

PHP : Les instructions de contrôle

L2 MPCIE - UE Développement Web

David Lesaint

david.lesaint@univ-angers.fr



Janvier 2019

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'expressions 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

```
1 // *****if...elseif...else *****
2 $cat="PC";
3 $prix=900;
4 if ($cat=="PC")
5 {
6     if ($prix>= 1000)
7     {
8         echo "<b>Pour l'achat d'un PC d'un montant de $prix &#8364;, la remise est
de 15 %</b><br>";
9         echo "<h3> Le prix net est de : ", $prix*0.85, "&#8364; </h3>";
10     }
11     else
12     {
13         echo "<b>Pour l'achat d'un PC d'un montant de $prix &#8364;, la remise est
de 10 %</b><br>";
14         echo "<h3> Le prix net est de : ", $prix*0.90, "&#8364; </h3>";
15     }
16 }
17 elseif ($cat=="Livres")
18 {
19     echo "<b>Pour l'achat de livres la remise est de 5 %</b><br />";
20     echo "<h3> Le prix net est de : ", $prix*0.95, "&#8364; </h3>";
21 }
22 else
23 {
24     echo "<b>Pour les autres achats la remise est de 2 %</b><br>";
25     echo "<h3> Le prix net est de : ", $prix*0.98, "&#8364; </h3>";
26 }
27
```

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;`

opérateur-coalescing.php

```
1 // 3 instructions équivalentes
2 $x = $_GET['user'] ?? 'aie';
3 $x = isset($_GET['user']) ? $_GET['user'] : 'aie';
4 if(isset($_GET['user'])) $x = $_GET['user']; else $x = 'aie';
```

Chaînage possible

```
1 $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

```
1 $dept=75;
2 switch ($dept) {
3     //Premier cas
4     case 75:
5     case "Capitale":
6         echo "Paris";
7         break;
8     //Deuxième cas
9     case 78:
10        echo "Hauts de Seine";
11        break;
12    //Troisième cas
13    case 93:
14    case "Stade de France":
15        echo "Seine Saint Denis";
16        break;
17    //la suite des départements...
18    //Cas par défaut
19    default:
20        echo "Département inconnu en Ile de France";
21        break;
22 }
```

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) jusqu'à 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

```
1 for ($i=1, $j=9; $i<10, $j>0; $i++, $j--) {  
2 // $i varie de 1 à 9 et $j de 9 à 1  
3     $css="style=\"border-style:double;border-width:3;\"";  
4     echo "<span ". $css. ">$i + $j=10</span>";  
5 }
```

Boucles imbriquées

exemple3-7.php

```

1 echo "<h2>Révissez votre table de multiplication!</ h2>";
2 //Début du tableau HTML
3 echo "<table border=\"2\"
style=\"background-color:yellow\"> <th>&nbsp;X &nbsp;</th>";
4 //Création de la première ligne
5 for ($i=1;$i<10;$i++)
6   echo "<th>&nbsp;$i&nbsp;</th>";
7 //Création du corps de la table
8 //Boucles de création du contenu de la table
9 for ($i=1;$i<10;$i++) {
10   //Création de la première colonne
11   echo "<tr><th>&nbsp;$i&nbsp;</th>";
12   //Remplissage de la table
13   for ($j=1;$j<10;$j++) {
14     echo "<td style=\"background-color:red;color:white\">
&nbsp;&nbsp;&nbsp;<b>". $i*$j."&nbsp;&nbsp;&nbsp;</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, "&nbsp;";
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 $n, "&nbsp; / ";
5 }
6 while ($n%7!=0);
```

La boucle `foreach`

Pour itérer sur les éléments d'un tableau

- Plus efficace que `for`.
- Adaptée aux tableaux associatifs : elle ne nécessite 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

```
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

```
1 //Création du tableau
2 for ($i=0; $i<=8; $i++) {
3     $tab[$i] = pow(2, $i);
4 }
5 //Lecture des indices et des valeurs
6 foreach ($tab as $ind=>$val) {
7     echo " 2 puissance $ind vaut $val <br />";
8 }
9 echo "Dernier indice ", $ind, " ,dernière valeur ", $val;
```

foreach pour récupérer clés et valeurs d'un tableau

exemple3-12.php

```
1 //Création d'un tableau associatif
2 for ($i=0; $i<=8; $i++) {
3     $stabass["client".$i] = rand(100,1000);
4 }
5 //Lecture des clés et des valeurs
6 foreach ($stabass 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 imbriquant l'instruction.

Sortie anticipée de boucle avec break

exemple3-13.php

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

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

Avancement de boucle avec continue

continue

Arrête l'itération en cours, pas la boucle.

exemple3-14.php

```
1 //Interruption d'une boucle for
2 for ($i=0; $i<20; $i++)
3 {
4     if ($i%5==0) { continue; }
5     echo $i, "<br />";
6 }
7 //Interruption d'une boucle foreach
8 $tab[1]="Ain";
9 $tab[2]="Allier";
10 $tab[27]="Eure";
11 $tab[28]="Eure-et-Loir";
12 $tab[29]="Finistère";
13 $tab[33]="Gironde";
14 foreach ($tab as $cle=>$valeur)
15 {
16     if ($tab[$cle][0]!="E") { continue; }
17     echo "code $cle : département ", $tab[$cle], "<br />";
18 }
```

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

```
1  for ($i=0; $i<10; $i++)
2  {
3      for ($j=0; $j<10; $j++)
4      {
5          for ($k=0; $k<10; $k++)
6          {
7              if ( ($i+$j+$k) %3==0) continue 3;
8              echo "$i : $j : $k <br /> ";
9          }
10     }
11 }
```

Gestion des erreurs ("inévitables")

Vise à produire un "rapport" d'erreurs utile à l'utilisateur

```
1 $x=2;$y=0;  
2 echo $x/$y;  
3 fopen("nofile.txt","r");
```

Suppression des messages d'erreur renvoyés au poste client

- `@f()` : faire précéder l'appel de la fonction `f` par `@`.
- `error_reporting(n)` ; en début de script n'affiche que les erreurs de niveau `n` :

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

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

```
1 $a=100;$b=0;
2 try{
3     if($b===0) {
4         throw new Exception("Division par 0",7);
5     } else {
6         echo "Résultat de : $a / $b = ",$a/$b;
7     }
8 } catch(Exception $e) {
9     echo "<hr>Message d'erreur : ",$e->getMessage();
10    echo "<hr>Code d'erreur : ",$e->getCode();
11    echo "<hr>Nom du fichier :",$e->getFile();
12    echo "<hr>Numéro de ligne :",$e->getLine();
13    echo "<hr>__toString : ",$e->__toString();
14 } finally {
15     echo "<hr>L'exception a été traitée, le script continue.";
16 }
17 echo "<hr>Vraie Fin";
```

Exemple amélioré avec boîte d'alerte

exemple3-18.php

```
1 $a=10;
2 $b=0;
3 try {
4     if ($b===0) {
5         throw new Exception("Division par 0",7);
6     } else {
7         echo "Résultat de : $a / $b = ", $a/$b;
8     }
9 } catch (Exception $e) {
10     $c = $e->getCode();
11     $m = $e->getMessage();
12     echo "<script>alert(' Erreur numéro ", $c, " \\n ", $m, "
' );</script>";
13 } finally {
14     echo "Tout est sous contrôle<br/>";
15 }
16 echo "FIN";
```

Exceptions personnalisées

Par création de classe

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

exemple3-19.php

```
1 class MonException extends Exception {
2     public function alerte() {
3         $this->message = "<script> alert(' Erreur numéro ". $this->getCode() . " \\n
\" . $this->getMessage() . " ' )</script> ";
4         return $this->getMessage();
5     }
6 }
7 // Utilisation de la classe
8 $a=100; $b=3;
9 try {
10     if ($b == 0) {throw new MonException("Division par 0",7);}
11     elseif ($a%$b != 0) {throw new MonException("Quotient entier impossible",55);}
12     else{echo "Résultat de : $a / $b = ", $a/$b;}
13 } catch (MonException $except) {
14     echo $except->alerte();
15 } finally {
16     echo "Le script continue sans problème<br/>";
17 }
18 echo "FIN";
```