Chapitre : Les Structures conditionnelles et répétitives

- Introduction
- Les Conditions
- Les Boucles
- Exercices
- Conclusion

Par Robert DIASSÉ

Introduction

Les structures en PHP permettent de contrôler le flux d'exécution du code et d'organiser la logique du programme. Elles offrent des mécanismes pour effectuer des actions conditionnelles, répétitives et séquentielles. En comprenant et en maîtrisant ces structures, vous serez en mesure de créer des programmes PHP plus efficaces et flexibles.

Les Conditions

En PHP, les conditions permettent d'exécuter certaines parties de code uniquement si une condition spécifique est remplie. Cela permet de prendre des décisions dans le flux d'exécution du programme en fonction de différentes situations. Les conditions en PHP sont généralement implémentées à l'aide de la structure de contrôle "if", qui évalue une expression ou une condition et exécute le bloc de code qui lui est associé si la condition est évaluée comme vraie.

- La structure générale if :

```
if (condition) {
   // Bloc de code à exécuter si la condition est vraie
} elseif (condition) {
   // Bloc de code à exécuter si une condition alternative est vraie
} else {
   // Bloc de code à exécuter si aucune des conditions précédentes n'est vraie
}
```

La condition est une expression qui est évaluée comme vraie ou fausse.
Si la condition du "if" est évaluée comme vraie, le bloc de code à l'intérieur des accolades {} est exécuté.
Si la condition du "if" est évaluée comme fausse, les conditions des éventuelles instructions "elseif" sont évaluées les unes après les autres. Si une de ces conditions est vraie, le bloc de code correspondant est exécuté.
Si aucune des conditions du "if" ou des "elseif" n'est vraie, le bloc de code à l'intérieur de l'instruction "else" est exécuté.
Il est important de noter que les instructions "elseif" et "else" sont optionnelles. Un "if" peut être utilisé seul sans les conditions alternatives.

Exemple

```php
<?php
//exemple1
$age = 25;
if ($age >= 18) {
    echo "Vous êtes majeur.";
}
//exemple2
$age = 15;
if ($age >= 18) {
    echo "Vous êtes majeur.";
} else {
    echo "Vous êtes mineur.";
}
//exemple3
$note = 80;
if ($note >= 90) {
    echo "Très bien !";
} elseif ($note >= 80) {
    echo "Bien !";
} elseif ($note >= 70) {
    echo "Pas mal.";
} else {
    echo "Il faut s'améliorer.";
}
?>
```

- La structure conditionnelle ternaire

La structure conditionnelle ternaire en PHP est une manière concise d'écrire une condition avec une seule instruction. Elle permet d'économiser de l'espace et d'améliorer la lisibilité du code.

La syntaxe générale de la structure ternaire est la suivante:
`$variable = (condition) ? valeur_si_vrai : valeur_si_faux`
Exemple

```php
<?php
$age = 25;
$resultat = ($age >= 18) ? "Majeur" : "Mineur";
echo $resultat;
?>
```

Dans cet exemple, si la condition `$age >= 18` est vraie, la variable `$resultat` sera affectée à la valeur "Majeur". Sinon, elle sera affectée à la valeur "Mineur". Le résultat affiché sera donc "Majeur".

La structure ternaire est souvent utilisée lorsque la condition et les valeurs attribuées sont simples et ne nécessitent pas de blocs de code plus complexes. Cependant, il est important de l'utiliser avec modération pour préserver la lisibilité du code. Si la condition et les valeurs sont plus complexes, il est préférable d'utiliser une structure conditionnelle if-else traditionnelle.

- La structure de controle Switch

Le switch est une structure de contrôle en PHP qui permet d'effectuer une comparaison de la valeur d'une expression avec plusieurs cas possibles et d'exécuter le bloc de code correspondant au cas trouvé. C'est une alternative pratique lorsque vous devez évaluer une seule expression avec plusieurs valeurs différentes.
a syntaxe générale du switch en PHP :

```php
switch (expression) {
    case valeur1:
        // Code à exécuter si expression correspond à valeur1
        break;
    case valeur2:
        // Code à exécuter si expression correspond à valeur2
        break;
    case valeur3:
        // Code à exécuter si expression correspond à valeur3
        break;
    // Ajouter autant de cas que nécessaire
    default:
        // Code à exécuter si aucune des valeurs précédentes ne correspond à
expression
        break;
}
```

L'expression est évaluée et comparée aux différentes valeurs spécifiées dans chaque cas.
Si une correspondance est trouvée, le bloc de code correspondant à ce cas est exécuté.
La directive break est utilisée pour sortir du switch une fois qu'un cas est exécuté. Cela empêche l'exécution des cas suivants.
Si aucun des cas ne correspond à l'expression, le bloc de code du cas par défaut (default) est exécuté, à moins qu'il n'y ait pas de cas par défaut spécifié.

Exemple

```php
<?php
$choix = 'b';
switch ($choix) {
    case 'a':
        echo "Vous avez choisi l'option A";
        break;
    case 'b':
        echo "Vous avez choisi l'option B";
        break;
    case 'c':
        echo "Vous avez choisi l'option C";
        break;
    default:
        echo "Choix invalide";
        break;
}
?>
```

Dans cet exemple, la variable `$choix` est comparée à chaque cas possible. Si `$choix` a la valeur 'b', le bloc de code correspondant à ce cas sera exécuté et affichera "Vous avez choisi l'option B". Si la valeur de `$choix` ne correspond à aucun des cas, le bloc de code du cas par défaut sera exécuté et affichera "Choix invalide".

Les Boucles

Les boucles en programmation servent à exécuter un bloc de code de manière répétée jusqu'à ce qu'une condition spécifique soit remplie. Elles permettent d'automatiser des tâches qui nécessitent de répéter des instructions plusieurs fois, en évitant ainsi la répétition manuelle du code.
Les boucles sont utilisées dans de nombreux cas de figure, tels que :
Parcourir des tableaux ou des listes : Les boucles permettent de parcourir les éléments d'un tableau ou d'une liste et de traiter chaque élément individuellement.
Traiter des ensembles de données : Les boucles permettent de manipuler des ensembles de données en répétant les mêmes opérations sur chaque élément.
Valider des entrées utilisateur : Les boucles sont utilisées pour demander à l'utilisateur de fournir des informations jusqu'à ce que les données saisies soient valides.
Générer des séquences ou des schémas : Les boucles sont utilisées pour générer des séquences de nombres, des schémas ou des motifs en répétant des instructions spécifiques.

Effectuer des opérations mathématiques itératives : Les boucles sont utilisées pour effectuer des calculs mathématiques complexes en itérant sur des valeurs intermédiaires.
Les boucles offrent une grande flexibilité et permettent d'optimiser le code en évitant la duplication inutile. Elles permettent également de résoudre des problèmes de manière efficace en automatisant les processus répétitifs.

En PHP, il existe plusieurs types de boucles pour répéter des blocs de code de manière itérative. Les boucles les plus couramment utilisées sont la boucle for, la boucle while et la boucle foreach.

- Boucle for :

La boucle for est utilisée lorsque vous connaissez à l'avance le nombre d'itérations à effectuer.
Syntaxe
`for (initialisation; condition; incrémentation) { // Code à exécuter à chaque itération }`

Exemple

```php
<?php
for ($i = 1; $i <= 5; $i++) {
    echo $i . " ";
}
// Affiche : 1 2 3 4 5
?>
```

Dans cet exemple, la boucle for s'exécute cinq fois (de 1 à 5). À chaque itération, la valeur de `$i` est affichée.

- Boucle while :
  La boucle while est utilisée lorsque vous ne connaissez pas à l'avance le nombre d'itérations à effectuer, mais vous avez une condition de sortie.
  Syntaxe :
  `while (condition) { // Code à exécuter tant que la condition est vraie }`

Exemple

```php
<?php
$i = 1;
while ($i <= 5) {
    echo $i . " ";
    $i++;
}
// Affiche : 1 2 3 4 5
?>
```

Dans cet exemple, la boucle while s'exécute tant que la valeur de `$i` est inférieure ou égale à 5. À chaque itération, la valeur de `$i` est affichée et incrémentée.

- Boucle do while
  En PHP, la boucle "do...while" est utilisée pour exécuter un bloc de code tant qu une condition est vrai
  Syntaxe:

```php
do {
    // Bloc de code à exécuter
} while (condition);
```

Break et Continue

Les instructions `break` et `continue` sont couramment utilisées avec les structures conditionnelles et répétitives en PHP pour contrôler le flux d'exécution du code. Voici comment elles fonctionnent :

1. **Instruction `break` :**
    - L'instruction `break` est utilisée pour sortir prématurément d'une boucle.
    - Lorsque `break` est rencontré à l'intérieur d'une boucle (`for`, `while`, `do...while`, `foreach`, etc.), l'exécution de la boucle est immédiatement interrompue, et le programme continue à exécuter les instructions après la boucle.
    - `break` est généralement utilisé pour sortir de la boucle lorsque certaines conditions sont remplies.
    - exemple :

```php
for ($i = 1; $i <= 10; $i++) {
    if ($i === 5) {
        break; // Sort de la boucle lorsque $i atteint 5
    }
    echo $i . " ";
}
```

Dans cet exemple, la boucle `for` est interrompue lorsque `$i` atteint 5, et le programme continue avec les instructions après la boucle.

2. **Instruction `continue` :**
    - L'instruction `continue` est utilisée pour sauter une itération de la boucle en cours et passer à l'itération suivante.
    - Lorsque `continue` est rencontré à l'intérieur d'une boucle, les instructions suivantes dans la boucle sont ignorées, et la boucle passe à l'itération suivante.
    - `continue` est souvent utilisé pour éviter l'exécution de certaines instructions dans une boucle en fonction de certaines conditions.
    - exemple :

```php
for ($i = 1; $i <= 5; $i++) {
    if ($i === 3) {
        continue; // Passe à l'itération suivante lorsque $i est égal à 3
    }
    echo $i . " ";
}
```

Dans cet exemple, lorsque `$i` est égal à 3, l'itération actuelle est sautée, et la boucle continue avec l'itération suivante.

Les instructions `break` et `continue` sont des outils utiles pour personnaliser le comportement des boucles en PHP en fonction de conditions spécifiques. Elles vous permettent de contrôler efficacement le flux d'exécution de votre code.

-Le bloc de code à l'intérieur des accolades est exécuté en premier, puis la condition est évaluée.
-Si la condition est vraie (c'est-à-dire qu'elle renvoie true), le bloc de code est réexécuté.
-Cette étape se répète tant que la condition est vraie.
-Une fois que la condition devient fausse (c'est-à-dire qu'elle renvoie false), la boucle se termine, et le programme continue avec le code suivant après la boucle "do...while".

- Boucle foreach :
  La boucle foreach en PHP est utilisée pour parcourir les éléments d'un tableau ou d'un objet de manière itérative. Elle permet d'accéder à chaque élément du tableau ou de l'objet sans avoir à manipuler des indices ou des compteurs.
  Syntaxe :
  `foreach ($array as $value) { // Code à exécuter pour chaque élément du tableau }`

`$array` : C'est le tableau ou l'objet que vous souhaitez parcourir. Il peut s'agir d'un tableau indexé, d'un tableau associatif ou d'un objet.
`$value` : C'est une variable temporaire qui représente chaque élément du tableau ou de l'objet à chaque itération de la boucle. Vous pouvez donner un nom à cette variable selon votre choix.
Pendant l'exécution de la boucle, la variable `$value` prend successivement la valeur de chaque élément du tableau ou de l'objet. Vous pouvez utiliser cette variable à l'intérieur de la boucle pour effectuer des opérations ou accéder aux propriétés de l'élément.
Exemple avec un tableau

```php
<?php
$fruits = array("pomme", "banane", "orange");
foreach ($fruits as $fruit) {
    echo $fruit . " ";
}
// Affiche : pomme banane orange
?>
```

Dans cet exemple, la boucle foreach itère sur chaque élément du tableau $fruits et affiche la valeur de chaque élément.

Exemple avec un tableau associatif :

```php
<?php
$personne = array("nom" => "Dupont", "prenom" => "Jean", "age" => 30);
foreach ($personne as $key => $value) {
    echo $key . ": " . $value . "<br>";
```

```
}
// Affiche :
// nom: Dupont
// prenom: Jean
// age: 30
?>
```

Dans cet exemple, la variable `$key` représente les clés du tableau associatif et la variable `$value` représente les valeurs correspondantes.

La boucle foreach est très pratique pour parcourir et manipuler les éléments d'un tableau ou d'un objet de manière simple et intuitive, sans avoir à gérer les indices ou les compteurs manuellement.

Exercices

1. Calcul de la factorielle
   Écrivez un programme PHP qui calcule la factorielle d'un nombre donné. Demandez à l'utilisateur de saisir un nombre entier positif, puis utilisez une boucle for pour calculer la factorielle de ce nombre. Affichez le résultat à l'utilisateur.

2. Vérification de nombre premier
   Écrivez un programme PHP qui vérifie si un nombre donné est premier ou non. Demandez à l'utilisateur de saisir un nombre entier positif, puis utilisez une boucle for pour vérifier si le nombre est divisible par d'autres nombres que 1 et lui-même. Affichez un message indiquant si le nombre est premier ou non.

3. Table de multiplication
   Écrivez un programme PHP qui affiche la table de multiplication d'un nombre donné. Demandez à l'utilisateur de saisir un nombre entier positif, puis utilisez une boucle for pour afficher la table de multiplication de ce nombre de 1 à 10.

4. Calcul de la somme des chiffres
   Écrivez un programme PHP qui calcule la somme des chiffres d'un nombre donné. Demandez à l'utilisateur de saisir un nombre entier positif, puis utilisez une boucle while pour obtenir la somme des chiffres de ce nombre. Affichez le résultat à l'utilisateur.

---

5. Génération de la suite de Fibonacci
   Écrivez un programme PHP qui génère les n premiers termes de la suite de Fibonacci. Demandez à l'utilisateur de saisir un nombre entier positif n, puis utilisez une boucle for pour calculer et afficher les n premiers termes de la suite de Fibonacci.La suite de Fibonacci est une séquence de nombres dans laquelle chaque nombre est la somme des deux nombres qui le précèdent. Elle commence généralement par les nombres 0 et 1.Voici les premiers termes de la suite de Fibonacci : 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, ...

Conclusion

En comprenant et en utilisant les structures en PHP, vous pouvez contrôler l'exécution de votre code de manière flexible et efficace. Les structures conditionnelles vous permettent de prendre des décisions basées

sur des conditions, tandis que les boucles vous permettent de répéter des actions jusqu'à ce qu'une condition soit remplie. En combinant ces structures, vous pouvez créer des programmes PHP puissants et dynamiques.