Université de Jijel
2ème Année Licence Informatique

Cours Développement d'application Web

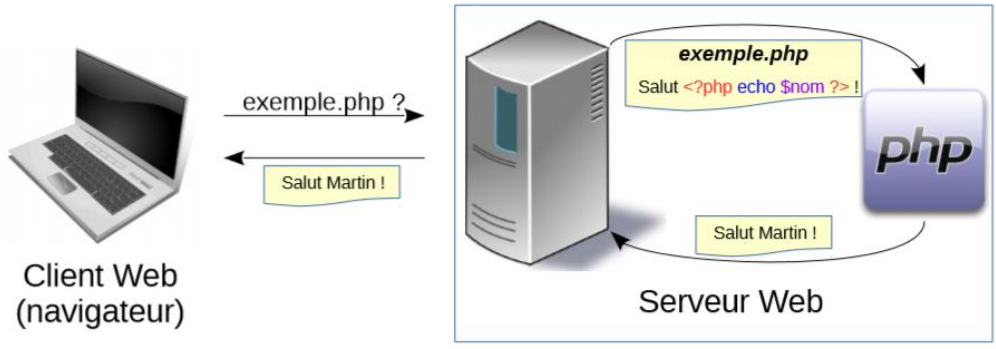
Programmation Web coté serveur :

Le langage PHP

Introduction (1/3)
Description

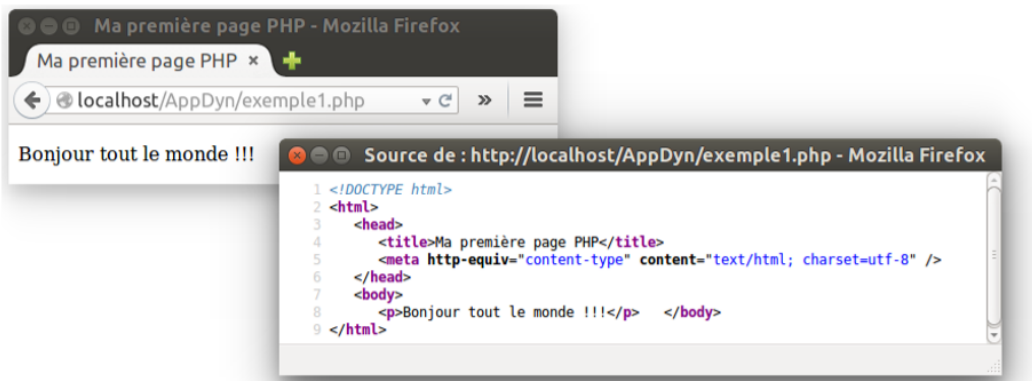
- PHP Hypertext Preprocessor
  - Langage de script intégré à HTML/XHTML, côté **serveur**
  - Le **serveur** parse les documents et interprète le code PHP
  - Le **client** reçoit uniquement le **résultat** du script (une page "générée")
  - Le code PHP est inclus dans des balises PHP : `<?php code-PHP ?>`
  - Voir : `http://www.php.net/`

Introduction (2/3)
Fonctionnement



Introduction (3/3)
Premier script PHP

```
<!DOCTYPE html>
<html>
  <head>
    <title>Page PHP</title>
    <meta http-equiv="content-type" content="text/html;charset=utf-8" />
  </head>
  <body>
    <?php echo "<p>Bonjour le monde !!</p>\n"; ?>
  </body>
</html>
```



Syntaxe (1/18)
Syntaxe de base

Insertion d'une commande PHP :

```
<?php code PHP ?>
```

Séparateur d'instructions : le point virgule " ;"

```
<?php instruction1; instruction2; ... ?>
```

Commentaires : syntaxe à la C, C++ ou Shell

```
/* ... */
// ...
# ...
```

Syntaxe (2/18)
Les variables

Le typage des variables est dynamique

Syntaxe : `$NomDeVariable[=val];`

- règle de nommage : `$[a-zA-Z\_]([a-zA-Z0-9\_])*`
- sensibilité à la casse
- assignation par :
  - valeur : `$var1=$var2;`
  - référence : `$var1=&$var2;`
- Portée : locale à la fonction où elle est déclarée

Exemple :

```
<?php
  $var1=10;
  $var2=&$var1;
  $var1=20;
  echo "<p>".$var1." ".
      $var2."</p>";
?>
```

Affichera :
20 20

Syntaxe (3/18)
Variables globales

Déclaration de variable globale : `global $var;`

global.php

```php
<!DOCTYPE html>
<html>
  <head>
    <title>Page PHP</title>
    <meta http-equiv="content-type" content="text/html;charset=utf-8" />
  </head>
  <body>
    <?php
      function affiche() {
        global $var1;
        echo "<p>Hello ".$var1." !!!</p>\n";
      }
    ?>
    <?php
        $var1="John";
        affiche();
    ?>
  </body>
</html>
```

Syntaxe (4/18)
Variables superglobales

Les variables superglobales sont utilisables sans le mot clef `global`

Quelques tableaux superglobaux :

- `$_GLOBAL` contient des références sur les variables de l'environnement d'exécution global (clefs = noms des variables globales)
- `$_SERVER` variables fournies par le serveur web
- `$_GET` et `$_POST` variables transmises par les méthodes `GET` et `POST` du protocole HTTP
- `$_COOKIE`, `$_REQUEST`, `$_SESSION`, `$_FILES`, `$_ENV`

Syntaxe (5/18)
Les constantes

Syntaxe : `define("NOM_DE_LA_CONSTANTE", valeur)`

Les constantes :

- ne commencent pas par $
- sont définies et accessibles globalement dans tout le code
- ne peuvent pas être redéfinies
- ne peuvent contenir que des booléens, des entiers, des flottants et des chaînes de caractères

Exemple

```
define("PHP","PHP Hypertext Preprocessor");
echo PHP;
```

Syntaxe (6/18)
Les types

4 types simples :

- entiers : `integer`
- réels : `float`, `double`
- booléens : `boolean` (`TRUE` ou `FALSE`)
- chaînes de caractères : `string`

2 types complexes :

- tableaux : `array`
- objets : `object`

2 types spéciaux :

- ressources : `resource` (ex : connexion BD)
- absence de valeur : `null`

Syntaxe (7/18)
Les tableaux

Principe : associations ordonnées de type *clef* ⇒ *valeur*

Déclaration : `array( [clef =>] valeur, ...)`

- la clef est facultative, elle est de type entier ou chaîne de caractères ; *en cas d'omission, la valeur sera associée à la clef d'indice max+1*
- la valeur peut être de n'importe quel type

fruits.php

```php
$tab=array("fruit"=>"pomme",42,"légume"=>"salade",1.5e3);
foreach($tab as $cle => $valeur) {
  echo "<p>".$cle."=>".$valeur."</p>";
}
echo "<p>tab[1]=".$tab[1]."</p>";
$tab[]="peu importe";
echo "<p>tab[2]=".$tab[2]."</p>";
echo "<p>tab['fruit']=".$tab["fruit"]."</p>";
```

Syntaxe (8/18)
Opérations sur les tableaux

- `count($array)` : nombre d'éléments
- `sort($array)` : trie le tableau

- `array_pop($array)` : récupère et supprime le dernier élément d'une liste (i.e. fonctionne comme une pile)
- `array_push($array, $elem1, ...)` : ajoute des éléments à la fin d'une liste (i.e. fonctionne comme une pile)
- `array_shift($array)` : récupère et supprime le premier élément d'une liste
- `array_unshift($array, $elem1, ...)` : ajout d'éléments en début de liste

- `array_merge($array1, $array2, ...)` : fusionne plusieurs tableaux
- `in_array($elem, $array)` : recherche d'un élément dans un tableau
- `array_key_exists($key, $array)` : recherche une clef dans un tableau
- `array_flip($array)` : inverse les clef et les valeurs d'un tableau

Syntaxe (9/18)
Détermination du type d'une variable

Type d'une variable : `string gettype($var);`

Test : `is_integer($var); is_double($var); is_array($var); ...`

Conversion dynamique : `$result = (type-désiré)$var;`

Instructions de vérification d'existence (formulaires) :

- `boolean isset($var);` retourne `FALSE` si `$var` n'est pas initialisée ou a la valeur `NULL`, `TRUE` sinon ;
- `boolean empty($var);` retourne `TRUE` si `$var` n'est pas initialisée, a la valeur `0`, `"0"`, ou `NULL`, `FALSE` sinon ;
- `boolean is_null($var);` retourne `TRUE` si `$var` n'est pas initialisée ou vaut `NULL`, `FALSE` sinon.

Syntaxe (11/18)
Opérateurs

Opérateurs identiques à ceux du C/C++/Java :

- opérateurs arithmétiques : `+ - * / %`
- in/décrémentation : `var++ var-- ++var --var`
- opérateurs logiques : `&& || !`
- comparaisons : `== != <= >= <>`
- concaténation de chaînes de caractères : `.`
- affectation : `= += -= *= ...`

Opérateurs spécifiques :

- `` `commande shell` `` (*ex* : `$a=`ls -ul``)
- `===` : teste la valeur et le type

Syntaxe (12/18)
Instructions de branchement

Proches du C/C++/Java :

Si-sinon-alors :

```
if(condition) {
  instructions
}
[elseif(condition) {
  instructions
}]
[else {
  instructions
}]
```

Switch-case :

```
switch(expression) {
  case 'valeur1':
    Instructions
    break;
  ...
  default:
    Instructions
    break;
}
```

Syntaxe (13/18)
Boucles

Proches du C/C++/Java :

Boucles for :

```
for($i=0; $i<N; $i++) {
  Instructions
}

foreach($tab as $val) {
  Instructions
}

foreach($tab as $cle=>$val) {
  Instructions
}
```

Boucles while :

```
while(condition) {
  Instructions
}

do {
  Instructions
} while(condition);
```

Syntaxe (14/18)
Répétition de code HTML

- Utilisation des boucles pour répéter du code HTML : Entrelacement code PHP / code HTML

repetitionHTML.php

```
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="content-type" content="text/html;charset=utf-8" />
    <title>Page PHP</title>
  </head>
  <body>
    <?php
      $tab=array("premier","second","troisième","...");
    ?>
    <ul>
      <?php foreach($tab as $elem) { ?>
        <li><?php echo $elem; ?></li>
      <?php } ?>
    </ul>
  </body>
</html>
```

Syntaxe (14/18)
Répétition de code HTML

- Utilisation des boucles pour répéter du code HTML : Entrelacement code PHP / code HTML

repetitionHTML.php

```
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="content-type" content="text/html;charset=utf-8" />
    <title>Page PHP</title>
  </head>
  <body>
    <?php
      $tab=array("premier","second","troisième","...");
    ?>
    <ul>
      <?php foreach($tab as $elem) { ?>
        <li><?php echo $elem; ?></li>
      <?php } ?>
    </ul>
  </body>
</html>
```

Syntaxe (16/18)
Conventions de nommage des fichiers

- Bonne pratique : .class.php
- Bonne pratique : .conf.php
- Bonne pratique : .inc.php
- Mauvaise pratique (par exemple) : `.inc` pour les librairies

Remarque

Les fichiers dont l'extension n'est pas `.php` ne sont pas parsés, et donc directement lisible par une requête HTTP

Syntaxe (17/18)
Les fonctions

Syntaxe

```
function nomDeFonction(arg1, arg2, ..., argN) {
  instructions
}
```

- passage d'arguments par valeur et référence supporté (&)
- les fonctions supportent un nombre variable d'arguments
- la surcharge et la redéfinition ne sont pas supportées
- les arguments supportent les valeurs par défaut
- retour d'une unique valeur par la directive `return`
- les noms de fonctions sont insensibles à la casse
- une fonction peut être utilisée avant sa définition

Syntaxe (18/18)
Exemples de fonction

fonctions.php

```php
<?php
  function puissance($nombre, $exposant=1) {
    $res = 1;
    for($i=abs($exposant)-1; $i>=0; $i--)
      $res = $res * $nombre;
    if($exposant > 0)
      $retour = $res;
    else
      $retour = 1/$res;
    return  $retour;
  }
  function incrementer(&$nombre, $increment=1) {
    $nombre += $increment;
  }
$val = 4;
incrementer($val, 2);
echo "val = ".$val."</p>";
echo "<p>puissance(2,-2) = ".puissance(2,-2)."</p>";
$fonction = 'puissance';
echo "<p>$fonction(2,4) = ".$fonction(2,4)."</p>";
 ?>
```

Chaînes de caractères (1/9)
Déclaration et fonctionnement

- Les chaînes peuvent être déclarées avec :
  - Simples quotes : `$t='texte';`
  - Doubles quotes : `$t="texte";`
- Fonctionnement différent : entre doubles quotes, les variables et les caractères échappatoires sont interprétés

Exemples, pour `$t="Mot";`

Double cotes	Résultat	Simple cote	Résultat
"Texte"	Texte	'Texte'	Texte
"Texte $t"	Texte Mot	'Texte $t'	Texte $t
"Texte \n Suite"	Texte Suite	'Texte \n Suite'	Texte \n Suite
"Texte \"guillemets\""	Texte "guillemets"	'Texte \"guillemets"\'	Texte \"guillemets"\
"L'heure"	L'heure	'L\'heure'	L'heure

Chaînes de caractères (2/9)
Opérations sur les chaînes de caractères

- Longueur : `int strlen(string $ch)`
- Répétition : `string str_repeat(string $cr, int nb)`
- Minuscules : `string strtolower(string $ch)`
- Majuscules : `string strtoupper(string $ch)`
- Initiales en majuscules : `string ucwords(string $ch)`
- 1^{ere} lettre en majuscule : `string ucfirst(string $ch)`
- Suppression de caractères en début de chaîne :
  `string ltrim(string $ch, string liste)`
- Suppression de caractères en fin de chaîne :
  `string rtrim(string $ch, string liste)`
- Suppression de caractères en début et fin de chaîne :
  `string trim(string $ch, string liste)`

Chaînes de caractères (3/9)
Exemple

traitementString.php

```
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="content-type" content="text/html;charset=utf-8" />
    <title>Page PHP</title>
  </head>
  <body>
    <?php
      $prenom = "...JEan__";
      $nom = "  BONNEAu";
      $adresse = "10 rue abraham lincoln";
      $email = "jean-BONNEAU@asi.insa-rouen.fr";

      $complet = ucfirst(strtolower(trim($prenom,'._')));
      $complet .= " ".strtoupper(ltrim($nom,' '));
      $espaces = strlen($complet)+3;
      echo $complet." : ".ucwords($adresse)."<br />";
      echo str_repeat(".", $espaces).strtolower($email);
    ?>
  </body>
</html>
```

Chaînes de caractères (4/9)
Sous-chaînes de caractères

- Recherche sensible à la casse (retourne tous les caractères de `$ch` depuis la 1^{ere} occurence de `$ch2` jusqu'à la fin) :
  `string strstr(string $ch, string $ch2)`
- Recherche insensible à la casse :
  `string stristr(string $ch, string $ch2)`
- Extraction de chaînes de caractères :
  `string substr(string $chr, int indice, int N)`
- Décompte du nombre d'occurences d'une sous-chaîne :
  `int substr_count(string $ch, string $ssch)`
- Remplacement :
  `string str_replace(string $oldssch, string $newssch, string $ch)`
- Position : `int strpos(string $ch, string $ssch)`

Chaînes de caractères (5/9)
Exemples

traitementString2.php

```
    <?php
      $ch = "Un pot de lait et un pot de miel";
      echo strstr($ch, "pot").<br />";
            // affiche "pot de lait et un pot de miel"
      echo substr($ch, 18, 6).<br />";
            // affiche "un pot"
      echo substr_count($ch, "pot").<br />";
            // affiche "2"
      echo str_replace("pot", "broc", $ch).<br />";
            // affiche "Un broc de lait et un broc de miel"
      echo strpos($ch, "un pot").<br />";
            // affiche "18"
    ?>
  </body>
</html>
```

Les fichiers (1/3)
Ouverture des fichiers

- **Ouverture :** `$fichier=fopen("NOM", "MODE");` avec `MODE` valant :
  - `r`, `r+` : lecture et lecture/écriture, pointeur au début
  - `w`, `w+` : écriture et lecture/écriture, avec création ou effacement, pointeur au début
  - `a`, `a+` : écriture et lecture/écriture, pointeur à la fin, avec création
  - `x`, `x+` : création en écriture et lecture/écriture, pointeur au début, erreur en cas d'existence du fichier
  - `c`, `c+` : création en écriture et lecture/écriture, pointeur au début, sans erreur
- **Verrouillage d'un fichier :**
  `bool flock($fichier, int $operation)`
- **Fermeture :** `fclose($fichier);`

Les fichiers (2/3)
Gestion des fichiers

- **Lecture d'une ligne :**
  `string fgets($fichier [, integer nbOctets])`
- **Lecture d'un caractère :**
  `string fgetc($fichier)`
- **Ecriture d'une ligne :**
  `integer fputs($fichier, string)`
- **Test de fin de fichier :**
  `boolean feof($fichier)`
- **Positionnement :** `fseek($fichier, int $position);`

Les fichiers (3/3)
Exemple

fichier.php

```php
<!DOCTYPE html>
<html>
  <head>
        <title>Lecture/Ecriture dans un fichier</title>
        <meta http-equiv="Content-type" content="text/html; charset=utf-8"/>
  </head>
     <?php
        $fichier=fopen("fichier.txt", "a");
        fputs($fichier, "Une phrase\n");
        fclose($fichier);

        $fichier=fopen("fichier.txt", "r");
        echo "<p>Dans le fichier fichier.txt :</p>";
        while(!feof($fichier)){
          echo fgetc($fichier);
        }
        fclose($fichier);
     ?>
</html>
```

Inclusion de fichiers

Quand le code grossit, il faut le répartir dans plusieurs fichiers :
`include("fichier.php")` : le contenu de ce fichier est inséré ici.

Variantes de `include` :

- `include_once` : le fichier est inclus que s'il ne l'a pas encore été (utile pour les bibliothèques de code)
- `require` : en cas d'absence du fichier, arrêter tout sur une erreur
- `require_once` : fusion des 2 précédents, le plus utilisé et le plus pratique

Formulaires HTML et variables PHP superglobales

Variables superglobales (toujours accessibles)

- créées en interne par PHP (et non par le code utilisateur)
- débutant par `$_`

3 superglobales utiles pour les formulaires :

- `$_GET` : tableau des paramètres GET (reçus par l'URL)
- `$_POST` : tableau des paramètres POST (reçus par le champ POST de l'en-tête HTTP)
- `$_REQUEST` : fusion de ces 2 tableaux

Donc `$_REQUEST` convient que le formulaire utilise *method='GET'* ou *method='POST'*.

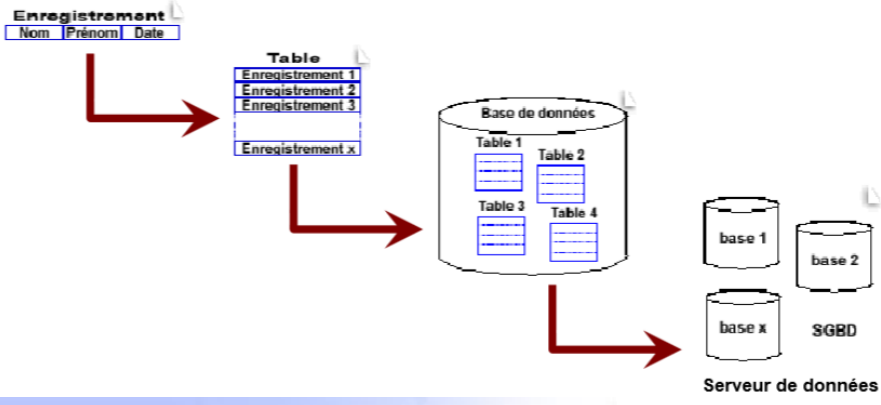
Utiliser un objet en PHP

Un objet est une variable qui regroupe des données et des fonctions.

```php
$objet = new MaClasseObjet();
$objet->attribut = "OK";
echo $objet->attribut;
$objet->methode(10);
```

ACCES aux SGBD

Un SGBD est un ensemble d'applications permettant de manipuler les données (ajout, suppression, modification et lecture), mais aussi de contrôler l'accès. Les données sont structurées de la manière suivante :



ACCES aux SGBD (3)

L'utilisation en général d'un SGBD (tel que MySQL) avec PHP s'effectue en 5 temps :

1. Connexion au serveur de données
2. Sélection de la base de données
3. Requête
4. Exploitation des requêtes
5. Fermeture de la connexion

ACCES aux SGBD (4)

1- Connexion au serveur de données

Pour se connecter au serveur de données, il existe 2 méthodes :

- ✓ Ouverture d'une connexion simple avec la fonction mysql_connect
- ✓ Ouverture d'une connexion persistante avec la fonction mysql_pconnect

Remarque : la deuxième méthode diffère de la première par le fait que la connexion reste active après la fin du script.

```
<?
 if( mysql_connect("ma_base" , $login , $password ) > 0 )
  echo "Connexion réussie ! " ;
 else
  echo "Connexion impossible ! " ;
?>
```

ACCES aux SGBD (5)

2- Sélection de la base de données

Pour faire cette sélection, utilisez la fonction mysql_select_db et vous lui passez en paramètre, le nom de votre base.

```
<?
 if( mysql_select_db("ma_base" ) == True )
  echo "Sélection de la base réussie" ;
 else
  echo "Sélection de la base impossible" ;
?>
```

Remarque : les étapes sélection et requête peuvent être faites en même temps, mais il est plus simple surtout pour une seule base, de sélectionner la table avant de commencer les requêtes. Ainsi, toutes les requêtes à venir utiliseront cette base par défaut.

ACCES aux SGBD (6)

3- Envoi d'une requête

Pour envoyer ces requêtes, on peut utiliser 2 fonctions :

- ✓ mysql_query dans le cas où la base de données serait déjà sélectionnée
- ✓ mysql_db_query dans le cas où l'on voudrait sélectionner la base en même temps.

```
<?
 $requête = "SELECT * FROM membres WHERE pseudo =
'président'  ";
 $résultat = mysql_query( $requête );
?>
```

ACCES aux SGBD (7)

4- Exploitation des requêtes

Après l'exécution d'une requête de sélection, les données ne sont pas "affichées", elles sont simplement mises en mémoire, il faut les chercher, enregistrement par enregistrement, et les afficher avec un minimum de traitement.

PHP gère un pointeur de résultat, c'est celui qui est pointé qui sera retourné.
Lorsque vous utilisez une fonction de lecture, le pointeur est déplacé sur l'enregistrement suivant et ainsi de suite jusqu'à ce qu'il n'y en ait plus.

Les fonctions qui retournent un enregistrement sont :
mysql_fetch_row, mysql_fetch_array et mysql_fetch_object et prennent comme paramètre l'identifiant de la requête.

Les 3 exemples suivants partent d'une requête "SELECT nom, prénom, date FROM membres."

ACCES aux SGBD (8)

mysql_fetch_row : Cette fonction retourne un enregistrement sous la forme d'un tableau simple.

```
<?
 $enregistrement = mysql_fetch_row
($résultat);

 // Affiche le champ - nom -
 echo $enregistrement[0] . "<br>";

// Affiche le champ - prénom -
 echo $enregistrement[1] . "<br> ";

// Affiche le champ - date -
 echo $enregistrement[2] . "<br> ";
?>
```

mysql_fetch_array : Cette fonction retourne un enregistrement sous la forme d'un tableau associatif.

```
<?
 $enregistrement = mysql_fetch_array
($résultat);

 // Affiche le champ - prénom -
 echo $enregistrement["prénom"]."<br>";

 // Affiche le champ - nom -
 echo $enregistrement["nom"] . "<br>";

 // Affiche le champ - date -
 echo $enregistrement["date"] . "<br>";
?>
```

ACCES aux SGBD (9)

mysql_fetch_object : Cette fonction retourne un enregistrement sous forme d'une structure (objet).

```
<?
 $enregistrement = mysql_fetch_object ($résultat );
 // Affiche le champ - date -
 echo $enregistrement->date . "<br>";
 // Affiche le champ - nom -
 echo $enregistrement->nom . "<br>";
 // Affiche le champ - prénom -
 echo $enregistrement->prénom . "<br>";
?>
```

ACCES aux SGBD (10)

Si il n'y a pas ou plus d'enregistrement à lire, ces fonctions retournent "false."
Pour savoir combien d'enregistrements ont été retournés par la sélection, la commande **mysql_num_rows** prend comme paramètre l'identifiant de la requête.

```
<?
 echo "Il y a " . mysql_num_rows( $résultat ) . " membre(s) ";
 while( $enregistrement = mysql_fetch_array( $résultat ))
 {
  echo $enregistrement['nom'] . " " . $enregistrement['prénom'];
  echo " – " . $enregistrement['date'] . "<br>" ;
 }
?>
```

ACCES aux SGBD (11)

5- Fermeture de la connexion

Vous pouvez fermer la connexion au moyen de la fonction mysql_close, mais il est bon de savoir que cette opération sera faite lorsque le script se terminera. C'est donc une opération *facultative*.

Gestion des erreurs

S'il y a une erreur dans la syntaxe de la requête, on utilise la fonction mysql_error qui ne prend pas de paramètres.

```
<?
 $résultat = mysql_query( $requête )
  or die ("Erreur dans la requête : " . $requête . "<br>Avec l\"erreur :
". mysql_error() );
?>
```

Interroger une table MySQL

Requête SQL :

```
CREATE TABLE famille_tbl ( id int(11) NOT NULL auto_increment, nom
varchar(255) NOT NULL, prenom varchar(255) NOT NULL, statut varchar(255) NOT NULL,
date date DEFAULT '0000-00-00' NOT NULL, PRIMARY KEY (id), KEY id (id), UNIQUE id_2
(id) );

INSERT INTO famille_tbl VALUES( '', 'Dupond', 'Grégoire', 'Grand-père', '1932-05-17');
INSERT INTO famille_tbl VALUES( '', 'Dupond', 'Germaine', 'Grand-mère', '1939-02-15');
INSERT INTO famille_tbl VALUES( '', 'Dupond', 'Gérard', 'Père', '1959-12-22');
INSERT INTO famille_tbl VALUES( '', 'Dupond', 'Marie', 'Mère', '1961-03-02');
INSERT INTO famille_tbl VALUES( '', 'Dupond', 'Julien', 'Fils', '1985-05-17');
INSERT INTO famille_tbl VALUES( '', 'Dupond', 'Manon', 'Fille', '1990-11-29');
```

Structure de la table (PhpMyAdmin) :

Champ	Type	Null	Defaut	Extra
id	int(11)	Non	0	auto_increment
nom	varchar(255)	Non		
prenom	varchar(255)	Non		
statut	tinyint(255)	Non	0	
date	date	Non	0000-00-00	

Contenu de la table "famille_tbl"

id	nom	prenom	statut	date
1	Dupond	Grégoire	Grand-père	1932-05-17
2	Dupond	Germaine	Grand-mère	1939-02-15
3	Dupond	Gérard	Père	1959-12-22
4	Dupond	Marie	Mère	1961-03-02
5	Dupond	Julien	Fils	1985-05-17
6	Dupond	Manon	Fille	1990-11-29

Affichage des résultats tels qu'ils sont dans la table sans condition :

Code PHP

```php
<?php
// on se connecte à MySQL
$db = mysql_connect('localhost', 'login', 'password');

// on sélectionne la base
mysql_select_db('nom_de_la_base',$db);

// on crée la requête SQL
$sql = 'SELECT nom,prenom,statut,date FROM famille_tbl';

// on envoie la requête
$req = mysql_query($sql) or die('Erreur SQL !<br>'.$sql.'<br>'.mysql_error());

// on fait une boucle qui va faire un tour pour chaque enregistrement
while($data = mysql_fetch_assoc($req))
    {
    // on affiche les informations de l'enregistrement en cours
    echo '<b>'.$data['nom'].' '.$data['prenom'].'</b> ('.$data['statut'].')';
    echo ' <i>date de naissance : '.$data['date'].'</i><br>';
    }

// on ferme la connexion à mysql
mysql_close();
?>
```

Donne à l'écran

Dupond Grégoire (Grand-père), *date de naissance : 1932-05-17*
Dupond Germaine (Grand-mère), *date de naissance : 1939-02-15*
Dupond Gérard (Père), *date de naissance : 1959-12-22*
Dupond Marie (Mère), *date de naissance : 1961-03-02*
Dupond Julien (Fils), *date de naissance : 1985-05-17*
Dupond Manon (Fille), *date de naissance : 1990-11-29*

Affichage des résultats par ordre alphabétique de prénom.

Le code PHP de la requête

```php
<?php
// Gardez le code ci-dessus, changez juste la requête SQL !
$sql = 'SELECT nom,prenom,statut,date FROM famille_tbl ORDER BY prenom';

// L'opérateur ORDER BY permet de classer soit alphabétiquement
// soit numériquement suivant le type du champ.

// Si l'on souhaite classer en décroissant (ex. de Z à A), nous
// y ajouterons DESC soit : ORDER BY prenom DESC
?>
```

Donne à l'écran

Dupond Gérard (Père), *date de naissance : 1959-12-22*
Dupond Germaine (Grand-mère), *date de naissance : 1939-02-15*
Dupond Grégoire (Grand-père), *date de naissance : 1932-05-17*
Dupond Julien (Fils), *date de naissance : 1985-05-17*
Dupond Manon (Fille), *date de naissance : 1990-11-29*
Dupond Marie (Mère), *date de naissance : 1961-03-02*

Affichage des résultats par comparaison de date.

Le code PHP de la requête

```php
<?php
// Gardez le code ci-dessus, changez juste la requête !
$sql = "SELECT nom,prenom,statut FROM famille_tbl WHERE date<'1960-01-01'";

// L'avantage d'avoir un type DATE dans notre base de données, c'est que
// nous pouvons comparer des dates dans la requête SQL.
// Ici nous ne souhaitons afficher que les membres de la famille qui sont
// nés avant le 1er janvier 1960, soit : WHERE date<'1960-01-01'

?>
```

Donne à l'écran

Dupond Grégoire (Grand-père), *date de naissance : 1932-05-17*
Dupond Germaine (Grand-mère), *date de naissance : 1939-02-15*
Dupond Gérard (Père), *date de naissance : 1959-12-22*

Affichage des résultats avec le commande LIKE.

La commande LIKE en SQL permet de fouiller le contenu de chaque champ. Je m'explique, je recherche tous les enregistrements dont le champ "**prenom**" commence par la lettre "**G**", voyons la syntaxe ci-dessous.

Le code PHP de la requête

```php
<?php
// Gardez le code ci-dessus, changez juste la requête !
$sql = "SELECT nom,prenom,statut,date FROM famille_tbl WHERE prenom LIKE 'G%'";

// Le signe pourcentage "%" placé après le G, indique que la lettre G peut
// être suivie, mais pas précédée, d'autres caractères !
// Notez aussi que LIKE n'est pas sensible à la casse, cela veut dire que
// la requête cherchera aussi bien des G majuscules que minuscules.

?>
```

Donne à l'écran

Dupond Grégoire (Grand-père), *date de naissance : 1932-05-17*
Dupond Germaine (Grand-mère), *date de naissance : 1939-02-15*
Dupond Gérard (Père), *date de naissance : 1959-12-22*

Maintenant voyons la même chose mais cette fois nous allons chercher la syllabe "**ma**" dans le champ "**prenom**", qu'elle soit placée au début ou au milieu d'autres caractères.

Le code PHP de la requête

```php
<?php
// Gardez le code ci-dessus, changez juste la requête !
$sql = "SELECT * FROM famille_tbl WHERE prenom LIKE '%MA%'";

// Le signe pourcentage "%" placé avant et après MA indique que la syllabe
// peut-être précédée ou suivie de caractères.
// Une fois de plus notez que LIKE n'est pas sensible à la casse, la requête
// cherchera aussi bien des MA majuscules que des ma en minuscules.

?>
```

Donne à l'écran

Dupond Marie (Mère), *date de naissance : 1961-03-02*
Dupond Manon (Fille), *date de naissance : 1990-11-29*

La commande INSERT INTO

Cette commande permet d'insérer des enregistrements dans une table en l'occurrence clients_tbl.

- `INSERT INTO clients_tbl(id,prenom,nom,ne_le,ville,enfants) VALUES('','Patrick','Martin','1965-10-08','Bordeaux','2')`

Ci-dessous la table : clients_tbl avec le nouvel enregistrement

id	prenom	nom	ne_le	ville	enfants
1	Patrick	Martin	1965-10-08	Bordeaux	2

La commande UPDATE

Cette commande permet de modifier les valeurs d'un enregistrement déjà présent dans la table :

- `UPDATE clients_tbl SET prenom='Jacques' WHERE id=1`

Cette commande ne pose vraiment pas de problème particulier , décortiquons la syntaxe :

UPDATE clients_tbl	Mise à jour de la table **Clients_tbl**
SET prenom='Jacques'	Modifier le champ **prenom** pour la valeur **Jacques**
WHERE id=1	Quand le champ **id** est **égal à 1**

Ci-dessous l'enregistrement de la table : clients_tbl une fois modifié.

id	prenom	nom	ne_le	ville	enfants
1	Jacques	Martin	1965/10/08	Bordeaux	2

Bien sûr nous pouvons changer plusieurs valeurs d'un même enregistrement dans la même requête :

- `UPDATE clients_tbl SET prenom='Jean-Pierre', nom='Papin', ville='Marseille', enfants=3 WHERE id=1`

La commande DELETE

Bon vous l'aurez sans doute compris cette commande sert à supprimer un ou plusieurs enregistrements d'une table ainsi :

- `DELETE FROM clients_tbl WHERE id=1`

Là non plus la commande ne pose vraiment pas de problème particulier, décortiquons la syntaxe :

`DELETE FROM clients_tbl`	Effacer de la table **Clients_tbl**
`WHERE id=1`	Quand l'id de l'enregistrement est **égal à 1**

Source : http://www.phpdebutant.org

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 écrire 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")
```

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(12)

- **Exemples d'utilisation de cookies**

```
//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)

Comment garder des données utilisateur entre 2 exécutions d'un script ?

```php
<?php
$compteur++; // vaudra toujours 1
echo "Nombre de visites connues : $compteur";
```

normal.php

```php
<?php
session_start();
$_SESSION['compteur']++;
echo "Nombre de visites connues : $_SESSION[compteur]";
```

session.php

Une session permet de rendre des données PHP persistantes entre les requêtes d'un même utilisateur.

Les sessions(2)

- **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)

Les sessions(3)

- **Principe**
  - 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    ?>
```

  - 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(4)

- **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(5)

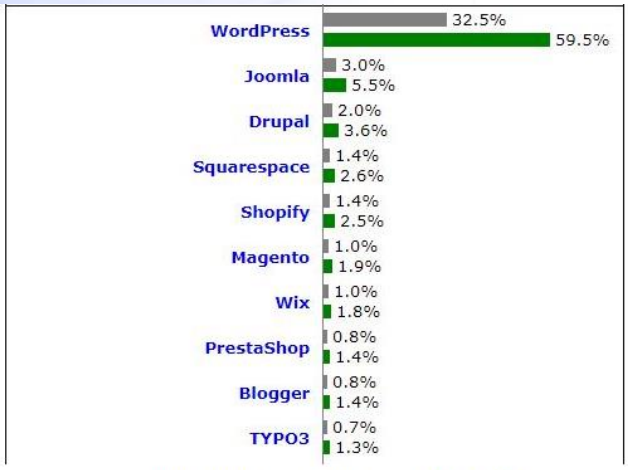
Exemple

```
<!-- 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(6)

```
<?
 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());
?>
```

Top 10 des logiciels CMS



Classement au 07 Décembre 2018 - Source : W3techs.com

1. WordPress : **32,5%** d'usage et **59,5%** de part de marché des CMS.

TOP10BEST
CRÉATION SITE INTERNET

WIX.com
1&1
SimpleSite
SiteW
SITE123
JIMDO
weebly
GoDaddy
one.com
web.com