Introduction au PHP, Mysql et programmation avancée

Plan :

1er Partie : PHP

1. Qu'es ce que le PHP ?
2. Que peu faire le PHP ?
3. Notion de serveur web
4. EasyPHP
5. Les balises d'insertion du code PHP
6. Votre première page en PHP(**TP1**)
7. Les commentaires, les variables et les Constantes
8. Les operateurs (arithmétique, comparaison, booléen…)
9. Les instructions conditionnelles et les boucles
10. Travaux Pratiques N°2(**TP2**)
11. Les fonctions utilisateurs
12. Gestion des formulaires (Envoi, Récupération, test, traitement)
13. Travaux Pratiques N°3(**TP3**)
14. Gestion des fichiers : création, Lecture/écriture
15. Travaux Pratiques N°4(**TP4**)

2ème Partie : MySQL

1. Introduction aux bases de données
2. Présentation de MySQL
3. Présentation de l'outil d'administration de MySQL : PHPMyAdmin
4. Fonctions PHP permettant de se connecter à MYSQL
5. Travaux Pratiques N°5(**TP5) :** Test de connexion
6. Fonctions PHP permettant d'exécuter et d'exploiter le résultat d'une requête
7. Etudes des différentes requêtes: Insertion, affichage, modification, suppression.
   - Travaux Pratiques N°5(**TP5) :** Formulaire d'insertion dans une table
   - Travaux Pratiques N°5(**TP6) :** Script d'affichage des enregistrements dans un tableau
8. Travaux Pratiques N°5(**TP7)**

Partie 1 : PHP

1. Qu'es ce que le PHP ?

PHP, signifie "*Hypertext Preprocessor*" (Préprocesseur HyperTexte) et est un langage de script. L'essentiel de sa syntaxe est emprunté aux langages C, Java et Perl, mais y ajoute plusieurs fonctionnalités uniques. Le but de ce langage est de permettre aux développeurs web de concevoir rapidement des sites aux contenus dynamiques.

C'est un langage incrusté au HTML et interprété côté serveur.

Il est extensible grâce à de nombreux modules et son code source est ouvert. Comme il supporte tous les standards du web et qu'il est gratuit, il s'est rapidement répandu sur la toile.

2. Que peu faire le PHP ?

Le langage PHP possède les mêmes fonctionnalités que les autres langages permettant d'écrire des scripts CGI, comme collecter des données, générer dynamiquement des pages web …etc.

La plus grande qualité et le plus important avantage du langage PHP est le support d'un grand nombre de bases de données. Réaliser une page web dynamique interfaçant une base de données est extrêmement simple en PHP.

PHP supporte plusieurs bases de données parmi lesquelles on peut citer : MySQL, Oracle, InterBase, PostgreSQL, dBase, et j'en passe.

3. Notion de serveur web

PHP est un langage coté serveur, pour pouvoir voir les résultats de vos pages PHP vous devez avoir un environnement dans lequel se trouve un serveur web.

Un serveur HTTP ou démon HTTP ou HTTPd (HTTP daemon) ou tout simplement serveur Web, est un logiciel servant des requêtes respectant le protocole de communication client-serveur HyperText Transfer Protocol (HTTP), qui a été développé pour le World Wide Web.

Un ordinateur sur lequel fonctionne un serveur HTTP est appelé serveur Web. Le terme «serveur Web » peut aussi désigner le serveur HTTP (le logiciel) lui-même.

Les serveurs HTTP les plus utilisés sont :

• Apache HTTP Server de la Apache Software Foundation

• Internet Information Services de Microsoft (IIS)
• Sun ONE de Sun Microsystems
• Le serveur Web Zeus de Zeus Technology
Le plus populaire est Apache HTTP Server qui sert environ 70% des sites Web

4. WampServer

WampServer installe et configure automatiquement un environnement de travail complet permettant de mettre en œuvre toute la puissance et la souplesse qu'offrent le langage dynamique PHP et son support efficace des bases de données. WampServer regroupe un serveur Apache, une base de données MySQL, le langage PHP ainsi que des outils facilitant le développement de vos sites ou de vos applications.

- **Installer WampServer :**

Télécharger **WampServer** sur le site www.baarako.com/ressources.php
Double cliquer sur l'exécutable téléchargé.
Sélectionner le répertoire d'installation et suivre la procédure d'installation.

- **Mise en route de WampServer**

A l'installation, un raccourci vers EasyPHP est créé dans le répertoire "Démarrer/Programmes/Start Wampserver".

Une fois WampServer lancé, une icône () se place dans la barre des tâches à côté de l'horloge.

Un clic gauche permet d'accéder à différents menus :
• Localhost: ouvre la page http://localhost/
• phpMyAdmin: ouvre la page http://localhost/phpmyadmin (interface d'administration de la base de données mysql)
• www directory: ouvre la racine du serveur (le repertoire www)
• Start all Service : démarre Apache et MySQL
• Stop all Service : arrête Apache et MySQL
• Restart all Service : redémarre Apache et MySQL
….

- **Utiliser le répertoire www**

Pour que vos pages PHP soient interprétées, il est impératif de placer vos fichiers dans le répertoire www. Le serveur Apache est configuré pour ouvrir automatiquement un fichier index lorsque vous saisissez l'adresse "http://localhost/" (à condition évidemment que le serveur Apache soit en route).

Cette page sert de page d'accueil au web local et permet de vérifier le bon fonctionnement de Wampserver. Il est conseillé de créer un répertoire par projet dans le répertoire www afin d'avoir une vision plus claire des développements.

5. Les balises d'insertion du code PHP

Les pages web sont au format html. Les pages web dynamiques générées avec PHP sont au format php. Le code source PHP est directement insérer dans le fichier html grâce au conteneur de la forme **<? Php ……. ?>**

Autres syntaxes d'intégration du code PHP dans du HTML :

- **<? ……. ?>**
- **<script language= "php"> …… </script>**
- **<% …… %>**

Mais la notation la plus utilisée et la plus conseillée est bien sur **<? Php ……. ?>**
La fonction echo "texte" : permet d'afficher texte sur le navigateur

6. Votre première page en PHP

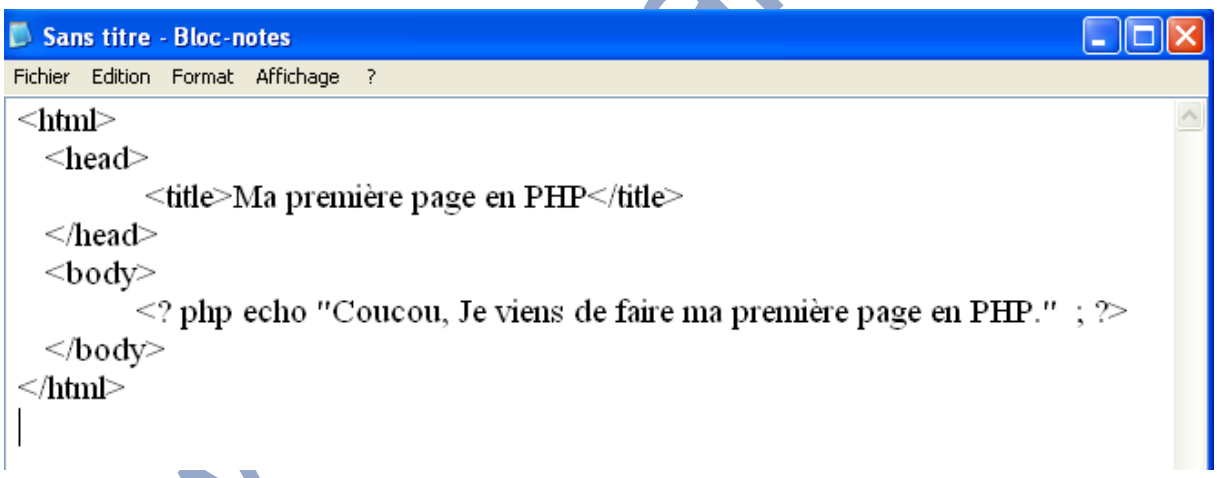
Avant toute chose, créer dans le répertoire www d'EasyPHP un sous dossier nommé "TP1".

Pour programmer en PHP on peu utiliser n'importe quelle éditeur de texte .Nous allons, dans cet exemple, utiliser un simple éditeur de texte (bloc-notes).

L'exemple consiste à afficher une phrase.

Le code PHP s'intègre directement au code HTML et commence par <?php et se termine par ?>.

Lancer le bloc-notes et tapez le code suivant :



- **Enregistrement de la page**

Enregistrez le document en lui donnant comme nom : "tp1.php" et placez le dans le dossier TP1
Attention, en enregistrant votre document vous devez choisir dans Type *tous les fichiers* pour éviter que le bloc-notes ne lui ajoute par défaut l'extension *.txt*.

- **Visualisation du résultat :**

Votre première idée pour lancer votre page ("tp1.php") est d'aller dans le répertoire www puis dans le dossier TP1 et double-cliquer sur le fichier "tp1.php" ! Et bien ne faites jamais cela sinon vous serez dessus.

Ce qu'il faut faire c'est de lancer EasyPHP (qui démarrera à son tout le serveur web APACHE). Ensuite lancer votre navigateur Internet Explorer par exemple et de tapez dans la bar d'adresse : http://localhost/tp1/tp1.php ou bien http://127.0.0.1/tp1/tp1.php et vous obtenez le résultat attendu c'est-à-dire une fenêtre contenant le message : *Coucou, Je viens de faire ma première page en PHP.*

7. Les commentaires, les variables et les Constantes

- **Les Commentaires :**

Un script PHP se commente comme en C ou en JAVA :

Exemple :

```
<? Php

// Commentaire sur une seule ligne

/* commentaire
sur plusieurs
lignes */

?>
```

Tout ce qui se trouve dans un commentaire est ignoré. Il est conseillé de commenter largement vos scripts pour une meilleure clarté du code.

- **Les variables :**

Une variable correspond à un espace de la mémoire où l'on peut stocker une information. Afin de pouvoir récupérer cette information lorsque l'on en a besoin, chaque variable a un nom.

En PHP, les variables sont représentées par une chaîne de caractères, ayant toujours comme premier caractère, le caractère dollar (**$**).

Un nom de variable valide doit commencer par une lettre ou un souligné (_) suivi de lettres, chiffres ou soulignés. Il n'ya pas d'espace entre les caractères d'un nom de variable.

Exemple : de nom de variable valide

```
$nom
$_nom
$mon_nom
$nom215_prenom21
$_12nom
```

Exemple : de nom de variable valide

```
$12nom      // invalide car commence par un chiffre
$mon nom    // invalide car contient d'espace
```

Le typage des variables est implicite en PHP. Il n'est donc pas nécessaire de déclarer leur type au préalable ni même de les initialiser avant leur utilisation.

Les variables peuvent être de type entier (**integer**), réel (**double**), chaîne de caractères (**string**), tableau (**array**), objet (**object**), booléen (**boolean**).

PHP est un langage sensible à la casse donc le nom des variables est sensible à la casse
La variable **$nom** n'est pas la même que **$NOM** qui est différent aussi de **$Nom**

Quand on veut affecter une valeur à une variable on utilise l'operateur =

Exemple :

```
<?php
$nom = "Souleymane";
// La variable $nom contient alors la chaîne de caractères Souleymane.

$mon_age = 25;
// La variable  $mon_age contient la valeur numérique 25.

?>
```

On peu alors utiliser ces variables facilement

```
<?php
$nom = "Souleymane";
// La variable $nom contient alors la chaîne de caractères Souleymane.

$mon_age = 25;
// La variable  $mon_age contient la valeur numérique 25.

echo "Bonjour  $nom, Vous avez $mon_age ans. "

?>
```

Ce qui va afficher le message : *Bonjour Souleymane, Vous avez 25 ans.*

Les tableaux : (array)

En informatique, un tableau est une variable contenant différente informations classées selon un numéro (indice du tableau).

Par exemple supposons qu'on a 5 types de fruits : mangue, orange, banane, pomme, pêche. Ces types de fruit peuvent être classés dans un tableau appelé fruit. On va avoir alors le code PHP suivant :

```
<?php

$fruit[0] = "mangues";
$fruit[1] = "oranges";
$fruit[2] = "bananes";
$fruit[3] = "pommes";
$fruit[4] = "pêches";

Echo "J'aime les $fruit [0] et les $fruit [3] mais les $fruit [1] me donnent des nausées"
?>
```

Ce qui va afficher « J'aime les mangues et les pommes mais les oranges me donnent des nausées »

Un tableau accepte des éléments de tout type. Les éléments d'un tableau peuvent être de types différents et sont séparés d'une virgule.

Un tableau peut être initialisé avec la syntaxe **array**.
Exemple:
$tab_couleurs = array ('rouge', 'vert', 'jaune', 'blanc');
$tab = array ('souleymane', 2002, 20.5, $name);

Mais il peut aussi être initialisé au fur et à mesure.
Exemples :
$prenoms[ ] = ''mimi'';** **$villes[0] = ''Paris'';
$prenoms[ ] = ''baba'';** **$villes[1] = ''Alger'';
$prenoms[ ] = ''soul'';** **$villes[2] = ''Bamako'';

L'appel d'un élément du tableau se fait à partir de son indice (dont l'origine est zéro).

Exemple : echo $tab[10]; // pour accéder au 11ème élément

- **Les constantes :**

Une constante est un identifiant (un nom) qui représente une valeur simple. Elles ne peuvent jamais être modifiée ou détruite durant l'exécution du script.
Vous pouvez définir une constante en utilisant la fonction define().
Seuls les types de données scalaires peuvent être placés dans une constante : booléen, entier, double et chaîne de caractères

Exemple :

```
<?php

 define("CONSTANTE", "J'aime PHP .");
 echo CONSTANTE;                          // affiche "J'aime PHP
?>
```

Quelques fonctions utilisées avec les variables :

empty($var) : renvoie vrai si la variable est vide
isset($var) : renvoie vrai si la variable existe
unset($var) : détruit une variable
gettype($var) : retourne le type de la variable
settype($var, ''type'') : convertit la variable en type type
is_long(), is_double(), is_string(), is_array(), is_object(), is_bool(), is_float(), is_numeric(), is_integer(), is_int()…

Quelques fonctions utilisées aux chaines de caractère:

strlen($str) : retourne le nombre de caractères d'une chaîne
strtolower($str) : conversion en minuscules

strtoupper($str) : conversion en majuscules
trim($str) : suppression des espaces de début et de fin de chaîne
substr($str,$i,$j) : retourne une sous chaîne de **$str** de taille **$j** et débutant à la position **$i**
strnatcmp($str1,$str2) : comparaison de 2 chaînes
addslashes($str) : déspécialise les caractères spéciaux (‘, ‘’, \)
ord($char) : retourne la valeur ASCII du caractère **$char**

8. Les operateurs (arithmétique, comparaison, booléen)

- **Les Operateurs arithmétiques :**

Exemple	Nom	Résultat
$a + $b	Addition	Somme de $a et $b
$a - $b	Soustraction	Différence de $a et $b
$a * $b	Multiplication	Produit de $a et $b
$a / $b	Division	Quotient de $a et $b
$a % $b	Modulo	Reste de $a divisé par $b

- **Les opérateurs de comparaison**

Exemple	Nom
$a == $b	Égal
$a != $b	Différent
$a < $b	Strictement inférieur
$a <= $b	Inférieur ou égal
$a > $b	Strictement supérieur
$a >= $b	Supérieur ou égal

- **Les operateurs Booléens :**

Exemple	Nom
$a and $b, $a && $b	ET
$a or $b, $a || $b	OU
$a xor $b	OU Exclusif (soit a ou b mais pas les deux à la fois)
! $a	NON

- **Les operateurs incrémentaux**

Exemple	Nom
$a++	Post-incrémente
++$a	Pré-incrémente
$a--	Post-décrémente
--$a	Pré-décrémente

- **Autres operateurs :**

Exemple	Fonction
$a = $b ; $a = 4 ;** **$a = "foot" ;	Affectation de la valeur de l'expression de droite dans la variable de gauche.
"bonjour "."Monde!"; **$b . $c; $b . "ajout";**	Concaténation de chaînes de caractères.
+=, -=, *=, /=, .=	Opérateurs d'assignation combinés (effectue l'opération puis affecte)

9. Les instructions conditionnelles et les boucles

- **L'instruction IF :**

L'instruction « **if** » est la structure de test la plus basique. Elle permet d'exécuter une suite d'instructions en fonction d'une condition.

Il est possible d'enchaîner une série d'instructions **if** (sans avoir besoin de les imbriquer) à l'aide de l'instruction **elseif**.

```
If ($mon_age <= 30)
{
echo "Vous êtes encore jeune mon ami";
} elseif ($mon_age >40) {
        echo "Vous êtes vieux mon ami";
    }
elseif ( $mon_age>30  &&  $mon_age<=40 )
 {echo "Vous avez le bel âge mon ami" ;
} else {echo "Je ne sais pas mon ami" ;}
```

(Remarquons déjà que les instructions qui doivent être exécutées lorsqu'un test est validé sont systématiquement comprises entre des crochets { }).

- **L'instruction switch :**

L'instruction « **switch** » équivaut à une série d'instructions if. En de nombreuses occasions, vous aurez besoin de comparer la même variable (ou expression) avec un grand nombre de valeurs différentes, et d'exécuter différentes parties de codes suivant la valeur à laquelle elle est égale. C'est exactement à cela que sert l'instruction **switch**.

Ces deux exemples suivant sont équivalent :

```
if ($i == 0) {
print "i égale 0";
}
elseif ($i == 1) {
print "i égale 1";
}
elseif ($i == 2) {
print "i égale 2";
}
else { print ”je ne sais pas “;}
```

```
switch ($i) {
case 0:
print "i égale 0";
break;
case 1:
print "i égale 1";
break;
case 2:
print "i égale 2";
break;
default :
print ”je ne sais pas “;
}
```

Il est important de comprendre que l'instruction **switch** exécute chacune des clauses dans l'ordre.
L'instruction **switch** est exécutée ligne par ligne. Au début, aucun code n'est exécuté.
Seulement lorsqu'un cas est vérifié, PHP exécute alors les instructions correspondantes. PHP continue d'exécuter les instructions jusqu'à la fin du bloc d'instructions du **switch**, ou bien dès

qu'il trouve l'instruction **break**. Si vous ne pouvez pas utiliser l'instruction **break** à la fin de l'instruction **case**, PHP continuera à exécuter toutes les instructions qui suivent.

- **La boucle WHILE :**

La boucle « **while** » est le moyen le plus simple d'implémenter une boucle en PHP.
Elle est le plus souvent utilisée pour récupérer et afficher des données provenant d'une base de données.
L'exemple le plus simple d'une boucle while est le suivant :

<?php
while (expression) commandes
?>

La signification d'une boucle **while** est très simple. Le PHP exécute l'instruction tant que l'expression de la boucle **while** est évaluée comme TRUE (vraie).
La valeur de l'expression est vérifiée à chaque début de boucle, et, si la valeur change durant l'exécution de l'instruction, l'exécution ne s'arrêtera qu'à la fin de l'itération (chaque fois que le PHP exécute l'instruction, on appelle cela une itération).
De temps en temps, si l'expression du **while** est FALSE (faux) avant la première itération, l'instruction ne sera jamais exécutée.

Soit l'exemple suivant :

```
<?php
$chiffre = 5;
$i = 0;
// Début de la boucle
while ($i < $chiffre) {
echo 'Notre chiffre est différent de '.$i.'<br />';
$i = $i + 1;
}
// Fin de la boucle
echo 'Notre chiffre est égal à '.$i;
?>
```

Ce qui affichera à l'écran :

Notre chiffre est différent de 0
Notre chiffre est différent de 1
Notre chiffre est différent de 2
Notre chiffre est différent de 3
Notre chiffre est différent de 4
Notre chiffre est égal à 5

Ici, on initialise notre variable $chiffre à 5, puis la variable $i à 0.
Ensuite, nous faisons le test suivant : "tant que $i et augmenter la valeur de $i de 1"
Puis dès que la condition $i < $chiffre n'est plus vérifiée, nous sortons de la boucle pour finir l'exécution des instructions qui suivent.

- **La boucle for**

La structure de contrôle **for** nous permet d'écrire des boucles. En clair, cela veut dire que nous allons exécuter une série d'instructions pour un nombre de fois bien déterminé.

Reprenons l'exemple précédent en utilisant la boucle for :

```
<?php
$chiffre = 5;
// Début de la boucle
for ($i=0; $i < $chiffre; $i++) {
echo 'Notre chiffre est différent de '.$i.'<br />';
}
// Fin de la boucle
echo 'Notre chiffre est égal à '.$i;
?>
```

Ce qui affichera exactement la même chose que précédemment.

En effet, on initialise notre variable $chiffre à 5. On démarre la boucle **for** qui dit que l'on va exécuter les instructions situées entre les crochets de la boucle (**{ }**) pour i variant de 0 à $chiffre-1 (donc jusqu'à 4), i étant incrémenter à chaque passage de boucle ($i++). ($i varie de 0 à $chiffre-1 car on impose que $i soit strictement inférieur à $chiffre).
On exécute alors 4 fois les instructions présentes dans la boucle, et à chaque passage, $i verra sa valeur augmentée de 1.

10.Travaux Pratiques N°2(TP2)

Créer une page « tp2.php ».Cette page contient la déclaration d'une variable nom, d'une constante appelée max qui sera le nombre de caractère maximum de la variable nom. Initialiser la variable nom par votre nom par exemple et la constante aura une valeur égale à 15. Calculer le nombre de caractère de la variable nom, si ce nombre dépasse la constante max afficher un message d'erreur, sinon compter le nombre de voyelle contenu dans la chaine nom.

11. Les fonctions utilisateurs

Les fonctions et les procédures regroupent un ensemble d'instructions réutilisables simplement et qui accomplissent un ensemble d'opérations. Elles peuvent (mais ce n'est pas obligé) accepter des valeurs (appelées "arguments" ou "paramètres"). S'il y en a plusieurs, les arguments sont séparés par des virgules.

- **une fonction** réalise une succession d'instructions et renvoie une (et une seule !) valeur issue d'un calcul (instruction return)
- **une procédure** réalise une succession d'instructions mais ne renvoie pas de valeur en retour.

Que l'on écrive une fonction ou une procédure, le nom de la fonction et/ou de la procédure est précédé du mot-clé **function**
Les instructions qui composent la fonction/procédure sont encadrées de **{**et**}**

Les identificateurs de fonctions sont insensibles à la casse.

Une fonction peut être définie en utilisant la syntaxe suivante :

```
<?php
function   nom_fonction ([param_1, …, param_n]) {

        instruction_1;
        …
        instruction_m;

        [return $resultat;]
}
?>
```

- ✓ **function :** mot clé annonçant la définition d'une fonction,
- ✓ **Nom_fonction() :** nom de la fonction, servira à son appel,
- ✓ **return :** mot clé permettant d'associer un résultat à la fonction. (optionnel)

Exemple :

```
< ?php

 Function somme($a,$b) {

            $s=$a+$b ;
           Return $s;
          }

   Function afficher($r){
       Echo "$r"
  }

?>
```

Nous avons définit ici une fonction appelé somme qui a deux paramètres et qui retourne la somme de ces deux paramètres.
L'appel a cette fonction se fait par :

```
< ?php

 Function somme($a,$b) {

            $s=$a+$b ;
           Return $s;
          }

$resultat=somme(15,5); // appel à la function somme avec la valeur des 2 paramètres
Afficher ($resultat);       // appel à la procédure afficher

?>
```

Transmission des paramètres : par valeur et par référence

Par défaut, "**Par valeur**" est le type de transmission par défaut, c'est-à-dire que la variable conserve sa valeur initiale après l'appel de la fonction même si celle-ci l'a modifiée.

Exemple : A partir d'une fonction qui calcule le double d'une valeur données en paramètre :

```
function double ($n) {
$n = $n * 2;
return $n;
 }
```

Si on passe l'argument "par valeur" à cette fonction double(), on obtient :

```
< ?php
  $val=5 ;
$d=double($val) ;
Echo "$d"     // d=25
Echo "$val"; // val=5 inchangé
?>
```

- "**Par référence"** signifie que est la valeur initiale peut être modifiée dans la fonction.
Syntaxe : Un symbole **&** précède le nom de l'argument (variable).
Exemple : Si on passe l'argument "par référence" à la même fonction double (), on obtient :

```
< ?php

  $val=5 ;
$d=double (&$val) ; //& signifie que l'appel se fait par référence à la variable val
Echo "$d"     // d=25
Echo "$val"; // val=25  changé par la fonction appelée

?>
```

Utiliser les mêmes fonctions et constantes dans différentes pages PHP

Parfois, une fonction, procédure, variable ou constante doit être utilisée dans des pages différentes. Plutôt que de les recopier dans chacune des pages, il est conseillé d'écrire ces scripts dans un fichier PHP et d'inclure ce fichier dans vos scripts.

PHP possède 2 fonctions internes qui concernent l'inclusion de fichiers externes :

- **require** ("Mapage.php") : A l'exécution, cette instruction est remplacée par le contenu de "Mapage.php".
- **include**("mapage.php") :Permet aussi d'inclure le fichier dans le script

Ces deux structures, qui permettent d'inclure un fichier dans la page, sont quasi-identiques, sauf au niveau de la gestion des erreurs. Si le fichier n'existe pas, **include** génère une erreur de niveau Warning (alerte) mais **require** génère une erreur fatale et l'arrêt du script

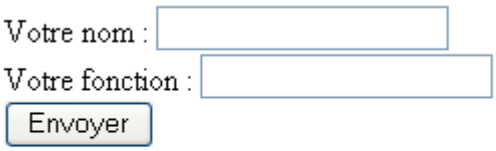
12. Gestion des formulaires (Envoi, Récupération, test, traitement)

Les formulaires vont permettre à vos visiteurs de soumettre des informations, que ce soit un nom, un prénom, un chiffre, etc.

Soit l'exemple suivant :

```
<html>
<head>
<title>Formulaire1</title>
</head>
<body>
<form action = "traitement.php"
method="post">
Votre nom : <input type = "text" name =
"nom"><br />
Votre fonction : <input type = "text" name =
"fonction"><br />
<input type = "submit" value = "Envoyer">
</form>
</body>
</html>
```

Ce qui donne cette page :



Lorsque l'utilisateur cliquera sur le bouton "Envoyer", les données du formulaire seront envoyées sur la page *traitement.php*.

Et dans la page traitement.php, nous allons récupérer une variable de type tableau ($_POST : car notre formulaire a comme *method* la valeur *post*). Dans le cas où le formulaire utilise une méthode get, nous utilisons la variable tableau $_GET.

En clair, dans la page traitement.php, on aura une variable $_POST['nom'] qui contiendra la chaîne de caractères qu'aura saisi le visiteur dans le champ "Votre nom : " (on a la variable $_POST['nom'], car dans l'attribut name de notre formulaire pour le champ concernant le nom). De même, on aura une variable $_POST['fonction'] qui contiendra la chaîne de caractères qu'aura saisi le visiteur dans la champ "Votre fonction : " (encore une fois, on a la variable $_POST['fonction'] car l'attribut name du champ prend la valeur fonction).

Donc le code de la page traitement.php peu être :

```
<Html>
<Head>
<title>Ma page de traitement</title>
</head>
<body>
<?PHP
// on teste la déclaration de nos variables
if ( (isset($_POST['nom']) && isset($_POST['fonction'])) {
// on affiche nos résultats
echo 'Votre nom est '.$_POST['nom'].' et votre fonction est '.$_POST['fonction'];
}
?>
</Body>
</Html>
```

13. Travaux Pratiques N°3(TP3)

Enoncé : Réalisez un formulaire de contact. Ce formulaire contient : trois champs de type *radiobutton* contenant les valeurs Mr, Mme, Mlle ; un champ nom, prénom, Fonction de type combo (Ministre, formateur, étudiante, autres) et deux bouton : de soumission et de reset. Cette page s'appelle « tp3.html ». Ce formulaire sera renvoyé en utilisant la méthode *Post* à une page appelée « tp3.php ». Cette page va récupérer les valeurs des différents champs en les mettant dans des variables, tester la valeur de ces variables a l'aide d'une fonction appelé tester () et enfin afficher la valeur de ces variables a l'aide d'une autres fonction appelé afficher (). La fonction tester () reçoit en paramètre l'ensemble des variables associé aux champs du formulaire (nom, prénom.) Et retourne vrai si ces variables contiennent des valeurs (penser à la fonction empty()). La fonction afficher () reçoit l'ensemble des variables à afficher.
La page « tp3.html » doit représenter à cela :

14. Gestion des fichiers : création, Lecture/écriture

PHP permet également de créer un fichier texte, de lire et d'écrire dans un fichier texte.
Afin de mettre en pratique cet exercice, vous allez créer un fichier *donnees.txt* que vous allez placer dans le même répertoire que le script (Vous n'êtes absolument pas obligé de mettre ce fichier texte dans le même répertoire que le script PHP.)

Ouvrez ce fichier et tapez un message dans ce fichier par exemple mettez dans le fichier la chaine : « J'aime vraiment le PHP »

A l'aide de PHP nous allons accéder à ce fichier et afficher son contenu.

Soit le code suivant :

```php
<?php
// Instruction 1
$fp = fopen ("donnees.txt", "r");
// Instruction 2
$contenu_du_fichier = fgets ($fp, 255);
// Instruction 3
fclose ($fp);
// Instruction 4
echo 'Notre fichier contient : '.$contenu_du_fichier;
?>
```

Ce qui va affiche le texte contenu dans le fichier.

Détaillons alors ce qui se passe :
- **Instruction 1** : on ouvre le fichier *donnees.txt* en lecture seule à l'aide de la fonction **fopen()** (la lecture seule est obtenue à l'aide du paramètre **r** ; nous détaillerons plus loin tous les paramètres possibles de cette fonction).

- **Instruction 2** : on lit le contenu du fichier à l'aide de la fonction **fgets()** et l'on place le contenu de ce fichier dans la variable $contenu_du_fichier (le paramètre 255 passé à la fonction **fgets()** correspond au nombre de caractères à lire : ici, on a donné 255, ce qui correspond a un choix totalement arbitraire. En effet, vous pouvez mettre n'importe quel nombre. En revanche, si vous mettez 20 et que votre fichier comporte 128 caractères, seuls les 20 premiers seront lus).
- **Instruction 3** : on referme le fichier donnees.txt à l'aide de la fonction **fclose()**. En effet, nous avons déjà le contenu du fichier dans la variable $contenu_du_fichier, alors le fichier ne nous intéresse plus.
- **Instruction 4** : on affiche donc le contenu du fichier donnees.txt

Etudions maintenant tous les paramètres possibles de la fonction **fopen()** :
- **r** : ouvre en lecture seule, et place le pointeur de fichier au début du fichier.
- **r+** : ouvre en lecture et écriture, et place le pointeur de fichier au début du fichier.
- **w** : ouvre en écriture seule; place le pointeur de fichier au début du fichier et réduit la taille du fichier à 0. Si le fichier n'existe pas, on tente de le créer.
- **w+** : ouvre en lecture et écriture; place le pointeur de fichier au début du fichier et réduit la taille du fichier à 0. Si le fichier n'existe pas, on tente de le créer.
- **a** : ouvre en écriture seule; place le pointeur de fichier à la fin du fichier . Si le fichier n'existe pas, on tente de le créer.
- **a+** : ouvre en lecture et écriture; place le pointeur de fichier à la fin du fichier. Si le fichier n'existe pas, on tente de le créer.

On peut également écrire dans un fichier texte a l'aide de la fonction
fputs(fichier,info_a_ecrire)

15.Travaux Pratiques N°4(TP4)

On souhaite comptabiliser le nombre de chargement d'une page (la page d'accueil par exemple). On va procéder de la façon suivante : le compteur numérique sera stocké dans un fichier, à la première ligne. Ce fichier contiendra seulement un nombre, celui des visites.

Ce fichier va s'appeler **compteur.cpt**.
Principe de l'algorithme : si le fichier n'existe pas encore (**file_exists**), alors on le crée et on l'ouvre en écriture (**fopen w**) et on initialise le compteur à zéro en écrivant la chaîne **'0'** en première ligne (**fputs**) et on le referme (**fclose**). Ensuite, ouverture du fichier en lecture plus écriture (**fopen r+**), lecture du nombre (**fgets**), incrémentation d'une unité du nombre (++), positionnement du pointeur courant du fichier en première ligne (**fseek 0**) et réécriture du nombre (**fputs**) puis fermeture (**fclose**).

On affiche ensuite le nombre de visite sur l'écran. Donc on ouvre le fichier en mode lecteur seule(r), on lit le contenu du fichier (fgets) on met ce contenu dans une variable, on ferme le fichier et on affiche le contenu de la variable.

Partie 2 : MySQL

1. Introduction aux bases de données

Une base de données (son abréviation est BD, en anglais DB, database) est une entité dans laquelle il est possible de stocker des données de façon structurée et avec le moins de redondance possible. Ces données doivent pouvoir être utilisées par des programmes, par des utilisateurs différents.

Une base de données peut être locale, c'est-à-dire utilisable sur une machine par un utilisateur, ou bien répartie, c'est-à-dire que les informations sont stockées sur des machines distantes et accessibles par réseau.

L'avantage majeur de l'utilisation de bases de données est la possibilité de pouvoir être accessibles par plusieurs utilisateurs simultanément.

Les informations d'une base de données sont classées dans une structure appelée ***table.*** Une table est constitué d'un ensemble de colonnes (appelée attribut).Un attribut est définit par son nom et le type de l'information qu'il va contenir. Les lignes de la table sont appelés enregistrement ou tuplet.

Exemple : Cette table à 3 attributs (numéro, nom, prénom) et contiens 2 enregistrements

Numéro	Nom	Prénom
1	Touré	Souleymane
2	Coulibaly	Amadou

Afin de pouvoir contrôler les données ainsi que les utilisateurs, le besoin d'un système de gestion s'est vite fait ressentir. La gestion de la base de données se fait grâce à un système appelé SGBD (système de gestion de bases de données) ou en anglais DBMS (Database management system).

Le SGBD est un ensemble de services (applications logicielles) permettant de gérer les bases de données, c'est-à-dire :
• permettre l'accès aux données de façon simple
• autoriser un accès aux informations à de multiples utilisateurs
• manipuler les données présentes dans la base de donnés (insertion, suppression…)

2. Présentation de MySQL

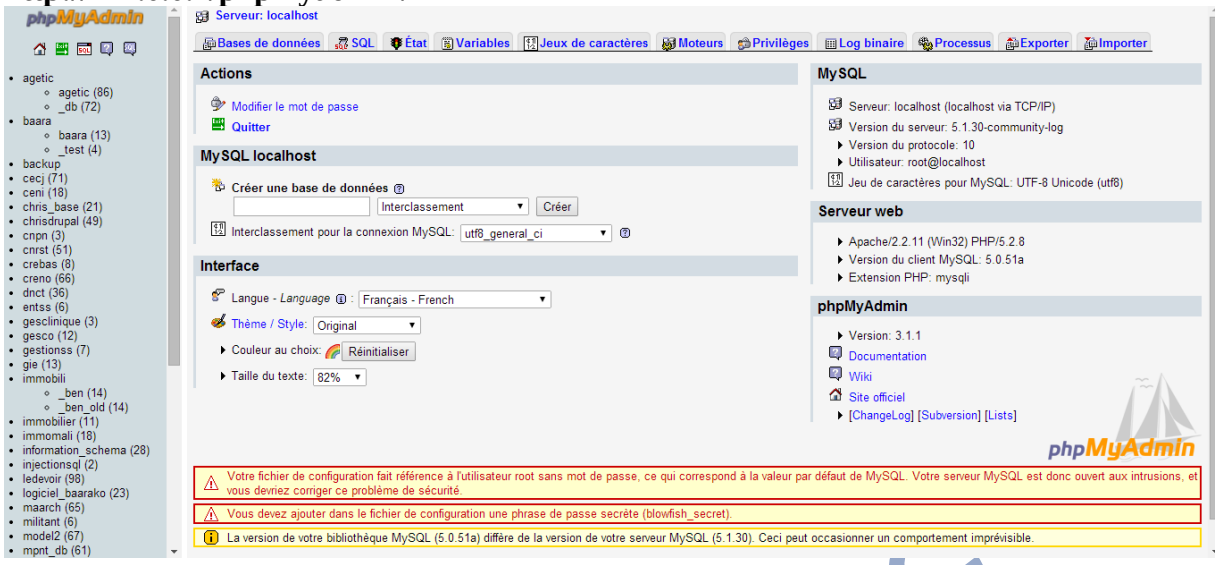
MySQL est un Système de Gestion de Bases de Données (SGBD) fonctionnant sous Linux et Windows.

Avec MySQL vous pouvez créer plusieurs bases de données sur un serveur.
MySQL est également installé et configuré automatiquement avec EasyPHP.

3. Présentation de l'outil d'administration de MySQL : PHPMyAdmin

Il existe un outil libre et gratuit développé par la communauté des programmeurs libres : ***phpMyAdmin*** qui permet l'administration aisée des bases de données MySQL avec PHP

Pour accéder à PHPmyAdmin, écrivez l'URL suivante dans Internet Explorer
http://127.0.0.1/phpmyadmin/



4. Fonctions PHP permettant de se connecter à MYSQL

Pour se connecter à une base depuis PHP, il faut spécifier un nom de serveur, un nom d'utilisateur, un mot de passe et un nom de base

Aucune connexion n'est possible sans authentification auprès du serveur de base de données. Les actions possibles de l'utilisateur sur la base à laquelle il se connecte dépendent des droits qui lui auront été fournis par l'administrateur de la base de données.

Les fonctions de connexion :
mysql_connect ($server,$user,$password) : permet de se connecter au serveur **$server** en tant qu'utilisateur **$user** avec le mot de passe **$password**, retourne l'identifiant de connexion si succès, FALSE sinon
mysql_select_db($base[,$id]) : permet de choisir la base **$base**, retourne TRUE en cas de succès, sinon FALSE
mysql_close([$id]) : permet de fermer la connexion
mysql_pconnect : idem que **mysql_connect** sauf que la connection est persistante, il n'y a donc pas besoin de rouvrir la connexion à chaque script qui travaille sur la même base.

Les identifiants de connexion ne sont pas nécessaires si on ne se connecte qu'à une seule base à la fois, ils permettent seulement de lever toute ambiguïté en cas de connexions multiples.

Exemple :

```
if( $id = mysql_connect("localhost", "utilisateur ", "motdepasse") ) {
        if(mysql_select_db("mabase ") ) {
                echo " Succès : Connecté au serveur ’’;
                /* code du script … */
        } else {
                die(‘’Echec de connexion à la base.’’);
        }
```

```
        mysql_close($id);
} else {
        die("Echec de connexion au serveur de base de données. ");
}
```

La fonction ***die()*** permet d'afficher un message d'erreur tout en arrêtant le script.

.

5. Travaux Pratiques N°5(TP5)

Réaliser une page « ***TP5.php*** » permettant de tester une connexion à votre base donnée. Cette page utilise (donc inclus) une page appelé « ***connexion.php ».*** La page « ***connexion.php »*** ne contient que la déclaration des variables : $machine= 'localhost',$utilisateur='le nom d'user',$mot_de_passe='motdepasse' et $db='base'. La page « ***TP5.php*** » après avoir inclus « ***connexion.php »*** utilises ses variable pour se connecter à la base et afficher un message selon que la connexion ai réussi ou pas.

6. Fonctions PHP permettant d'exécuter et d'exploiter le résultat d'une requête

En PHP toutes opération de type Insertion, affichage, modification, suppression se fait au moyen des requêtes SQL.

Pour envoyer une requête à une base de donnée, il existe la fonction : **mysql_query($str)** qui prend pour paramètre une chaîne de caractères qui contient la requête écrite en SQL et retourne un identificateur de résultat ou FALSE si échec.

Les requêtes les plus couramment utilisées sont : **CREATE** (création d'une table), **SELECT** (sélection), **INSERT** (insertion), **UPDATE** (mise à jour des données), **DELETE** (suppression), **ALTER** (modification d'une table), etc.

mysql_fetch_row($result) : retourne une ligne de résultat sous la forme d'un tableau. Les éléments du tableau étant les valeurs des attributs de la ligne. Retourne FALSE s'il n'y a plus aucune ligne.

Quelques fonctions supplémentaires très utiles :
mysql_free_result($result) : efface de la mémoire du serveur les lignes de résultat de la requête identifiées par **$requet**. Très utile pour améliorer les performances du serveur.
mysql_insert_id([$id]) : retourne l'identifiant d'un attribut clé primaire AUTO_INCREMENT de la dernière insertion.
mysql_num_fields($result) : retourne le nombre d'attributs du résultats.
mysql_num_rows($result) : retourne le nombre de lignes du résultats. Et ainsi permet de remplacer le **while** par un **for**.

Penser à bien tester la valeur de retour des fonctions (**mysql_query** et les autres) afin de détecter toute erreur et d'éviter de polluer votre page avec des *Warnings*.

7. Etudes des différentes requêtes: Insertion, affichage, modification, suppression

- **Insertion :** Supposons qu'on a une table appelé ***contact***, cette table a les attributs suivant : numéro, nom, prénom, Email. L'insertion d'un nouvel enregistrement dans cette table se fait au moyen de la requête suivante :

INSERT INTO contact **VALUES**
("1","Toure","Souleymane","souleymanetoure@yahoo.fr")

On remarque tout de suite que la syntaxe pour une insertion est relativement simple.
En effet, étudions ce code :
On insère où ? --> Dans la table contact
On insère quoi ? --> La valeur des différents attributs, c'est-à-dire une première valeur(1) qui correspond à l'attribut numéro, puis on insère la valeur Toure pour l'attribut nom, la valeur Souleymane pour l'attribut prénom et enfin souleymanetoure@yahoo.fr pour l'attribut Email

Exemple : Code d'insertion d'un nouveau contact

```
< ?php

if( $id = mysql_connect("localhost","utilisateur","motdepasse") ) {
if(mysql_select_db("mabase") ) {

$requete INSERT INTO contact VALUES ('1','Toure', 'Souleymane',
'souleymanetoure@yahoo.fr')

    $resultat=mysql_querry ($requete);
    If ($resultat) {
                    Echo  " le nouveau contact est inséré dans la base ";
                }
     Else die (" Echec lors de l'exécution de la requête. ");

} else {
                die("Echec de connexion à la base.");
        }
mysql_close($id);
} else {
        die ("Echec de connexion au serveur de base de données. ");
}

?>
```

- **Travaux Pratiques N°5(TP5) : Formulaire d'insertion dans une table**

Réaliser une page « tp5.html » contenant un formulaire dont les champs représentent exactement les attributs de la table *contact.* Ce formulaire sera renvoyé à une page « tp5.php » et cette dernière va se charger d'insérer les informations du formulaire dans la table *contact*.

- **Affichage :** supposons qu'on veut afficher la liste des contacts. Donc on affiche seulement tout le contenu de la table

SELECT numero, nom, prenom, email **FROM** contact

On sélectionne les attributs à afficher (on peut utiliser le signe * si on veut sélectionner tous les attributs) puis on détermine la table dans laquelle sélectionner. On peu ajouter une condition de sélection des enregistrements(WHERE) à afficher mais ici comme on veut afficher la totalité de la table pas besoin de condition.

Exemple:

```
$requet = " SELECT * FROM contact";
If ($result = mysql_query($requet)) {
        $ligne = mysql_fetch_row($result);        // pointe sur le premier enregistrement
      while ($ligne) {
            $num = $ligne[0];
            $nom = $ligne[1];
            $prenom = $ligne[2];
            $email = $ligne[3];
            echo "$num - $nom, $prenom <br />";
       $ligne = mysql_fetch_row($result) ;          // aller au suivant
      }
} else {
      echo  " Erreur de requête de base de données. ";
}
```

Ici, on accède aux valeurs de la ligne par leur indice dans le tableau.

- **Travaux Pratiques N°6(TP6) : Script d'affichage des enregistrements dans un tableau**

Reprenez le même exemple mais cette fois-ci les enregistrements serons afficher dans un tableau que vous allez créer.

- **Modification :** Dans une base de données on peu souvent avoir besoin de modifier les enregistrements d'une table donnée. Supposons que dans l'enregistrement inséré précédemment Touré Souleymane veux changer son Email par exemple.

UPDATE contact **SET** email= "stoure@agetic.gov.ml" **WHERE** nom="Toure"

On remarque tout de suite que la syntaxe pour une modification est vraiment logique.
En effet, étudions ce code pas à pas :

On modifie quelle table : **contact**
Quel(s) attribut(s) on va modifier : dans ce cas un seul attribut ***email***
Et enfin on modifie l'email pour quel enregistrement : celui dont le nom égal à ***Toure***.

Il est à noter que la modification concerne ici tous les enregistrements pour lesquels le nom est égal à Toure. On peut modifier plusieurs attributs à la fois (séparé par des virgules). La clause WHERE peut aussi prendre plusieurs conditions.

- **Suppression :** Après avoir vu l'affichage des données provenant d'une base de données, l'insertion et la modification de ces mêmes données, voyons maintenant la **dernière opération fondamentale** concernant ces base de données : la **suppression** des enregistrements

DELETE FROM *contact* **WHERE** nom=’Toure’

Trop simple:
Supprimer le(les) enregistrement(s) de la table contact pour lequel (lesquels) l’attribut nom a pour valeur Toure.

8. Travaux Pratiques N°7(TP7)

Créer une nouvelle base de donnée et dans cette dernière créer une table *session* (motdepasse : varchar, login : varchar, nom,prenom : varchar).les attribut motdepasse et login représentent la clé primaire de cette table.

Créer une page « index.html » contenant les liens HyperText suivants : Connexion, Nouveau. Le lien *connexion* fait appel à une page « connexion.html » contenant un formulaire de 2 champs (login, mot de passe) qui sera envoyé à la page (« login.php ») qui va se charger de récupérer les informations du formulaire, de tester les valeurs, et enfin de vérifier si ces valeurs existent dans la table session. Si oui un message de bienvenue sera afficher et 2 liens HyperText *modifier mon compte* et *supprimer mon compte* (qui permettent bien sur de mettre à jour ses information). Sinon un message d’erreur lui sera affiché.
Le lien nouveau fait appel à la page « nouveau.php » contenant un formulaire d’insertion d’un nouvel enregistrement dans la table *session*.