Compléments sur les bases de PHP

Alexandre Niveau

GREYC — Université de Caen

En partie adapté du cours de Jean-Marc Lecarpentier

Bizarreries des tableaux

- Le fait que les tableaux, même numériques, soient en fait des *ordered map* peut causer des surprises !

```php
<?php
$tab = array();
$tab[0] = "zéro";
$tab[1] = "un";
$tab[100] = "cent";
$tab[] = "je suis le dernier";
$tab[2] = "deux";
$tab[] = "où suis-je ?";
foreach ($tab as $key => $val) {
  echo "$key --> $val\n";
}
```

```
0 --> zéro
1 --> un
100 --> cent
101 --> je suis le dernier
2 --> deux
102 --> où suis-je ?
```

- Lors du parcours avec un foreach, seul l'ordre d'insertion dans le tableau compte.
- La construction `$tab[] = ...` ajoute la valeur à une clef *numérique* qui est la plus grande clef numérique + 1
- Attention, on ne peut pas utiliser comme indice une chaîne de caractères « contenant » un entier, car elle est convertie en entier. Une façon de l'éviter est de faire commencer la chaîne par un +.

```php
<?php
$tab = array();
$tab['0'] = 'chaîne 0';
$tab[0] = 'entier 0';
$tab['1.0'] = 'chaîne 1.0';
$tab[1.0] = 'flottant 1.0';
$tab[1.1] = 'flottant 1.1';
$tab['+1'] = 'chaîne +1';
foreach ($tab as $key => $val) {
  echo "$key --> $val\n";
}
```

```
0 --> entier 0
1.0 --> chaîne 1.0
1 --> flottant 1.1
+1 --> chaîne +1
```

détails sur la conversion des indices dans le manuel [http://php.net/manual/en/language.types.array.php]

- Les tableaux sont manipulés « par copie » : dans le code suivant, la 2e ligne crée

une copie du tableau !

```php
<?php
$x = array('toto' => 1, 'tata' => 2);
$y = $x;
$x['titi'] = 3;
echo count($x) . "\n";
echo count($y) . "\n";
```

```
3
2
```

- Ce n'est pas une erreur d'utiliser un nombre ou `null` comme un tableau. Ça renvoie simplement toujours `null`.

```php
$tab = null;
var_export($tab['toto']);  // NULL
$tab = 1234;
var_export($tab['toto']);  // NULL
```

Fonctions utiles avec les tableaux

- Principales fonctions utiles avec les tableaux :
  - *`key_exists($key, $array)`* : renvoie `true` si le tableau `$array` contient la clef `$key`, et `false` sinon (il existe l'alias plus long `array_key_exists`)
  - *`in_array($val, $array, true)`* : renvoie `true` si le tableau `$array` contient la valeur `$val`, et `false` sinon.
    ➤ Le troisième paramètre sert à utiliser la comparaison stricte (celle de ===) plutôt que la comparaison faible (celle de ==), qui est le défaut. **À ne pas oublier !** Si on l'enlève ou si on le met à `false`, les résultats peuvent être très surprenants... [http://php.net/manual/en/function.in-array.php#106319]
  - *`array_search($val, $array, true)`* pour récupérer l'indice d'une valeur dans un tableau. Attention à la comparaison faible :
    - il faut passer `true` comme 3e paramètre pour que la comparaison lors de la recherche soit stricte, comme pour `in_array`
    - mais il faut aussi faire attention à la valeur de retour : la fonction peut renvoyer 0 ou `false`, ce n'est pas du tout pareil !
  - *`unset($array[$key])`* pour supprimer un élément ; **attention**, il n'y a pas de redécalage des indices, utiliser `array_values` derrière si besoin
  - Voir aussi `array_keys`, `array_values`, `array_slice`, et éventuellement `array_flip`
- toutes les fonctions sur les tableaux [https://www.php.net/manual/en/ref.array.php]
- tuto sur la manipulation de tableaux [https://code.tutsplus.com/tutorials/working-with-php-arrays-in-the-right-way--cms-28606]

Fonctions anonymes

- PHP permet d'utiliser des fonctions anonymes, comme en JavaScript.
- On peut ainsi mettre une fonction dans une variable :

```php
<?php
$toto = function () {
    echo "je suis anonyme\n";
};
$toto();

$titi = function ($x) {
    echo "\$x = $x\n";
};
$titi("salut");

$titi = $toto;
$titi();
```

```
je suis anonyme
$x = salut
je suis anonyme
```

- Cela évite de déclarer une fonction qui n'est utilisée qu'une seule fois

  ➤ Particulièrement utile comme paramètre de certaines fonctions (voir la suite)

Map/Filter/Reduce sur des tableaux

- Exemple d'utilisation de fonctions anonymes : filtrage ou modification des valeurs d'un tableau à l'aide d'une fonction
- Modifier les valeurs d'un tableau : `array_map($fonction, $tab)` applique la fonction `$fonction` à chaque élément du tableau `$tab` et retourne le résultat (sans modifier `$tab`)
- Filtrer les valeurs d'un tableau : `array_filter($tab, $fonction)` renvoie un tableau ne contenant que les valeurs du tableau `$tab` pour lesquelles la fonction `$fonction` a renvoyé `true`

```php
<?php
$tab = [ 1, 12, 3, 7, 15, 20, -24 ];
echo "Tableau initial :\n";
var_export($tab);
echo "\n";

$tab2 = array_map(function ($value) {
  return $value * 3;
}, $tab);
echo "Tableau modifié :\n";
var_export($tab2);
echo "\n";

$tab3 = array_filter($tab2, function ($value) {
  return $value < 25;
});
echo "Tableau résultant filtré :\n";
var_export($tab3);
echo "\n";

echo "NB: le tableau initial\n n'a pas bougé :
\n";
var_export($tab);
echo "\n";
```

```
Tableau initial :
array (
  0 => 1,
  1 => 12,
  2 => 3,
  3 => 7,
  4 => 15,
  5 => 20,
  6 => -24,
)
Tableau modifié :
array (
  0 => 3,
  1 => 36,
  2 => 9,
  3 => 21,
  4 => 45,
  5 => 60,
  6 => -72,
)
Tableau résultant filtré :
array (
  0 => 3,
  2 => 9,
  3 => 21,
  6 => -72,
)
```

```
NB: le tableau initial
 n'a pas bougé :
array (
  0 => 1,
  1 => 12,
  2 => 3,
  3 => 7,
  4 => 15,
  5 => 20,
  6 => -24,
)
```

- *Attention*, attention, l'ordre des paramètres n'est pas le même dans les deux fonctions
- Dans la même famille, mais plus subtil, il existe aussi array_reduce [https://www.php.net/manual/fr/function.array-reduce.php], pour effectuer une opération sur toutes les valeurs à la fois
  ➤ cependant les réductions les plus courantes ont une fonction dédiée :
  - `array_sum` pour la somme, `array_product` pour le produit
  - `min` et `max` pour le minimum et le maximum

Tableaux de tableaux

- On peut bien sûr mettre des tableaux dans des tableaux, et récursivement, sans difficulté :

```php
<?php
$couleurs = array(
    "aqua" => array(0, 255, 255),
    "black" => array(0, 0, 0),
    "blue" => array(0, 0, 255),
);

$couleurs["white"] = array(255, 255, 255);

// On parcourt le tableau $couleurs
foreach ($couleurs as $c => $rgb) {
    // Ici $rgb est lui-même un tableau
    echo "$c = rgb($rgb[0], $rgb[1], $rgb[2])\n";
}
```

```
aqua = rgb(0, 255, 255)
black = rgb(0, 0, 0)
blue = rgb(0, 0, 255)
white = rgb(255, 255, 255)
```

Tableaux en paramètres des fonctions

- Passer un gros tableau à une fonction peut sembler une mauvaise idée, car la sémantique des paramètres est celle du « passage par copie »
- Cependant, le moteur de PHP fait cette copie *paresseusement*, c'est-à-dire uniquement au moment où ça devient nécessaire

  ➤ si le tableau n'est jamais modifié, il n'est jamais copié !
- On a parfois besoin de faire des fonctions qui *modifient* un tableau donné en place ; dans ce cas on peut passer en paramètre une *référence* vers le tableau, en ajoutant `&` devant le paramètre formel

```php
<?php
function no_modif($tab) {
  $tab[0] = 'youpi';
}
function yes_modif(&$tab) {
  $tab[0] = 'youpi';
}

$x = [ 'toto', 'tutu' ];
echo "no_modif ne modifie pas le tableau :
\n";
no_modif($x);
var_export($x);

echo "\n--------\n";
echo "yes_modif modifie le tableau :\n";
yes_modif($x);
var_export($x);
```

```
no_modif ne modifie pas le tableau
array (
  0 => 'toto',
  1 => 'tutu',
)
--------
yes_modif modifie le tableau :
array (
  0 => 'youpi',
  1 => 'tutu',
)
```

- NB : le `&` est à mettre seulement lors de la définition de la fonction, pas lorsqu'on l'appelle. On ne passe pas explicitement un « pointeur ».

Manipulation simple de fichiers

- Les fonctions suivantes sont simples à utiliser, mais pas forcément pertinentes sur de gros fichiers (on charge tout le fichier en mémoire).
- Lire le contenu d'un fichier : `file_get_contents("nomdufichier")`, qui renvoie le contenu du fichier sous forme d'une chaîne de caractères.
- Pour récupérer les lignes du fichier, on peut utiliser `file("nomdufichier")`, qui renvoie un tableau contenant toutes les lignes du fichier (avec les sauts de ligne).
- Écrire un fichier : `file_put_contents("nomdufichier", "contenu")`
- Ajouter du contenu à un fichier : `file_put_contents("nomdufichier", "contenu", FILE_APPEND)` (on utilise un paramètre optionnel de la fonction)

```php
<?php
$file = "demo/fichier.txt";

$contents = file_get_contents($file);
echo "Contenu du fichier, V1 :
\n$contents";

// get contents of a file line by line
$lines = file($file);
foreach ($lines as $num => $line) {
  echo "Ligne $num : « $line »\n";
}
```

```
Contenu du fichier, V1 :
Coucou,
je suis un fichier !

Au revoir

Ligne 0 : « Coucou,
 »
Ligne 1 : « je suis un fichier !
 »
Ligne 2 : «
 »
Ligne 3 : « Au revoir
 »
Ligne 4 : «
 »
```

Manipulation subtile de fichiers

- Pour des besoins plus précis, ou pour des fichiers lourds, il existe des fonctions bas niveau c'est en fait une surcouche très fine de la bibliothèque standard de C :
  - La fonction `fopen` prend en 2e paramètre le mode d'ouverture du fichier : `r r+ w w+ a a+`
  - La fonction `fread` lit le contenu du fichier jusqu'à une limite donnée
  - La fonction `fgets` envoie la ligne courante sur laquelle se trouve le pointeur du fichier
  - La fonction `fwrite` permet d'écrire dans un fichier
  - La fonction `fclose` permet de fermer la ressource (elle est cependant fermée automatiquement à la fin du script)

```php
<?php
$file = "demo/fichier.txt";

// à l'ancienne
$fd = fopen( $file, "r" );
$contents = fread($fd, filesize($file));
echo "Contenu du fichier, V2 :
\n$contents";
fclose($fd);

// get contents of a file line by line
$fd = fopen( $file, "r" );
while ($line = fgets($fd, 4096)) {
  echo "<p>une ligne : « $line »</p>";
}
fclose($fd);
```

```
Contenu du fichier, V2 :
Coucou,
je suis un fichier !

Au revoir


une ligne : « Coucou,
 »

une ligne : « je suis un fichier !
 »

une ligne : «
 »

une ligne : « Au revoir
 »

une ligne : «
 »
```

Expressions régulières

- Plusieurs fonctions existent [https://www.php.net/manual/fr/ref.pcre.php] pour faire des recherches et remplacements dans des chaînes ou des tableaux en utilisant des expressions régulières (regex)
- Le format des regex est celui de Perl («PCRE»), comme la plupart des langages modernes (notamment Python ou Java)
- Voir la doc pour tous les détails sur le format des regex ou les options des fonctions [https://www.php.net/manual/fr/book.pcre.php]

Spécifications et normes

- Manuel de PHP [http://www.php.net/manual/fr/index.php] (en français)

Tutoriels

- Working with PHP arrays the right way [https://code.tutsplus.com/tutorials/working-with-php-arrays-in-the-right-way--cms-28606]

 [http://creativecommons.org/licenses/by-nc-sa/4.0/]