Internet et bases de données

Clara Bertolissi

SIL-NTI

Université de Provence

Structure de l'enseignement

- Cours + TD: 7 x 3h30
  (9-10-11, 16-17 janvier 9h00 – 12h30,
  5-6 mars 9h00 – 12h30)

- TPs :
  (9-10-11, 16-17-18 janvier ,
  5-6-7 mars)

Contenu du cours

- Introduction au langage PHP
- La syntaxe : variables, structures de contrôle, fonctions.
- Passage d'informations entre les pages : méthodes GET et POST, sessions, cookies.
- Interfaçage avec une base de donnée :
  langage MySql
  connexion et requêtes à une base via Php/Mysql
- Ajax : asynchronous Javascript + XML
- Logiciels Wiki pour la creation de sites web collaboratifs.

Contenu des TPs

JANVIER

Apprendre à réaliser un site web interactif qui s'interface avec une base de données :

- Gestion d'une DVDthèque

MARS

Introduction à Ajax

Projet

- Création d'une page web qui utilise l'ensemble des technologies internet vues en cours.

- Projet dû le 8 mars,
  Soutenance individuelle le 17-18 mars.

Evaluation

- Contrôle continu (Partiel, sous forme de TP noté, le 6 mars)
- Examen final écrit le 17 mars.
- Projet (développement + soutenance)

Plan

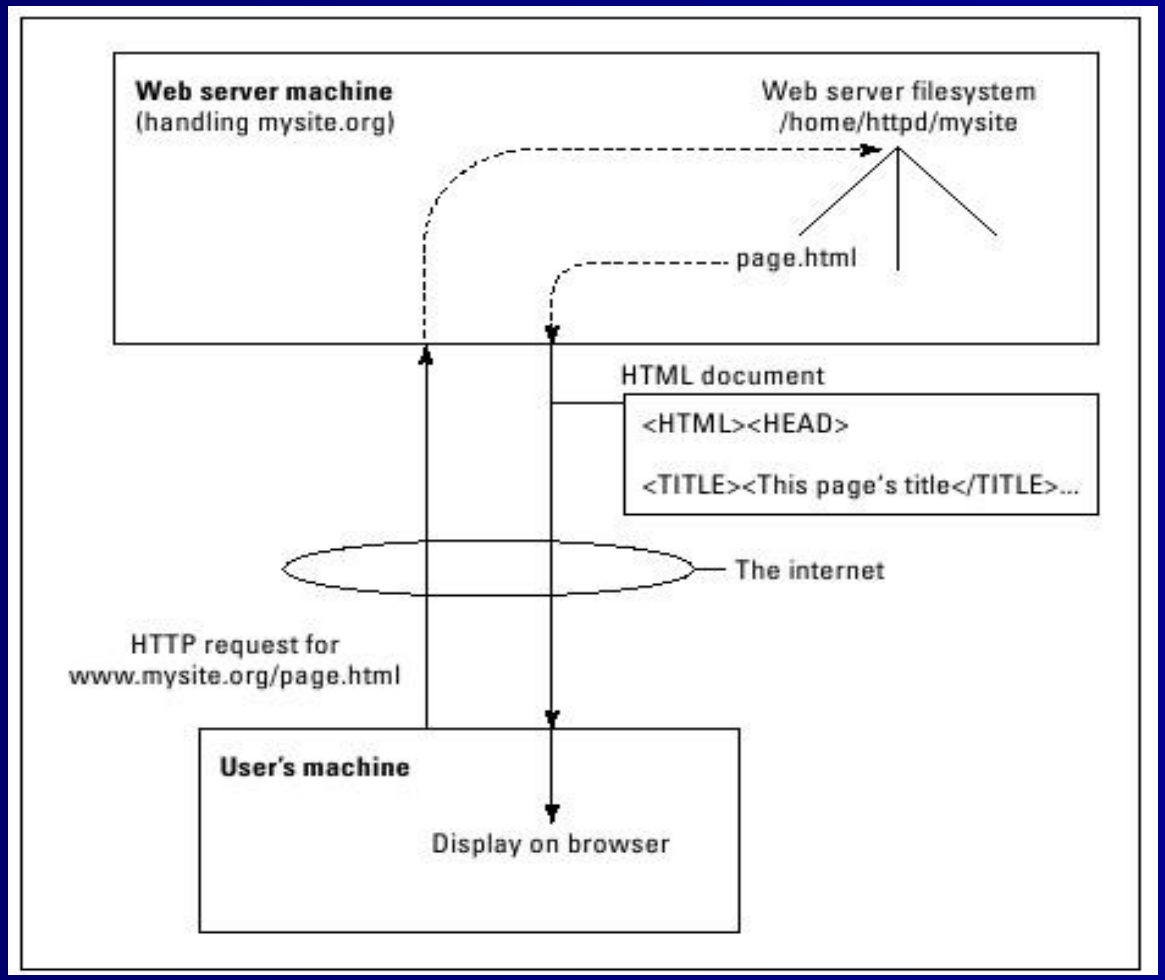
- Introduction
- La syntaxe: variables, constantes, opérateurs, tableaux, …
- Les blocs de contrôle: if, boucles while, for, foreach, switch
- Les fonctions

Généralités

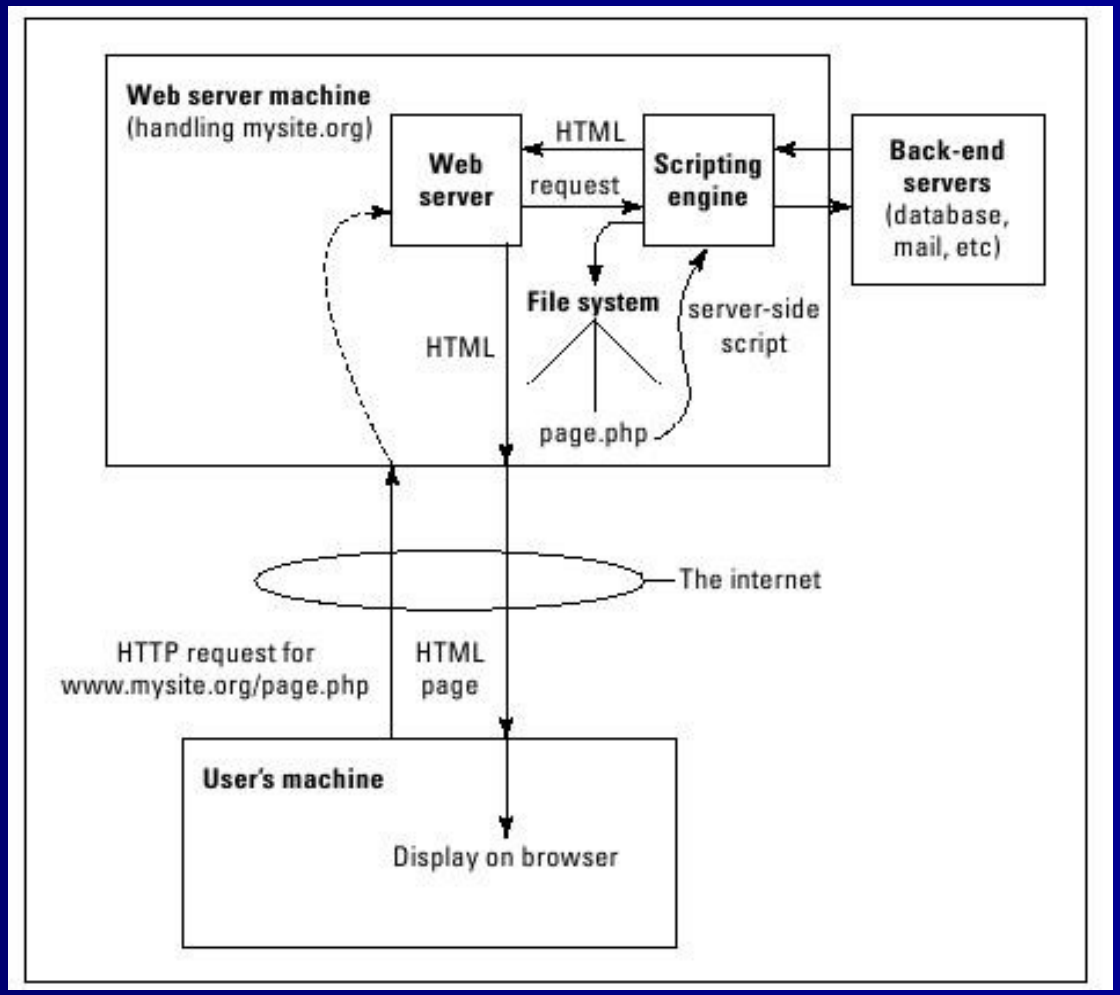
- Langage de script coté serveur
- Développé en C
- Soutenu par la Apache Fundation Software
- Logiciel libre et gratuit

- Site officiel http://www.php.net

Communication client-serveur



Script du coté serveur



Références

- http://www.php.net
- http://www.phpinfo.net
- http://www.phpfrance.com
- http://www.developpez.com/php/

- Autres Ressources en français:
  - http://www.phpindex.com
  - http://dev.nexen.net/

- Ressources en anglais
  - http://www.phpbuilder.com
  - http://www.allhtml.com

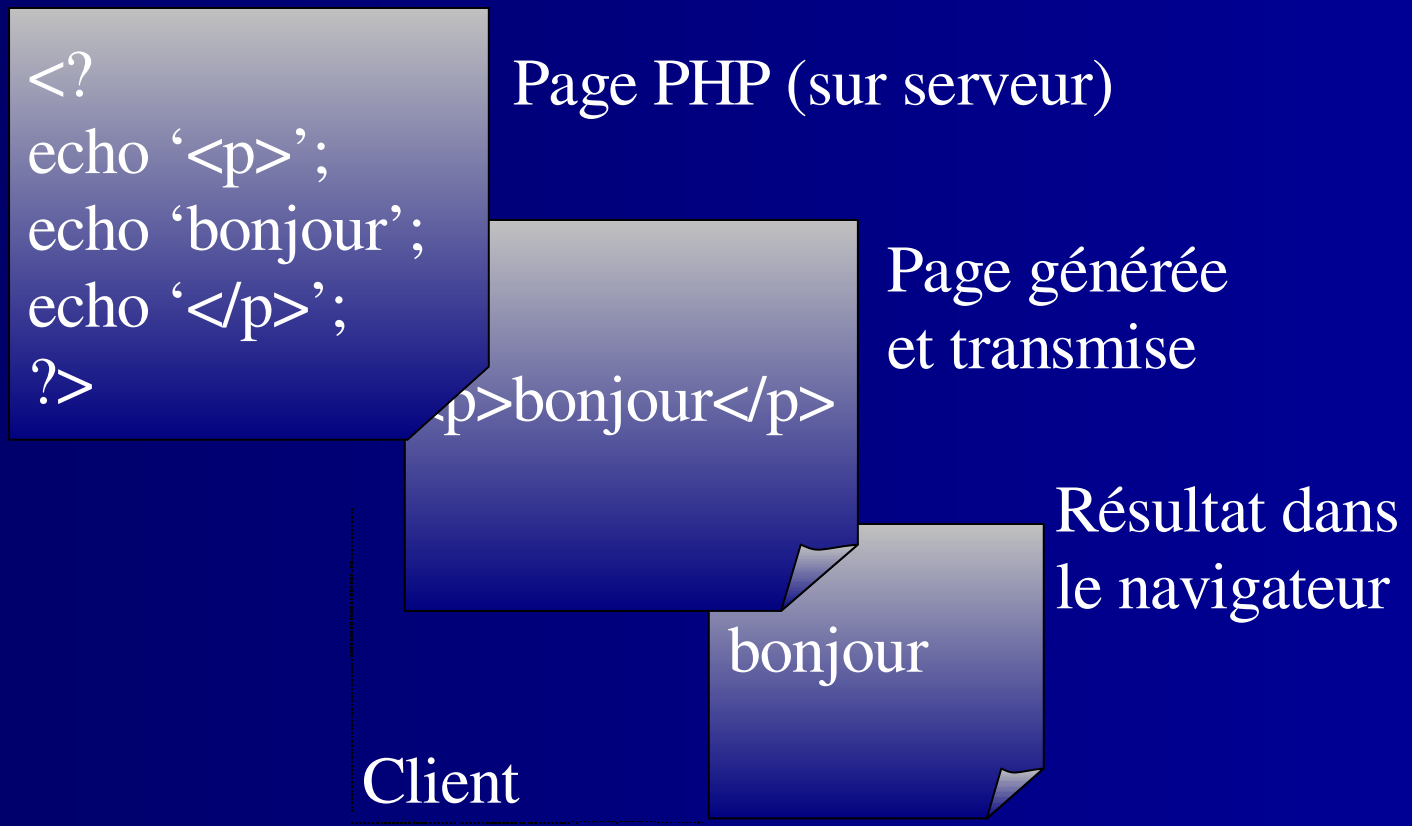
Pourquoi Php ?

- Performances élevées.
- Multi-base (MySQL, Oracle, Informix, Interbase, Sybase, ODBC, JDBC, ...)
- Bibliothèques de fonctions, de scripts
- conçu pour le Web
- Simplicité d'apprentissage
- Gratuité, Open Source

Php en quelques mots

- Très proche du C, C++, Java
- Directement intégré dans les pages HTML
- Entre les balises <?php Et ?>
- Simplicité d'écriture
- Typage faible
- Pas de forte structuration du code
- Allocation / libération de mémoire automatique
- Seul le développeur est garant de la lisibilité du code

Visibilité du code



Premier exemple

Exemple:

```
<html>
<body>
<?php
    echo ”Bonjour”;
?>
</body>
</html>
```

Autres syntaxes d’intégration :

```
<? ... ?>
<script language=“php”> ... </script>
<% ... %>
```

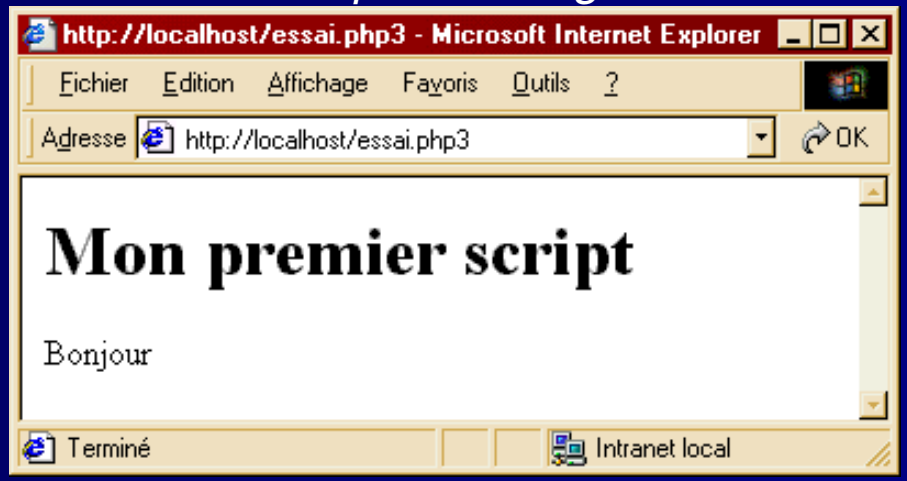
Deux scripts équivalents : code source

```
<html>
<body>
<h1>Mon premier
   script</h1>
<?php echo
   "Bonjour\n"; ?>
</body>
</html>
```

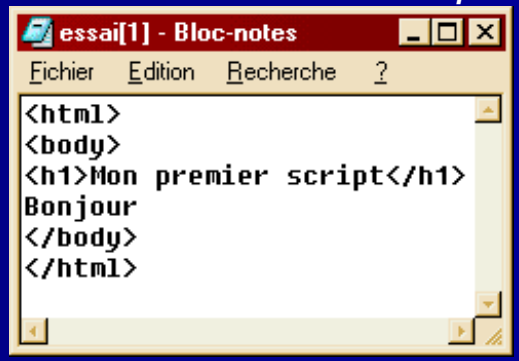
```
<?php
echo
   "<html>\n<body>\n";
echo "<h1>Mon premier
   script</h1>\n";
echo "Bonjour\n";
echo
   "</body>\n</html>\n";
?>
```

Résultat cote client

Résultat affiché par le navigateur :



Code source (côté client) de la page résultant du script



Commentaires

- <?php
- // commentaire de fin de ligne
- /* commentaire
- sur plusieurs
- lignes */
- # commentaire de fin de ligne comme en Shell #
- ?>

Variables

- Précédée d’un ‘$’
- Affectation avec '='
- Pas de « déclaration » (automatique)
- Faiblement typé
- Portée :
- Locale
- Globale
- Super-Globale

Variables

Quelques fonctions :

- **empty($var)** : renvoie vrai si la variable est vide
- **isset($var)** : renvoie vrai si la variable existe
- **unset($var)** : détruit une variable
- **gettype($var)** : retourne le type de la variable
- **settype($var, ”type”)** : convertit la variable en type **type** (cast)
- **is_long(), is_double(), is_string(), is_array(), is_object()**…

- Une variable peut avoir pour identificateur la valeur d’une autre variable.
- *Syntaxe :* **${$var} = valeur;**
- *Exemple :*
- **$toto = ”foobar”;**
- **${$toto} = 2002;**
- **echo $foobar;** // la variable **$foobar** vaut **2002**

Variables

- Le type d'une variable est également défini par la valeur qui lui a été affectée lors de sa création. Il existe 5 types de données :
  - Entier (int, integer)
  - Décimal (real, double, float)
  - Chaîne de caractères (string)
  - Tableau (array)
  - Objet (object)

Variables

- Selon l'endroit du code où les variables sont définies, ces dernières auront une portée plus ou moins grande.

- 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.

Variables d’environnement

- On peut en Php utiliser des variables d'environnement du serveur, et notamment celles du serveur HTTP.
- Avec PHP toutes les variables d'environnement sont des variables globales. Ainsi il suffit de les utiliser directement dans le code.

exemple :

Tableau $_SERVER ;

$IP=$_SERVER[‘**REMOTE_ADDR**’] ;

echo $IP;

Variables d'environnement

- **SERVER_NAME** : Le nom du serveur hôte qui exécute le script suivant.
- **PHP_SELF :** Le nom du fichier du script en cours 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.
- **HTTP_REFERER** : URL de la source ayant renvoyée le client sur la page en cours

Constantes

L'utilisateur peut définir des constantes dont la valeur est fixée une fois pour toute.

Elles se définissent avec le mot clé **define** et s'utilisent avec leur nom sans $ devant.

define("cons",valeur) : définit la constante cons de valeur **valeur**

Exemple 1 :

```
define("author","Hugo");
echo author;          // affiche Hugo
```

Exemple 2 :

```
define('TAUX_EURO', 6.55957);
```

Contrairement aux variables, les identificateurs de constantes (et aussi ceux de fonction) ne sont pas sensibles à la casse.

Opérations

■+=	$a+=$b	$a = $a+$b	++$a $a++
■-=	$a-=$b	$a=$a-$b	--$a $a--
■*=	$a*=$b	$a=$a*$b	
■/=	$a/=$b	$a=$a/$b	
■%=	$a%=$b	$a=$a%$b	
■.=	$a.=$b	$a=$a.$b	Concaténation

$a = true;
$b = !$a; ($b == false)

&&	et
||	ou
xor	ou exclusif

$a = 5;
$a = &$b;

Référence

On peut à la manière des pointeurs en C faire référence à une variable grâce à l'opérateur **&** (ET commercial).

Exemple 1 :

```
$toto = 100;   // la variable $toto est initialisée à la valeur 100
$foobar = &$toto; // la variable $foobar fait référence à $toto
$toto++;              // on change la valeur de $toto
echo $foobar;      // qui est répercutée sur $foobar qui vaut alors 101
```

Exemple 2 :

```
function change($var) {
    $var++;     // la fonction incrémente en local l'argument
}
$nbr = 1;        // la variable $nbr est initialisée à 1
change(&$nbr);   // passage de la variable par référence
echo $nbr;     // sa valeur a donc été modifiée
```

Variables booléennes

prennent pour valeurs **TRUE** (vrai) ou **FALSE** (faux).

Une valeur entière nulle, une chaîne de caractères vide **“”,** les chaînes **‘’0’’** et **‘0’** et l’entier **0** lui même sont évalués a` **FALSE**.

- Exemple :
- **if(0) echo 1;**
- **if("") echo 2;**
- **if("0") echo 3;**
- **if("00") echo 4;**
- **if('0') echo 5;**
- **if('00') echo 6;**
- **if(" ") echo 7;**

Affichera ?

Chaînes de caractères

Une variable chaîne de caractères n'est pas limitée en nombre de caractères. Elle est toujours délimitée par des simples quotes ou des doubles quotes.

Exemples :

$nom = ''Bruni'';

$prenom = 'Carla';

Les doubles quotes permettent l'évaluation des variables et caractères spéciaux contenus dans la chaîne (comme en C ou en Shell) alors que les simples ne le permettent pas.

Exemples :

echo ''Nom: $nom''; // affiche **Nom: Bruni**

echo 'Nom: $nom'; // affiche **Nom: $nom**

Quelques caractères spéciaux : **\n** (nouvelle ligne), **\r** (retour à la ligne), \\ (antislash), **\$** (caractère dollars), **\''** (double quote).

Exemple :

echo ''Hello Word !\n'';

Chaînes de caractères

Opérateur de concaténation de chaînes : **.** (point)

- **$name = ”Henry”;**
  $whoiam = $name.“IV”;

- **$out = ‘Patati’;**
  $out .= ‘’ et patata’... ’;

- **$foo = ”Hello”;**
  $bar = ”Word”;
  echo $foo.$bar;

- Affichage d’une chaîne avec **echo**:

echo ‘Hello Word.’;

echo ”Nom : ”, $name;

echo ”Hello ${name}\n”;

echo(”Bonjour”);

Chaînes de caractères

Quelques fonctions:

- **strlen($str)** : retourne le nombre de caractères d'une chaîne
- **strtolower($str)** : conversion en minuscules
- **strtoupper($str)** : conversion en majuscules
- **trim($str)** : suppression des espaces de début et de fin de chaîne
- **substr($str,$i,$j)** : retourne une sous chaîne de **$str** de taille **$j** et débutant à la position **$i**
- **strnatcmp($str1,$str2)** : comparaison de 2 chaînes
- **addslashes($str)** : déspécialise les caractères spéciaux (', ", \)
- **ord($char)** : retourne la valeur ASCII du caractère **$char**

Chaînes de caractères

- On peut délimiter les chaînes de caractères avec la syntaxe *Here-doc.*

- *Exemple :*

```
$essai = <<<EOD
Ma chaîne ”essai”
sur plusieurs lignes.
EOD;
echo $essai;
```

- La valeur de la variable **$essai** est délimitée par un identifiant que vous nommez librement. Sa deuxième apparition doit se faire en premier sur une nouvelle ligne.
- Cette valeur chaîne se comporte comme si elle était délimitée par des doubles quotes ” “ dans le sens où les variables seront évaluées.

Affichage

Les fonctions d'affichage :

- **echo()** : écriture dans le navigateur
- **print()** : écriture dans le navigateur
- **printf([$format, $arg1, $arg2])** : écriture formatée comme en C, i.e. la chaîne de caractère est constante et contient le format d'affichage des variables passées en argument

- *Exemples :*

```
echo "Bonjour $name";
print("Bonjour $name");
printf("Bonjour %s", $name);
```

Les tableaux

Les éléments d'un tableau peuvent être de types différents et sont séparés d'une virgule.

Un tableau peut être initialisé avec la syntaxe **array**.

Exemple :

```
$tab_colors = array( 'red', 'yellow', 'blue', 'white');
$tab = array('foobar', 2002, 20.5, $name);
```

Mais il peut aussi être initialisé au fur et à mesure.

Exemples :

```
$prenoms[ ] = ''Clément'';          $villes[0] = ''Paris'';
$prenoms[ ] = ''Justin'';           $villes[1] = ''Londres'';
$prenoms[ ] = ''Tanguy'';           $villes[2] = ''Lisbonne'';
```

L'appel d'un élément du tableau se fait à partir de son indice (dont l'origine est zéro comme en C).

Exemple : `echo $tab[10];` // pour accéder au 11ème élément

Les tableaux

Parcours d'un tableau.

$tab = array('Hugo', 'Jean', 'Mario');

Exemple 1 :

```
$i=0;
while($i <= count($tab)) {          // count() retourne le nombre d'éléments
  echo $tab[$i].'\n';
  $i++;
}
```

Exemple 2 :

```
foreach($tab as $elem) {
  echo $elem."\n";
}
```

La variable **$elem** prend pour valeurs successives tous les éléments du tableau **$tab**.

Les tableaux

Quelques fonctions:

- **count($tab), sizeof($tab)** : retournent le nombre d'éléments du tableau
- **in_array($var,$tab)** : dit si la valeur de **$var** existe dans le tableau **$tab**
- **list($var1,$var2)..** : transforme une liste de variables en tableau
- **range($i,$j)** : retourne un tableau contenant un intervalle de valeurs
- **shuffle($tab)** : mélange les éléments d'un tableau
- **sort($tab)** : trie alphanumérique les éléments du tableau
- **rsort($tab)** : trie alphanumérique inverse les éléments du tableau
- **implode($str,$tab)** : retournent une chaîne de caractères contenant les éléments du tableau **$tab** joints par la chaîne de jointure **$str**
- **explode($delim,$str)** : retourne un tableau dont les éléments résultent du hachage de la chaîne **$str** par le délimiteur **$delim**
- **array_merge($tab1,$tab2,$tab3)..** : concatène les tableaux passés en arguments

Les tableaux

- Attention, les variables tableaux ne sont pas évaluées lorsqu'elles sont au milieu d'une chaîne ce caractère délimitée par des doubles quottes.

- *Exemple :*
- **echo "$tab[3]";** // syntaxe invalide
- **echo $tab[3];** // syntaxe valide

Les tableaux

Un tableau associatif est appelé aussi *dictionnaire* ou *hashtable*. On associe à chacun de ses éléments une clé dont la valeur est de type chaîne de caractères.

L'initialisation d'un tableau associatif est similaire à celle d'un tableau normal.

Exemple 1 :

```
$personne = array("Nom" => "César", "Prénom" => "Jules");
```

Exemple 2 :

```
$personne["Nom"] = "César";
$personne["Prénom"] = "Jules";
```

Ici à la clé **"Nom"** est associée la valeur **"César"**.

Les tableaux

Parcours d'un tableau associatif.

```
$personne = array("Nom" => "César", "Prénom" => "Jules");
```

Exemple 1 :

```
foreach($personne as $elem) {
  echo $elem;
}
```

Ici on accède directement aux éléments du tableau sans passer par les clés.

Exemple 2 :

```
foreach($personne as $key => $elem) {
  echo "$key : $elem";
}
```

Ici on accède simultanément aux clés et aux éléments.

Les tableaux

Quelques fonctions :

- **array_count_values($tab)** : retourne un tableau contenant les valeurs du tableau **$tab** comme clés et leurs fréquence comme valeur (utile pour évaluer les redondances)
- **array_keys($tab)** : retourne un tableau contenant les clés du tableau associatif **$tab**
- **array_values($tab)** : retourne un tableau contenant les valeurs du tableau associatif **$tab**
- **array_search($val,$tab)** : retourne la clé associée à la valeur **$val**

- L'élément d'un tableau peut être un autre tableau.

- Les tableaux associatifs permettent de préserver une structure de données.

Les tableaux

Quelques fonctions alternatives pour le parcours de tableaux (normaux ou associatifs) :

- **reset($tab)** : place le pointeur sur le premier élément
- **current($tab)** : retourne la valeur de l'élément courant
- **next($tab)** : place le pointeur sur l'élément suivant
- **prev($tab)** : place le pointeur sur l'élément précédant
- **each($tab)** : retourne la paire clé/valeur courante et avance le pointeur

- *Exemple :*
- **$colors = array("red", "green", "blue");**
- **$nbr = count($colors);**
- **reset($colors);**
- **for($i=1; $i<=$nbr; $i++) {**
- ** echo current($colors)."
";**
- ** next($colors);**
- **}**

Conditionnelle

```
If ($condition){
   ...Traitement si vrai
}else{
   ...Traitement si faux
}
```

Exemple :

```
If ($UtilisateurConnu){
   echo «Bienvenue
    $Utilisateur »;
   ...Contenu de la page...
}else{
   header ("location:
    http://monsite.fr/login.p
    hp");
   exit;
}
```

Boucle Tant que

```
while ($condition){
   ...Traitement
}
```

Exemple

```
$i = 0;
$NotFinished = true;
while ($NotFinished){
   if (!isset ($Tableau[$i])){
      $NotFinished = false;
   }else{
      $Tableau[$i] =
    $Tableau[$i]*2;
      $i++;
   }
}
```

Boucle Pour

```
for (départ; condition;
    passage){
    Traitement unitaire
}
```

Exemple

```
function TableMulti ($TableDe){
   for ($i=0; $i<=10; $i++){
      echo 'TableDe [' . $i . ']=' .
    $TableDe[$i] . "\n";
   }
}
```

Switch

```
switch( <variable>) {
  case <valeur1> :
     <traitement si
     variable=valeur1>
     break;
  case <valeur2> :
  case <valeur3> : <traitement
     si variable=valeur2 ou
     valeur3>
      break;
 default :
    <traitement par defaut>
}
```

Exemple

```
switch($var){
  case 'toto': echo "c'est toto !";
     break;
  case 'marc': echo "marc est
     là"; break;
  default:
     echo "connais pas";
}
```

Structures de contrôle

L'instruction **break** permet de quitter prématurément une boucle.

Exemple :

```
while ($tab[$i] != $val){
    echo $tab[$i]. "<br />";
    if ($tab[$i] == $val)
     break;
}
```

L'instruction **continue** permet d'éluder les instructions suivantes de l'itération courante de la boucle pour passer à la suivante.

Exemple :

```
for($i=1; $i<=10; $i++) {
    if($tab[$i] == $val)
     continue;
    echo $tab[$i];
}
```

Fonctions *each() et list()*

Ces fonctions sont aussi étroitement liées aux boucles.

each($tab)

Parcour des éléments d'un tableau.

Elle retourne la combinaison clé-valeur courante, puis se positionne sur l'élément suivant.

Lorsque la fin du tableau est atteinte, each( ) retourne la valeur faux (false).

list($val,$val,…)

affectation des éléments du tableau dans des valeurs distinctes.

Très souvent associée à la fonction each().

Fonctions *each() et list()*

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

```
0 : val1
1 : val2
2 : val3
```

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

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

Arrêt prematuré

Pour stopper prématurément un script, il existe deux fonctions.

die arrête un script et affiche un message d'erreur dans le navigateur.
Exemple :

```
if(mysql_query($requet) == false)
    die(''Erreur de base de données à la requête : <br />$requet'');
```

exit l'arrête aussi mais sans afficher de message d'erreur.
Exemple :

```
function foobar() {
    exit();
}
```

Ces fonctions stoppent tout le script, pas seulement le bloc en cours.

Les fonctions

Même sans paramètre, un entête de fonction doit porter des parenthèses **( )**. Les différents arguments sont séparés par une virgule **,** . Et le corps de la fonction est délimité par des accolades **{ }**.

Quelques exemples :

```
function afficher($str1, $str2) {     // passage de deux paramètres
    echo "$str1, $str2";
}
function bonjour() {          // passage d'aucun paramètre
    echo "Bonjour";
}
function GetColor() {         // retour d'une chaîne
    return "black";
}
```

Les fonctions

Une fonction peut être définie après son appel (du fait de la compilation avant exécution).

Exemple :

```
function foo() {              // définition de la fonction foo
    echo ‘’Foo’’... ’’;
}
foo();          // appel de la fonction foo définie plus haut
bar();          // appel de la fonction bar pas encore définie
function bar() {              // définition de la fonction bar
    echo ‘’bar!<br />’’;
}
```

Cet exemple affichera : **Foob. ar!**.

Les fonctions

Les fonctions peuvent prendre des arguments dont il n'est pas besoin de spécifier le type. Elles peuvent de façon optionnelle retourner une valeur.

L'appel à une fonction peut ne pas respecter son prototypage (nombre de paramètres). Les identificateurs de fonctions sont insensibles à la caisse.

Exemple :

```
function mafonction($toto) {
    $toto += 15;
    echo "Salut !";
    return ($toto+10);
}
$nbr = MaFonction(15.1);
/* retourne 15.1+15+10=40.1, les majuscules n'ont pas
    d'importance */
```

Les fonctions

global permet de travailler sur une variable de portée globale au programme. Le tableau associatif **$GLOBALS** permet d'accéder aux variables globales du script (**$GLOBALS["var"]** accède à la variable **$var**).

Exemple :

```
function change() {
    global $var;       // définit $var comme globale
    $GLOBALS["toto"] ++; // incrémente la variable globale $toto
    $var++;              // cela sera répercuté dans le reste du programme
}
```

static permet de conserver la valeur d'une variable locale à une fonction.

Exemple :

```
function change() {
    static $var;      // définit $var comme statique
    $var++;           // sa valeur sera conservée jusqu'au prochain appel
}
```

Les fonctions

On peut donner une valeur par défaut aux arguments de la fonction.
Qui sera utilise' si un argument est « oublié » lors de l'appel de la fonction.

Exemple :

```
function Set_Color($color=''black'') {
    global $car;
    $car[''color''] = $color;
}
```

Forcer le passage de paramètre par référence (équivalent à user de **global**):
Exemple :

```
function change(&$var) { // force le passage systématique par référence
    $var += 100;     // incrémentation de +100
}
$toto = 12;         // $toto vaut 12
change($toto); // passage par valeur mais la fonction la prend en  référence
echo $toto;         // $toto vaut 112
```

Inclusion

- On peut inclure dans un script php le contenu d'un autre fichier.

- **require** insert dans le code le contenu du fichier spécifié même si ce n'est pas du code php. Est équivalent au préprocesseur *#include* du C.
- *Exemple :*
- **require("fichier.php");**

- **include** évalue et insert à chaque appel (même dans une boucle) le contenu du fichier passé en argument.
- *Exemple :*
- **include("fichier.php");**

Aujourd'hui on a vu

- Syntaxe: Variables, tableaux, constantes, Opérateurs
- Les blocs de contrôle: if, boucles while, for, foreach, switch
- Définition et appel de fonctions