



République Tunisienne  
Ministère de l'Enseignement Supérieur,  
de la Recherche Scientifique

République Tunisienne  
Ministère de l'Enseignement Supérieur, de la Recherche  
Scientifique  
Des Technologies De L'Information  
Et De La Communication  
Université de Tunis  
École nationale supérieure d'ingénieurs de Tunis

# Java

## Création de pages Web Dynamiques Côté serveur (en PHP)

École nationale supérieure d'ingénieurs de  
Tunis

2018-2019

**Support de cours :**

<https://classroom.google.com>: k0furj

• Dr. Mehrez Boulares

# Introduction

## ■ Les langages de script-serveur : Définition

- Un langage de script -serveur est :
  - un programme stocké sur un serveur et exécuté par celui-ci,
  - qui passe en revue les lignes d'un fichier source pour en modifier une partie du contenu,
  - avant de renvoyer à l'appelant ( un navigateur par exemple) le résultat du traitement.
- La tâche d'interprétation des ordres à exécuter est déléguée à un composant, souvent appelé moteur,
  - installé sur le serveur,
  - qui est doté d'une API et d'un fonctionnement identique quel que soit la plate-forme utilisée pour gérer le serveur

# Introduction

- Pages web dynamiques côté serveur ou côté client
  - Langage côté serveur : le travail d'interprétation du programme est réalisé par le serveur
    - Sont indépendants de la machine et du logiciel de navigation utilisés pour la consultation.
    - Sont compatibles avec tous les navigateurs et toutes leurs versions.
    - Permettent de masquer les sources de ses programmes
    - Nécessitent de recharger la page chaque fois que celle-ci est modifiée.
  - Pages côté serveur et côté client :
    - Script côté client pour des calculs et des traitements simples
    - Scripts côté serveur pour des calculs, des traitements et des mises à jours plus conséquents

# Introduction

- Les langages de création de page web dynamiques côté serveur
  - PHP
    - Connaît un succès toujours croissant sur le Web et se positionne comme un rival important pour ASP
    - L'environnement Linux est sa plateforme de prédilection
    - Combiné avec le serveur Web Apache et la base de données MySQL, PHP offre une solution particulièrement robuste, stable et efficace
    - Gratuité : Tous les logiciels sont issus du monde des logiciels libres (Open Source).

# PHP : C'est QUOI ?

## ■ Définition

- Un langage de scripts permettant la création d'applications Web
- Indépendant de la plate-forme utilisée puisqu'il est exécuté côté serveur et non côté client.
- La syntaxe du langage provient de celles du langage C, du Perl et de Java.
- Ses principaux atouts sont:
  - La gratuité et la disponibilité du code source
  - La simplicité d'écriture de scripts
  - La possibilité d'inclure le script PHP au sein d'une page HTML
  - La simplicité d'interfaçage avec des bases de données
  - L'intégration au sein de nombreux serveurs web (Apache, Microsoft IIS, ...)

# Intégration PHP et HTML (1)

## ■ Principe

- Les scripts PHP sont généralement intégrés dans le code d'un document HTML
- L'intégration nécessite l'utilisation de balises
  - avec le style xml : <? ligne de code PHP ?>
  - Avec le style php: <?php ligne de code PHP ?>
  - avec le style JavaScript :  
<script language=<php>> ligne de code PHP </script>
  - avec le style des ASP : <% ligne de code ASP %>

# Intégration PHP et HTML (2)

## ■ Forme d'une page PHP

- Intégration directe

```
< ?php  
//ligne de code PHP  
?>  
  
<html>  
<head> <title> Mon script PHP  
</title> </head>  
  
<body>  
//ligne de code HTML  
  
< ?php  
//ligne de code PHP  
?>  
//ligne de code HTML  
  
...  
</body> </html>
```

# Intégration PHP et HTML (3)

- Forme d'une page PHP
  - Inclure un fichier PHP dans un fichier HTML : include()

Fichier Principal

```
<html>
<head>
<title> Fichier d'appel </title>
</head>
<body>
<?php
$salut = " BONJOUR" ;
include "information.inc" ;
?>
</body>
</html>
```

Fichier à inclure : information.inc

```
<?php
$chaine=$salut. " , C'est PHP " ;
echo " <table border= \"3\" >
      <tr> <td width = \" 100%\" >
          <h2> $chaine</h2>
      </td> </tr></table> ";
?>
```

- On donne généralement aux fichiers de code PHP l'extension .inc ou .inc.php:

- **avantage** : protéger les données confidentielles (paramètres de connexion à la base de données).

- Si le fichier ne contient que vos paramètres dans des variables, le serveur ne renvoie rien au poste client si quelqu'un tente de l'exécuter, alors qu'un navigateur affiche le contenu d'un fichier avec l'extension .inc seule.

# Un formulaire HTML et sa réponse en PHP

```
<!doctype html>
<html>
<head>
<meta charset="utf-8" />
<title>
Formulaire html
</title>
</head>
<body>
<form action="reponse.php" method="GET">
Votre nom :<input type="text" name="nom">
Votre âge :<input type="text" name="age">
<p>
<input type=submit value="Envoyer">
</form>
</body>
</html>
```

# Un formulaire HTML et sa réponse en PHP

```
<!doctype html>
<html>
<head>
<meta charset="utf-8" />
<title>
Test Formulaire PHP
</title>
</head>
<body>
<h1>Bonjour, <?php echo $_GET['nom'] ?></h1>
<h2>Vous semblez avoir <?php echo $_GET['age'] ?></h2>
<?php
    $n = $_GET['nom'];
    $a = $_GET['age'];
?
Votre nom est stocké dans la variable $n
dont le type est <?php echo gettype($n) ?>

Votre âge est stocké dans la variable <b>$a</b>
<br/> dont le type est <i><?php echo gettype($a); ?></i>
<br/> On peut la transformer en <i>integer</i> en faisant :
    <?php settype($a,"integer"); ?>
<br/>
    Type de $a :<?php echo gettype($a); ?>
</body>

</html>
```

# Les variables en PHP

## Déclaration simple:

```
<?php  
$variable = "une variable en PHP";  
// Une autre variable :  
$Variable = 1000;  
?>
```

## Existence de variables, la fonction isset() :

```
<?php  
$a = "une variable en PHP";  
if(isset($a)) echo "la variable a existe";  
unset($a);  
echo "la variable a a été supprimée ...";  
?>
```

## Test de variables, la fonction empty() :

```
<?php  
$a = "une variable en PHP";  
if (!empty($a)) echo " La variable existe et elle n'est  
pas vide !";  
?>
```

# Les variables en PHP

## Les opérateurs ? et ??

- L'opérateur ? permet de remplacer avantageusement une instruction if...else en évaluant une expression et en attribuant à une variable une première valeur si la condition est vraie ou une autre valeur si elle est fausse. Sa syntaxe est la suivante :
- Elle est équivalente à :

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

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

- En complément de l'opérateur précédent, PHP 7 ajoute l'opérateur ?? qui retourne la valeur du premier opérande s'il existe et s'il n'a pas la valeur NULL ou celle du second opérande dans le cas contraire. Sa syntaxe est :

```
$nom=$_POST['nom'] ?? 'Anonyme' ;
```

si l'utilisateur saisit son nom dans le champ de formulaire nommé 'nom', la variable \$nom le contiendra, sinon elle aura la valeur 'Anonyme'.

# Les variables en PHP

## Les opérateurs ? et ??

Ce qui permet de limiter le recours à *isset* dans de nombreuses situations comme :

```
<?php
// Récupère la valeur de $_GET['email'] et retourne 'nobody'
// si elle n'existe pas.
$mail = $_GET['email'] ?? 'nobody@null';
// Équivalent à :
$mail = isset($_GET['email']) ? $_GET['email'] :
    'nobody@null';

// Coalescing ?? peut être chainé :
// On renvoie la première valeur définie parmi
// $_GET['email'], $_POST['email'], et 'nobody@null.com'.
$mail = $_GET['email'] ?? $_POST['email'] ?? 'nobody@null';
echo "$mail\n";
```

# Les variables en PHP

## Portée des variables:

- Par défaut, toutes les variables sont **locales**
- Leur portée se réduit à la fonction ou au bloc de leur déclaration
- Pour déclarer une variable globale, on peut utiliser le tableau `$_GLOBALS[ ]`

```
<?php $_GLOBALS ['MaVar']="Bonjour"; ?>
```

## Constantes :

```
<?php
define ("USER", "TOTO");
echo USER; // Notez l'absence de $ ici
?>
```

# Les variables en PHP

## ■ Variables locales et variables globales

- variables en PHP : global, static, local
- toute variable déclarée en dehors d'une fonction est globale
- utiliser une variable globale dans une fonction, l'instruction **global** suivie du nom de la variable
- Pour conserver la valeur acquise par une variable entre deux appels de la même fonction : l'instruction static.

➤ Les variables statiques restent locales à la fonction et ne sont pas réutilisables à l'extérieur.

```
function cumul ($prix)
{
    static $cumul = 0;
    static $i = 1;
    echo "Total des achats $i = ";
    $cumul += $prix;
    $i++;
    return $cumul;
}
echo cumul (175), "<br />";
echo cumul (65), "<br />";
echo cumul (69), "<br />";
```

Total des achats 1 = 175  
Total des achats 2 = 240  
Total des achats 3 = 309

# Les chaînes en PHP:

## Les bases :

### Guillemets ou Cotes :

```
<?php  
$var="Hello PHP";  
$machaine="le contenu de \$var est $var<br>";  
echo $machaine;  
//ou avec des ' ' :  
$mystring='le contenu de $var est '.\$var;  
echo $mystring;  
?>
```

dont le résultat sera toujours :

```
le contenu de $var est Hello PHP
```

### La concaténation :

A l'aide de .

### La longueur d'une chaine :

```
<?php int lg=strlen($chaine); ?>
```

# Les chaînes en PHP:

## Les chaînes en PHP:

### Les bases :

#### Accéder au caractère i d'une chaîne :

```
<?php echo $chaine[i]; ?>
```

La chaîne est traitée comme un tableau indiqué par un \*entier\*

La plupart des tableaux de PHP sont indiqués par des chaînes...

#### Mettre en majuscules/minuscules :

- avec *strtoupper()* pour obtenir des majuscules
- avec *strtolower()* pour mettre en minuscules
- avec *ucfirst()* pour mettre en majuscule la première lettre d'une chaîne
- avec *ucwords()* pour mettre en majuscule la première lettre de chaque mot dans une chaîne

#### Recherche de sous-chaînes ou de motifs dans une chaîne :

- avec *strstr()*
- avec *stristr()*
- avec *ereg()* ou *ereg()*

# Les chaînes en PHP:

## Les chaînes en PHP:

### Les bases :

**Indication :** Les variantes de ces fonctions comportant un i indiquent une insensibilité à la casse c'est à dire que les majuscules et minuscules sont considérées comme identiques.

```
<?php
$AGENT=$_SERVER['HTTP_USER_AGENT'];
echo $AGENT;
echo("\n<P>");
if (stristr($AGENT, "MSIE"))
    echo "Vous semblez utiliser Internet Explorer !</b>";
elseif (ereg("Firefox", $AGENT))
    echo "Vous semblez utiliser Firefox !</b>";
elseif (eregi("chrome", $AGENT))
    echo "Vous semblez utiliser Chrome !</b>";
?>
```

```
//int strlen (string $ch)
string strtolower(string $ch)
//retourne la chaîne avec tous les caractères en minuscules.
string strtoupper(string $ch)
//retourne la chaîne avec tous les caractères en majuscules.
string ucwords(string $ch)
//retourne la chaîne avec toutes les initiales des mots qui la composent en majuscules.
string ucfirst(string $ch)
//retourne la chaîne avec uniquement la première lettre en majuscule.
```

# Les chaînes en PHP:

## Les chaînes en PHP:

### Les bases :

```
$nom = "ENgels" ;
$prenom = "jEan" ;
$adresse = "21, rue compoint" ;
$ville = "75018 pAris" ;
$mail = "ENGELS@funPHP.Com" ;
$prenom = ucfirst(strtolower($prenom)) ;
$nom = strtoupper($nom) ;
$adresse = ucwords(strtolower($adresse)) ;
$ville = strtoupper($ville) ;
$mail = strtolower($mail);
echo "Mes coordonnées <br />";
echo $prenom, $nom, "<br />";
echo $adresse, "<br />" ;
echo $ville, "<br />" ;
```

# Les chaînes en PHP:

## Les chaînes en PHP:

### Les bases :

```
//Gestion des espaces
//Vous disposez pour cela des trois fonctions, ltrim, rtrim et trim.
string ltrim (string $ch [,string liste])
//renvoie la chaîne $ch nettoyée des espaces situés en début de chaîne.
string rtrim (string $ch [,string liste])
//supprime les espaces situés en fin de chaîne.
string trim (string $ch [,string liste])
//supprime les espaces situés en début et en fin de chaîne.
//Le paramètre liste permet de définir une liste de caractères à supprimer, qu'ils soient
//des caractères d'espacement ou des caractères quelconques.
$a = "...Jean " ;
$b = "Dupont__";
echo $a,$b,"<br />";
echo trim($a,' .')," ",rtrim($b,' _');
```

...Jean Dupont\_\_  
Jean Dupont

# Les chaînes en PHP:

## Les chaînes en PHP:

### Entités HTML et caractères spéciaux:

- Pour ajouter automatiquement le caractère d'échappement \ devant les caractères spéciaux tels que apostrophe ('), guillemets ("), antislash (\) et le caractère NUL, vous devez utiliser la fonction addslashes(), dont la syntaxe est la suivante : **string addslashes (string \$ch)**
- Cette fonction peut être utilisée avant d'enregistrer des chaînes dans une base de données. Réciproquement, la fonction stripslashes() enlève les caractères d'échappement.

```
$ch="Le répertoire est : 'C:\PHP_doc\php5'";
$ch = addslashes($ch);
echo $ch;
$ch =stripslashes($ch);
echo $ch;
```

---

Le répertoire est : \'C:\\PHP\_doc\\php5\'  
Le répertoire est : 'C:\PHP\_doc\php5'

---

# Les chaînes en PHP:

## Les chaînes en PHP:

### Entités HTML et caractères spéciaux:

- Pour créer du code HTML ou XML, vous devez transformer certains caractères spéciaux (&, ", ', <, >) en entités de caractères. Vous utilisez pour cela la fonction **htmlspecialchars()**, dont la syntaxe est la suivante :
- **string htmlspecialchars (string \$ch [, int CTE[,string charset]])**
- Le paramètre CTE est une constante qui détermine le traitement des guillemets. Elle prend les valeurs suivantes :
  - **ENT\_COMPAT** ou 2 (valeur par défaut) convertit les guillemets doubles mais pas les guillemets simples.
  - **ENT\_QUOTES** ou 3 convertit les guillemets doubles et simples.
  - **ENT\_NOQUOTES** ou 0 ne traite aucun des guillemets.
  - Le paramètre **charset** désigne le jeu de caractères utilisé. Par défaut, il s'agit de **UTF-8**.

# Les chaînes en PHP:

## Les chaînes en PHP:

### Entités HTML et caractères spéciaux:

- La fonction `htmlentities()`, dont la syntaxe est la suivante :
- **string htmlentities (string \$ch [, int CTE[,string charset]])**
- retourne une chaîne dans laquelle tous les caractères spéciaux en HTML, donc tous ceux dont le code **UNICODE** est supérieur à 128, en entités de caractère interprétables par les navigateurs.
- Les paramètres **CTE** et **charset** ont les mêmes significations que dans la fonction **htmlspecialchars()**.
- Appliquée aux saisies des visiteurs, cette fonction empêche la création intempestive de code HTML en cas de saisie de balises dans une zone de texte.

# Les chaînes en PHP:

Les chaînes en PHP:

Entités HTML et caractères spéciaux:

```
$ch = "Cliquez sur l'icône en tête pour démarrer" ;
$ch2 = "L'élément HTML du bouton est <button>" ;
echo htmlentities($ch);
echo "<br />";
echo htmlentities($ch2);
```

Cliquez sur l'icône en tête pour démarrer  
L'élément HTML du bouton est <button>

# Le typage en PHP:

## Le typage en PHP: Les fonctions `gettype()` et `settype()` :

`gettype()` renvoie l'un des résultats suivants :

- integer
- double
- string
- array
- object
- class
- « unknown type »

`settype()` change le type d'un élément :

```
<?php  
$a=3.5;  
settype ($a,"integer");  
  
echo "le contenu de la variable a est ".$a;  
?>
```

dont le résultat sera :

```
le contenu de la variable a est 3
```

# Le typage en PHP:

## Le typage en PHP: Fonctions de test :

- *is\_int()*
- *is\_long()*
- *is\_double()*
- *is\_array()*
- *is\_object()*
- *is\_string()*

**Attention :** N'oubliez pas comme en JavaScript la différence entre l'opérateur `==` et `===`

Le premier vérifie l'égalité des contenus en ne tenant pas compte d'une éventuelle différence de typage (int ou string par exemple) tandis que le second vérifie une égalité stricte.

En d'autres termes : `5 == « 5 »` est VRAI tandis que `5 === « 5 »` est FAUX

# Quelques particularités de PHP:

## Valeurs des variables :

```
<?php  
$toto = "Bonjour<br/>\n";  
$var = "toto";  
echo $$var;  
?>
```

dont le résultat sera toujours :

## La fonction eval() : eval — Exécute une chaîne comme un script PHP

```
<?php  
$string = 'tasse';  
$name = 'café';  
$str = 'Ceci est une $string avec mon $name dedans.<br />';  
echo $str;  
eval( "$str = \"$str\";" );      Ceci est une $string avec mon $name dedans.  
echo $str;                      Ceci est une tasse avec mon café dedans.  
?>
```

**Attention** La construction de langage eval() est très dangereuse car elle autorise l'exécution de code PHP arbitraire. Son utilisation est vivement déconseillée. Si vous avez soigneusement vérifié qu'il n'y a pas d'autres options que de l'utiliser, gardez une attention toute particulière à ne pas y passer de données provenant d'un utilisateur sans les avoir précédemment validées minutieusement.

# Syntaxe de base : Les tableaux(1)

## ■ Principe

- Création à l'aide de la fonction `array()`
- Uniquement des tableaux à une dimension
  - Les éléments d'un tableau peuvent pointer vers d'autres tableaux
- Les éléments d'un tableau peuvent appartenir à des types distincts
- L'index d'un tableau en PHP commence de 0
- Pas de limites supérieures pour les tableaux
- La fonction `count()` pour avoir le nombre d'éléments d'un tableau

# Syntaxe de base : Les tableaux(2)

## ■ Les tableaux indicés et les tableaux associatifs

- Tableau indicé

- Accéder aux éléments par l'intermédiaire de numéros

```
$tableau[indice] = valeur;
$jour[3] = "Mercredi";
$note[0] = 20;


```

# Syntaxe de base : Les tableaux(4)

## ■ Tableaux multidimensionnels

- Pas d'outils pour créer directement des tableaux multidimensionnels
- L'imbrication des tableaux est possible

```
$tab1 = array(Val0, Vall, ..., ValN);
$tab2 = array(Val0, Vall, ..., ValN);
// Création d'un tableau à deux dimensions
$tableau = array($tab1, $tab2);
$mois = array("Janvier", "Février", "Mars", "Avril", "Mai", "Juin",
    "Juillet", "Août", "Septembre", "Octobre", "Novembre",
    "Décembre");
$jour = array("Dimanche", "Lundi", "Mardi", "Mercredi", "Jeudi",
    "Vendredi", "Samedi");
&element_date = array(&mois, &jour);

$variable = $tableau[indice][indice];
$MM = $element_date[0][0]; //affecte "Janvier" à $MM
echo $element_date[1][5] . " 7 " . $element_date[0][2] . "2002"; //
retourne "Jeudi 7 Mars 2002"
```

# Syntaxe de base : Les tableaux(5)

## ■ Lecture des éléments d'un tableau

- Avec une boucle for

```
for ($i=0; $i<count($tab) ; $i++){  
    if ($tab[$i]== "a") {echo $tab[$i], "<br />" ; } }
```

- Avec une boucle while

```
$i=0;  
while ($tab[$i]){  
    if ($tab[$i][0] =="a" ) {echo $tab[$i], "<br />" ; } }
```

- Avec La boucle foreach

```
$jour = array( "Dimanche" , "Lundi" , "Mardi" , "Mercredi" , "Jeudi" ,  
    "Vendredi" , "Samedi" );  
$i = 0;  
foreach($jour as $JJ) { echo "La cellule n° ". $i . " : " . $JJ .  
    "<br>" ; $i++ ; }
```

# Syntaxe de base : Les tableaux(6)

## ■ Lecture des éléments d'un tableau

- Parcours d'un tableau associatif
  - Réalisable en ajoutant avant la variable \$valeur, la clé associée

```
$tableau = array(clé1 => val1, clé2 => val2, ..., cléN => valN);  
foreach($tableau as $clé => $valeur) {  
    echo "Valeur ($clé): $valeur"; }
```

```
$jour = array("Dimanche" => 7, "Lundi" => 1, "Mardi" => 2,  
             "Mercredi" => 3, "Jeudi" => 4, "Vendredi" => 5, "Samedi" => 6);  
foreach($jour as $sJJ => $nJJ) {  
    echo "Le jour de la semaine n° ". $nJJ . " : " . $sJJ . "<br>"; }
```

# Syntaxe de base : Les tableaux(7)

## ■ Fonctions de tri

### • Tri selon les valeurs

- La fonction **sort()** effectue un tri sur les valeurs des éléments d'un tableau selon un critère alphanumérique :selon les codes ASCII :
  - « a » est après « Z » et « 10 » est avant « 9 »)
  - Le tableau initial est modifié et non récupérables dans son ordre original
  - Pour les tableaux associatifs les clés seront perdues et remplacées par un indice créé après le tri et commençant à 0
- La fonction **rsort()** effectue la même action mais en ordre inverse des codes ASCII.
- La fonction **asort()** trie également les valeurs selon le critère des codes ASCII, mais en préservant les clés pour les tableaux associatifs
- La fonction **arsort()** la même action mais en ordre inverse des codes ASCII
- la fonction **natcasesort()** effectue un tri dans l'ordre alphabétique non ASCII (« a » est avant « z » et « 10 » est après « 9 »)

# Les tableaux en PHP:

Tableaux associatifs - parcours avec boucle foreach :

```
<?php
$jours=array ("Lu"=>"Lundi", "Ma"=>"Mardi",
              "Me"=>"Mercredi", "Je"=>"Jeudi", "Ve"=>"Vendredi",
              "Sa"=>"Samedi", "Di"=>"Dimanche" );
foreach($jours as $key=>$val) echo $key." ".$val."<br>\n";
?>
```

Ce qui donne :

```
Lu Lundi
Ma Mardi
Me Mercredi
Je Jeudi
Ve Vendredi
Sa Samedi
Di Dimanche
```

# Les tableaux en PHP:

## Tableaux associatifs - Affichage avec print\_r()

```
<?php  
    print_r($jours);  
?>
```

### Résultat brut html :

```
Array  
(  
    [Lu] => Lundi  
    [Ma] => Mardi  
    [Me] => Mercredi  
    [Je] => Jeudi  
    [Ve] => Vendredi
```

```
    [Sa] => Samedi  
    [Di] => Dimanche
```

```
)
```

# Les tableaux en PHP:

## Utilisation de la fonction array\_walk

```
<?php array_walk($jours, 'aff_tab'); ?>
```

En ayant défini au préalable :

```
<?php
function aff_tab($val, $key) {
echo "$key-$val<br/>\n";
}
?>
```

On obtient le même résultat qu'avec la boucle foreach

## Tri simple d'un tableau

```
<?php
sort($jours);
array_walk($jours, 'aff_tab');
?>
```

0-Dimanche  
1-Jeudi  
2-Lundi  
3-Mardi  
4-Mercredi  
5-Samedi  
6-Vendredi

# Les tableaux en PHP:

C'est à dire que :

- Le tableau est trié selon l'ordre de ses valeurs — les clefs sont effacées et réaffectées avec des entiers.

Si on veut préserver également les clefs du tableau associatif, il faut utiliser la méthode suivante :

Tri selon l'ordre naturel avec natsort

```
<?php
$jours=array ("Lu"=>"Lundi", "Ma"=>"Mardi",
"Me"=>"Mercredi", "Je"=>"Jeudi", "Ve"=>"Vendredi",
"Sa"=>"Samedi", "Di"=>"Dimanche" );
var_dump($jours);
natsort($jours);
var_dump($jours);
?>
```

```
array(7) {
["Lu"]=>
string(5) "Lundi"
["Ma"]=>
string(5) "Mardi"
["Me"]=>
string(8) "Mercredi"
["Je"]=>
string(5) "Jeudi"
["Ve"]=>
string(8) "Vendredi"
["Sa"]=>
string(6) "Samedi"
["Di"]=>
string(8) "Dimanche"
}
array(7) {
["Di"]=>
string(8) "Dimanche"
["Je"]=>
string(5) "Jeudi"
["Lu"]=>
string(5) "Lundi"
["Ma"]=>
string(5) "Mardi"
["Me"]=>
string(8) "Mercredi"
["Sa"]=>
string(6) "Samedi"
```

Résultat brut html

# Les tableaux prédéfinis de PHP

Les tableaux concernant le protocole HTTP:

- `$_GET[ ]`, `$_POST[ ]` ou `$_REQUEST[ ]` qui englobe les 2
- `$_SERVER[ ]`: Variables décrivant le client ou la page courante
- `$_GLOBALS[ ]` variables globales
- `$_COOKIE[ ]` pour les cookies
- `$_SESSION[ ]` pour les sessions

# Les tableaux prédéfinis de PHP

Exemple récupération de `$_SERVER[ ]` grâce à la fonction `getenv()` :

```
<?php
function infos() {
    $env = array('remote_addr', 'http_accept_language', 'http_
    host',
    'http_user_agent', 'script_filename', 'server_addr',
    'server_name', 'server_signature', 'server_software',
    'request_method', 'query_string', 'request_uri', 'script_name
    ');
    // Construction d'un tableau associatif
    // Avec les valeurs lues dans l'environnement
    $retour =array();
    foreach ($env as $clef) $retour[$clef] = getenv($clef);
    return $retour;
}

echo("Voici les infos disponibles:<BR>");
$tab = infos();

foreach ($tab as $clef=>$val) echo $clef." :".$val."<br>\n";
?>
```

# Les tableaux prédéfinis de PHP

## Résultat

```
Voici les infos disponibles:  
remote_addr ::1  
http_accept_language :fr-fr  
http_host :localhost  
http_user_agent :Mozilla/5.0 (Macintosh; U; Intel Mac OS X  
→10_6_4; fr-fr)  
AppleWebKit/533.18.1 (KHTML, like Gecko) Version/5.0.2  
→Safari/533.18.5  
script_filename :/Users/roza/Sites/php/exemples/infospy.php  
server_addr ::1  
server_name :localhost  
server_signature :  
server_software :Apache/2.2.14 (Unix) mod_ssl/2.2.14  
OpenSSL/0.9.81 DAV/2 PHP/5.3.2  
request_method :GET  
query_string :  
request_uri :/~roza/php/exemples/infospy.php  
script_name :/~roza/php/exemples/infospy.php  
`User-Agent <http://localhost/~roza/php/exemples/infospy.  
→php>`_
```

# L'inclusion de fichiers externes

## *include :*

- Semblable aux *include* du C/C++
- Réalise une inclusion physique du fichier demandé

## *include\_once :*

- identique au include
- protège contre d'éventuelles inclusions multiples
- qui pourraient mener à des erreurs (redéclarations, etc.)

```
<?php include_once ("connect.php"); ?>
```

# L'inclusion de fichiers externes

## *require et require\_once :*

- fonctionnent comme le include et le include\_once respectivement
- mais le programme s'arrête si le fichier inclus n'existe pas

```
<?php  
    require ("malib.php");  
    require_once ("connect.php");  
?>
```

## *dirname()*

Pour savoir dans quel répertoire on se trouve on peut utiliser la fonction PHP dirname()

```
<?php  
include_once(dirname(__FILE__) . '/config/config.inc.php');  
?>
```

---

**Indication :** Lorsqu'on inclus ou désigne des fichiers, il vaut mieux utiliser des chemins relatifs pour repérer les fichiers (comme ci dessus) plutôt que de donner un chemin absolu par rapport à la racine du serveur du style /home/user/www/config/config.inc.php Cela sera beaucoup plus portable d'un serveur à l'autre et vous évitera bien des déboires !

---

# Syntaxe de base : les opérateurs (1)

## ■ Les opérateurs

- les opérateurs de calcul
- les opérateurs d'assignation
- les opérateurs d'incrémentation
- les opérateurs de comparaison
- les opérateurs logiques
- les opérateurs bit-à-bit
- les opérateurs de rotation de bit

# Syntaxe de base : Les opérateurs(2)

## ■ Les opérateurs de calcul

Opérateur	Dénomination	Effet	Exemple	Résultat
+	opérateur d'addition	Ajoute deux valeurs	$$x+3$	10
-	opérateur de soustraction	Soustrait deux valeurs	$$x-3$	4
*	opérateur de multiplication	Multiplie deux valeurs	$$x*3$	21
/	plus: opérateur de division	Divise deux valeurs	$$x/3$	2.3333333
=	opérateur d'affectation	Affecte une valeur à une variable	$$x=3$	Met la valeur 3 dans la variable \$x

# Syntaxe de base : Les opérateurs(3)

## ■ Les opérateurs d'assignation

Opérateur	Effet
<code>+=</code>	addition deux valeurs et stocke le résultat dans la variable (à gauche)
<code>-=</code>	soustrait deux valeurs et stocke le résultat dans la variable
<code>*=</code>	multiplie deux valeurs et stocke le résultat dans la variable
<code>/=</code>	divise deux valeurs et stocke le résultat dans la variable
<code>%=</code>	donne le reste de la division deux valeurs et stocke le résultat dans la variable
<code>  =</code>	Effectue un OU logique entre deux valeurs et stocke le résultat dans la variable
<code>^=</code>	Effectue un OU exclusif entre deux valeurs et stocke le résultat dans la variable
<code>&amp;=</code>	Effectue un Et logique entre deux valeurs et stocke le résultat dans la variable
<code>.=</code>	Concatène deux chaînes et stocke le résultat dans la variable

# Syntaxe de base : Les opérateurs(4)

## ■ Les opérateurs d'incrémentation

Opérateur	Dénomination	Effet	Syntaxe	Résultat (avec x valant 7)
<code>++</code>	Incrémantation	Augmente d'une unité la variable	<code>\$x++</code>	8
<code>--</code>	Décrémantation	Diminue d'une unité la variable	<code>\$x--</code>	6

## ■ Les opérateurs de comparaison

Opérateur	Dénomination	Effet	Exemple	Résultat
<code>==</code>	opérateur d'égalité	Compare deux valeurs et vérifie leur égalité	<code>\$x==3</code>	Retourne 1 si \$X est égal à 3, sinon 0
<code>&lt;</code>	opérateur d'infériorité stricte	Vérifie qu'une variable est strictement inférieure à une valeur	<code>\$x&lt;3</code>	Retourne 1 si \$X est inférieur à 3, sinon 0
<code>&lt;=</code>	opérateur d'infériorité	Vérifie qu'une variable est inférieure ou égale à une valeur	<code>\$x&lt;=3</code>	Retourne 1 si \$X est inférieur à 3, sinon 0
<code>&gt;</code>	opérateur de supériorité stricte	Vérifie qu'une variable est strictement supérieure à une valeur	<code>\$x&gt;3</code>	Retourne 1 si \$X est supérieur à 3, sinon 0
<code>&gt;=</code>	opérateur de supériorité	Vérifie qu'une variable est supérieure ou égale à une valeur	<code>\$x&gt;=3</code>	Retourne 1 si \$X est supérieur ou égal à 3, sinon 0
<code>!=</code>	opérateur de différence	Vérifie qu'une variable est différente d'une valeur	<code>\$x!=3</code>	Retourne 1 si \$X est différent de 3, sinon 0

# Syntaxe de base : Les opérateurs(5)

## ■ Les opérateurs logiques

Opérateur	Dénomination	Effet	Syntaxe
ou OR	OU logique	Vérifie qu'une des conditions est réalisée	((condition1)    (condition2))
&& ou AND	ET logique	Vérifie que toutes les conditions sont réalisées	((condition1)&&(condition2))
XOR	OU exclusif	Opposé du OU logique	((condition1)XOR(condition2))
!	NON logique	Inverse l'état d'une variable booléenne (retourne la valeur 1 si la variable vaut 0, 0 si elle vaut 1)	(!condition)

## ■ Les opérateurs bit-à-bit

Opérateur	Dénomination	Effet	Syntaxe	Résultat
&	ET bit-à-bit	Retourne 1 si les deux bits de même poids sont à 1	9 & 12 (1001 & 1100)	8 (1000)
	OU bit-à-bit	Retourne 1 si l'un ou l'autre des deux bits de même poids est à 1 (ou les deux)	9   12 (1001   1100)	13 (1101)
^	OU bit-à-bit	Retourne 1 si l'un des deux bits de même poids est à 1 (mais pas les deux)	9 ^ 12 (1001 ^ 1100)	5 (0101)
~	Complément (NON)	Retourne 1 si le bit est à 0 (et inversement)	~9 (~1001)	6 (0110)

# Syntaxe de base : Les opérateurs(6)

## ■ Les opérateurs de rotation de bit

Opérateur	Dénomination	Effet	Syntaxe	Résultat
<<	Rotation à gauche	Décale les bits vers la gauche (multiplie par 2 à chaque décalage). Les zéros qui sortent à gauche sont perdus, tandis que des zéros sont insérés à droite	<code>6 &lt;&lt; 1</code> (11001100 << 1)	12 (1100)
>>	Rotation à droite avec conservation du signe	Décale les bits vers la droite (divise par 2 à chaque décalage). Les zéros qui sortent à droite sont perdus, tandis que le bit non-nul de poids fort est recopié à gauche	<code>6 &gt;&gt; 1</code> (01100110 >> 1)	3 (0011)

## ■ Autres opérateurs

Opérateur	Dénomination	Effet	Syntaxe	Résultat
.	Concaténation	Joint deux chaînes bout à bout	<code>"Bonjour"."Au revoir"</code>	"BonjourAu revoir"
\$	Référencement de variable	Permet de définir une variable	<code>\$MaVariable = 2;</code>	
->	Propriété d'un objet	Permet d'accéder aux données membres d'une classe	<code>\$MonObjet-&gt;Propriete</code>	

# Syntaxe de base : Les opérateurs(7)

## ■ Les priorités

# Syntaxe de base : Les instructions conditionnelles(1)

- L'instruction if

- if (condition réalisée) { liste d'instructions }

- L'instruction if ... Else

- if (condition réalisée) { liste d'instructions }
  - else { autre série d'instructions }

- L'instruction if ... elseif ... Else

- if (condition réalisée) { liste d'instructions }
  - elseif (autre condition ) {autre série d'instructions }
  - else (dernière condition réalisée) { série d'instructions }

- Opérateur ternaire

- (condition) ? instruction si vrai : instruction si faux

# Syntaxe de base : Les instructions conditionnelles(2)

## ■ L'instruction switch

```
switch (Variable) {  
    case Valeur1: Liste d'instructions break;  
    case Valeur1: Liste d'instructions break;  
    case Valeurs...: Liste d'instructions break;  
    default: Liste d'instructions break;  
}
```

# Syntaxe de base : Les instructions conditionnelles(3)

- La boucle for

- `for ($i=1; $i<6; $i++) { echo "$i<br>"; }`

- La boucle while

- `While(condition) {bloc d'instructions ;}`
  - `While (condition) :Instruction1 ;Instruction2 ;`  
`.... endwhile ;`

- La boucle do...while

- `Do {bloc d'instructions ;}while(condition) ;`

- La boucle foreach (PHP4)

- `Foreach ($tableau as $valeur) {insts utilisant $valeur ;}`

# Syntaxe de base : Les fonctions(1)

## ■ Déclaration et appel d'une fonction

```
Function nom_fonction($arg1, $arg2, ...$argn)
{
    déclaration des variables ;
    bloc d'instructions ;
    // fin du corps de la fonction
    return $resultat ;
}
```

## ■ Fonction avec nombre d'arguments inconnu

- **func\_num\_args()** : fournit le nombre d'arguments qui ont été passés lors de l'appel de la fonction
- **func\_get\_arg(\$i)** : retourne la valeur de la variable située à la position \$i dans la liste des arguments passés en paramètres.
  - Ces arguments sont numérotés à partir de 0

# Syntaxe de base : Les fonctions(2)

## ■ Fonction avec nombre d'arguments inconnu

```
<?php
function produit()
{
    $nbarg = func_num_args() ;
    $prod=1 ;
    // la fonction produit a ici $nbarg arguments
    for ($i=0 ; $i <$nbarg ; $i++)
    {
        $prod *= func_get_arg($i)
    }
    return $prod;
}
echo "le produit est : ", produit (3, 77, 10, 5, 81, 9), "<br
/>" ;
// affiche le produit est 8 419 950
?>
```

# Syntaxe de base : Les fonctions(4)

## ■ Appel dynamique de fonctions

- Exécuter une fonction dont le nom n'est pas forcément connu à l'avance par le programmeur du script
- L'appel dynamique d'une fonction s'effectue en suivant le nom d'une variable contenant le nom de la fonction par des parenthèses

```
$datejour = getdate(); // date actuelle
//récupération des heures et minutes actuelles
$heure = $datejour[hours] ;
$minute=$datejour[minutes] ;
function bonjour()
{
    global $heure;
    global $minute;
    echo "<b> BONJOUR A VOUS IL EST : ", $heure, " H ", $minute, "</b> <br  />" ;
}
function bonsoir ()
{
    global $heure ;
    global $minute ;
    echo "<b> BONSOIR A VOUS IL EST : ", $heure, " H ", $minute , "</ b> <br  />" ;
}
if ($heure <= 17)
{
    $salut = "bonjour" ;
}
else $salut="bonsoir" ;
//appel dynamique de la fonction
$salut() ;
```

- Gestion des exceptions

- Une exception est un mécanisme qui permet d'intercepter une erreur générée par un script et de déclencher une action en réponse à cette erreur. Si PHP 4 ne permettait pas d'effectuer une gestion des exceptions, la version 5 fournit un mécanisme qui permet de gérer les conséquences d'une erreur.

- La classe Exception

- Le constructeur de l'objet Exception créé dans l'instruction throw reçoit deux paramètres, correspondant aux propriétés message et code de l'objet.
- Le premier est une chaîne contenant le message d'erreur et le second un entier qui définit un code d'erreur facultatif.

```
try
{
    // Code à surveiller
    if(erreur prévue){throw new Exception();}
    else{// Résultat;}
}
catch(Exception $except)
{
    // Gestion de l'erreur
}
finally
{
    //Code qui sera forcément exécuté
}
```

## ▪ La classe Exception

- Cet objet est utilisé dans le bloc catch en appelant ses méthodes pour afficher des informations sur 'exception'.

```
$a=100;
$b=0;
try
{
    if($b==0)
    {
        throw new Exception("Division par 0",7);
    }
else
{
    echo "Résultat de : $a / $b = ",$a/$b;
}
} catch(Exception $except)
{
    echo "Message d'erreur :",$except->getMessage(),"<hr>";
    echo "Nom du fichier :",$except->getFile(),"<hr>";
    echo "Numéro de ligne :",$except->getLine(),"<hr>";
    echo "Code d'erreur :",$except->getCode(),"<hr>";
    echo "__toString :",$except->__toString(),"<hr>";
}
finally
{
    echo "L'exception a &#233;t&#233; g&#233;r&#233;e, le script continue<hr/>";
}
echo "Vraie Fin";
```

- La classe Exception
- Cet objet est utilisé dans le bloc catch en appelant ses méthodes pour afficher des informations sur 'exception'.

Message d'erreur :Division par 0

---

Nom du fichier :C:\wamp\www\ExemplesCours\chaine.php

---

Numéro de ligne :126

---

Code d'erreur :7

---

\_\_toString :exception 'Exception' with message 'Division par 0' in C:\wamp\www\ExemplesCours\chaine.php:126 Stack trace: #0 {main}

---

L'exception a été gérée, le script continue

---

Vraie Fin

# Les cookies (1)

## ■ Principe

- Un cookie est un fichier texte créé par un script et stocké sur l'ordinateur des visiteurs d'un site
- Les cookies permettent de conserver des renseignements utiles sur chaque utilisateur, et de les réutiliser lors de sa prochaine visite
  - Exemple : personnaliser la page d'accueil ou les autres pages du site
    - un message personnel comportant par exemple son nom, la date de sa dernière visite, ou tout autre particularité.
- Les cookies étant stockés sur le poste client, l'identification est immédiate et ne concernent que les renseignements qui le concernent
- Pour des raisons de sécurité, les cookies ne peuvent être lus que par des pages issues du serveur qui les a créés

# Les cookies(2)

## ■ Principe

- Le nombre de cookies qui peuvent être définis sur le même poste client est limité à 20 et la taille de chacun est limitée à 4ko.
- Un navigateur peut stocker un maximum de 300 cookies
- La date d'expiration des cookies est définie de manière explicite par le serveur web chargé de les mettre en place.
- Les cookies disponibles sont importés par PHP sous forme de variables identifiées sous les noms utilisés par ces cookies
- La variable globale du serveur `$_COOKIES` enregistre tous les cookies qui ont été définis

# Les cookies(3)

- Exemple d'application des cookies
  - Mémorisation des paniers dans les applications d'e-commerce
  - Identification des utilisateurs
  - Des pages web individualisées
    - Afficher des menus personnalisés
    - Afficher des pages adaptées aux utilisateurs en fonction de leurs précédents visites

# Les cookies(4)

## ■ Écrire des cookies

- L'écriture de cookies est possible grâce à la fonction **setcookie()**
  - il faut cette fonction dès le début du script avant l'envoi d'aucune autre information de la part du serveur vers le poste client.

```
Setcookie( "nom_var" , "valeur_var" , date_expiration , "chemin" ,  
"domain" , " secure" )
```

```
<?php  
setcookie ("PremierCookie" , "Salut" , time( ) +3600*24*7) ;  
...  
if (! $PremierCookie) {  
echo "le cookie n'a pas été défini";}  
else {  
echo $premierCookie , "<br>" ;  
}  
?>
```

# Les cookies(5)

## ■ Écriture de cookies

- **Nom\_var** : nom de la variable qui va stocker l'information sur le poste client et qui sera utilisée pour récupérer cette information dans la page qui lira le cookie.
  - C'est la seule indication obligatoire pour un cookie
- **Valeur\_var** : valeur stockée dans la variable. Par exemple une chaîne de caractères ou une variable chaîne, en provenance d'un formulaire
- **Date\_expiration** : la date à laquelle le cookie ne sera plus lisible et sera effacé du poste client
  - on utilise en général la date du jour, définie avec la fonction time() à laquelle on ajoute la durée de validité désirée
  - Si l'attribut n'est pas spécifié, le cookie expire à l'issue de la session

# Les cookies(6)

- Écriture de cookies
  - **Chemin** : définit la destination (incluant les sous-répertoire) à laquelle le navigateur doit envoyer le cookie.
  - **Domain set** : le nom du domaine à partir duquel peuvent être lus les cookies.
    - On peut aussi utiliser la variable d'environnement \$SERVER\_NAME à la place.
  - **Secure** : un nombre qui vaut 0 si la connexion n'est pas sécurisée, sinon, il vaut 1 pour une connexion sécurisée

# Les cookies(7)

## ■ Lecture de cookies

- Si la directive register\_globals de php.ini est TRUE, un cookie sera automatiquement passé en paramètre au script et sa valeur sera directement accessible dans la variable **\$NomDuCookie**.  
A DECONSEILLER
- Les cookies sont accessibles dans le tableau associatif **\$\_COOKIE**
  - Si celui-ci visite une des pages PHP de ce même domaine dont le chemin est inclus dans le paramètre chemin (si le chemin est / le cookie est valide pour toutes les pages de ce site).
  - Il faut d'abord vérifier l'existence des variables dont les noms et les valeurs ont été définis lors de la création du cookie.
    - Cette vérification s'effectue grâce à la fonction **isset(\$\_COOKIE["nom\_var"])** qui renvoie true si la variable **\$nom\_var** existe et false sinon.

```
<?php
if (isset($_COOKIE["nom_var"]){
    echo "<h2> Bonjour " . $_COOKIE["nom_var"] . "</h2>" ;
} else echo "pas de cookie" ;
?>
```

# Les cookies(8)

## ■ Écriture de plusieurs variables par un cookie

- Utilisation de la fonction **compact()** pour transformer les variables en un tableau
- Convertir le tableau en une chaîne de caractère à l'aide de la fonction **implode()**
- Affecter la chaîne créée à l'attribut « nom\_cookie »

```
<?
$col="#FF0000" ;
$size=24;
$font="Arial" ;
$text="Je suis le cookie" ;
$arr=compact("col" , " size" , " font" , "text" );
$val=implode(" &" , $arr);
Setcookie("la_cookie" , $val, time() +600);
```

# Les cookies(9)

- Lecture de plusieurs variables d'un cookie
  - Décomposé la chaîne stockée par la cookie et retrouver un tableau en utilisant la fonction `explode()`

```
<?php
echo " <b> voici le contenu de la chaîne cookie : </b><br>" ;
echo $le_cookie, "<br> <br>" ;
}
$arr=explode("&" , $la_cookie);
echo "<b> ces variables ont été établies à partir de la chaîne
cookie : </b> <br> <br>" ;
foreach ($arr as $k=>$elem) {
echo "$k=>$elem <br>" ;
}
?>
```

# Les cookies(10)

## ■ Supprimer un cookie

- Il suffit de renvoyer le cookie grâce à la fonction setcookie() en spécifiant simplement l'argument NomDuCookie

```
<?php  
setcookie( "Visites" );  
?>
```

- Une autre méthode consiste à envoyer un cookie dont la date d'expiration est passée:

```
<?php  
setcookie( "Visites" , "" , time() - 1 )  
?>
```

# Les cookies(11)

## ■ Quelques précisions sur les cookies

- Setcookie() doit être utilisée avant l'envoi de données HTML vers le navigateur
- Le cookie n'est pas visible avant le prochain chargement de page
- Avec PHP3, si plusieurs cookies sont envoyées de suite, les appels seront traités en ordre inverse, alors qu'avec PHP4 il seront traités dans l'ordre
- Certains navigateurs ne traitent pas bien certains cas liés aux cookies
  - Microsoft Internet Explorer 4 avec le Service Pack 1 ne traite pas correctement les cookies qui ont le paramètre chemin défini
  - Netscape Communicator 4.05 et Microsoft Internet Explorer 3.x ne traitent pas correctement les cookies qui n'ont pas les paramètres chemin et expiration définis.

# Les cookies(12)

## ■ Exemples d'utilisation de cookies

```
//Un script permettant de savoir si un visiteur est déjà venu sur le site
pendant le mois
setcookie("Visites","Oui",time()+2592000,"/",".mondomaine.fr",0);

// Envoi d'un cookie qui disparaîtra après la fermeture du navigateur
SetCookie("UserSessionCookie",$login.":".$pass);

// Envoi d'un cookie qui restera présent 24 heures
SetCookie("DejaVisite","1",time()+3600*24,"/",". mondomaine.fr ",0);

// Envoi d'un cookie qui s'effacera le 1er janvier 2003
SetCookie("An2002","1",mktime(0,0,0,1,1,2003),"/",". mondomaine.fr ",0);

//script permettant de compter le nombre de visite de la page par le
visiteur
<?php
$Visites++;
setcookie("Visites",$Visites,time()+2592000,"/",". mondomaine.fr ",0);?>
```

# Les sessions(1)

## ■ Principe

- Est un mécanisme permettant de mettre en relation les différentes requêtes du même client sur une période de temps donnée.
- Les sessions permettent de conserver des informations relatives à un utilisateur lors de son parcours sur un site web
- Des données spécifiques à un visiteur pourront être transmises de page en page afin d'adapter personnellement les réponses d'une application PHP
- Chaque visiteur en se connectant à un site reçoit un numéro d'identification dénommé identifiant de session (SID)
- La fonction **session\_start()** se charge de générer automatiquement cet identifiant unique de session et de créer un répertoire. Elle doit être placée au début de chaque page afin de démarrer ou de continuer une session.

```
<?php  
session_start();  
$Session_ID = session_id();  
// $Session_ID = 7edf48ca359ee24dbc5b3f6ed2557e90    ?>
```

# Les sessions(2)

## ■ Principe

- Un répertoire est créé sur le serveur à l'emplacement désigné par le fichier de configuration php.ini, afin de recueillir les données de la nouvelle session.

[Session]

```
session.save_path= C:\PHP\sessiondata  
; Rép session = \sess_7edf48ca359ee24dbc5b3f6ed2557e90
```

- Le fichier php.ini peut également préciser un nom de session par l'option **session.name** ou sa durée de vie par **session.gc\_maxlifetime**
- La session en cours peut être détruite par la fonction **session\_destroy()**. Cette commande supprime toutes les informations relatives à l'utilisateur.

```
session_destroy();
```

# Les sessions(3)

- Le traitement des variables de session
  - Les variables de session sont chargées dans une session par l'intermédiaire de la fonction `session_register()`

```
<?php
    session_start();
    session_register("nom_variable");
    ...
    session_register("nom_variableN");
?>
```
  - Une fois la variable enregistrée, elle est accessible à travers le tableau associatif `$_SESSION["nom_variable"]`

# Les sessions(4)

## ■ Le traitement des variables de session

Le transport des informations entre les documents est réalisé par l'entremise

- soit d'un cookie
- soit d'une requête HTTP

- Cette dernière solution est la plus fiable puisque les cookies peuvent ne pas être acceptés par le client ou celui-ci pourrait les détruire en cours de session.
- Il suffit de concaténer l'identifiant de session à l'adresse URL de la page cible pour que cette dernière puisse accéder aux informations conservées par la session.

```
echo '<a href=' http://www.site.com/doc.php? '. session_name(). '=' . session_id().'>  
lien</a>'
```

# Les sessions(5)

## ■ Le traitement des variables de session

- Par défaut, PHP tente de passer par les cookies pour sauvegarder l'identifiant de session dans le cas où le client les accepterait. Il est possible d'éviter cela, il suffit de désactiver l'option de configuration session.use\_cookies dans le fichier php.ini.

[Session]

**session.use\_cookies 0; //désactive la gestion des sessions par cookie**

# Les sessions(6)

## *Exemple*

```
<!-- Fichier : formulaire.html -->
<html><body>
  <form method="post" action="traitement.php">
    <table border="0">
      <tr>
        <td><u>Nom :</u></td>
        <td><input type="text" name="Nom" size="20" value="RIVES"></td></tr>
      <tr>
        <td><u>Prénom :</u></td>
        <td><input type="text" name="Prenom" size="20" value="Jean-Pierre"></td></tr>
      <tr>
        <td><u>eMail :</u></td>
        <td><input type="text" name="cEmail" size="20" value="du@du.com"></td></tr>
      <tr><td> </td>
        <td><input type="submit" name="soumettre" value="Envoyer"></td></tr></table>
    </form>
  </body>
</html>
```

# Les sessions(7)

```
<?
session_start();
$nom = $_POST["Nom"];
$prenom = $_POST["Prenom"];
$email = $_POST["cEmail"];
session_register("nom");
session_register("prenom");
session_register("email");
$_SESSION["nom"]=$nom;
$_SESSION["prenom"]=$prenom;
$_SESSION["email"]=$email;
header("Location: session.php?" . session_name() . "=" . session_id());
?>
```

# Les sessions(8)

```
<?
    session_start();
?>
<html><body><?
echo("<u>Identifiant de session :</u> <b>" . session_id() . "</b><br>");
echo("<u>Nom de la session :</u> <b>" . session_name() . "</b><br><br>");
echo("<u>Nom :</u> <b>". $_SESSION["nom"] . "</b><br>");
echo("<u>Prénom :</u> <b>" . $_SESSION["prenom"] . "</b><br>");
echo("<u>eMail :</u> <b>" . $_SESSION["email"] . "</b><br>");
//session_destroy();
?>
</body>
</html>
```

# Les sessions (9)

## ■ Les fonctions de sessions

`session_start()` -- Initialise les données de session

`session_id()` -- Affecte et/ou retourne l'identifiant de session courante

`session_name()` -- Affecte et/ou retourne le nom de la session courante

`session_register()` -- Enregistre une variable dans la session courante

`session_destroy()` -- Détruit toutes les données enregistrées d'une session

`session_is_registered()` -- Indique si une variable a été enregistrée dans la session ou pas

`session_unregister()` -- Supprime une variable dans la session courante

`session_unset()` -- Détruit toutes les variables de session

`session_cache_expire()` -- Retourne la date d'expiration du cache de la session

`session_save_path()` -- Affecte et/ou retourne le chemin de sauvegarde de la session courante

`session_decode()` -- Décode les données de session à partir d'une chaîne

`session_encode()` -- Encode les données de session dans une chaîne

# Les sessions (10)

## ■ Quelles sont les erreurs possibles ?

- **Répertoire de session inaccessible**

Warning: open(/tmp\sess\_3c80883ca4e755aa72803b05bce40c12, O\_RDWR)  
failed: m (2) in c:\phpdev\www\bp\header.php on line 2

Le répertoire de sauvegarde est défini dans le php.ini : session.save\_path = /tmp

Il faut donc

Créer un répertoire

Lui donner les droits d'écriture pour tous

En spécifier le chemin dans le **php.ini**

- **PHP n'est pas autorisé à utiliser les sessions**

Il faut s'assurer que le PHP est bien autorisé à créer des sessions. C'est juste un paramètre à activer. Faire un `phpinfo()` pour voir ces paramètres. La commande `phpinfo()` se contente d'afficher dans le navigateur le contenu du fichier de configuration `php.ini`.

# Les sessions (11)

- Quelles sont les erreurs possibles ?
  - Avoir déjà écrit dans la page

Warning: Cannot send session cookie - headers already sent by (output started at /home/SiteWeb/SiteAnalyse/index.php:3) in /home/SiteWeb/SiteAnalyse/index.php on line 6

Cette erreur survient lorsqu'on tente d'ouvrir une session après avoir déjà écrit dans le document, ce qui interdit.

Ce qu'il ne faut pas faire :

```
<html>
<body>
<?php session_start();>
...
ceci non plus :
```

```
<?php echo "<html>";>
...
session_start();
```

# PHP OO

# Les bases de la POO

## Introduction

- Une classe / un objet
- Constructeur / destructeur
- Visibilité
- Différences entre PHP 4 et PHP 5
- Classes abstraites et interfaces
- Héritage / surcharge
- Sérialisation
- Clonage

# Les bases de la POO

## Qu'est-ce qu'une classe

- Une classe peut être vue comme une abstraction.
- Elle contient des propriétés et des méthodes :
  - Propriété = variable de classe
  - Méthode = fonction de classe
- L'environnement de la classe est protégé :
  - Les propriétés et les méthodes peuvent être déclarées comme non accessibles en dehors de la classe.
  - Les propriétés ne se mélangent pas aux variables globales.

# Les bases de la POO

## Exemple de classe

```
class TaskManager
{
    public $taskTitle;
    private $xmlData;

    public function displayTasks()
    {
        // ...
    }
    private function cleanTasks()
    {
        // ...
    }
}
```

Propriétés

Méthodes

# Les bases de la POO

## Qu'est-ce qu'un objet

- Un objet est une instance de classe.
- Avec une classe, on peut créer autant d'objets qu'on veut (sauf si dans l'implémentation on décide de contrôler l'instanciation).
- Une classe est une déclaration, alors qu'un objet est contenu dans une variable, créé à partir de notre classe avec le mot clé « new ».
- Un objet est une instance de ma classe.

# Les bases de la POO

## Exemple de création d'objets

```
// Une classe Task
class Task
{
    public $title;
    public $description;
}

// Un objet "Task".
$task1 = new Task();

// Un deuxième objet "Task".
$task2 = new Task();

// Accès à un objet
$task1->title = "Comprendre la POO";
$task1->description = "La POO est magnifique car ...";
```

# Les bases de la POO

## Composition d'une classe

- La déclaration
  - Le 'moule' de la classe
- Les propriétés
  - Les 'variables' de la classe
- Les méthodes
  - Les 'fonctions' de la classe
- La classe mère, les interfaces
  - Elles font évoluer les possibilités et les fonctionnalités de la classe

# Les bases de la POO

## Exemple de classe (1/2)

- Un exemple simple de classe

```
<?php

// Déclaration de la classe
class User
{

    // Une propriété (variable de classe)
    var $name;

    // Une méthode (fonction de classe)
    function getName()
    {
        return ucfirst($this->name);
    }

}
```

# Les bases de la POO

## Exemple de classe (2/2)

- Un exemple simple de classe

```
// Création d'un objet à partir de la classe
$pierre = new User();

// Modification d'une propriété
$pierre->name = 'pierre';

// Accès à une méthode
echo $pierre->getName();

// Affiche 'Pierre'
```

# Les bases de la POO

## Exemple de classe

### ■ Création d'une classe et d'un objet

- **Une classe est composée de deux parties:**

- Les attributs: il s'agit des données représentant l'état de l'objet
- Les méthodes : il s'agit des opérations applicables aux objets

```
<?php
    class client {
        var $nom; var $ville; var $naiss ;
        function age() {
            $jour = getdate(); $an=$jour["year"]; $age = $an - $this->naiss;
            echo "Il a $age ans cette année <br />" ;}
        //création d'un objet
        $client1 = new client();
        //affectation des propriétés de l'objet
        $client1 -> nom = "Dupont" ; $client1-> naiss = "1961" ; $client1->ville =
            "Angers" ;
        //utilisation des propriétés
        echo "le nom du client1 est ", $client1->nom, "<br />" ;
        echo "la ville du client1 est ", $client1-> ville, "<br />" ;
        echo "le client1 est né en ", $client1->naiss, "<br />" ;
        //appel de la méthode age()
        $client1->age() ;
    ?>
```

# Les bases de la POO

## Exemple de classe

### *Constructeur*

Fonction appelée automatiquement lors de la création d'une instance. Le constructeur a le même nom que la classe en PHP4 (`__construct` en PHP5). Il peut avoir 0, 1 ou plusieurs arguments.

Ex : `Humain($nom, $prenom, $sexe, $habite, $an)`

```
class Humain{                                // Ex de classe en PHP 4
    // proprietes (donnees membres)
    var $nom;
    var $prenom;
    var $sexe;
    var $ville;
    var $annee;

    // constructeur
    function Humain($nom, $prenom, $sexe, $habite, $an){
        $this->nom      = $nom;
        $this->prenom   = $prenom;
        $this->sexe     = $sexe;
        $this->ville    = $habite;
        $this->annee    = $an;
    }

    // methode (fonction membre)
    function age(){
        return (date("Y") - $this->annee) ;
    }
}
```

# Les bases de la POO

## Manipulation des classes et des objets

- Manipulation des classes et des objets
  - Php n'inclue pas dans sa version 4 de niveaux de visibilité des éléments de la classe, il n'y a donc pas de concept d'encapsulation
  - Instanciation de la classe
    - **\$Nom\_de\_1\_objet = new Nom\_de\_la\_classe;**
  - Accéder aux propriétés d'un objet
    - **\$Nom\_de\_1\_objet->Nom\_de\_la\_donnee\_membre = Valeur;**
  - Accéder aux méthodes d'un objet
    - **\$Nom\_de\_1\_objet->Nom\_de\_la\_fonction\_membre(parametre1,parametre2,...);**
  - La variable \$this
    - **\$this->age = \$Age;**

# Les bases de la POO

## Manipulation des classes et des objets

### Créer et manipuler des instances de classe

#### Création

##### *Syntaxe*

```
$instance = new NomClasse();  
$instance = new NomClasse(arg1, ..., argN);
```

##### *Exemple*

```
$client1 = new Humain('Dupond', 'Pierre', 'M', 'Marseille', 1950);
```

#### Accès aux propriétés

##### *Syntaxe*

```
$nom_instance->nom_propriete
```

##### *Exemple*

```
echo "{$client1->prenom} {$client1->nom}"; // affichera Pierre Dupond
```

Un objet obj peut être *sérialisé* (lecture/écriture d'un objet) par session `$_SESSION['nomobj']`.

### Méthodes

##### *Syntaxe*

```
function nom_fonction(arg1, ..., argN){  
    // code fonction  
}
```

# Les bases de la POO

## Exemple de classe

```
<?php // fichier humain.php
class Humain{
    // donnees membres
    private $nom;           // variables d'instance
    private $prenom;
    private $sexe;
    private $ville;
    private $annee;
    private static $nb = 0; // variable de classe
    const doigts = 5;        // constante de classe

    // constructeur
    function construct($nom, $prenom, $sexe, $habite, $an){
        $this->nom      = $nom;
        $this->prenom    = $prenom;
        $this->setSexe($sexe);
        $this->ville     = $habite;
        $this->setAnnee($an);
        self::$nb++;
    }

    // accesseurs
    function getNom() { return $this->nom; }
    function getPrenom(){ return $this->prenom; }
    function getSexe() { return $this->sexe; }
    function getVille() { return $this->ville;}
    function getAnnee() { return $this->annee; }
```

# Les bases de la POO

## Exemple de classe

```
// modificateurs
function setAnnee($annee){
    if (($annee > 1900) && ($annee < date("Y"))){
        $this->annee = $annee;
    }
    else{ echo "annee $annee non valide pour $this->nom<br>" ; }
}

function setSexe($sexe){
    if (($sexe == 'M') || ($sexe == 'F')){
        $this->sexe = $sexe;
    }
    else{ echo "sexe $sexe incorrect pour $this->nom<br>" ; }
}

// fonction membre (methode d'instance)
function age(){
    if ($this->annee != 0){
        return (date("Y") - $this->annee) ;
    }
    else{ return "undefined" ; }
}

// fonction membre (methode de classe)
function nombre(){
    return self::$nb;
}

?>
```

# Les bases de la POO

## Exemple de classe

```
<?php
// inclusion de la classe Humain
require("humain.php");

// creation d'instances de la classe
$liste[] = new Humain('Dupond', 'Pierre', 'M', 'Marseille', 1950);
$liste[] = new Humain('Gantt', 'Paul', 'M', 'Paris', 1970);
$liste[] = new Humain('Dujardin', 'Anne', 'F', 'Bordeaux', "azer");
$liste[] = new Humain('Armand', 'Ludovic', 'x', 'Toulous', 1995);

// modification d'une proprietee d'une instance
$liste[1]->setAnnee(2007);

// affichage des instances
echo "<br>Liste :<br>";
foreach ($liste as $client){
    echo "- {$client->getPrenom()} {$client->getNom()} a ",
          $client->age(), " ans<br>";
}

// affichage des donnees de classe
echo "<br>il y a ", Humain::nombre(), " humains<br>";
echo "un humain a ", Humain::doigts, " doigts par main";
?>
```

# Les bases de la POO

## Manipulation des classes et des objets

### *this*

*this* est utilisé dans les méthodes pour faire référence à l'objet en cours (l'instance qui a reçu le message). Par exemple dans la fonction `age` `$this->année` fournira l'année de naissance de l'objet pour lequel une demande d'âge a été effectuée.

### *Envoi de message à l'objet*

```
$nom_instance->nom_methode(arg1, ..., argN)
```

### *Exemple*

```
echo "{$client1->prenom} {$client1->nom} a ", $client1->age(), " ans ";
// affichera Pierre Dupond a 56 ans
```

## Comparaisons d'objets

PHP 4 : deux instances de la même classe peuvent être comparées avec l'opérateur « == ». Deux objets sont égaux s'ils ont les mêmes valeurs et les mêmes attributs et s'ils sont des instances de la même classe.

PHP 5 :

- ✓ Opérateur « == » : deux instances sont égales si elles ont les mêmes attributs, les mêmes valeurs et sont des instances de la même classe
- ✓ opérateur « === » : deux objets sont identiques si et seulement si ils réfèrent à la même instance de la même classe.

# Les bases de la POO

## Le constructeur

- Comment marche un constructeur ?
  - Le constructeur est une méthode dite "magique" : elle est appelée automatiquement.
  - Le constructeur est appelé lors de l'instanciation d'une classe, c'est à dire la création d'un objet à partir d'une classe.
- A quoi ça sert ?
  - On appelle souvent le constructeur pour charger des propriétés (par exemple une connexion à la base de données).



En PHP4, le constructeur est la méthode qui porte le nom de la classe. Cette variante est encore valable en PHP5 bien que dépréciée.

# Les bases de la POO

## Le destructeur

- Comment marche un destructeur ?
  - De la même manière que le constructeur, le destructeur est appelé à la fin de la vie d'un objet.
  - Le destructeur sert généralement à faire des nettoyages.
- A quoi cela peut-il servir ?
  - Fermeture de la connexion à la base de données.
  - Nettoyage du contexte d'un objet (suppression de fichiers temporaires, vidage cache, etc.)



La fin de vie d'un objet peut subvenir à la fin du script, lorsque l'on supprime la variable qui contient l'objet (`unset`) ou lorsque l'on remplace son contenu (`affectation`).

# Les bases de la POO

## Exemple (construct, destruct)

- Un constructeur et un destructeur

```
class DB
{
    var $dbConnexion = null;

    var function __construct()
    {
        // Connexion à la base de données
    }

    function __destruct()
    {
        // Déconnexion à la base de données
    }
}
```

# Les bases de la POO

## La visibilité

- Application
  - La visibilité s'applique à une propriété ou une méthode
- Privé (private)
  - N'est accessible que dans la classe courante.
- Protégé (protected)
  - Est accessible dans la classe courante et dans les classes dérivées.
- Publique (public)
  - Est accessible partout, même à l'extérieur de la classe.

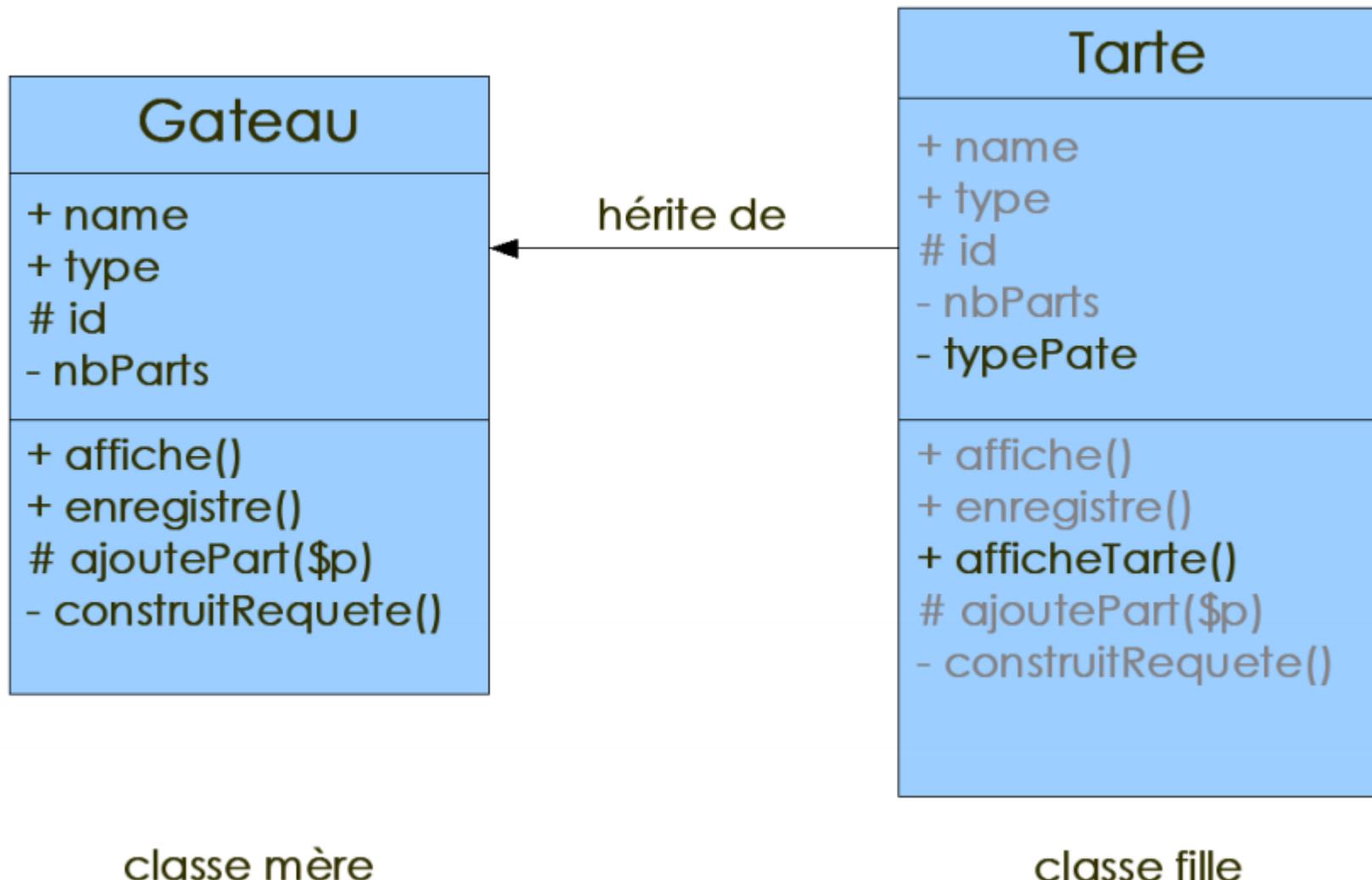
# Les bases de la POO

## L'héritage

- L'héritage permet de hiérarchiser les classes.
- Les classes mères sont génériques, elles mettent en place des fonctionnalités communes à une ou plusieurs classes.
- Les classes filles sont spécifiques, elles héritent des fonctionnalités des classes mères.

# Les bases de la POO

## L'héritage (UML)



# Les bases de la POO

## Héritage (1/2)

- Exemple d'héritage

```
<?php

class User // Une classe Utilisateur
{
    private $name;

    public function setName($name)
    {
        $this->name = $name;
    }

    public function getName()
    {
        return ucfirst($this->name);
    }
}
```

# Les bases de la POO

## Héritage (2/2)

- Exemple d'héritage

```
// Une classe 'Administrator' qui hérite de 'User'  
class Administrator extends User  
{  
    private $level = 'administrator';  
}  
  
// Administrator hérite de User donc on peut  
// s'en servir comme d'un User  
  
$admin = new Administrator();  
$admin->setName('jean');  
echo $admin->getName();
```

# Les bases de la POO

## Héritage

```
class Transport {  
    var $consommation;  
    var $vitesse;  
    var $nb_voyageurs;  
  
    function duree($km){  
        // calcul de la duree du trajet  
    }  
    function consomme($km){  
        // calcul de la consommation  
    }  
}  
  
class Voiture extends Transport{  
    var $marque_pneus;  
  
    function dist_freinage($vitesse, $etat_sol){  
        // calcul de la distance de freinage  
    }  
}  
  
class Voilier extends Transport{  
    var $nb_voiles;  
  
    function portance($incidence, $corde, $hauteur, $vitesse_vent){  
        // calcul de la portance  
    }  
}
```

# Les bases de la POO

## Héritage

### ■ L'opérateur ::

- faire référence à une fonction définie dans une super-classe à partir d'une classe héritant de cette dernière

```
class nouvelle_classe extends super_classe
{function fonction()
    {echo "Blocs d'instructions de la fonction fonction() . "
     dans la nouvelle-classe.";
        super_classe::fonction(); }
}
```

# Les bases de la POO

## Héritage

### ■ L'opérateur **parent**

- faire référence à des variables ou des fonctions présentes dans la super-classe à partir d'une autre classe héritant de cette dernière

```
class nouvelle_classe extends super_classe
{
    function fonction()
    {
        echo "Blocs d'instructions de la fonction fonction()"
        . " dans la nouvelle-classe.";
        // se réfère à la fonction fonction() de la super_classe
        parent::fonction();
    }
}
```

- Un héritage permet de regrouper des objets de manière verticale
  - Une classe mère regroupe plusieurs classes filles
  - Chaque classe fille hérite de fonctionnalités de la classe mère
- Une interface permet de regrouper des objets de manière transversale
  - Un groupe d'objets peut implémenter une interface
  - Une interface impose la présence de fonctionnalités dans ces objets
  - Si une classe mère implémente une interface, ses classes filles l'implémentent aussi par héritage

# Les bases de la POO

## Les interfaces : en pratique

- Une interface se déclare comme une classe, avec le mot clé 'interface'
- Une interface ne contient que des déclarations de méthodes, sans contenu
  - Ces méthodes ainsi déclarées doivent obligatoirement être redéclarées dans les classes qui implémentent l'interface

# Les bases de la POO

## Exemple d'interface (1/3)

- Définit un objet de type 'géométrique'

```
<?php

// Une interface 'Surface_Interface' qui impose 'getAire()'
interface Surface_Interface
{
    public function getAire();
}

class Rectangle implements Surface_Interface
{
    // ...
    public function getAire()
    {
        // ...
    }
}
```

# Les bases de la POO

## Exemple d'interface (2/3)

- Définit un objet de type 'géométrique'

```
// Implémente Geometric_Interface par héritage
// avec Rectangle
class Carre extends Rectangle
{
    // ...
}

// Une interface 'Volume' qui impose 'getVolume()'
interface Volume_Interface
{
    public function getVolume();
}
```

# Les bases de la POO

## Exemple d'interface (3/3)

- Définit un objet de type 'géométrique'

```
// Rectangle implémente Geometric_Interface
class Cube implements Surface_Interface, Volume_Interface
{
    // ...
    public function getAire()
    {
        // ...
    }

    public function getVolume()
    {
        // ...
    }
}
```

# Les bases de la POO

## Les classe abstraite

- Une classe abstraite ne peut être instanciée
  - En d'autres termes : on ne peut pas créer un objet avec une classe abstraite
- Une classe abstraite est destinée à être utilisée comme classe mère d'autres classes
- La déclaration d'une classe abstraite se fait avec le mot clé 'abstract'
  - `abstract class Parallelogramme { ... }`

# Les bases de la POO

## La sérialisation

- Principe

- Les objets en tant que tel ne peuvent être transmis d'un programme à l'autre via un réseau, car ils contiennent des caractères non imprimables
- Sérialisation = transformation d'une donnée complexe (objet, tableau) en une chaîne de caractère imprimable
- Désérialisation = transformation d'une chaîne sérialisée en sa donnée d'origine

- Utilité

- Transmission de données complexes
- Stockage de ces données en base ou dans des fichiers

# Les bases de la POO

## La sérialisation

- Exemple de sérialisation, désérialisation

```
<?php

class User
{
    public $name;
}

$user = new User();
$user->name = "Seth";

$serializedUser = serialize($user);
echo $serializedUser . "\n\n";

$user2 = unserialize($serializedUser);
print_r($user2);
```

# Les bases de la POO

## Sérialisation (affichage)

- Résultat du script précédent

```
0:4:"User":1:{s:4:"name";s:4:"Seth";}

User Object
(
    [name] => Seth
)
```

# Les bases de la POO

## Sérialisation

- Sauvegarde des objets
  - La sauvegarde et la relecture des objets s'effectuent respectivement par **serialize** et  **unserialize**
  - **serialize** permet de transformer un objet en une chaîne de caractères pouvant être facilement transmise à une autre page lors d'une session
  - **unserialize** permet de reconstituer l'objet à partir de la chaîne de caractères précitée

# Les bases de la POO

## Sérialisation

- Sauvegarde des objets : exemple

```
<?php
// Page de définition de la classe trigo.inc
class trigonometrie
{
    var $AB;
    var $BC;
    var $AC;
    function hypothenuse()
    {
        $resultat = sqrt(pow($this->BC, 2) + pow($this->AC, 2));
        return number_format($resultat, 2, ',', ',');
    }
}
?>
```

# Les bases de la POO

## Sérialisation

### ■ Sauvegarde des objets : exemple suite

```
<?php // Première page : saisie.php
    include("trigo.inc"); // inclusion de la définition de classe

$trigo = new trigonometrie; // crée une instance de l'objet

$objet_chaine = serialize($trigo); // sérialise l'objet

$fichier = fopen("fic", "w"); // ouvre un fichier en écriture seule

fputs($fichier, $objet_chaine); // écrit l'objet linéarisé dans le
                                // fichier

fclose($fichier); // ferme le fichier

?>
```

# Les bases de la POO

## Sérialisation

### ■ Sauvegarde des objets : exemple suite

```
<form action="resultat.php" method="post">
    <table border="0">
        <tr>
            <th colspan="2">
                <h3>Calcul de l'hypothénuse d'un triangle rectangle</h3>
            </th></tr>
        <tr>
            <td><u>longueur :</u></td>
            <td><input type="text" name="longueur" size="10" maxlength="10">
            </td></tr>
        <tr>
            <td><u>hauteur :</u></td>
            <td><input type="text" name="hauteur" size="10" maxlength="10">
            </td></tr>
        <tr>
            <th colspan="2"><input type="submit" value="Calculer"></th></tr>
    </table>
</form>
```

# Les bases de la POO

## Sérialisation

```
<?php // Seconde page : resultat.php
    include("trigo.inc"); // inclusion de la définition de classe
/* regroupe tous les éléments du tableau retourné par la fonction file dans une chaîne */
$objet_chaine = implode("", file("fic"));
$trigo = unserialize($objet_chaine); // désérialise l'objet
// appelle deux propriétés et une méthode de l'objet
$trigo->BC = $hauteur;
$trigo->AC = $longueur;
?>


| <h3>Calcul de l'hypothénuse d'un triangle rectangle</h3> |   |  |
|----------------------------------------------------------|---|--|
| hauteur (BC)                                             | = |  |
| <?php echo \$trigo->BC ?>                                |   |  |
| longueur (AC)                                            | = |  |
| <?php echo \$trigo->AC ?>                                |   |  |
| hypothénuse (AB)                                         | = |  |
| <?php echo \$trigo->hypotenuse() ?>                      |   |  |


```

# Les bases de la POO

## Sérialisation

- Les objets PHP sont des tableaux associatifs

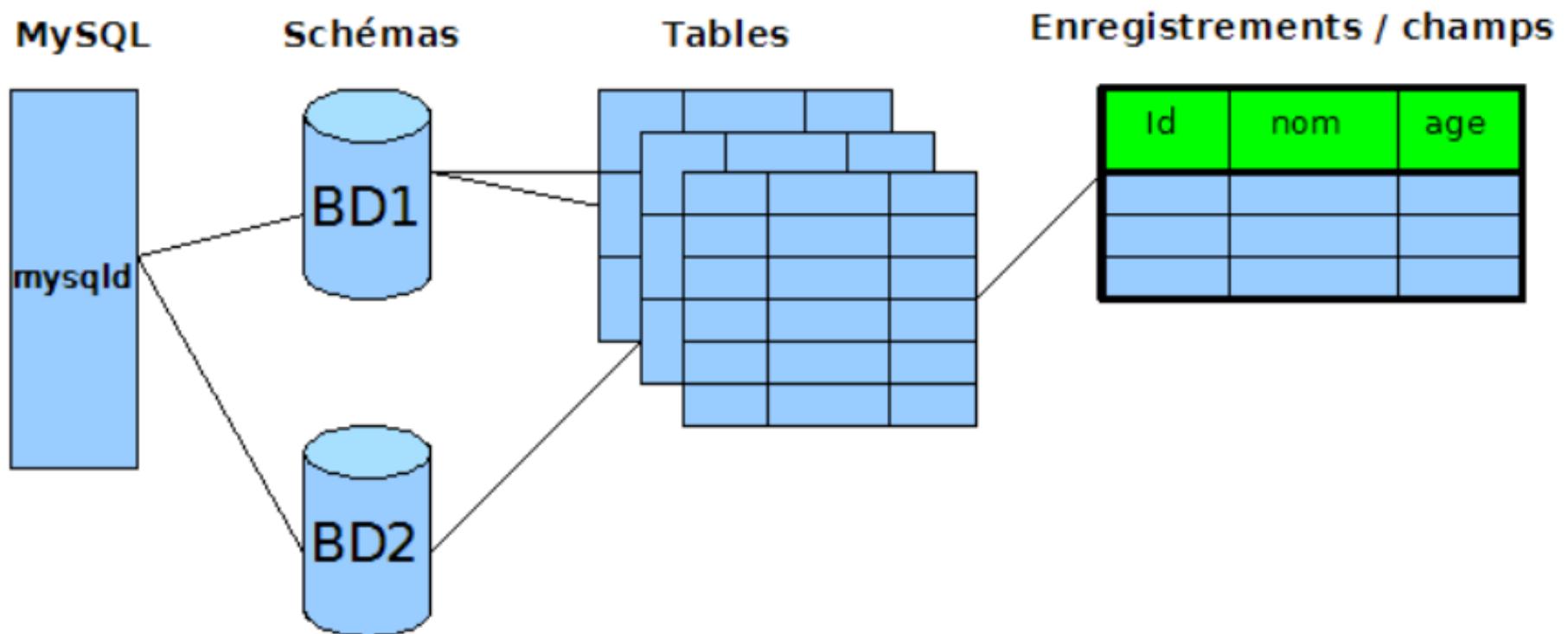
- Les noms des variables sont conçus comme des mots-clés
- Les valeurs des variables comme les éléments d'un tableau associatif

```
<?
Class ClasseTest {
var $col = "#0000E0" ;
var $txt= "Salut PHP" ;
var $ft = "Arial" ;
function ClasseTest() {
echo "<FONT FACE=\\" COLOR=\"$this->col\\" >$this-
>txt</FONT><br> ; } } ;

$obj = new ClasseTest;
Reset($obj);
Foreach ($obj as $key=>$elem){
Echo "$key=>$elem<br>" ;
} ?>
```

# MySql

# Connexion à une base de données Mysql



# Connexion à une base de données Mysql

- 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

# Connexion à une base de données Mysql

- Connexion au serveur MySQL

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

# Connexion à une base de données Mysql:

## *- 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

# Connexion à une base de données Mysql:

## - *Créer un fichier de configuration*

- 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);
```

# Connexion à une base de données Mysql:

## *- 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

# Connexion à une base de données Mysql:

- *Envoyer une requête au serveur*

- Envoi d'une requête

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

# Connexion à une base de données Mysql:

## *- 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

# Connexion à une base de données Mysql:

## - 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>';
}
```

# Connexion à une base de données Mysql:

## *- Fermer une connexion*

- La fonction `mysqli_close()` permet de fermer la connexion
- Elle prend en argument l'identifiant de connexion

# Connexion à une base de données Mysql:

## - Fermer une connexion

- mysqli\_close() : ferme la 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)
```

# PDO (Php Data Object)

# PDO

## Introduction

- PDO : **PHP Data Objects**
- Extension PHP fournissant une interface pour accéder à une base de données
- Fournit une interface d'**abstraction** pour l'accès aux données
- Ne fournit **PAS** une **abstraction** de base de données
  - SQL spécifique au moteur
  - Fonctionnalités présentes / absentes
- Interface orientée objet

# PDO

## Introduction

- PDO est une interface d'accès aux bases de données.
- PDO n'est pas un système d'abstraction complet
  - Il ne manipule pas les requêtes.
  - Il met à disposition des classes et des fonctions communes à un ensemble de SGBD.
  - Si certains SGBD ne disposent pas de certaines fonctionnalités (gestion des transactions, requêtes paramétrées, etc.), il les simule.

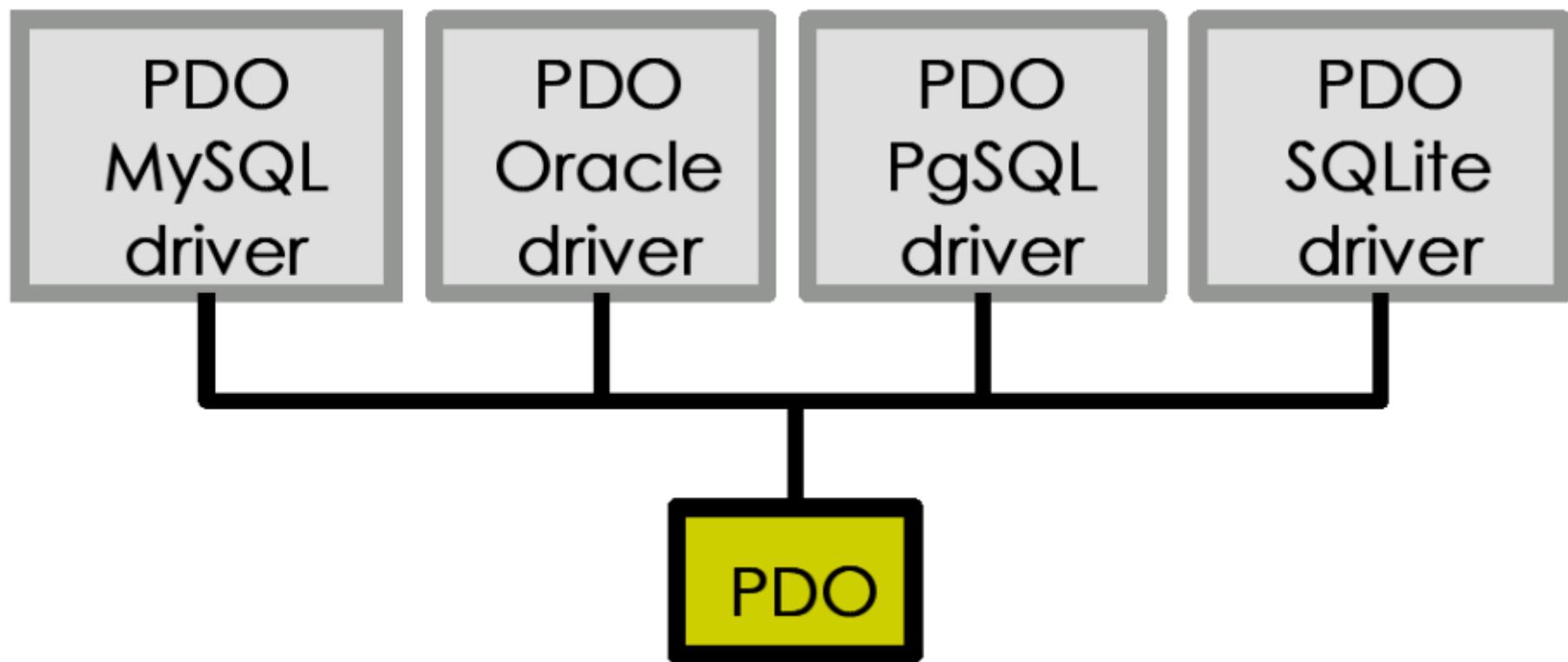
# PDO

## Architecture / principe

- PDO fonctionne avec un ensemble d'extensions
  - Une extension PDO de base qui définit l'interface commune.
  - Une extension Driver PDO relative à chaque SGBD que l'on veut gérer.

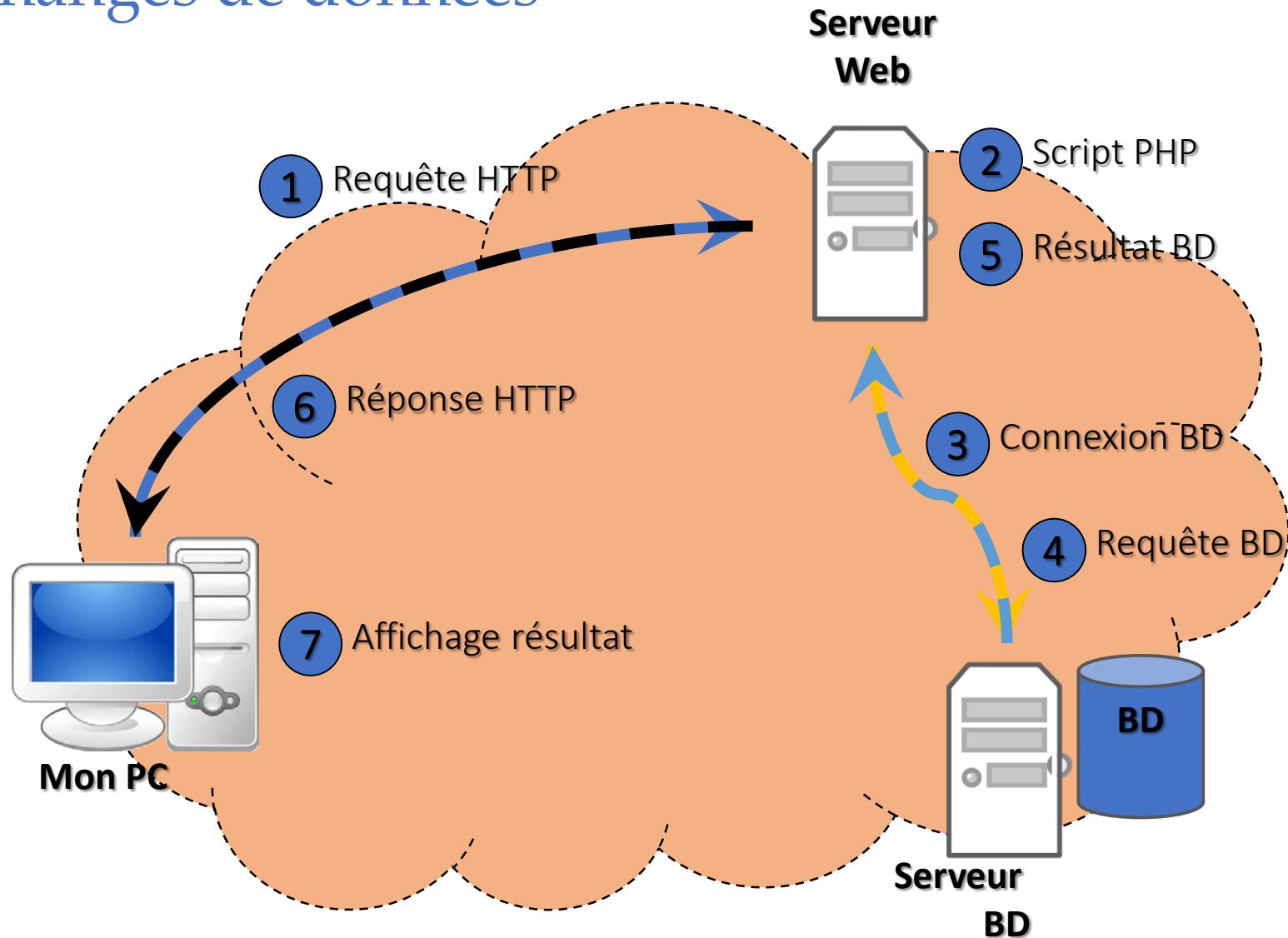
# PDO

## Architecture / principe



# PDO

## Échanges de données



# PDO

## Les classes de l'extension

- La classe PDO gère l'accès aux SGBD ainsi que les fonctionnalités de base :
  - Connexions
  - Lancement des requêtes
- La classe PDOStatement gère une liste de résultats.
- La classe PDOException est une classe d'exception personnalisée interne pour PDO.

# PDO

## Connexions et gestionnaire de connexion

- Instanciation d'un objet **PDO**
- `$dbh=new PDO(DSN [, user [, pass [, options]])`;
- *DSN* : **Data Source Name**
  - nom\_du\_driver:syntaxe\_spécifique\_au\_driver
  - Ex : mysql:host=localhost;dbname=ma\_base
- *user* : nom d'utilisateur, *pass* : mot de passe
- *options* : tableau associatif
  - spécifiques au driver
  - Ex : array(PDO::ATTR\_PERSISTENT => true)) ;
- Fin de connexion : `$dbh=null` ; ou `unset($dbh)` ;

# PDO

## Connexion simple à PDO

- Déclaration du DSN, connexion et requête

```
<?php

// DSN pour se connecter à MySQL
$dsn = 'mysql:host=localhost;dbname=mybase';

// Création d'un objet pour manipuler des requêtes
$dbh = new PDO($dsn, 'login', 'pass');

// Execution d'une requête SELECT
$result = $dbh->query('SELECT * from FOO');

// Itération sur les résultats d'une requête
foreach ($result as $row) {
    print_r($row);
}
```

## Gestion des erreurs de connexion

- Connexion par construction d'un objet
- Gestion envisageable des erreurs
  - Aucune
  - Fin brutale (exit, die)
  - État
  - Exception
- En cas d'erreur de connexion
  - Objet **PDOException** lancé
  - PDOException hérite de **Exception**

# PDO

## Gestion des erreurs de connexion

```
<?php
try {
    $dbh = new PDO('mysql:host=h;dbname=db',
                    $user, $pass) ;

...
$dbh = null ;
}
catch (PDOException $e) {
    echo "<p>Erreur: ".$e->getMessage() ;
    die();
}
?>
```

# PDO

## Gestion des erreurs de connexion

- PDO::ERRMODE\_SILENT (*par défaut*)
  - Mode silencieux, mise en place d'un code d'erreur
  - PDO : `errorCode()` / `errorInfo()`
  - PDOStatement : `errorCode()` / `errorInfo()`
- PDO::ERRMODE\_WARNING
  - Mise en place du code d'erreur
  - Émission d'une erreur de type E\_WARNING
- PDO::ERRMODE\_EXCEPTION
  - Mise en place du code d'erreur
  - Objet PDOException lancé

# PDO

## Gestion des erreurs de connexion

```
<?php
try {
    $dbh = new PDO('mysql:host=h;dbname=db',
                    $user, $pass) ;
    $dbh->setAttribute(PDO::ATTR_ERRMODE,
                        PDO::ERRMODE_EXCEPTION);

...
$dbh = null ;
}
catch (PDOException $e) {
    echo "<p>Erreur: ".$e->getMessage() ;
    die();
}
```

# PDO

## Gestion des erreurs de connexion

```
<?php  
$pdo = new PDO("mysql:host=localhost") ;  
$pdostat = $pdo->query("COUCOU")  
if ($pdo->errorCode()) {  
    echo "ERREUR !!\n" ;  
    echo "<pre>\n" ;  
    var_dump($pdo->errorInfo())  
    echo "</pre>\n" ;  
}  
  
ERREUR !!  
array(3) {  
    [0]=> string(5) "42000"  
    [1]=> int(1064)  
    [2]=> string(47) "Erreur de syntaxe près de 'COUCOU' à la ligne 1"  
}
```

Code SQLSTATE

Code erreur spécifique du driver

Chaîne erreur spécifique au driver

# PDO

## Gestion des erreurs de connexion

```
<?php  
try {  
    $pdo = new PDO("mysql:host=localhost") ;  
    $pdo->setAttribute(PDO::ATTR_ERRMODE,  
                      PDO::ERRMODE_EXCEPTION) ;  
    $pdostat = $pdo->query("COUCOU") ;  
}  
catch (Exception $e) {  
    echo "<p>ERREUR : ". $e->getMessage() ;  
}
```

ERREUR : SQLSTATE[42000]: Syntax error or access violation: 1064  
Erreur de syntaxe près de 'COUCOU' à la ligne 1

Code  
SQLSTATE

Chaîne erreur spécifique  
au driver

Code erreur spécifique  
du driver

# PDO

## Requêtes préparées

- Les requêtes préparées sont un modèle de requête enregistré sur le serveur le temps de l'exécution du script (par opposition aux procédures stockées qui sont stockées de manière permanente).
- Avec les requêtes paramétrées il n'est plus nécessaire de se protéger des attaques par injection SQL car le serveur sait ce qu'il attend.

# PDO

## Requêtes préparées

Une requête doit toujours être préparée !  
query est donc à bannir

- PDOStatement **PDO::query** ( string *statement* )

Résultat de requête

Requête

```
<?php
try {
    $pdo = new PDO("mysql:host=localhost") ;
    $pdostat = $pdo->query("SELECT * FROM clients") ;
}
catch (Exception $e) {
    echo "<p>ERREUR : ".$e->getMessage() ;
}
```

# PDO

## Exploitation des résultats d'une requête

- Récupération des données ligne à ligne
- Une ligne peut être :
  - un tableau indexé
  - un tableau associatif
  - un tableau mixte (par défaut)
  - un objet anonyme
  - un objet d'une classe définie par l'utilisateur
- Récupération des données d'une colonne

# Parcours du résultat d'une requête

- Parcourir le résultat de la requête

```
SELECT *  
FROM morceau  
ORDER BY mor_id
```

Résultat de requête

Curseur interne

mor_id	mor_nom
872	With A Little Help From My Friends
873	The Letter
874	Marjorine
875	Midnight Rider
876	You Are So Beautiful
877	Feelin' Alright
878	Cry Me A River
...	

The diagram illustrates the traversal of a query result. A blue rounded rectangle represents the result set, containing a table with columns 'mor\_id' and 'mor\_nom'. The table data includes song IDs (872-878) and their names. Three yellow arrows point from the left towards the first three rows of the table. A diagonal line with an arrowhead points from the text 'Curseur interne' (internal cursor) to the second row of the table. A grey bracket labeled 'Résultat de requête' (query result) is positioned above the blue box.

```
+-----+-----+  
| mor_id | mor_nom |  
+-----+-----+  
| 872 | With A Little Help From My Friends |  
| 873 | The Letter |  
| 874 | Marjorine |  
| 875 | Midnight Rider |  
| 876 | You Are So Beautiful |  
| 877 | Feelin' Alright |  
| 878 | Cry Me A River |  
...  
+
```

# Exploitation des résultats d'une requête (1)

```
try {
    $pdo=new PDO("mysql:host=localhost;dbname=mysql") ;
    $pdo->setAttribute(PDO::ATTR_ERRMODE,
                        PDO::ERRMODE_EXCEPTION);
    $pdostat = $pdo->query("SELECT name FROM user") ;
    $pdostat->setFetchMode(PDO::FETCH_ASSOC) ;
    foreach ($pdostat as $ligne) {
        echo "<p>" . $ligne['name'] . "\n" ;
    }
}
catch (Exception $e) {
    echo "<p>ERREUR : ".$e->getMessage() ;
}
```

## Exploitation des résultats d'une requête (2)

```
try {
    $pdo=new PDO("mysql:host=localhost;dbname=mysql") ;
    $pdo->setAttribute(PDO::ATTR_ERRMODE,
                        PDO::ERRMODE_EXCEPTION);

    $pdostat = $pdo->query("SELECT name FROM user") ;
    foreach ($pdostat->fetchAll(PDO::FETCH_ASSOC)
            as $ligne) {
        echo "<p>" . $ligne['name'] . "\n" ;
    }
}
catch (Exception $e) {
    echo "<p>ERREUR : ".$e->getMessage() ;
}
```

# Exploitation des résultats d'une requête (3)

```
try {
    $pdo=new PDO("mysql:host=localhost;dbname=mysql") ;
    $pdo->setAttribute(PDO::ATTR_ERRMODE,
                        PDO::ERRMODE_EXCEPTION);
    $pdostat = $pdo->query("SELECT name FROM user") ;
    while ($ligne
          = $pdostat->fetch(PDO::FETCH_ASSOC)) {
        echo "<p>" . $ligne['name'] . "\n" ;
    }
}
catch (Exception $e) {
    echo "<p>ERREUR : ".$e->getMessage() ;
}
```

# Modes de récupération des données (1)

- **PDO::FETCH\_ASSOC**

- retourner **chaque ligne** dans un **tableau indexé par les noms des colonnes** comme elles sont retournées dans le jeu de résultats correspondant. Si le jeu de résultats contient **de multiples colonnes avec le même nom**, PDO::FETCH\_ASSOC retourne **une seule valeur par nom de colonne**.

- **PDO::FETCH\_NUM**

- retourner **chaque ligne** dans un **tableau indexé par le numéro des colonnes** comme elles sont retournées dans le jeu de résultats correspondant, en **commençant à 0**.

# Modes de récupération des données (2)

- **PDO::FETCH\_BOTH** (*par défaut*)

- retourner **chaque ligne** dans un **tableau indexé par les noms des colonnes ainsi que leurs numéros**, comme elles sont retournées dans le jeu de résultats correspondant, en **commençant à 0**.

- **PDO::FETCH\_OBJ**

- retourner **chaque ligne** dans un **objet avec les noms de propriétés correspondant aux noms des colonnes** comme elles sont retournées dans le jeu de résultats.

# Exemple avec PDO::FETCH\_CLASS

```
$stmt = $pdo->query(<<<SQL
    SELECT id, name
    FROM artist
    WHERE id = 12
SQL
) ;
$stmt->setFetchMode(PDO::FETCH_CLASS, 'Artist') ;
if (($object = $stmt->fetch()) !== false) {
    return $object ;
}
```

Instancie un objet de la classe Artist  
dont les attributs sont supposés être id et name

# Préparation d'une requête

- PDOStatement **PDO::prepare(string statement [, array driver\_options])**
  - *statement* : la requête à préparer. Peut contenir des paramètres anonymes (?) ou nommés (:nom)
  - *driver\_options* : tableau d'options du driver
  - retourne un objet **PDOStatement** qui effectuera l'association des paramètres et exécutera la requête

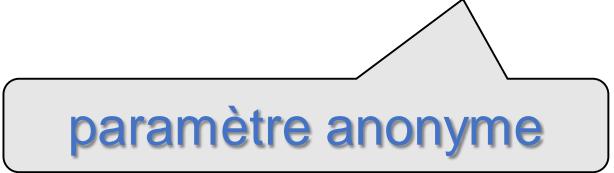
```
$pdo=new PDO( "mysql:host=localhost;dbname=mysql" ) ;  
$pdostat = $pdo->prepare(  
    "SELECT * FROM user WHERE User= ?" ) ;
```

# Association des paramètres d'une requête

- bool **PDOStatement::bindValue**(*mixed parameter, mixed value [, int data\_type]*)
  - *parameter* : le paramètre (nom ou position [1...n])
  - *value* : sa valeur
  - *data\_type* : le type de la valeur
    - **PDO::PARAM\_BOOL** booléen.
    - **PDO::PARAM\_NULL** NULL SQL.
    - **PDO::PARAM\_INT** INTEGER SQL.
    - **PDO::PARAM\_STR** CHAR, VARCHAR ou autre chaîne.
    - **PDO::PARAM\_LOB** "objet large" SQL.
- bool **PDOStatement::execute([array parameters])**
  - *parameters* : tableau associatif ou indexé des valeurs

# Préparation puis exécution d'une requête (1)

```
$pdo=new PDO("mysql:host=localhost;dbname=mysql") ;  
$pdo->setAttribute(PDO::ATTR_ERRMODE,  
                    PDO::ERRMODE_EXCEPTION);  
  
$pdostat = $pdo->prepare(  
    "SELECT * FROM user WHERE User= ?");  
  
$pdostat->bindValue(1, 'root') ;  
$pdostat->execute() ;  
// Utilisation du résultat  
  
$pdostat->bindValue(1, 'cutrona') ;  
$pdostat->execute() ;  
// Exécution de la requête
```



paramètre anonyme

## Préparation puis exécution d'une requête (2)

```
$pdo=new PDO("mysql:host=localhost;dbname=mysql") ;  
$pdo->setAttribute(PDO::ATTR_ERRMODE,  
                    PDO::ERRMODE_EXCEPTION);  
  
$pdostat = $pdo->prepare(  
    "SELECT * FROM user WHERE User= ':utilisateur'" );  
$pdostat->bindValue('utilisateur', 'root') ;  
$pdostat->execute() ;  
// Utilisation du résultat  
$pdostat->bindValue('utilisateur', 'cutrona') ;  
$pdostat->execute() ;  
// Exécution de la requête
```

paramètre nommé

# Préparation puis exécution d'une requête (3)

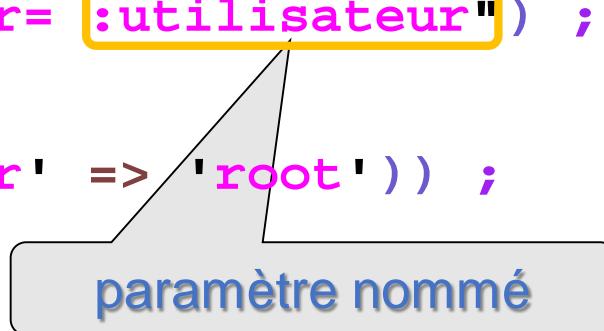
```
$pdo=new PDO("mysql:host=localhost;dbname=mysql") ;  
$pdo->setAttribute(PDO::ATTR_ERRMODE,  
                    PDO::ERRMODE_EXCEPTION);  
  
$pdostat = $pdo->prepare(  
    "SELECT * FROM user WHERE User=?");  
$pdostat->execute(array('root')) ;  
// Utilisation du résultat  
$pdostat->execute(array('cutrona')) ;  
// Utilisation du résultat
```

paramètre anonyme

Exécution de la requête

# Préparation puis exécution d'une requête (4)

```
$pdo=new PDO("mysql:host=localhost;dbname=mysql") ;  
$pdo->setAttribute(PDO::ATTR_ERRMODE,  
                      PDO::ERRMODE_EXCEPTION);  
  
$pdostat = $pdo->prepare(  
    "SELECT * FROM user WHERE User= :utilisateur") ;  
$pdostat->execute(  
    array('utilisateur' => 'root')) ;  
// Utilisation du résultat  
$pdostat->execute(  
    array('utilisateur' => 'cutrona')) ;  
// Exécution de la requête
```



paramètre nommé

# Intérêt des requêtes préparées

- Amélioration des performances en cas d'exécutions répétées
- Émulation faite par PDO si le driver ne les supporte pas nativement
- Protection automatique des valeurs des paramètres pour **interdire les attaques par injection de code SQL**

# La gestion des fichiers avec PHP

# La gestion des fichiers avec PHP (1)

## ■ Principe

- PHP prend en charge l'accès au système de fichiers du système d'exploitation du serveur
- Les opérations sur les fichiers concernent la création, l'ouverture, la suppression, la copie, la lecture et l'écriture de fichiers
- Les possibilités d'accès au système de fichiers du serveur sont réglementées par les différents droits d'accès accordés au propriétaire, à son groupe et aux autres utilisateurs
- La communication entre le script PHP et le fichier est repérée par une variable, indiquant l'état du fichier et qui est passée en paramètre aux fonctions spécialisées pour le manipuler

# La gestion des fichiers avec PHP (2)

## ■ Ouverture de fichiers

- La fonction **fopen()** permet d'ouvrir un fichier, que ce soit pour le lire, le créer ou y écrire

:entier **fopen(chaine nom du fichier, chaine mode);**

- **mode** : indique le type d'opération qu'il sera possible d'effectuer sur le fichier après ouverture. Il s'agit d'une lettre (en réalité une chaîne de caractères) indiquant l'opération possible:
  - **r** (comme read) indique une ouverture en lecture seulement
  - **w** (comme write) indique une ouverture en écriture seulement (la fonction crée le fichier s'il n'existe pas)
  - **a** (comme append) indique une ouverture en écriture seulement avec ajout du contenu à la fin du fichier (la fonction crée le fichier s'il n'existe pas)
- lorsque le mode est suivie du caractère + celui-ci peut être lu et écrit
- le fait de faire suivre le mode par la lettre **b** entre crochets indique que le fichier est traité de façon binaire.

# La gestion des fichiers avec PHP (3)

## ■ Ouverture de fichiers

Mode	Description
r	ouverture en lecture seulement
w	ouverture en écriture seulement (la fonction crée le fichier s'il n'existe pas)
a	ouverture en écriture seulement avec ajout du contenu à la fin du fichier (la fonction crée le fichier s'il n'existe pas)
r+	ouverture en lecture et écriture
w+	ouverture en lecture et écriture (la fonction crée le fichier s'il n'existe pas)
a+	ouverture en lecture et écriture avec ajout du contenu à la fin du fichier (la fonction crée le fichier s'il n'existe pas)

# La gestion des fichiers avec PHP (4)

## ■ Ouverture de fichiers

- Exemple

```
$fp = fopen("fichier.txt", "r"); //lecture
```

```
$fp = fopen("fichier.txt", "w"); //écriture depuis début du fichier
```

- De plus, la fonction fopen permet d'ouvrir des fichiers présents sur le web grâce à leur URL.
  - Exemple : un script permettant de récupérer le contenu d'une page d'un site web:

```
<?
$fp = fopen("http://www.mondomaine.fr", "r"); //lecture du fichier
while (!feof($fp)) { //on parcourt toutes les lignes
    $page .= fgets($fp, 4096); // lecture du contenu de la ligne}
?>
```

# La gestion des fichiers avec PHP (5)

## ■ Ouverture de fichiers

- Il est généralement utile de tester si l'ouverture de fichier s'est bien déroulée ainsi que d'éventuellement stopper le script PHP si cela n'est pas le cas

```
<?
if (!$fp = fopen("fichier.txt", "r")) {
echo "Echec de l'ouverture du fichier";
exit;}
else {// votre code;}
?>
```

- Un fichier ouvert avec la fonction **fopen()** doit être fermé, à la fin de son utilisation, par la fonction **fclose()** en lui passant en paramètre l'entier retourné par la fonction fopen()

# La gestion des fichiers avec PHP (6)

## ■ Lecture et écriture de fichiers

- Il est possible de lire le contenu d'un fichier et d'y écrire des informations grâce aux fonctions:
  - **fputs()** (ou l'alias **fwrite()**) permet d'écrire une chaîne de caractères dans le fichier. Elle renvoie 0 en cas d'échec, 1 dans le cas contraire
    - booléen **fputs(entier Etat\_du\_fichier, chaine Sortie);**
  - **fgets()** permet de récupérer une ligne du fichier. Elle renvoie 0 en cas d'échec, 1 dans le cas contraire
    - **fgets(entier Etat\_du\_fichier, entier Longueur);**
      - Le paramètre Longueur désigne le nombre de caractères maximum que la fonction est sensée récupérer sur la ligne
      - Pour récupérer l'intégralité du contenu d'un fichier, il faut insérer la fonction **fgets()** dans une boucle while. On utilise la fonction **feof()**, fonction testant la fin du fichier.

# La gestion des fichiers avec PHP (7)

## ■ Lecture et écriture de fichiers

```
<?
if (! $fp = fopen("fichier.txt", "r"))
    {echo "Echec de l'ouverture du fichier";}
else { $Fichier= "";
    while(!feof($fp)) {
        // On récupère une ligne
        $Ligne = fgets($fp, 255);
        // On affiche la ligne
        echo $Ligne;
        // On stocke l'ensemble des lignes dans une variable
        $Fichier .= $Ligne. "<BR>";
    }
    fclose($fp); // On ferme le fichier
}
?>
```

# La gestion des fichiers avec PHP (8)

## ■ Lecture et écriture de fichiers

- Pour stocker des informations dans le fichier, il faut dans un premier temps ouvrir le fichier en écriture en le créant s'il n'existe pas
  - Deux choix : le mode 'w' et le mode 'a'.

```
<?
$nom= "Jean"; $email= "jean@dupont.fr";
$fp = fopen( "fichier.txt" , "a" ); // ouverture du fichier en
// écriture
fputs($fp, "\n"); // on va à la ligne
fputs($fp, $nom." | ".$email); // on écrit le nom et email dans
// le fichier
fclose($fp);
?>
```

# La gestion des fichiers avec PHP (9)

## ■ Les tests de fichiers

➤ **is\_dir()** permet de savoir si le fichier dont le nom est passé en paramètre correspond à un répertoire.

- La fonction `is_dir()` renvoie 1 s'il s'agit d'un répertoire, 0 dans le cas contraire

booléen `is_dir(chaine Nom_du_fichier);`

➤ **is\_executable()** permet de savoir si le fichier dont le nom est passé en paramètre est exécutable.

- La fonction `is_executable()` renvoie 1 si le fichier est exécutable, 0 dans le cas contraire

booléen `is_executable(chaine Nom_du_fichier);`

# La gestion des fichiers avec PHP (10)

## ■ Les tests de fichiers

- **is\_link()** permet de savoir si le fichier dont le nom est passé en paramètre correspond à un lien symbolique. La fonction **is\_link()** renvoie 1 s'il s'agit d'un lien symbolique, 0 dans le cas contraire  
**booléen is\_link(chaine Nom\_du\_fichier);**

- **is\_file()** permet de savoir si le fichier dont le nom est passé en paramètre ne correspond ni à un répertoire, ni à un lien symbolique.

- La fonction **is\_file()** renvoie 1 s'il s'agit d'un fichier, 0 dans le cas contraire

**booléen is\_file(chaine Nom\_du\_fichier);**