Informatique

Introduction au PHP, langage de programmation

Bruno Bernard SIMON
http://www.bbs-consultant.net

Avant-propos

Variables

Opérateurs

Structures de contrôle

Boucles

Fonctions

Historique

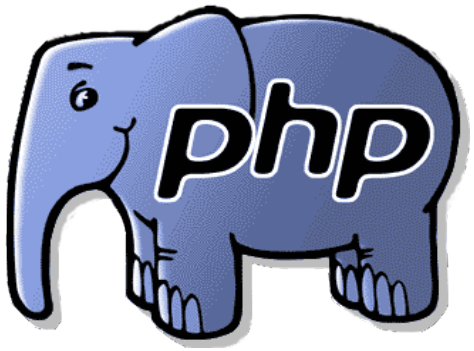
Rasmus LERDORF
Invente le PHP en 1993
pour ses propres besoins.

- Rapidement il ouvre le code de son langage selon le principe de l'« *open source* » (logiciel libre).
- Aujourd'hui PHP en est à la version 5.

Personal Home Page ?

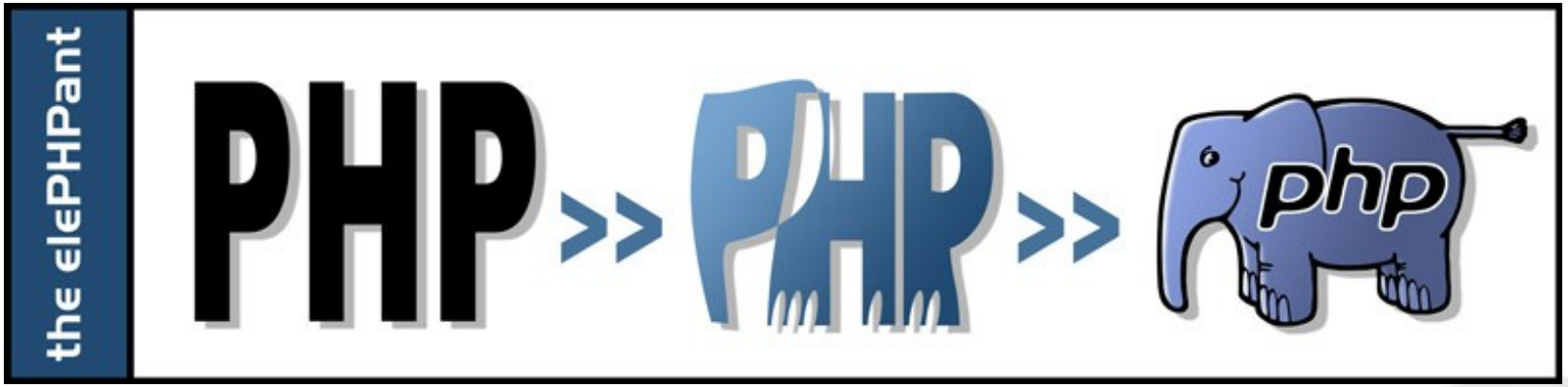
PHP signifie « PHP Hypertext Processor ».

Sigle récursif en clin d'oeil aux boucles infinies, cauchemar des informaticiens !



i.e. La vache qui rit...

De PHP à l'éléphant



Trois avantages majeurs

Primo : un langage de script facile

- A la fois lisible, robuste et presque rapide.

Secundo : un langage portable

- Installé comme module du serveur Apache !

Tertio : un logiciel libre

- Logiciel libre développé et maintenu par une centaine de développeurs bénévoles.

Les défauts

Un langage peu rigoureux :

- Pas de déclaration de variables ;
- Pas de typage des variables ;
- Pas de cohérence dans les noms ;
- Pas de notion de « persistance » ;
- Pas de programmation objet (POO) ;
- Culte de l'optimisation ridicule (echo/print).

Langage de script

Le code PHP s'insère sans difficulté dans une page HTML, entre deux balises entrante et fermante :

```
<?php
   ...
?>
```

Langage de programmation

Contrairement au HTML qui est un langage de description, le PHP est un vrai langage de programmation « orienté objet ».

- Un langage de description est aisément lisible, un langage de programmation nécessite un apprentissage !

Un module d'Apache

Le langage PHP implanté commet un module d'extension du logiciel Apache :

- Le plus réputé et le plus déployé des serveurs Web;
- 72,45% de parts de marché pour Apache en décembre 2008.

Licence GNU-GPL

La Licence publique générale GNU, abrégée « GPL », est une licence qui fixe les conditions légales de distribution des logiciels libres du projet GNU.

Richard Stallman et Eben Moglen, deux des grands acteurs de la Free Software Foundation, en furent les premiers rédacteurs. Sa dernière version est la GNU GPL version 3 publiée le 29 juin 2007.

Outils « éditeurs » de PHP

Pour rédiger du code PHP, si Textedit suffit, il est préférable d'utiliser un véritable éditeur :

Pour Mac OS

- Smultron (français, gratuit)
- JEdit (anglais, gratuit)
- TextWrangler (anglais, gratuit)
- Sublime Text (anglais, 70 €)
- BBEdit (anglais 50 €)

Outils « éditeurs » de PHP

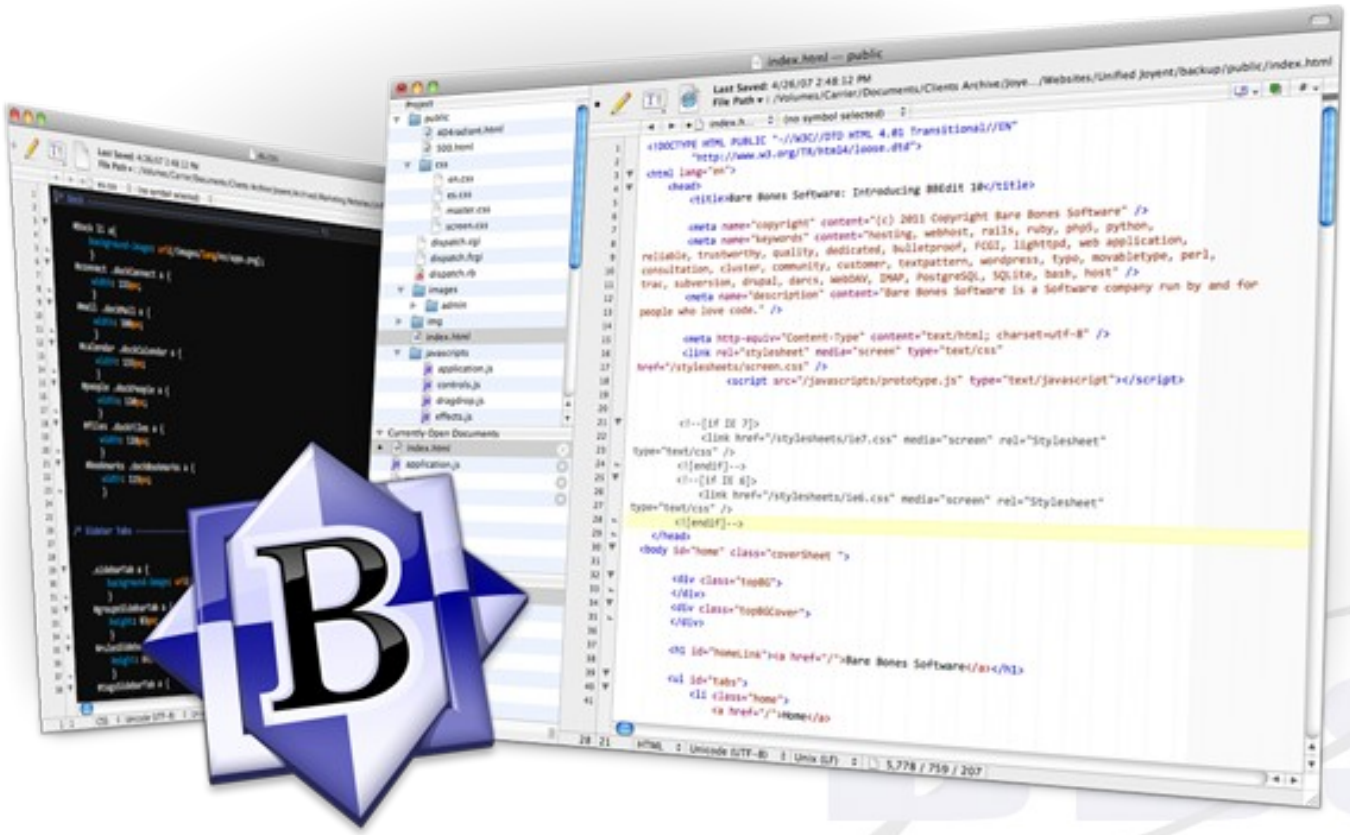
Sous Windows :

- NotePad ++
- JEdit
- Pspad
- ConTEXT

Sous Linux :

- Emacs
- gEdit
- Kate
- Vim
- jEdit

Outils « éditeurs » de PHP



Avant-propos
Variables
Opérateurs
Structures de contrôle
Boucles
Fonctions

Types de variables

Variables scalaires

Variables numériques (nombres entiers)

Variables numériques (virgule flottante)

Variables tabulaires (tableaux)

Variables booléennes

Variables alphanumériques

Constantes

Variables scalaires

Chaque variable est proprement nommée.

En PHP, toutes les variables scalaires comment par le signe « $ »

- Le second caractère doit être une lettre ;
- Le nom ne peut pas contenir d'espace (_) ;
- Le nom ne devrait pas contenir de diacritiques.

Variables scalaires

Une variable scalaire ne contient qu'une seule valeur :

- nombre entier INT
- nombre à virgule flottante FLOAT
- chaînes de caractères STRING
- booléennes FALSE / TRUE

Variables à nombres entiers

```
$a = 2;
$b=3;
$c = $a + $b;
$d="9";
$c-=1; ($c = $c - 1;)
$d++;  ($d = $d + 1;)
$x=$c+$d;
```

(le signe « = » a pour fonction d'affecter une valeur à une variable)

Variables à virgule flottante

$nombre_d_or = 1.61803399;

- Approximation sur le résultat en cas de résultat d'opération qui donne un nombre de chiffres infini après la virgule.
- ATTENTION ! La virgule est le point anglo-saxon « . »

Format de sortie des flottantes

```
$x=3.1416 ;
echo number_format($x,2)  →    3.14

$x=1.618 03399;
echo number_format($x,2)  →    1.62
```

Variables booléennes

```
$ok=TRUE; ($ok=true)
$ko=FALSE; ($ko=false)

if ($ok==TRUE) echo "OK";
if ($ok) echo "OK";
If (!$ko) echo "NOT OK";
```

Variables booléennes

```
$ok=false;
$ok=0;
$ok=0.0;
$ok="";
$ok="0";
$ok=NULL;
if (!$ok) echo "NOT OK";
```

Variables chaînes de caractères

```
$departement_ain = 1;
$departement_ain = "01";
$departement_ain = '01';
```

Les espaces y sont significatifs !

```
$chef_lieu =  "Bourg en Bresse";
```

Variables textuelles longues

Variable « HereDoc » *(IciDocumentation)*

```
$variable_longue= <<<DOC
  Exemple de chaîne
  s'étalant sur
  plusieurs lignes
  avec la syntaxe heredoc
DOC;
```

Caractères spéciaux

\n = saut de ligne

\t = tabulation

```
$Chaine = ''Mon nom est \n\tPersonne '' ;
echo $Chaine ;
```

Mon nom est
	Personne

Guillemets & apostrophes

Les variables chaînes acceptent les guillemets « " » comme tel si elles sont définies avec les apostrophes :

```
$fonte = "<font face='Arial'>";
$fonte = '<font face="Arial">'; HTML valide
```

Guillemets & apostrophes

Les variables chaînes acceptent les apostrophes « ' » comme tel si elles sont définies avec les guillemets :

```
$ville = "Villeneuve d'Ascq";
```

Guillemets & apostrophes

Les chaînes entre apostrophes sont affectées à une variable comme tel, les chaînes entre guillemets sont extrapolées :

```
$nom= ''Bruno'' ;
$variable_1 = '$nom' ;          -> $nom
$variable_2 = ''$nom'' ;        ->Bruno
```

Caractère d'échappement

Si une variable chaîne définie avec des guillemets doit contenir des guillemets, ou une variable définie avec des apostrophes des apostrophes, le caractère d'échappement « \ » doit être obligatoirement utilisé !

Caractère d'échappement

```
$fonte = "<font face="Arial">";
```
KO

```
$fonte = "<font face=\"Arial\">";
```
OK

```
$ville = 'Villeneuve d'Ascq';
```
KO

```
$ville = 'Villeneuve d\'Ascq';
```
OK

Avant-propos
Variables
Opérateurs
Structures de contrôle
Boucles
Fonctions

Extrapolation & variables

Une variable comprise dans une variable chaîne est toujours extrapolée :

```
$quantieme=20;
echo "<p>Le $quantieme si&egrave;cle</p>";
```

Extrapolation & variables

Une variable comprise dans une chaîne ET collée à un mot, doit être extrapolée à l'aide de des opérateurs « { » et « } » :

```
$quantieme=20;
echo "Le $quantiemee si&egrave;cle";     KO
echo "Le ${quantieme}e si&egrave;cle";   OK
```

Concaténation de chaînes

La concaténation de chaînes se fait à l'aide de l'opérateur « . »

```
$departement_ain = "01";
$chef_lieu = "Bourg en Bresse";
$espace=" ";
$titre=$departement_ain.$espace.$chef_lieu;
```

Extrapolation & constantes

Une constante comprise dans une variable chaîne n'est jamais extrapolée :

```
echo "Le nombre d'or est NOMBRE_D_OR";

echo "Le nombre d'or est ".NOMBRE_D_OR;
```

Variables dynamiques

La création d'une variable dynamique passe par la création initiale d'une variable qui donnera le nom de la variable dynamique :

```
$nom_dynamique="France";
$$nom_dynamique="33";

echo $France ; // affiche 33
```

Variables « tabulaires »

Un tableau est une variable tabulaire qui contient plusieurs valeurs, grâce à un système d'index :

- soit un index numérique;
- soit un index nominatif (tableau associatif, dit de « hachage »).

Tableaux numériques

Les tableaux numériques sont déclarés à l'aide de la commande « array » :

```
$departements = array("Ain","Aisne");
```

La commande « echo » ne permet pas d'afficher le contenu d'un tableau.La commande « print_r » doit être utilisée :

```
print_r($departements);
```

Tableaux numériques

Tout tableau numérique commence avec l'indice « 0 ».

- L'indice du dernier élément du tableau correspond donc au nombre d' éléments moins un.

Tableaux associatifs

Un tableau qui uilise un système d'indice nominatif est « de hachage » :

```
$departement['Nord']='59';
$departement['Pas_de_Calais']='62';
```

Clef et valeur

Dans les deux types de tableaux, l'indice est appelé « clef » et le contenu « valeur ».

```
$departement['Nord']="59";
$departements[0]="Côtes d'Armor";
```

Convertir un tableau en variables

Extraire un tableau en variables :

```
$tableau=array('rouge'=>'gueules','bleu'=>'azur') ;
extract($tableau) ;
echo ''Rouge en héraldique se dit : $rouge'' ;
echo ''Bleu en héraldique se dit : $bleu'' ;
```

Convertir des variables en tableau

Compacter des variables en un tableau :

```
$blason = "De geules plain";
$famille = "D'Albret";
$ville = "Nérac";
$localisation = array("famille","ville");
$result = compact("blason", "zglorb", $localisation);
print_r($result);
```

Tableaux multidimensionnels

Un tableau multidimensionnel est un tableau contenant un tableau !

```
$film[]="Gran Torino";
$film[]="Million Dollar Baby";
$film[]="Impitoyable";
$scenariste['Clint Eastwood']=$film;
print_r($scenariste['Clint Eastwood'][1]);
```

Les superglobales

$GLOBALS

$_POST[ ]

$_GET[ ]

$_COOKIE[ ]

$_SESSION[ ]

$_FILES[ ]

Constantes

Une constante est le contraire d'une variable.

Une fois définie, elle ne peut jamais être modifiée durant l'exécution du script *(sauf les constantes magiques)*.

Par défaut, le nom d'une constante est sensible à la casse.

Constantes

Elles n'utilisent pas le signe $, se définissent par la commande « define » et ne s'écrivent par convention qu'en MAJUSCULES !

```
define ('NOMBRE_D_OR', 1.61803399);
```

Les constantes magiques

__LINE__	Le n° de ligne courante.
__FILE__	Le chemin complet.
__DIR__	Le dossier du fichier (PHP 5.3.0.)
__FUNCTION__	Le nom de la fonction courante.
__CLASS__	Le nom de la classe courante.
__TRAIT__	(Ajouté en PHP 5.4.0)
__METHOD__	Le nom de la méthod courante.
__NAMESPACE__	L'espace de noms courant (5.3.0.)

Types d'opérateurs

Il existe plusieurs types d'opérateurs :

- opérateurs arithmétiques
- opérateurs d'affectation
- opérateurs de comparaison
- opérateurs unaires & ternaires
- opérateurs logiques
- opérateurs de chaînes de caractères
- opérateurs de contrôle d'erreur

Opérateurs arithmétiques

Les plus simples : + - * : %

```
$x=$a+$b;
$y=$a-$b;
$z=$a*$b;
$w=$a/$b;
$m=$a%$b; modulo de la division de a/b
```

Opérateurs arithmétiques

Ordre de priorité :

- ++, --, !
- *, /, %
- +, -,
- ==, <, >, <=, >=, !=
- &&, ||, and, or
- =

Opérateurs d'affection

Notations abrégées à connaître :

```
$a=1;
$a+=3;          =       $a = $a + 3;
$a-=2;          =       $a = $a -2;
$a%=3;
$jour= "vendredi ";
$quantieme="13";
$date=$jour.$quantieme;
```

Opérateurs unaires

Ils permettent d'incrémenter ou décrémenter une variable numérique :

$a++ ++$a $a-- --$a

La position de l'opérateur définit si l'incrémentations e situe avant ou après l'action de la fonction :

$x=0;	Déclaration de x
echo $x++;	Affiche 0 ! Mais $x vaut 1
echo ++$x;	Affiche 2 ! Incrémentation préalable

Opérateurs de comparaison

Utilisés avec la fonction « if »

$a == $b;	Vrai si a est égal à b
$a === $b;	Vrai si a et b sont égaux ET de même type
$a <> $b;	Vrai si a est différent de b
$a != $b;	Vrai si a est différent de b
$a !== $b;	Vrai si a et b sont inégaux OU de type différent
$a < $b;	Vrai si a est inférieur à b
$a > $b;	Vrai si a est supérieur à b
$a >= $b;	Vrai si a est supérieur ou égal à b

Erreur fréquente !

```
$x=112;
$y=38;
if ($x=$y) echo (''X et Y sont égaux'');
```

Trouvez la faille !

Exemple

Utilisé pour raccourcir les comparaisons :

```
if ($a==$b)
  {
  $c=1;
  }
else
  {
  $c=0;
  }
```

Opérateur ternaire

L'expression précédente peut s'écrire sous forme d'opérateur ternaire :

```
$a=3;
$b = $a == 3 ? 6 : 0;
echo $b; // $b = 6

$a==$b ? $c=1 : $c=0;
echo $c; // $c = 0
```

Opérateur ternaire en cascade

L'expression précédente peut s'écrire sous forme d'opérateur ternaire :

```
$f=false;
$g=false;
$h=true;
$x = $f==true ? 1 : $g==true ? 1 : $h==true ? 1 : 0;
echo "<br>",$x;
```

Opérateurs de chaînes

Il existe 2 opérateurs de chaînes de caractères : « . » et « .= »

```
$x=''bonjour '';
$y=''tout le monde'';
$i='' ! '';
$z=$x.$y;
echo ($z);
$z.=$i;
echo ($z);
```

Opérateur de contrôle d'erreur

Utilisé pour ne pas afficher les messages d'erreur de PHP.

- Le symbole « @ » à coller devant une fonction PHP que l'on veut rendre muette :

```
$requete = @mysql_db_query($bdd,$sql,$pointeur);
```

A ne pas utiliser avant la fin de la l'écriture du programme car les messages d'erreur aident au débogage !

Priorité des opérateurs

$a=8-8/8+10; $a = 17

La division a priorité sur l'addition et la division !

$b=8+7*10-1; $b = 77

La multiplication a priorité sur l'addition et la division !

$c=8/8-2*10/10+8; $c = ?

- http://php.net/manual/fr/language.operators.precedence.php

Priorité des opérateurs

$a=8-8/8+10; $a = 17

La division a priorité sur l'addition et la division !

$b=8+7*10-1; $b = 77

La multiplication a priorité sur l'addition et la division !

$c=8/8-2*10/10+8; $c = 7

- http://php.net/manual/fr/language.operators.precedence.php

Parenthèses & opérateurs

$a=8-8/8+10; $a = 17

$b=8-8/(8+10); $b = 7.5555555

Les parenthèses forcent l'ordre des priorités !

$b=8-8/18 ;

$c=8-(8/8+10); $c = ?

Parenthèses & opérateurs

$a=8-8/8+10; $a = 17

$b=8-8/(8+10); $b = 7.5555555

Les parenthèses forcent l'ordre des priorités !

$b=8-8/18 ;

$c=8-(8/8+10); $c = -3

Priorités des opérateurs

```
$d = 9+100/10*87-12;        $d = ?
$e = 9+(100/10*87)-12;      $e = ?
$f = 9+100/(10*87)-12;      $f = ?
$g = 9+(100/10)*(87-12);    $g = ?
$h = 9+(100/10)*(87)-12;    $h = ?
```

Priorités des opérateurs

$d = 9+100/10*87-12;	$d = 867
$e = 9+(100/10*87)-12;	$e = 867
$f = 9+100/(10*87)-12;	$f = -2.885
$g = 9+(100/10)*(87-12);	$g = 759
$h = 9+(100/10)*(87)-12;	$h = 867

Priorité des opérateurs

```
if ($a > $b AND $b > 0 OR $b>-5) echo ''ok'';
else echo ''ko'';
```

L'opérateur AND a priorité sur OR !

- http://php.net/manual/fr/language.operators.precedence.php

Avant-propos
Variables
Opérateurs
Structures de contrôle
Boucles
Fonctions

Exécution conditionnelle

Les structures de contrôle sont caractérisées par l'utilisation des symboles accolade ouvrantes et fermantes : {...}

Elles évaluent la validité d'une variable ou d'une expression multi-variables et décident de l'exécution ou non d'une partie du script.

if(){} … else{}

```
$x=10;
$y=100;
if ($x > $y)
   {
   echo 'x est strictement supérieur à y';
   }
else echo 'x est inférieur ou égal à y';
```

if(){} … else{}

```
$x=10;
$y=100;
if ($x > $y){
   echo 'x est strictement supérieur à y';
   }
else{
   echo 'x est inférieur ou égal à y';
   }
```

If(): else : endif ;

Syntaxe alternative :

```
$a=10;
$b=100;
if ($a > $b): echo 'a est supérieur à b';
else: echo 'a est égal ou inférieur à b';
endif;
```

switch

```
<?php
switch ($i) {
    case "mammifère":
        echo "porte ses petits en gestation";
        break;
    case "poisson":
        echo "vit dans l'eau";
        break;
    case "oiseau":
        echo " pond des oeufs";
        break;
}
?>
```

Attention !

L'instruction switch est exécutée ligne par ligne :

- lorsqu'un cas *(case)* est vérifié, PHP exécute alors les instructions correspondantes.
- Mais PHP continue d'exécuter les instructions des autres cas, jusqu'à la fin du bloc d'instructions du switch, ou bien dès qu'il trouve l'instruction de fin *(break)*.

Erreur fréquente

```
switch ($i)
    {
    case 0:
        echo "i égal 0";
    case 1:
        echo "i égal 1";
    case 2:
        echo "i égal 2";
    }
```

Trouvez la faille !

Cas spécial « *default* »

```
switch ($i) {
    case 0:
        echo "i égal 0";
        break;
    case 1:
        echo "i égal 1";
        break;
    case 2:
        echo "i égal 2";
        break;
    default:
       echo "i n'est ni égal à 2, ni à 1, ni à 0.";
}
```

Syntaxe alternative

```
switch ($i):
    case 0:
        echo "i égal 0";
        break;
    case 1:
        echo "i égal 1";
        break;
    case 2:
        echo "i égal 2";
        break;
    default:
        echo "i n'est ni égal à 2, ni à 1, ni à 0";
endswitch;
```

case : ou case ; ?

```
switch($beer)
    {
    case 'Kronembourg';
    case '1664';
        echo 'Mauvais choix';
        break;
    case 'Chimay';
    case 'Queue de Charrue';
    case 'Guinness';
        echo 'Bon choix';
       break;
    default;
        echo 'Merci de faire un choix...';
    break;
    }
```

Avant-propos
Variables
Opérateurs
Structures de contrôle
Boucles
Fonctions

Le coeur de la programmation

Le principe de la boucle est un principe de base en informatique :

Tous les algorithmes en utilisent le principe ; on peut même que la boucle est ce qui identifie un langage de programmation par rapport à un langage de description.

Boucle *while*

La signification d'une boucle while est très simple ; PHP exécute l'instruction tant que l'expression de la boucle *while* est évaluée comme TRUE :

```
$i = 1;
while ($i <= 10)
 {
 echo $i++;
 }
```

Syntaxe alternative

while et endwhile :

```
$i = 1;
while ($i <= 10):
    echo $i;
    $i++;
endwhile;
```

Boucle do... while

La boucle do … while est une variante peu usitée de la boucle *while* :

```
do
  {
  bloc d'instruction(s);
  }
while(condition);
```

- Elle peut être utilisée pour incrémenter ou décrémenter une limite AVANT une boucle.

Exemple

```
$i = 1;
do
    {
     echo $i . '<br />';
    $i++;
    }
while( $i < 5 );
```

Boucle for()

La boucle for (…) est sans doute la plus utilisée des boucles en PHP.

- Elle utilise un compteur (une variable) qui initialise, incrémente et contrôle l'exécution :

for (*initialisation ; limite ; incrémentation*)

{

bloc d'instruction(s);

}

Exemples

Exemple 1 & 2 :

```
for ($i = 1 ; $i <= 10 ; $i++) echo ($i);
for ($i = 1 ; $i <= 10 ; print $i, $i++);
```

Exemple 3 :

```
for ($i = 1 ; ; $i++)
    {
    if ($i > 10) break;
    echo $i;
    }
```

Exemples

Exemple 4 :

```
$i = 1;
For ( ; ; )
    {
    if ($i > 10)
        {
        break;
        }
    echo $i;
    $i++;
    }
```

Exemples

Exemple 5 :

Il est posible d'ajouter un indice dans les instructions de contrôle :

```
for ($i=1, $j=0 ; $i<=10 ; $j+=$i, $i++)
  {
  Echo ('<br/>');
  echo $i;
  echo $j;
  }
```

Boucle foreach()

Apparue avec PHP4, cette boucle a pour principale utilité de parcourir des tableaux (variables tabulaires) afin d'en extraire les indices et les valeurs :

```
foreach ($tableau as $indice => $valeur)
  {
  bloc d'instruction(s);
  }
```

Exemple

Exemple 1 :

```
$tableau = array(1, 2, 3, 4);
foreach ($tableau  as $valeur)
   {
    $resultat  = $valeur * 2;
    echo ($resultat);
   }
```

Exemple

Exemple 2 :

```
if(isset($_POST['envoi'])){
echo("<p>Hello, " . $_POST['name'] . ", formulaire
envoy&eacute;</p>");
}
else{
    ?>
    <form action="<?=$_SERVER['PHP_SELF']?>" method="POST">
    <input type="text" name="name"><br>
    <input type="submit" value="Envoi" name="envoi">
    </form>
    <?
    }
```

break & continue

Deux instructions de contrôle de boucle :

- « break » permet de sortit d'une boucle sur une condition, par exemple ;
- « continue » permet de court-circuiter une instruction de boucle en passant à l'itération suivante.

Avant-propos
Variables
Opérateurs
Structures de contrôle
Boucles
Fonctions

Nombreuses fonctions

Les fonctions de PHP sont nombreuses et rassemblées en famille selon leur portée :

- fonctions de variables & tableaux
- fonctions de chaînes de caractères
- fonctions de date
- fonctions de répertoires
- fonctions de fichiers

fonctions de BDD MySQL

Fonctions de variables

isset();

gettype(); settype();

is_numeric(); is_float(); [is_double();] is_int();

is_array(); is_bool(); is_scalar();

is_null();

empty();

unset();

Fonctions de tableaux

Comparaison de tableaux array_intersect()

```
$a=array('Lille','Roubaix','Tourcoing') ;
$b=array('Mouvaux','Roncq','Tourcoing') ;
$c=array('Tourcoing','Lille','Halluin') ;
$d=array_intersect($a, $b, $c) ;
print_r($d);
```

Array ([1] => Roubaix)

Fonctions de tableaux

Comparaison de tableaux array_diff()

```
$a=array('Lille','Roubaix','Tourcoing') ;
$b=array('Mouvaux','Roncq','Tourcoing') ;
$c=array('Wattrelos','Lille','Halluin') ;
$d=array_diff($a, $b, $c) ;
print_r($d);
```

Array ([1] => Roubaix)

Fonctions de tableaux

Parcourir un tableau :

```
$tableau=array('Lille','Roubaix','Tourcoing');
$x=current($tableau); // $x = 'Lille'
$next($tableau)
$x=current($tableau); // $x = 'Roubaix'
$end($tableau);
$x=current($tableau); // $x = 'Tourcoing'
```

Fonctions de tableaux

foreach(... as … => ...) parcours un tableau et en récupère le contenu :

```
foreach ($tableau as $indice => $valeur)
  {
  la variable $tableau est parcourue,
  la variable $indice est affectée de l'indice,
  et $valeur est affectée de la valeur.
  }
```

Fonctions de tableaux

Tris de tableau :

sort($tab) // tri PAR valeur & CHANGEMENT de clef
rsort($tab) // idem AVEC inversion

asort($tab) // tri PAR la valeur & clef inchangée
arsort($tab) // idem AVEC inversion

ksort($tab) // tri PAR CLEF & CHANGEMENT de clef
krsort($tab) // idem AVEC inversion

Fonctions de tableaux

Fusion de tableau : array_merge()

```
$a=array('Lille','Roubaix') ;
$b=array('Tourcoing','Wattrelos') ;
$c=array_merge($a,$b) ;
echo $c[0] ; // affiche Lille
echo $c[1] ; // affiche Roubaix
echo $c[2] ; // affiche Tourcoing
echo $c[3] ; // affiche Wattrelos
```

Fonctions de tableaux

Dissociation de tableau : array_slice()

```
$a=array('Lille','Roubaix','Tourcoing','Marcq');
$b=array_slice($a,1,2) ;
print_r($b) ;
```

Array ([0] => Roubaix [1] => Tourcoing)

Fonctions de tableaux

Fusion de tableau : array_merge()

```
$a=array('ville1'=>'Lille','ville2'=>'Tourocing')
;
$b=array('ville3'=>'Roncq','ville1'=>'Marcq') ;
$c=array_merge($a,$b) ;
echo $c['ville1'] ; // affiche Marcq
echo $c['ville2'] ; // affiche Tourcoing
echo $c['ville3'] ; // affiche Roncq
```

Fonctions de tableaux

list() affecte des variables depuis un tableau :

```
$tableau=array('Lille','Roubaix','Tourcoing');
list($a, $b, $c) = $tableau;
echo $a; // affiche Lille
echo $b; // affiche Roubaix
echo $c; // affiche Tourcoing
```

Fonctions de tableau

implode(*delimiteur, tableau*) permet de convertir un tableau en chaîne :

```
$tableau=array('Lille','Roubaix','Tourcoing') ;
$chaine=implode(', ', $tableau) ;
echo $chaine ; // Affiche Lille, Roubaix,Tourcoing
```

Fonctions de tableau

explode(*delimiteur, chaîne*) permet de convertir une chaîne en tableau :

```
$chaine= 'Lille,Roubaix,Tourcoing') ;
$tableau=explode(', ', $chaine) ;
var_dump($tableau);
```

```
[0] => Lille
[1] => Roubaix
[2] => Tourcoing
```

Fonctions de tableau

extract(*tableau*) permet de créer des variables depuis un tableau associatif :

```
$x=villes('Roubaix'=>94000,'Tourcoing'=>92000) ;
Extract($villes) ;
```

Équivalent à :

```
$Roubaix=94000 ;
$Tourcoing=92000 ;
```

Fonctions de tableau

array_sum() // addition si entier naturel

array_unique() // dé-doublonnage

array_flip() // intervertir clef / valeur

Fonctions de chaînes

Les plus utilisées :

- strlen($chaîne);
- trim($chaîne); ltrim($chaîne) ; rtrim($chaîne) ;
- substr($chaîne,i,j); substr_count($chaîne) ;
- addSlashes($chaîne) & stripSlashes($chaîne);
- str_replace($cherche,$remplace,$chaîne);

Autres fonctions de chaînes

- strtoupper($chaine); strtolower($chaîne);
- ucfirst($chaîne) ; ucwords($chaîne) ;
- str_repeat($chaîne,entier);
- strstr($chaîne,$valeur,TRUE) ;
- stristr($chaîne,$valeur,TRUE) ;

  strpos($chaîne,$argument) ; strrpos() ;
- strrev($chaîne) ;

Fonctions de date

Les ordinateurs sous UNIX dénombre le temps en comptant le nombre de secondes depuis le 1er janvier 1970 :

C'est la notion de « tampon horodateur » ou de « timestamp »

Fonctions de date

- time(); retourne le tampon horaire UNIX du moment *(timestamp)*

- mktime(s,m,h,j,m,a) retourne le tampon horaire d'une date passée en paramètre :

```
$jour = mktime(0, 0, 0, 3, 0, 2013);
```

Fonctions de date

- date(*format,timestamp*); renvoie selon de nombreux paramètres de format des données de datation sur le tampon passé en arguement à la date du moment :

```
echo date("j M Y"); // 26 Jan 2014
echo date("l j M Y"); // Sunday 26 Jan 2014
echo date("l j F Y"); // Sunday 26 January 2014
echo date("l j, F Y"); // Sunday 26, January 2014
```

Fonctions de date

- getdate($timestamp); renvoie un tableau associatif de la date passée en arguement, et s'il est vide, le tableau associatif du moment :

```
print_r(getdate());
```

```
Array ( [seconds] => 58 [minutes] => 44 [hours] => 16
[mday] => 26 [wday] => 0 [mon] => 1 [year] => 2014
[yday] => 25 [weekday] => Sunday [month] =>
January [0] => 1390751098 )
```

Fonctions de date

checkdate ($mois , $jour , $an) :

- Le mois doit être compris entre 1 et 12 ;
- Le jour doit être compris dans le mois * ;
- L'année maximum est 32767.

* Les années bissextiles sont prises en compte.

Fonctions de répertoire

```
$pointeur=opendir('.');
while ($donnees=readdir($pointeur))
  {
  Print (''$donnees.<br/>'');
  }
closedir($pointeur);
```

Fonctions de fichiers

```
include 'fichier.ini' ;
include_once 'fichier.ini' ;
require 'conf.php' ;
require_once 'conf.php' ;
```

Fonctions de fichier

chmod(); changer le statut d'un fichier.
file_exist();
stats();
lstats()
filesize(); retourne la taille du fichier.
filetype(); retourne le type de fichier;
fileatime(); retourne le THU d'accès au fichier.
filemtime(); retourne le THU de modification.

Fonctions de fichier

- copy(); copier un fichier.
- rename(); renommer/recopier un fichier.
- flock(*pointeur,verrou*); pose un verrou sur un pointeur.
- fopen(*fichier, mode*); pointeur sur un fichier.
- fclose(*pointeur*); ferme le pointeur.
- ftruncate(*pointeur,taille*);
- unlink(*pointeur*); détruit un fichier.

Fonctions de fichier

fread(*pointeur,taille*); lit le contenu du fichier.

fwrite(*pointeur,contenu*); écrire dans un fichier.

fputs(); idem.

fgetc(); retourne un caractère.

fgets(); retourne une ligne du pointeur de fichier.

Fonctions déclarées

```
$nom='simon' ;
$prenom='bruno' ;
echo formatage_nom($prenom,$nom) ;

function formatage_nom($a,$b)
    {
    return ucfirst($a).' '.strtoupper($b) ;
    }
```

Fonctions déclarées

```
$prix_ht=115 ;
$tva=20 ;
echo 'Prix TTC : ', calcul_tva($prix_ht, $tva) ;


function calcul_tva($x, $y)
    {
    return $x + $x * $y / 100;
    }
```

Passage par valeur

Fonctions déclarées

```
$prix=115 ;
calcul_tva_20($prix) ;
echo $prix // affiche 138

function calcul_tva_20(&$x)
    {
    $x+=$x*20/100;
    }
```

Passage par référence

Fonctions MySQL

Voir le diaporama PHP-MySQL

Fin du module

Bruno B. SIMON

www.bbs-consultant.net

contact@bbs-consultant.com