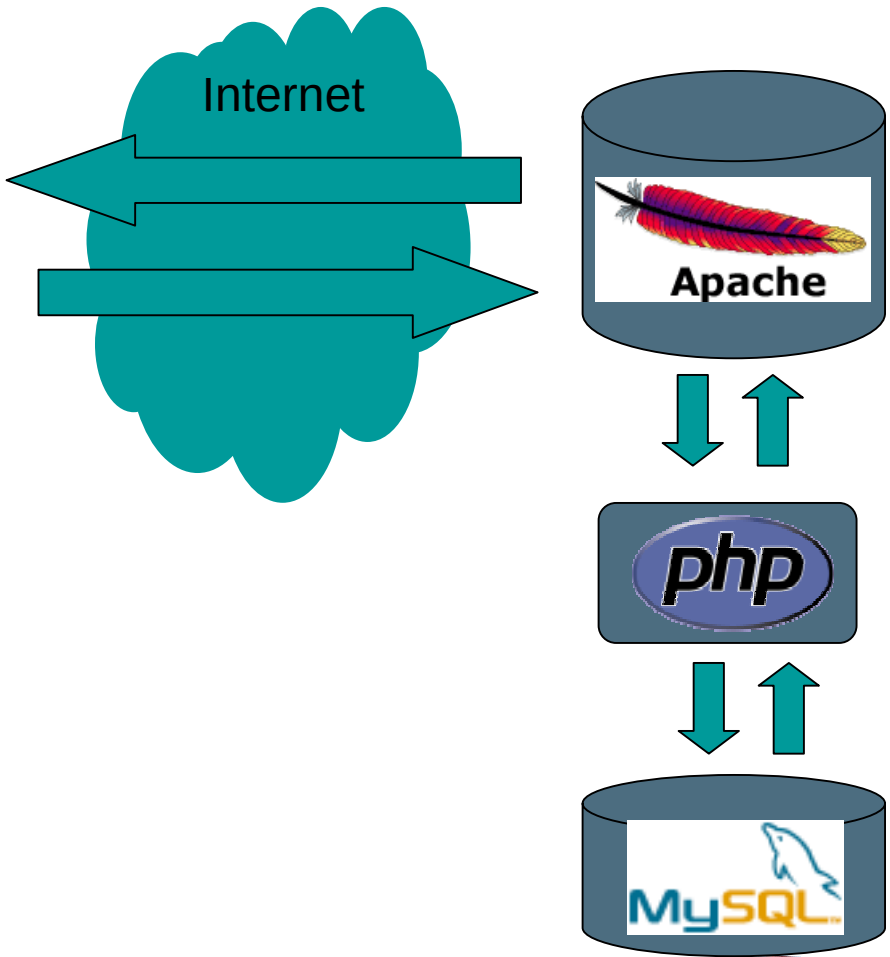
# Cours de programmation web

ENSAE 2006-2007

# Cours 2 - PHP


1. Rappels sur PHP
2. Syntaxe de base
3. Structures de contrôle
4. Fonctions, classes
5. Interaction avec l'utilisateur
6. Interaction avec une base de données
7. Bibliothèque de fonctions

# Rappels sur PHP



# Rappels sur PHP


- Générateur de HTML (Javascript, CSS)
- Langage interprété (pas de compilation)
- L'utilisateur requête une URL, le serveur interprète le PHP et retourne la page HTML générée.

# Syntaxe de base



Les blocs de code sont insérés dans la page entre <?php et ?> :

```
<html lang="fr">
  <head>
    <title>Mon premier document PHP</title>
  </head>
  <body>
<?php
$pseudo = "Robert";
echo "<p>Bienvenue " .$pseudo."</p>;
?>
  </body>
</html>
```

# Syntaxe de base


- Instructions séparées par un « ; »
- Commentaires

```
//commentaire de fin ligne

/* bloc
multiligne de commentaires */
```

# Syntaxe de base : les variables


- Les noms commencent par $
- Le typage est fort et dynamique

```
$a = 1234;
$aussi_long_que_tu_veux = 5555;
$a = 1.255;
$a = « Bonjour »;
$b = ‘Bonjour’;
```

# Syntaxe de base : Les tableaux


- Tableaux scalaires :

  **$nombres = array(1, 2, 3, 4, 5, 6);**

  **$nombres[0]=1; $nombres[1]=2; $nombres[2]=3; … $nombres[5]=6;**

- Tableaux associatifs :

  **$panier=array( ‘fruit’ => ‘banane’, ‘legume’ => ‘haricot’ );**

  **$panier[‘fruit’]=‘banane’; $panier[‘legume’]= ‘haricot’;**

- Rem : les chaines de caractères sont des tableaux

# Syntaxe de base : Les tableaux


- Création dynamique :
  **$a[0]=1;$a[1]=5;**
  **$a[]=-10; ⇔$a[2]=-10;**

- Tableaux de tableaux :
  **$a=array(**
  **array(1,2,3),**
  **array(« fruit » => 2, « legume » => 5),**
  **« 10 »);**
  - ⇔ **$a[0]= array(1,2,3); $a[1]= array(« fruit » => 2, « legume » => 5);$a[2]=« 10 »;**
  - ⇔ **$a[0][0]=1;$a[0][1]=2;$a[0][2]=3…**

# Syntaxe de base : portée des variables




- Portée locale / globale :

```
$a = 1; // portée globale
Function Test () {
    echo $a; // portée locale
}
Test (); // n'écrira rien
```

- Constantes : Portée globale

```
define(« taux_euro »,6.55957);
```

On y accède sans $ !

# Syntaxe de base : opérateurs arithmétiques


- **$a + $b** Addition Somme de $a et $b.
- **$a - $b** Soustraction Soustraction de $b
- **$a * $b** Multiplication Produit de $a et $b
- **$a / $b** Division Dividende de $a par $b
- **$a % $b** Modulo Reste de la division de $a par $b

# Syntaxe de base : opérateurs


- Concaténation de chaînes de caractères :
  - **$a = "Hello "; $b = $a . "World!";**

- Assignement : La valeur d'une expression d'assignement est la valeur assignée
  - **$a = ($b = 4) + 5;** => $a= ?
  - **$a = 3; $a += 5;** => $a= ?
  - **$b = "Hello "; $b .= "There!";** => $b= ?

# Syntaxe de base : Opérateurs logiques


- **$a and $b** ET Vrai si $a ET $b sont vrais
- **$a && $b** ET Vrai si $a ET $b sont vrais
- **$a or $b** OU Vrai si $a OU $b est vrai
- **$a || $b** OU Vrai si $a OU $b est vrai
- **!$a** NON Vrai si $a est faux

# Syntaxe de base : opérateurs de comparaison


- **$a == $b** Egal à Vrai si $a est égale à $b.
- **$a != $b** Différent de Vrai si $a est différent de $b.
- **$a < $b** Plus petit que Vrai si $a est plus petit strictement que $b.
- **$a > $b** Plus grand que Vrai si $a est plus grand strictement que $b.
- **$a <= $b** Plus petit ou égal à Vrai si $a est plus petit ou égal à $b.
- **$a >= $b** Plus grand ou égal à Vrai si $a est plus grand ou égal à $b.

# Structures de contrôle : if/else


- if (expression) instruction

```
if($a > $b) print "a est plus grand que b";
```

- **else / elseif**

```
if ($a > $b) {
        print "a est plus grand que b";
} elseif ($a == $b) {
        print "a est égal à b";
}  else {
        print "a est plus petit que b";
}
```

# Structures de contrôle : if


- Autre syntaxe possible :
  if (expression): instructions ... endif;

```
<?php     if (date('Y')==2003): ?>
On est en 2003 !
<?php endif; ?>
```

# Structures de contrôle : while


- while (expression) instruction
- while (expression): instructions ... endwhile;

```
while ($pas >= 1e-3) {
    $milieu = ($a+$b) / 2;
    if(f($milieu)<0) {
       $a=$milieu;
    } else {
       $b=$milieu;
    }
    $pas = abs($b-$a);
}
```

# Structures de contrôle : do / while


- do instruction while (expression);

```
$i = 0;
do {
   print $i;
} while ($i>0);
```

# Structures de contrôle : for

- for (expr1; expr2; expr3) instruction

  for (expr1; expr2; expr3): instructions ...;
  endfor;

```
for ($i = 1; $i <= 10; $i++) {
    print $i;
}
```

# Structures de contrôle : foreach


- foreach(tableau as $value) instructions

```
$a = array (1, 2, 3, 17);
foreach ($a as $v) {
    print "Valeur courante de \$a: $v.\n";
}
```

# Structures de contrôle : switch


- Remplace une suite de if

```
switch ($i) {
    case 0:
        print "i égale 0";
        break;
    case 1:
        print "i égale 1";
        break;
    case 2:
        print "i égale 2";
        break;
}
```



# Structures de contrôle : break / continue


- break : pour sortir d'une boucle

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

- continue : passer à l'itération suivante

```
$i = 0;
while ($i++ < 5) {
    if (($i % 2) == 0) continue;
    print $i;
}
```

# Structures de contrôle : include


- La fonction include() inclus et évalue le fichier spécifié en argument (au niveau de interpréteur PHP)

```
include(« file.inc.php »);
include(« connexion.php »);
```

# Fonctions / Classes : Fonction


- Déclaration d'une fonction

```
function foo ($arg_1, $arg_2, ..., $arg_n) {
    echo "Exemple de fonction.\n";
    return $retval;
}
```

- Autant d'arguments que nécessaire
- Une seule valeur de retour
- Toutes les variables sont locales à la méthode (y compris les arguments)

# Fonctions / Classes : valeur par défaut


- On peut préciser les valeurs par défaut :

```
function servir_apero ($type = "ricard") {
    return "Servir un verre de ".$type.".\n";
}
echo servir_apero();
echo servir_apero("whisky");
```

# Fonctions / Classes : Classes


- Une classe est une collection de variables et de fonctions qui fonctionnent avec ces variables.
- Permettent une bonne organisation du code en vue de sa réutilisation
- Héritage avec *extends*

```
class Caddie {
    var $date_du_jour;
    var $items;
    function Caddie() {
        $this->date_du_jour = date("d/m/Y");
        $this->item = array()
    }
    function add_item ($artnr, $num) {
        $this->items[$artnr] += $num;
    }
}
$cart = new Caddie;
$cart->add_item("10", 1);
$another_cart = new Caddie;
$another_cart->add_item("0815", 3);
```




# Interaction avec l'utilisateur : Formulaires


- Transmises dans la requête HTTP
- Variables globales $_POST et $_GET (selon le mode d'appel à la page) :

  http://citron/~dcabasson/mapage.php?var1=valeur

  **echo $_GET['var1'];**

# Interaction avec l'utilisateur : Cookies


- Permet le stockage d'informations sur le client (implémentation décrite dans HTTP)
- $_COOKIE contient les valeurs des variables
- Problème de sécurité
- Stockage sur le long terme (détecter les navigateurs étant déjà venus)

# Interaction avec l'utilisateur : Sessions




- Variables stockées coté serveur ($_SESSION)
- Un client est identifié comme un navigateur durant une plage de temps donnée

```
session_start();
if (!isset($_SESSION['compteur'])) {
    $_SESSION['compteur'] = 0;
} else {
    $_SESSION['compteur']++;
}
```

# Interaction avec une base de données


- mysql_connect – Connexion à un gestionnaire de bases de données
- mysql_select_db – Choix de la base de données

```
$link = mysql_connect("mysql_hote",
   "mysql_utilisateur", "mysql_mot_de_passe")
    or die("Impossible de se connecter");
echo "Connexion réussie";
mysql_select_db("my_database") or die(«
   Impossible de selectionner la base");
```

# Interaction avec une base de données


- mysql_query – Exécution d'une requête SQL

```
$query = "SELECT * FROM my_table";
$result = mysql_query($query) or die("
    Echec de la requête");
```

# Interaction avec une base de données


- mysql_fetch_assoc – parcours du résultat

```
echo "<table>\n";
while ($line = mysql_fetch_assoc($result)) {
    echo "\t<tr>\n";
    foreach ($line as $col_value) {
        echo "\t\t<td>$col_value</td>\n";
    }
    echo "\t</tr>\n";
}
echo "</table>\n";
```

# Interaction avec une base de données


- mysql_free_result – fermeture d'un résultat
- mysql_close – fermeture d'une connexion

```
mysql_free_result($result);

mysql_close($link);
```

# Bibliothèque : Variables


- *isset* -- Savoir si la variable est affectée
- *unset* -- Désaffecter une variable
- *settype*(string var, string type) -- typer une variable
- *gettype* -- récupérer le type
- *is_array, is_integer, is_string*.. -- test sur le type

# Bibliothèque : Mathématiques


- *Abs* -- Valeur absolue
- *Sin, Cos, ACos, Tan*..
- *Ceil, Floor, Round* -- Arrondis
- *Log, Log10, exp* -- Logarithme & Exponentielle
- *max, min* -- renvoie le min ou le max d'un tableau
- *pi* -- Retourne la valeur de pi
- Sqrt -- Racine carrée.
- *srand, rand* -- génération d'un nombre aléatoire
- *Number_format* -- Formatage de nombres



# Bibliothèque : Date - Time


- *time* --  Retourne le timestamp UNIX actuel (nombre de secondes depuis le 1/1/1970)
- *date* -- Formate une date/heure locale
- *getdate* -- Retourne la date/heure (sous forme de tableau associatif)
- *checkdate* -- Valide une date/heure

# Bibliothèque : Chaines de caractères


- *explode* -- Scinde une chaîne en morceaux, grâce à un délimiteur. (opposé : *join*)
- *ucfirst* -- Force le premier caractère d'une chaîne en majuscule.
- *strtolower* -- Met tous les caractères en minuscules. (opposé : *strtoupper* )
- *substr* -- Retourne une partie de la chaîne.
- *strpos* -- Recherche la première occurrence d'un caractère dans une chaîne.
- *trim* -- Enlève les espaces de début et de fin de chaîne

# Bibliothèque : Accès aux fichiers


- *opendir, readir, chdir, closedir* -- accès aux repertoires
- *is_dir, is_file, is_readable, is_writable* -- tests
- *file, fopen* -- ouverture d'un fichier ou d'une URL
- *mkdir, unlink*, *chmod* -- modification paramètres du fichier

# Bibliothèque : manipulation de tableaux



- *array* — Crée un tableau
- *array_push, array_unshift, array_pop, array_shift* -- modification sur les tableaux
- *array_merge* — Fusionne un ou plusieurs tableaux
- *array_keys, array_values* — Retourne toutes les clés (valeurs) d'un tableau
- *array_walk* — Exécute une fonction sur chacun des éléments d'un tableau
- *in_array* — Indique si une valeur appartient à un tableau
- *asort, arsort, ksort, usort, uksort, shuffle* – Tri
- *reset, next, prev, end, current* -- description du tableau
- *compact, extract* -- transfert tableau-variable