PHP – Les bases – Partie 1

Définition Wikipédia

PHP: Hypertext Preprocessor, plus connu sous son sigle PHP (Acronyme récursif), est un langage de scripts libre principalement utilisé pour produire des pages Web dynamiques via un serveur HTTP, mais pouvant également fonctionner comme n'importe quel langage interprété de façon locale, en exécutant les programmes en ligne de commande.

PHP est un langage disposant depuis la version 5 de fonctionnalités de modèle objet complètes. En raison de la richesse de sa bibliothèque, on désigne parfois PHP comme une plate-forme plus qu'un simple langage.

Bref historique

Le langage PHP fut créé en 1994 par Rasmus Lerdorf pour son site web. C'était à l'origine une bibliothèque logicielle en Perl dont il se servait pour conserver une trace des visiteurs qui venaient consulter son CV.

Au fur et à mesure qu'il ajoutait de nouvelles fonctionnalités, Rasmus a transformé la bibliothèque en une implémentation en langage C, capable de communiquer avec des bases de données et de créer des applications dynamiques et simples pour le Web.

Rasmus décida alors en 1995 de publier son code, pour que tout le monde puisse l'utiliser et en profiter. PHP s'appelait alors PHP/FI (pour Personal Home Page Tools/Form Interpreter).

En 1997, deux étudiants, Andi Gutmans et Zeev Suraski, redéveloppèrent le cœur de PHP/FI. Ce travail aboutit un an plus tard à la version 3 de PHP, devenu alors PHP: Hypertext Preprocessor.

Peu de temps après, Andi Gutmans et Zeev Suraski commencèrent la réécriture du moteur interne de PHP. Ce fut ce nouveau moteur, appelé Zend Engine - le mot Zend est la contraction de ZEev et aNDi - qui servit de base à la version 4 de PHP.

La version 5 apporta sont lot de nouveautés, dont l'orienté objet avancé et un typage plus stricte.

La version actuelle stable la plus utilisée est la version 7.4 et sera ensuite suivie par la version 8 qui est finalisée depuis peu.

.

La dernière mise à jour est la 8.0.7 datant du 3 juin 2021

Utilisation :

Avec le HTML on peut déjà créer des pages web statiques. Le rôle de PHP est de les rendre dynamiques. Grâce à son interaction avec la base de données MySQL et son intégration facile dans du code HTML, PHP est devenu le langage le plus utilisé en développement web.

PHP permet de développer des sites de vente en ligne, blog, tchat, forum et bien d'autres choses...

Bien que PHP soit utilisé généralement pour la création de site web, sa capacité n'est pas limitée à ce domaine.

Le PHP est open source. Il est distribué via une licence propre qui précise qu'il peut être utilisé, modifié, utilisé librement et gratuitement (licence à peu près identique à celle d'Apache).

Vous pouvez télécharger PHP gratuitement depuis le site officiel http://www.php.net. L'autre avantage de PHP est qu'il est facile à prendre en main pour les débutants. De plus, il offre des fonctionnalités avancées pour les experts.

Comment fonctionne le langage PHP

Contrairement au code Javascript présent dans une page web , le code PHP est exécuté sur le serveur (on dit alors 'interprété'). Lors de la visite d'une page web, il y a ce que l'on appelle une requête client-serveur.

Dans un premier temps le client saisit une adresse à travers un navigateur (Firefox par exemple) et en retour, le navigateur retourne la page recherchée.

Par définition, un serveur web est un "ordinateur" présent sur l'Internet et accessible via le réseau qui héberge la page que vous demandez.

Sur ce serveur on trouve Apache, logiciel apte à traiter les requêtes HTTP que vous envoyez lorsque vous demandez une page web.

Apache va donc chercher le fichier demandé dans son arborescence et renvoie à votre navigateur la page HTML5. Votre navigateur interprète les différents langages se trouvant dans ce fichier (HTML, JavaScript, CSS, etc. ...) et affiche la page.

En réalité, Il y a une étape qui vient s'ajouter entre la demande (côté client) et l'envoi (côté

serveur) d'une requête. Avant l'envoi d'une réponse, le serveur lit d'abord soigneusement le fichier PHP afin d'y chercher s'il y a des tâches à accomplir. Une fois que le serveur a terminé ce qu'il est censé faire, alors seulement le résultat est envoyé au client. Il importe de savoir que le client ne voit que le résultat des travaux du serveur, et pas les instructions proprement dites.

Cela signifie que si vous cliquez sur « voir la source de la page » pour une page PHP, vous ne verrez pas le code PHP, seulement des balises HTML élémentaires. On ne peut donc pas voir comment est fabriquée une page PHP avec le « voir la source de la page ». PHP s'apprend donc autrement.

Installer un serveur local sur votre ordinateur

L'installation d'un serveur web sur son ordinateur personnel permet de tester ses scripts sans avoir à les renvoyer sans cesse par FTP sur le serveur distant.

L'ordinateur va alors servir à la fois de client et de serveur. Alors que pour tester des pages web html en local il suffit d'ouvrir le fichier dans un navigateur, il faut un serveur web sur votre PC local pour tester une page PHP.

Pour cela si on travaille sur Windows, on utilise un des nombreux utilitaires très pratiques qui installeront Apache (le serveur permettant de distribuer des pages web), PHP, MYSQL (le système de gestion de base de données), et si on travaille sur Linux, il y a fort à parier qu'Apache soit déjà automatiquement installé.

Les outils de développement : EasyPHP et WAMP ou Xampp

Les outils les plus connus sous windows sont : Xampp, EasyPHP et finalement WAMP, que nous allons installer.

Pour Mac Mamp est le plus connu, Xampp à l'avantage d'être multiplate-forme.

Les IDE

Pour le codage, il existe plusieurs sortes d'éditeur mais un simple éditeur de texte peut déjà suffire comme Notepad++ ou même le bloc note.

D'autres IDE (integrated development environment) plus aboutis sont nombreux : Netbeans (freeware), Sublime text (shareware), Visual Studio Code (freeware), PHPStorm (shareware) etc...

Le choix pour mon cours de PHP sera Netbeans, principalement pour sa capacité à gérer des projets multiples, une prise en charge des principaux framework, une historique très pratique (bien que lourde niveau mémoire), la possibilité de travailler en équipe sur le même projet et j'en passe....

J'utiliserai parfois VSC pour sa grande utilisation multi langage et son succès.

Nous passerons à PHPStorm pour ceux qui le souhaitent

Structure de PHP

Si le code HTML est placé entre une balise ouvrante <html> et une balise fermante < /html>, il en est de même pour le PHP, sauf que la balise de début est: <?php et la balise de fin est ?>.

Ceci permet d'indiquer au serveur qu'il s'agit bien d'un code PHP. Afin de mieux comprendre, voici un premier exemple de ce que pourrait être un programme PHP.

```
<?php
echo "Notre premier test";
?>
```

Pour le tester, il faudra enregistrer la page avec l'extension .php, par exemple "001.php" dans un sous dossier (nommé « basephp ») du dossier de travail indiqué lors de l'installation (souvent nommé par défaut www ou htdocs).

Puis allez dans le menu "Localhost" de WAMP, la page d'accueil s'ouvre. Là, si vous avez bien créé votre dossier contenant votre code en PHP dans le répertoire www comme indiqué, vous devriez voir un lien vers votre dossier créé. Cliquez dessus.

Une page web s'ouvre indiquant tous les fichiers qui se trouvent dans le dossier " basephp ". Vous devriez avoir le fichier " 001.php", en cliquant sur ce fichier: votre ordinateur génère alors le code PHP puis ouvre la page. Vous avez le résultat de votre code.

Pour faire plus simple, vous pouvez aller directement à cette URL:

http://localhost/basephp/001.php

ou celle-ci

http://127.0.0.1/basephp/001.php

Il existe également des virtualhost, qui permettront de créer un serveur par site (bonne pratique en local)

Affichage de texte en PHP

L'exemple ci-dessous nous affiche le texte "Notre premier test" à l'écran. Nous devons cela à la fonction echo. Notez bien les guillemets contenant le texte à afficher.

On met toujours le texte entre guillemets, ça permet de repérer ce qu'on veut afficher. Deux choix nous sont offerts : mettre un guillemet simple ou mettre un guillemet double.

Cependant, il faut faire bien attention à ne pas fermer un guillemet double avec un guillemet simple.

La fonction echo peut afficher plusieurs éléments que l'on concatène avec des '.'

```
<?php
echo ' un texte '.' et un autre texte';
?>
```

Il y aussi une autre fonction pour l'affichage. Il s'agit de la fonction print(). Elle donne le même résultat que celui de la fonction echo sauf qu'elle a(vait) besoin de parenthèses.

```
<?php
print (' un texte ');
print 'Encore du texte' ;
?>
```

Le séparateur d'instructions

Toujours dans le même exemple, nous appelons la partie: echo "Notre premier test"; une instruction. A la fin d'une phrase dans un texte, il y a un point pour terminer la phrase. De même, les instructions en php sont séparées par un point virgule. Ce point virgule indique la fin de chaque instruction.

```
<?php
echo "instruction 1 et";
echo "instruction 2" ;
?>
```

Essayez d'enlever le point virgule à la fin et vous verrez qu'il y aura un message d'erreur "Parse Error". Le ; n'est pas indispensable devant « ?> » ou « } », mais mieux vaut en mettre trop que pas assez...

Les caractères spéciaux

Maintenant, parlons d'une chose très spéciale qui mérite une grande attention.

On avait dit qu'il faut entourer de guillemets les chaînes de caractère à afficher. Mais si le texte à afficher contient lui-même des guillemets. Comment la fonction echo peut savoir qu'il ne s'agit pas encore de la fermeture de guillemets ? Évidement, elle sera dans l'embarras si on ne lui indique rien. En effet, une instruction comme celle-ci:

```
<?php
echo ' Voici l'instruction ';
?>
```

Donnera à l'écran une erreur : **`Parse error: syntax error, unexpected T_STRING, expecting ',' or ';' in chemin\fichier.php on line 2`**
On peut échapper à ce genre de problème, en mettant tout simplement des antislashs. En voici alors une version corrigée de l'instruction précédente :

```
<?php
echo ' Voici l\'instruction ';
?>
```

On fait pareil au cas où il y aurait des "" ou des \ à l'intérieur des textes à afficher.

Ajouter des commentaires dans un script PHP

Il n'est pas obligatoire mais vivement conseillé de mettre des commentaires dans vos codes. Vous pouvez utiliser les commentaires pour expliquer votre code.

Supposons qu'il y très longtemps que vous n'ayez pas touché à vos codes. En revenant de vacances, vous décidez de les reprendre. Avec une centaine de lignes de code, il est tout à fait probable que vous galériez un peu si vous ne vous souvenez pas à quoi sert telle ou telle fonction..

Afin d'éviter la perte de temps dans ce genre de situations, ça ne vous coûte rien de mettre des commentaires dans vos codes. De plus, si vous travaillez à plusieurs sur un même projet, les autres comprendront plus rapidement votre code en lisant les commentaires.

Ceci étant dit, comment faire du commentaire alors ? Très simple, il y a deux façons :

Pour un commentaire sur une ligne, il suffit d'ajouter deux slash //

```
<?php
echo ' Voici l\'instruction '; //affichage du texte :Voici
l'instruction
?>
```

Et pour ajouter des commentaires sur plusieurs lignes, mettez-les entre /* et */

```
<?php
/* Ceci un programme php pour l'affichage d'un texte.
Et aussi sur les caractères spéciaux.
Désormais je commente mon code.
*/
echo ' Voici l\'instruction '; //affichage du texte : Voici
l'instruction
?>
```

Mais attention, il ne faut surtout pas imbriquer les commentaires comme ceci :

```
/* commentaire /*
```

Les variables

Une variable est un objet ou une information stockée en mémoire temporairement. C'est-à-dire qu'en PHP, la variable existe tant que la page est en cours de génération. Après l'exécution du programme, toutes les variables sont supprimées de la mémoire car elles ne servent plus à rien.

Une variable est caractérisée par son nom. Donc il est préférable de lui donner un nom significatif pour savoir de quoi il s'agit. En revanche, il faut noter que le nom de variable est sensible à la casse (majuscule / minuscule).

Un nom de variable valide doit commencer par une lettre ou un souligné (_), suivi de lettres, chiffres ou soulignés (nous éviterons aussi les caractères spéciaux et accentués, ainsi que des termes anglais pouvant être réservé par PHP) . La déclaration d'une variable se fait par l'écriture du symbole '$' devant la variable à manipuler.

Exemple :

```
$prix
```

Le nom de cette variable est prix.

Affectation d'une valeur à une variable en PHP

A une variable, on assigne une valeur. L'instruction d'affectation permet d'affecter une valeur à une variable. Il suffit de mettre un signe égal après la déclaration de la variable et de mettre la valeur associée comme suit.

```
$prix = 10; //La variable prix a comme valeur 10
```

Comme toute instruction PHP, il ne faut jamais oublier le signe; (point virgule) à la fin.

Les Types de variables en PHP

Selon la valeur qu'elle va contenir, une variable a un type. Contrairement à de nombreux autres langages, il n'est pas obligatoire en PHP de préciser les types.

Toutefois, il existe plusieurs types de variables en PHP dont les booléens, les réels, les entiers, les chaînes de caractères, les tableaux.

Les Booléens

Une variable de type booléen ou boolean a deux valeurs constantes possibles : TRUE ou FALSE. Ces constantes sont insensibles à la casse. Il s'agit donc d'une variable qui ne peut être que true ou false et rien d'autre.

Exemple :

```
<?php
$jeune = true;
```

```
?>
```

Les Entiers
Un entier ou integer est un nombre appartenant à l'ensemble des entiers Z: Z = {…,-1,0, 1, 2, ...}.

Exemple :

```
<?php
$age = 10;
?>
```

Les Réels
Ce sont les nombres décimaux connus aussi sous le vocable de " double ", " float " ou "nombre réels". Un prix peut par exemple être un réel.

Exemple:

```
<?php
$prix = 2.542;
?>
```

Les chaînes de caractères
Ce sont les variables de type texte, ou string. Comme l'on avait dit précédemment, il faut les mettre entre guillemet. Les chaînes de caractères sont des séquences de caractères.

Exemple :

```
<?php
$nom = "Dupont";
?>
```

Il y a encore d'autres types de variables comme les tableaux, les objets, etc, mais que nous verrons plus tard.

Les Constantes
Il s'agit des variables dont la valeur ne pourra être changée lors de l'exécution d'un programme. On utilise donc le type constante pour les valeurs qui ne seront pas susceptibles de changer au cours de l'exécution du programme. C'est le cas du nombre pi par exemple, ou de la TVA, etc

La fonction utilisée pour la définition d'une constante en PHP est la fonction define() qui nécessite en paramètres le nom de la variable et sa valeur. La syntaxe de la fonction define() est la suivante: define("Nom_de_la_constante", Valeur);

Exemple :

```
<?php
define("TVA"', 19.6);
echo TVA;
?>
```

Tout au long de l'exécution la valeur de TVA utilisée sera 19.8. Quand on utilise la constante TVA, on ne met plus de guillemet.
La différence entre les constantes et les variables PHP est qu'elles ne sont pas précédées du signe $. Par convention, on écrit les constantes en majuscules.

Les conditions

Une structure conditionnelle permet d'exécuter ou non une série d'instructions en fonction d'une condition d'origine. Si le calcul de cette condition retourne TRUE alors le bloc d'instructions concerné est exécuté.

Les expressions évaluées peuvent être plus ou moins complexes, c'est-à-dire qu'elles peuvent être constituées d'une combinaison d'opérateurs de comparaison, d'opérateurs logiques et même de fonctions. Le langage PHP introduit 4 constructions conditionnelles: if, elseif, else et switch.

Avant de les voir, il est nécessaire que l'on aborde les opérateurs de comparaison et les opérateurs logiques.

Les opérateurs de comparaison

Les opérateurs de comparaison sont utilisés pour comparer deux valeurs. Elles permettent souvent de définir des conditions dans les structures conditionnelles.

Egal : $a == $b retourne true si la valeur de $a est égale à celle de $b

Différent : $a != $b retourne true si la valeur de $a est différente de celle de $b

Inférieure : $a < $b retourne true aussi si la valeur de $a est strictement inférieure à celle de $b

Inférieure ou égal : $a < = $b retourne true aussi si la valeur de $a est inférieure ou égale à celle de $b

Supérieure : $a > $b retourne true aussi si la valeur de $a est strictement supérieure à celle de $b

Supérieure ou égal : $a >= $b retourne true aussi si la valeur de $a est supérieure ou égale à celle de $b

Les opérateurs logiques

En PHP, il y a différents opérateurs logiques qui permettent de combiner des conditions entre elles.

La structure conditionnelle if, else

Cette structure permet de n'exécuter qu'un bloc d'instructions uniquement si l'expression est vraie. Le mot clé if() signifie en anglais 'si'. Autrement dit, si la condition est vérifiée, on exécute l'instruction. Dans le cas contraire, l'instruction sera simplement ignorée ou une autre instruction alternative sera exécutée.

Par conséquent, un if peut être employé seul. Le else étant l'alternative, on peut s'en passer pour n'exécuter un code que si une condition est réalisée.

Le if

La syntaxe est :

```
<?php
if(condition)
{
  // Bloc d'instructions
}
?>
```

On met la ou les conditions entre parenthèses, et le bloc d'instructions entre deux accolades.

Exemple :

```
<?php
$vitesse = 100;

if($vitesse > 50)
{
  echo "excès de vitesse !";
}
?>
```

Le else

La clause else (traduite par sinon), ajoutée après l'accolade fermante du bloc if(), permet de définir une série d'instructions qui seront exécutées si l'expression if testée est fausse (c'est-à-dire si elle renvoie FALSE).

Voici la syntaxe :

```
<?php
if (condition) {
  // instruction au cas où la condition serait réalisée;
}else{
  // instruction au cas où la condition ne serait pas réalisée;
}
?>
```

Exemple :

```
<?php
$age = 22;

if($age >= 18)
{
  echo "Vous êtes majeur.";
}
else
{
  echo "Vous êtes mineur.";
}
?>
```

Imbrication else if

On peut imbriquer les if les uns dans les autres. Simplement lorsqu'un if imbriqué aura fini d'être exécuté, il retournera à l'étape logique suivante du rang hiérarchique supérieur.

Exemple :

```
<?php
$nombre = -4;
if($nombre == 0)
{
  echo "le nombre est égal à zéro";
}
else
{
  if($nombre > 0) {
     echo "le nombre est positif";
  } else {
     echo "le nombre est négatif";
  }
}
?>
```

On peut simplifier le code en utilisant elseif. Et le nombre d'elseif est illimité. Sauf que le else à la fin est obligatoire et il est exécuté lorsque aucune des conditions au dessus ne sont pas exécutées.

Exemple :

```
<?php
$nombre = -4;
if($nombre == 0)
{
echo "le nombre est égal à zéro";
}
elseif($nombre > 0)
{
  echo "le nombre est positif";
```

```
}
else
{
  echo "le nombre est négatif";
}
?>
```

Le switch

Il existe une autre alternative à la structure if() / elseif() / else ou bien aux imbrications de blocs if(). Elle se nomme switch() (traduit par 'au cas où').

Sa syntaxe est assez simple et repose sur l'utilisation de 3 mots clés : switch, case et default.

Cette instruction conditionnelle permet de tester toutes les valeurs possibles que peut prendre une variable.

Exemple d'utilisation de l'instruction switch en PHP

```
<?php
$legume = "rien";

switch($legume)
{
 case 'salade':
  echo'Vous avez acheté de la salade !';
  break;

 case 'Carotte':
  echo'Vous avez acheté de la Carotte !';
  break;

 case 'poivrons':
  echo'Vous avez acheté des poivrons!';
```

```
  break;

 case 'aubergines':
  echo'Vous avez acheté des aubergines!';
  break;

 default :
  echo 'Vous avez acheté un autre légume' ;
  break;
}

?>
```

Dans cet exemple, $legume est la variable à tester.

Les différents 'case' testent la valeur, et exécutent le code contenu entre le 'case' en question et le 'break'.

L'instruction contenue dans la clause default est l'instruction à exécuter par défaut lorsque la variable $legume ne prend aucune des valeurs définies dans les différents 'case'.

Les boucles

Les boucles sont des structures de contrôle permettant d'effectuer un certain nombre de fois les mêmes opérations. Concrètement, il s'agit d'une répétition d'instructions.

Par exemple afficher une suite de nombres que l'on incrémentera à chaque tour de boucle en fonction d'une ou plusieurs conditions à vérifier.

Les opérateurs d'incrémentation et de décrémentation

Incrémenter une valeur signifie qu'on augmente la valeur d'un certain ordre. Pour dire que la valeur d'une variable $i a augmenté d'un, on écrit :

```
$i = $i+1 ;
```

Cette instruction peut être écrite d'une manière plus simple en utilisant une instruction

équivalente :

```
$i += 1 ;
```

ou tout simplement :

```
$i ++ ;
```

A l'inverse, décrémenter une valeur veut dire qu'on diminue la valeur de $i d'un certain nombre de points. La syntaxe est la même que celle de l'incrémentation, sauf qu'à la place du signe + (plus) on met un signe – (moins).

Il est tout à fait possible de faire : $i += 2; pour dire que la valeur de $i augmentera de deux points.

La boucle while

C'est le moyen le plus simple pour faire une boucle. On traduit la boucle while par 'tant que'. Tant que la/les conditions est/sont vérifiée/s, on traite les instructions situées dans la boucle. La particularité de cette instruction est que la condition est testée à chaque début de boucle.

Syntaxe de while :

```
<?php
while(condition)
{
  // instruction 1;
  // instruction 2;
  } ?>
```

Dans cet exemple, on affichera tous les nombres pairs qui sont inférieurs à 20.

Exemple :

```
<?php
$i = 0; // $i est un nombre que l'on incrémentera.
while($i < 20)
{
  echo $i . "<br />";
  $i += 2 ;
}
```

```
?>
```

L'essentiel à retenir est que la boucle while() signifie que l'on va répéter un bloc d'instructions tant que la condition passée en paramètre reste vraie (TRUE). Lorsque celle-ci deviendra fausse (FALSE), le programme sortira de la boucle.

Il est donc impératif que le bloc d'instructions, d'une manière ou d'une autre, agisse sur le paramètre. Dans l'exemple précédent, si l'on ne met pas la ligne $i += 2, alors la boucle ne s'arrêtera pas ! Ce qui générera un temps d'attente pouvant bloquer le serveur (il y a heureusement des protections contre ce genre de problèmes, comme le timeout)

La boucle for

La boucle for() se traduit par 'pour chaque itération'. Il faut connaître par avance la condition d'arrêt. Autrement dit, la valeur qui rendra la condition fausse et stoppera la boucle. Sa syntaxe est simple et prend 3 paramètres obligatoires:

Syntaxe de for

```
<?php
for(initialisation; condition; incrémentation)
{
  bloc d'instructions;
}
?>
```

L'initialisation est l'expression qui permet d'initialiser la boucle (valeur de départ). Généralement, les boucles débutent à la valeur 0, mais ce n'est pas une généralité. Le second paramètre correspond à la condition d'arrêt de la boucle.

Cette condition est recalculée à chaque itération (passage de boucle) afin de déterminer si l'on continue de boucler ou bien si l'on sort de la boucle.

Enfin, le dernier paramètre détermine l'expression qui sera exécutée à la fin d'une itération. Généralement on prévoit ici une incrémentation (ou décrémentation) pour mettre à jour le compteur de la boucle.

Exemple :

```
<?php
// Boucle générant la table de multiplication du 8
for($i=0; $i<=10; $i++)
{
  echo "8 x " . $i . " = " . (8*$i) . "<br/>";
}
?>
```

La boucle do-while

L'instruction do{ ... } while() traduite par: 'répéter / faire ... tant que' est une alternative à l'instruction while().

Elle permet de tester la condition après la première itération et exécution du premier bloc d'instructions. Dans le cas de la boucle while, la condition est examinée avant la boucle tandis que pour la boucle do-while elle est examinée à la fin. Ainsi, même si cette condition n'est pas vérifiée, la boucle s'exécutera au moins une fois.

Syntaxe de do while :

```
<?php
do
{
  // bloc d'instructions;
}
while(condition); ?>
```

Exemple :

```php
<?php
// Déclaration et initialisation du compteur
$i = 0;
// Boucle générant la table de multiplication du 8
do
{
  echo "8 x ". $i . " = " . (8*$i) . "<br/>";
  // Incrémentation du compteur
  $i++;
}
while($i <= 10);
?>
```

Remarques: Il existe deux instructions qui permettent de modifier l'exécution des boucles, il s'agit de break et de continue.

L'instruction break permet de sortir de la boucle courante. C'est-à-dire qu'il interrompe directement l'itération. La condition de boucle est une condition qui est toujours vérifiée et, dans la boucle, on utilise break pour quitter celle-ci une fois que l'on est arrivé à nos fins.

'break' permet aussi de gérer des événements plus exceptionnels, comme des erreurs: en cas d'erreur, on quitte la boucle et on affiche un message.

La seconde instruction est l'instruction continue. Cette instruction permet de sauter les instructions de l'itération courante, afin de passer directement à l'itération suivante.

Reprenons l'exemple sur les nombres pairs. Cette fois-ci, la variable $i sera incrémentée de 1. On utilisera l'instruction continue pour éviter les nombres impairs.

```php
<?php
$i = 0; // $i est un nombre que l'on incrémentera.
while($i < 20)
{
 if (! ($i % 2) )
 {
  continue;
 }
```

```
echo $i . "<br />";
$i+=1 ;
}
?>
```