Base de donnée 2: PHP and Mysql

Crée par :
Lamiae DOUNAS
Lamiae.dounas@gmail.com

Plan du cours

1. Rappel SQL
2. Introduction au web et PHP
3. Variables et constantes
4. Opérateurs
5. Tableaux
6. Structures de contrôles
7. Fonctions
8. Gestion des formulaires
9. Programmation orienté objet (POO)
10. Accès aux bases de données(Mysql)
11. cinq mini Projets à réaliser

RAPPEL SQL

- Travaux pratiques :

Révision des opérations : CREATE TABLE, INSERT, UPDATE, ALTER TABLE, ADD CONSTRAINTS dans TP1.

Introduction au Web

Qu’est ce que le web?

- Terme crée par par Tim Berners-Lee

(désigne toile d’araignée (en anglais)).

- Enorme réseau d’ordinateurs connectés qui hébergent des sites web.

Quel rapport entre le web et internet ?

- Web est une application d’internet juste comme la messagerie instantanée!!

Introduction au Web

Qu'est ce qu'un site web?

- Ecrit en 3 langages: html pour marquer le contenu du site. Css pour gérer le style et javascript pour le rendre interactif.
- Accessible depuis une dresse unique : URL(Uniform Resource Locator)

Protocole	Mot de passe (facultatif)	Nom du serveur	Port (Facultatif Si 80)	chemin
http://	user:password@	www.ccm.net		glossair/ glossair.php

Introduction au PHP

Php : php hypertext preprocessor.

- Langage interprété pour créer des sites dynamiques.
- Langage de script côté serveur.
- langage « Embedded HTML ».
- Open source : PHP a permis de créer un grand nombre de sites web célèbres, comme Facebook, Wikipédia, etc.

Introduction au PHP

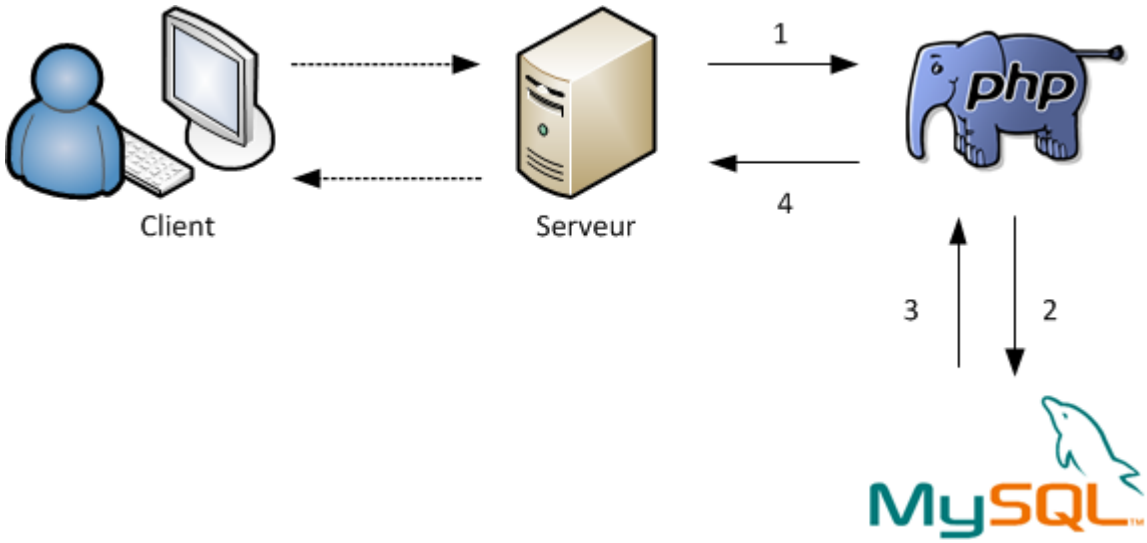
Site statique vs site dynamique

- Site statique : écrit en HTML(éventuellement du JavaScript) , même contenu pour tous les utilisateurs.

- Site Dynamique : contenus adaptés aux besoins des utilisateurs (mais nécessite un temps de traitement).

Introduction au PHP

Relation Client/Serveur.



Installation et configuration de PHP

Il suffit de télécharger le serveur web :

- WAMP : http://www.wampserver.com/
- MAMP : http://www.mamp.info/
- XAMPP : https://www.apachefriends.org/fr/

Et Editeur de texte :

- Sublim Text : http://www.sublimetext.com/2

Introduction au PHP

Syntaxe de base :

```
<!doctype html>
<html>
  <head>
<title>Titre</title>
  </head>
 <body>
<?php echo "Hello World !"; ?>
</body>
 </html>
```

<!DOCTYPE HTML PUBLIC "-//W3C//
DTD HTML 4.01//EN" "http://
www.w3.org/TR/html4/strict.dtd">

Introduction au PHP

Commentaires :

- Commentaires hérités du langage C et Perl

// Ceci est un commentaire sur une seule ligne

/ Ceci est un commentaire sur*
plusieurs lignes */

- Commentaire style shell

Ceci est un commentaire sur une seule ligne

Types de données

PHP supporte les types de données suivants :

- nombres entiers,
- nombres à virgule flottante,
- chaînes de caractères,
- tableaux,
- objets (développés dans la section programmation orientée objet).

La fonction « **gettype »** permet de connaitre le type de la variable. *Si la variable n'est pas définie, elle renvoie "string".*

Ex : $a= 12;
echo gettype($a) ; // => "integer »

Types de données

Tous les noms de variables sont précédés d'un $:

Pour spécifier une variable de type *entier :*

- *$toto = 123 ; # est un entier en base 10,*
- *$toto = -123 ; # est un entier négatif.*

Pour spécifier une variable de type *chaîne de caractères* :

- *$personne = '$toto Smith' ; # est une chaîne de caractères pour afficher les caractères spéciaux.*
- *$personne = "M. Smith" ; # est aussi une chaîne de caractères.*
- *$chaine=« chaine1» . «chaine2» ; // affiche chaine1chaine2*
- *$chaine=« chaine1 » . $toto ; // affiche chaine1 123*

Types de données

Opérateur sur les chaînes de caractères :

– concaténation : chaine1 . Chaine2

Opérateurs logiques :

- AND ou && (vrai si $a et $b vrais)
- OR ou || (vrai si $a ou $b sont vrais)

Opérateurs arithmétiques :

- addition : $a + $b,
- soustraction : $a - $b,
- multiplication : $a * $b,
- division : $a / $b,
- modulo (reste de la division entière) : $a % $b.

Types de données

Opérateurs arithmétiques :

- Attention : lorsqu'une chaine de caractère est évaluer comme un une valeur numérique, les règles suivantes s'appliquent :
- $toto = 1 + "4.5" ; # $toto vaut 5.5
- $toto = 1 + "titi + 149" ; # $toto vaut 1 car la chaine vaut 0 si c'est du texte ou,
- $toto = 1 + "149 + titi" ; # $toto vaut 150 car la chaine vaut 149 (commence par une valeur numérique).

Types de données

Opérateurs de comparaison :

- égal à : $a == $b,
- différent de : $a != $b,
- supérieur à : $a > $b,
- inférieur à : $a < $b,
- supérieur ou égal à : $a >= $b,
- inférieur ou égal à : $a <= $b.

Exemple : echo $toto == 0 ? " Vrai" : " Faux" ;

Types de données

Date

Déclaration avec :

- *DATETIME*

  $date = new DateTime('2000-01-05');

- DATE_CREATE : un Alias de DateTime::__construct()

$date2 = date_create('2000-01-01');

Extraction du Mois, année , …… :

- Avec style procédural en utilisant DATE_FORMAT

echo date_format($date, "m");

- Avec style orienté objet (OO) :

echo $date->format('Y');

Types de données

Date

Exemples :

- $date=date("d-m-y ");

  echo « ceci est la date du jour » .$date ;

- $heure = date("h:i:s");

  echo "c'est l'heure du jour " .$heure ;

Constantes

- Les constantes sont des variables dont la valeur ne peut pas changer durant la durée de vie d'un script.
- Les constantes sont utilisées pour définir les paramètres des applications web tq : racine du site, nom de la base de données….
- Le nom de constante valide commence par une lettre ou un souligné (_), suivi d'un nombre quelconque de lettre, chiffres ou soulignés
- On définit une constante en utilisant la fonction define() ;

  Exemple : define("MA_CONSTANTE", "Hello") ;
- Pour afficher une constante :

  echo MA_CONSTANTE ou echo const("MA_CONSTANTE");

Exemple : PHP définit certaines constantes comme PHP_OS qui indique le système d'exploitation utilisé par la machine qui fait tourner le PHP (ex : Linux) .

Tableaux

Les tableaux de PHP ressemblent aux tableaux associatifs(*hash-tables*).

- L’ index est appelé *clé*
- La valeur associée à la clé est appelée *valeur .*

On déclare un tableau avec deux façons :

- Utiliser la fonction array() pour créer un tableau ;
- Ou affecter directement les valeurs au tableau ;

Tableaux

Fonctions sur les tableaux :

- sizeof() : retourne le nombre d'éléments d'un tableau, ou
- count() : retourne le nombre d'éléments d'un tableau s'il existe, 1 si la variable n'est pas un tableau et 0 si la variable n'existe pas.

Tableaux

Exemple:

- $suite = array(1, 2, 3, 4) ;
- $tab[0] = 1 ;
- $tab[1] = "toto" ; # on peut mélanger les contenus
- $tab[" chaine"] = " valeur" ; # on peut mélanger les clés.
- $personne = array("type" => "M.", "nom" =>
- "Smith") ;

Tableaux

Parcourir un tableau :

```
<?php
// On crée notre array $prenoms
$prenoms = array ('François', 'Michel', 'Nicole', 'Véronique', 'Benoît');

// Puis on fait une boucle pour tout afficher :
for ($numero = 0; $numero < 5; $numero++)
{
   echo $prenoms[$numero] . '<br />';
}
?>
```

Tableaux

Parcourir un tableau associatif :

```
<?php
  $personne = array("type" => "M.", "nom" =>
 "Smith") ;
  foreach($personne as $cle => $valeur) {
   echo "cle=" .$cle .“   valeur= " . $valeur ;
  }
```

Conditions : « if … else »

Prototype :

```
if (condition1) {
    # instructions à exécuter si la condition1 est vraie...
} elseif (condition2) {
    # instructions à exécuter si la condition2 est vraie...
    ...
} else {
    # instructions à exécuter si aucune des conditions n'est vraie...
}
```

- Les fonctionnalités de l'instruction if sont les mêmes en PHP qu'en C;

Conditions : « if … else »

Exemple :

```
if ($type == "Femme") {
echo "Bonjour Madame" ;
} elseif ($type == "Homme") {
echo "Bonjour Monsieur" ;
} else {
echo "Bonjour, vous êtes bizarre !" ;
}
```

Conditions : « switch »

Prototype :

```
switch ($var) {
    case val1:
        # instructions à exécuter si val1…
        break;
    case val2:
        # instructions à exécuter si val2…
        break;
    default:
        # Ce cas est utilisé lorsque tous les autres cas ont échoué.
        break;
}
```

Boucles : while

La boucle while est le moyen le plus simple d'implémenter une boucle en PHP;

- Prototype :

```
while (condition) {
# instructions à exécuter tant que la condition est vraie...
}
```

Prototype simplifié :

```
while (condition) :
# instructions à exécuter tant que la condition est vraie...
endwhile ;
```

Boucles : while

Exemple :

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

- Affiche 1 2 3 4 5 6 7 8 9 10.

Boucles : For

La boucle « for » fonctionne comme la boucle for du langage C;

Prototype :

```
for (expression1 ; condition ; expression2)
 {
# instructions à exécuter tant que la condition est vraie...
}
```

Prototype simplifié :

```
for (expression1 ; condition ; expression2) :
# instructions à exécuter tant que la condition est vraie...
endfor;
```

Boucles : For

Exemple :

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

- Affiche 1 2 3 4 5 6 7 8 9 10.

Exercices 1:

Écrire un script PHP pour afficher la version du PHP et d’autres information sur la configuration.

Exercices 1:

Solution 1 :

```
<?php   phpinfo();    ?>
```

Solution 2 :

```
<?php
   echo "<pre>";
   print_r($GLOBALS); //affiche le nom des super globales.
   echo "</pre>";
   ?>
```

Exercices 2:

$var = 'PHP Tutorial'.

1-Mettre la variable var dans une section h3 tag , la centrer et l'afficher avec une couleur rouge dans un document HTML .

2-Souligner le titre

3- Ajouter le paragraphe suivant : " Ceci est un paragraphe".

Exercices 2:

Solution :

```
<?php
$var = 'PHP Tutorial';
?>
<!DOCTYPE html>
<html>
<head> <title><?php echo $var; ?> </title></head>
  <body>
  <u><h3> <font color="red">
  <?php
    echo $var;
   ?></h3></u>
</font>
  <p> ceci est un paragraphe.</p>
</body>
</html>
```

Exercices 3:

compter de 10 jusqu’à 22

Exercices 3:

Solution

```
for ($i = 10 ; $i <= 22 ; $i++) {
echo "$i " ;
}
```

Fonctions

Une fonction peut être définie par le mot clé « function » en utilisant la syntaxe suivante :

```
function nomd-DelaFonction ($arg_1, $arg_2, /* ..., */ $arg_n) {
    # instructions...
    return $valeur_de_retour ;   // optionnel
}
```

IL n'y a pas de distinction fonctions / procédures en PHP.

Fonctions

Les fonctions n'ont pas besoin d'être définies avant d'être utilisées, SAUF lorsqu'une fonction est définie conditionnellement ou à l'intérieur d'une autre fonction :

1. Une fonction conditionnelle :

```
// imposible d'appeler foo() car la fonction n'existe pas
if (condition) {  function foo()   {
        echo "Je n'existe pas tant que le programme
        n'est pas     passé ici.\n";   } }
```

Fonctions

2. Une fonction dans une autre fonction

```
function foo() {
    function bar()    {
        echo "Je n'existe pas tant que foo() n'est
        pas appelé.\n";   } }
/* Impossible d'appeler bar() ici, car il n'existe pas. */
 foo();
/* Maintenant, nous pouvons appeler bar(),  car
l'utilisation de foo() l'a rendue  accessible. */
 bar();
```

Fonctions

PHP supporte le passage d'arguments :

- Par valeur : comportement par défaut
- Par référence : pour que la fonction puisse changer la valeur des arguments.

Ex : $toto=2;

```
function mafonctionRef(&$titi) {
$titi=$titi+1;
echo $titi;}
MafonctionRef($toto); // affiche 3
echo($toto) ; // affiche 3;
```

Fonctions

- Une liste variable d'arguments : le mot clé … indique que la fonction accepte un nombre variable des arguments.

  Ex :

```
Function sum(...$vars) {
 $somme = 0;
 foreach ($vars as $i) {
 $somme += $i;     }
 return $somme; }
 echo sum(1, 2, 3, 4);  //  affiche 10 comme résultat.
Echo sum(1,2); // affcihe 3;s
```

Fonctions

- les valeurs par défaut des arguments.

```
function servir_cafe ($type = "cappuccino") {
return "Servir un $type.\n"; }
echo    servir_cafe();  // affiche « servir un  cappuccino
```

→Les arguments sans valeur par défaut doivent être en premiers, sinon erreur

```
function melangeDarguments ($type = "parDefault", $varSansDefault) {
return  "Message  $type $varSansDefault.\n"; }
echo melangeDarguments ( "test");   // affiche une      erreur
```

Fonctions

- A partir de la version 7.0, php supporte la définition du type de retour.

Ex: function maFonction($var) : string {

return $var; }

- Si return est omis, la valeur NULL sera retournée.
- Pour renvoyer plusieurs valeurs en même temps , il suffit de retourner un tableau ;

Ex : function couleurs() {

return array("res1", "res2", "res3") ; }

}

list($var1, $var2, $var3) = couleurs();

Fonctions

- Pour retourner une référence d'une fonction, utilisez l'opérateur & aussi bien dans la déclaration de la fonction que dans l'assignation de la valeur de retour.

Ex : ?php>

```
function & mafonction( &$var ){
    $var=$var+2;
    return $var;
}
$toto= & mafonction( $var2 );
```

?>

→ Les propriétés de l'objet retourné par la fonction mafonction() sont liées à $toto. Il ne s'agit pas d'une copie de $var.

Fonctions

- PHP ne supporte pas la surcharge, la destruction et la redéfinition de fonctions déjà déclarées.
- Il est possible d'appeler des fonctions récursives en PHP. A noter qu'un appel récursif infini est considéré comme une erreur de programmation.

Fonctions utiles en PHP

- **gettype**($var) : retourne le type de la variable.
- **Addslashes**(): Ajoute des antislash devant les caractères spéciaux.

  Ex : $res = addslashes("L'a"); // retourn L\’a.
- **Strstr**(texte, chaineAchercher) : Trouve la première occurrence dans une chaîne.

  Ex : $email = 'name@example.com';

  $domain = strstr($email, '@');

  echo $domain; // Affiche : @example.com

Fonctions utiles en PHP

- **Htmlspecialchars** (chaine, flags) : Convertit les caractères spéciaux en entités HTML. Elle remplace par exemple & (ET commercial) en *& un example de flag est ENT_QUOTES qui Convertit les guillemets doubles et les guillemets simples.*

  Ex : $str = "This is some bold text.";

  echo htmlspecialchars($str); // affiche : This is some <b>bold</ ` b> text.
- **strip_tags** (chaine, allowableTags): Supprime les balises HTML et PHP d'une chaîne. Les commentaires HTML et PHP sont également supprimés. Ce comportement ne peut être modifié avec le paramètre allowableTags .

  Ex :

  $text = ' Test paragraph. <!-- Comment --> Other text';

  // Autorise et <a>

  echo strip_tags($text, ' <a>'); // affiche : * Test paragraph. Other text*

Fonctions utiles en PHP

- **strlen**() : Retourne la longueur de la chaîne
- **Trim**() : efface les espaces blancs au début et à la fin d'une chaine.
- **Strtolower**() : passe tous les caractères en miniscules.
- **Strtoupper**() : passe tous les caractères en majuscules.
- **Strpos**(texte,chaine) : recherche la position de la première « chaine » trouvée.
- **Ereg**(chaine, texte) : recherche dans texte , les séquences des caractères « chaine », Retourne la longueur de l'occurrence trouvée si une occurrence a été trouvée dans la chaîne string et FALSE dans le cas contraire ou si une erreur est survenue.

  Ex : if(ereg("BCD","ABCDEF")) {echo "oui";} else {echo "non";}

Fonctions utiles en PHP

- **isset**($var1,$var2,…) : Détermine si une variable ou plusieurs sont définies et sont différentes de NULL.

  Ex : if (isset($var)) { echo 'Cette variable existe, donc je peux l\'afficher.';
- **unset**($var1,$var2,…) : Détruit les variables.
- **empty** ($var1) : Détermine si une variable est vide.
- **array_key_exists** (key, array) : Vérifie si une clé existe dans un tableau.

Fonctions utiles en PHP

- **in_array**(valeur, array) : Indique si une valeur appartient à un tableau.
- **array_key**s(array, valeur) : Retourne toutes les clés ou un ensemble des clés contenant « valeur », si valeur n’est pas spécifié alors retourne toutes les clés du tableau.
- **array_values**(array) : Retourne toutes les valeurs d'un tableau.

Exercice 1

- Ecrire un script qui permet d'afficher les nombres premiers parmi 0 et 1000. Utiliser 2 méthodes différentes (While, for).

Exercice 1

Solution

```
<?php
$i=1; while($i<=1000) {
$z=0;
for($j=2;$j<$i;$j++)
{
if(($i%$j)==0 && ($i!=$j))
$z++; }
if($z==0) echo $i.' ';
$i++; }
?>
```

Exercice 2

- Ecrire un script qui permet de créer un tableau contenant les coordonnées d'une personne puis d'afficher ses éléments. Utiliser un tableau numéroté puis un tableau associatif.

Exercice 2

Solution

- <?php
  $coordonnees= array('Camille', 'Dupont', 21, 'Paris');
  for($i=0;$i<count($coordonnees);$i++)
  echo $coordonnees[$i].' ';
  ?>
- Tableau associatif :

```
<?php
  $coordonnees= array(
"prenom" => "Camille",
"nom" => "Dupont",
"age" => 21,
"ville" => "Paris" );
foreach( $coordonnees as $cle =>$value ){
  echo $cle . "= ". $value . "<br/>";
  }
 ?>
```

Exercice 3

- Ecrire une fonction ''moisfr'' qui permet d’afficher un mois en toutes lettres et en français en utilisant les instructions de Switch Case Afficher le mois 4 en faisant appel à la fonction.

Exercice 3

Solution

```
<?php
function moisfr($date)
{
switch($date)
{
case 1 : return 'Janvier'; break; case 2 : return 'Fevrier'; break; case 3 : return 'Mars';
break;
case 4 : return 'Avril'; break;
case 5 : return 'Mai'; break;
case 6 : return 'Juin'; break;
case 7 : return 'Juillet'; break;
case 8 : return 'Aout'; break;
case 9 : return 'Septembre'; break; case 10 : return 'Octobre'; break; case 11 : return
'Novembre'; break; case 12 : return 'Décembre'; break; default : return 'mois non
valide';
}}
echo moisfr(4);
?>
```

Exercice 4

Ecrire un script php permettant d'afficher le résulat suivant :

Astuce : Utiliser un tableau associatif et la foncion gettype();

Test sur les variables

la variable toto est de type entier, sa valeur est 10.

la variable titi est de type String , sa valeur est « chaine».

Exercice 4

Solution :

```
<html>
<body>
<center><h1> Test sur les variables </h1></center>
<?php
  $variables= array(
 "toto" => 10,
 "titi" => "chaine" );
 foreach( $variables as $cle => $value ){
    echo "la variable " .$cle . " est de type ". gettype( $value ). ",
sa valeur est " . $value . "<br/>";
   }
  ?>
</body>
</html>
```

Formulaires

La définition des formulaires se fait avec la balise **<FORM>** en utilisant deux attributs suivant :

- **METHOD** : pour définir la méthode de transfert des données vers le serveur ; deux valeurs possible «POST» ou «GET» (les données du formulaire seront encodées dans l'URL).
- **ACTION** : pour préciser l'action à exécuter lors du l'envoie des données.

Exemple d'utilisation :

<FORM METHOD="POST" ACTION= "*cible.php*" >

….

</FORM>

Formulaires

Les éléments d'un formulaire:

I. **INPUT** : Champs de saisie de texte et différents types de boutons.
II. **TEXTAREA**: zones de texte multi-lignes.
III. **SELECT**: Listes déroulantes.
IV. **FIELDSET**: regrouper des éléments du formulaire

Formulaires

I. **INPUT** : Champs de saisie de texte et différents types de boutons à définir avec l'attribut **TYPE** :

1. **Text**: une zone de texte à une seule ligne

<input type="text" id="toto" name="titi" *{autofocus/ required}* />

- **ID** : est un attribut lu par le navigateur (en HTML, CSS ou Javascript) ;
- **NAME** : sert à PHP lors de la validation d'un formulaire. Coté PHP

→la valeur de la zone du texte est récupérée en PHP par la variable _$POST['titi'], par contre la variable utilisée en css pour donner du style à la zone est #toto{…}.

- **Value :** permet de pré-remplir le champ avec une valeur par défaut.
- **Placeholder :** permet de donner une indication sur le contenu du champ. Cette indication disparaîtra dès que le visiteur aura cliqué à l'intérieur du champ.

Formulaires

2. **Password:** Zone de mot de passe (caratères saisis ne s'affichent pas à l'écran)

<input type="password" name="pass" id="pass" />

3. **Checkbox** : Les cases à cocher

<input type="checkbox" name="choix" id="choix" value=" choix1 " *checked* />

4. **Radio**: Les zones d'options

<input type= "radio" name="choix" id="choix" value=" choix1 " />

Formulaires

5. Boutons

- **Reset** : remise à zéro du formulaire

```
<input type="reset" value="Effacer">
```

- **Submit** : validation du formulaire

```
<input type="submit" value="Envoyer" />
```

Formulaires

II. TEXTAREA : créer une zone de texte multi-lignes.

<textarea name="ameliorer" id="ameliorer"> </textarea>

Il y a deux façons de modifier la taille de cette zone :

- En CSS : en utilisant les propriétés CSS « width » et « height ».
- Avec des attributs : « **rows** » et « **cols** » de la balise <textarea>. Le premier indique le nombre de lignes de texte qui peuvent être affichées simultanément, et le second le nombre de colonnes.

Formulaires

III. SELECT : faire un choix parmi plusieurs possibilités sous forme de liste.

- L’attribut **« multiple » :** indique qu'on peut sélectionner plusieurs options parmi celles offertes dans le contrôle. Par défaut, si cet attribut n'est pas utilisé, seule une option peut être sélectionnée.
- A l’intérieur de **<select> </select>**, placer plusieurs balises **<option> </option>** et/ou grouper les options avec la balise **<optgroup> </ optgroup>**.

Formulaires

- **Exemple avec Select :**

```
<select  id="GID"  multiple >
  <optgroup label="Groupe 1">
    <option>Option 1.1</option>
  </optgroup>
  <optgroup label="Groupe 2">
    <option>Option 2.1</option>
    <option>Option 2.2</option>
  </optgroup>
  <optgroup label="Groupe 3" disabled>
    <option>Option 3.1</option>
    <option>Option 3.2</option>
    <option>Option 3.3</option>
  </optgroup>
</select>
```

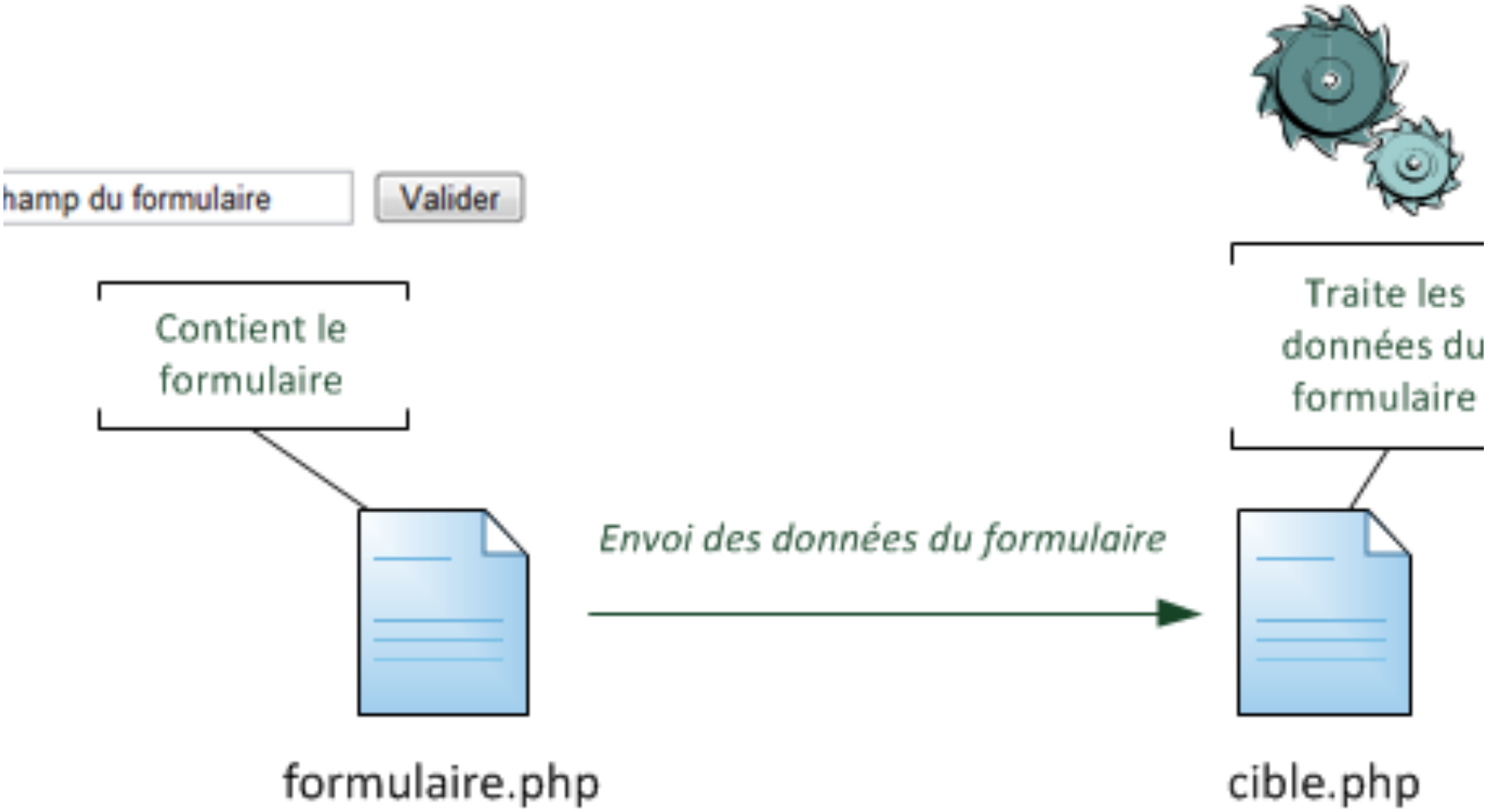
Formulaires

IV. FIELDSET : regrouper des éléments du formulaire qui ont un rapport entre eux.

- L'attribut « **Legend** » permet de donner un titre à ce groupement.

```
<fieldset>
    <legend>Vos coordonnées</legend>
    <label for="nom">Quel est votre nom ?</label>
    <input type="text" name="nom" id="nom" />
    <label for="prenom">Quel est votre prénom ?</label>
   <input type="text" name="prenom" id="prenom" />
</fieldset>
```

Traitement des Formulaires



Traitement des Formulaires

Selon la méthode utilisée dans le formulaire (spécifiée dans l'attribut « method ») :

- **$_GET** est une variable super-globale pour récupérer les données du formulaire avec la méthode GET.
- **$_POST** est une variable super-globale pour récupérer les données du formulaire avec la méthode POST.
- Un élément d'un formulaire est identifié en php par son attribut « name ».

Traitement des Formulaires

Ex 1 :

```
<form method="post" action="cible.php">
<input type="text" id="toto" name="titi"   />
<input type="submit" value="Envoyer" />
</form>
```

En php, pour récupérer la valeur saisie dans la zone texte :

```
 <p>Je sais comment tu t'appelles, hé hé. Tu t'appelles <?php echo $_POST[’titi']; ?> !</p>
```

Traitement des Formulaires

Ex 2 :

```
<form method='GET' action= "cible.php">
Qu'est-ce que vous aimer ?<br />
  <input type="checkbox" name="loisirs[]" value="shopping" /
>  Faire du shopping<br />
 <input type="checkbox" name="loisirs[]" value="sport" />
Faire du sport<br />
  <input type="submit" value="Ma réponse"/> </form>
```

En php, pour récupérer les valeurs cochées :

```
 $loisirs = $_GET["loisirs"];
 echo "<b>Vous aimez </b><br />";
 for ($i=0; $i<count($loisirs); $i++) {
  echo $loisirs[$i]."<br />";     }
```

Exercices

- Exercice 1 : Transmettre les valeurs saisies des variables Nom , Prénom et Age d'un formulaire d'un fichier (source.php) à une autre page (Cible.php) qui permet de les afficher.

Exercices

Exercice 1 : Solution

source.php

```
<form method=“post” action=”cible.php”>
Prenom : <input type=”text” name= “prenom” ><br/> Nom : <input type=”text” name= “nom” ><br/>
Age : <input type=”text” name= “age” >
<input type=”submit” value= “envoyer” >
```

cible.php

```
<?php
$prenom = $_POST[“prenom”] ;
$nom = $_POST[“nom”] ;
$age=$_POST[“age”] ;
echo “Bonjour <b>$prenom $nom</b>, votre age est : $age <br>”; ?>
```

Exercices

Exercice 2 : Transmettre à travers un lien, les valeurs des variables Nom et Age d'un fichier (source.php) à une autre page (Cible.php) qui permet de les afficher.

(sans utiliser les formulaires

Utiliser plutôt un lien hypertexte ( )

Exercices

Exercice 2 : solution

Source.php :

```
<html><body>
<?php
$nom="george" ;
$age=21;
?>
<a href='pagecibleTransValssForm.php?nom=<?php echo $nom ?>&age=<?php echo $age ?>' >
Lien vers cible </a>
</body></html>
```

Cible.php :

```
<html><body>
<?php
nom=$_GET['nom'] ;
$age=$_GET['age'] ;
echo 'Bonjour '. $nom .', votre age est : '. $age .'<br>';
?>
</body> </html>
```

Exercices

Exercice 3 : Ecrire un script qui permet de protéger le contenu d'une page secrète avec un login et un mot de passe en utilisant un formulaire. Tout le script doit être écrit dans la même page.

Exercices

Exercice 3 : Solution

```
<?php
$log="login";
$pass="motDepass";
if(isset($_POST['login']) and isset($_POST['passw'])) :
$login=$_POST['login'];
$passw=$_POST['passw'] ;
Endif;
if($login==$log && $passw==$pass): ?>
<html><body>
**************************CONTENU CONFIDENTIEL**********************<br/>
<b> Code secret : XXXXXYYEYEYEYE</b>
</body></html>
<?php
else : ?>
<html><body>
<form method="post">
Login : <input type="text" name="login" /><br>
Mot de passe : <input type="password" name="passw" ><br/>
<input type="submit" value= "Envoyer" />
</body> </html>
<?php endif; ?>
```

Visibilité des variables

- En PHP il existe 3 niveaux de visibilité d'une variables selon le contexte :
- **Les variables superglobales** : sont disponibles n'importe où dans le programme.
- **Les variables globales** : ce sont toutes les variables, tableaux, objets et constantes que nous créons nous-même dans le programme principal. Ils sont généralement visible que dans le programme principal
- **Les variables locales** : ce sont toutes les variables d'une fonction (paramètres compris). Leur visibilité n'est que locale, et le programme principal ne peut pas agir sur ces variables.

Portée des variables globales

- En PHP, une variable globale peut être utilisée à l'intérieur d'une fonction sans la passer en paramètre de celle-ci. Cela est possible en utilisant le mot clé « global ». Cependant cette variable doit être déclarée à l'intérieur de chaque fonction afin de pouvoir être utilisée dans celle-ci.

Exemple :

```
<?php
$a = 1;
$b = 2;
function somme() {
 global $a, $b;
$b = $a + $b;
 echo $b ;
//  dans cet exemple, on a déclarer $a et $b avec le mot clé « global »  à l'intérieur de somme pour qu'elles seront manipuler par celle-ci.
}
```

- Une deuxième méthode pour accéder aux variables globales est d'utiliser la variable superglobale $GLOBALS :

Exemple :

```
<?php
$a = 1;
$b = 2;
function somme() {
  $GLOBALS['b'] = $GLOBALS['a'] + $GLOBALS['b']; }
   echo $b;}
?>
```

Les variables superglobales

- Les variables superglobales sont générées automatiquement par PHP :
  - elles sont disponibles quel que soit le contexte du script même à l'intérieur d'une fonction sans être passées en paramètre de celle-ci.
  - elles sont écrites en majuscules et commencent, par un underscore (_).
  - Elles sont généralement des tableaux associatifs.

Les variables superglobales

Notons :

- $GLOBALS : rassemble les variables globales.
- $_ENV : ce sont des variables d'environnement toujours données par le serveur.
- $_SESSION : se sont des variables de session. Ces variables restent stockées sur le serveur le temps de la présence d'un visiteur.
- $_COOKIE : contient les valeurs des cookies enregistrés sur l'ordinateur du visiteur.
- $_GET : contient les données envoyées en paramètres dans l'URL.
- $_POST : contient les informations qui viennent d'être envoyées par un formulaire.

Les variables superglobales

Exemple 1 :

```
<html>
<head>
        <title>La variable $_SERVER</title>
</head>
<body>

<?php
echo 'Nom du fichier  en cours d'exécution à partir de la racine:'.$_SERVER['PHP_SELF'];
echo '<br/>' ;
echo 'Nom de la racine du script:' .$_SERVER['DOCUMENT_ROOT'] ;
echo '<br/>';
echo 'Nom du client HTML:'. $_SERVER['HTTP_USER_AGENT'].'<br>';
echo 'Nom du serveur qui exécute le script:'. $_SERVER['SERVER_NAME'] ;
echo '<br/><br/>';
echo 'Votre adresse IP est : '. $_SERVER['REMOTE_ADDR'] ;
?>
</body>
 </html>
```

Inclusion des fichiers

L’inclusion des fichiers PHP sert à réutiliser une partie de code identique dans plusieurs endroits dans une même page ou dans des pages différentes.

L’inclusion des fichiers se fait avec include() et require() :

1. include() :

Cette fonction renvoie une erreur de type WARNING, de ce fait si elle n’arrive pas à ouvrir le fichier, le code qui suit sera exécuté.

2. require()

Cette fonction est identique à include sauf que lorsqu'une erreur survient, une erreur de type FATAL stoppera l’exécution du script.

Inclusion des fichiers

Lorsqu'un fichier est importé, le code se trouvant à l'intérieur est exécuté. Ainsi, les variables, les constantes du fichier importé peuvent être réutilisés dans la suite du programme.

Exemple :

fichier1 .php :

```
<?php
 // Définition des variables
 $a = 2;
 $b = 4;
 // Affichage d'un texte
 echo 'Un peu de mathématiques...';
?>
```

fichier2 réutilise le code du fichier1 :

```
?php
 // Importation et exécution du fichier1
 require('fichier1.php');
 // Calcul du produit
$produit = $a * $b;
 // Affichage de la somme
 echo 'Produit de $a  et $b = ', $produit;  // affichera 8
?>
```

Redirection vers une page

- En PHP, La fonction header() se charge d'envoyer au serveur les entêtes passés en paramètre.
- Pour créer une redirection avec PHP, on utilise cette fonction pour envoyer des entêtes de type Location (adresse).
- Règle importante : l'appel de ‘header()’ doit se faire avant tout envoi au navigateur

Redirection vers une page

Script de redirection :

```
<?php   header('Location: http://
www.votresite.com/pageprotegee.php');
?>
```

Variables de session

- Les variables de session servent à stocker temporairement des informations sur l'utilisateur dans le serveur.
- Elles sont présentes dans toutes les page de votre site pendant la présence d'un visiteur.
- Elles sont accessible via la superglobale $_SESSION

variables de session

Gestion des sessions

- un visiteur arrive sur un site. une session est créée pour lui. PHP génère un identifiant unique pour lui, appelé ‘session ID’.
- une fois la session est démarrée automatiquement ou avec session_start(), une infinité de variables de session peuvent être créées selon le besoin.
- La session sera fermée par session_destroy() si le visiteur ne charge plus de page dans le site pendant quelques minutes ou si le visiteur souhaite se déconnecter.

Variables de session

Pour que les variables de sessions soient accessibles dans toutes les pages d'un site web, il faut appeler session_start() sur chacune de ces pages avant décrire le moindre code HTML (avant même <! doctype>.

Variables de session

- Exemple :

```
<?php  // démarer une session
        session_start();
        echo 'Bienvenue à la page numéro 1’;
       // On s'amuse à créer quelques variables de session dans $_SESSION
      $_SESSION['prenom'] = 'Jean';
      $_SESSION['motPass'] = 'MotDePasse’;   ?>
<html>
<head><title>Les sessions </title></head>
<body>
<?php
      if(isset($_SESSION['prenom'])) {
      echo "<p> salut je sais qui es-tu, tu t'appelles ".  $_SESSION['prenom'] . "</p>";
      echo "<p> <a href=\"fichierUtilisantVariablesSession.php\">cliquer ici SVP </a></p> »;  }
  else {
      echo "les variables de session ne sont pas déclarées<br/>" ; }  ?>
</body>
 </html>
```

Cookies

- Un cookie est un mécanisme pour stocker des informations sur les visiteurs afin de l'identifier plus tard .
- Contrairement aux variables de session, les cookies sont des petits fichiers stockés sur l'ordinateur de l'utilisateur pour une durée de vie que nous pouvons définir.
- Les cookies sont accessibles via la superglobale $_COOKIE.

Cookies

- la fonction setcookie() permet de définir un cookie qui sera envoyé avec le reste des en-têtes HTTP.

  →Les cookies font partie des en-têtes HTTP, ce qui impose que setcookie() soit appelée avant tout affichage de texte et avant même <!doctype >.

  La fonction setcookie() peut prendre jusqu'à 7 paramètres. Seul le premier est obligatoire car il définit le nom du cookie, le deuxième définit la valeur du cookie et le troisième définit la durée de vie du cookie en seconde.

  Exemple : setcookie('nom', 'nomUtilisateur', time()+3600*24);

  echo $_COOKIE['nom'] ; // affichage du cookie.
- La fonction setrawcookie() est exactement la même que setcookie() excepté que la valeur du cookie ne sera pas automatiquement encodée URL lors de l'envoi au navigateur.
- Pour détruire un cookie : appeler setCookie() avec une date d'expiration dans le passé :
- Ex: setcookie("Nomcookie", "valeurCookie", **time()-3600**);

Cookies

- Exemple de cookie sous forme de tableau :

```
<?php
setcookie("cookie[three]", "cookiethree », time()+3600*1);
setcookie("cookie[two]", "cookietwo », time()+3600*1);
setcookie("cookie[one]", "cookieone », time()+3600*1);
    // l'affichage de ces cookies :
    <?php
    if (isset($_COOKIE['cookie'])) {
     // exemple d'appel du cookie :
     echo 'le cokie two est :'. $_COOKIE['cookie']['two']. ' sinon on peut les parcourir avec foreach comme suit : <br/>';
        foreach ($_COOKIE['cookie'] as $name => $value) {
            $name = htmlspecialchars($name);
            $value = htmlspecialchars($value);
            echo "$name : $value <br />";
        }
    }
    ?>
```

Exercice

Ecrire un script qui permet de protéger le contenu d'une page secrète avec un login et un mot de passe en utilisant un formulaire.

Note :

1-écrire une page authentification.php contenant un formulaire login/mot de passe. Et un lien hypertexte permettant d'afficher la page secrète si une variable de session 'login' a été déjà créée.

Manipuler sur la même page les données du formulaire comme suit :

2-Diriger l'utilisateur vers pageSecret1.php contenant code secret : ZZZZZZZZZZ si login=='login1' et pass=='pass1'.

Au début il faut vérifier que login==login1 et afficher « bonjour M/Mme valeurdelogin»

Afficher le code secret

Créer par la suite un lien vers page précédente

Un lien de déconnexion permettant de rester sur la même page tout en créant une variable deconnecter=« yes ».

3-Diriger l'utilisateur vers pageSecret2.php contenant code secret : ZZZZZZZZZZ si login=='login2' et pass=='pass2'.

Au début il faut vérifier que login==login2 et afficher « bonjour M/Mme valeurdelogin»

Afficher le code secret

Créer par la suite un lien vers page précédente

Un lien de déconnexion permettant de rester sur la même page tout en créant une variable deconnecter=« yes ».

4-Sinon afficher sur la même page un message « Veuillez entrer un login et mot de passe valide » et tout en laissant le formulaire visible afin de permettre à l'utilisateur de s'authentifier.

Programmation Orienté Objet (POO) « Classe et Objet »

- Une classe permet la définition d'un nouveau type de variable qui rassemble plusieurs attributs.
- Elle permet également la définition de fonctions manipulant ces attributs, en POO, ces fonctions s'appellent des 'méthodes'.
- Un objet est une instance d'une classe. Un objet est alors assimilé à une variable et sa classe au type de cet objet.

Programmation Orienté Objet (POO) « concepts de base »

La POO a deux buts :

- faciliter la réutilisation du code que vous avez déjà écrit grâce à l'héritage.
  - L'héritage permet, à partir d'une classe déjà existante, d'en créer une nouvelle qui reprendra ses caractéristiques et de les adapter aux besoins sans modifier la classe de base.
  - Il est possible alors de redéfinir une méthode dans des classes héritant d'une classe de base sauf si cette méthode a été défini comme *final*. L'appel de la méthode d'un objet est possible sans se soucier de son type intrinsèque : il s'agit du polymorphisme.
    - Le polymorphisme traite de la capacité de l'objet à posséder plusieurs formes.
- l'encapsulation des données et les traitements correspondants.
  - L'encapsulation permet de regrouper un ensemble d'attribut avec un ensemble de méthodes en une classe permettant de les manipuler.

POO en PHP
« syntaxe de base »

Une définition de classe commence par le mot-clé « **class** », suivi du nom de la classe.

A la création (*instanciation*) de l'objet, une méthode portant le même nom que la classe est appelée automatiquement : c'est le **constructeur.** Ce dernier est définit par le mot clé « **__construct** ».

Lors de la définition des méthodes de classes, les attributs non statiques sont appelés par l'operateur d'objet **$this->** nomDePropriété.

Exemple :

```
class Personne {
  // déclaration des attributs
public $nom, $age ;
// déclaration du constructeur
     public  function __construct ($nom, $age){
        $this->nom=$nom; // $this représente l'objet que nous sommes entrain de définir.
         $this->age=$age;
}
// déclaration des méthodes
   public function affiche () {
       echo $this->nom;
       echo   $this->age;    }
}
```

POO en PHP

« Visibilité d'un attribut ou d'une méthode »

- « public » : si un attribut ou une méthode est public, alors on pourra y avoir accès depuis n'importe où.
- « private »: impose quelques restrictions. L'accès aux attributs et méthodes est seulement possible depuis l'intérieur de la classe.
  - Ne mettez jamais le constructeur avec le type de visibilité private sinon, elle ne pourra jamais être appelé, vous ne pourrez donc pas instancier votre classe
- « protected » a le même effet que *private, à l'exception que toutes les classes filles auront accès aux attributs protégés.*

POO en PHP
«mot clé static »

Le mot clé « **static** » est utilisé après le type de visibilité pour déclarer une méthode ou attribut statique.

Les attributs et les méthodes statiques peuvent être utilisés sans avoir besoin d'instancier la classe, On peut y accéder directement en utilisant le nom de la classe.

Les attributs statiques sont dites ‘attributs de classe’. Ainsi, tous les objets auront accès à ces attributs et auront les mêmes valeurs pour tous les objets.

Les méthodes statiques d’une classe servent à manipuler les attributs statiques. Ainsi, les méthodes statiques ne peuvent pas manipuler les attributs non statiques de la classe.

POO en PHP
« l'opérateur :: »

L'opérateur de résolution de portée « **::** » (double deux point) fournit un moyen d'accéder aux membres statiques ou constantes, ainsi qu'aux attributs ou méthodes surchargées d'une classe.

→Le mot clé« **self ::** » est utilisé pour accéder aux méthodes et attributs statiques depuis la classe.

→Le mot clé « **parent::** » est utilisé pour accéder aux propriétés ou aux méthodes surchargés ou constantes d'une classe depuis la classe fille.

→Pour référencez ces éléments en dehors de la classe, utilisez plutôt le nom de la classe « nomClasse*:: »* **.**

POO en PHP
« Héritage »

Une classe peut hériter d'une autre classe en utilisant le mot-clé « *extends »* dans la déclaration.

Pour accéder aux méthodes ou une propriétés statiques de la classe mère, l'opérateur «**parent::**» est utilisé.

Exemple :

```
class ExtendClass extends SimpleClass {
  // Redéfinition de la méthode parente et/ou l’ajout des attributs
  function afficher()   {
  echo "Classe étendue";
  parent::afficher();   }
 }
$extended = new ExtendClass;  // instanciation
$extended->afficher();   // polymorphisme.
```

POO en PHP
« Abstraction »

Classe abstraite : est une classe qu'on ne peut pas instancier directement.

- Elle est définit par le mot clé « abstract » dans la déclaration.
- Pour exploiter une classe abstraite, il faut créer une classe qui hérite de la classe abstraite.

Une classe abstraite permet de définir des comportements (méthodes) dont l'implémentation (le code dans la méthode) se fait dans les classes filles :

- Si on rend une méthode abstraite en utilisant le mot clé « abstract » avant la visibilité de la méthode, alors toutes les classes filles sont forcés à écrire cette méthode.

POO en PHP
« Abstraction »

Exemple : classe abstraite

```
abstract class NomAbstractClass  {
    // Force les classes filles à définir les deux méthode suivants
    abstract protected function getValue();
    abstract protected function prefixValue($prefix);
    // méthode commune
    public function printOut() {
    print $this->getValue() . "\n";    }
    }
```

POO en PHP
« constante de classe »

Il est possible de définir des valeurs constantes à l'intérieur d'une classe, qui ne seront pas modifiables.

La visibilité par défaut des constantes de classe est *public.*

Exemple :

```
class MyClass {
const CONSTANT = 'valeur constante';      function showConstant() {
 echo  self::CONSTANT . "\n";
// self::CONSTANT='nouvelle valeur' entrainera une erreur
  } }
echo MyClasse:CONSTANT;
```

POO en PHP
«constante de classe »

Attention à ne pas confondre les attributs statiques et constantes de classe :

→un attribut statique peut tout à fait changer de valeur au cours du temps, à la différence d'une constante dont la valeur est fixée.

POO en PHP
« constante de classe »

Cependant, il est possible de déclarer une constante dans la classe de base, et de changer sa valeur dans la classe fille seulement en utilisant la méthode « get_called_class » qui retournera le nom de la de la classe depuis laquelle la constante est appelée.

→ Ainsi, la nouvelle valeur de la constante sera propre à la classe fille et la classe mère garde la valeur initiale de la constante.

POO en PHP
« constante de classe »

Exemple :

```
<?php
 class Personne {
  // déclaration de la constante
   const nomClasse="Personne";
   public  function __construct (){
    echo ‘ ceci est le constructeur’ ;
     /* on ne peut pas changer la valeur de la constante  ici par  :  self::nomClasse=‘test’  */}
    public function affiche () {
     $c=get_called_class();
      echo ’la valeur de la constante nomClasse est : '. $c::nomClasse. '<br/>' ;   } }
class HPErsonne extends Personne {
      const nomClasse="HPErsonne" ;
       public function __construct (){
        parent::__construct( );
         //parent::nomClasse=‘test2’’ retournera encore une erreur }}
$p=new Personne();
 $p->affiche();
$hp=new HPErsonne();
 $hp->affiche();
?>
```

POO en PHP
« mot clé final »

Le mot-clé « final » indique qu'un élément ne peut être changé dans la suite du programme.

- Il s'appliquer aux méthodes d'une classe ainsi que la classe elle-même.
- Les attributs ne peuvent être déclarés comme final. Une méthode indiquée comme final ne peut être redéfinie dans une classe dérivée.
- On peut s'en servir pour forcer le comportement d'une méthode dans les sous-classes.

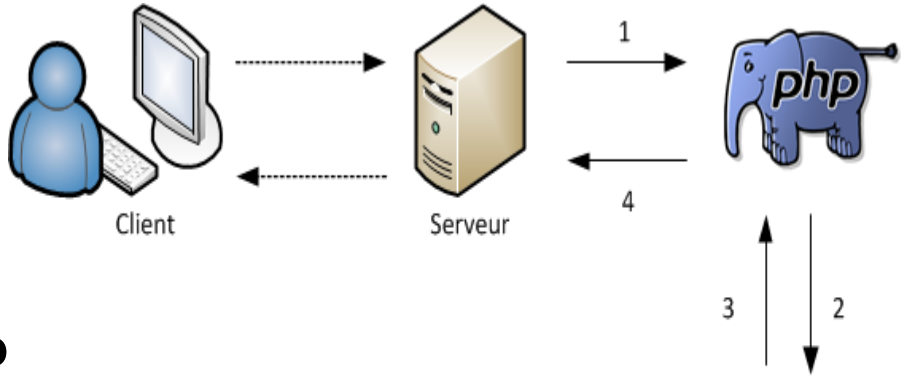
- Les classes final ne peuvent être dérivées (non possibilité d’héritage).

POO en PHP
« mot clé final »

Exemple : classe final

```
final class BaseClass {
   public function test() {
   echo "BaseClass::test() appelée";    }
 // Ici la méthode suivante est finale
final public function moreTesting() {
 echo "BaseClass::moreTesting() appelée";    }
 }
class ChildClass extends BaseClass { }
// Résultat : Fatal error: Class ChildClass may not inherit from
final class (BaseClass)
```

SGBS avec PHP

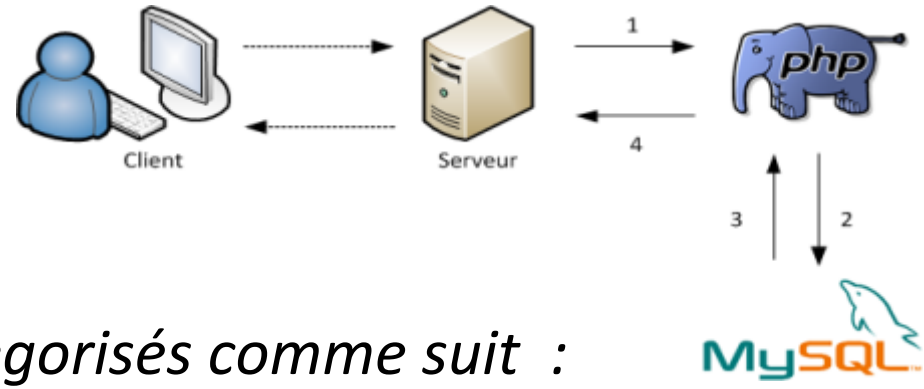


- Qu'est ce qu'un SGBD ??

  →« est un logiciel système destiné à stocker et à partager des informations dans une base de données, en garantissant la qualité, la pérennité et la confidentialité des informations, tout en cachant la complexité des opérations.» WIKIPEDIA

→*Parmi les logiciels les plus connus, on trouve : MySQL, PostgreSQL, SQLite, Oracle et MAriaDB. Pour une liste détaillée des SGBD, consulter ce lien http://fadace.developpez.com/sgbdcmp/#LI*

SGBS avec PHP



Ces systèmes peuvent être catégorisés comme suit :

SGDB Relationnel

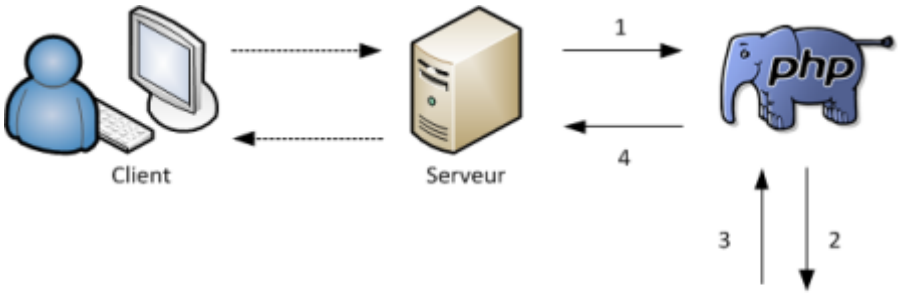
Système basé sur le modèle relationnel : données stockées dans des tables structurées (en colonnes, lignes) et des relations qui lient les tables entre elles.

- *Système libre **:** MySQL, PostgreSQL, MariaDB, Firebird, Ingres, HSQLDB, Derby.*
- *Système propriétaire : Oracle Database, Microsoft SQL Server, DB2, MaxDB, 4D, dBase, Informix, Sybase.*

***SGBD Orienté objet** : ZODB, db4o Embarqué : SQLite, Berkeley DB.*

***SGBD NoSQL** : Cassandra, Redis, MongoDB, SimpleDB, BigTable, CouchDB, HBase, LevelDB, RethinkDB, Memcached.*

SGBS avec PHP

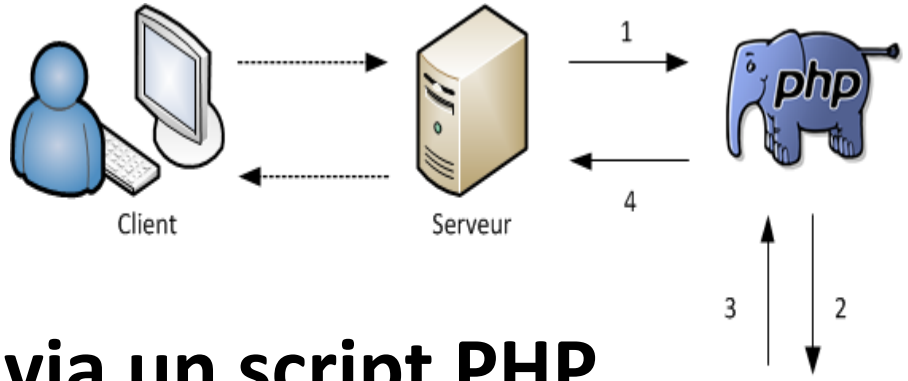


Manipulation des bases de données mysql se fait soit via interface graphique comme 'PhpMyAdmin' soit en exécutant un script php.

1. PhpMyadmin

- Un programme permettant d'avoir une vue rapide de l'ensemble des données.
- C'est un des outils les plus connus permettant de manipuler une base de données MySQL.
- PhpMyAdmin est livré avec WAMP, et presque tous les hébergeurs permettent d'utiliser phpMyAdmin. (chemin d'accès: ***http://localhost:8888/phpMyAdmin***)

SGBS avec PHP



2. Utilisation d’un SGBD via un script PHP

la manipulation d’une base de données s'effectue en 5 temps :

- Connexion au serveur de données
- Sélection de la base de données
- Requête
- Exploitation des requêtes
- Fermeture de la connexion

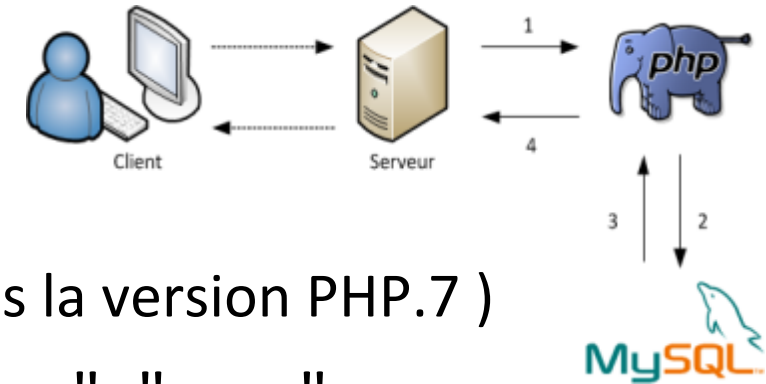
MYSQL avec PHP

Exemple d'un SGBD MYSQL: PHP offre 3 API pour se connecter à MySQL :

1.Mysqli

```
// connexion à la base de données
$link = mysqli_connect("localhost", "root", "passwords",  "dbName");
if (mysqli_connect_errno()) {/* Vérification de la connexion */
 printf("Échec de la connexion : %s\n", mysqli_connect_error());
exit();}
$query = "SELECT Name FROM tablename ORDER by ID DESC";
 if ($result = mysqli_query($link, $query)) {  /* exécution de la requête */
while ($row = mysqli_fetch_assoc($result)) {/* Récupère un tableau associatif */
       printf ("%s <br/>", $row['Name']);  }
mysqli_free_result($result); /* Libère le jeu de résultats */  }
/ mysqli_close($link); /* Fermeture de la connexion * ?>
```

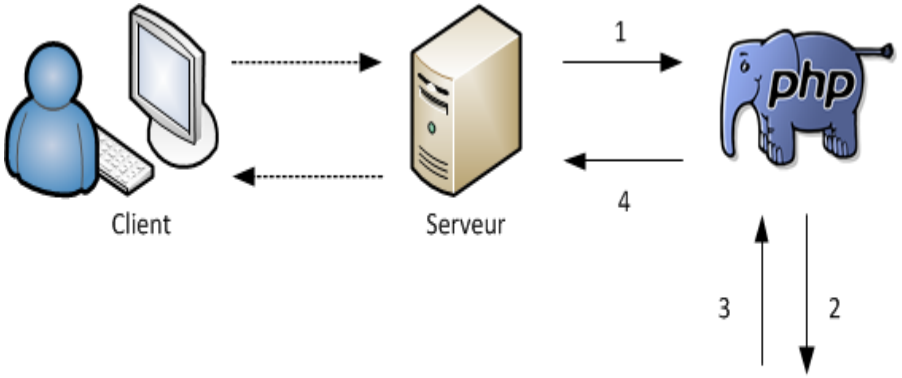
MYSQL avec PHP



2. Mysql (obsolète et supprimé depuis la version PHP.7)

```
 $c = mysql_connect("example.com", "user", "password");
mysql_select_db("database");
 $result = mysql_query("SELECT nom from tableName");
 $row = mysql_fetch_assoc($result);
echo htmlentities($row[’nom']); ?>
```

MYSQL avec PHP



3. **PDO** (OO)

```
$pdo = new PDO('mysql:host=example.com;dbname=database', 'user', 'password’)
$statement = $pdo->query("SELECT nom from tableName");
$row = $statement->fetch(PDO::FETCH_ASSOC);
 echo htmlentities($row[‘nom']);
```

RQ : PHP offre une possibilité d’exploiter l’API mysqli façon orienté objet (OO) :

```
$mysqli = new mysqli("localhost", "root",  "password", "world");
 $result = $mysqli->query("SELECT Name from tablename");
while($row = $result->fetch_assoc()){
 echo htmlentities($row['Name']). "<br/>";}
```

MYSQL avec PHP POO

Manipulation des résultats d'une requête sous forme d'objets :

Exemple 1 :

```
<?php
$mysqli = new mysqli("localhost", « root", « root", "world");
/* Vérification de la connexion */
if (mysqli_connect_errno()) {
   printf("Échec de la connexion : %s\n", mysqli_connect_error());
   exit(); }
$query = "SELECT ID, Name, CountryCode FROM citoyen ORDER by ID DESC LIMIT 50,5";
if ($result = $mysqli->query($query)) {
   /* Récupère un tableau d'objets */
   while ($obj = $result->fetch_object()) {
      printf ("%s (%s)\n", $obj->Name, $obj->CountryCode);  }
   /* free result set */
   $result->close();
} /* Fermeture de la connexion */
$mysqli->close();
?>
```

NOTEZ que fetch_object( ) affecte les attributs de l'objet avant d'en appeler le constructeur. Il faut alors vérifier si des valeurs ne sont pas assignées aux attributs avant de leurs affecter des nouvelles valeurs dans la déclaration du constructeur.

MYSQL avec PHP POO

Manipulation des résultats d'une requête sous forme d'objets : avec la déclaration de la class

Exemple 2 :

// Définition de la classe :

```
class Personne {
  // déclaration des attributs
Private $id, $nom, $age ;
// déclaration du constructeur
     public  function __construct ($id=0){
      if( !$this->id)  $this->id=$id;
}
public function __ToString () {
Return 'nom: '. $this->nom. '  age : '. $this->age. '<br/>;    }
}
```

// connexion à la base de données

….

// affichage du résultats :

```
while ($obj = $result->fetch_object('Personne')) {
      echo $obj ;}
```

Traquer les erreurs avec PDO try-catch(PDOException)

Pour identifier la source des erreurs lors de l'exécution du script PHP, Ajouter le 6ème paramètre dans PDO comme suit :

<?php $bdd = new PDO ('mysql:host=localhost;dbname=world;charset=utf8', 'root', '', array(PDO::ATTR_ERRMODE => PDO::ERRMODE_EXCEPTION)); ?>

→Désormais, toutes vos requêtes SQL qui comportent des erreurs vont avoir un message beaucoup plus clair.

Exercices PHP/MYSQL

Exercice 1 : Ecrire un script PHP pour afficher les lignes de la table citoyen en affichant la date au format jj-mm-yyyy.

Exercice 2: REFAIRE l'exercice du code secret en utilisant une base de données.

-login1/pass1 va devoir lire la page secret1.php

-login2./pass2 va devoir lire la page secret2.php.

Transaction : Modèle ACID

Transaction est un ensemble de traitements (requêtes) exécutés en tous ou rien.

ACID = Atomicité, Cohérence, Isolation et Durabilité.

1. Atomicité assure qu'une transaction se fait au complet ou pas du tout.

→Cette règle doit être respectée dans toute les situations, notamment lors d'une panne d'électricité, une défaillance de l'ordinateur

Début de transaction : **START TRANSACTION**

Requêtes : SELECT, UPDATE ..

Validation de la transaction : **COMMIT**

OU Annulation de la transaction en cas d'erreur : **ROLLBACK.**

RQ : S'il y a une erreur d'intégrité de données , le roolback se fait automatiquement

Transaction : Modèle ACID

2. Cohérence assure que chaque transaction amènera le système d'un état valide (t) à un autre état valide t+1 (t : avant l'exécution et t+1 après l'exécution).

Transaction : Modèle ACID

3. Isolation : aucune dépendance entre les transactions.

Exemple : utilisation de la notion de '**verrou**' pour bloquer en lecture et /ou en écriture l'accès à une base de données.

4. Durabilité : assure que lorsqu'une transaction a été confirmée, elle demeure enregistrée même à la suite d'une panne d'électricité, d'une panne de l'ordinateur ou d'un autre problème ...

Transactions avec MYSQL

MYSQL a la particularité de gérer plusieurs moteurs de stockage dans une même BD, aussi appelé moteur de table.

- Un moteur de stockage est un ensemble d'algorithmes permettant de stocker et d'accéder aux données dans un SGBD. En principe , un seul moteur est utilisé par un SGBD.
- Pour lister tous les moteurs , utilisez la requête :

  SHOW ENGINES;

Transactions avec MYSQL

- Principaux moteurs de stockage MYSQL :

  MyISAM : très populaire

  ✓ Très simple d'utilisation.

  ✓ très performant sur des tables fréquemment ouvertes (très rapide pour les opération count() et en lecture)

  ✓ Offre un index FULL-TEXT qui permet de faire des recherche précise sur textes.

  × Ne supporte ni clés étrangères, ni les transactions

  × Gère le verrouillage au niveau de la table (bloque la table entière lors des opérations d'insertions, suppressions ou MAJ).

Transactions avec MYSQL

- Principaux moteurs de stockage MYSQL :

  Memory : Stocke les données de la table en mémoire (RAM).

  ✓ Rapidité d'accès

  × En cas de panne, les données stockées sont supprimées.

  InnoDB : souvent utilisé dans les secteurs sensibles.

  ✓ Gestion des clés étrangères et support des transactions (chaque requête est considérée comme une transaction).

  ✓ Gère le verrouillage au niveau de la ligne.

  × Ne propose pas d'index FULL-TEXT, légèrement plus lent dans les opérations.

Transactions avec MYSQL

- Exemple de gestion de transaction avec Mysql (achat en ligne)

1. Pour supporter les transaction, il faut spécifier le moteur InnoDB lors de la création de la table

CREATE TABLE Compte (ID int , Solde int not null) ENGINE=InnoDB ;
SET autocommit=0; // désactiver la validation automatique des requêtes.
START TRANSACTION;
UPDATE compte SET solde=solde+100 where id=IDVendeur;
UPDATE compte SET solde=solde-100 where id=IDAcheteur;
COMMIT;

Transactions avec PDO

Exemple :

- Début de la transaction :

```
$bdd->beginTransaction();
```

- Specification de Transaction :

```
$bdd->query();
$bdd->query();
..
```

Validation / annulation de la transaction avec :

```
$bdd->commit();
$bdd->rollback();
```

Exercice

Écrire un script php qui va compter le nombre de visiteurs sur votre site.

Remarque :

1. Avant d'écrire le script, pensez à choisir le moteur de stockage adéquat.

2. Utiliser le driver PDO.

Références

Manuel PHP :

- http://php.net

Cours en ligne :

- https://openclassrooms.com/courses/programmez-en-oriente-objet-en-php
- http://caron.ws/data/livre/PHP_v1.pdf