



Le langage PHP

03 - STRUCTURES DE CONTRÔLE



Rappels : variables, types et opérateurs

Dans la mission précédente, vous avez appris comment :

- **déclarer** une **variable** (définie par un **nom**)
- **affecter** une **valeur** à un variable
- « *manipuler* » les **opérateurs** principaux

Vous êtes capable d'écrire :

- des **expressions** simples
- des **instructions** simples, constituées d'expressions
- des **programmes élémentaires**

Mais cela ne suffit pas... :

- Comment écrire des **programmes complexes** ?
- nécessité de disposer de **structures de contrôle**



Les structures de contrôle

Deux grandes catégories :

- Structures **conditionnelles** :
 - **Exécution** des parties d'un programme **sous certaines conditions**
 - Partie d'un programme : *bloc d'instructions*
 - Mots-clés : `if... else...`
- Structures **itératives** (aussi appelées « *boucles* ») :
 - Répétition de l'**exécution** des parties d'un programme **un certain nombre de fois**
 - Partie d'un programme : *bloc d'instructions*
 - Mot-clé : `for...`



PHP : Structure conditionnelle

Syntaxe générale :

```
if (condition) {  
    instruction1;  
    instruction2;  
    ...;  
}  
else {  
    instruction3;  
    instruction4;  
    ...;  
}
```

Explication :

- `condition` doit être une **expression booléenne**
- `condition` vaut donc :
 - soit vrai (`true`)
 - soit faux (`false`)
- **Si** `condition` est vrai :
 - `instruction1` puis `instruction2` sont exécutées
- **Sinon**
 - `instruction3` puis `instruction4` sont exécutées



PHP : Structure conditionnelle

Exemple simple (ci-contre) :

- condition définie par la variable booléenne `$gameOver`
- le premier *bloc d'instructions* est exécuté
- **Note** : les *blocs d'instructions* sont écrits entre **accolades**

`{ ... }`

```
<?php  
  
$gameOver = true;  
  
if ($gameOver) {  
    echo 'Vous avez perdu !';  
} else {  
    echo 'Vous avez gagné !';  
}  
  
?>
```



Vous avez perdu !



PHP : Structure conditionnelle

Exemple plus complexe (ci-contre) :

- conditions élaborées :
 - ! : **négation** logique
 - && : **ET** logique
 - >, <= : **comparaison** de valeurs
 - valeur de la condition : true ou false
 - utilisation des parenthèses (...)
- structure différente :
 - **if {...} else if {...} else {...}**

```
<?php

$gameOver = false;
$score = 1789;
$highScore = 1492;

if (!$gameOver && ($score > $highScore)) {
    echo 'Vous êtes le meilleur !';
} else if (!$gameOver && ($score <= $highScore)) {
    echo 'Encore un effort !';
} else {
    echo 'Vous avez perdu !';
}

?>
```



Vous êtes le meilleur !



PHP : Structure itérative

Syntaxe générale :

```
for (initialisation; condition; incrementation) {  
    instruction1;  
    instruction2;  
    ...;  
}
```

Explication :

- **initialisation** : état initial de la variable de boucle
- **condition** : condition d'arrêt
- **incrementation** : incrémentation de la variable de boucle



PHP : Structure itérative

Exemple simple (ci-contre) :

- `$i` : variable de boucle
- `$i = 0` : initialisation de la variable de boucle
- `$i < 5` : condition d'arrêt
- `$i++` : incrémentation de la variable de boucle
- **Note** : le *bloc d'instructions* est écrit entre **accolades**

`{ ... }`

```
<?php

for ($i = 0; $i < 5; $i++) {
    echo 'i vaut : ' . $i . '<br>';
}

?>
```



```
i vaut : 0
i vaut : 1
i vaut : 2
i vaut : 3
i vaut : 4
```




PHP : Structure itérative

Explication (suite) :

- **initialisation** : déclaration de la variable `$i` avec comme valeur initiale `0`
- **condition** : la boucle est répétée tant que la condition est vraie (la boucle s'arrête donc dès que `$i` atteint la valeur `5`)
- **incrementation** : `$i++` indique que `$i` est incrémenté d'une unité (i.e. de `1`) à chaque nouveau tour de boucle

Fonctionnement :

- au **premier tour de boucle**, `$i` prend la valeur `0`, donc la condition (`$i < 5`) est vraie
- le bloc d'instructions est exécuté avec cette valeur `0` (affichage de « *i vaut : 0* »)
- un **second tour de boucle** commence : `$i` est incrémenté de `1`, `$i` vaut alors `1` (`0 + 1`)
- la condition est testée (`$i < 5`) : elle vaut encore `true`
- le bloc d'instructions est exécuté avec la nouvelle valeur de `i` (affichage de « *i vaut : 1* »)
- ... et le processus est répété jusqu'à ce que `$i` atteigne la valeur `5`



PHP : Structure itérative

Exemple plus complexe (ci-contre) :

- `$j` : variable de boucle
- le **pas d'incrément** est de **2**
- `$j` prends successivement les valeurs : **-3, -1, 1, 3**
- la boucle s'arrête avec une valeur de `$j` égale à **5**

```
<?php

for ($j = -3; $j <= 3; $j+=2) {
    echo $j . ' x 2 = ' . $j * 2;
    echo '<br>';
}
echo 'Ici, $j vaut : ' . $j;

?>
```



```
-3 x 2 = -6
-1 x 2 = -2
1 x 2 = 2
3 x 2 = 6
Ici, j vaut : 5
```



PHP : Itérations et tableaux

Un cas classique d'utilisation des « *boucles* » :

- parcourir un ensemble de données
- afficher la valeur de la donnée correspondant à chacun des index

Idée générale (représentation d'un tableau de joueurs) :

Index n° →	0	1	2	3	4	5	6
Valeurs →	James	William	Peter	John	David	Oliver	Jack



PHP : Itérations et tableaux

Exemple en langage PHP :

```
$ranking = array('James', 'William', 'Peter', 'John', 'David', 'Oliver', 'Jack');  
$playersNum = count($ranking);  
  
for ($i = 0; $i < $playersNum; $i++) {  
    echo 'N°' . $i . ' - ' . $ranking[$i];  
    echo '<br>';  
}
```



N°0 - James
N°1 - William
N°2 - Peter
N°3 - John
N°4 - David
N°5 - Oliver
N°6 - Jack



PHP : Itérations et tableaux

Explication générale :

```
$ranking = array('James', 'William', '...');
```

→ **Déclaration** d'une variable nommée `$ranking` de type **tableau** (array) et **affection** d'un ensemble de **valeurs initiales** à ce tableau (...)

Explication détaillée :

- la liste de valeurs entre (...) définit les valeurs initiales du tableau (ici : des chaînes de caractères entre '...')
- la valeur du premier index vaut toujours 0



PHP : Opérations sur les tableaux

Opérations sur les tableaux :

- Accès à la valeur d'index n°4 du tableau (5^{ème} valeur) :
→ `$ranking[4] : vaut 'David'`
- Récupération de la taille du tableau (nombre de valeurs) :
→ `count($ranking) : vaut 7`
- Modification de la valeur d'index n°5 du tableau :
→ `$ranking[5] = 'Matthew';`
- **Raccourci syntaxique** pour définir un tableau (opérateurs `[]`) :
→ `$ranking = ['James', 'William', '...'];`



PHP : Opérations sur les tableaux

Opérations sur les tableaux :

Tableau `$ranking`

0	1	2	3	4	5	6
James	William	Peter	John	David	Oliver Matthew	Jack

Index °4 Index °5

`$ranking[4]` `$ranking[5] = 'Matthew'`



PHP : Structures de contrôle - Résumé

- + Structure **conditionnelle** : `if {...} else {...}`
- + Structure **itérative** : `for (...; ...; ...) {...}`
- + **Types scalaires vs types composés** (ex. : tableau) : `array()`



Base de tout programme élaboré reposant sur des données évoluées



Possibilité d'écrire une **quasi-infinité** de programmes



Conclusion

- Autres structures de contrôle en PHP (*abordées plus tard*) :
 - a. Conditionnelles : `switch (...) {...}`
 - b. Itératives : `while (...) {...}` | `do {...} while (...)` | `foreach (...) {...}`
- Autres structures de données évoluées (*abordées plus tard*) :
 - a. Objets : `Object`
 - b. Énumérations : `Enum`

>>> MISSION START ! <<<