Module *: PHP-MYSQL* ***Chargé du module*** **:** M. KONATE

Chapitre 1 : PHP (suite)

I. Les structures de contrôle

Très souvent, lorsque vous écrivez du code, vous souhaitez effectuer différentes actions pour différentes conditions. Pour ce faire, vous pouvez utiliser des instructions conditionnelles dans votre code.

En PHP, nous avons les instructions conditionnelles suivantes:

- Structure **if** : exécute du code si une condition est vraie.
- Structure **if...else** : exécute du code si une condition est vraie et un autre code si cette condition est fausse.
- Structure if...**elseif...else** : exécute des codes différents pour plus de deux conditions
- Structure **switch** : sélectionne l'un des nombreux blocs de code à exécuter

1. Structure if

L'instruction **if** exécute du code si une condition est vraie.

Syntaxe

```
if (condition) {
  Code à exécuter si la condition est Vrai
}
```

Exemple :

```
<?php
        $num=12;
        if($num < 100){
                echo ( ''$num est inférieur à 100 '');
        }
?>
```

2. Structure if...else

L'instruction **if...else** exécute du code si une condition est vraie et un autre code si cette condition est false.

Syntaxe

```
if (condition) {
  code à exécuter si la condition est vraie;
}
else {
  code à exécuter si la condition est faux;
}
```

Exemple

```
<?php
        $num=12;
        if($num%2 = = 0){
                echo (" $num est un nombre pair ");
        }else{
                echo ("$num est un nombre impair ");
        }
?>
```

3. Structure if...elseif...else

L'instruction **if...elseif...else** exécute des codes différents pour plus de deux conditions.

Syntaxe

```
if (condition1) {
 code à exécuter si la condition1 est vraie;
} elseif (condition2) {
 code si la condition2 est vraie;
}

else {
   code à exécuter à exécuter si toutes les conditions données sont fausses ;
}
```

4. Structure switch

L'instruction switch est utilisée pour exécuter une instruction à partir de plusieurs conditions.

Syntaxe

```
switch(expression){
    case value1:
         //code à exécuter
     break;
    case value2:
        //code à exécuter
    break;
    .....
    default:
    code à exécuter si tous les cas ne sont pas correspondants;
}
```

Exemple :

```
<?php
    $num=20;
    switch($num){
        case 10:
            echo("le nombre est égal à 10 ");
        break;
        case 20:
            echo("le nombre est égal à 20 ");
        break;
        case 30:
            echo("le nombre est égal à 30 ");
        break;
    default:
        echo("le nombre n'est pas égal à 10, 20 ou 30 ");
    }
?>
```

5. L'operateur ternaire

Il existe également un opérateur capable d'examiner une condition et retourner une valeur parmi deux en fonction du résultat (vrai ou faux).

Syntaxe

< ? php (condition) ? resultatVrai : resultatFaux ?>

Exemple :

```
< ?php

$noteExamen  = 11 ;
$resultatExamen = ($noteExamen >= 10 ) ?  "admis"    :   "recalé" ;
echo ("Résultat de votre examen :  $resultatExamen") ;
?>
```

Remarque : l'utilisation de l'opérateur ternaire **? :** permet de réduire le nombre de déclaration **if …. else**.

II. Les itérations

Souvent, lorsque vous écrivez du code, vous voulez que le même bloc de code s'exécute encore et encore un certain nombre de fois. Ainsi, au lieu d'ajouter plusieurs lignes de code presque égales dans un script, nous pouvons utiliser des boucles.

Les boucles sont utilisées pour exécuter le même bloc de code encore et encore, tant qu'une certaine condition est vraie.

1. While

Effectue une boucle à travers un bloc de code tant que la condition spécifiée est vraie.

Syntaxe

```
while (condition) {
   code à exécuter;
}
```

Exemple :

```
<?php
$x = 1;
while($x <= 5) {
        echo "Le nombre est: $x <br>";
         $x++;
 }
?>
```

Cet exemple affiche les nombres de 1 à 5.

2. Do-While

La boucle Do-While parcourt une fois un bloc de code, puis répète la boucle tant que la condition spécifiée est vraie.

Syntaxe

```
do{
  Code à exécuter
} while (condition);
```

Exemple :

```
<?php
  $x = 1;

  do {
  echo "Le nombre est: $x <br>";
     $x++;
 } while ($x <= 5);
?>
```

Remarque :

La principale différence entre les deux boucles est que la boucle while vérifie la condition au début, tandis que la boucle do-while vérifie la condition à la fin de la boucle.

3. For

La boucle **For** est utilisée lorsque vous savez à l'avance combien de fois le script doit s'exécuter.

Syntaxe

```
for(initialisation; condition; increment/decrement){
        Code à exécuter
```

}

Exemple :

```
<?php
  for ($x = 0; $x <= 10; $x++) {
    echo "Le nombre est: $x <br>";
 }
?>
```

L'exemple ci-dessous affiche les nombres compris entre 0 et 10.

4. Foreach

La boucle **foreach** est utilisée pour parcourir les éléments du tableau. Il ne fonctionne que sur le tableau et l'objet.

Syntaxe

```
foreach ($array as $value) {
 Code à exécuter
}
```

Ou

```
foreach ($array as $key => $element) {
        Code à exécuter
}
```

Exemple :

```
<?php
  $colors = array("red", "green", "blue", "yellow");
  foreach ($colors as $value) {
    echo "$value <br>";
 }
?>
```

5. Break

Vous avez déjà vu l'instruction **switch break**. Il a été utilisé pour sauter d'une déclaration.

L'instruction peut également être utilisée pour sortir d'une boucle.

Exemple :

```
<?php
for ($x = 0; $x < 10; $x++) {
  if ($x == 4) {
    break;
  }
  echo "Le nombre is: $x <br>";
 }
?>
```

Cet exemple saute hors de la boucle lorsque **x** est égal à **4**.

6. Continue

L'instruction **continue** interrompt une itération (dans la boucle) si une condition spécifiée se produit, et continue avec l'itération suivante dans la boucle.

Exemple :

```php
<?php
for ($x = 0; $x < 10; $x++) {
  if ($x == 4) {
    continue;
  }
  echo "Le nombre est: $x <br>";
 }
?>
```

Cet exemple ignore la valeur 4**.**

Fin