PHP : Les instructions de contrôle

L3 Informatique - UE Développement Web

David Lesaint

david.lesaint@univ-angers.fr

Janvier 2020

Les instructions de contrôle

Instructions de contrôle des scripts

Syntaxe et sémantique proches du C/C++.

Trois types

- Instructions conditionnelles :
  - `if` et `if...else`
  - `?` et `??`
  - `switch...case`
- Instructions de boucle :
  - `for`
  - `while` et `do...while`
  - `foreach`
  - `break`, `continue` , ~~`goto`~~
- Gestion des erreurs :
  - `error_reporting`
  - `try...catch...finally`

L'instruction `if`

`if(exp) instruction;`

`instruction` est exécutée si `exp` est évaluée à `TRUE`.

`if(exp) {bloc}`

Le `bloc` d'instructions est exécuté si `exp` est évaluée à `TRUE`.

Différentes types d'`exp`ressions possibles :

- Booléenne (`$x>2`) ou non (`$y` avec `$y="abc"`).
- Atomique (`$x>2`) ou composite (`$x>2 || $y`).

***Confer* cours 2**

- Tableau des règles d'évaluation booléenne d'expression.
- Liste des opérateurs logiques de comparaison et composition.

L'instruction `if...else`

`if(exp) ins1; else ins2;`

Exécute `ins1` si `exp` est évaluée à `TRUE` ou `ins2` sinon.

`if(exp) {bloc1} else {bloc2}`

Exécute `bloc1` si `exp` est évaluée à `TRUE` ou `bloc2` sinon.

exemple3-1.php

```
1  $prix=55;
2  if ($prix>100)
3  {
4   echo "<b>Pour un montant d'achat de $prix &#8364;, la remise est de 10 %
</b><br>";
5   echo "Le prix net est de ",$prix*0.90;
6  }
7  else
8  {
9   echo "<b>Pour un montant d'achat de $prix &#8364;, la remise est de 5
%</b><br>";
10  echo "<h3>Le prix net est de ",$prix*0.95," &#8364;</h3>";
11 }
```

Les `if` imbriqués avec `if...elseif...else`

exemple3-2.php

```php
// ********************if...elseif...else*****************
$cat="PC";
$prix=900;
if ($cat=="PC")
{
 if ($prix>= 1000)
 {
  echo "<b>Pour l'achat d'un PC d'un montant de $prix &#8364;, la remise est de 15 %</b><br>";
  echo "<h3> Le prix net est de : ",$prix*0.85, "&#8364; </h3>";
 }
 else
 {
  echo "<b>Pour l'achat d'un PC d'un montant de $prix &#8364;, la remise est de 10 %</b><br>";
  echo "<h3> Le prix net est de : ",$prix*0.90, "&#8364; </h3>";
 }
}
elseif ($cat=="Livres")
{
 echo "<b>Pour l'achat de livres la remise est de 5 %</b><br />";
 echo "<h3> Le prix net est de : ",$prix*0.95, "&#8364; </h3>";
}
else
{
 echo"<b>Pour les autres achats la remise est de 2 %</b><br>";
 echo "<h3> Le prix net est de : ",$prix*0.98, "&#8364; </h3>";
}

```

L'opérateur ternaire ?

`exp ? val1 : val2`

Renvoie `val1` si `exp` est évaluée à `TRUE` ou `val2` sinon.

Usage : `$var = exp ? val1 : val2;`

Equivaut à : `if(exp){$var=val1;} else {$var=val2;}`

exemple3-3.php

```
1 $ch = "Bonjour ";
2 $sexe="M";
3 $ch .= ($sexe=="F")?"Madame":"Monsieur";
4 echo "<h2>$ch</h2>";
5 $nb = 3;
6 $pmu ="Il faut trouver ".$nb;
7 $mot = ($nb==1)?" cheval":" chevaux";
8 echo "<h3> $pmu $mot </h3>";
```

L'opérateur binaire *null coalescent* `??` (PHP 7)

`$var ?? exp`

Renvoie `$var` si `$var` initialisée à non `NULL` ou `exp` sinon.

Usage : `$x = $y ?? exp;`

Equivaut à : `$x = isset($y) ? $y : exp;`

operateur-coalescing.php

```php
// 3 instructions équivalentes
$x = $_GET['user'] ?? 'aie';
$x = isset($_GET['user']) ? $_GET['user'] : 'aie';
if(isset($_GET['user'])) $x = $_GET['user']; else $x = 'aie';
```

Chaînage possible

```php
$x = $_GET['user'] ?? $_POST['user'] ?? 'aie';
```

L'instruction `switch...case`

Alternative à `if...elseif...else`

Pour simplifier l'écriture de branchements conditionnels imbriqués.

Syntaxe

```
switch(exp) {
   case val1:
      bloc1;
      break;
   ...
   case valN:
      blocN;
      break;
   default:
      blocD;
      break;
}
```

Sémantique

- Si `exp` vaut `val1`, le `bloc1` est exécuté et l'exécution passe à la fin du bloc `switch`.
- Sinon, la procédure est réitérée sur les valeurs `val2` ... `valN` jusqu'à concordance.
- Si aucune concordance n'est trouvée, le `blocD` est exécuté.

L'instruction `switch...case`

exemple3-4.php

```php
$dept=75;
switch ($dept) {
//Premier cas
      case 75:
      case "Capitale":
            echo "Paris";
            break;
//Deuxième cas
      case 78:
            echo "Hauts de Seine";
            break;
//Troisième cas
      case 93:
      case "Stade de France":
            echo "Seine Saint Denis";
            break;
//la suite des départements...
//Cas par défaut
      default:
            echo "Département inconnu en Ile de France";
            break;
}
```

La boucle `for`

Syntaxe

`for(exp1;exp2;exp3){instruction;}`
ou `for(exp1;exp2;exp3){bloc}`

Sémantique

1. `exp1` est évaluée.
2. `exp2` est évaluée en contexte booléen : si elle vaut `TRUE`, l'`instruction` (ou le `bloc` d'instructions) est exécutée, sinon on sort du bloc `for`.
3. `exp3` est exécutée et la boucle reprend à l'étape (2) jusqà ce que `exp2` soit évaluée à `FALSE`.

exemple3-5.php

```
1 for($i=1;$i<7;$i++) {
2       echo "<h$i> $i :Titre de niveau $i </h$i>";
3 }
```

Boucle à plusieurs variables

Sous-expressions séparées par des virgules

- Multiples initialisations (eg, plusieurs compteurs).
- Multiples conditions.
- Multiples in-/dé-crémentations.

exemple3-6.php

```php
for ($i=1,$j=9;$i<10,$j>0;$i++,$j--) {
//$i varie de 1 à 9 et $j de 9 à 1
      $css="style=\"border-style:double;border-width:3;\"";
      echo "<span ".$css.">$i + $j=10</span>";
}
```

Boucles imbriquées

exemple3-7.php

```
1  echo "<h2>Révisez votre table de multiplication!</ h2>";
2  //Début du tableau HTML
3  echo "<table border=\"2\"
style=\"background-color:yellow\"> <th> X  </th>";
4  //Création de la première ligne
5  for ($i=1;$i<10;$i++)
6    echo "<th> $i </th>";
7  //Création du corps de la table
8  //Boucles de création du contenu de la table
9  for ($i=1;$i<10;$i++) {
10   //Création de la première colonne
11   echo "<tr><th> $i </th>";
12   //Remplissage de la table
13   for ($j=1;$j<10;$j++) {
14     echo "<td style=\"background-color:red;color:white\">
   <b>". $i*$j."   </td>";
15   }
16   echo "</b></tr>";
17 }
18 echo "</table>"
```

La boucle `while`

Alternative à `for`

Quand on ne connaît pas a priori le nombre limite d'itérations.
Exemple : afficher les résultats d'une requête sur une BDD.

`while(exp){instruction;}` ou `while(exp){bloc}`

Tant que `exp` est évaluée à `TRUE`, exécute l'`instruction` (ou le `bloc` d'instructions).

exemple3-8.php

```
1 $n=1;
2 while($n%7!=0)
3 {
4  $n = rand(1,100);
5  echo $n,"  /";
6 }
```

La boucle `do...while`

`do {bloc} while(exp);`

- Similaire à `while` mais `bloc` est au moins exécutée une fois avant d'évaluer `exp`.
- Les variables évaluées dans `exp` peuvent être initialisées dans `bloc`.

exemple3-9.php

```
1 do
2 {
3  $n = rand(1,100);
4  echo $n,"  / ";
5 }
6 while($n%7!=0);
```

La boucle `foreach`

Pour itérer sur les éléments d'un tableau

- Plus efficace que `for`.
- Adaptée aux tableaux associatifs : elle ne nécéssite pas d'en connaître la taille ou les clés.

Deux syntaxes selon que l'on souhaite récupérer

- Les valeurs uniquement.
- Les valeurs et les clés/indices.

`foreach` pour récupérer les valeurs d'un tableau

`foreach($tab as $val){bloc}`

- La variable `$val` contiendra successivement chacune des valeurs du tableau `$tab`.
- La variable `$tab` peut être remplacée par une expression de type `array`.

- Veiller à ne pas utiliser un nom de variable existant pour `$val` (écrasement).
- Les variables utilisées dans la boucle ne sont pas locales et gardent leur dernière valeur en sortie de boucle.

exemple3-10.php

```php
1 //Création du tableau de 9 éléments
2 for($i=0;$i<=8;$i++) { $tab[$i] = pow(2,$i);}
3 $val ="Une valeur";
4 echo $val,"<br />";
5 //Lecture des valeurs du tableau
6 echo"Les puissances de 2 sont :";
7 foreach($tab as $val) { echo $val." : ";}
```

`foreach` pour récupérer indices et valeurs d'un tableau

`foreach($tab as $key=>$val){bloc}`

La paire de variables (`$key`,`$val`) correspondra successivement à chaque élément (indice,valeur) du tableau `$tab`.

exemple3-11.php

```php
//Création du tableau
for($i=0;$i<=8;$i++) {
  $tab[$i] = pow(2,$i);
}
//Lecture des indices et des valeurs
foreach($tab as $ind=>$val) {
  echo " 2 puissance $ind vaut $val <br />";
}
echo "Dernier indice ",$ind, " ,dernière valeur ",$val;
```

`foreach` pour récupérer clés et valeurs d'un tableau

exemple3-12.php

```php
1 //Création d'un tableau associatif
2 for ($i=0;$i<=8;$i++) {
3   $tabass["client".$i] = rand(100,1000);
4 }
5 //Lecture des clés et des valeurs
6 foreach ($tabass as $cle=>$val) {
7   echo " Le client de login <b>$cle</b> a le code
<b>$val</b><br />";
8 }
```

`foreach` pour itérer sur les propriétés d'un objet

Assimilées aux éléments d'un tableau associatif

- Nom de propriété = clé.
- Valeur de propriété = valeur.

Sortie anticipée de boucle avec `break`

`break`

Arrête une boucle `for`, `foreach` ou `while` avant son terme

- N'arrête pas le script contrairement à `exit`.
- Arrête uniquement la boucle qui le contient dans le cas de boucles imbriquées.
- `break n;` arrête les `n` boucles les plus internes l'imbriquant.

Sortie anticipée de boucle avec `break`

exemple3-13.php

```php
1  //Création d'un tableau de noms
2  $tab[1]="Basile"; $tab[2]="Conan";
3  $tab[3]="Albert"; $tab[4]="Vincent";
4  //Boucle de lecture du tableau
5  for($i=1;$i<count($tab);$i++) {
6   if ($tab[$i][0]=="A") {
7    echo "Le premier nom commençant par A est le numéro $i:
",$tab[$i];
8    break;
9   }
10 }
```

- `$tab[$i][n]` ($n \geq 0$) correspond à la `n`+1-ième lettre de la chaîne `$tab[$i]`.
- `count($tab)` évitera une boucle infinie éventuelle si aucun mot de `$tab` ne démarre par `A`.

Avancement de boucle avec `continue`

`continue`

Arrête l'itération en cours, pas la boucle.

exemple3-14.php

```php
//Interruption d'une boucle for
for($i=0;$i<20;$i++)
{
 if($i%5==0) { continue;}
 echo $i,"<br />";
}
//Interruption d'une boucle foreach
$tab[1]="Ain";
$tab[2]="Allier";
$tab[27]="Eure";
$tab[28]="Eure-et-Loir";
$tab[29]="Finistère";
$tab[33]="Gironde";
foreach($tab as $cle=>$valeur)
{
 if($tab[$cle][0]!="E") { continue;}
 echo "code $cle : département ",$tab[$cle],"<br />";
}
```

Avancement de boucle avec `continue`

`continue n;`

Arrête les `n`-1 boucles les plus internes l'imbriquant et l'itération courante de la `n`-ième.

exemple3-15.php

```php
for ($i=0;$i<10;$i++)
{
 for ($j=0;$j<10;$j++)
 {
  for ($k=0;$k<10;$k++)
  {
   if (($i+$j+$k)%3==0) continue 3;
   echo "$i : $j : $k <br /> ";
  }
 }
}
```

Gestion des erreurs

Filtrage des messages d'erreur renvoyés au poste client

- Nomenclature (partielle) des erreurs en PHP :

Constante	Valeur	Niveau d'affichage
`E_ERROR`	1	Erreur fatale (eg. appel de fonction inexistante) : le script s'arrête.
`E_WARNING`	2	Avertissement (eg. division par 0) : le script se poursuit.
`E_PARSE`	4	Erreur de syntaxe : le script s'arrête.
`E_NOTICE`	8	Avis de problème simple.
...		
`E_ALL`	2^{15}	Toute erreur.

- Le type d'erreurs relayées est prédéfini et configurable dans le fichier de configuration `php.ini`
- On peut l'ajuster pour chaque script en y insérant `error_reporting(n);` au début ce qui n'affichera que les erreurs dont les valeurs correspondent aux bits positionnés dans `n`.

Gestion des erreurs

erreurs.php

```php
//error_reporting(E_ERROR); // E_ALL, E_WARNING ...
$x=2;$y=0;
echo $x/$y;
fopen("nofile.txt","r");
jenexistepas();
```

Suppression des messages d'erreur par fonction

- `@f()` : faire précéder l'appel de la fonction `f` par `@`.

Exceptions

Mécanismes d'interception et traitement d'erreurs

Gestion d'exception avec `try...catch...finally` et utilisant la classe prédéfinie `Exception`.

Syntaxe

```
try {
  if(test){
    throw new Exception(m,c);
  } else {
    blocI
}
catch(Exception $e)
{
    blocG
}
finally
{
    bloc
}
```

Sémantique

`try{...}` délimite le bloc dans lequel peut survenir l'erreur :

- `test` est la condition nécessaire à l'erreur.
- `throw` lance un objet `Exception` créé par `new` avec message d'erreur `m` et code d'erreur `c`.
- `blocI` est le bloc exécuté si l'erreur est évitée.

`catch(...){...}` intercepte n'importe quel objet `e` de classe `Exception` lancé dans `try{...}` :

- `blocG` est le bloc exécuté utilisant les méthodes de `e` pour récupérer et afficher des informations sur `e` (`m`, `c`, ...).

`finally{...}` délimite un `bloc` systématiquement exécuté.

La classe `Exception`

Son constructeur prend 2 arguments qui initialise 2 des propriétés de chaque objet créé :

- `message` : le message d'erreur sous forme de chaîne de caractères.
- `code` : le code d'erreur sous forme d'entier.

Méthodes de la classe `Exception`

Méthode	**Définition**
`__construct(m,c)`	Constructeur de l'objet (appelé implicitement avec `new Exception(m,c)`) prenant les deux paramètres `m` et `c`.
`getCode()`	Retourne la valeur de la propriété `message`.
`getMessage()`	Retourne la valeur de la propriété `code`.
`getFile()`	Retourne la valeur de la propriété `file` contenant nom et chemin d'accès du fichier dans lequel s'est produit l'erreur.
`getLine()`	Retourne la valeur de la propriété `line` correspondant au numéro de ligne à laquelle a été créée l'exception.
`__toString()`	Retourne une chaîne contenant toutes les informations sur l'exception.

Exemple

Alternative à l'absence de gestion d'erreur ou à la suppression d'avertissement via `error_reporting`

```php
$a=100;$b=0;
try{
    if($b===0) {
        throw new Exception("Division par 0",7);
    } else {
        echo "R&#233;sultat de : $a / $b = ",$a/$b;
    }
} catch(Exception $e) {
    echo "<hr>Message d'erreur : ",$e->getMessage();
    echo "<hr>Code d'erreur : ",$e->getCode();
    echo "<hr>Nom du fichier :",$e->getFile();
    echo "<hr>Num&#233;ro de ligne :",$e->getLine();
    echo "<hr>__toString : ",$e->__toString();
} finally {
    echo "<hr>L'exception a &#233;t&#233;
g&#233;r&#233;e, le script continue.";
}
echo "<hr>Vraie Fin";
```

Exemple amélioré avec boîte d'alerte

exemple3-18.php

```php
$a=10;
$b=0;
try {
    if($b===0) {
        throw new Exception("Division par 0",7);
    } else {
        echo "Résultat de : $a / $b = ",$a/$b;
    }
} catch(Exception $e) {
    $c = $e->getCode();
    $m = $e->getMessage();
    echo "<script>alert(' Erreur numéro ",$c," \\n ",$m,"
' );</script>";
} finally {
    echo "Tout est sous contrôle<br/>";
}
echo "FIN";
```

Exceptions personnalisées

Par création de classe

Héritant et spécialisant la classe `Exception` avec nouvelles propriétés et méthodes.

exemple3-19.php

```php
class MonException extends Exception {
   public function alerte() {
      $this->message ="<script> alert(' Erreur numéro ".$this->getCode()." \\n
".$this->getMessage() ." ' )</script> ";
      return $this->getMessage();
   }
}
// Utilisation de la classe
$a=100; $b=3;
try {
   if($b == 0) {throw new MonException("Division par 0",7);}
   elseif($a%$b != 0) {throw new MonException("Quotient entier impossible",55);}
   else{echo "Résultat de : $a / $b = ",$a/$b;}
} catch(MonException $except) {
   echo $except->alerte();
} finally {
   echo "Le script continue sans problème<br/>";
}
echo "FIN";
```