Structures conditionnelles et itératives

Technologies web

Chris Chevalier - chevalier@chris-freelance.com

Les opérateurs

Structures conditionnelles et itératives

Chris Chevalier - chevalier@chris-freelance.com

Les opérateurs PHP (1/2)

- Il existe **12 types d'opérateurs PHP** :
  - Les **opérateurs arithmétiques** (`+`, `-`, `*`, `**`, `/` et `%`)
  - Les **opérateurs d'incrémentation et de décrémentation** (`++` et `--`)
  - L'**opérateur d'assignation** (`=`)
  - Les **opérateurs de chaine de caractères** (`.` et `.=`)
  - Les **opérateurs de comparaison** (`==`, `===`, `!=`, `!==`, `<>`, `<`, `<=`, `>` et `>=`)
  - Les **opérateurs logiques** (`&&`, `AND`, `||`, `OR`, `XOR` et `!`)
  - Les **opérateurs binaires** (`&`, `|`, `^`, `~`, `<<` et `>>`)
  - Les **opérateurs combinés** (`+=`, `-=`, `*=`, `/=`, `%=`, `&=`, `|=`, `^=`, `<<=` et `>>=`)
  - L'**opérateur de contrôle d'erreur** (`@`)
  - L'**opérateur d'exécution** de commande **Shell** (`` `` ``)

Les opérateurs PHP (2/2)

- Les **opérateurs sur les tableaux** (`+`, `==`, `===`, `!=`, `!==` et `<>`)
- L'**opérateur de type d'objet** (`instanceof`)

Les conditions

Structures conditionnelles et itératives

Les conditions

- Comme son nom l'indique, une **condition** permet d'effectuer une série d'actions seulement si la ou les **conditions** testées sont **vérifiées**
- Ainsi, une **structure conditionnelle** permet d'exécuter ou non un certain nombre d'**instructions** en fonction d'une **condition**
- Si le résultat de cette **condition** est égal à la valeur `true`, alors, le **bloc d'instructions** est **exécuté**
- Ces **expressions** évaluées sont généralement constituées d'**opérateurs de comparaison**, d'**opérateurs logiques** ou encore de **fonctions** renvoyant une valeur **booléenne**

Les conditions PHP - `if()`

- **PHP** propose **4 types de constructions conditionnelles** - `if()`, `elseif()`, `else()` et `switch()`

- L'instruction `if()` se retrouve dans tous les langages de programmation et se charge d'exécuter un **bloc d'instructions** seulement si **l'expression est vérifiée**

```php
<?php
    // Déclaration de la variable '$vitesse'
    $vitesse = 60;

    // Test de la valeur de la variable '$vitesse'
    if($vitesse > 50) {
        echo "Vous êtes en excès de vitesse !";
    }
?>
```

Les conditions PHP - `else()`

- La **clause** `else()` ne doit apparaître qu'**après la fermeture de l'accolade** d'un bloc `if()`
- Elle permet de définir une **suite d'instructions** à exécuter dans le cas où l'**expression testée** par le `if()` est **fausse**

```php
<?php
    // Déclaration de la variable '$vitesse'
    $vitesse = 60;

    // Test de la valeur de la variable '$vitesse'
    if($vitesse > 50) {
        echo "Vous êtes en excès de vitesse !";
    } else {
        echo "Vous n'êtes pas en excés de vitesse";
    }
?>
```

Les conditions PHP - L'opérateur ternaire

- L'**opérateur ternaire** permet de réaliser les mêmes traitements que la structure conditionnelle `if` / `else` mais de manière plus concise

- Voici utilisateur **l'opérateur ternaire** du code disponible sur la diapositive précédente :

```
10  <?php
11      $vitesse = 60;
12
13      echo $vitesse > 50 ? "Vous êtes en excés de vitesse !" : "Vous n'êtes pas en excés de vitesse";
14  ?>
```

- Ainsi, **l'expression** `(expr1) ? (expr2) : (expr3)` exécute `expr2` si `expr1` est évaluée à `true` et `expr3` dans le cas contraire

Les conditions PHP - `elseif()` (1/2)

- La **clause** `elseif()` se place également **après l'accolade fermante** d'un **bloc** `if()` - Il est possible d'en cumuler **plusieurs**

- Elle permet d'éviter une imbrication de nombreux **blocs** `if()`

Les conditions PHP - `elseif()` (2/2)

```php
<?php
    // Déclaration de la variable '$vitesse'
    $vitesse = 74;

    // Test de la valeur de la variable '$vitesse'
    if($vitesse > 50 && $vitesse < 70) {
        echo "Vous perdez 1 point pour excès de vitesse";
    } else if ($vitesse > 70 && $vitesse < 90) {
        // Ce message s'affiche
        echo "Vous perdez 2 points pour excès de vitesse";
    } else if ($vitesse > 90) {
        echo "Vous perdez 3 points pour excès de vitesse";
    } else {
        echo "Vous n'êtes pas en excés de vitesse";
    }
?>
```

Les conditions PHP - `switch()` (1/2)

- La **clause** `switch()` offre une alternative à la **structure** `if()` / `elseif()` / `else()`
- Elle permet de tester toutes les valeurs possibles que peut prendre une **variable**

Les conditions PHP - `switch()` (2/2)

```php
<?php
    // Déclaration de la variable '$personnage'
    $personnage = "mage";

    // Test de la valeur de la variable '$personnage'
    switch ($personnage) {
        case 'guerrier':
            echo "Votre personnage est un guerrier";
            break;
        case 'voleur':
            echo "Votre personnage est un voleur";
            break;
        case 'mage':
            // Ce message s'affiche
            echo "Votre personnage est un mage";
            break;
        default:
            echo "Veuillez choisir un personnage disponible";
            break;
    }
?>
```

Les boucles

Structures conditionnelles et itératives

Les boucles

- Une **boucle** permet de **répéter** `n` fois une ou plusieurs actions en quelques millisecondes
- Il s'agit d'une **structure de contrôle** capable d'exécuter un certain nombre de fois une **suite d'instructions** en fonction d'une ou plusieurs **conditions à valider**
- Pour éviter les **boucles infinies** qui feront planter vos programmes, il faudra toujours s'assurer que la ou les **conditions** passent à *false* à un moment donné afin de **sortir de la boucle**
- **PHP** propose **3 types de boucles** - `for()`, `while()` et `foreach()`

Les boucles PHP - `for()`

- Présente dans la majorité des langages de programmation, la **boucle** `for()` n'est pas forcément la plus utilisée

- Dans la plupart des langages de programmation, elle permet le **parcours de tableaux** - Or, **PHP** dispose déjà d'une **boucle dédiée au parcours de tableaux** (`foreach()`)

- La particularité de la **boucle** `for()` vient du fait qu'il faut connaître en avance sa **condition d'arrêt** (la valeur qui rend la condition **fausse** et stoppe la **boucle**)

```php
<?php
    // Boucle générant la table de 9
    for ($i = 0; $i <= 10; $i++) {
        echo "9 x {$i} = " . 9 * $i . "<br>";
    }
?>
```

9 x 0 = 0
9 x 1 = 9
9 x 2 = 18
9 x 3 = 27
9 x 4 = 36
9 x 5 = 45
9 x 6 = 54
9 x 7 = 63
9 x 8 = 72
9 x 9 = 81
9 x 10 = 90

Les boucles PHP - `while()`

- Traduisible par « **tant que** », la **boucle** `while()` permet de répéter le bloc d'instructions **tant que** la **condition** passée en **argument** est **vraie**

- Bien entendu, le programme **sort de la boucle** lorsque cette dernière devient **fausse**

- Avec cette **structure de contrôle**, la condition n'est testée qu'après l'exécution complète du **bloc d'instructions** - Ainsi, même si la **condition** devient **fausse** en cours de lecture, le reste du **bloc** est exécuté

Les boucles PHP - `while()`

```php
<?php
    // Déclaration du compteur '$i'
    $i = 0;

    while ($i <= 10) {
        echo "9 x {$i} = " . 9 * $i . "<br>";
        // Incrémentation du compteur
        $i++;
    }
?>
```

9 x 0 = 0
9 x 1 = 9
9 x 2 = 18
9 x 3 = 27
9 x 4 = 36
9 x 5 = 45
9 x 6 = 54
9 x 7 = 63
9 x 8 = 72
9 x 9 = 81
9 x 10 = 90

Les boucles PHP - `foreach()`

- La **structure de contrôle** `foreach()` permet de **parcourir le contenu d'un tableau** simplement

- Il s'agit d'une **boucle** particulière qui avance le **pointeur** du **tableau** à chaque **itération** - Elle a été intégré depuis la **version 4** du langage

Les boucles PHP - `foreach()`

```php
<?php
    // Déclaration du tableau à index numériques '$legumes'
    $legumes = ["salade", "oignon", "piment", "carotte"];
    // Déclaration du tableau associatif '$couleurs'
    $couleurs = ["rouge" => "#ff0000", "vert" => "#00ff00", "bleu" => "#0000ff"];

    foreach ($legumes as $legume) {
        echo "{$legume} <br>";
    }

    foreach ($couleurs as $couleur => $codeHexa) {
        echo "{$couleur} : {$codeHexa} <br>";
    }
?>
```

salade
oignon
piment
carotte
rouge : #ff0000
vert : #00ff00
bleu : #0000ff

Les boucles PHP - L'instruction de continuité

- **PHP** propose également **2 instructions** propres aux **boucles** :
  - L'**instruction de continuité** `continue`
  - L'**instruction d'arrêt** `break`

- L'instruction de continuité `continue` force le passage à l'**itération suivante** en sautant tout ou partie du **bloc d'instructions**

```php
<?php
    for ($i = 0; $i <= 10; $i++) {
        // On n'affiche pas les 5 premiers chiffres
        if ($i <= 5) {
            continue;
        }
        echo "{$i} <br>";
    }
?>
```

6
7
8
9
10

Chris Chevalier - chevalier@chris-freelance.com

Les boucles PHP - L'instruction d'arrêt

- L'**instruction d'arrêt** `break` force le programme à **quitter** une **structure conditionnelle** `for()`, `while()`, `foreach()` ou `switch()`

```php
<?php
    for ($i = 0; $i <= 10; $i++) {
        // On sort de la boucle si la valeur de '$i' est supérieure à 5
        if ($i > 5) {
            break;
        }
        echo "{$i} <br>";
    }
?>
```

```
0
1
2
3
4
5
```

TP1 - Les comparaisons et les conditions

Structures conditionnelles et itératives

1 - Comparaison

- Dans la section `<?php ?>` d'un fichier `tp1.php`, créez une **variable** `$test` de valeur `143` et une **variable** `$bis` de valeur `219`
- Utilisez tous les **opérateurs de comparaison** avec ces **variables** et affichez les résultats
- 💡 `$test > $bis; // affiche false`

2 - Condition

- Créez une **variable** `$limit` de valeur `50` et une **variable** `$score` de valeur `64`
- Créez une **condition** : si le score est supérieur ou égal à la limite, affichez `"Ok good !"`, sinon affichez `"Oh nooo…"`
- Changez la valeur de `$score` pour changer le résultat

3 - Condition II

- Créez une **variable** `$password` de valeur `“azerty“`
- À l'aide d'une **condition**, affichez `"The password is secure"` si `$password` a une longueur plus grande que `5`

4 - Condition III

- Combinez les deux **conditions** précédentes (exercices 2 et 3)
- Affichez `"Everything is good"` si les deux **conditions** sont vraies
- Affichez `"Something is good"` si une des deux **conditions** est vraie
- Affichez `"Nothing is good"` si aucune des deux **conditions** n'est vraie

5 - Boucle

- Créez une **variable** `$total` de valeur `0` et une **variable** `limit` de valeur `10`
- À l'aide d'une **boucle** `for` allant de `0` à `$limit`, augmentez la valeur de `$total` en lui ajoutant la valeur de `$i` (`$i` est l'indice de votre **boucle**)
- Affichez la valeur de `$total` après votre **boucle**, `$total` doit valoir `55 = 1 + 2 + 3 + 4 + 5 + 6 + 7 + 8 + 9 + 10`

6 - Reverse

- Créez une **variable** `$sentence` valant `“Hello Hexagone !“`
- A l'aide d'une **boucle** `for` et **sans** utilisez la **fonction** `strrev()`, faites en sorte d'afficher l'inverse de votre **variable** `$sentence` (`“! enogaxeH olleH“`)

TP2 - Mise en pratique

Structures conditionnelles et itératives

TP - Mise en pratique (1/2)

- Dans un nouveau fichier **PHP** `even.php`, déclarez une **boucle** parcourant les valeurs entières comprises entre `0` et `10` **inclus** (Cf. diapo 16)
- Au sein de la **boucle**, affichez uniquement les **valeurs paires** - Un nombre est pair **si** son **modulo** de `2` est égal à `0` (`$n % 2 == 0`)
- Dans la cas contraire (`else`), affichez une virgule afin de séparer proprement les éléments affichés
- Ajoutez une instruction permettant d'arrêter la **boucle** si la valeur parcourue est égale à `9` (Cf. diapo 22)

TP - Mise en pratique (2/2)

- Voici une proposition d'affichage final attendu pour valider le TP :

Les nombres paires compris entre 0 et 10 : 0, 2, 4, 6, 8

⭐ TPs Bonus

Structures conditionnelles et itératives

⭐ - Bonus I (1/3)

- Créez une **variable** `$random` qui contient une valeur **aléatoire** entre `1` et `6` (nombres **entiers** seulement)

- À l'aide d'une **condition**, affichez `"Yes I win !"` si `$random` est égal à `6`, et `"So close..."` dans tous les autres cas

⭐ - Bonus I (2/3)

- Créez une **variable** `$month` de valeur `“January“`
- À l'aide d’un `switch`, affichez `"Winter"`, `"Spring"`, `"Summer"` ou `"Fall"` selon la valeur du mois
- Changez la valeur de `$month` pour changer le résultat

⭐ - Bonus I (3/3)

- Créez une **variable** `$number` de valeur `3.6`
- En utilisant uniquement `floor()`, `ceil()`, des **opérations mathématiques** et un `if () {} else {}` reproduisez le résultat de `round()`
- Changez la valeur de `$number` (avec `3.3`, `3.8` et `12.4`) pour changer le résultat

⭐ - Bonus II (1/2)

- Créez une boucle qui part de `0` et s'arrête à `100`

- À chaque itération :
  - Si `$i` est un multiple de `3` ⇒ affichez `“fizz“`
  - Si `$i` est un multiple de `5` ⇒ affichez `“buzz“`
  - Si `$i` est un multiple de `3` et `5` ⇒ affichez `“fizzbuzz“`
  - Si `$i` est un multiple de `7` ⇒ ne l'affichez pas
  - Sinon, affichez la valeur de `$i`

- 💡 Utilisez l’opérateur **modulo** `%` pour savoir si un nombre est le multiple d’un autre

⭐ - Bonus II (2/2)

- Reproduisez l'exercice 5 avec une **boucle** `while`

⭐ - Bonus III

- Créez un **tableau** vide
- Avec une **boucle** `for`, ajoutez `20` entiers aléatoires compris entre `0` et `100`
- Affichez ce **tableau**
- Avec une seconde **boucle** `for`, trouvez l'entier le plus grand de votre **tableau** (pas le droit d'utiliser la fonction `max()`)

⭐ - Bonus IV

- Écrivez un programme pour afficher des nombres de 10 à 1 en utilisant une **fonction récursive** (une fonction qui s'appelle elle-même)

- Exemple : `10, 9, 8, 7, 6, 5, 4, 3, 2, 1`

```php
<?php

    function decrement($n)
    {
        // Écrivez votre code ici
    }

    decrement(10);

```

⭐ Bonus V

- Écrivez un programme PHP pour supprimer les **doublons** d'un tableau
- Exemple : `[1, 2, 2, 3, 3, 3, 4, 5, 5]`
- Sortie prévue : `[1, 2, 3, 4, 5]`

⭐ Bonus VI

- Écrivez un programme pour calculer la **factorielle** d'un nombre en utilisant la boucle `for` en PHP
- Sortie prévue : `La factorielle de 3 est 6`
- Écrivez un programme PHP pour trouver la **factorielle** d'un nombre en utilisant une **fonction récursive**
- Sortie prévue : `La factorielle de 3 est 6`

⭐ Bonus VII

- Écrivez un programme pour afficher le triangle d'étoile suivant en utilisant une boucle `for` :

```
*
**
***
****
*****
******
*******
********
*********
**********
```

⭐ Bonus VIII

- Écrivez un programme PHP pour afficher la table de multiplication jusqu'à 5 * 5 :

1	2	3	4	5
2	4	6	8	10
3	6	9	12	15
4	8	12	16	20
5	10	15	20	25

Des questions ?