# Formation Développement Web : Backend : PHP

## Partie : 2

## I. Les conditions en PHP

### A. Conditions simples

Une condition simple consiste en une instruction **'if'** qui permet d'exécuter un bloc de code si une certaine condition est vraie. Voici comment vous pouvez expliquer cela dans votre cours :

#### 1. Syntaxe

```
<?php
    if (condition) {
       // Code à exécuter si la condition est vraie
    }
?>
```

#### 2. Exemple

```
<?php
    $age = 18;

    if ($age >= 18) {
       echo "Vous êtes majeur.";
    } else {
       echo "Vous êtes mineur.";
    }
?>
```

### B. Conditions complex

Les conditions complexes peuvent inclure des opérateurs logiques tels que **'&&'** (et), '||' (ou), et **'!'** (non) pour combiner plusieurs conditions. Voici une explication simple :

#### 1. Syntaxe

```
<?php
    if (condition1 && condition2) {
       // Code à exécuter si condition1 ET condition2 sont vraies
    } elseif (condition3 || condition4) {
       // Code à exécuter si condition3 OU condition4 est vraie
    } else {
       // Code à exécuter si aucune des conditions n'est vraie
    }
?>
```

#### 2. Exemple

```
<?php
    $score = 85;

    if ($score >= 90 && $score <= 100) {
       echo "Excellent !";
    } elseif ($score >= 70 && $score < 90) {
```

```
        echo "Bien joué !";
    } else {
        echo "Besoin d'amélioration.";
    }
?>
```

### C. Switch case

**'switch'** est utilisé pour gérer plusieurs cas différents en fonction de la valeur d'une expression. Voici comment vous pouvez expliquer cela :

#### 1. Syntaxe

```
<?php
    switch ($expression) {
        case $valeur1:
            // Code à exécuter si $expression équivaut à $valeur1
            break;
        case $valeur2:
            // Code à exécuter si $expression équivaut à $valeur2
            break;
        // Vous pouvez ajouter autant de cas que nécessaire
        default:
            // Code à exécuter si aucune des valeurs ne correspond
    }
?>
```

#### 2. Exemple

```
<?php
    $jourSemaine = "lundi";

    switch ($jourSemaine) {
        case "lundi":
            echo "C'est le début de la semaine.";
            break;
        case "vendredi":
            echo "Le week-end approche !";
            break;
        default:
            echo "Jour ordinaire.";
    }
?>
```

## II. Les boucles

### A. La boucle For()

La boucle For() permet de répéter une bloc de code plusieurs fois, cette boucle se compose de trois partie :

- L'initialisation d'une variable : la valeur de départ
- La condition d'arrête :Valeur d'arriver
- La mise à jour de la variable : Les Pas

1. Syntaxe

```
<?php
    for ($valeur_depart = 0; $valeur_depart <= $valeur_arrivee; $valeur_depart++) {
       // Code à exécuter à chaque itération
    }
?>
```

2. Exemple

```
<?php
    for ($i = 0; $i <= 5; $i++) {
       echo "La valeur de i est : " . $i . "<br>";
    }
?>
```

*a) Résulta*

**La valeur de I est : 0**
**La valeur de I est : 1**
**La valeur de I est : 2**
**La valeur de I est : 3**
**La valeur de I est : 4**
**La valeur de I est : 5**

## B. La boucle While()

La boucle While() est une structure de contrôle en programmation qui exécute un bloc de code tant qu'une condition est vraie.

1. Syntaxe

```
<?php
    while (condition) {
       // Code à exécuter tant que la condition est vraie
    }
?>
```

2. Exemple

```
<?php
    $a = 0;

    while ($a < 6) {
       $a++;
       echo "La valeur de A est : " . $a . "<br>";
    }
?>
```

*a) Résulta*

**La valeur de A est : 1**
**La valeur de A est : 2**
**La valeur de A est : 3**
**La valeur de A est : 4**
**La valeur de A est : 5**
**La valeur de A est : 6**

### C. La boucle Do While()

La boucle Do While() est une structure de contrôle en programmation qui exécute un bloc de code au moins une fois, puis continue à le répéter tant qu'une condition donnée est vraie.

#### 1. Syntaxe

```
<?php
    do {
      // Code à exécuter au moins une fois
    } while (condition);
?>
```

#### 2. Exemple

```
<?php
    $a = 0;

    do {
      $a++;
      echo "La valeur de A est : " . $a . "<br>";
    } while ($a <= 2);
?>
```

##### *a) Résulta*

**La valeur de A est : 1**
**La valeur de A est : 2**
**La valeur de A est : 3**

## III. Tableau

Les tableaux sont des structures de données essentielles en programmation. Ils vous permettent de stocker et de manipuler des ensembles de données de manière organisée. En PHP, vous pouvez travailler avec des tableaux à une dimension (tableaux simples) ou à deux dimensions (tableaux associatifs). Voici comment travailler avec les tableaux en PHP :

### A. Déclarer un tableau

#### 1. Tableau simple

Un tableau simple est une structure de données dans laquelle des éléments sont stockés de manière séquentielle, chaque élément étant accessible par un index unique.

##### *a) Syntaxe* à une dimension

```
<?php
    // Tableau simple (à une dimension)
    $monTableau = array(1, 2, 3, 4, 5);
?>
```

##### *b) Syntaxe à deux dimensions*

```
<?php
    // Tableau simple à deux dimensions (tableau de tableaux)
    $ monTableauDeuxDimensions = array(
      array(1, 2, 3),
      array(4, 5, 6),
      array(7, 8, 9)
    );
?>
```

### 2. Tableau associatif

Un tableau associatif est une structure de données dans laquelle des paires clé-valeur sont stockées de manière séquentielle, chaque paire étant accessible par une clé unique.

#### *a) Syntaxe à une dimension*

```php
<?php
    // Tableau associatif (à une dimension)
    $personne = array("nom" => "Karim", "âge" => 30, "ville" => "Fès");
?>
```

#### *b) Syntaxe à deux dimensions*

```php
<?php
    // Tableau à deux dimensions (tableau de tableaux)
    $etudiants = array(
       array("nom" => "Said", "note" => 95),
       array("nom" => "Oumayma", "note" => 87),
       array("nom" => "Sihame", "note" => 92)
    );
?>
```

## B. Récupérer des Valeurs d'un Tableau par des Indices ou des Clés

### 1. Tableau simple

#### a) *Syntaxe* à une dimension

```php
<?php
    $valeur = $monTableau[2]; // Récupère la valeur à l'indice 2 (qui est 3)
    echo "La valeur à l'indice 2 est : " . $valeur;
?>
```

#### *b) Syntaxe à deux dimensions*

```php
<?php
    $valeur = $monTableauDeuxDimensions [1][2]; // Récupère la valeur à l'indice 1,2 (qui est 6)
    echo "La valeur à l'indice 1,2 est : " . $valeur;
?>
```

### 2. Tableau associatif

#### a) *Syntaxe* à une dimension

```php
<?php
    $nom = $personne["nom"]; // Récupère la valeur associée à la clé "nom" (qui est "Karim")
    echo "Le nom de la personne est : " . $nom;
?>
```

#### *b) Syntaxe à deux dimensions*

```php
<?php
    $noteOumayma = $etudiants[1]["note"]; // Récupère la note de l'étudiant à l'indice 1 (Oumayma)
    echo "La note de Oumayma est : " . $noteOumayma;
?>
```

## C. Modifier les Valeurs d'un Tableau

### 1. Tableau simple

#### a) *Syntaxe* à une dimension

```
<?php
    $monTableau[2] = 99; // Modification de la valeur de l'élément à l'indice 2
?>
```

#### *b) Syntaxe à deux dimensions*

```
<?php
    $monTableauDeuxDimensions [0][1] = 5 ; // Modification de la valeur de l'élément à l'indice 0
    , 1
?>
```

### 2. Tableau associatif

#### a) *Syntaxe* à une dimension

```
<?php
    $personne["âge"] = 31; // Modification de l'âge de la personne
?>
```

#### *b) Syntaxe à deux dimensions*

```
<?php
    $etudiants[2]["note"] = 90; // Modification de la note de l'étudiant à l'indice 2
?>
```

## D. Supprimer les Valeurs d'un Tableau

### 1. Tableau simple

#### a) *Syntaxe* à une dimension

```
<?php
    unset($monTableau[0]); // Supprime l'élément à l'indice 0
?>
```

#### *b) Syntaxe à deux dimensions*

```
<?php
    unset($monTableauDeuxDimensions [1]) ; // Supprimer l'élément à l'indice 1
?>
```

### 2. Tableau associatif

#### a) *Syntaxe* à une dimension

```
<?php
    unset($personne["ville"]); // Supprime la clé "ville" du tableau associatif
?>
```

#### *b) Syntaxe à deux dimensions*

```
<?php
    unset($etudiants[0]); // Supprime le premier étudiant du tableau à deux dimensions
?>
```

## E. Vérifier l'existence d'un indice

### 1. Tableau simple

#### a) *Syntaxe* à une dimension

```
<?php
    if (isset($monTableau[2])) {
       echo "La valeur à l'indice 2 existe dans le tableau.";
    } else {
       echo "La valeur à l'indice 2 n'existe pas dans le tableau.";
    }
?>
```

#### *b) Syntaxe à deux dimensions*

```
<?php
    if (isset($monTableauDeuxDimensions [2][0])) {
       echo "La valeur à l'indice 2,0 existe dans le tableau.";
    } else {
       echo "La valeur à l'indice 2,0 n'existe pas dans le tableau.";
    }
?>
```

### 2. Tableau associatif

#### a) *Syntaxe* à une dimension

```
<?php
    if (isset($personne["âge"])) {
       echo "La clé 'âge' existe dans le tableau associatif.";
    } else {
       echo "La clé 'âge' n'existe pas dans le tableau associatif.";
    }
?>
```

#### *b) Syntaxe à deux dimensions*

```
<?php
    if (isset($etudiants[2]["nom"])) {
       echo "La clé 'nom' existe pour l'étudiant à l'indice 2.";
    } else {
       echo "La clé 'nom' n'existe pas pour l'étudiant à l'indice 2.";
    }
?>
```

## F. Les boucles sur les tableaux

### 1. Tableau simple

#### a) *Syntaxe* à une dimension

```
<?php
    for ($i = 0; $i < count($monTableau); $i++) {
       echo $monTableau[$i] . "<br/>";
    }
?>
```

*b) Syntaxe à deux dimensions*

```
<?php
    for ($row = 0; $row < count($monTableauDeuxDimensions); $row++) {
       for ($col = 0; $col < count($monTableauDeuxDimensions[$row]); $col++) {
          echo $monTableauDeuxDimensions[$row][$col] . "<br/>";
       }
        echo "<br>";
    }
?>
```

2. Tableau associatif

a) *Syntaxe* à une dimension

```
<?php
    foreach ($personne as $cle => $valeur) {
       echo $cle." : ".$valeur. "<br>";
    }
?>
```

*b) Syntaxe à deux dimensions*

```
<?php
    foreach ($etudiants as $etudiant) {
       foreach ($etudiant as $cle => $valeur) {
          echo  $cle." : ".$valeur. "<br>";
       }
       echo "<br>";
    }
?>
```

# IV. Les Fonctions en PHP

## A. Définition

En programmation, les fonctions sont des blocs de code réutilisables qui effectuent des tâches spécifiques. Elles permettent d'organiser le code en morceaux plus petits et autonomes, améliorant ainsi la lisibilité, la maintenabilité et la réutilisation du code.

## B. Syntaxe

### 1. Déclarer une fonction

```
<?php
    function maFonction($arg1, $arg2, ...) {
       // Code à exécuter
    }
?>
```

### 2. Exécuter une fonction

```
<?php
    maFonction($arg1, $arg2, ...) ;
?>
```

### C. Fonction sans Paramètre

```
<?php
    function afficher() {
       echo "L'exécution de la fonction afficher()";
    }

    // Exécuter la fonction
    afficher();
?>
```

### D. Fonction avec Paramètre

```
<?php
    function calculerLaSomme($a, $b) {
       $c = $a + $b;
       echo "La somme est : " . $c;
    }

    // Exécuter la fonction
    $x = 3;
    $y = 2;
    calculerLaSomme($x, $y);
?>
```

### E. Fonction qui Retourne une Valeur

```
<?php
    function calculerLaMultiplication($a, $b) {
       $c = $a * $b;
       return $c;
    }

    // Exécuter la fonction
    $x = 3;
    $y = 2;
    $resultat = calculerLaMultiplication($x, $y);
    echo "La multiplication est : " . $resultat;
?>
```