LES BASES DU PHP

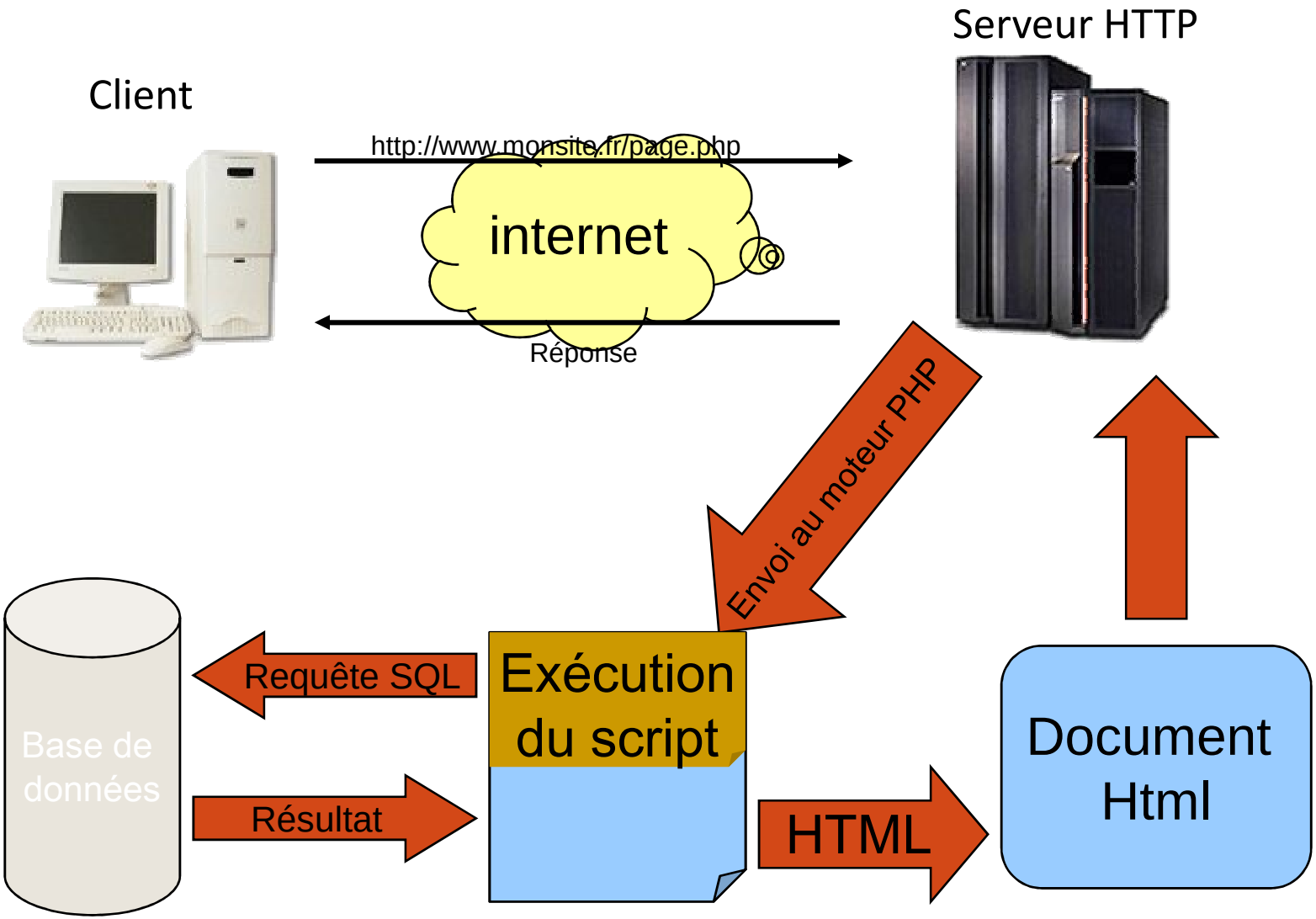
Introduction

- **Historique**
  - PHP signifie **PHP: Hypertext Preprocessor** . La confusion vient du fait que la première lettre de l'acronyme représente l'acronyme lui-même. Ce type d'acronyme est appelé un acronyme récursif.
  - Historiquement, cette acronymie récursive était l'abréviation de **Personal Home Page**

- **Qu'est-ce que PHP ?**
  - PHP est un langage de script HTML, exécuté coté serveur. Sa syntaxe est empruntée aux langages C, Java et Perl, et est facile à apprendre. Le but de ce langage est de permettre aux développeurs web d'écrire des pages dynamiques rapidement, mais vous pouvez faire beaucoup plus avec PHP.

STRUCTURE DE BASE

Fonctionnement d'un script php



Intégration d'un script dans une page

Les page contenant du php doivent porter l'extension .php ou .php3 (version > 3)

Pour insérer du code dans le HTML il suffit de le placer entre les balises : <?php … ?>

Exemple:

```
1  <?php
2        $sTexteHtml ='<p>Bonjour, vous êtes sur la première page</p>';
3  ?>
4  <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
5  <html>
6    <head>
7    <meta http-equiv="content-type" content="text/html; charset=UTF-8">
8    <title></title>
9    </head>
10   <body>
11 <?php
12       echo  $sTexteHtml ;
13 ?>
14   </body>
15 </html>
```

Les Commentaires

- **Un script php se commente comme en C ou javascript :**

```
<?php
  // commentaire de fin de ligne
  /* commentaire
       sur plusieurs
   lignes */

  #commentaire de fin de ligne comme en Shell
?>
```

- **Tout ce qui se trouve dans un commentaire est ignoré.**
- **Il est conseillé de commenter largement ses scripts.**

Mon premier script

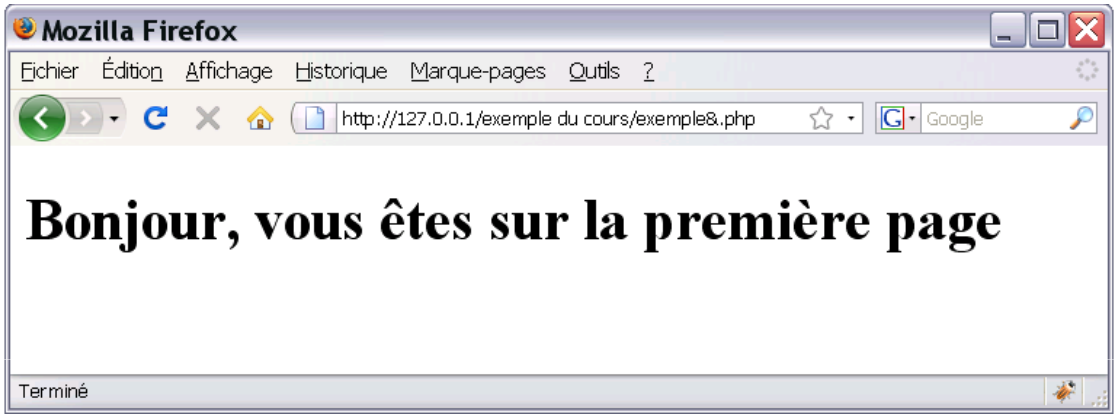
Fichier PHP sur le serveur

```php
<?php
      $sTexteHtml ='<h1>Bonjour, vous êtes sur la première page</h1>';
?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
  <meta content="text/html; charset=UTF-8" http-equiv="content-type" />
  <title></title>
</head>
<body>

<?php
      echo  $sTexteHtml ;
?>
  </body>
</html>
```

Mon premier script(2)

Résultat affiché sur le navigateur du poste client



Code source généré

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
  <meta content="text/html; charset=UTF-8" http-equiv="content-type" />
  <title></title>
</head>
<body>

<h1>Bonjour, vous êtes sur la première page</h1>  </body>
</html>
```

VARIABLES ET CONSTANTES

Déclaration

- **Pas de déclaration : une variable est implicitement déclarée dès qu'elle est utilisée.**
- **Le type de la variable se détermine en fonction du contenu de la variable et peut évoluer au fur et à mesure des utilisations.**
- **Exemple :**

```php
<?php
$sNom="Cangimrac";
$sPrenom="Laurent";
?>
```

Règle de nommage des variables

- **Syntaxe :**
  - **sensibles à la casse**
  - **commencent toujours par $**
  - **continuent par une lettre ou par _ mais pas un chiffre**
  - **se finissent par un nombre quelconque de lettres, chiffres et _**

Règle de nommage des variables

- **Convention de nommage que vous utiliserez dans tous vos script php :**
  - ***Le nom de la variable doit être évocateur de son contenu***
  - ***La première lettre d'une variable est une minuscule***
  - ***Chaque variable commence par une lettre en minuscule qui identifie le type de la donnée que portera la variable :***
    - *s pour string (chaine de caractères)*
    - *i pour integer (entier)*
    - *r pour réel*
    - *b booléen*
    - *t pour tableau*
    - *o pour objet*

```php
<?php
$sNom_Prof="Carmignac";
$sPrenom_Prof="Laurent";
$iAge=46;
$rTaille=1.80;
?>
```

  - ***chaque changement de mot est précédé du caractère _ ou commence par une majuscule***

Portée

- **Variables locales : visibles que dans le contexte (block) où elles ont été créées.**

- **Par défaut, les variables de PHP sont des variables locales.**

- **Pour utiliser une variable dans un autre contexte, il faut la passer en paramètre, ou utiliser une variable globale.**

Porté

- **Variables globales :**
  - **Utilisation de la super-globale *$GLOBALS[]***

```php
<?php
function message($_msg1,$_msg2)
{
   echo $_msg1." ".$GLOBALS['sNom']." ".$_msg2." ".$GLOBALS['sPrenom'];
}

$sNom="Cangimrac";
$sPrenom="Laurent";
message("Mon nom est","Mon prénom est")
?>
```

Mon nom est Cangimrac Mon prénom est Laurent

  - **Utilisation de l'instruction *global***

```php
<?php
function message($_msg1,$_msg2)
{
   global $sNom,$sPrenom;
   echo $_msg1." ".$sNom." ".$_msg2." ".$sPrenom;
}

$sNom="Cangimrac";
$sPrenom="Laurent";
message("Mon nom est","Mon prénom est")
?>
```

Mon nom est Cangimrac Mon prénom est Laurent

Les variables dynamiques

- **On peut à la manière des pointeurs en C, faire référence à une variable grâce à l'opérateur & (ET commercial).**
  - ***Exemple 1 :***
    - `$iNbr = 100;` *// la variable $var est initialisée à la valeur 100*
    - `$adr_iNbr= & $iNbr;` *// la variable $adr_var fait référence à $var*
      // elle contient l'adresse de $var
    - `$iNbr ++;` *// on change la valeur de $var = 101*
      `echo $adr_iNbr;`

Constantes

- **Identifiant :**
  - **Mêmes règles de nommage que pour les variables**
  - **Par convention, les identifiants des constantes sont en majuscules**
  - **Une constante ne commence jamais par un $**
- **Déclaration :**

  define("NOM CONSTANTE", "Valeur")
- **Exemple :**

```
define("TVA",19.6);
echo TVA;
```

Traitement sur les variables

- **Quelques fonctions :**

  empty($Var) : renvoie vrai si la variable $Var n'existe pas ou si elle contient une chaine vide('') ou 0

  isset($Var) : renvoie vrai si la variable $Var existe ,

  unset($Var) : détruit une variable $Var,

  gettype($Var) : retourne le type de la variable $Var,

  settype($Var, "type") : convertit la variable $Var, en type (cast)

  Une variable peut avoir pour identificateur la valeur d'une autre variable.

  Syntaxe : ${$var} = valeur;

  Exemple :

```
$sToto = "foobar";
${$sToto} = 2002;
echo $foobar;          // la variable $foobar vaut 2002
```

Traitement sur les variables(2)

```php
<?php
  $sNom='';
  if (empty($sNom))
    echo 'La variable <i>$sNom</i> est vide. Elle est de type :<i>'.gettype($sNom).'</i>';


  $iAge=0;
  if (empty($iAge))
    echo '<br />La variable <i>$iAge</i> est vide. Elle est de type :<i>'.gettype($iAge).'</i>';


  $iAge=20;
  if (isset($iAge))
    echo '<br />La variable <i>$iAge</i> est définie et elle contient une valeur';


  unset($iAge); // destruction de la variable
  if (!isset($iAge))
    echo '<br />La variable <i>$iAge</i> n\'existe plus';
?>
```

La variable *$sNom* est vide. Elle est de type :*string*

La variable *$iAge* est vide. Elle est de type :*integer*

La variable *$iAge* est définie et elle contient une valeur

La variable *$iAge* n'existe plus

TYPES DE DONNÉES

Deux familles

- **Les variables PHP supportent plusieurs types de données**
  - **les types scalaires (types simples)**
    - **entier : int, integer** *($iNombre=12;)*
    - **réel : real, float, double** *($iReel=1.12; $iReel_1=5e-3;)*
    - **chaîne : string** *($sNom="cangimrac";)*
    - **booléen : bool, boolean** *($bDrapeau=TRUE;)*
  - **les types composés**
    - **tableau : array**
    - **objet : object**

Les chaînes de caractères

- **On peut les définir :**
  - **soit à l'aide d'une paire de simple quote (')**
  - **soit à l'aide d'une paire de double quote (")**
- **Les deux formes ne sont pas équivalentes...**

```php
$sNom = 'Laurent';
echo 'bonjour '.$sNom.'<br>';
echo "bonjour $sNom <br>";
echo '<p id="para">boujour '.$sNom.'</p><br>';
echo "<p id=\"para\">$sNom </p><br>";
```

Les chaînes de caractères

- **Quelques fonctions de traitement de chaines :**
  - 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,$str2) : comparaison de 2 chaînes
  - ord($char) : retourne la valeur ASCII du caractère $char
  - chr (ascii) : retourne le caractère correspondant à la valeur ACSII
  - explode(del,chaine) : Cette fonction renvoie dans un tableau, les valeurs comprises entre un séparateur que vous indiquez en argument.

    $liste=explode($delimiteur,$variable);

  - implode(tab,del) : Cette fonction a l'effet inverse de explode. Elle permet à partir d'une liste de créer une chaîne simple. Les différents éléments de la liste seront "recollés" via ce séparateur.

    $variable=implode($liste,$delemiteur);

  - ……

Les chaînes de caractères

```php
$sEntete = "num;nom;Prenom;Adresse";
echo ' ma chaîne avant explode : '.$sEntete.'<br />';

echo 'aprés explode : <br />';

$Tab=explode(';',$sEntete);
echo '<pre>';
print_r($Tab);
echo '</pre>';

echo 'Construction d\'une chaine séparée par le caractére \'@\' <br />
    la chaine contient les éléments du tableau précédent<br />';

$sNewCh = implode('@',$Tab);
echo ' ma nouvelle chaîne après implode : '.$sNewCh.'<br />';
```

```
ma chaîne avant explode : num;nom;Prenom;Adresse
aprés explode :

Array
(
    [0] => num
    [1] => nom
    [2] => Prenom
    [3] => Adresse
)

Construction d'une chaine séparée par le caractére '@'
la chaine contient les éléments du tableau précédent
ma nouvelle chaîne après implode : num@nom@Prenom@Adresse
```

Connaître le type d'une variable

- **Connaître le type d'une variable : *gettype (…);***

```php
echo gettype("bonjour");    // Affiche string

$iNbr = 19;
echo gettype($iNbr);        // Affiche integer

$rNbr_Reel = 19.6;
echo gettype($rNbr_Reel); // Affiche double

$bBoll = TRUE;
echo gettype($bBoll);       // Affiche boolean

$bTest = $iNbr==19;
echo gettype($bTest);       // Affiche boolean
```

- **Tester le type (fonctions d'accès rapide) :**
  - **Types scalaires :**
    - **is_string(…)**
    - **is_double (…) et is_float(…)**
    - **is_int (…) et is_integer(…)**
    - **is_boolean(…)**
  - **Types composés : is_object(…) et is_array(…)**

LES OPÉRATEURS

Opérateurs d'affectation

- **Affectation par copie :**

```php
$a = 1;
$b = $a;
$a = $a + 1;
echo $a;  //Affiche 2, car on a ajouté 1
echo $b;  //Affiche 1, sa valeur n'a pas été modifiée
```

- **L'opérateur d'affectation renvoie la valeur affectée : ceci permet des affectations en chaîne.**

```php
$a = $b = 3;
$a = ($b = 3);

$b = 3;
echo ($a=$b);
```

Opérateurs d'affectation

- **Affectation par référence**

```php
$a = 1;
$b = & $a;
$a = $a + 1;
echo $a;  //Affiche 2, car on a ajouté 1
echo $b;  //Affiche 2, sa valeur a été modifiée
```

- **Effacer une référence avec la fonction *unset(…)* :**

```php
$a = 1;
$b = & $a;
$a = 2;
echo 'Valeur de $b : ', $b, '<br>'; //Affiche 2

unset($a);
echo 'Valeur de $a : ', $a, '<br>'; //N'affiche plus rien
echo 'Valeur de $b : ', $b, '<br>'; //Affiche 2
```

Opérateurs de bases

- **Opérateurs arithmétiques**
  - **Addition : +**
  - **Soustraction : -**
  - **Multiplication : ***
  - **Division : /**
  - **Modulo : %**
- **Opérateur de concaténation de chaînes : . (le point)**
- **Opérateurs combinés : +=, -=, *=, /=, %=, .=, &=, |=, ^=**
- **Opérateurs d'incrémentation : ++, --**

Opérateurs de bases

- **Opérateurs logiques :**
  - **ET logique : && , and**
  - **OU logique : || , or**
  - **NON logique : !**
  - **OU exclusif logique : xor**

- **Opérateurs de comparaison :**
  - **Egalité : ==, !=**
  - **Identité : ===, !==**
  - **Relation d'ordre : <, <=, >, >=**

Opérateurs de bases

- **Quelques exemples**

```php
$i = 1;
echo $i++;  // Affiche 1
echo $i;    // Affiche 2

$t = array(0 => "Zéro", 1 => "Un");
$i = 0;
echo $t[$i];    // Zéro car $i vaut 0

$s1 = "Hello ";
$s2 = "world";
echo $s1 . $s2;   // Hello world
$s1 .= $s2;       // ou $s1 = $s1 . $s2;
echo  $s1;        // Hello world
```

Priorités entre opérateurs

Priorité	Opérateurs
1	( ) [ ]
2	-- ++ !
3	* / %
4	+ -
5	< <= >= >
6	== != === !==
7	&
8	|
9	&&
10	||
11	Affectation, opérateurs combinés (= += -= etc.)
12	AND
13	OR
14	XOR

LES STRUCTURES DE CONTRÔLE

Structures conditionnelles

- même syntaxe qu'en langage C

```
if( condition )
  {  …  }
else  // elseif
  {  …  }
```

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";  }
```

```
switch( variable )
   {case … :  …
              break ;
     …
     default :  …     }
```

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; }
```

Opérateur ternaire comme en C :

```
(condition)?expression1:expression2;
```

```
$var4=2;
echo  ($var4>3 )? 'plus grand<br />' : 'plus petit<br />';
```

Remarques:

la condition doit être entre des parenthèses

Les boucles

```
for( Val initiale; condition de sortie ; incrément ou décrément)
 {
    …
 }
-----------------------------------------------
while( condition pour rester da la boucle )
 {
    …
 }
-----------------------------------------------
do
 {
    …
 } while(condition pour rester de la boucle);
```

La différence avec le while est que les instructions dans la boucle seront éxécuté au moins une fois

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

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

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

Instructions de boucle

- **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 -

LES TABLEAUX

Les tableaux

- **En PHP, les tableaux :**
  - **sont dynamiques, leur taille peut évoluer au cours de l'exécution du script.**
  - **peuvent contenir des éléments de types différents : scalaires ou composés.**

- **Il existe deux types de tableaux :**
  - **Les tableaux indicés : c'est une liste d'éléments repérés par une position unique, son indice. En PHP, cet indice commence à zéro.**
  - **Les tableaux associatifs : C'est une liste d'éléments repérés par un identifiant arbitraire unique appelé la clé.**

Déclaration des tableaux indicés

Une variable tableau est de type array. Un tableau accepte des éléments de tout type. Les éléments d'un tableau peuvent être de types différents. Un tableau en php est dynamique (pas de taille prédéfinie)

Un tableau peut être initialisé avec la syntaxe array.

```
$tab_couleurs = array("rouge", "jaune", "bleu", "blanc");
$tab = array("laurent", 2002, 20.5, $nom);
```

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

```
$prenoms[ ] = "Clément";
$prenoms[ ] = "Justin";
$prenoms[ ] = "Tanguy";

$villes[0] = "Paris";
$villes[1] = "Londres";
$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).

```
echo $tab[10];
```

Parcours d'un tableau indicés

- 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.

Quelques fonctions de tableaux

count($tab) : retournent le nombre d'éléments du tableau

in_array($var,$tab) : renvoie VRAI 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

Array_merge($tab1,$tab2,$tab3…) : concatène les tableaux passés en arguments

array_rand($tab) : retourne un élément du tableau au hasard

Les tableaux associatifs : déclaration

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";
```

```
Array
(
    [Nom] => César
    [Prénom] => Jules
)
```

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

Parcours d'un tableau associatif

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

- **Exemple 1 :**

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

César
Jules

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

- **Exemple 2 :**

```
foreach($personne as $key => $elem)
   {
   echo $key .' : '.$elem.'<br />';
  }
```

Nom : César
Prénom : Jules

Ici on accède simultanément aux clés et aux éléments.

Fonctions des tableaux associatifs

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 (comme en C).

Les tableaux à n dimensions

- **Construction :**

```php
<?php

$tabDep[13]=array("marseille","arles", "aix", "salon");
$tabDep[77]=array("Melun","Montereau", "Fontainebleau");
$tabDep[10]=array("Troyes","Romilly-Sur-Seine","La Chapelle-Saint-Luc");
```

- **Parcours :**

```php
foreach ($tabDep as $dep => $tabVille)
  {
   echo '<p><b>'.$dep.'</b><br />';
   foreach( $tabVille as $ville)
      echo '<i>'.$ville.'</i><br />';
   echo '</p>';
  }
?>
```

13
marseille
arles
aix
salon

77
Melun
Montereau
Fontainebleau

10
Troyes
Romilly-Sur-Seine
La Chapelle-Saint-Luc

```
Array
(
    [13] => Array
        (
            [0] => marseille
            [1] => arles
            [2] => aix
            [3] => salon
        )

    [77] => Array
        (
            [0] => Melun
            [1] => Montereau
            [2] => Fontainebleau
        )

    [10] => Array
        (
            [0] => Troyes
            [1] => Romilly-Sur-Seine
            [2] => La Chapelle-Saint-Luc
        )

)
```

Les fonctions each() et list()

- Ces fonctions sont aussi étroitement liées aux boucles et utilisées pour le parcours de tableaux

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

Les fonctions each() et list()

```php
<?php
//on crée un tableau associatif de 2 couples de valeurs
$personne = array("Nom" => "César", "Prénom" => "Jules");

while ( list($cle, $valeur) =  each($personne ) )
 {
   echo $cle.' : '.$valeur.' <br>';
 }
?>
```

Nom : César
Prénom : Jules

Les principaux tableaux globaux du php

- **Ces tableaux sont des tableaux réservés à php, ils sont disponibles.**

$GLOBALS

Contient une référence sur chaque variable. Les clés de ce tableau sont les noms des variables globales. $GLOBAL['var1']

$_SERVER

Les variables fournies par le serveur web, ou bien directement liées à l'environnement d'exécution du script courant.. $_SERVER['PHP_SELF']

$_GET

Les variables fournies au script via la chaîne de requête URL. $_GET['id']

$_POST

Les variables fournies par le protocole HTTP en méthode POST. $_POST['nom']

$_FILES

Les variables fournies par le protocole HTTP, suite à un téléchargement de fichier. $_FILE['fic1']['size']

$_SESSION

Les variables qui sont en fait enregistrées dans la session attachée au script. $_SESSION['var2']=2;

Constantes du PHP

- **Les constantes prédéfinies du PHP :**

  __FILE__ : nom et chemin du fichier en cours
  __LINE__ : numéro de ligne en cours
  PHP_VERSION : version de PHP
  PHP_OS : nom du système d'exploitation de la machine sur laquelle est installée le moteur PHP

- **Exemple :**

```php
<?php
echo __FILE__.' <br />N° de ligne'.__LINE__ .'<br />';
echo 'PHP_VERSION :'.PHP_VERSION.'<br />';
echo 'PHP_OS :'.PHP_OS.'<br />';
echo '$_SERVER[\'PHP_SELF\'] :'.$_SERVER['PHP_SELF'].'<br />';
echo '$_SERVER[\'HTTP_REFERER\'] :'.$_SERVER['HTTP_REFERER'].'<br />';
?>
```

```
E:\COURS IUT ARLES\1 - Cours DUT INFO\OMGL S3 - SGBD 3 (PHP+MySQL)\1- Cours PHP INFO02\2005-2006\Test exemple\constantes.php
N° de ligne16
PHP_VERSION :5.2.0
PHP_OS :WINNT
$_SERVER['PHP_SELF'] :/Test PHP COURS/constantes.php
$_SERVER['HTTP_REFERER'] :http://127.0.0.1/Test%20PHP%20COURS/
```

LES FONCTIONS

Les fonctions

Comme tout langage de programmation, php permet l'écriture de 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 casse.

Exemple :

```php
<?php
function MaFonction($toto)
{
  $toto+=15;
  return ($toto+10);
}
echo MaFonction(120).'<br />';

?>
```

Les fonctions

- ## Valeur par défaut des arguments

```
function Set_Color($color="black")
        {
         echo $color.'<br />';
        }

// appel de la fonction
$sCouleur='red';
Set_Color();            // sans valeur de paramètre => retourne black
Set_Color('rouge');     // retourne rouge
Set_Color($sCouleur);// passage de variable en paramètre => retourne red
```

Les fonctions

- ## Passage de paramètre par référence

```
function change(&$var) // force le passage systématique par référence
        {
          $var += 100; // incrémentation de +100
        }

$iNbr = 12;             // $toto vaut 12
change($iNbr );         // passage par valeur mais la fonction la prend en  référence
echo $iNbr;             // $toto vaut 112
```

Les fonctions

- **Retour de plusieurs valeurs par une fonction.**

  En retournant un tableau ayant pour éléments les variables à retourner. Dans l'appel de la fonction, il faudra alors utiliser la procédure list() qui prend en paramètre les variables à qui ont doit affecter les valeurs retournées. On affecte à list() le retour de la fonction.

```
function trigo($nbr)
 {
   return array(sin($nbr), cos($nbr), tan($nbr));          // retour d'un tableau
 }

$rRayon = 12;
list($a, $b, $c) = trigo($iRayon ); // affectation aux variables $a,$b et $c des
                                    // éléments du tableau retourné par la fonction trigo
// affichage des variables
echo "sin($rRayon )=$a, cos($rRayon )=$b, tan($rRayon )=$c";
```

Cet exemple affichera ceci :
sin(12)=-0,5365729180, cos(12)=0,8438539587, tan(12)=-0,6358599286

INCLUSION DE FICHIERS

Inclusion de fichiers

- **PHP offre la possibilité d'inclure des fichiers depuis des scripts ce qui permet par exemple, de se créer des bibliothèques de fonctions, qui pourront être utilisées par plusieurs scripts.**

- **Les instructions permettant de réaliser des inclusions sont *include(…)* et *require(…)*.**

- **Il existe également *include_once(…)* et *require_once(…)* qui ont la même fonction mais s'assurent en plus que le fichier n'a pas déjà été inclut.**

Inclusion de fichiers

- **include (…) :**
  - **Inclut et exécute un fichier PHP passé en argument.**
  - **Si le fichier n'existe pas, la fonction génère une alerte.**

- **require(…) :**
  - **Inclut le contenu d'un fichier et évalue le fichier PHP contenant la commande.**
  - **Si le fichier n'existe pas, la fonction génère une erreur fatale.**

- **Globalement, *require()* et *include()* sont identiques, sauf dans leur façon de gérer les erreurs. Ils produisent tous les deux une Alerte mais *require()* génère une erreur fatale.**

Inclusion de fichiers

- **Exemple avec include(…) :**
  - **Fichier vars.php :**

```php
<?php
$sCouleur = 'verte';
$sFruit = 'pomme';
?>
```

  - **Fichier test.php qui inclut le fichier vars.php:**

```php
<?php
echo "Une $sFruit $sCouleur"; // Affiche : Une
include 'vars.php';
echo "Une $sFruit $sCouleur"; // Affiche : Une pomme verte
?>
```

PASSAGE DE PARAMÈTRES À UN SCRIPT

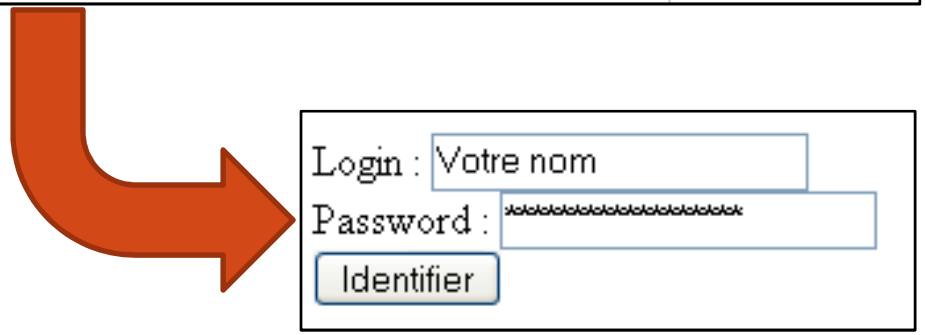
Passage de paramètres à un script(1)

- **Méthode des formulaires.**

  La méthode POST permet de passer des variables, saisies dans un formulaire par l'utilisateur, à un script php en vu d'un traitement

- **Exemple :**

```php
<?php
  echo  '<form method="post" action="check.php">
          Login : <input type="text" name ="login" value="Votre nom " /><br />
          Password : <input type="password" name ="pass" value="votre mot de passe" /><br />
          <input type="submit" value="Identifier" />
          </form>';
?>
```



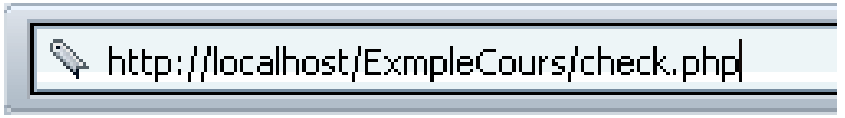
Passage de paramètres à un script(2)

- Comment récupérer la valeur des variables POSTEES

```
// dans le fichier check.php
  if($_POST['pass']=="xrT12")
      echo 'Bienvenu '. $_POST['login'];
  else
      echo "Acces interdit.";
```

(Attention cette exemple n'a qu'une valeur pédagogique !!!!)

Dans cet exemple, on contrôle la validité du mot de passe du formulaire précédent qui a été passé en paramètre au script check.php par la méthode POST.



Les données saisies n'apparaîtront pas dans l'URL et ne seront donc pas stockées dans les fichiers de log du serveur, contrairement à la méthode GET

Passage de paramètres à un script(3)

- **Méthode des ancres.**

  Les variables peuvent directement être passées par l'intermédiaire des URL.

- **Exemple :**

```
// dans le fichier fic1.php
  $iId = 20;
  echo '<a href="fichier.php?action=Vendre&ident='.$id.' ">Acheter</a>';

// dans le fichier fic2.php
  echo $_GET['action'].' - '.$_GET['ident'].' <br />';
            // retourne => Vendre - 20.
```

L'envoi de variables par cette méthode est la méthode GET

La redirection

- Commande de redirection (header)

```
<?php
  if($pass=="xrT12")
    header ("Location:Accueil.php"); // Redirige le client vers la page Accueil.php
  else
    header ("Location:Authent.php");
                  // Redirige le client vers la page d'authentification
?>
```

Attention un header() doit toujours être placé avant toutes autres commandes, rien ne doit être envoyé sur le flux http

```
<?php
    echo  "bonjour"; // NON!! RIEN AVANT LE HEADER
    header("Location:http://www.php.net/")
?>
```

bonjour
Warning: Cannot add header information - headers already sent by (output started at…

LA PROGRAMMATION ORIENTÉE OBJET

La Programmation Orientée Objet(1)

- **Avertissement**

  Php4 ne supporte pas l'objet de manière convenable (pas de restriction d'accès sur les membres de la classe).

  Si vous êtes adepte de la Programmation Orientée Objet (POO) vous devez installer php5 sur votre serveur

La Programmation Orientée Objet(2)

- **Déclaration et définition**

```php
<?php
class Voiture // déclaration de la classe
{
  // déclaration des données membres
  private $couleur;   // déclaration d'un attribut
  private $type = "sport"; // initilaisation de la donnée membre

  // déclaration des fonctions membres
  public function __construct() // constructeur
    {
      $this->couleur = "noire";
    }    // le mot clé $this faisant référence à l'objet est obligatoire

  public function Set_Couleur($couleur)
    {   $this->couleur = $couleur;  }

  public function Get_Couleur()
    {   return $this->couleur;  }
}
```

Restriction d'accès :
- Public
- Private
- protected

- **Instanciation**

```php
$mavoiture = new Voiture();   // création d'une instance
echo 'couleur de la voiture '.$mavoiture->Get_Couleur().'<br />';

$mavoiture->Set_Couleur("blanche"); // appel d'une méthode
echo 'couleur de la voiture '.$mavoiture->Get_Couleur().'<br />';
?>
```

```
couleur de la voiture noire
couleur de la voiture blanche
```

La Programmation Orientée Objet(3)

- **L'héritage**

Le système de classes de php est très succinct, il ne gère que l'héritage simple.

```php
class Voituredeluxe extends Voiture // déclaration de la sous classe
  {
  private $belle;

  public function __construct()        // constructeur
     {
       parent::__construct();
       $this->belle=TRUE();
      }
  }

?>
```

Héritage

L'opérateur de résolution de portée
parent :: membre de la classe parent
membres parent public ou protected
self :: membre de la même classe