Programmation Web
cours PHP

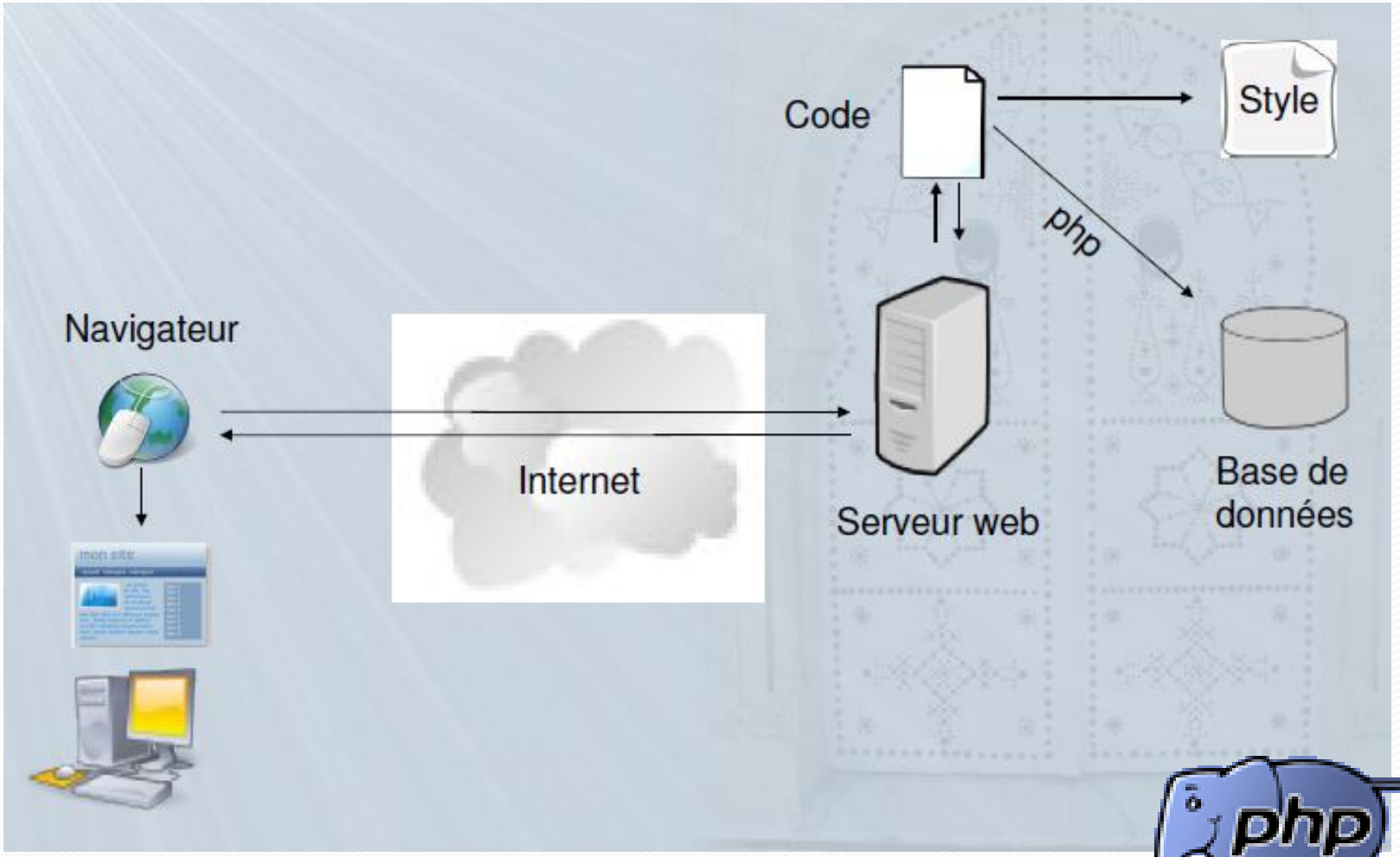
Hachem EL YOUSFI ALAOUI

DUT GLR - Février 20013

ENSET DE RABAT

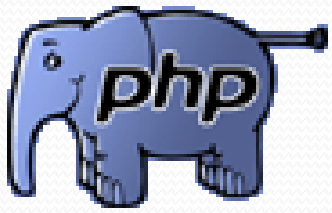
UNIVERSITE MOHAMMED v SOUISSI

Principe du Web



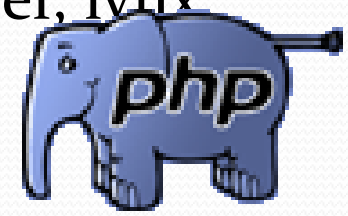
Serveur web

- Un **serveur web** est un hôte sur lequel fonctionne un **serveur HTTP** (ou **serveur web**). Un serveur web **héberge** les ressources qu'il dessert.
- Un **navigateur web** est un logiciel client HTTP conçu pour accéder aux ressources du web. Sa fonction de base est de permettre la consultation des documents HTML disponibles sur les serveurs HTTP. Le support d'autres types de ressource et d'autres protocoles de communication dépend du navigateur considéré.
- Une **page web** (ou **page**) est un document destiné à être consulté avec un navigateur web. Une page web est toujours constituée d'une ressource centrale (généralement un document HTML) et d'éventuelles ressources liées automatiquement accessibles (par exemple, des images).
- Voir http://www.wampserver.com
  http://www.easyphp.org

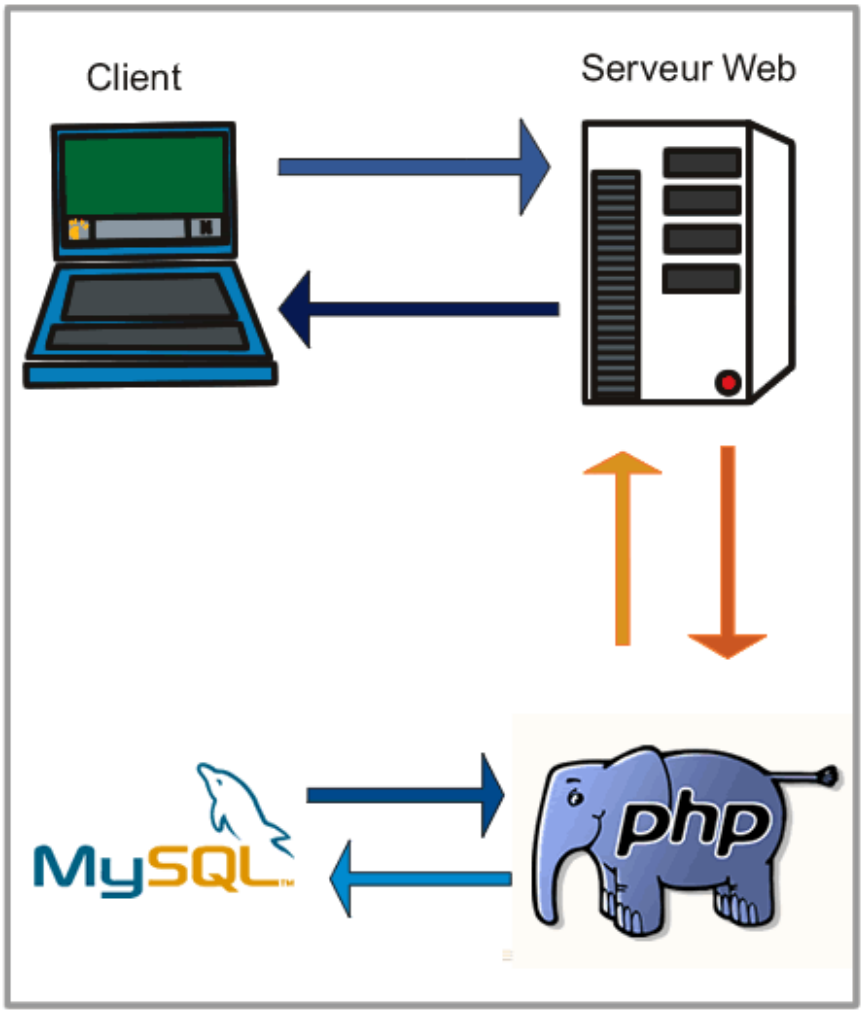


Serveur HTTP

- Il existe différents serveurs http
- Apache : http://httpd.apache.org
- IIS : www.microsoft.com
- Information sur le protocole http: www.w3.org/Protocoles/
- Échange d'information entre serveur et clients
- Pour échanger des informations entre le serveur et, les clients on utilise le protocole HTTP (ensemble de règles de codes)
- Nestcape Navigator, Microsoft Internet Explorer, lynx, iCab, …



Principe Web PHP



EasyPHP

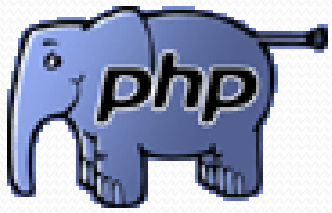
- 1ér package WAMP . (/LAMP)

(Windows, Appache, MySQL et PHP / Linux, Appache, MySQL et PHP)

Il s'agit d'une plateforme de développement Web, permettant de faire fonctionner localement des scripts PHP.

C'est un environnement comprenant :

- Deux serveurs :
  - Un serveur web Apache
  - Un serveur de bases de données MySQL
- Un interpréteur de script (PHP)
- Une administration SQL phpMyAdmin.



EasyPHP

- **Installation d'EasyPHP**

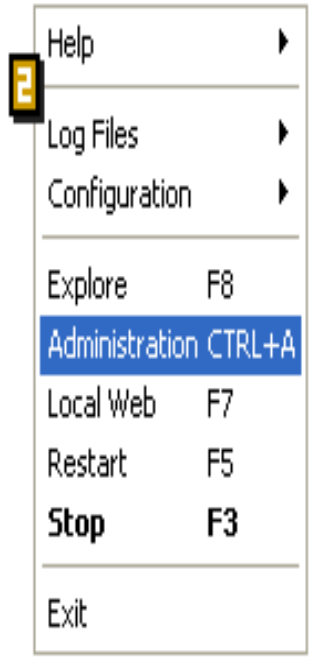
  Téléchargez la dernière version d'EasyPHP (serveur de développement : e.devserver) à partir du site Web http://www.easyphp.org.

- Exécutez le fichier télécharger et suivre l'assistant
- Remarque: Risque de conflit de port pour Apache

EasyPHP

- **Lancement et vérification**

Avant d'ouvrir l'"Administration" ou le "Web local", vérifier qu'easyphp.exe est lancé (ex.: Démarrer>Programmes>EasyPHP>EasyPHP) et que les serveurs fonctionnent (double-cliquer sur l'icône d'EasyPHP à côté de l'horloge [1] : les status d'Apache et de MySQL doivent être au vert).

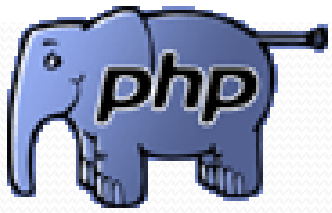


Administration

Faire un clic droit sur l'icône d'EasyPHP [1] et sélectionner "Administration"[2]. Cette page vous permet entre autre de gérer vos bases de données et d'administrer vos alias.

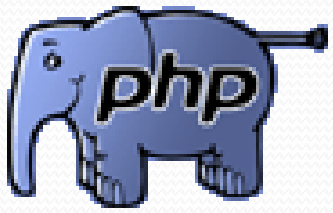
Web local

Faire un clic droit sur l'icône d'EasyPHP [1] et sélectionner "Web local" [2] afin de visualiser vos fichiers (cf. "Introduction à EasyPHP").



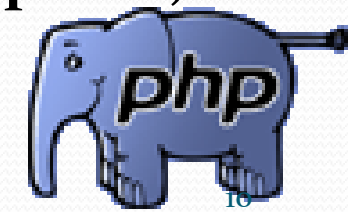
PHP Hypertext Preprocessor: les bases

- PHP signifiait à l'origine Personnal Home page.
- PHP (officiellement "PHP: Hypertext Preprocessor")
- langage de script (Interprété) qui est principalement utilisé pour être exécuté par un serveur HTTP, mais il peut fonctionner comme n'importe quel langage interprété en utilisant les scripts et son interpréteur sur un ordinateur
- Sa syntaxe et sa construction ressemblent à celles des langages C++ et Perl, à la différence que le PHP peut être directement intégré dans du code HTML.
- Extension d'un fichier PHP : .php



PHP Hypertext Preprocessor: les bases

- La gratuité e t la disponibilité du code source (PH P est distribué sous licence GNU GPL)
- Une grande communauté de développeurs partageant des centaines de milliers d'exemples de script PHP
- La simplicité d' écriture de scripts
- La possibilité d'inclure le script PHP au sein d'une page HTML (contrairement aux scripts CGi , pour lesquels il faut écrire des lignes de code pour afficher chaque ligne en langage HTML)
- La simplicité d'interfaçage avec des bases de données (de nombreux SGBD sont supportés, m ais le plus utilisé avec ce langage est MySQL, un SGBD gratuit disponible sur de nombreuses plate formes : Unix, Linux, Windows, MacOs X, Solaris, etc...)
- L'intégration au sein de nombreux serveurs web (Apache, Microsoft IIS, etc.).

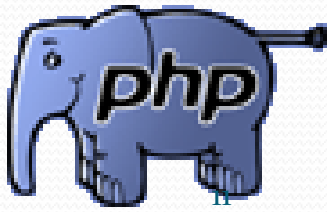


HTML et PHP

Les instructions PHP peuvent être insérées dans les commandes HTML, ce qui facilite le développement des sites web dynamiques

Dans un fichier PHP, deux types de "zones" :

- **Zones délimitées par:**
  - **<?PHP /* Instruction PHP */ et ?> ou <? /*Instruction PHP */ et ?>**
  - **<script language= "php" >.....< /script> ou <%..... %>**
  - ✓ **Tout ce qui se trouve entre ces balaises sera considéré par le serveur Web comme étant des commandes PHP à exécuter**
  - ✓ **Génère du texte intégré au contenu HTML**
- **Zones à l'extérieur de ces balises :**
  - **Contient du texte et des balises HTML**
  - **Directement recopié dans le contenu HTML généré**



HTML et PHP

- **Exemple**

```
<body>
<p>Ici du texte HTML
<? php
echo 'et la suite affichée par PHP' ;
?>
</p>
```

- **Insertions de commentaire en php**

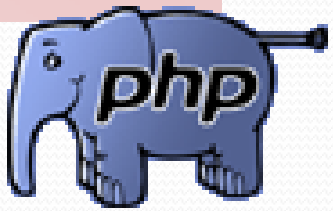
Commentaire Unilingue (Comme en C) : // commentaire1

Commentaire Multilingue (C++): /* commentaire2 */

Commentaire à L'Unix (moins utilisé) #Commentaire3

Affichage du texte

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="fr" lang="fr">
   <head>
      <title>Notre première instruction : echo</title>
      <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
   </head>
   <body>
      <h2>Affichage de texte avec PHP</h2>
      <p>
         Cette ligne a été écrite entièrement en (x)HTML.<br />
         <?php
               echo 'Hello world';
               print"Hello world";
              echo("Hello PHP");
              print("Bienvenue sur ma page PHP");
             ?>
      </p>
   </body>
```



Variables

Variables en PHP :

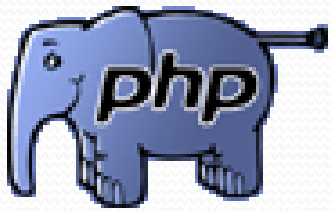
- Le nom d'une variable commence par un $
- Tout nom de variable doit commencer par une lettre (majuscule ou minuscule) ou un "_", mais jamais un chiffre.
- Sensible à la casse. Les espaces sont interdis
- Affectation comme en C :

  $i = valeur ;

  - Les variables ne sont pas explicitement déclarées comme en C (une variable existe dès qu'une valeur lui est affectée)
  - Une variable peut contenir un nombre, une chaîne de caractères, un booléen (en réalité un entier comme en C) ou un tableau
  - Il est possible de convertir une variable en un type primitif grâce au cast(comme en C).

Exemple :

```
$str = ”12”;          // $str vaut la chaîne ‘’12”
$nbr = (int)$str;     // $nbr vaut le nombre 12
```



Variables

Les variables référencées (&)

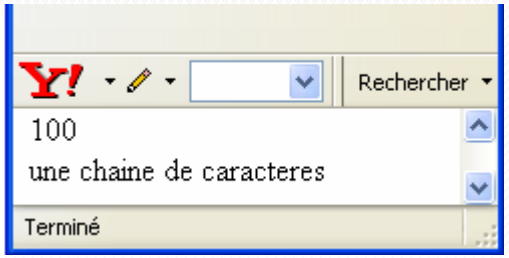
$nomcomplet = "Paul".&$nom; en PHP4

Exemple :

```
<?php
$foo = 'Pierre'; // Assigne la valeur 'Pierre' à $foo
$bar = &$foo; // Référence $foo avec $bar.
$bar = "Mon nom est Pierre"; // Modifie $bar...
echo $foo; // $foo est aussi modifiée
echo $bar;
?>
```

Le type d'une variable est à liaison superficielle

```
<?php
$a = 100;
echo $a;
$a = "une chaine de caracteres";
echo $a;
?>
```



Variables

Portée des variables

La portée d'une variable dépend du contexte dans lequel la variable est définie.
Les constantes, les fonctions et classes sont partout visible donc accessible.

Variable globales

- **Les variables globales ne peuvent être utilisées telles quelles au niveau du corps des fonctions.**
- **Une variable globale ne peut être au sein d'une fonction que si elle est précédée du mot réservé global.**

Variable locales

- **Elles sont définies au niveau du corps des fonctions**

Variables

Toute variable utilisée dans une fonction est par définition, locale.

Par exemple :

```
<?php
$a = 1;
function test() {
echo $a; /* portée locale */
}
test();
?>
```

Le script n'affichera rien à l'écran car la fonction echo utilise la variable locale $a, et celle–ci n'a pas été assignée préalablement dans la fonction.

Variables

En PHP, une variable globale doit être déclarée à l'intérieur de chaque fonction afin de pouvoir être utilisée dans cette fonction. Par exemple:

```
<?php
$a = 1;
$b = 2;
function somme()
{
global $a, $b;
$b = $a + $b;
}
somme();
echo $b;
?>
```

Le script ci–dessus va afficher la valeur "3". En déclarant globales les variables $a et $b locales de la fonction somme(), toutes les références à ces variables concerneront les variables globales. Il n'y a aucune limite au nombre de variables globales qui peuvent être manipulées par une fonction.

Variables

Une autre caractéristique importante de la portée des variables est la notion de variable static . Une variable statique a une portée locale uniquement, mais elle ne perd pas sa valeur lorsque le script appelle la fonction. Prenons l'exemple suivant:

```
<?php
function test() {
$a = 0;
echo $a;
$a++;
}
?>
```

```
<?php
function test() {
static $a = 0;
echo $a;
$a++;
}
?>
```

Cette fonction est un peu inutile car à chaque fois qu'elle est appelée, elle initialise $a à 0 et affiche "0". L'incrémentation de la variable ($a++) ne sert pas à grand chose, car dès que la fonction est terminée la variable disparaît.

Pour faire une fonction de comptage utile, c'est–à–dire qui ne perdra pas la trace du compteur, la variable $a est déclarée comme une variable statique (voir script au-dessus). L'appel de la fonction Test() affichera une valeur de $a incrémentée de 1.

Variables

- **Conversion de type**

La fonction gettype() permet de lire le type d'une variable.
On peut assigner les types suivants

- Array
- Class
- Double
- Integer
- String

La fonction settype() permet de changer le type d'une variable

```
<?php
$a = 3.14;
If(gettype($a)=="d
ouble")
settype($a,
"integer" );
?>
```


Variables, types et opérateurs

Opérateurs arithmétiques :
+(addition), - (soustraction), * (multiplié), / (divisé), % (modulo), ++ (incrément), --(décrément).

Opérateurs d'assignement :
= (affectation), *= ($x*=$y équivalent à $x=$x*$y), /=, +=, -=, %=

Opérateurs logiques :
and, && (et), or, || (ou), xor (ou exclusif), ! (non)

Opérateurs de comparaison :
== (égalité), < (inférieur strict), <= (inférieur large), >, >=, != (différence)

Un autre opérateur conditionnel est l'opérateur ternaire (":?"), qui fonctionne comme en langage C

```
<?php
(expr1) ? (expr2) : (expr3);
?>
```

Cette expression renvoie la valeur de l'expression expr2 si l'expression expr1 est vraie, et l'expression expr3 si l'expression expr1 est fausse.

Variables, types et opérateurs

	Opérateur	Exemple	Résultat si $bar = 3
=	affectation	`$foo = $bar`	3
+	addition	`2+3+$bar;`	8
-	soustraction	`2-3-$bar;`	-2
*	multiplication	`2*3*$bar;`	18
/	division	`$bar/2;`	1.5
%	modulo	`$bar%2`	1
.	concaténation	`$bar." ok "`	"3 ok"

Variables, types et opérateurs

	Opérateur	Exemple	Résultat
==	égalité de valeur	"3" == 3	TRUE
===	égalité valeur et type	"3" === 3	FALSE
<	inférieur strict	"3" < 3	FALSE
<=	inférieur ou égal	"3" <= 3	TRUE
>	supérieur strict	"3" > 3	FALSE
>=	supérieur ou égal	"3" >= 3	TRUE
!=	différent (valeur)	"3" != 3	FALSE
!==	différent (valeur ou type)	"3" !== 3	TRUE

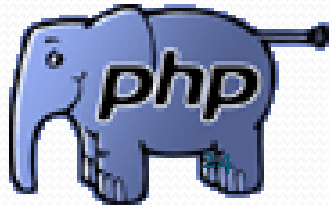
Variables, types et opérateurs

Quelques fonctions agissants sur les variables :

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()**, **is_bool()**, **is_float()**, **is_numeric()**, **is_integer()**, **is_int()**...

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
```



Constantes

L'utilisateur peut définir des constantes dont la valeur est fixée une fois pour toute. Les constantes ne portent pas le symbole **$** (dollars) en début d'identificateur et ne sont pas modifiables.

- **Define("var",valeur)** : définit la constante **var** (sans **$**) de valeur **valeur**

Exemple 1 :

- **define("author", "Maalouf");**
- **echo author;**

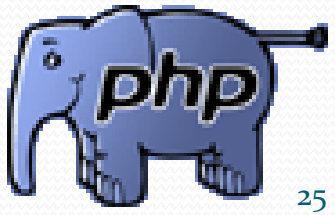
// affiche **Maalouf**

Exemple 2 :

- **define("MY_YEAR",1980);**
- **echo MY_YEAR;**

// affiche **1980**

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



Chaînes de caractères (I)

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

Exemples :

$nom = ''Mohammed'';

$prenom = 'Ali';

- Les doubles quottes 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: Etiévant**

echo 'Nom: $nom';

// affiche **Nom: $nom**

- Quelques caractères spéciaux : **\n** (nouvelle ligne), **\r** (retour à la ligne), **\t** (tabulation horizontale), \\ (antislash), **\$** (caractère dollars), \'' (double quotte) voir page tableau p96.

Exemple :

echo ''Hello Word !\n'';

Chaînes de caractères (II)

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

- Exemple 1 :

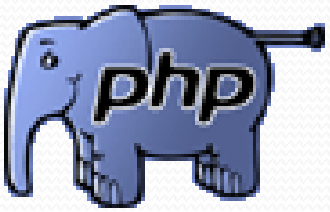
```
$foo = ‘’Hello”;
$bar = ‘’Word”;
 echo $foo.$bar;
```

- Exemple 2 :

```
$name = ‘’Henry”;
$whoiam = $name.“IV”;
```

- Exemple 3 :

```
$out = ‘Patati’;
$out .= ‘’ et patata…”;
```



Chaînes de caractères (III)

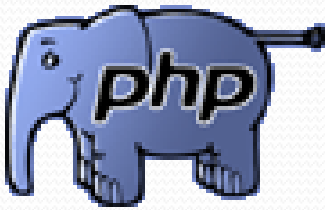
- Affichage d'une chaîne avec **echo**:

•**Exemples:**

```
echo 'Hello Word.';
echo "Hello ${name}\n";
echo "Nom : ", $name;
echo("Bonjour");
```

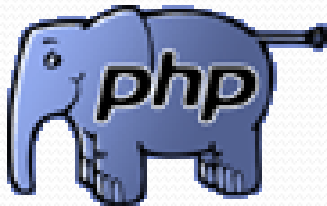
- **Quelques fonctions agissants sur les chaines de carctères:**

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**



Les tableaux

- Les tableau (array) représentent la structure de données la plus importante du langage PHP
- Les tableaux sont en PHP d'une seule dimension
- La seule manière de construire des tableaux multidimensionnel est d'utiliser des tableaux de pointeurs.
- L'indexation d'un tableau commence toujours à l'indice 0 (zéro)
- Deux types d'indexation:
  - indexage numérique
  - indexage associatif



Les tableaux

```php
<?php
// Création (facultatif)
$montab = array();
// Initialisations
$montab[0] = 1;
$montab[1] = 32.5;
$montab[2] = "Hello World";
// Tableaux à plusieurs dimensions
$montab[3] = array();
$montab[3][0] = 1;
$montab[3][2] = "Hello World";
// Initialisation sans connaître le numéro
$montab[] = 32; // Équivalent à $montab[4] = 32;
// Création et initalisation
$tableau2 = array(1,32.5,"Hello World",32);
?>
```


Les tableaux

Indexation numérique

Indexation explicite

```
$couleur[0] = "blanc", $couleur[1] = "bleu", $couleur[2] = "rouge";
```

Indexation automatique

```
$couleur[] = "blanc", $couleur[] = "bleu", $couleur[] = "rouge";
```

- Avec array sans indice

```
$couleur = array("blanc", "bleu", "rouge");
```

- Avec array + indice

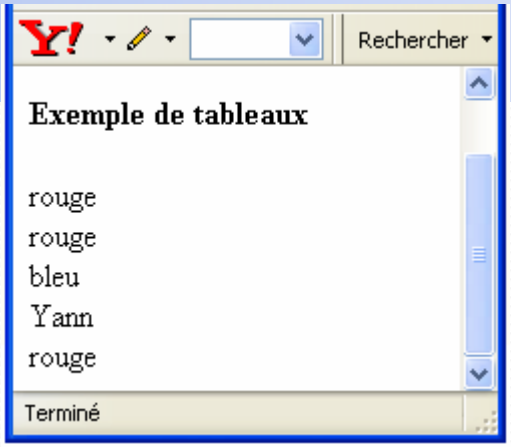
```
$couleur = array(5=>"blanc", "bleu",  "rouge");
```

Indexation associative

```
$personne = array("nom" =>« Moamed", "prenom"=>« Ali");
$couleur = array(1=>"blanc", 3=>"bleu", 5=>"rouge");
```

Les tableaux

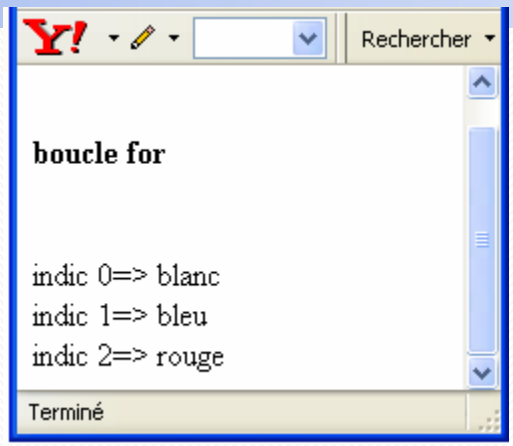
```php
<?php
$couleur[] = "blanc"; $couleur[] = "bleu"; $couleur[] = "rouge";
echo "$couleur[2]";
$couleur2 = array("blanc", "bleu", "rouge");
echo "<br /> $couleur2[2]";
$couleur3 = array(5=> "blanc", "bleu", "rouge");
echo "<br /> $couleur3[6]";
$personne = array("nom"=> "NSIRI", "prenom"=>  "Yann");
echo "<br />$personne[prenom]";
$couleur4 = array(1=> "blanc", 3=> "bleu", 5=> "rouge");
echo "<br />$couleur4[5]";
?>
```



Les tableaux

Parcours d'un tableau : for

```
<?php
$couleur[] = "blanc"; $couleur[] = "bleu"; $couleur[] = "rouge";
for($index=0; $index < count($couleur); $index++)
{
echo "<br />indic $index=> $couleur[$index]";
?>
```



Les tableaux

- Tableaux multidimensionnels

```php
<?php
//création d'un tableau contenant les villes du Maroc
$villes_maroc[]= "Rabat";
$villes_maroc[]= "Temara";
$villes_maroc[]= "Casablanca";
// stockage du tableau des villes de Maroc dans le tableau des villes
$villes["Maroc"]=$villes_maroc;
//tableau de tableau
//viles de France
$villes_france[]= "Paris";
$villes_france[]= "Brest";
$villes_france[]= "Quimper";
// stockage du ville de France
$villes["France"]=$villes_france;
?>
```

Méthode 2

```php
<?php
//villes du maroc
$villes["Maroc"][]= "Rabat";
$villes["Maroc"][]= "Temara";
$villes["Maroc"][]= "Casablanca";
//villes de france
$villes["France"][]= "Paris";
$villes["France"][]= "Brest";
$villes["France"][]= "Quimper";
?>
```

Les tableaux

Tableau multidimensionnels

En utilisant la fonction array:

Méthode 1

```
<?php
$villes_maroc[]= array("Rabat", "Temara", "Casablanca");
$villes_france[]= array("Paris", "Brest", "Quimper");
$villes=array("Maroc"=>$villes_maroc,"France"=>$villes_france);
?>
```

Méthode 2

```
<?php
$villes=array("Maroc"=>array("Rabat", "Temara", "Casablanca"),
"France"=>array("Paris", "Brest", "Quimper"));
?>
```

Les tableaux

La structure de parcours de tableau foreach

La forme foreach **reste la plus adaptée pour passer en revue les éléments d'un tableau. Il y a deux syntaxes possibles :**

Première syntaxe :

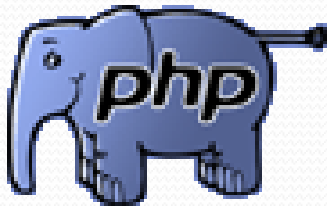
```
<?php
foreach($tableau as $value)
instructions;
?>
```

A chaque itération de la boucle, $valeur **contient la valeur de l'élément du tableau** $tableau

. Cette syntaxe permet de parcourir le tableau de début à la fin.

Deuxième syntaxe

```
<?php
foreach($tableau as $clé=>$value)
instructions;
?>
```



Les formulaires

- Utilisés pour la gestion interactive d'un site.
- Sont à la base des pages web dynamiques

Mise en œuvre

- Un formulaire XHTML est défini entre les balises <form> et </from>

Exemple:

```
<form method="post" action="traitement.php">
<p>Texte à l'intérieur du formulaire</p>
</form>
```

- **method** : Mode de transmission vers le serveur des informations saisies dans le formulaire.
- **action** : le script qui va traiter le formulaire.

- **traitement.php**: page réceptrice
- **Get** : les données du formulaire sont transmises dans l'URL.
- **Post** : les données du formulaire sont transmises dans le corps de la requête.

Les formulaires

- **Les méthodes Get et Post**

La transmission d'un formulaire s'effectue selon l'une des deux méthodes d'envoi : GET **ou** POST.

- GET **retrouve les variables au sein de la** superglobale $_GET
- POST **retrouve les variables au sein de la** superglobale $_POST.
- **Pour obtenir les valeurs des variables en utilisant les superglobale, procédez comme suit :**
- Syntaxe

$valarg =$_GET['nomarg']; //GET

$valarg =$_POST['nomarg']; //POST

Les formulaires

- **L'élément *input* permet de créer les différents types d'éléments de formulaire:**
  "text", "password", "checkbox", "radio", "submit", "reset", "file" et "hidden

- **Les champs de saisie simple**

```
<form action= " champs_simple.php" method="post" >
<p>
<input type= " text «   name= " nom_du_champs"
   value="valeur_par_defaut"  size=50>
<p>
</form>
```

- **Les champs de saisie de texte long**

```
<form action= " champs_simple.php" action="post" >
<textarea name= "nom_du_champs"  cols= " 3 "rows= " 4" un texte par
   défaut>
</textarea>
</form>
```

Les formulaires

- **Les listes de choix**

```
<form action="traitement.php">
<p>
<select name= "liste_choix" size="2">
<option value= "option_1" >option_1</option>
<option value= "option_2" > option_2</option>
</select>
<p>
</form>
```

Les boutons radio

```
<form action="input_radio.php">
<p>Donnez votre mode de paiement:</p>
<p>
<input type="radio" name="mode_paiement" value="Mastercard"> Mastercard<br>
<input type="radio" name="mode_paiement" value="Visa"> Visa<br>
<input type="radio" name="mode_paiement" value="AmericanExpress"> American
    Express
</p>
</form>
```

Les formulaires

- Les cases a cocher

```
<form action=" input_checkbox.php">
<p>
<input type="checkbox" name="composant" value="salami"> Salami<br>
<input type="checkbox" name="composant" value="champignons">
    champignons<br>
<input type="checkbox" name="composant" value="anchois"> anchois
</p>
</form>
```

- Boutons file

```
<form enctype="multipart/form-data" action="TP2.php" method="post">
<input type="hidden" name="MAX_FILE_SIZE"size="100000" />
Transfère le fichier <input type="file" name="monfichier"
    accept="text/html" />
<input type="submit" />
</form>
```


Les formulaires

- password

```
< form action ="TP2.php" method="post">
<input type="password" name="pw" size="5" maxlength="5"/>
<input type="submit" />
</ form >
```

- Les boutons submit et cancel

```
<form action=" bouton_submit.php">
<p>
<input type= " submit" name= " Valider" value= " Valider" >
</p>
</form>
<form action=" bouton_annuler.php">
<p>
<input type="reset"  name="Annuler" value="Annuler" >
</p>
</form>
```

Les formulaires

- **La méthode Get**
  - **La méthode** GET **envoie les données sous forme d'une suite de couples** nom/valeur **ajoutés à l'URL de la page appelée.**
  - **La partie d'une** URL **précédée par le caractère point d'interrogation** (?) **est appelée chaîne de requête. Si la chaîne de requête contient plusieurs éléments, alors chaque élément/valeur doit être séparé par le caractère** &
  - **Par ailleurs, elle ne peut pas dépasser 255 caractères. Les données transmises au serveur par la méthode** GET **sont visibles par les utilisateurs directement dans la barre d'adresse du navigateur.**

Exemple:

http://www.monsite.com/infos.php?jour=27&mois=07&annee=2003&titre=Informations

4 variables seront créées :

```
$_GET['jour'] = 27;
$_GET['mois'] = 07;
$_GET['annee'] = 2003;
$_GET['titre'] = "Informations";
```

Les formulaires

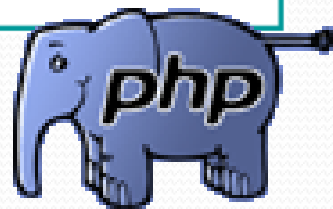
- Exemple

infos.php

```
<html><body>
<p>Lien vers la page appel.php, avec des variables aux valeurs
    différentes: </p>
<p><a href="appel.php?nom=Yossef&amp;prenom=Amine">Lien vers
    cible.php?nom=Yossef&prenom=Amine</a></p>
</body></html>
```

appel.php

```
<p>Bonjour !</p>
<p>Votre nom est <?php echo $_GET['nom']; ?> , et votre prénom est <?php
    echo $_GET['prenom']; ?>. </p>
<p>Faites un autre essai, <a href="infos.php">cliquez ici</a> pour revenir à
    infos.php</p>
```



Les formulaires

- **La méthode Post:**

Place les informations directement à la suite de l'adresse URL de la page appelée.

La partie d'une URL regroupe les informations dans l'en-tête d'une requête HTTP

Ainsi, les données transmises par un formulaire restent confidentielles et n'apparaissent pas dans l'URL. La fonction isset() est très pratique lorsqu'on écrit des traitements de formulaires. Elle permet de déterminer si une variable est affectée (o compris mais ni NULL ni FALSE). En utilisant cette fonction, il est possible de déterminer les champs d'un formulaire n'ayant pas été renseignés par l'utilisateur. Mais isset() présente une difficulté : le test d'une chaîne de caractères vide renvoie TRUE

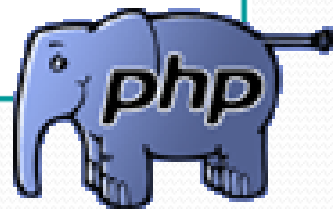
Les formulaires

- Exemple

```
<form action="cible.php" method="post">
<label>Entrer votre nom:
<input type="text" name="prenom" /> <input type="submit" value="Valider"
/>
</label></form>
```

- **Cible.php**

```
<p>Bonjour !</p>
<p>Je sais comment tu t'appelles. Tu t'appelles <?php echo
    $_POST['prenom']; ?> !</p>
<p>Si tu veux changer de prénom, <a href="form.php">clique ici</a> pour
    revenir à form.php</p>
```



Les formulaires

- **LE FORMULAIRE ET LE SCRIPT PHP**

PHP peut intervenir à deux endroits par rapport au formulaire :

- Pour la construction du formulaire, si ce dernier doit contenir des informations dynamiques ;
- Pour le traitement du formulaire ;

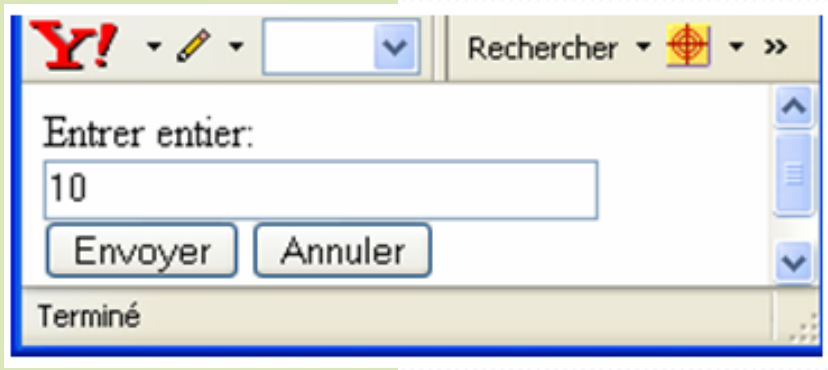
Les méthodes utilisables pour faire interagir un formulaire et un script PHP sont :

- Placer le formulaire dans un document xHTML, dans ce cas le formulaire ne contient aucun élément dynamique, et indiquer le nom de script qui doit traiter le formulaire dans l'option action de la balise <form>
- Placer le formulaire dans un script PHP et faire traiter le formulaire par un autre script PHP ;
- Placer le formulaire dans un script PHP et le faire traiter par le même script PHP.

Les formulaires

Exemple:

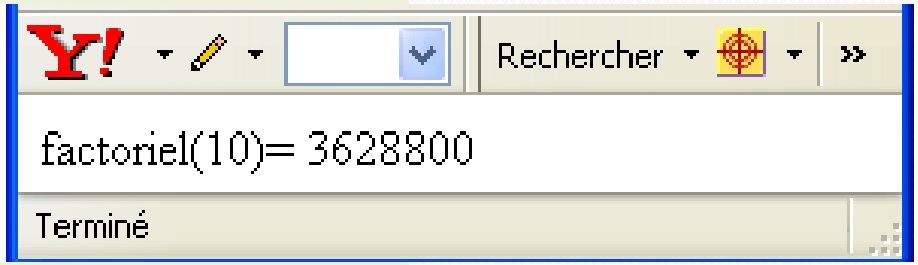
```
<?php
echo ("<html>");
echo ("<head>");
echo ("<title> calcul du factoriel </title>");
echo ("</head>");
echo ("<body>");
echo ("<head>");
echo ("<form method=\"post\"
    action=\"traitement.php\">");
echo("<label> Entrer entier:
<input type=\"text\" name=\"nombre\" size=\"30\"/>
</label>");
echo("<input type=\"submit\" /> <input type=\"reset\" />");
echo("</form>");
echo("</html>");
?>
```



Les formulaires

Traitement.php

```
<?php
function factoriel($n){
        if($n==0)  return 1;
        else  return $n*(factoriel($n-1));
}
$var=$_POST['nombre'];
print("factoriel($var)=");
print(factoriel($var));
?>
```



Les formulaires

Exemple: Variables de formulaires complexes

```
<?php
if (isset($_POST['action']) && $_POST['action'] == 'submitted')
{
echo '<pre>';
print_r($_POST);
echo '<a href="". $_SERVER['PHP_SELF'] .">Essayez à
   nouveau</a>';
echo '</pre>';
}
else
{
}
?>
```

Les instructions de contrôle

- Structure if. . .else

Structure conditionnelle similaire à celle de C :

- Si conditions renvoient true, exécution des instructions
- dans le bloc du if
- Sinon, on exécute les instructions dans le bloc du else
- *Instruction else facultative*

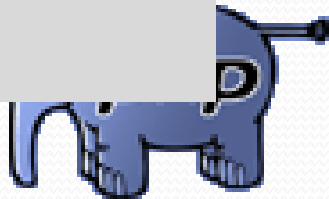
Syntaxe de if ... else ...

```
if(conditions) {
instructions pour conditions vérifiées
}
[ else {
instructions pour le cas où aucune condition n'a été vérifiée
} ]
```

Les instructions de contrôle

```
<?php
$cat="PC";
$prix=4900;
if($cat=="PC")
{
if($prix >= 1000)
{
echo "<b>Pour l'achat d'un PC d'un montant de
    $prix &euro;, la remise est de
15 %</b><br />";
echo "<h3> Le prix net est de : ",$prix*0.85, " DH;
    </h3>";
}
else
{
echo "<b>Pour l'achat d'un PC d'un montant de
    $prix &euro;, la remise est de
10 %</b><br />« ;
```

```
echo "<h3> Le prix net est de : ",$prix*0.90, " DH;
    </h3>";
}
}
elseif($cat=="Livres")
{
echo "<b>Pour l'achat de livres la remise est de 5
    %</ b><br />";
echo "<h3> Le prix net est de : ",$prix*0.95, " DH;
    </h3>";
}
else
{
echo"<b>Pour les autres achats la remise est de 2
    %</ b><br />";
echo "<h3> Le prix net est de : ",$prix*0.98, " DH;
    </h3>";
}
?>
```



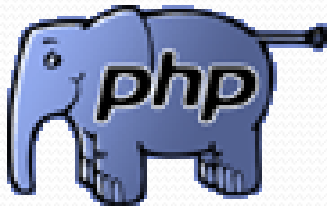
Les instructions de contrôle

Les boucles

- Permettent de répéter un bloc d'instructions tant qu'une condition est vérifiée
- Utile pour parcourir un tableau, ou lire un fichier, etc.

Trois types de boucles en PHP :

- while
- for
- foreach



Les instructions de contrôle

Boucle while

La boucle while (“tant que”) exécute les instructions tant que les conditions sont vérifiées

Syntaxe du while

```
while(conditions) {
instructions
}
```

Exemple

```
<?php
do
{
$n = rand(1,100);
echo $n,"  / ";
}
while($n%7!=0);
?>
```

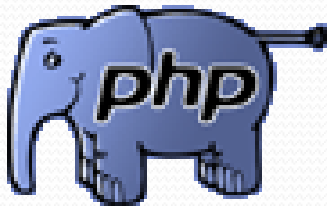
Vérification si un nombre n est premier ?

Les instructions de contrôle

Boucle while

La boucle while (“tant que”) exécute les instructions tant que les conditions sont vérifiées

```
<?php
do
{
$n = rand(1,100);
echo $n,"  / ";
}
while($n%7!=0);
?>
```

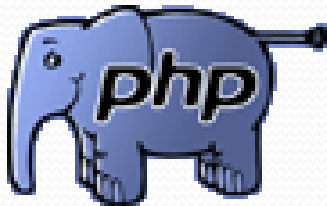


Les instructions de contrôle

Boucle while

La boucle while (“tant que”) exécute les instructions tant que les conditions sont vérifiées

```
<?php
do
{
$n = rand(1,100);
echo $n,"  / ";
}
while($n%7!=0);
?>
```



Les instructions de contrôle

Boucle for

La boucle for (“pour”) exécute les instructions tant que les conditions sont vérifiées. L’étape initialisation est exécutée une fois en début de boucle. Si la condition est vérifiée, instructions et itération sont exécutées, puis si la condition est à nouveau vérifiée, instructions et itération sont exécutées, . . .

Syntaxe de la boucle for

```
for(initialisation ; conditions ; itération) {
instructions
}
```


Les instructions de contrôle

Exemple boucle for

```
<?php
echo "<h2> Révisez votre table de multiplication!</ h2>";
//Début du tableau HTML
echo "<table border=\"2\" style=\"background-color:yellow\"> <th>  X  </th>";
//Création de la première ligne
for($i=1;$i<10;$i++)
{
echo "<th> $i </th>";
}
//Fin de la boucle 1
//*****************************
//Création du corps de la table
//*****************************
```

```
//Boucles de création du contenu de la table
for($i=1;$i<10;$i++)
{
//Création de la première colonne
echo "<tr><th> $i </th>";
//Remplissage de la table
for($j=1;$j<10;$j++)
{
echo "<td style=\"background-color:red;color:white\">    <b>". $i*$j.
"   </td>";
}
echo "</b></tr>";
}
echo "</table>"
?>
```

Les instructions de contrôle

Boucle foreach

La boucle foreach (“pour chaque”) permet de parcourir des tableaux vus comme un ensemble de valeurs

Syntaxe de la boucle foreach

```
for($tableau as $valeur) {
instructions
}
```

Voir exemple foreach2.php

Exemple de la boucle foreach

```
<?php
//Création du tableau de 11 éléments
for($i=0;$i<=10;$i++)
{
$tab[$i] = pow(2,$i);
}
$val ="Une valeur";
echo $val,"<br />";
//Lecture des valeurs du tableau
echo"Les puissances de 2 sont :";
foreach($tab as $val)
{echo $val." : ";}
?>
```

Les instructions de contrôle

L'opérateur ?

Syntaxe

```
$var = expression ? valeur1 : valeur2
```

Elle est équivalente à :

```
if(expression) {$var=valeur1;}
else {$var=valeur2;}
```

Exemple

```
<?php
$prix=150;
$var = ($prix>100)? "la remise est de 10 %":"la remise est de 5 %";
echo "<b>Pour un montant d'achat de $prix &euro;: $var </b><br />";
?>
```

Les instructions de contrôle

L'opérateur ?

Exemple 2

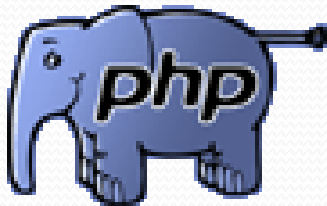
```
<?php
$ch = "Bonjour ";
$sexe="M";
$ch .= ($sexe=="F")?"Madame":"Monsieur";
echo "<h2>$ch</h2>";
$nb = 3;
$pmu ="Il faut trouver ".$nb;
$mot = ($nb==1)?" cheval":" chevaux";
echo "<h3> $pmu $mot </h3>";
?>
```


Les instructions de contrôle

L'instruction switch…case

Syntaxe

```
switch(expression)
{
case valeur1:
//bloc d'instructions 1;
break;
case valeur2:
//bloc d'instructions 2;
break;
……..
case valeurN:
//bloc d'instructions N;
break;
default:
//bloc d'instructions par défaut;
break;
}
```

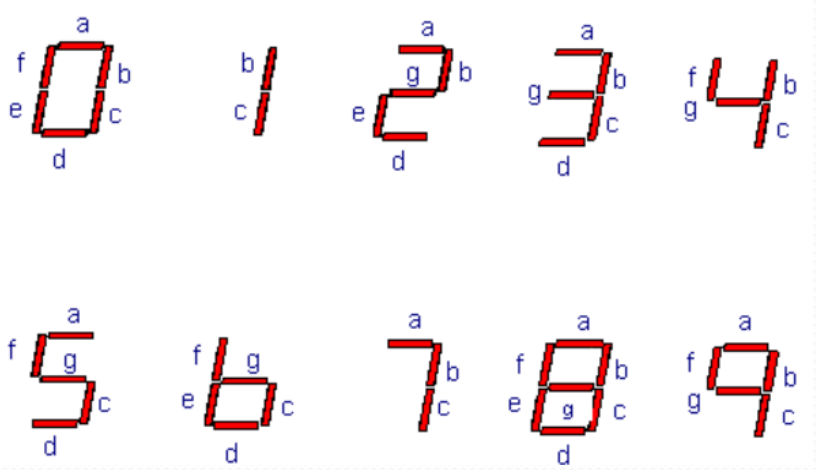



Les instructions de contrôle

L'instruction switch…case

Exercice d'application

Afficheur è segments



Nombre décimal	Code BCD				Sorties						
	D	C	B	A	a	b	c	d	e	f	g
0	L	L	L	L	ON	ON	ON	ON	ON	ON	OFF
1	L	L	L	H	OFF	ON	ON	OFF	OFF	OFF	OFF
2	L	L	H	L	ON	ON	OFF	ON	ON	OFF	ON
3	L	L	H	H	ON	ON	ON	ON	OFF	OFF	ON
4	L	H	L	L	OFF	ON	ON	OFF	OFF	ON	ON
5	L	H	L	H	ON	OFF	ON	ON	OFF	ON	ON
6	L	H	H	L	OFF	OFF	ON	ON	ON	ON	ON
7	L	H	H	H	ON	ON	ON	OFF	OFF	OFF	OFF
8	H	L	L	L	ON	ON	ON	ON	ON	ON	ON
9	H	L	L	H	ON	ON	ON	OFF	OFF	ON	ON

Les instructions de contrôle

La boucle while

Syntaxe

```
while(expression)
{
//Bloc d'instructions à répéter;
}
```

La boucle while permet d'exécuter un bloc d'instructions tant qu'une conditions est vérifiée

Exemple

```
<?php
do
{
$n = rand(1,100);
echo $n,"  / ";
}
while($n%7!=0);
?>
```

Les instructions de contrôle

La boucle do … while

Syntaxe

```
do {
//bloc d'instructions
}
while(expression);
```

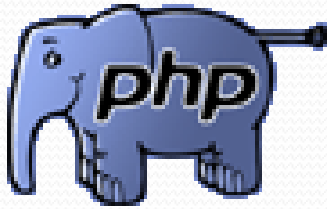
La boucle do… while permet d'exécuter un bloc d'instructions tant qu'une conditions est vérifiée

Exemple

```
<?php
do
{
$n = rand(1,100);
echo $n,"  / ";
}
while($n%7!=0);
?>
```

Sortie anticipée des boucles

- **L'instruction *break*:** permet d'arrêter une boucle **for, foreach** ou **while** avant son terme normal
- ***break n*; les n boucles internes seront arrêtées.**
- **L'instruction *continue:* n'arrête pas la boucle en cours, mais les instructions situées après Continue ne seront pas exécutées.**



Fonctions

1 Introduction

PHP offre la possibilité de définir des fonctions avec tous les avantages associés.

Les fonctions en PHP peuvent prendre des arguments sans spécifier leurs types.

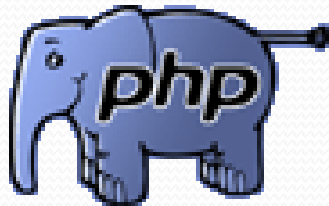
Elles peuvent de façon optionnelle retourner une valeur.

2 Déclaration et appel

Le mot clé function permet d'introduire la définition d'une fonction qui peut être définie selon la syntaxe suivante :

```
<?php
function nom_function($paramètre0, $paramètre1, $paramètre2, ...)
{
instructions ;
}
?>
```

nom_function doit respecter les règles des noms de variable



Fonctions

```
<?php
// fonction avec 2 paramètres retourne la somme des deux paramètres
function Somme($a, $b) {
return $a+$b;
}
$res = Somme(10, 11); echo "$res= " , $res;
//fonction sans paramètre qui affiche “Ceci est un exemple“
function Afficher_message(){
echo "ceci est un message <br />";
}
Afficher_message();
?>
```

Fonctions

Valeurs de retour

Il est possible de retourner plusieurs valeurs d'une fonction sous forme d'un tableau. Dans l'appel de cette fonction, il faudra affecter le tableau retourner à la Procédure list() qui prend en paramètre la taille de ce tableau. On affecte à list() le retour de la fonction

```
<?php
function opération($arg1,$arg2){
return array ($arg1+$arg2, $arg1-$arg2, $arg1*$arg2 ) ;
}
$a=5 ; $b=3 ;
list($a1,$a2,$a3)= opération($a,$b) ;
echo " somme : $a1 <br />" ;
echo " soustraction : $a2 <br />" ;
echo " produit : $a3 <br />" ;
?>
```

Fonctions

Visibilité de la fonction

Une fonction est utilisable uniquement dans le script où elle est définie. Pour l'employer dans plusieurs scripts, il faut, soit recopier sa définition dans les différents scripts, soit la définir dans un fichier inclus partout où la fonction est nécessaire.

Exemple :

Fichier fonctions.inc contenant des définitions de fonctions :

```
<?php
function somme($arg1,$arg2){
return $arg1+$arg2;
}
?>
```

Script utilisant les fonctions définies dans fonctions.inc :

```
<?php
Include 'fonctions.inc' ; //inclusion du fichier fonctions.inc
echo somme(3,3) ; //utilisations de la fonction  somme
?>
```

Mathématiques (I)

Il existe une miriade de fonctions mathématiques.

abs($x) : valeur absolue

ceil($x) : arrondi supérieur

floor($x) : arrondi inférieur

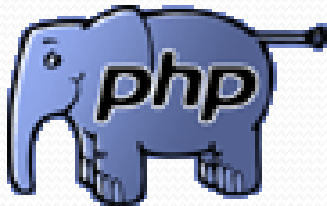
pow($x,$y) : x exposant y

round($x,$i) : arrondi de x à la ième décimale

max($a, $b, $c ...) : retourne l'argument de valeur maximum

pi() : retourne la valeur de Pi

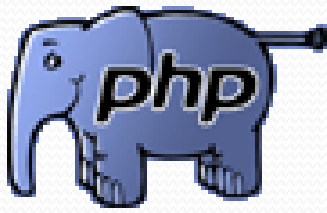
Et aussi : **cos**, **sin**, **tan**, **exp**, **log**, **min**, **pi**, **sqrt**...



Mathématiques (II)

Nombres aléatoires :

- **rand([$x],[$y])** : valeur entière aléatoire entre 0 et RAND_MAX si x et y ne sont pas définis, entre x et RAND_MAX si seul x est défini, entre x et y si ces deux paramètres sont définis.
- **srand()** : initialisation du générateur aléatoire
- **getrandmax()** : retourne la valeur du plus grand entier pouvant être généré
- L'algorithme utilisé par la fonction **rand()** - issu de vieilles bibliothèques libcs - est particulièrement lent et possède un comportement pouvant se révéler prévisible. La fonction **mt_rand()** équivalente à **rand()** est plus rapide et plus sûre puisque l'algorithme utilisé se base sur la cryptographie.
- En cas d'utilisation de la fonction **mt_rand()**, ne pas oublier d'utiliser les fonctions de la même famille : **mt_rand([$x[,$y]])**, **mt_srand()** et **mt_getrandmax()**.



Dates et calendriers

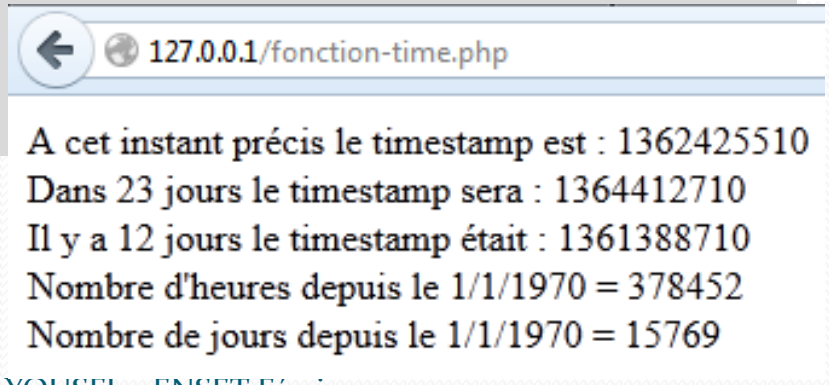
- **Les fonctions de jours, dates et heures** sont incontournables sur Internet et sont indispensables pour la conversion en français des dates fournies par la base de données MySQL qui les code au format anglophone **(YYYY-DD-MM hh:mm:ss).**
- **Quelques fonctions :**
- **time():** retourne le timestamp UNIX de l'heure locale (utilisée pour calculer des durées et determiner des dates future ou passées.
- **Date():** retourne une chaîne de caractères contenant la date et/ou l'heure locale au format spécifié.
- **getdate() , checkdate($month, $day, $year), mktime($hour, $minute, $second, $month, $day, $year)**

Dates et calendriers

time()

- Retourne le timestamp de l'instant présent
- Cette valeur n'est affichée
- Sert pour le calcul du temps
- Sert à stocker une date à un seul nombre

```
<?php
echo " A cet instant précis le timestamp est : ", time(), "<br />";
echo "Dans 23 jours le timestamp sera : ", time()+23*24*3600,   "<br />";
echo "Il y a 12 jours le timestamp était : ", time()-12*24*3600,"<br />";
echo"Nombre d'heures depuis le 1/1/1970 = ",round(time()/ 3600),"<br />";
echo"Nombre de jours depuis le 1/1/1970 = ",round(time()/3600/ 24),"<br />";
?>
```



Dates et calendriers

Définir une date mktimee()

- Syntaxe:

int mktime(int heure, int minute, int seconde, int mois, int jour, int année, int été)

La dernier paramètre prend 1 pour l'heure d'hiver et 0 pour l'été.

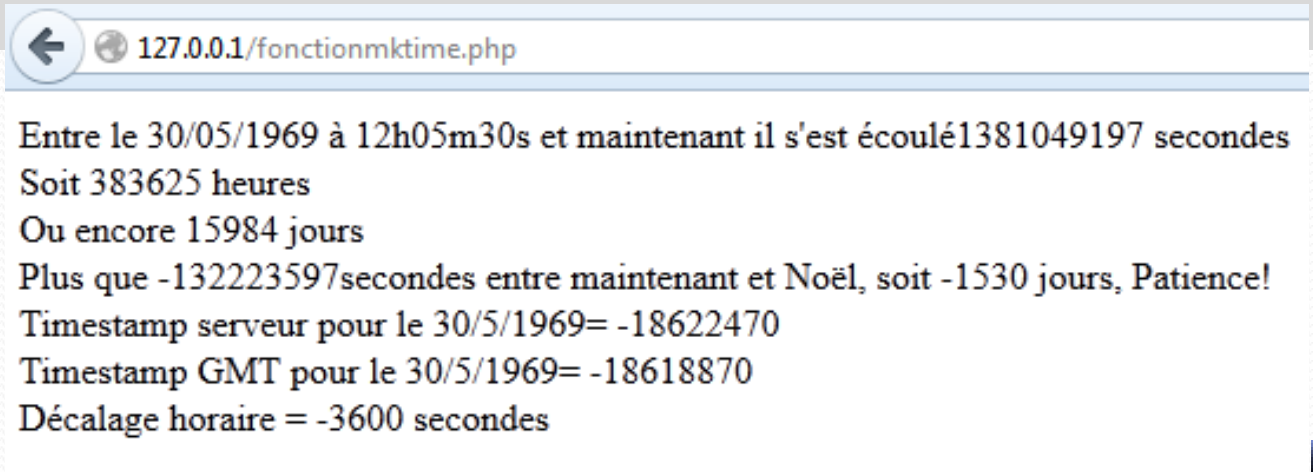
Exemple d'application

```
<?php
//la fonction mktime()
$timepasse= mktime(12,5,30,5,30,1969);
$timeaujour = time();
$duree = $timeaujour-$timepasse;
echo "Entre le 30/05/1969 à 12h05m30s et maintenant il s'est
écoulé",$duree,  " secondes <br />";
echo "Soit ",round($duree/3600), " heures <br />";
echo "Ou encore ",round($duree/3600/24)," jours <br />";
```

Dates et calendriers

Définir une date mktimee()

```
$timefutur = mktime(12,5,30,12,25,2008);
$noel = $timefutur-$timeaujour;
echo "Plus que ",$noel, "secondes entre maintenant et Noël, soit ",
    round($noel/3600/24)," jours, Patience! <br />";
//la fonction gmmktime()
$timepassegmt = gmmktime(12,5,30,5,30,1969);
echo "Timestamp serveur pour le 30/5/1969= ",$timepasse,"<br />";
echo "Timestamp GMT pour le 30/5/1969= ",$timepassegmt,"<br />";
echo "Décalage horaire = ",$timepasse-$timepassegmt," secondes<br />";
?>
```



Dates et calendriers

Formulaire de vérification de date

Vérifier une date

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
"http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="fr">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
<title>Validation de date</title>
</head>
<body>
<form method="post" action="<?= $_SERVER["PHP_SELF"] ?>" >
<fieldset>
<legend>Entrez votre date de naissance sous la forme JJ/MM/AAAA</legend>
<input type="text" name="date" />
<input type="submit" value="Envoi"/>
</fieldset>
</form>
<?php
//checkdate
```

Dates et calendriers

```
if(isset($_POST["date"]))
{
$date=$_POST["date"];
$tabdate=explode("/",$date);
if(!checkdate($tabdate[1],$tabdate[0],$tabdate[2]) ) {echo "<hr />
➥La date $date n'est pas valide. Recommencez! <hr />";}
else {echo "<h3> La date $date est valide. Merci!</h3>";}
}
?>
</body>
</html>
```

Dates et calendriers

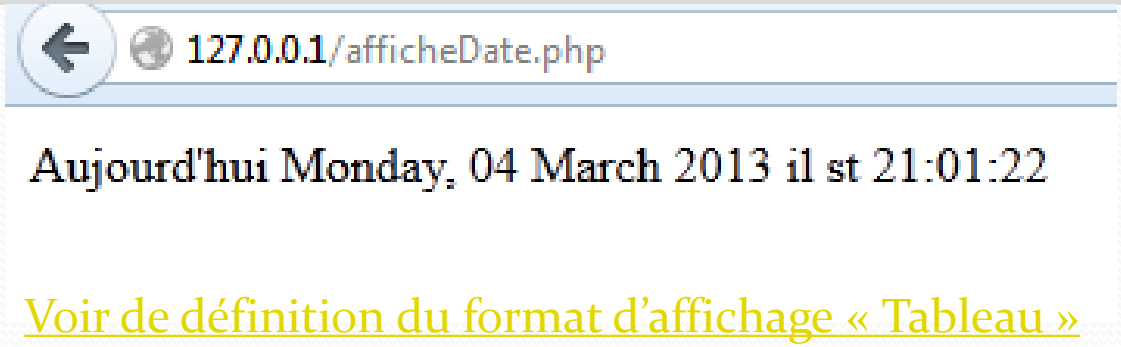
Date()

- Syntaxe:

string date(string format_de_date,[int timestamp])

- Exemple:

```
<?php
echo "Aujourd'hui ",date("l, d F Y \i\l \e\s\\t H:i:s ");
?>
```



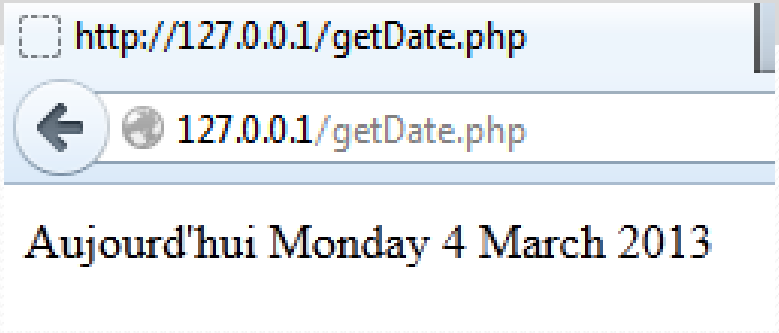
Voir de définition du format d'affichage « Tableau »

Dates et calendriers

Fonction getdate()

- Retourne un tableau d'informations sur la date.
- Syntaxe: arry getdate([int timestamp])
- Exemple:

```
<?php
$jour = getdate();
echo "Aujourd'hui {$jour["weekday"]} {$jour["mday"]} {$jour["month"]}
    {$jour["year"]}";
?>
```

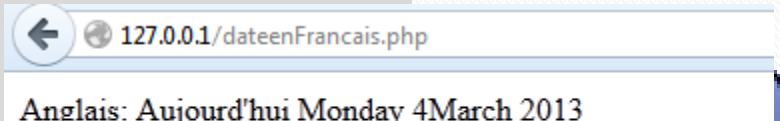


Dates et calendriers

Date en français

- Créer deux tableaux indexés des et jours mois en français.

```
<?php
//Date en français
$jour = getdate();
echo "Anglais: Aujourd'hui {$jour["weekday"]}
{$jour["mday"]}{$jour["month"]}
 {$jour["year"]} <br />";
$semaine = array(" dimanche "," lundi "," mardi "," mercredi "," jeudi ",
                 " vendredi "," samedi ");
$mois =array(1=>" janvier "," février "," mars "," avril "," mai "," juin ", "
juillet ",       " août "," septembre "," octobre "," novembre "," décembre ");
//Avec getdate()
echo "Français: Avec getdate() : Aujourd'hui ",$semaine[$jour['wday']] ,
$jour['mday'],
                  $mois[$jour['mon']], $jour['year'],"<br />";
//Avec date()
echo " Français: Avec date() : Aujourd'hui ", $semaine[date('w')] ,"
",date('j'),"
                 ", $mois[date('n')], date('Y'),"<br />";
?>
```



Dates et calendriers

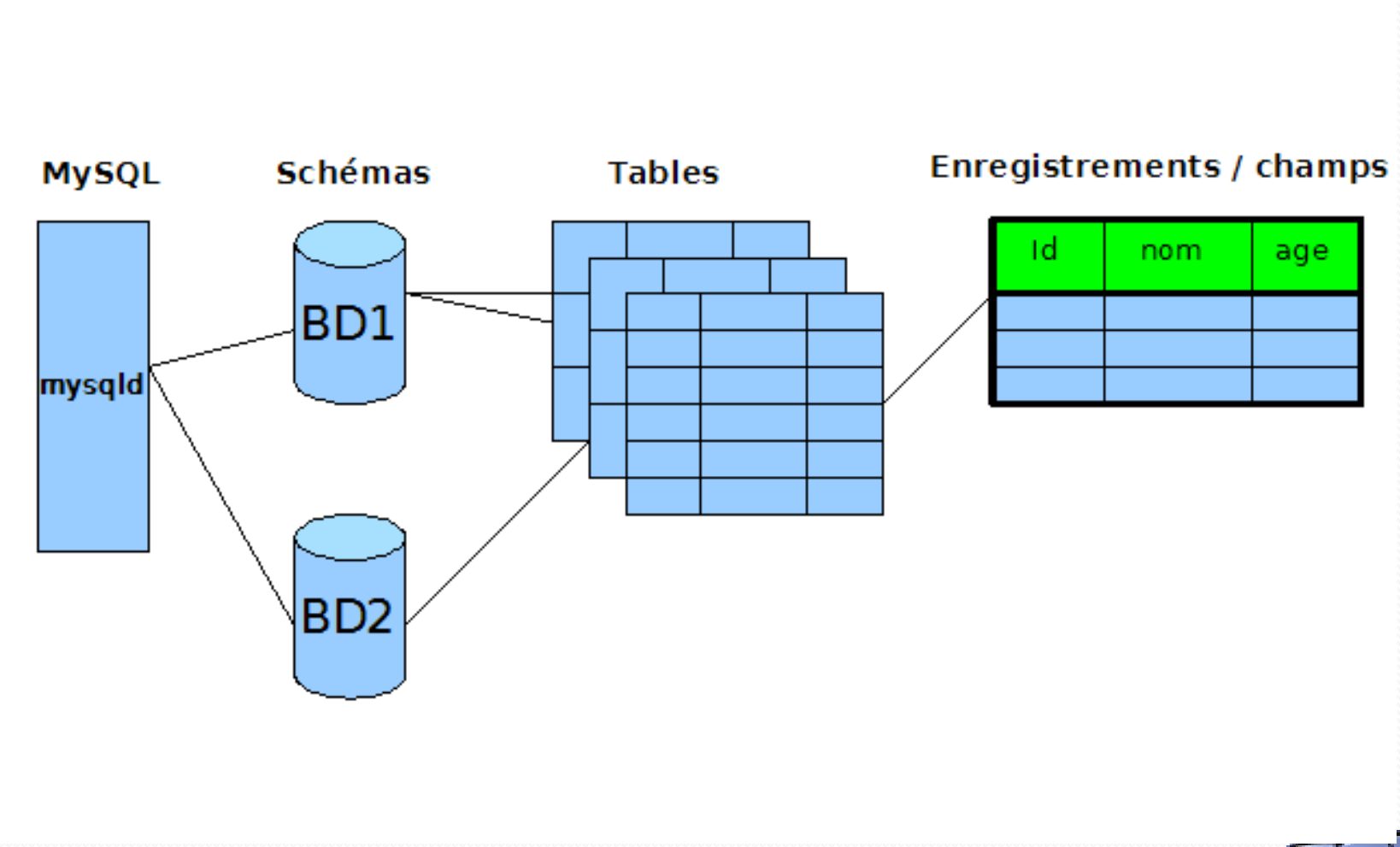
- Application:

Solution (QuelJourCetait.txt)

Fonctions spéciales:

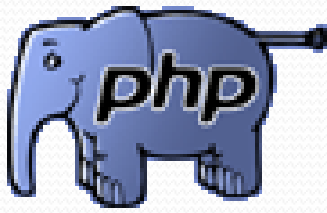
```
$jd = gregoriantojd($mois,$jour,$an);
$semaine[jddayofweek($jd,1)]
```

Rappel MySQL
Structure de la base de données



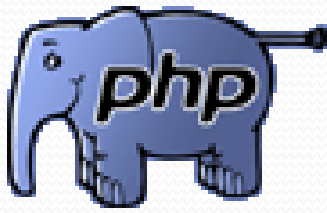
Structure de la base de données

- Un champ est une donnée, définie par un type, une longueur et des contraintes.
- Un enregistrement est une ensemble de champs.
- Une table contient des enregistrements et des
- champs.
- Un schéma (nom ANSI d'une base de données) est un ensemble de tables.
- Le serveur peut contenir plusieurs schémas.



Rappel MySQL
L'extension mysqli

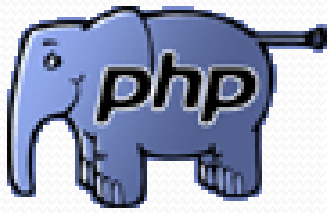
- Permet de profiter des fonctionnalités de MySQL 4.1 et +
- A partir de PHP 4.1.3
- PHP doit être compilé avec le support de l'extension mysqli (linux)
- L'extension mysqli doit être activée dans php.ini (windows)



Se connecter

- La fonction mysqli_connect(), possède 4 arguments principaux :
  - l'adresse du serveur
  - le nom d'utilisateur
  - le mot de passe pour l'authentification
  - la base de données à utiliser

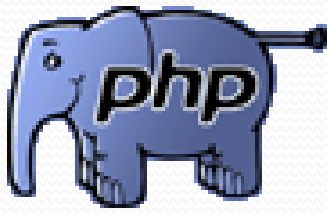
```
<?php
$link = mysqli_connect('sql.zend.fr', 'monlogin', 'secret', 'mabase');
?>
```



Créer un fichier de configuration

- L'accès aux bases de données se fait en plusieurs points de l'application
- Factorisez les informations de connexion dans un fichier de configuration

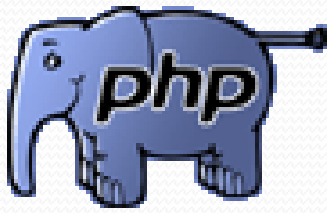
```
<?php
$mysql_host = 'sql.zend.fr';
$mysql_login = 'login';
$mysql_pass = 'secret';
$mysql_db = 'mabase';
$link = mysqli_connect($mysql_host, $mysql_login,
$mysql_pass,
$mysql_db);
?>
```



Envoyer une requête au serveur

- La fonction mysqli_query() permet d'envoyer une requête au serveur MySQL
- Elle prend 2 paramètres :
  - un identifiant de connexion vers le serveur
  - une requête SQL

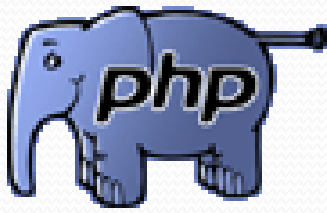
```
<?php
include_once 'configuration.php';
$sql = "SELECT nom, prenom FROM client WHERE ville
= 'Paris'";
$resultat = mysqli_query( $link, $sql );
?>
```



Récupérer le résultat

3 fonctions pour récupérer le résultat d'une requête :

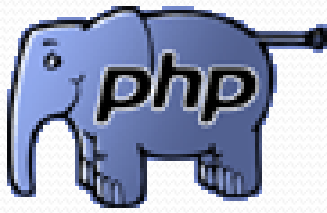
- mysqli_fetch_assoc() : Récupère le résultat sous forme de tableau associatif
- mysqli_fetch_row() : Récupère le résultat sous forme de tableau indexé
- mysqli_fetch_object() : Récupère le résultat sous forme d'objet



Récupérer le résultat

Exploiter le résultat d'une requête SELECT

```
<?php
include_once 'configuration.php';
$sql = "SELECT nom, prenom FROM client WHERE ville = 'Paris'";
$resultat = mysqli_query($link, $sql);
$enregistrement = mysqli_fetch_assoc($resultat);
while ($enregistrement) {
// Affiche le champ prenom
echo
$enregistrement['prenom'], ' ';
// Affiche le champ nom
echo
$enregistrement['nom'], '<br>';
}
```

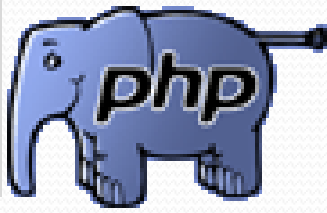


Fermer une connexion

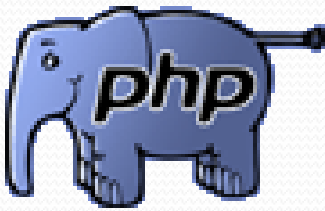
La fonction mysqli_close() permet de fermer la connexion

Elle prend en argument l'identifiant de connexion

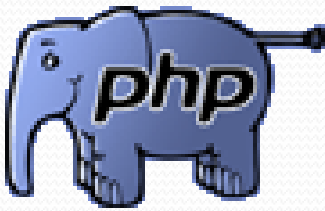
```
<?php
include_once('configuration.php');
$sql = "SELECT nom, prenom FROM client WHERE ville = 'Paris'";
$resultat = mysqli_query($link, $sql);
$enregistrement = mysqli_fetch_assoc($resultat);
while ($enregistrement) {
// Affiche le champ prenom
echo
$enregistrement['prenom'], ' ';
// Affiche le champ nom
echo
$enregistrement['nom'], '<br>';
}
//ferme la connexion
mysqli_close($link)
```



Rappel MySQL
L'extension mysqli



Rappel MySQL
L'extension mysqli



Rappel MySQL
L'extension mysqli