Université Abdelmalek Essaâdi
Faculté des Sciences
Tétouan – Maroc

جامعة عبد المالك السعدي
كلية العلوم
تطوان

Polycopié

Langage PHP
SMI
Semestre 3

Préparé par
Youssef Zaz

2018-2019

Le langage PHP

PHP (Hypertext Preprocessor) est un langage de scripts généraliste, Open Source, et spécialement conçu pour le développement d'applications web. Il peut être intégré facilement à vos pages HTML. Le code PHP inséré dans les pages WEB est repéré par un serveur WEB (s'il est muni de l'extension PHP).
Grâce à des portions de code PHP que vous allez insérer dans vos pages WEB, PHP vous permettra d'écrire rapidement des pages WEB à contenus dynamiques. Surtout s'il est couplé avec un serveur de bases de données relationnelles tel que MySQL.

Le fonctionnement

Il est à noter une différence avec les autres scripts CGI écrits dans d'autres langages tels que le Perl ou le C : au lieu d'écrire un programme avec de nombreuses lignes de commandes afin de générer une page HTML, avec PHP, vous écrivez une page HTML avec du code PHP inclus à l'intérieur afin de réaliser une action précise.
Le code PHP est inclus entre une balise de début et une balise de fin qui permettent au serveur web de passer en "mode PHP".
La connaissance du code HTML est primordiale pour travailler en PHP.
Il faut également savoir que lorsque vous insérez le moindre petit bout de code PHP dans une page HTML, vous devrez changer l'extension de ce fichier en .php.
Il est tout à fait possible de mélanger, au sein d'une même page WEB, des instructions HTML et des instructions PHP.
Seulement, pour que le serveur qui vous héberge puisse repérer les portions de code en PHP, il suffit simplement de lui indiquer le début ainsi que la fin du code PHP. Ces marques qui délimitent la portion de code, s'appellent des balises :
- on utilisera la balise **<?php** pour marquer le début d'une portion de code PHP
- on utilisera la balise **?>** pour marquer la fin d'une portion de code PHP

Les instructions du code PHP se placeront naturellement entre ces deux balises.

exemple1

```
1. <html>
2. <head><title>Test</title></head>
3.  <body>
4. <p>un bout de code en HTML</p>
5. <?php
6. echo 'Mon premier script en PHP';
7. ?>
8. </body>
9. </html>
```

exemple2

```
1. <?php
2. // ceci est un commentaire sur une seule ligne
3.  /* ceci est
4. un commentaire
5. sur plusieurs lignes */
6. ?>
```

Ce qui distingue le PHP des langages de script comme le Javascript est que le code est exécuté sur le serveur.
Si vous avez un script similaire sur votre serveur, le client ne reçoit que le résultat du script, sans aucun moyen d'avoir accès au code qui a produit ce résultat. Vous pouvez configurer votre serveur web afin qu'il analyse tous vos fichiers HTML comme des fichiers PHP.

Ainsi, il n'y a aucun moyen de distinguer les pages qui sont produites dynamiquement des pages statiques.

Les besoins pour bien commencer

Pour travailler en php, il vous faut deux choses essentielles :
- un éditeur de texte pour écrire vos portions de code en PHP tel que notepad++
- un environnement de développement afin de tester ces portions de code.

Pour l'environnement de développement, plusieurs packages sont disponibles sur Internet :

Windows: Xampp - Wamp - EasyPHP
Linux : Lamp
Apple: Mamp

Déclarer des variables

En PHP, les variables sont représentées par une chaîne de caractères, ayant toujours comme premier caractère, le caractère dollar ($).
Les variables peuvent avoir n'importe quelle lettre en deuxième caractère du moment qu'il ne s'agit pas d'un chiffre.
De plus, on ne peut pas mettre d'espace dans le nom d'une variable.

Puis, pour assigner une valeur à une variable, on utilise l'opérateur =, tout en prenant soin de toujours placer la variable qui reçoit le résultat d'une opération à gauche du signe =.

exemple1

```
1. <?php
2. $nom = "FSTet";
3. // $nom contient alors la chaîne de caractères FSTet.
4. $mon_chiffre = 12;
5. // $mon_chiffre contient la valeur numérique 12.
6. $5toto = "test";
7. // Cette déclaration n'est pas valide car le nom de la variable commence par un chiffre
8. ?>
```

Déclaration des variables de type tableau (array).

Imaginons un classeur d'écolier (ce sera notre tableau) contenant différentes feuilles (qui seront les indices du tableau).
Imaginons également que ces feuilles soient numérotées, et chaque feuille contienne un texte particulier.
Dés lors, on peut chercher le contenu d'une feuille de ce classeur grâce à son numéro (on cherche donc l'information contenu dans le classeur à la page numéro x).

En informatique, un tableau, c'est exactement la même chose que notre classeur. Il s'agit d'une variable contenant différentes informations (les textes) et ces informations sont classées suivant le numéro de l'indice (c'est à dire le numéro de la feuille).

Par exemple, supposons que l'on a la variable $fruit de type ***array***.

On pourrait alors avoir le code suivant :

exemple2

```
1. <?php
2. $fruit    = Array();
3. $fruit[0] = "fraise";
4. $fruit[1] = "banane";
5. $fruit[2] = "abricot";
6. ?>
```

En reprenant l'exemple du classeur, c'est comme si nous avions un classeur de nom fruit, ayant 3 pages :
- sur la page 0, on aurait l'information fraise
- sur la page 1, on aurait l'information banane
- sur la page 2, on aurait l'information abricot

Nous venons, dans ce bout de code, de déclarer une variable de type array qui comporte 3 éléments (les pages).

Nous aurions eu le même résultat en exécutant le bout de code suivant :

exemple3

```
1. <?php
2. $fruit   = array();
3. $fruit[] = "fraise";
4. $fruit[] = "banane";
5. $fruit[] = "abricot";
6. ?>
```

En revanche, cette syntaxe est moins lisible, vu que souvent, on n'arrive plus vraiment à savoir à quelle page se trouve l'information recherchée.

- Au lieu d'utiliser des chiffres pour les indices, nous pouvons très bien utiliser des chaînes de caractères.

exemple4

```
1. <?php
2. $fruit = array();
3. $fruit['le_meilleur'] = "fraise";
4. $fruit['le_prefere_de_Hamid] = "banane";
5. $fruit['mon_prefere'] = "abricot";
6. ?>
```

Dans ce cas, il faut utiliser pour chaque indice du tableau, une chaîne de caractère unique.

Afficher le contenu des variables

la commande echo() permet d'afficher le contenu des variables.

exemple1

```
1. <?php
2. $nom = "FS Tétouan";
3. echo 'Bonjour ';
4. echo $nom;
5. echo ' !';
6. ?>
```

Ce qui affichera à l'écran :

Bonjour FS Tétouan !

exemple2

```
1. <?php
2. $date_du_jour = date ("d-m-Y");
3. $heure_courante = date ("H:i");
4. echo 'Nous sommes le : ';
5. echo $date_du_jour;
6. echo ' Et il est : ';
7. echo $heure_courante;
8. ?>
```

Ce qui affichera à l'écran :

Nous sommes le 02-01-2017 Et il est 09:10

Voici la liste des paramètres possibles pour la fonction date() :

- a : "am" (matin) ou "pm" (après-midi)
- A : "AM" (matin) ou "PM" (après-midi)
- d : Jour du mois, sur deux chiffres (éventuellement avec un zéros) : "01" à "31"
- D : Jour de la semaine, en trois lettres (et en anglais) : par exemple "Fri" (pour Vendredi)
- F : Mois, textuel, version longue; en anglais, i.e. "January" (pour Janvier)
- h : Heure, au format 12h, "01" à "12"
- H : heure, au format 24h, "00" à "23"
- g : Heure, au format 12h sans les zéros initiaux, "1" à "12"
- G : Heure, au format 24h sans les zéros initiaux, "0" à "23"
- i : Minutes; "00" à "59"
- j : Jour du mois sans les zéros initiaux: "1" à "31"
- l : Jour de la semaine, textuel, version longue; en anglais, i.e. "Friday" (pour Vendredi)
- L : Booléen pour savoir si l'année est bissextile ("1") ou pas ("0")
- m : Mois; i.e. "01" à "12"
- n : Mois sans les zéros initiaux; i.e. "1" à "12"
- M : Mois, en trois lettres (et en anglais) : par exemple "Jan" (pour Janvier)
- s : Secondes; i.e. "00" à "59"
- S : Suffixe ordinal d'un nom

Les variables prédéfinies

PHP propose une série de variables qui sont déjà présentes dans le langage sans que vous n'ayez à les déclarer. Ces variables s'écrivent toujours en majuscules et nous fournissent divers renseignements.

Voici quelques variables d'environnement:

Variable	Description
$_SERVER['DOCUMENT_ROOT']	Racine du serveur
$_SERVER['HTTP_ACCEPT_LANGUAGE']	Langage accepté par le navigateur
$_SERVER['HTTP_HOST']	Nom de domaine du serveur
$_SERVER['REMOTE_ADDR']	Adresse IP du client
$_SERVER['REMOTE_PORT']	Port de la requête HTTP
$_SERVER['SERVER_ADDR']	Adresse IP du serveur
$_SERVER['SERVER_ADMIN']	Adresse de l'administrateur du serveur
$_SERVER['SERVER_NAME']	Nom local du serveur

Ces variables peuvent être utilisées n'importe quand dans vos scripts.
Voici un exemple où vous pouvez afficher l'adresse IP de la personne qui se connecte sur votre site :

exemple1

```
1. <?php
2. echo 'Votre adresse IP est : '.$_SERVER['REMOTE_ADDR'];
3. ?>
```

Ce qui affichera à l'écran :

Votre adresse IP est : 192.168.1.40

Concaténer deux chaînes

Prenant l'exemple suivant :
exemple1

```
1. <?php
2. $nom = "FS";
3. echo 'Bonjour ';
4. echo $nom;
5. echo ' !';
6. ?>
```

➔

```
1. <?php
2. $nom = "FS";
3. echo 'Bonjour '.$nom.' !';
4. ?>
```

Ce qui affichera à l'écran : Bonjour FS !

On affiche en fait la chaîne Bonjour concaténée avec le contenu de la variable $nom, soit FS, également concatène avec la chaîne !, ce qui au final, se résume par l affichage de la chaîne Bonjour FS.

VI - Les structures de contrôles

Les structures de contrôles permettent de faire des tests entre les variables et d'exécuter diverses boucles.
Les principales structures de contrôles :

Instruction	Signification
if	Si
else	Sinon
elseif	Sinon si
switch	Selon
for	Pour chaque (boucle)
while	Tant que (boucle)
==	Strictement égal
!=	Différent
<	Strictement inférieur
>	Strictement supérieur
<=	Inférieur ou égal
>=	Supérieur ou égal
and ou &&	ET logique
or ou ||	OU logique

Les structures de contrôles.

if, else, elesif

exemple1:

```
1. <?php
2. $nombre = 11;   // on initialise la variable à 11
3. if ($nombre >= 0 && $nombre < 10) {     // on teste si la valeur est comprise entre 0 et 9
4.    echo $nombre.' est compris entre 0 et 9'  ;}
5. elseif ($nombre >= 10 && $nombre < 20){// on teste si la valeur est comprise entre 10 et 19
6.    echo $nombre.' est compris entre 10 et 19';}
7. else {        // si les deux tests précédents n'ont pas aboutis, alors on tombe dans ce cas
8.    echo $nombre.' est plus grand que 19'     ;}
9. ?>
```

A l'affichage on aura :

11 est compris entre 10 et 19

switch

Le switch représente exactement la même chose qu'une succession d'un if et de plusieurs elseif. En revanche, utiliser un switch à un certain avantage comparé à un if et à plusieurs elesif, c'est que sa structure est beaucoup moins lourde et nettement plus agréable à lire.

exemple2

```
1. <?php
2. $prenom = "ALI";
3.  switch ($nom) {
4.    case 'HAMID' :
5.       echo 'Votre prénom est Hamid.';     break;
6.    case 'ADIL' :
7.       echo 'Votre prénom est Adil.';      break;
8.    case 'ALI' :
9.       echo 'Votre prénom est Ali.';       break;
10.   default :
11.       echo 'Je ne sais pas qui vous êtes !!!';
12. }
13. ?>
```

Dans notre cas, vu que $prenom contient la chaîne de caractère ALI, on verra alors s'afficher à l'écran la phrase suivante :

Votre Prénom est Ali.

exemple3

```
1. <?php
2. $prenom = "ALI";
3. if     ($prenom == "HAMID"){echo 'Votre prénom est Hamid.'; }
4. elseif ($prenom == "ADIL") {echo 'Votre prénom est Adil.';  }
5. elseif ($prenom == "ALI")  {echo 'Votre prénom est Ali.';   }
6. else   {echo 'Je ne sais pas qui vous êtes !!!';            }
7. ?>
```

for

La structure de contrôle for nous permet d'écrire des boucles.

exemple4

```
1. <?php
2. $chiffre = 5;
3. for ($i=0; $i < $chiffre; $i++)
4. { echo 'Notre chiffre est différent de '.$i.'<br />';  }
5.  echo 'Notre chiffre est égal à '.$i;
6. ?>
```

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

while

Reprenons l'exemple précédent, et écrivons le à l'aide de la boucle while, on a :

exemple5

```
1. <?php
2. $chiffre = 5;  $i = 0;
3. while ($i < $chiffre) {
4.       echo 'Notre chiffre est différent de '.$i.'<br />';
5.       $i = $i + 1;  }
6.  echo 'Notre chiffre est égal à '.$i;
7. ?>
```

Ce qui affichera à l'écran exactement la même chose que ce qu'affiche le code que l'on a utilise pour la boucle for.

Les fonctions utilisateurs

En PHP, il y a une liste assez impressionnante de fonctions mises à votre disposition. En revanche, vous aussi, vous pouvez très bien écrire vos propres fonctions.

Nous allons donc écrire une fonction qui va nous permettre d'écrire un texte en gras, tout en spécifiant la couleur de ce texte, ainsi que sa taille.

On a alors le code suivant :

exemple1

```
1. <?php
2. function affichage_texte ($taille, $couleur, $texte) {
3. echo '<font size = "'.$taille.'" color='.$couleur.'">'.$texte.'</font>';
4. }
5. ?>
```

Nous plaçons ce code dans un fichier nommé fonctions.php.

Soit ensuite le code du fichier index.php :

exemple2

```
1. <?php
2. // on inclut le code de fonctions.php, donc le code de notre fonction
3. include ('fonctions.php');
4.  // on affiche un texte
5. affichage_texte ("2", "red", "Mon texte");
6. ?>
```

Ce qui affichera à l'écran :

Mon texte

Lire et écrire dans un fichier texte

Tentons maintenant de lire et d'écrire dans un fichier texte, fichier se trouvant sur votre serveur FTP. 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 PHP.

Supposons que ce fichier texte contienne la ligne suivante :

"Salut à tous :)".

Soit alors, le code PHP suivant :

exemple1

```
1. <?php
2.  // ouverture du fichier donnees.txt en lecture seule
3. $fp = fopen ("donnees.txt", "r");
4.  // on lit le contenu du fichier avec la fonction fgets() et l'on place le contenu du fichier
5.  // dans la variable $contenu_fichier (le paramètre 255 correspond au nombre de caractères à lire
6. $contenu_fichier = fgets ($fp, 255);
7.  // on referme le fichier donnees.txt
8. fclose ($fp);
9.  // on affiche le contenu du fichier donnees.txt
10.  echo 'Notre fichier contient : '.$contenu_fichier;
11. ?>
```

Ce qui affichera à l'écran :

Notre fichier contient : Salut à tous :)

Les paramètres possibles de la fonction fopen() sont:
- **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 file. 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.

Un exemple concret : un mini compteur du nombre de visites.

Tachons maintenant de voir un exemple concret de lecture et d'écriture dans un fichier texte. En effet, nous allons réaliser un mini compteur de visites fait grâce à PHP et aux lecture/écriture dans un fichier texte.

Tout d'abord vous allez créer un fichier compteur.txt que vous allez placer dans le même répertoire que le script qui va suivre. Placez le chiffre "0" dans ce fichier.

Soit alors le bout de code PHP suivant :

exemple3

```
1. <?php
2. $fp = fopen ("compteur.txt", "r+");
3. $nb_visites = fgets ($fp, 11);
4. // on augmente le nombre de visites de 1.
5. $nb_visites = $nb_visites + 1;
6. // on se positionne au tout début de notre fichier.
7. fseek ($fp, 0);
8. // on écrit dans le fichier la nouvelle valeur correspondant au nombre de visites.
9. fputs ($fp, $nb_visites);
10.     fclose ($fp);
11.     echo 'Ce site compte '.$nb_visites.' visiteurs !';
12. ?>
```

Récupérer les données des formulaires

Un formulaire permet aux visiteurs de soumettre des informations, que ce soit un nom, un prénom, un chiffre, etc...

Prenons le code suivant :

exemple1

```
1. <html>
2. <head><title>Ma page de test</title></head>
3. <body>
4. <form action = "traitement.php" method="post">
5.     Votre nom      : <input type = "text"   name  ="nom"     > <br />
6.     Votre fonction : <input type = "text"   name  ="fonction"> <br />
7.                      <input type = "submit" value ="Envoyer" >
8. </form>
9. </body>
10. </html>
```

Puis, 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 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).

Prenons ensuite le code suivant pour la page traitement.php :

exemple2

```
1. <html>
2. <head><title>Ma page de traitement</title></head>
3. <body>
4. <?php
5. // on teste la déclaration de nos variables
6. if (isset($_POST['nom']) && isset($_POST['fonction'])) {
7. // on affiche nos résultats
8. echo 'Votre nom est '.$_POST['nom'].' et votre fonction est
   '.$_POST['fonction'];
9. }
10.     ?>
11.     </body>
12. </html>
```

En supposant que l'on écrive "Ali" dans le champ "Votre nom" et "Webmaster" dans le champ "Votre fonction", on verra alors s'afficher à l'écran :

Votre nom est Ali et votre fonction est Webmaster

NB : dans le cas où le formulaire utilise une méthode get, nous utilisons la variable tableau $_GET.

Le cas des formulaires munis d'un champ de type file (formulaire permettant le téléchargement de fichiers sur votre site).
On a le formulaire suivant :

exemple3

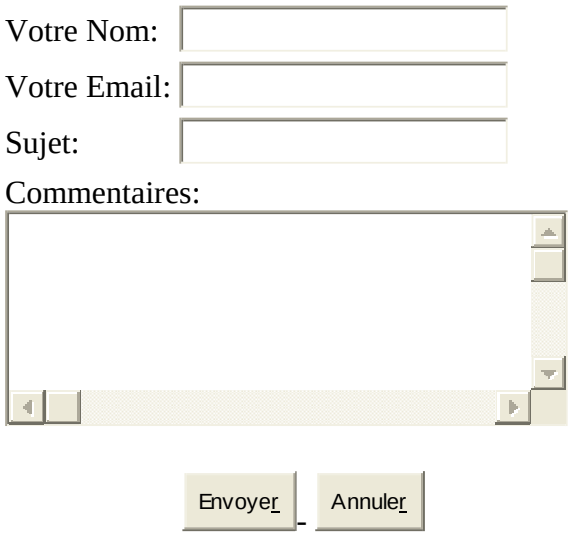
```
1. <html>
2. <head><title>Ma page de test</title></head>
3. <body>
4. <form action = "traitement.php" method="post" enctype="multipart/form-data">
5. Votre fichier : <input type = "file" name = "mon_fichier"><br />
6. <input type = "hidden" name="MAX_FILE_SIZE" value="20000">
7. <input type = "submit" value = "Envoyer">
8. </form>
9. </body>
10. </html>
```

Pour récupérer votre fichier, vous avez à votre disposition le tableau $_FILES qui aura plusieurs entrées :
- $_FILES['mon_fichier']['tmp_name'] : le nom temporaire du fichier sur le serveur
- $_FILES['mon_fichier']['name'] : le nom original du fichier sur la machine cliente
- $_FILES['mon_fichier']['type'] : le type MIME du fichier
- $_FILES['mon_fichier']['size'] : la taille du fichier

Naturellement, vous pourrez utiliser ces valeurs pour tester votre fichier.
S'il correspond à vos attentes, vous pourrez finaliser votre téléchargement à l'aides des fonctions copy ou move_uploaded_file (afin de copier le fichier téléchargé sur le disque dur du serveur).

Envoyer un email avec php

Le principe du **form** mail est de permettre à vos visiteurs de remplir un formulaire et ensuite de vous envoyer un email avec ce qui a été saisi. Pour faire cela nous allons utiliser PHP (votre hébergeur doit offrir cette possibilité ou qu'il faut installer un serveur web http avec le plugin php).
Exemple :



Pour cela il faut insérer dans une page le code html suivant:

```
<div align=center>
<form method=POST action=formmail.php >
<input type=hidden name=subject value=formmail>
<table>
<tr><td>Votre Nom:  </td><td><input type=text name=realname size=30></td></tr>
<tr><td>Votre Email:</td><td><input type=text name=email size=30>   </td></tr>
<tr><td>Sujet:      </td><td><input type=text name=title size=30>   </td></tr>
<tr><td colspan=2>Commentaires:<br><textarea COLS=50 ROWS=6 name=comments></textarea>
</td></tr>
</table>
<br> <input type=submit value=Envoyer> -  <input type=reset value=Annuler>
</form>
</div>
```

La page formmail.php est la suivante:

```
<?php
$TO = "email1@mail.com";
$h  = "From:email2@mail.com";
$message = " Message:
".$_POST['realname']."
".$_POST['email']."
".$_POST['comments'];;
$subject="Le commentaire du site";
mail($_POST['email'], $subject, $message, $h);
Header("Location: message.html");
?>
```

destinataire

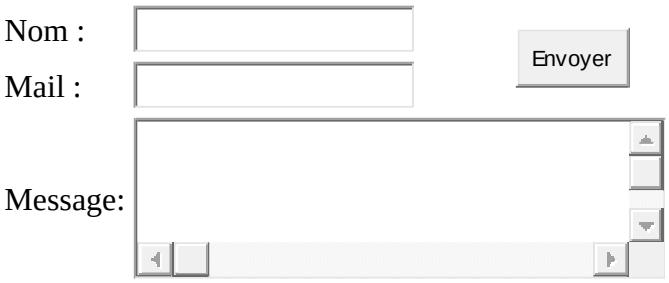
expediteur

Pour que cela fonctionne il faut modifier la variable $TO, et mettre votre adresse email entre les "". Une fois l'email envoyé, le visiteur sera redirigé vers une autre page, cette page doit être saisie juste après le http://, c'est a dire que site votre site est à l'adresse http://www.monsite.com/ et que vous avez créé une page merci.html, vous devrez avoir:

Header("Location: http://www.monsite.com/merci.html");
Attention, n'oubliez pas l'espace après *Location*

Livre d'or

Exemple d'un livre d'or:



Vous obtiendrez cela en insérant le code suivant, dans une page **guestbook.php**

```
<form method=post action=writeguest.php
onSubmit='if ( this.nom.value.length < 2 ) { return false ;}' >
<table>
<tr><td>Nom :</td>   <td><input type=text name=nom size=25 ></td>
<td rowspan=2><input type=submit value=Envoyer></td></tr>
<tr><td>Mail :</td>  <td><input type=text name=mail size=25 ></td></tr>
<tr><td>Message:</td><td colspan=2>
        <textarea name=message rows=4 cols=47></textarea></td></tr>
</table>
</form>
<?php include("guestbook.doc"); ?>
```

Notez à la fin de la page la ligne *include* qui permet d'ajouter le contenu du livre d'or que nous stockerons dans le fichier guestbook.doc. Le formulaire appelle la page writeguest.php après validation par le visiteur. Cette page est la suivante:

```
<?
//Ouverture du fichier en écriture
$fp = fopen("guestbook.doc",a);
//On convertit les caracteres html
$nom = htmlspecialchars($_POST['nom']);
$mail = htmlspecialchars($_POST['mail']);
$message = stripslashes(nl2br(htmlentities($_POST['message'])));
$d = date ( "d/m/Y H:i:s" );
$page = "";
$lemail = "<a href=\"mailto:$mail\">$mail</a>";
$page .= "<b>$nom</b> (".$lemail.") - $d<br>$message<br><hr>\n";
//On rajoute le message
fwrite($fp,"$page" ,strlen("$page"));
//fermeture du fichier
fclose($fp);
//On affiche le message enregistré
echo "Merci $nom, nous avons enregistré: <br>";
echo "email : $mail <br> message : $message";
?>
<a href="guestbook.php">Retour au guestbook</a>
```

Vous avez donc deux pages, **guestbook.php** qui contient le formulaire et **writeguest.php** qui s'occupe du traitement et écrit le message dans le livre d'or. Il ne vous reste plus qu'à les transférer sur votre serveur chez votre hébergeur. Vous devez également transférer un fichier *guestbook.doc*, un fichier vide, dans le même répertoire que les deux fichiers précédents. Vous devez également donner les droits en écriture sur ce fichier à votre serveur web. Il existe de nombreuses méthodes pour cela en fonction des outils que vous utilisez ou des facilités offertes par votre hébergeur. Vous pouvez par exemple utiliser l'exploreur de fichiers de windows pour cela. Dans la barre d'adresse saisissez: ftp://login@ftp.monsite.com
Remplacez login par l'identifiant fournir par votre hébergeur et ftp.monsite.com par le nom du serveur ftp correspondant. L'explorer va vous demander votre mot de passe, saisissez le et validez. Ensuite avec le bouton droit de la souris cliquer sur le fichier *guestbook.doc* et sélectionnez *propriétés*. Cochez toutes les cases de la colonne *Ecriture* et faites *OK*.

Manipulation des bases de données MySQL avec le PHP

Les bases de données prennent aujourd'hui une proportion importante dans les sites WEB. En utilisant les bases de données, vous allez découvrir toutes les possibilités d'un environnement PHP / MySQL.

Prenons un exemple simple et concret : supposons qu'on désire développer une base de données contenant une liste de CD audio. Cette liste de CD sera en fait composée de tous les CD que possède chaque personne d'un groupe d'amis. Et ceci, afin de pouvoir se prêter mutuellement les différents CD, et de savoir exactement qui à quoi comme CD.

On suppose que le groupe d'amis est composé de 3 personnes :

- ALI
- Hamid
- Aziz

Chaque personne a un numéro de téléphone, et chaque personne possède un certain nombre de CD. On prendra aussi en considération le titre de l'album et le nom de l'interprète.

On aurait alors très bien pu obtenir (sous forme d'un tableau) la base de données suivante :

Propriétaire	N. tél	Auteur	Titre
ALI	06-61-85-20-54	Camilia	Amitié
ALI	06-61-85-20-54	Daft Punk	A la découverte
Hamid	06-61-74-26-01	Camilia	Amitié
Hamid	06-61-74-26-01	Télénor	Monde sans frontières
Aziz	06-63-01-59-36	Calairis	Bonjour

Notez bien que ce tableau, en termes de base de données, se nomme une table et que chaque ligne du tableau se nomme un enregistrement. La première ligne du tableau comporte les attributs de la table (Propriétaire, N. tél, Auteur et Titre sont les attributs de notre table).
Note : une base de données peut contenir plusieurs tables.
Faisons maintenant quelques interrogations sur cette base de données :
Qui possède un album de Camilia ?
>> réponse : ALI et Hamid
Quel est le numéro de téléphone de Aziz ?
>> réponse : 06-63-01-59-36
Quels sont les albums des Daft Punk disponibles dans la liste de CD ?
>> réponse : A la découverte (il n'y en a qu'un seul)
A première vue donc, le principe des bases de données est très facilement assimilable.

Il faut également savoir que dans la table d'une base de données, on ne peut pas avoir 2 enregistrements (donc 2 lignes du tableau) ayant les mêmes éléments.

Imaginons maintenant que Hamid change son numéro de portable.

Supposons que son nouveau numéro est 06-61-98-78-12 et qu'en plus il vienne de s'acheter un nouveau CD : Paradise de Bob Dinar.

On insère alors une nouvelle ligne dans notre table (un nouvel enregistrement), et l'on obtient donc :

Propriétaire	N. tél	Auteur	Titre
ALI	06-61-85-20-54	Camilia	Amitié
ALI	06-61-85-20-54	Daft Punk	A la découverte
Hamid	06-61-74-26-01	Camilia	Amitié

Hamid	06-61-74-26-01	Télénor	Monde sans frontières
Aziz	06-63-01-59-36	Calairis	Bonjour
Hamid	06-61-98-78-12	Bob Dinar	Paradise

Imaginons maintenant que j'interroge à nouveau ma base de données.

Quel est le numéro de téléphone de Hamid ?
>> réponse : 06-61-98-78-12 ou bien 06-61-74-26-01
Nous remarquons tout de suite qu'un problème majeur arrive : c'est à dire que Hamid possède deux numéros de téléphone alors qu'il ne devrait (en théorie) n'en posséder qu'un seul.

Pour remédier à ce problème, il faudrait, par exemple modifier tous les premiers enregistrements et ainsi mettre à jour le numéro de téléphone de Hamid.

Dans notre cas, cette solution n'est pas vraiment gênante, en revanche, lorsque la table comporte quelques centaines voir milliers d'enregistrements, c'est déjà beaucoup plus gênant.

En fait, ce problème survient généralement à cause d'une mauvaise conception de la base de données.

En effet, au lieu de créer une seule table contenant toutes les informations, nous aurions du créer deux tables :
- une contenant la liste des CD (Auteur et Titre)
- une contenant les informations des propriétaires des CD (Propriétaire et N. de tel)
Ensuite, il nous resterait a faire un lien entre les tables, nous permettant de savoir qui possède tel ou tel CD.

Mettons cette solution en pratique. On a alors notre table contenant la liste des propriétaires qui aura les attributs suivants :
- numéro du propriétaire
- nom du propriétaire
- numéro de téléphone du propriétaire

Et la table contenant la liste de CD, aura les attributs suivants :
- numéro du propriétaire du CD
- Auteur du CD
- Titre du CD
(Et c'est grâce au numéro du propriétaire que l'on fera la liaison entre les deux tables, l'opération est nommée une jointure)
On aura alors :

N. du propriétaire	Propriétaire	N. tél
1	ALI	06-61-85-20-54
2	Hamid	06-61-98-78-12
3	Aziz	06-63-01-59-36

N. du propriétaire	Auteur	Titre
1	Camilia	Amitié
1	Daft Punk	A la découverte
2	Camilia	Amitié
2	Télénor	Monde sans frontières
3	Calairis	Bonjour
2	Bob Dinar	Paradise

On remarque que si une personne change de numéro de téléphone, et bien nous avons qu'une seule modification à effectuer (vu que chaque numéro de téléphone n'apparaît qu'une fois dans toute la base).
Puis dans ce cas, nous remarquons également que c'est le N. de propriétaire qui effectue la liaison entre les deux tables (c'est à dire la jointure).
En conclusion :
Faites extrêmement attention au moment où vous créer les tables de votre base de données afin de ne pas se retrouver dans une situation où tout retour en arrière serait impossible : visualiser bien votre idée, écrivez sur papier ce dont vous avez réellement besoin pour votre base de données, et tentez au maximum d'éviter d'avoir des redondances dans vos tables.

Création des tables en SQL

Le SQL (Structured Query Langage) est un langage de requêtes qui nous permet de faire des interrogations (les requêtes) sur un SGBD (Système de Gestion de Base de Données).

Nous allons voir maintenant, comment utiliser le SQL pour développer cette base de données.

Tout le code donné dans ce document ne peut être interprété que par votre SGBD. Ce code ne pourra pas être inclus dans vos pages PHP.

En effet, vous devrez utiliser ce code dans votre PHPMyAdmin par exemple.
Reprenons l'exemple précédent, et on observe comment faire pour créer nos deux tables.
Créons tout d'abord la table liste_proprietaire :

```
CREATE TABLE liste_proprietaire (
numero INT(5) NOT NULL,
nom VARCHAR(20) NOT NULL,
telephone VARCHAR(14) NOT NULL
) TYPE=MyISAM;
```

Nous venons ici de créer notre table liste_proprietaire, table contenant trois attributs :
- numéro qui correspond à un nombre entier (INT) de 5 chiffres
- nom qui correspond à une suite de caractères (VARCHAR) de 20 caractères
- téléphone qui correspond à une suite de caractères (VARCHAR) de 14 caractères

Nous utilisons VARCHAR pour des chaînes de caractères pouvant mêler du texte et des nombres. Pour chacun de ces attributs, nous imposons que pour chaque enregistrement donné, aucun attribut ne peut être vide (on a mis un NOT NULL pour tous les attributs).

Le type MyISAM précise que nous avons affaire à une base de données de type MySQL.

Créons ensuite la table liste_disque :

```
CREATE TABLE liste_disque (
auteur VARCHAR(50) NOT NULL,
titre VARCHAR(50) NOT NULL,
numero INT(5) NOT NULL
) TYPE=MyISAM;
```

On peut utiliser AUTO_INCREMENT pour les attributs numero de la table liste_proprietaire. En effet, si on imagine la page WEB nous permettant d'insérer des membres dans la liste des propriétaires, on s'imagine mal qu'il faille préciser à chaque fois le numéro du nouveau propriétaire.

Afin de palier à ce léger désagrément, nous allons appliquer un extra à l'attribut de la première table, nous permettant de faire une incrémentation automatique à chaque insertion d'un nouveau propriétaire.

On aura alors la table liste_proprietaire définit comme ceci :

```
CREATE TABLE liste_proprietaire (
numero INT(5) NOT NULL AUTO_INCREMENT,
nom VARCHAR(20) NOT NULL,
telephone VARCHAR(14) NOT NULL,
PRIMARY KEY (numero)
) TYPE=MyISAM;
```

Remarquons alors la ligne PRIMARY KEY (numero) qui nous indique que la clé primaire de notre table est l'attribut numero. Ceci veut dire que l'on est capable d'identifier n'importe quel enregistrement de la table rien qu'à partir de l'attribut numero.

Passons en revue l'intégralité des types possibles pour les attributs d'une table SQL/
- TINYINT : entier de 0 à 255 (non signé)
- SMALLINT : entier de 0 à 65 535 (non signé)
- MEDIUMINT : entier de 0 à 16 777 215 (non signé)
- INT : entier de 0 à 4 294 967 295 (non signé)
- BIGINT : entier de 0 à 18 446 744 073 709 551 615 (non signé)
- DECIMAL : un nombre à virgule flottante (soit un nombre réel)
- DATE : une date allant du 1000-01-01 au 9999-12-31
- DATETIME : une date comportant une heure allant du 1000-01-01 00:00:00 au 9999-12-31 23:59:59
- TIMESTAMP : une date comportant une heure exprimée en secondes depuis le 1er janvier 1970 jusqu'à l'instant présent
- TIME : une mesure de l'heure qui va de -838:59:59 à 838:59:59
- YEAR : une année qui va de 1901 à 2155
- CHAR : une chaîne de caractères de taille fixée (de 1 à 255 caractères)
- VARCHAR : une chaîne de caractères de taille variable (de 1 à 255 caractères)
- TINYTEXT ou TINYBLOB : un objet BLOB ou TEXT ayant une longueur maximale de 255 caractères
- TEXT ou BLOB : un objet BLOB ou TEXT ayant une longueur maximale de 65 535 caractères
- MEDIUMTEXT ou MEDIUMBLOB : un objet BLOB ou TEXT ayant une longueur maximale de 16 777 215 caractères
- LONGTEXT ou LONGBLOB : un objet BLOB ou TEXT ayant une longueur maximale de 4 294 967 295 caractères

Remarque
Il est tout de fois possible de créer des tables directement à partir d'une page en PHP. Dans ce cas, pour créer les tables, il n'est absolument pas nécessaire de passer par votre PHPMyAdmin, chose que je déconseille fortement pour les débutants.
En clair, au début, utiliser PHPMyAdmin pour créer vos tables, puis, une fois que vous aurez bien compris le principe, vous pourrez alors créer vos tables directement à partir de vos scripts (ce qui est tout de même déconseillé).

Fonctions PHP pour MySQL

Nous venons de voir comment créer nos tables SQL à l'aide d'un PHPMyAdmin (par exemple). Avant de voir comment faire pour insérer, modifier, supprimer et afficher des enregistrements de notre base de données, il est bon de connaître les fonctions PHP permettant de manoeuvrer ces enregistrements.

Les fonctions PHP pour MySQL commence toujours par mysql_.

Voici donc la liste de ces fonctions (nous étudierons que les principales, regardez la documentation pour de plus amples connaissances) :

Fonction	**Signification**
mysqli_close	Ferme la connexion à une base de données
mysqli_connect	Etablit une connexion vers la base de données spécifiée dans les arguments
mysqli_error	Retourne la description textuelle d'une erreur générée par une action sur une base de données
mysqli_fetch_array	Retourne un tableau qui représente tous les enregistrements sélectionnés (un indice du tableau correspond à un attribut des enregistrements obtenus). Chaque appel récupère l'enregistrement suivant jusqu'à ce qu'il n'y en ait plus
mysqli_free_result	Libère la mémoire associé à la requête spécifiée
mysqli_num_rows	Retourne le nombre d'enregistrement dans un résultat
mysqli_query	Permet d'exécuter une requête SQL sur une base de données
mysqli_select_db	Sélectionne la base de données par défaut

Pour se connecter à une base de données, vous devez indiquer, sur toutes vos pages PHP où vous utilisez votre base ainsi que les différents paramètres de connexion.

Etudions le code suivant :
exemple1

```
1. <?php
2. $base = mysqli_connect ("mon_serveur", "login", "password", "ma_base_de_donnees");
3. ?>
```

La chaîne de caractères mon_serveur doit être remplacé par celle qui correspond au nom de votre serveur (en règle générale, il s'agit de *localhost* ; si ce n'est pas le cas, veuillez contacter votre hébergeur pour de plus amples informations).
login correspond à votre login pour accéder à votre base par défaut *root*.
password, votre mot de passe, par défaut, un champ vide.
Et ma_base_de_donnees correspond au nom de votre base de données.
Grâce à ce code nous allons donc pouvoir effectuer toutes nos requêtes SQL sur les tables de notre base de données créée.

Remarque
Ce code doit toujours être présent avant toute opération sur votre base de données (une seule fois par page suffit par contre).
Mon conseil : faites-vous un fichier connect_base.php.inc ou apparaîtra seulement ce morceau de code, et dans chaque page ou vous souhaitez utiliser votre base de données, vous n'aurez alors qu'à include() ce fichier de connexion.

Afficher les données de votre base

Une fois les tables de la base de données sont créées, nous allons pouvoir voir comment faire pour pouvoir interroger cette base de données, et par conséquent afficher les résultats sur vos pages WEB.

Avant de voir le code PHP pour faire ces interrogations, nous allons voir comment s'effectuent ces interrogations par le biais de requêtes SQL.

Et pour ce faire, nous allons nous baser sur l'exemple précèdent.

La table liste_proprietaire :

N. du propriétaire	Propriétaire	N. tél
1	ALI	06-61-85-20-54
2	Hamid	06-61-98-78-12
3	Aziz	06-63-01-59-36

La table liste_disque :

N. du propriétaire	Auteur	Titre
1	Camilia	Amitié
1	Daft Punk	A la découverte
2	Camilia	Amitié
2	Télénor	Monde sans frontières
3	Calairis	Bonjour
2	Bob Dinar	Paradise

Nous allons alors interroger la table pour connaître par exemple le numéro de téléphone de ALI. On aura alors:

```
SELECT telephone
FROM liste_proprietaire
WHERE nom="ALI";
```

Etudions ce code :

- on sélectionne l'attribut que l'on désire obtenir (ici l'attribut telephone qui correspond effectivement au numero de téléphone des propriétaires).

- on utilise la table liste_proprietaire pour faire notre sélection (en fait, on n'a besoin que de cette table pour faire notre sélection, on verra plus tard que la clause FROM peut contenir plusieurs tables, notamment dans le cas des jointures).

- on impose une condition, en effet, on veut le téléphone de qui ? De ALI, donc en écrivant WHERE nom="ALI", on impose au SGBD de ne sélectionner dans notre table que les enregistrements qui possèdent l'attribut nom qui est égal à ALI. En revanche, en n'écrivant pas cette ligne, on aura obtenu tous les numéros de téléphone de notre table (car dans ce cas, on n'aurait eu aucune condition quant au nom du propriétaire).

Etudions maintenant le cas où l'on effectue une sélection lorsque l'on doit effectuer une jointure entre deux tables.

Interrogeons alors notre base de données pour connaître les noms des propriétaires de l'album Amitié de Camilia.

On aura alors :

```
SELECT liste_proprietaire.proprietaire
FROM liste_proprietaire, liste_disque
WHERE liste_disque.auteur = "Camilia"
AND liste_disque.titre = "Amitié"
AND liste_proprietaire.numero = liste_disque.numero
ORDER BY liste_proprietaire.proprietaire ASC;
```

- on sélectionne l'attribut proprietaire (ce que l'on veut obtenir) tout en indiquant que cet attribut faire partie de la table liste_proprietaire. En revanche, ici, il n'est pas vraiment nécessaire de préciser que l'on sélectionne l'attribut proprietaire de la table liste_proprietaire puisque cet attribut de table n'apparaît que dans une seule table (en l'occurrence la table liste_proprietaire).

En effet, si l'attribut proprietaire avait existé dans plusieurs tables, nous aurions dû préciser de quelle table il s'agit (le SGBD n'aurait pas su de quelle table l'attribut dont nous parlons fait partie). Toutefois, je vous recommande chaudement (sauf dans les cas extrêmement simple, de toujours préciser au SGBD à quelle table appartient l'attribut que vous sélectionnez).

- On utilise les tables liste_proprietaire et liste_disque pour faire notre requête.
- On fait notre recherche sur la liste de disque en ne retenant que les disques dont l'auteur est Camilia.
- Puis on retient les disques dont le titre est Amitié.
- Ensuite (le plus important), on effectue une jointure entre les tables, en disant que l'attribut numero de la table liste_proprietaire correspond à l'attribut numero de la table liste_disque.
- Enfin, on impose au SGBD de nous fournir les résultats dans l'ordre alphabétique des noms de propriétaire (et ceci grâce à la clause ORDER BY).
- Pour organiser les résultats suivant l'ordre inverse de l'ordre alphabétique, on aurait pu mettre en dernière condition, la condition ORDER BY liste_proprietaire.proprietaire DESC.

- La clause ORDER BY est également valable dans le cas d'attribut numérique. Dans ce cas, le SGBD organise les résultats suivant un ordre croissant (lorsque l'on met ASC à la fin) ou décroissant (lorsque l'on met DESC à la fin).

- Cette astuce fonctionne également dans le cas où l'attribut est de type date.

Remarque :

Dans nos requêtes SQL nous pouvions imposer au SGBD de ne sélectionner que les enregistrements dont on impose la valeur de certains attributs (comme par exemple en imposant que l'attribut auteur soit égal à Camilia par le biais de la ligne WHERE liste_disque.auteur = "Camilia").

En revanche, nous pouvons également faire une recherche en n'imposant pas réellement la valeur de l'attribut mais plutôt en ne sélectionnant que les enregistrements dont l'attribut commence par une certaine chaîne de caractères ou bien même de ne sélectionner que les enregistrements dont l'attribut ne fait que contient une chaîne de caractères. Tout ceci se fera grâce à la clause LIKE.

Prenons l'exemple suivant :

```
SELECT liste_proprietaire.proprietaire
FROM liste_proprietaire, liste_disque
WHERE liste_disque.auteur LIKE "C%"
AND liste_disque.titre = "Amitié"
AND liste_proprietaire.numero = liste_disque.numero
ORDER BY liste_proprietaire.proprietaire ASC;
```

Dans ce cas, nous n'avons plus la ligne WHERE liste_disque.auteur = "Camilia" mais la ligne WHERE liste_disque.auteur LIKE "c%".

Ce changement implique que nous allons choisir non pas les disques dont l'auteur est Camilia mais les disques dont l'auteur commence par la lettre c.

On aurait également pu faire :

```
SELECT liste_proprietaire.proprietaire
FROM liste_proprietaire, liste_disque
WHERE liste_disque.auteur LIKE "%m%"
AND liste_disque.titre = "Amitié"
AND liste_proprietaire.numero = liste_disque.numero
ORDER BY liste_proprietaire.proprietaire ASC;
```

Et dans ce cas, nous aurions sélectionné les enregistrements dont l'attribut auteur de la table liste_disque contient la lettre m.

Remarque :
La clause LIKE n'est pas limitée à une seule lettre. En effet, on peut très bien faire un LIKE avec un mot complet.
Maintenant, nous allons voir comment intégrer ces requêtes SQL à vos pages PHP.
Créons une page PHP nous permettant de réaliser exactement la même requête que la première de ce document, c'est-à-dire la sélection du numéro de téléphone de ALI.
On a alors le code suivant :
exemple1 : bd1.php

```
<?php
$con = mysqli_connect("localhost","root","","MaBasedeDonnees");
?>
<html>
<body>
<?php
// lancement de la requete
$sql = 'SELECT telephone FROM liste_propreitaire WHERE nom = "ALI"';
// on lance la requête (mysql_query) et on impose un message d'erreur si la requête ne se passe pas bien (or die)
$req = mysqli_query($con, $sql) or die('Erreur SQL !<br />'.$sql.'<br />'.mysqli_error($con));
// on recupere le resultat sous forme d'un tableau
$data = mysqli_fetch_array($req);
// on libère l'espace mémoire alloué pour cette interrogation de la base
mysqli_free_result ($req);
mysqli_close ($con);
?>
Le numéro de téléphone de ALI est :<br />
<?php echo $data['telephone']; ?>
</body>
</html>
```

Ce qui affichera à l'écran :

Le numéro de téléphone d'ALI est : 06-61-85-20-54
Mettons maintenant dans le cas où l'interrogation de la base de données ne retourne pas un, mais un certain nombre d'enregistrements (nombre que l'on ne connaît pas).
En effet, recherchons tous les noms de propriétaires de disque, ainsi que leur numéro de téléphone.
On aura alors le code suivant :
exemple2 : bd2.php

```php
<?php
$con = mysqli_connect("localhost","root","","MaBasedeDonnees");
?>
<html>
<body>
<?php
// lancement de la requête (on n'impose aucune condition puisque l'on désire obtenir la liste complète des propriétaires
$sql = 'SELECT telephone, nom FROM liste_proprietaire';
// on lance la requête (mysqli_query) et on impose un message d'erreur si la requête ne se passe pas bien (or die)
$req = mysqli_query($con, $sql) or die('Erreur SQL !<br />'.$sql.'<br />'.mysqli_error($con));
 // on va scanner tous les enregistrements un par un
while ($data = mysqli_fetch_array($req)) {
    // on affiche les résultats
    echo 'Nom : '.$data['nom'].'<br />';
    echo 'Son tél : '.$data['telephone'].'<br /><br />';
}
mysqli_free_result ($req);
mysql_close ($con);
?>
</body>
</html>
```

Et ainsi, grâce à la boucle while, nous pouvons parcourir tous les enregistrements obtenus par la requête SQL.
Pour finir, nous pouvons juste dire que lorsque l'on effectue une sélection qui contient une jointure, le principe reste exactement le même.

Afin d'améliorer les sélections, vous pouvez faire dépendre vos sélections du résultat obtenu à un formulaire.
En effet, imaginons une première page avec un formulaire nous permettant de saisir le nom d'un propriétaire. Ensuite, dans la page où vous allez faire votre requête (qui donc être également la page contenue dans le champ action de votre formulaire), vous allez récupérer une variable, par exemple $_POST['nom_proprio'] .
De plus, imaginons que l'on désire retrouver le numéro de téléphone de ce propriétaire.
On aura alors:

exemple3 : bd3.php

```php
<?php
// on se connecte à notre base
$con = mysqli_connect("localhost","root","","MaBasedeDonnees");
?>
<html>
<body>
<?php
if (isset($_POST['nom_proprio'])) {
    $sql = 'SELECT telephone FROM liste_proprietaire WHERE nom =
'".$_POST['nom_proprio']."";
    $req = mysqli_query($con, $sql) or die('Erreur SQL !<br />'.$sql.'<br />'.mysql_error($con));
 // on récupère le résultat sous forme d'un tableau
    $data = mysqli_fetch_array($req);
    mysqli_free_result ($req);
    mysqli_close ($con);
// on affiche le résultat
    echo 'Le numéro de téléphone est : '.$data['telephone'];
}
```

```
    else {
        echo 'La variable nom_proprio n\'est pas déclarée';
    }
    ?>
    </body>
    </html>
```

Pour le fichier du formulaire : form3.html

```
<html>
<body>
<form action = "db3.php" method="post">
Nom du propriétaire: <input type = "text" name = "nom_proprio"><br />
<input type = "submit" value = "Envoyer">
</form>
</body>
</html>
```

Insérer des données dans la base

Nous allons maintenant voir comment faire pour insérer des données dans les tables de votre base de données.
Nous allons tout d'abord voir comment faire ces insertions en SQL, puis nous verrons comment les faire directement à partir des pages WEB.
Supposons alors que l'on décide d'ajouter un nouveau propriétaire de disques : Toto par exemple.
Pour insérer ce nouveau propriétaire, il faut fournir au SGBD les informations lui permettant d'insérer ce nouvel enregistrement dans la table liste_proprietaire. Ces informations sont :

- le numéro du nouveau propriétaire
- le nom du nouveau propriétaire
- son numéro de téléphone

Donc il faut fournir tous les attributs de la table afin de produire un nouvel enregistrement.
En revanche, comme nous allons le voir, il n'est pas nécessaire de fournir au SGBD le numéro du nouveau propriétaire car cet attribut à été déclaré AUTO_INCREMENT lors de la création de la table. Ceci implique que le SGBD sait, lors d'une nouvelle insertion, qu'il faut qu'il prenne dans la table liste_proprietaire le numéro le plus grand et qu'il l'augmente de un, et ce nouveau numéro (augmente de un) correspondra au numéro de notre nouveau propriétaire.
On aura alors :

INSERT INTO liste_proprietaire VALUES ('','Toto','06-68-42-01-36');
On remarque tout de suite que la syntaxe pour une insertion est relativement simple.
En effet, étudions ce code :
On insére où ? --> dans la table liste_proprietaire
On insérer quoi ? --> la valeur des différents attributs, c'est-à-dire une première valeur qui correspond à l'attribut numero (qui je vous le rappelle est AUTO_INCREMENT, on n'a donc pas l'utilité de préciser sa valeur, le SGBD sachant quoi mettre), puis on insère la valeur Toto pour l'attribut nom, et enfin la valeur 06-68-42-01-36 pour l'attribut telephone.

Passons tout de suite à l'insertion d'un nouveau enregistrement, et ce, à partir d'une page WEB. Supposons que l'on désire insérer exactement le même enregistrement que dans l'exemple précédent (c'est-à-dire que Toto fait tellement partie de nos amis, qu'on a envie de partager nos disques avec lui).
On aura alors :
exemple1 : db4.php

```
<?php
$con = mysqli_connect("localhost","root","","MaBasedeDonnees");
?>
<html>
<body>
<?php
$sql = 'INSERT INTO liste_proprietaire VALUES ("4", "Toto", "06-68-42-01-36")';
mysqli_query($con, $sql) or die ('Erreur SQL !'.$sql.'<br />'.mysqli_error($con));
mysql_close($con);
?>
Toto vient d'être inséré dans la base.
</body>
</html>
```

Imaginons, alors que l'on désire alors ajouter à la base un disque grâce à la contribution de Toto.

Dans l'absolu, réfléchissons une minute sur ce que nous avons besoin :
- il s'agit d'un nouveau disque (soit son auteur et son titre).
- mais aussi, nous avons besoin du numéro qu'a pris le propriétaire Toto dans la table liste_proprietaire.

En effet, on voit bien que si l'on insère directement un nouveau disque, la jointure entre les deux tables risque de ne pas se faire.
En fait, il faudrait tout d'abord sélectionner le numéro qu'a pris Toto dans la table_proprietaire (par le biais d'une requête SQL de type SELECT).
Cependant, imaginons que nous n'avons pas encore inséré le propriétaire Toto dans notre base de données, et que l'on désire directement insérer ce nouveau propriétaire ainsi qu'un disque lui appartenant. Nous allons voir comment récupérer simplement le nouveau numero qui vient d'être inséré (donc celui de Toto) et ainsi l'utiliser pour insérer notre disque.
On a :
exemple2 : db5.php

```
<?php
$con = mysqli_connect("localhost","root","","MaBasedeDonnees");
?>
<html>
<body>
<?php
$sql = 'INSERT INTO liste_proprietaire VALUES("4", "Toto", "06-68-42-01-36")';
mysqli_query($con, $sql) or die ('Erreur SQL !'.$sql.'<br />'.mysqli_error($con));
// on récupère le dernier numéro inséré, soit le numéro de Toto
$numero_insere = mysql_insert_id();
$sql = 'INSERT INTO liste_disque VALUES ("'.$numero_insere.'", "Smalto", "The player")';
mysql_query ($sql) or die ('Erreur SQL !'.$sql.'<br />'.mysql_error());
mysql_close();
?>
Toto vient d'être inséré dans la base, ainsi que son nouveau disque : The player des Supermen lovers.
</body>
</html>
```

Imaginons que l'on désire insérer des nouveaux disques. Supposons que l'on dispose d'une page html contenant un formulaire permettant de saisir le nom du propriétaire, et que ce formulaire vous demande également le titre d'un album ainsi que son interprète (on suppose également que le champ action de notre formulaire correspond au nom de la page PHP qui traite les données, soit la page contenant le code ci-dessous).

On suppose enfin, que le champ du formulaire contenant le nom du propriétaire porte le nom proprio et le champ contenant l'interprète porte le nom interprete et le champ contenant le titre porte le nom titre.
On aura alors :

exemple3 : db6.php

```
<?php
$con = mysqli_connect("localhost","root","","MaBasedeDonnees");?>
<html>
<body>
<?php
// on teste si les variables du formulaire sont bien déclarées
if (isset($_POST['proprio']) && isset($_POST['interprete']) && isset($_POST['titre'])) {
  // on prépare la requête pour récupérer le numero du propriétaire
     $sql = 'SELECT numero FROM liste_proprietaire WHERE nom = "'.$_POST['proprio'].'"';
  // on lance la requête (mysqli_query) et on impose un message d'erreur si la requête ne se passe pas bien (or die)
```

```
    $req = mysqli_query($con, $sql) or die('Erreur SQL !<br />'.$sql.'<br />'.mysqli_error($con));
   $data = mysqli_fetch_array($req);
    mysqli_free_result ($req);

    $sql = 'INSERT INTO liste_disque VALUES("'.$_POST['numero'].'",
"'.$_POST['interprete'].'", "'.$_POST['titre'].'")';
    mysqli_query($con, $sql) or die ('Erreur SQL !'.$sql.'<br />'.mysql_error($con));
   // on ferme la connexion à la base
    mysql_close($con);
  echo 'Nous venons d\'insérer un nouveau disque : '.$_POST['titre'].' de '.$_POST['interprete'].'
appartenant à '.$_POST['proprio'];
}
else {
    echo 'Les variables du formulaire ne sont pas déclarées';
}
?>
</body>
</html>
```

Pour le fichier de formulaire :form6.html

```
<html>
<body>
<form action = "db6.php" method="post">
Numéro Propriétaire: <input type = "text" name = "proprio"><br />
Interprete: <input type = "text" name = "interprete"><br />
Titre: <input type = "text" name = "titre"><br />
<input type = "submit" value = "Envoyer">
</form>
</body>
</html>
```

Modifier des données de la base

Après avoir vu comment on affiche les données d'une base de données et comment on les insère, voyons maintenant comment modifier ces données.
En SQL tout d'abord comment modifié un enregistrement de la table liste_proprietaire. Supposons que Aziz vienne de changer son numéro de portable (et que son nouveau numéro est : 06-65-99-10-00), il faudra alors faire la modification dans la base de données afin que soit ancien numéro soit remplacé par le nouveau.
On aura alors :
UPDATE liste_proprietaire SET telephone="06-65-99-10-00" WHERE nom="Aziz";
En effet, on modifie quelle table ?
On modifie liste_proprietaire.
Quel attribut modifie-t-on ?
On modifie l'attribut telephone (qui prendra la valeur 06-65-99-10-00).
Et on fait les modifications pour quel(s) enregistrement(s) ?

On modifie le(s) enregistrement(s) où l'attribut nom prend la valeur Aziz (dans notre cas, seul un enregistrement sera modifié car notre table comporte qu'un seul enregistrement où l'attribut nom prend la valeur Aziz).

En revanche, notez bien que dans notre clause WHERE, nous n'avons mis qu'une seule condition. Bien évidemment, tout est possible, vous pouvez en mettre plusieurs (ainsi que des clauses utilisant le LIKE vu précédemment).

Tout dépend des enregistrements que vous voulez modifier.

Cependant, imaginons que nous désirons modifier plusieurs attributs d'un même enregistrement.

En effet, supposons alors que nous possédons dans notre base de données une table ressemblant à la table liste_proprietaire (que nous avons déjà étudié), mais qui comportera plus d'attributs, comme par exemple l'adresse du propriétaire ainsi que son age.

On pourrait alors très bien avoir une table ressemblant à ceci :

N.	Nom	N. tél	Adresse	Age
1	ALI	06-61-85-20-54	2, rue des lilas	23
2	Hamid	06-61-98-78-12	4, rue des fauvettes	22
3	Aziz	06-65-99-10-00	2, rue des tulipes	66
4	Toto	06-68-42-01-36	8, rue du facteur	23

Supposons alors que l'on se soit trompé dans l'age de Aziz et qu'au lieu d'avoir 66 ans (il n'est pas si vieux que ça notre Aziz), et bien, il a tout simplement 65 ans
Supposons également que l'adresse d'Aziz soit erronée et qu'il n'habite pas 2 rue des tulipes, mais 3 rue des tulipes.
On aura alors :
UPDATE liste_proprietaire SET adresse="3, rue des tulipes", age="65" WHERE nom="Aziz";
On remarque alors qu'il suffit de séparer les diverses modifications opérées sur un même enregistrement par une simple virgule.
Voyons maintenant comment effectuer ces modifications dans une page PHP.
Pour ce faire, prenons notre deuxième modification, celle concernant l'adresse et l'age de Aziz.
On aura alors :
exemple1 : db7.php

```
<?php
$con = mysqli_connect("localhost","root","","MaBasedeDonnees");
```

```
?>
<html>
<body>
<?php
$sql ='UPDATE liste_proprietaire SET adresse="3, rue des tulipes", age="65" WHERE nom="Aziz"';
mysqli_query($con, $sql) or die('Erreur SQL !'.$sql.'<br />'.mysql_error($con));
mysql_close($con);
?>
L'adresse et l'age de Aziz viennent d'être modifiés.
</body>
</html>
```

Les modifications peuvent être vraiment dangereuses pour votre base de données.
En effet, si vous ne prenez pas un minimum de précaution pour effectuer vos modifications, vous pouvez très bien modifier un enregistrement d'une table et perde en même temps la jointure avec une autre table.

En effet, nous savons que nos tables liste_disque et liste_proprietaire sont liées par l'intermédiaire du numéro de propriétaire (la jointure).

Ceci implique donc que si pour une raison ou pour une autre, nous sommes amenés à modifier ce numéro (dans la table liste_proprietaire par exemple), il faudra également penser à faire la modification de ce même numéro dans l'autre table (liste_disque) afin que la jointure entre les deux tables soit toujours fonctionnelle.

Supposons que l'on dispose d'un page WEB comportant un formulaire disposant des champs suivants :
- un champ texte (de NAME proprio) permettant de saisir le nom d'un propriétaire.
- un champ texte (de NAME nouvelle_adresse) permettant de saisir une nouvelle adresse pour le propriétaire saisie.

Supposons ensuite que ce formulaire a pour balise ACTION la page traitement.php qui nous permet de modifier l'adresse du propriétaire en question.
On aura alors le code suivant (pour la page traitment.php) :
exemple2 : db8.php

```
<?php
$con = mysqli_connect("localhost","root","","MaBasedeDonnees");
?>
<html>
<body>
<?php
if (isset($_POST['nouvelle_adresse']) && isset($_POST['proprio'])) {
    $sql = 'UPDATE liste_proprietaire SET adresse="'.$_POST['nouvelle_adresse'].'" WHERE nom="'.$_POST['proprio'].'"';
    mysqli_query($con, $sql) or die('Erreur SQL !'.$sql.'<br />'.mysqli_error($con));
    mysqli_close($con);
    echo 'La nouvelle adresse de '.$_POST['proprio'].' est : '.$_POST['nouvelle_adresse'];
}
else {
    echo 'Les variables du formulaire ne sont pas déclarées';
}
?>
</body>
</html>
```

Supprimer des données de la base

Reprenons nos deux tables liste_proprietaire et liste_disque que nous avons utilisé aux exemples précédents.
La table liste_proprietaire :

N. du propriétaire	Propriétaire	N. tél
1	ALI	06-61-85-20-54
2	Hamid	06-61-98-78-12
3	Aziz	06-65-99-10-00
4	Toto	06-68-42-01-36

La table liste_disque :

N. du propriétaire	Auteur	Titre
1	Camilia	Amitié
1	Daft Punk	A la découverte
2	Camilia	Amitié
2	Télénor	Monde sans frontières
3	Calairis	Bonjour
2	Bob Dinar	Paradise
4	Smalto	The player

Voyons, en SQL, comment supprimer un enregistrement de la table liste_proprietaire.
Supposons que l'on désire supprimer Toto de notre base de données. On écrira alors :
DELETE from liste_proprietaire WHERE nom="Toto";

En effet, on efface un enregistrement (ou plusieurs, tout dépend de la clause WHERE) de quelle table ?
De la table liste_proprietaire.
Quel(s) enregistrement(s) efface-t-on ?
On efface tous les enregistrements de la table liste_proprietaire ou l'attribut nom prend la valeur Toto (dans notre cas, un seul enregistrement porte la valeur Toto pour l'attribut nom).

Notez bien que la clause WHERE peut très bien contenir plusieurs conditions, elles seront alors séparés par des opérateurs booléens (AND correspondant à un ET logique ou OR correspondant à un OU logique).
Cependant, je vous rappelle que des requêtes SQL peuvent être beaucoup plus complexes, et dans ce cas, je vous renvoie à la documentation MySQL.

Remarque

Lorsque l'on effectue une suppression des enregistrements, il faut toujours faire attention à effacer non seulement les enregistrements de la table dont on veut supprimer le(s) élément(s) mais éventuellement les autres enregistrements d'une autre table (si les deux tables sont jointes par le biais d'un attribut).

En effet, dans notre exemple, nous venons de supprimer de la table liste_proprietaire l'enregistrement dont l'attribut nom valait Toto. Cependant, on remarque que la table liste_disque comporte des éléments qui étaient liés à Toto. Or, vu que ces éléments ne nous servent plus à rien maintenant (car on a supprimé Toto de la liste), il faut également penser à les supprimer (afin d'avoir une base de données homogène).

Voyons maintenant comment effectuer ces suppressions par le biais d'une page WEB écrite en PHP. Prenons par exemple le cas d'une page PHP permettant la suppression de Toto de la base de données ainsi que de toutes les informations le concernant (c'est-à-dire les disques lui appartenant).
On aura alors :
exemple1 : db9.php

```
1. <?php
```

```
2.  $base = mysql_connect ('serveur', 'login', 'pass');
3.  mysql_select_db ('ma_base', $base) ;
4.  ?>
5.  <html>
6.  <body>
7.  <?php
8.  // lancement de la requête pour effacer Toto
9.  $sql ='DELETE from liste_proprietaire WHERE nom="Toto"';
10. // on exécute la requête (mysql_query) et on affiche un message au cas où la requête ne se passait pas bien (or die)
11. mysql_query($sql) or die('Erreur SQL !'.$sql.'<br />'.mysql_error());
12. // lancement de la requête pour effacer les disques de Toto (je vous rappelle que Toto à le numéro 4)
13. $sql ='DELETE from liste_disque WHERE numero="4"';
14.  // on exécute la requête (mysql_query) et on affiche un message au cas où la requête ne se passait pas bien (or die)
15. mysql_query($sql) or die('Erreur SQL !'.$sql.'<br />'.mysql_error());
16.  // on ferme la connexion à la base
17. mysql_close();
18. ?>
19. Toto et tous ces disques ont étés supprimés de la base de données.
20. </body>
21. </html>
```

Afin de rendre vos pages beaucoup plus dynamiques, il serait intéressant de faire une page WEB contenant un formulaire possédant un champ permettant de saisir le nom du membre à effacer.
On suppose alors que un champ NAME qui prend la valeur proprio, et que le formulaire a son champ ACTION qui prend la valeur traitement.php.
Ceci implique que dans la page traitement.php, on aura une variable $proprio qui contient le nom du propriétaire à supprimer.
On aura alors le code suivant :

exemple2 : db10.php

```
1.  <?php
2.  $base = mysql_connect ('serveur', 'login', 'pass');
3.  mysql_select_db ('ma_base', $base) ;
4.  ?>
5.  <html>
6.  <body>
7.  <?php
8.  // on teste si la variable du formulaire est bien déclarée
9.  if (isset($_POST['proprio'])) {
10.      // on recherche le numero du membre à supprimer
11.      $sql = 'SELECT numero FROM liste_proprietaire WHERE nom = "'.$_POST[proprio].'"';
12.   // on lance la requête (mysql_query) et on impose un message d'erreur si la requête ne se passe pas bien (or die)
13.      $req = mysql_query($sql) or die('Erreur SQL !<br />'.$sql.'<br />'.mysql_error());
14.      // on récupère le résultat sous forme d'un tableau
15.      $data = mysql_fetch_array($req);
16.      // on récupère la valeur qui nous intéresse
17.      $numero_du_proprio = $data['numero'];
18.      // on libère l'espace mémoire alloué pour cette interrogation de la base
19.      mysql_free_result ($req);
20.      // lancement de la requête pour effacer notre membre
21.      $sql ='DELETE from liste_proprietaire WHERE nom="'.$_POST['proprio'].'"';
22.   // on exécute la requête (mysql_query) et on affiche un message au cas où la requête ne se passait pas bien (or die)
23.      mysql_query($sql) or die('Erreur SQL !'.$sql.'<br />'.mysql_error());
```

```
24.       // lancement de la requête pour effacer les disques de notre membre
25.       $sql ='DELETE from liste_disque WHERE numero="'.$numero_proprio.'";
26.       // on exécute la requête (mysql_query) et on affiche un message au cas où la requête ne se
      passait pas bien (or die)
27.       mysql_query($sql) or die('Erreur SQL !'.$sql.'<br />'.mysql_error());
28.       // on ferme la connexion à la base
29.       mysql_close();
30.       // un petit message afin de voir ce qui s'est passé
31.       echo 'Nous venons de supprimer '.$_POST['proprio'].' de la base ainsi que tous ces disques';
32. }
33. else {
34.       echo 'La variable de notre formulaire n\'est pas initialisée.';
35. }
36. ?>
37. </body>
38. </html>
```