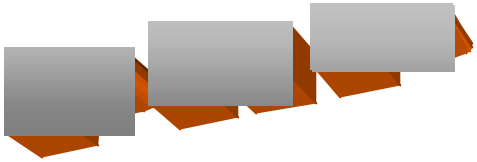


Cours PHP

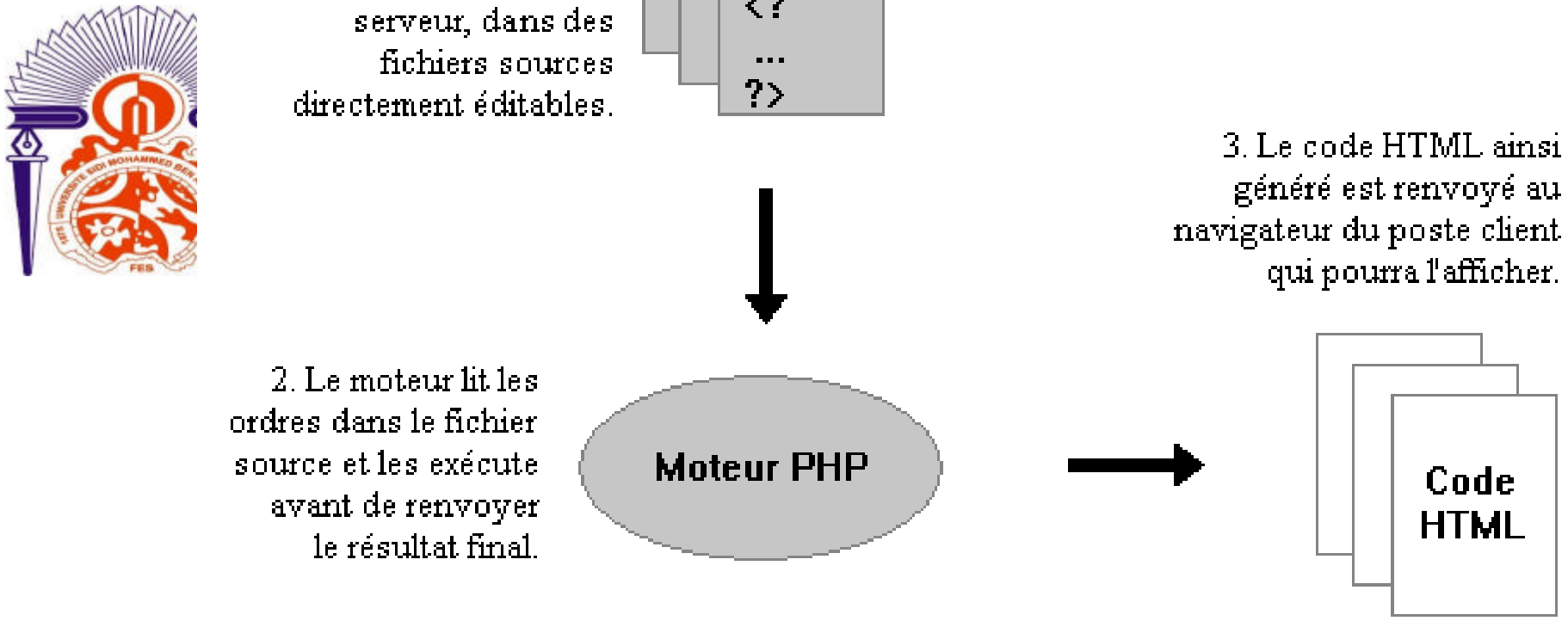
Introduction

- Le PHP permet de réaliser des sites Internet élaborés.
- Le HTML simple ne suffit pas à répondre aux impératifs imposés par les dernières évolutions du Web.
- PHP est un disponible dans plusieurs environnements, tels qu'Unix (Linux, AIX), Windows (95, 98, NT) et Macintosh.
- PHP est un langage de programmation spécialisé dans la génération de code, dont le langage de prédilection est le HTML.
- Il possède une impressionnante quantité d'outils (manipulation d'images, traitement de fichiers, accès aux bases de données, etc.)

La langage

- Le PHP est un langage complet, écrit en C, qui reprend une grande partie des spécificités techniques et sémantiques de ce langage.
- Le moteur d'interprétation du langage lit un fichier source PHP, puis génère un flux destination, en respectant les définitions et règles suivantes :
  - Un bloc PHP est un groupe continu de lignes, encadré par deux balises : <? et ?> ou <?ph et php?>
  - Toute ligne située à l'extérieur de ces balises n'est pas interprétée et est envoyée telle quelle dans le flux de sortie.
  - Toute ligne située à l'intérieur de ces balises est considérée comme une instruction PHP et est donc interprétée par le moteur.
  - Les instructions PHP n'apparaissent pas dans le résultat généré.
  - Lorsqu'une erreur survient, un message est intégré dans le flux de sortie, et la génération du script est interrompue.

Schéma de fonctionnement général



Premier programme

- La conception du script PHP est réalisé avec un éditeur quelconque, Emacs (hé hé hé !) est très bien pour cela.

```
<HTML>
<HEAD><TITLE>Mon premier programme</TITLE></HEAD>
<BODY>
<? echo "Ca marche !"; ?>
</BODY>
</HTML>
```

- L'extension de votre fichier doit être .php ou .php3 afin d 'être interprété par le serveur (Apache par exemple)

Les variables

- Contrairement à beaucoup de langages, PHP ne contient de partie déclarative clairement définie. Pour déclarer une variable, il suffit de l'initialiser. Celle-ci sera immédiatement accessible, et le restera jusqu'à la fin du script.
- Les variables en PHP sont toutes dotés du préfixe $

exemple :

$toto = 1;

- Le type d'une variable est également défini par la valeur qui lui à été affecté lors de sa création. Il existe5 types de données :
  - Entier (int, integer)
  - Décimal (real, double, float)
  - Chaîne de caractères (string)
  - Tableau (array)
  - Objet (object)

Les constantes

- Il est inutile de revenir sur l'intérêt des constantes. Il existe deux constantes prédéfinis en PHP.
- Pour définir une constante il suffit d'utiliser l'instruction
- *define*

exemple :

```
define(“MACONSTANTE”, “Hello World”) ;
if (defined(‘’MACONSTANTE’’)) {
   print ‘’La valeur de ma constante est : ‘’.MACONSTANTE //Attention, on accède au contenu sans le $.
}
```

Portée des variables

- Selon l'endroit du code où les variables sont définies, ces dernières auront une portée plus ou moins grande, c-à-d qu'elles seront définies soit pour une parie du code (fonction) soit pour sa totalité.
- Il existe trois niveaux de définition de variables :
  - Le niveau global. Il définit des variables dans l'intégralité du code d'une page PHP.
  - Le niveau local. Il définit des variables propres à une fonction, dont la durée de vie ne dépasse pas le temps de cette fonction.
  - Le niveau static. Il définit des variables propres à une fonction, qui persistent pendant l'intégralité du code de la page PHP.

Les tableaux

- Dans PHP, comme dans tout autre langage, les tableaux sont une structure de données incontournable.
- La déclaration d'un tableau se fait de la même manière que la déclaration d'une variable avec un indice se trouvant entre [et].

Les tableaux

- *$tableau[0] = 1; //* on crée un tableau, et sa première valeur est 1
- Pour un tableau à deux dimensions, il suffit de mettre un second indice au moment de l'affectation.

exemple :

- *$tableau[0][0] = 1; //* on crée un tableau, et sa première valeur est 1
- Il n'est pas obligatoire de préciser l'indice pour affecter une valeur.

exemple :

- *$tableau[] = 1; // équivaut à $tableau[0] = 1;*
- *$tableau[] = 45; // équivaut à $tableau[1] = 45;*
- *$tableau[] = 6; // équivaut à $tableau[2] = 6;*
- La déclaration et l'initialisation d'un tableau peuvent également se faire par l'intermédiaire de la fonction 'array()'. Cette fonction permet de préciser les indices ainsi que les valeurs du tableau (à l'aide de l'opérateur =>).

Les tableaux

Exemple :

- *$tableau[ ] = array(0=>1, 1=>45, 2=>6);*
- *$tableau[ ] = array("rouge"=>"red", "vert"=>"green", "bleu"=>"blue");*
- La navigation dans les éléments du tableau s'effectue à l'aide des fonctions 'next()' , 'prev()' et 'each()'.
- Le nombre d'éléments d'un tableau peut être obtenu à l'aide de la fonction 'count()'. Le tri des tableaux est facilité par de nombreuses fonctions : asort(), ksort(), sort(), usort(), etc...

Variables d'environnement

- L'un des aspects fondamentaux d'une application construite sur une architecture de type intranet est l'utilisation des variables d'environnement du serveur, et notamment celles du serveur HTTP.
- Avec PHP toutes les variables d'environnement du serveur sont automatiquement reprises dans les scripts PHP en tant que variables globales. Ainsi il suffit de les utiliser directement dans le code.

Variables d'environnement

exemple :
Tableau $_SERVER /HTTP_SERVER_VARS

*$IP=$_SERVER[‘**REMOTE_ADDR**’]*
echo $IP; @ IP de la machine cliente

Exemples de variables d'environnement

- **SERVER_NAME** : Le nom du serveur hôte qui exécute le script suivant. Si le script est exécuté sur un hôte virtuel, ce sera la valeur définie pour cet hôte virtuel.
- **PHP_SELF :** Le nom du fichier du script en cour d'exécution, par rapport au document root.
- **DOCUMENT_ROOT** : La racine sous laquelle le script courant est exécuté, comme défini dans la configuration du serveur.
- **REMOTE_ADDR** : L'adresse IP du client qui demande la page courante.
- **REMOTE_PORT** :Le port utilisé par la machine cliente pour communiquer avec le serveur web.
- **SCRIPT_FILENAME** : Le chemin absolu jusqu'au script courant.
- **SERVER_PORT** : Le port de la machine serveur utilisé pour les communications. Par défaut, c'est '80'. En utilisant SSL, par exemple, il sera remplacé par le numéro de port HTTP sécurisé.
- **REQUEST_URI** : L'URI qui a été fourni pour accéder à cette page. Par exemple : '/index.html'.

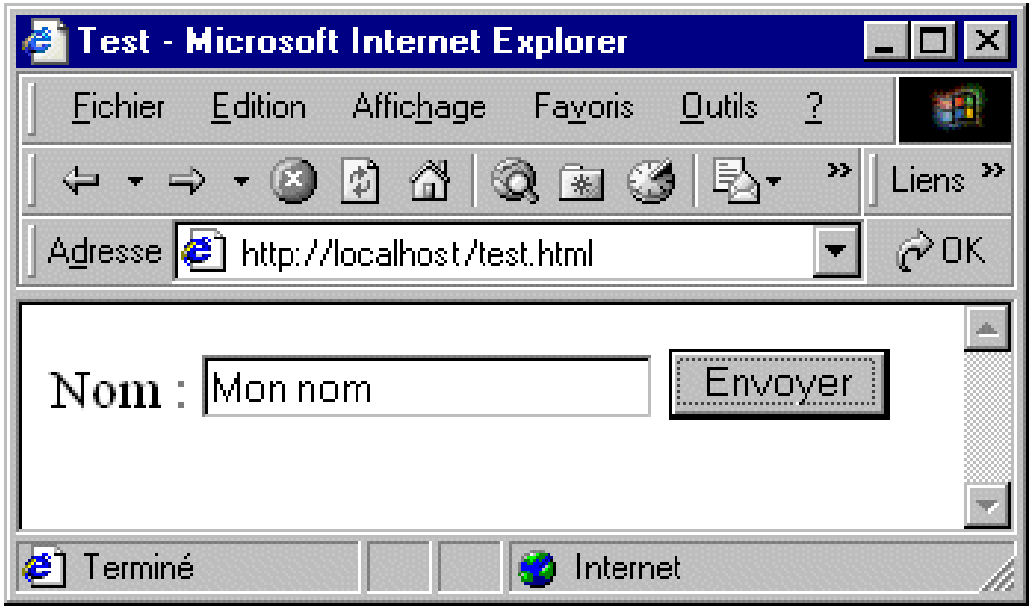
Variables issues de formulaires

- Les variables sont issues de formulaires HTML, elles correspondent aux différents champs positionnés entre les balises <FORM> et </FORM> de ce formulaire. La page qui reçoit ces variables est celle qui est désignée par l'attribut ACTION de la balise <FORM>.

exemple (fichier *test.html*) :

```
<HTML>
<FORM ACTION="test.php" METHOD=POST>
<INPUT TYPE=hidden NAME="toto" VALUE="bonjour">
Nom : <INPUT TYPE=text NAME="nom">
<INPUT TYPE=submit VALUE="Envoyer">
</FORM>
</HTML>
```

Variables issues de formulaires



Variables issues de formulaires

- Tableaux :
  $HTTP_GET_VARS/$_GET
  $HTTP_POST_VARS/$_POST

- Exemple

```
$nom = $_POST['toto']
echo $nom
```

Opérateurs Logiques

Ils permettent de combiner plusieurs tests entre eux.

exemple : Ici $a et $b peuvent prendre les valeurs booléennes vrai ou faux.

Opérateur	Exemple	Résultat
and (&&) $a && $b	$a and $b	vrai si $a et $b sont vrai tous les deux
or (||) $a || $b	$a or $b	vrai si $a est vrai ou $b est vrai, ou encore si $a et $b sont vrai tous les deux
not (!)	not $a	vrai si $a est faux !$a

Opérateurs de Comparaison

Ils permettent de comparer les valeurs de deux variables.

exemple : Ici $a et $b sont du même type de variable.

Opérateur	Exemple	Résultat
==	$a == $b	vrai si $a est égal à $b
!=	$a != $b	vrai si $a est différent de $b
<	$a $lt; $b	vrai si $a est inférieur $b
>	$a > $b	vrai si $a est supérieur $b
<=	$a <= $b	vrai si $a est inférieur ou égal à $b
>=	$a >= $b	vrai si $a est supérieur ou égal à $b

Tests : if... then... else

Test de base que l'on trouve dans la majeure partie des langages. Si une condition est vrai alors on exécute des instructions sinon (facultatif) on en exécute d'autre. On peut changer de condition avec un 'elseif'.

syntaxe :

```
if ( condition1 ) {
   Action 1
} elseif ( condition2 ) {
   Action 2
   }
} else {
Action 3
}
```

Tests : if... then... else

exemple :

```
if ($a==$b) {
   echo "A est égal à B";
} elseif ($a > $b) {
   echo "A est supérieur à B";
} else {
   echo "A est inférieur à B";
}
```

Tests : switch... case... default

Si les conditions successives ne portent que sur la valeur d'une variable, on pourra avantageusement remplacer le test 'if... elseif... else' par 'switch'. Dans ce test, la condition est associée à la valeur d'une variable, de plus l'instruction 'break' est primordiale à la fin de chaque bloc de conditions, sinon toutes les conditions seront vérifiées et exécutées.

exemple **:**

```
switch ($a) {
  case $b:
    echo "A est égal à B";
    break;
  case >$b:
    echo "A est supérieur à B";
    break;
  default:
    echo "A est inférieur à B";
    break;
}
```

Boucles

```
while ( condition ) {
   Action;
}

do {
   Action;
} while (condition);

for (expr1; expr2; expr3) {
   Action;
}
```

Boucles

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

$i=1;
do {
   echo "- $i -";
   $i++;
} while ($i <= 10)

for ($i = 1; $i <= 10; $i++) {
   echo "- $i -";
}
```

Break;Continue

Break :Cette instruction permet de sortir de n'importe quelle boucle, à n'importe quel moment.

Continue : Cette instruction permet de ne pas exécuter le code contenu dans la boucle et de passer à l'itération suivante.

```
for ($i=1; $i<=10; $i++) {
   if ($i<=5) { echo $i; }
   else {
     break;
   }
 echo "- ";
}
// cette boucle affichera : 1 - 2 - 3 - 4 - 5
```

Fonctions each() et list()

Ces fonctions sont aussi étroitement liées aux boucles.

each(..)

Cette fonction permet de parcourir tous les éléments d'un tableau sans se soucier de ses bornes. Elle retourne la combinaison clé-valeur courante du tableau passé en paramètre, puis se positionne sur l'élément suivant, et cela du premier au dernier indice. Lorsque la fin du tableau est atteinte, each( ) retourne la valeur faux (false).

list(..)

Cette fonction est très souvent associée à la fonction each(), elle permet d'affecter les éléments du tableau dans des valeurs distinctes.

Fonctions each() et list()

exemple :

```
$tableau = array("val1","val2","val3","val4");    //on crée un tableau avec 4 valeurs
while ($var = each($tableau)) {
   echo "$var[0] : $var[1]";
}
```

Ce code produit le résultat suivant :

0 : val1

1 : val2

2 : val3

3 : val4

ici l'indice est affecté au premier élément de $var et la valeur au deuxième élément $var

Fonctions *each() et list()*

```
$tableau = array("val1","val2","val3","val4");    //on crée un
    tableau avec 4 valeurs
while (list($cle, $valeur)= each($tableau)) {
  echo "$cle : $valeur";
}
```

Les Fonctions

Comme dans tout langage structuré, en PHP, les fonctions sont la base d'une programmation claire et efficace. Une fonction est une sorte de sous-programme isolé du reste du code, exécutable à tout moment, depuis n'importe quelle partie du code principal ou n'importe quelle autre fonction, par simple appel. De plus, les avantages des fonctions sont :

La non répétition de la même séquence de code

De cet avantage découlent :

- Le gain de productivité.
- La meilleure lisibilité du code.
- La maintenance facilitée.

Déclaration, paramètres, valeurs de retour

Une fonction est une séquence de code qui peut remplir n'importe quelle tâche. Tout code valide peut figurer dans le corps d'une fonction.

Déclaration :

La syntaxe de déclaration s'appuie sur le mot clé *function*. Ce mot clé est immédiatement suivi du nom de la fonction, puis de parenthèses (obligatoires) destinées à accueillir les éventuels paramètres à passer à

la fonction.

Déclaration, paramètres, valeurs de retour

exemple :

```
function bonjour( ) {
   echo « Bonjour ! » ;
}
```

Cette fonction ne fait qu'afficher *'bonjour'* et ne retourne aucun résultat, on l'utilisera de la manière suivante :

```
Bonjour( ) ; // affiche 'bonjour' à l'écran
```

Pour que la fonction retourne un résultat, on utilisera le mot clé return

```
function bonjour2( ) {
   return « Bonjour !» ;
}
echo bonjour2( ) ;
```

bonjour affiche le résultat elle-même, alors qu'il faut afficher le résultat de bonjour2 pour obtenir une action similaire.

Déclaration, paramètres, valeurs de retour

```
function dire_texte($qui, $texte = 'Bonjour') {
   if(empty($qui)){ // $qui est vide, on retourne faux
   return false;
   }else{
      echo "$texte $qui"; // on affiche le texte
      return true; // fonction exécutée avec succès
      }
}
```

Déclaration, paramètres, valeurs de retour

Passage par valeur par référence :

Le passage des paramètres tel qu'on l'a vu précédemment est ce que l'on appelle le passage par valeur. Il existe une autre manière de procéder :le passage par référence. On passe à la fonction la référence (adresse mémoire) d'une variable existante, et la fonction modifie directement la valeur de cette variable.

exemple :

```
function bonjour(&$phrase) {
$phrase= « bonjour Toto Dupont » ;
}
$chaine = « Phrase qui va disparaître » ;
bonjour($chaine) ;
echo $chaine ; // affiche 'bonjour Toto Dupont' à l'écran
```

Gestion de la date

- Fonction Date : nombreux paramètres

```
echo date("d-m-Y"); // affiche : "12-12-2000"
```

Gestion des fichiers

La fonction de base est la fonction *fopen( ).* C'est elle qui permet d'ouvrir un fichier, que ce soit pour le lire, le créer, ou y écrire. Sa syntaxe est :

entier fopen(chaine nomdufichier, chaine mode);

Différents modes disponibles

r : ouverture en lecture seulement

w : ouverture en écriture seulement (la fonction crée le fichier s'il n'existe pas)

a : ouverture en écriture seulement avec ajout du contenu à la fin du fichier (la fonction crée le fichier s'il n'existe pas)

r+ : ouverture en lecture et écriture

w+ : ouverture en lecture et écriture (la fonction crée le fichier s'il n'existe pas)

a+ : ouverture en lecture et écriture avec ajout du contenu à la fin du fichier (la fonction crée le fichier s'il n'existe pas).

Gestion des fichiers

Exemples :

```
$fp = fopen("../fichier.txt","r"); //lecture
$fp = fopen("ftp://localhost/pub/fichier.txt","w");
        //écriture depuis début du fichier
$fp = fopen("http://www.test.com/fichier.txt","a");
        //écriture depuis fin du fichier
```

Lecture dans un fichier

```
< ?
$monfichier = fopen(''monfichier.txt'', ''r'') ; // ouverture en lecture
if ( !($monfichier)) {
  print(« Impossible d'ouvrir le fichier ») ;
  exit ;
}
while ( !feof($monfichier) ) {
  $ligne = fgets($monfichier,255); // 255 caractères max. ou bien fin de ligne.
  print '' $ligne <BR> '' ;
}
fclose ($monfichier) ;
?>
```

Écriture dans un fichier

```
< ?
$monFichier = fopen(‘’monfichier.txt’’,’’ w’’) ; // ouverture en écriture
if ( !($monfichier)) {
  print(‘’ Impossible de créer le fichier \n’’) ;
  exit ;
}

fputs($monfichier, ‘’ligne 1’’) ; // on écrit deux lignes
fputs($monfichier, ‘’ligne 2’’) ;

fclose($monfichier) ; // on ferme le fichier, on libère les ressources

?>
```

Fonction Explode

- Il est fréquent d'avoir des fichiers contenant différents champs séparés par un délimiteur quelconque. Une fonction très utile dans ces cas là est la fonction *explode*. Sa syntaxe est la suivante

  explode (« caractère délimiteur », chaîne de donnée)

- *Roose | Philippe | 27*
- *Goudin | David | 28*

Fonction Explode

```
<html> <head> <title>Exemples de Explode</title> </head> <body>
<?php
if (!file_exists(« test.txt »)) {
  print "<H3><BR>Erreur, fichier compteur manquant<BR>";
  exit;
} else {
  $fd = fopen($fic,"r");
   while (!feof($fd)) {
    $ligne = fgets($fd,255);
    $tab=explode("|",$ligne);
    print "Nom : $tab[0]<br>";
    print "Prénom : $tab[1]<br>";
    print "Age : $tab[2]<br>";
}
   fclose($fd);
}
php?>
```

Fonctions Mathématiques

- Elles y sont toutes !

Abs, cos, sin, tan, sqrt, exp, ...
pi( )

Chaînes de caractères

- ***strpos (chaîne, sous chaîne)*** : retourne la position de la sous chaîne dans la chaîne. Dans le cas où la chaîne existe en plusieurs exemplaires, c'est la position de la première occurrence qui est retournée. *strrpos* retourne quand à elle la position de la dernière occurrence.
- ***strstr (chaîne, sous chaîne)*** retourne la portion de la chaîne à partir de la première occurrence de la sous chaîne.
- ***foreach (nom tableau)*** : A chaque appel, cette fonction retourne la valeur suivante du tableau.
- ***strlen (chaîne)*** : retourne la taille de la chaîne.
- ***strtolower|strtoupper (chaîne)*** : retourne la chaîne passée en paramètres an minuscules (resp. majuscules).
- ***str_replace (car d'origine, car de destination, chaîne)*** : remplace le caractère d'origine par le caractère de destination dans la chaîne.
- *g* supprime les caractères invisibles (espaces,\n, …) au début et à la fin de la chaîne.
- ***ereg(chaîne à chercher, chaîne)*** : retourne vrai si la chaîne à chercher (sous forme de chaîne ou sous forme d'expression régulière) est contenue dans chaîne.

Mail

- Il existe une méthode PHP permettant d'envoyer un mail directement, sans appeler un quelconque gestionnaire de courrier.
- La fonction *mail* (ou *email* parfois) permet de réaliser cela. Elle nécessite au moins trois paramètres :
  - ❑ Le destinataire,
  - ❑ L'objet du message,
  - ❑ Le corps du message.

```
< ?
mail(''dupont@mondomaine.fr'', ''Test de la commande mail'', ''Voici le
    corps du mail'') ;
?>
```

- Enverra un mail à *dupont@mondomaine.fr* avec comme sujet de mail « Test de la commande mail », et comme corps du mail : « Voici le corps du mail ».

Bases de Données

```
<?
   $bdd= "mabase"; // Base de données
   $host= "sql.estfes.fr";
   $user= "root";  // Utilisateur
   $pass= "12345";

   mysql_connect($host,$user,$pass) or die ("Impossible
    de se connecter à la base de données");
   mysql_select_db($bdd);
?>
```

Requêtes SQL

```
$query =
    "SELECT num, pays, date, circuit FROM $nomtable ";
$result= mysql_query($query);

if (mysql_error()) {
    print "Erreur dans la base de données :".mysql_error(); exit();
}
```

Récupération résultats

- La fonction *mysql_fetch_row( )* retourne un tableau qui représente tous les champs d'une rangée de résultat (un tuple). Chaque appel produit le tuple jusqu'à ce qu'il n'y en ai plus.
- Il existe d'autres fonctions pour réaliser cela, mais celle-ci est la plus rapide pour obtenir des résultats à partir d'une requête.

Récupération résultats

```
while ($row=mysql_fetch_row($result)) // $result a été
    obtenu par le msql_query précédent.
    {
    // récupération des informations
    $num = mysql_result($rowt[0]);
    $pays = mysql_result($row[1]);
    $date = mysql_result($row[2]);
    $circuit = mysql_result($row[3]);

    print '' ... '' ;
}
```

Exemple de script

```
<html> <head> <title>Page perso</title> </head> <body>
je mets ici tout le code nécessaire avant d 'afficher le compteur.
<?php
$fic="cpt.txt";
if (!file_exists($fic)) {
   print "<H3><BR>Erreur, fichier compteur manquant<BR>";
   exit;
}
else
   $fd = fopen($fic,"w+");
   $hits = fgets($fd,10);
   $hits++;
   fseek($fd,0);
   fputs($fd,$hits);
   fclose($fd);
   print "<bold>$hits</bold>";
php?>
```

Formulaire/Mail

```
<html> <head> <title>Formulaire/Mail</title></head><body>

<FORM method="POST" action="mail.php3">
<P> Adresse <INPUT TYPE=test NAME=adr_mail VALUE=""
    SIZE=50 MAXLENGTH=50 </P>
<input type="submit" value= »Mail">
</FORM>
```

PHP et les Images

- Formats supportés
  JPG, GIF, PNG
- Fonctionnalités
  Récupération,
  Création,
  Modification,
  Manipulation,

Fonctions liées aux images (1)

- *Id = getimagesize (nom de fichier)*
  hauteur, largeur, type, balise HTML
- *id = imagecreate(largeur, hauteur)*
- *imagedestoy(identifiant image)*
- *imagefill* (image, x, y, couleur)
- *imagechar($image, police de car., x, y, " car ", couleur)*
- *imagecharup* ...

Fonctions liées aux images (2)

- *imagestring et imagestringup*
- *imagecolorat(image, x, y)*
- *imagecolorstotal($image)*
- *imagecolortransparent(entier image, entier couleur)*

Fonctions liées aux images (3)

- *imagecreatefrom[jpeg|gif|png](nom du fichier)*
- *imagedashedline(image, x début, y début, x fin, y fin, couleur)*
- *imagefilledpolygon(image, tableau de points, nombre de points, couleur)*
- *imagefilledrectangle($image, x haut gauche, y haut gauche, x bas droite, y bas droite, couleur)*

Fonctions liées aux images (4)

- *imageloadfont(nom de la police de caractère)*
- fonction *imagesetpixel(image, x, y, couleur)*
- *imagesx(image)* et *imagesy(image)*
- *image[jpg|gif|png] (id image) -> le navigateur.*

Programmation OO

```
classe MaClasse {
   $attribut1 ;
  $attribut2 ;

  function maMethode1( ) {
    ...
}
  function maMethode2($param1, $param2= " je suis une chaîne ")
     { // paramètre par défaut...
...
}
} // fin de la déclaration de la classe.
```

```
<?php
  class classBD {
  var $base; var $host; var $user; var $passwd;var $request;
  var $result;var $tuple;

  function classBD($hote, $utilisateur, $mp) {   // Initialisations
     $this->host = $hote;
     $this->user = $utilisateur;   $this->passwd = $mp;
     // Connexion   @mysql_connect($this->host,$this->user,$this-
    >mp) or die(  "Impossible de se connecter à la base de
    données"); }function selectBD($nombase) {   $this-
    >base=$nombase;   @mysql_select_db($this->base); }

... function close() {
      MYSQL_CLOSE(); }

}?>
```

```
<?php
    include("./classBD.inc");
?>
<HTML><TITLE>Utilisation de classes en PHP </TITLE>
<BODY>Exemple d'accès à une BD sous mySQL <BR>
<?php$o = new classBD();
...
?>
</BODY></HTML>
```

PHP3 vs PHP4

- PHP/FI (Personal Home Page / Form Interpreter) fût la première version officielle de PHP (tout est parti d'une librairie Perl à l'origine).
- PHP3 résulte d'une réécriture complète de PHP/FI.
- PHP3 et PHP/FI ne sont plus officiellement supportés. Néanmoins, ils persistent encore en particulier chez certains hébergeurs.
- Quant à PHP4, il résulte à son tour d'une réécriture de PHP3 et utilise le moteur Zend. C'est la branche courante de PHP.

PHP3 vs PHP4 : nouveautés

- PHP4 : ré-écriture complète du moteur d 'interprétation. D 'une interprétation ligne à ligne, on passe à une compilation du code : souhaitable pour de gros développements.
- Auteurs : Zeev Zuraski, Andi Gutmans.

PHP3 vs PHP4

- Meilleure gestion du ‘ Garbage Collector ’
- Gestion des sessions : évite de passer par des variables partagées et/ou des cookies. Elles sont stockées dans un SGBD-R et/ou dans un fichier ASCII.
- « Bufferisation » des sorties.
- Comparaison valeur à valeur & type à type : ===
- ajout d ’un type Booléen

PHP3 vs PHP4

- Gestion des COM sous Windows

exemple tiré de www.phpheaven.net

```
<?php
$word = new COM( "word.application") or die( "Impossible
   to instantiate WordApp");
print  "Word is running, version {$word->Version}\n<br>";
$word->Visible=1;
$word->Documents->Add();
$word->Selection->TypeText( "This is a test...");
$word->Documents[1]->SaveAs( "test_com_php.doc");
$word->Quit();
?>
```

PHP3 vs PHP4

- Facilité de production d ’HTML

$maVariable = « PHP4 »

print <<< mylabel

<h1>Coucou</h1>

J ’utilise une autre technique pour faire du HTML

Grâce à

$maVariable et l ’utilisation d ’étiquettes.

mylabel;

PHP vs ASP

- En premier lieu, PHP et les ASP sont très proches dans la philosophie et la mise en oeuvre (code directement intégré aux pages HTML).
- Comparé à PHP3, les ASP se montrent parfois plus rapides et implémentent un concept très utile, celui des sessions. Les sessions ne sont pas supportées en natif par PHP3, même si de multiples solutions existent (PHPLib : http://phplib.netuse.de/).
- Comparé à PHP4, ces différences s'estompent. Ce dernier gagne en rapidité et introduit à son tour le concept de sessions.
- Notons aussi que les ASP sont largement inféodés au monde Win32, même si des solutions Unix existent (ChiliSoft : http://www.chilisoft.com/)
- PHP quant à lui est disponible sous NT comme sous Unix.
- Enfin, précisons que PHP dispose d'un plus grand nombre de fonctions, s'interface aisément avec des librairies et outils externes. En particulier, il est en mesure de s'interfacer, le plus souvent en natif, avec l'ensemble des SGBD standards du marché. Les ASP ne proposent qu'une approche ODBC, bien moins performante.

PHP en chiffres

- PHP = 500 000 développeurs à travers le monde
- Un marché de 4,8 milliards d'€.
- Le langage (en fait la plate-forme) est utilisée par plus de 1/3 des sites web dans le monde (46 % en France), ce qui représente 17 millions de domaines. Parmi les entreprises du CAC40, 78% en 2003 et 87% en 2004 utilisent le PHP.
- Interopérabilité : PHP peut instancier des objets COM, .NET mais également Java.
- PHP peut se connecter à l'ensemble des ténors des SGBD, mais aussi aux annuaires (LDAP), à Lotus Notes, à SAP, permet l'utilisation des services web (SOAP/WSDL), gère le XML, s 'interface avec les systèmes de paiement en ligne (VeriSign, Cybercash, Crédit Mutuel, etc.), génère du Flash (extension Ming), du PDF (classe FPDF).
- Robustesse (performances/fiabilités) : 90% des sites web les plus fréquentés dans l'hexagone avec des sites dépassant les 500 000 connexions/jour. Une des plus forte plate-forme au monde supportant PHP permet 150 000 utilisateurs simultanément (avec 220 serveurs en cluster).