001111'),  
2 => array('prenom' => 'Sharon', 'nom' => 'Dain', 'telephone' => '00221111'),  
3 => array('prenom' => 'Marta', 'nom' => 'Blanca', 'telephone' => '003311111')  
);  
echo $personnes[1]['prenom']; //Jessy  
?>*

En utilisant l'instruction foreach , on aura :

*<?php  
$personnes = array(  
1 => array('prenom' => 'Jessy', 'nom' => 'Brown', 'telephone' => '00001111'),  
2 => array('prenom' => 'Sharon', 'nom' => 'Dain', 'telephone' => '00221111'),  
3 => array('prenom' => 'Marta', 'nom' => 'Blanca', 'telephone' => '003311111')  
);  
  
foreach($personnes as $cle1 => $valeur1)  
{  
echo "personne n°:" . $cle1 . "<br />";  
  
foreach ($valeur1 as $cle2=>$valeur2)  
  
{  
echo "Clé : ".$cle2 .", Valeur: " . $valeur2 . "<br />\n";  
}  
}  
?>*

# Manipuler les tableaux avec des fonctions

**PHP propose de nombreuses fonctions pour manipuler les tableaux.**

# Connaitre la taille d'un tableau count

Vu précédemment.

On entend par taille d'un tableau, le nombre d'éléments qu’il contient.

Cela est très utile au niveau des **compteurs de boucle**. Si l'on désire utiliser une boucle pour l'affichage d'un tableau, il est nécessaire de connaitre la taille d'un tableau pour spécifier l'arrêt du compteur.

La fonction PHP **count()** renvoie la taille du tableau qu'on lui passe en paramètre.

*< ?php   
$tab[0]=1;   
$tab[1]=2;   
$tab[2]=3;   
$tab[3]=4;   
$tab[4]=5;   
$tab[5]=6;   
$taille =count($tab);   
echo 'La taille du tableau est :'.$taille;  
?>*

# Recherche d'un élément dans un tableau in\_array

La fonction **in\_array()** indique **si une valeur appartient à un tableau**. Cette fonction nous informe de la présence d'un élément dans le tableau en nous renvoyant **TRUE** si elle l'a trouvé et **FALSE** dans le cas contraire. Elle a en premier argument l'élément recherché, et en deuxième argument le tableau en question.

*< ?php   
$etudiant = array ("Alissa", "Marianne", "Mickael", "Shania", "Odile", "Stefanie");  
$recherche = "Marianne";  
if(in\_array($recherche, $etudiant))  
{  
echo $recherche." a été trouvé dans la liste des étudiants";  
}  
else  
{  
echo $recherche." n'est pas dans la liste des étudiants";  
}  
?>*

# Recherche de la clé d'un élément d’un tableau array\_search

Une fonction nous permet d'avoir **la clé correspondante de l'élément recherché**. Il s'agit de la fonction **array\_search()** qui s’utilise de manière similaire à in\_array().

La différence est qu'elle retourne la clé correspondante de l'élément qu'elle a trouvé et FALSE si l'élément n'a pas été trouvé.

*< ?php   
$etudiant = array ("Alissa", "Marianne", "Mickael", "Shania", "Odile", "Stefanie");  
$recherche = "Marianne";  
$cle = array\_search($recherche, $etudiant);  
if($cle == FALSE)  
{  
echo $recherche." n'est pas dans la liste des étudiants";   
}  
else  
{  
echo $recherche." a été trouvé dans la liste des étudiants. Son rang est : ".$cle ;  
}  
?>*

# Nombre d'occurrences d'un élément array\_count\_value

**La fonction array\_count\_value() renvoie pour chaque valeur le nombre d'occurrences trouvé dans un tableau associatif.**

Prenons toujours l'exemple des étudiants, en comptant le nombre de fois où Marianne apparait dans le tableau.

*< ?php   
$etudiant = array ("Alissa", "Marianne", "Mickael", "Shania", "Odile", "Stefanie", "Marianne");  
$occurence = array\_count\_values($etudiant);  
echo "L'étudiante Marianne apparait ". $occurence['Marianne']." fois dans le tableau";  
?>*

# Suppression de doublons array\_unique

Nous avons vu que Marianne apparait deux fois dans le tableau. Dans le cas où l'on a besoin de **supprimer les doublons, la fonction array\_unique()** est à notre disposition. Elle retourne un nouveau tableau sans doublons quand on lui passe en paramètre le premier tableau.

*< ?php   
$etudiant = array ("Alissa", "Marianne", "Mickael", "Shania", "Odile", "Stefanie", "Marianne");  
//tableau avant : avec doublons  
echo "avec doublons : ";  
echo "<pre?>";  
print\_r($etudiant);  
echo "</pre?>";  
  
//enlever les doublons  
$etudiant= array\_unique($etudiant);  
  
echo "sans doublons : ";   
//tableau après : sans doublons  
echo "<pre?>";  
print\_r($etudiant);  
echo "</pre?>";  
?>*

# Trier les tableaux

Il est possible de trier un tableau de **différentes manières**.

Les fonctions intégrées de PHP nous simplifient la tâche.

# Tri d'un tableau par valeur sort

La manière la plus simple et la plus utilisée pour trier les tableaux est de les trier par valeur.

Si l'on veut avoir par exemple un tableau des étudiants par ordre alphabétique, la fonction **sort()** permet de faire le tri. L'argument fourni sera le tableau lui-même.

***< ?php  
$etudiant = array ("Alissa", "Mickael", "Shania", "Odile", "Stefanie", "Marianne");   
echo "avant tri : ";   
echo "<pre>";   
print\_r($etudiant); //une fonction d’affichage bien pratique  
echo "</pre>";  
//trier le tableau  
sort($etudiant);  
//après tri  
echo "après tri : ";  
echo "<pre>";  
print\_r($etudiant);  
echo "</pre>";  
?>***

# Les clés d'un tableau SORT\_NUMERIC, SORT\_STRING

Quelle est la différence entre ces 2 listes triées ?

|  |  |
| --- | --- |
| ***1 2 3 4 11 12 13*** | ***1 11 12 13 2 3 4*** |

Pour forcer une **comparaison numérique**, il faut fournir **SORT\_NUMERIC** en second paramètre de la fonction **sort()**. Il en est de même pour la **comparaison textuelle** : **SORT\_STRING**.

# Tri inversé rsort

Il est aussi possible de faire des tris en ordre inverse, c'est-à-dire un **tri décroissant**.

Cela est possible grâce à la fonction **rsort()**.

# Fonctions de tri ksort, krsort, natsort et usort

Une variété très complète de façons de trier un tableau :

* tri par clé : **ksort() et krsort()**
* tri naturel : **natsort()**
* tri avec une fonction utilisateur : **usort()**

**A découvrir et utiliser selon vos besoins.**