Université de Montréal

COURS 3
TABLEAUX EN PHP

Les tableaux en PHP (à partir de la version 8 et plus) demeurent un pilier fondamental du langage. Ils sont très flexibles (indexés ou associatifs), faciles à manipuler (avec les nombreuses fonctions natives) et permettent des structures de données complexes (multidimensionnelles). Les nouveautés comme array_is_list() et l'utilisation de propriétés typées rendent le code plus robuste et plus clair.

IFT1147
PROGRAMMATION WEB COTE SERVEUR AVEC PHP

1. INTRODUCTION AUX TABLEAUX EN PHP

En PHP, un tableau est une structure de données qui peut contenir plusieurs valeurs, référencées par des clés.

- Les clés peuvent être **numériques** (pour un tableau indexé) ou **textuelles** (pour un tableau associatif).
- PHP gère également les tableaux multidimensionnels (tableaux contenant d'autres tableaux).

1.1. LES NOUVEAUTÉS ET ASPECTS MODERNES

- **Syntaxe courte []** : depuis PHP 5.4, on préfère la syntaxe [] à la place de array(). C'est désormais la norme en PHP 8.
- **Propriétés typées (PHP 7.4+)** : si vous avez une classe qui contient une propriété de type tableau, vous pouvez déclarer cela explicitement, par exemple private array $monTableau;.
- **Fonction array_is_list() (PHP 8.1+)** : permet de déterminer si un tableau est une "liste" (i.e. ses clés sont des entiers consécutifs à partir de 0).

Exemple d'utilisation de array_is_list() :

```php
<?php

$fruits = ['Pomme', 'Poire', 'Banane'];

$assoc = ['fruit1' => 'Pomme', 'fruit2' => 'Poire'];

var_dump(array_is_list($fruits)); // bool(true)

var_dump(array_is_list($assoc));  // bool(false)
```

2. CRÉATION DE TABLEAUX

2.1. TABLEAU INDEXÉ (NUMÉRIQUE)

```php
<?php

// Syntaxe moderne

$fruits = ['Pomme', 'Poire', 'Banane'];

// Accès à un élément

echo $fruits[0]; // Pomme

echo $fruits[2]; // Banane
```

- Les clés sont gérées automatiquement à partir de 0.

2.2. TABLEAU ASSOCIATIF

```php
<?php

// Syntaxe moderne

$livre = [

   'titre'  => 'Le Petit Prince',

   'auteur' => 'Antoine de Saint-Exupéry',

   'annee'  => 1943,

];

echo $livre['titre']; // Le Petit Prince
```

- Les clés sont définies manuellement et peuvent être des chaînes de caractères ou des entiers (mais non séquentiels).

3. MANIPULATION DE TABLEAUX

3.1. AJOUTER DES ÉLÉMENTS

Dans un tableau indexé

```php
<?php

$fruits = ['Pomme', 'Poire'];

$fruits[] = 'Banane';

// $fruits => ['Pomme', 'Poire', 'Banane']

array_push($fruits, 'Fraise', 'Orange');

// $fruits => ['Pomme', 'Poire', 'Banane', 'Fraise', 'Orange']
```

Dans un tableau associatif

```php
<?php
$livre = [
   'titre'  => 'Le Petit Prince',
   'auteur' => 'Antoine de Saint-Exupéry',
];
$livre['annee'] = 1943;
// $livre => ['titre' => 'Le Petit Prince', 'auteur' => 'Antoine de Saint-Exupéry', 'annee' => 1943]
```

3.2. METTRE À JOUR UN ÉLÉMENT

```php
<?php

$fruits = ['Pomme', 'Poire', 'Banane'];

$fruits[1] = 'Abricot';

// $fruits => ['Pomme', 'Abricot', 'Banane']

$livre['auteur'] = 'A. de Saint-Exupéry';

// Met à jour la valeur de 'auteur'
```

3.3. SUPPRIMER UN ÉLÉMENT

Pour supprimer un élément spécifique, on utilise unset() :

```php
<?php

$fruits = ['Pomme', 'Poire', 'Banane'];

unset($fruits[1]);

// $fruits => [0 => 'Pomme', 2 => 'Banane']
```

Attention : la clé 1 est supprimée, mais les autres clés ne sont pas réindexées automatiquement.

4. PARCOURIR UN TABLEAU

4.1. BOUCLE FOREACH

C'est la méthode la plus simple et la plus moderne pour parcourir un tableau :

```php
<?php

$fruits = ['Pomme', 'Poire', 'Banane'];

foreach ($fruits as $fruit) {

    echo $fruit . "\n";

}

// Pour récupérer à la fois la clé et la valeur :
foreach ($fruits as $index => $fruit) {

    echo "Index : $index, Fruit : $fruit\n";

}
```

Avec un tableau associatif :

```php
<?php
$livre = [
    'titre'  => 'Le Petit Prince',
    'auteur' => 'Antoine de Saint-Exupéry',
    'annee'  => 1943,
];
foreach ($livre as $cle => $valeur) {
    echo "Clé : $cle, Valeur : $valeur\n";
}
```

4.2. BOUCLE FOR

Moins utilisée pour les tableaux en PHP moderne, mais peut servir :

```php
<?php
$fruits = ['Pomme', 'Poire', 'Banane'];
$taille = count($fruits);


for ($i = 0; $i < $taille; $i++) {
    echo $fruits[$i] . "\n";
}
```

5. FONCTIONS UTILES SUR LES TABLEAUX

PHP propose énormément de fonctions natives pour manipuler les tableaux. Voici les plus courantes :

5.1. COUNT() ET SIZEOF()

- **count($array)** : renvoie le nombre d'éléments du tableau.
- **sizeof($array)**: alias de count().

```php
<?php

$fruits = ['Pomme', 'Poire', 'Banane'];

echo count($fruits);  // 3
```

5.2. ARRAY_PUSH() ET ARRAY_POP()

- **array_push($array, $val1, $val2, ...)** : ajoute un ou plusieurs éléments à la fin du tableau.
- **array_pop($array)** : retire et renvoie le dernier élément du tableau.

```php
<?php

$fruits = ['Pomme', 'Poire'];

array_push($fruits, 'Banane', 'Fraise');

// $fruits => ['Pomme', 'Poire', 'Banane', 'Fraise']

$dernier = array_pop($fruits);

// $dernier => 'Fraise'

// $fruits => ['Pomme', 'Poire', 'Banane']
```

5.3. ARRAY_SHIFT() ET ARRAY_UNSHIFT()

- **array_shift($array)** : retire et renvoie le premier élément du tableau.
- **array_unshift($array, $val1, ...)** : ajoute un ou plusieurs éléments au début du tableau.

```php
<?php

$fruits = ['Pomme', 'Poire', 'Banane'];

$premier = array_shift($fruits);

// $premier => 'Pomme'

// $fruits => ['Poire', 'Banane']

array_unshift($fruits, 'Fraise', 'Cerise');

// $fruits => ['Fraise', 'Cerise', 'Poire', 'Banane']
```

5.4. EXPLODE() ET IMPLODE()

- **explode($delimiter, $string)** : découpe une chaîne selon un délimiteur en un tableau.
- **implode($delimiter, $array)** : rassemble les éléments d'un tableau en une chaîne (délimiteur).

```php
<?php

$chaine = "Pomme,Poire,Banane";

$fruits = explode(',', $chaine);

// $fruits => ['Pomme', 'Poire', 'Banane']

$fruitsString = implode('-', $fruits);

// $fruitsString => "Pomme-Poire-Banane"
```

5.5. IN_ARRAY() ET ARRAY_SEARCH()

- **in_array($value, $array)** : renvoie true si la valeur existe, false sinon.
- **array_search($value, $array)** : renvoie la clé de la valeur si elle existe, false sinon.

```php
<?php

$fruits = ['Pomme', 'Poire', 'Banane'];

if (in_array('Banane', $fruits)) {

    echo "Banane est présente\n";

}

$cle = array_search('Poire', $fruits);

if ($cle !== false) {

    echo "Poire a pour clé : $cle\n"; // Affiche 1

}
```

5.6. ARRAY_MERGE()

Combine plusieurs tableaux en un seul :

```php
<?php

$fruits1 = ['Pomme', 'Poire'];

$fruits2 = ['Banane', 'Fraise'];

$allFruits = array_merge($fruits1, $fruits2);

// $allFruits => ['Pomme', 'Poire', 'Banane', 'Fraise']
```

5.7. SORT(), ASORT(), KSORT()

- **sort($array)** : trie les valeurs dans l'ordre croissant (et réindexe).
- **asort($array)** : trie les valeurs tout en **préservant les clés** (associatifs).
- **ksort($array)** : trie le tableau associatif en fonction de ses clés.

```php
<?php

$fruits = ['Fraise', 'Pomme', 'Banane', 'Abricot'];

sort($fruits);

// $fruits => ['Abricot', 'Banane', 'Fraise', 'Pomme']

$personne = [

   'nom'   => 'Dupont',

   'prenom'=> 'Jean',

   'ville' => 'Paris'

];

asort($personne);

// Tri par valeur en conservant les clés

// Résultat (exemple) : ['ville' => 'Paris', 'prenom' => 'Jean', 'nom' => 'Dupont']


ksort($personne);

// Tri par clé

// Résultat : ['nom' => 'Dupont', 'prenom' => 'Jean', 'ville' => 'Paris']
```

5.8. ARRAY_IS_LIST() (PHP 8.1+)

Vérifie qu'un tableau est une liste : c'est-à-dire que les clés sont des entiers consécutifs commençant à 0.

```php
<?php

$array1 = ['Pomme', 'Poire', 'Banane']; // liste

$array2 = [1 => 'Pomme', 2 => 'Poire'];  // pas une liste

$array3 = ['fruit1' => 'Fraise'];       // pas une liste

var_dump(array_is_list($array1)); // true

var_dump(array_is_list($array2)); // false

var_dump(array_is_list($array3)); // false
```

6. TABLEAUX MULTIDIMENSIONNELS

Un tableau multidimensionnel est un tableau dont les éléments sont eux-mêmes des tableaux.

Exemple :

```php
<?php

$etudiants = [

    ['nom' => 'Dupont', 'age' => 20, 'ville' => 'Paris'],

    ['nom' => 'Durand', 'age' => 22, 'ville' => 'Lyon'],

    ['nom' => 'Moreau', 'age' => 19, 'ville' => 'Marseille'],

];

// Accès direct

echo $etudiants[1]['ville']; // Lyon

// Parcours

foreach ($etudiants as $etudiant) {

    echo "Nom : {$etudiant['nom']}, Age : {$etudiant['age']}, Ville : {$etudiant['ville']}\n";

}
```

7. BONNES PRATIQUES (PHP 8+)

1. **Utiliser la syntaxe courte []** : plus lisible et plus standard pour définir des tableaux.
2. **Propriétés typées (optionnel)** : si vous utilisez un tableau en tant que propriété de classe, vous pouvez préciser private array $monTableau; pour plus de clarté et de robustesse (à partir de PHP 7.4).
3. **foreach pour la lisibilité** : privilégiez foreach à la place de for pour la plupart des usages.
4. **Faire attention au réindexage** : certaines fonctions comme sort(), array_values(), etc., réindexent les clés.
5. **Vérifier si un tableau est une liste** (si besoin) avec array_is_list() en PHP 8.1+.
6. **Utiliser des noms de variables cohérents** : cela améliore la lisibilité (ex. $fruits, $livres, $personnes, etc.).