



PHP & SQL - Bases - 5 - Les Boucles

⌚ Cours	PHP
⌚ Type	Guide
📎 Supports	https://www.php.net/manual/fr/language.control-structures.php
⌚ Difficulté	Facile

Dans un programme informatique, on a souvent besoin de répéter la même chose plusieurs fois (que ce soit un nombre de répétitions déterminé à l'avance ou non). Par exemple, en PHP il peut être intéressant d'utiliser une boucle lorsqu'un utilisateur est invité à entrer un mot de passe. Si celui-ci n'est pas bon, alors on lui redemandera le mot de passe. Et on peut effectuer ceci un nombre de fois à l'infini. Bien sûr, pour sécuriser notre site, il vaut mieux limiter le nombre d'essais à 3 avant de bloquer l'accès quelques minutes. Là aussi on utilisera une boucle !

Une boucle en PHP s'écrit de multiples manières. On peut faire des boucles pour toute sorte de choses et nous allons détailler les principales ensemble.

La boucle while

La boucle While correspond à un “TANT QUE”. En français on pourrait dire : “Tant que le mot de passe n'est pas bon, demander à l'utilisateur d'entrer un mot de passe”.

La syntaxe est la suivante :

```
<?php  
$password = "azerty"; // Le mot de passe souhaité  
$password_input = "chocobon"; // Le mot de passe entré par l'utilisateur  
  
while ($password_input != $password) {  
    ask_password(); // On demandera à l'ordinateur de demander un nouveau mot de passe.  
}
```

Bon. Évidemment, le mot de passe ne sera jamais écrit comme ça dans un programme. Mais nous avons besoin de vulgariser un peu en outrepassant certaines règles pour éviter de compliquer votre apprentissage.

 `ask_password()` est une fonction que nous avons nous-mêmes définie. Nous verrons les fonctions lors de notre prochain chapitre.

Pour que notre programme soit un peu plus précis, on peut limiter le nombre d'essais à 3.

```
<?php  
$password = "azerty"; // Le mot de passe souhaité  
$password_input = "chocobon"; // Le mot de passe entré par l'utilisateur  
$essai = 0;  
  
while (($password_input != $password) && ($essai <= 3)) {  
    ask_password(); // On demandera à l'ordinateur de demander un nouveau mot de passe.  
    $essai++;  
}
```

La boucle do-while

Dans un contexte comme celui-ci, il serait encore plus judicieux d'utiliser la boucle `do while`. Cette boucle permet d'exécuter la boucle AU MOINS une fois avant de se

répéter. Ce qui dans notre cas est exactement ce que nous souhaitons, non ? On demande d'abord à l'utilisateur de saisir un mot de passe.

Voici la syntaxe :

```
<?php  
$password = "azerty"; // Le mot de passe souhaité  
$password_input = "chocobon"; // Le mot de passe entré par l'utilisateur  
  
do {  
    ask_password();  
} while ($password_input != $password);
```

Le problème avec cette syntaxe, c'est qu'elle est plus difficile à retenir. On écrit d'abord un Do, suivi d'un while à la fin du bloc qui sert de condition. Pour la retenir, je vous invite à vous forcer d'appliquer cette boucle le plus souvent possible, même lorsque ce n'est pas approprié pour vous entraîner.

i Avez vous déjà entendu parler de boucle infinie ? Il s'agit d'une boucle qui s'exécute en boucle car la condition est toujours vraie. Cette boucle maudite vous prendra de plus en plus de mémoire et pourra potentiellement faire cracher votre programme. Faites-donc attention lorsque vous utilisez des boucles sans connaître le nombre d'itérations.

Exercice 7 :

Écrivez un programme qui écrit la phrase suivante “Voici une ligne”. (Utilisez un echo).

Répétez cette instruction 5 fois.

Bonus : Essayez de faire un saut de ligne entre chaque phrase.

La boucle for

La boucle for est connue pour être l'une des plus difficiles à maîtriser lorsque l'on débute en PHP. Sa structure est un peu différente de la boucle While. Cependant, vous allez la rencontrer dans la majorité des programmes. C'est pourquoi je ne me fais pas de soucis concernant votre capacité à maîtriser cette boucle.

La boucle for permet d'itérer un nombre déterminé (ou non) de fois, l'instruction souhaitée. Elle est aussi utilisée pour parcourir les éléments d'un tableau et effectuer des instructions sur chacun de ces éléments 😊.

Voici la syntaxe :

```
<?php
for ($i = 1; $i <= 10; $i++) {
    echo $i;
}
```

Cette boucle commence donc par le mot clé for suivi d'une condition un peu particulière entre les parenthèses.

La première instruction `$i = 1` est une variable, que nous avons initiée et que nous allons utiliser pour effectuer cette boucle.

La deuxième instruction `$i <= 10` est une condition (Voyez ça comme le `while`). Tant que `$i` est inférieur ou égal à 10, répétons la boucle.

La dernière instruction est l'incrémentation de notre variable `$i` après l'exécution de la boucle.

Si cette dernière instruction se trouve ici (et pas à la fin du bloc), c'est tout simplement parce que la boucle for attend 3 paramètres pour ensuite s'exécuter. Elle n'a pas besoin d'attendre la fin de la boucle pour savoir quoi faire, elle se relancera d'elle-même selon les conditions fixées par le développeur.

Il existe d'autres syntaxes que vous pouvez rencontrer dans la documentation officielle. Je pense qu'il est utile d'y jeter un coup d'œil pour les connaître, mais ne vous sentez pas obligés de maîtriser ces autres syntaxes. La plupart des développeurs utilisent uniquement la première.

Maintenant il est temps de vous présenter la syntaxe lorsque l'on parcourt un tableau (c'est ce qui va être le plus difficile au début).

```
<?php
$student = [
    "john",
    "jane",
    "max",
```

```

    "hugo"
];

for($i = 0; $i < count($student); ++$i) {
    echo "$student[$i]" . '<br>';
}

```

La méthode `count()` permet de compter les éléments d'un tableau. (On lui a donc spécifié quel tableau elle doit compter).

Nous avons commencé par créer un tableau contenant des étudiants. Ensuite nous avons créé une boucle `for` qui s'exécutera en prenant comme valeur de départ 0 (contenu dans la variable `$i`). L'avantage avec cette technique, c'est que nous pouvons maintenant manipuler chaque élément du tableau en utilisant cette variable pour préciser l'indice du tableau `$student`.

Ainsi à chaque répétition, nous exécuterons dans l'ordre : Afficher étudiant 0, Afficher étudiant 1, Afficher étudiant 2 etc..

La boucle foreach

Voyez ça comme de la chance, mais si vous avez des difficultés à utiliser la boucle `for` lorsque vous souhaitez parcourir un tableau, sachez que la boucle `foreach` peut vous aider ! Celle-ci sert spécifiquement à parcourir les éléments d'un tableau (ou d'un objet mais on n'a pas encore vu ça).

La syntaxe se présentera de cette manière :

```

<?php
$student = [
    "john",
    "jane",
    "max",
    "hugo"
];
foreach ($student as &$element) {
    $element = "maxime"; // On renomme tous les prénoms en "maxime".
}
// Ajoutez un petit print_r($student) ici pour tester.
unset($element);

```

La boucle `foreach` commence par le mot clé `foreach` (étonnant !), suivi d'arguments contenus dans les parenthèses. Ces arguments sont les suivants : Le tableau que

l'on va parcourir est suivi d'un `as` qui va permettre d'affecter un nom à chaque élément de ce tableau (on a choisi `$element` mais vous pouvez écrire ce que vous voulez).

Enfin, le `unset()` est une petite spécificité particulière que vous devriez garder en tête le jour où vous faites plusieurs `foreach` les uns à la suite des autres.

Sans celui-ci, vous pouvez toujours appeler `$element` en dehors de la boucle. Ce qui veut dire qu'actuellement (si `unset()` n'existe pas) `$element` est égal à `$student[3]` (qui est la dernière valeur parcourue).

i Il existe une dernière petite variante de cette boucle `foreach` qui consiste à réaffecter non seulement les valeurs à un nouveau nom, mais aussi les clés de chaque valeur. Cela ressemble à quelque chose comme : `foreach ($student as $cleDeLelement => &$element)`.

💡 Normalement, il n'est pas possible de renommer chaque valeur de cette manière. Ceci est rendu possible grâce au petit `&` qui a été ajouté devant `$element` dans les arguments de la boucle `foreach`. Essayez de le retirer pour apercevoir un autre résultat. Nous verrons cette petite chose bien plus tard lors de la POO (programmation orientée objet) mais gardez-le dans un petit coin de votre tête pour impressionner vos collègues débutants 😊.

Les boucles imbriquées

On passe maintenant à une notion un peu plus complexe. Il est possible de mettre tout un tas de choses dans une boucle, y compris, une autre boucle. Vous pouvez même avoir un programme qui contient une boucle, dans une boucle, dans une boucle, dans une boucle... (bon, là, par contre, vous avez sûrement un petit problème d'organisation).

On va essayer de voir ça ensemble de manière théorique car vous allez devoir bosser dessus vous-mêmes pour notre premier projet ! 😊

Imaginons que l'on souhaite multiplier 1 par 0, puis 1, puis 2 jusqu'à 10. Imaginons ensuite que nous souhaitions répéter cela 10 fois en incrémentant notre premier

nombre à chaque itération. Il nous faudra alors deux boucles pour effectuer ce programme et pour obtenir une table de multiplication.