**SUPPORT DE COURS PHP/MySQL**

Table des matières

[Qu’est-ce que PHP ? 4](#_Toc268268429)

[Ce que l’on peut faire avec PHP 4](#_Toc268268430)

[Ce que l’on ne peut pas faire avec PHP 4](#_Toc268268431)

[Les outils de développement 4](#_Toc268268432)

[Les ressources 4](#_Toc268268433)

[Les principes client-serveur et PHP-MySQL 5](#_Toc268268434)

[Intégration de PHP dans une page HTML 5](#_Toc268268435)

[Variables 6](#_Toc268268436)

[Chaînes de caractères 6](#_Toc268268437)

[Caractères d’échappement et spéciaux 6](#_Toc268268438)

[Fonction include() 8](#_Toc268268439)

[La concaténation 8](#_Toc268268440)

[Passage de valeurs par formulaire 10](#_Toc268268441)

[Les structures de contrôle 10](#_Toc268268442)

[If then else elseif 11](#_Toc268268443)

[switch 12](#_Toc268268444)

[while 13](#_Toc268268445)

[for 13](#_Toc268268446)

[Les fonctions utilisateurs 14](#_Toc268268447)

[Variables d’environnement 14](#_Toc268268448)

[Création d’une base de données MySQL 16](#_Toc268268449)

[Concepts fondamentaux 16](#_Toc268268450)

[Créer une table 16](#_Toc268268451)

[Modifier une table 18](#_Toc268268452)

[Supprimer une table 19](#_Toc268268453)

[Insert et select 19](#_Toc268268454)

[La commande INSERT INTO 20](#_Toc268268455)

[La commande SELECT 20](#_Toc268268456)

[update et delete 23](#_Toc268268457)

[La commande UPDATE 23](#_Toc268268458)

[La commande DELETE 24](#_Toc268268459)

[Where 25](#_Toc268268460)

[Exploiter une base MySQL en PHP 32](#_Toc268268461)

[Interroger une table 32](#_Toc268268462)

[Alimenter une ou plusieurs tables 37](#_Toc268268463)

[Alimenter une table 37](#_Toc268268464)

[Alimenter deux tables et créer une liaison 40](#_Toc268268465)

[Les sessions 41](#_Toc268268466)

[Principe de fonctionnement 42](#_Toc268268467)

[Etude de cas 42](#_Toc268268468)

[Déconnexion de session 45](#_Toc268268469)

[Librairies et API en PHP 45](#_Toc268268470)

# **Qu’est-ce que PHP ?**

**PHP** (Hypertext Preprocessor)
Technologie permettant la création de sites Web dynamiques. PHP se pose en concurrent de technologies telles que ASP et ASP.net (Microsoft) ou encore JSP et Java (Sun). Très en vogue dans le monde du logiciel libre, PHP est le plus souvent associé au système d'exploitation Linux, au serveur Web Apache et à la base de données MySQL.

Langage interprété (un langage de script) exécuté du côté serveur et non du côté client. La syntaxe du langage provient de celles du langage C, du Perl et de Java.

Le langage PHP a été conçu durant l’automne 1994 par Rasmus Lerdorf. Les premières versions (qui restèrent privées) étaient utilisées afin de savoir qui venait consulter son CV en ligne.

## Ce que l’on peut faire avec PHP

* Automatiser la gestion de news, d'articles ou autres éléments de votre site qui ont un caractère répétitif,
* Réaliser un webmail afin de gérer ses mails avec un navigateur internet,
* Mettre à disposition des autres sites automatiquement des informations de son site (via le format RSS par exemple),
* Gérer des galeries photos, des annuaires de liens, des sondages, des forums, des moteurs de recherche internes à votre site, etc. PHP est l'idéal dans ces domaines là.

## Ce que l’on ne peut pas faire avec PHP

* Je souhaite qu’un pop-up s'ouvre quand la page se charge.

Ici, PHP n'est pas la solution. Vous devez utiliser un script qui soit exécuté par le navigateur car c'est lui qui gère l'ouverture des pop-up. Le langage JavaScript est fait pour cela.

* Je souhaite connaître la résolution de l'écran du visiteur afin de faire des statistiques.

De même PHP ne peut pas gérer cela car il s'exécute côté serveur.

* Je souhaite faire un chat en PHP.

Il n'est pas impossible de réaliser un script de chat, il en existe d'ailleurs plusieurs. Cependant, les limitations du protocole HTTP font que ce n'est vraiment pas souple et agréable à utiliser. Le mieux reste d'utiliser IRC.

# Les outils de développement

Bien que le bloc note Windows soit suffisant pour développer en PHP, il est fortement recommandé d’avoir recours à des logiciels de coloration syntaxique généralistes comme Notepad++ ou mieux Phpdesigner qui existe en version gratuite et offre un espace de développement complet HTML / Javascript / PHP / SQL / CSS.

L’utilisation de Dreamweaver est à proscrire car ne produisant pas un code propre ni facilement maintenable.

Un serveur de développement est nécessaire, EasyPHP ou WAMPServer sont des environnements valables.

# Les ressources

Le manuel PHP consultable en ligne <http://www.php.net> est le document de référence.

# Les principes client-serveur et PHP-MySQL

A gauche : Principe de fonctionnement d’un serveur HTML simple (ex. Apache)

A droite : Principe de fonctionnement d’un serveur HTML + PHP + MySQL (LAMP)

 

Le serveur PHP traite les commandes, éventuellement interroge une base MySQL et renvoie le tout sous forme de HTML : Un internaute lambda n’a donc jamais accès au code source PHP.

* Il faut bien distinguer le client et le serveur. Votre navigateur est le client. C'est lui qui demande la page web que vous avez entrée. Le serveur est l'ordinateur sur l'Internet qui héberge cette page web. PHP s'exécute donc côté serveur.

En voici quelques conséquences :

* + Tout ce qui a trait à la présentation de la page (couleur du texte, etc..) est à faire en HTML et CSS, exécutés côté client. PHP n'a rien à voir avec le design de votre page
	+ Tout ce qui touche au comportement du navigateur est du domaine du JavaScript, lui aussi exécuté par le client
	+ L'intérêt de PHP est de générer du HTML ou du JavaScript dynamiquement. Le travail effectué avec PHP sur votre page est totalement invisible pour le visiteur.
* Le SQL est un langage a part entière de PHP, il ne faut surtout pas confondre les deux. C'est MySQL qui parse (c'est à dire analyse et exécute) votre code SQL, PHP ne fait qu'envoyer une requête au serveur MySQL

# Intégration de PHP dans une page HTML

Le code PHP est toujours encadré par des balises le signalant. Les balises possibles sont :

* <?php      ?>
* <?      ?>
* <%      %>
* <script language="php">      </script>

L'idéal est d'utiliser les plus correctes : <?php    ?> .

Les autres sont à fuir le plus souvent possible, sauf en cas de nécessité de compatibilité avec un éditeur ASP par exemple.

La première chose à savoir c'est qu'une syntaxe se termine TOUJOURS (sauf quelques exceptions que nous verrons bien plus loin) par un point-virgule (voir ci-dessous), si vous l'oubliez vous verrez apparaître PARSE ERROR lors de l'exécution de votre fichier.

|  |  |
| --- | --- |
| Code PHP | Donne comme résultat à l'écran |
| <?phpecho 'Bonjour le monde !'**;**?> | Bonjour tout le monde |

# Variables

Les variables ont des noms précédés du symbole distinctif « $ ».

Elles ne nécessitent pas de déclaration avant usage. Leur type est fixé dès leur première affectation.
Ce très faible typage des variables qui rappelle JavaScript, s'il peut paraitre simplificateur est lourd de danger. Il est préférable de ne pas en changer dynamiquement par affectation. En fait, cette absence de contrôle de type devient une difficulté pour le programmeur, qui doit rester particulièrement vigilant (il n'y a pas de mode "strict" comme en Perl).

Pour affecter une valeur à une variable, on utilise le symbole « = ». Exemple : $MaVar = 32 signifie que la variable $MaVar contient la valeur 32.

**Attention :** Une variable commence toujours par une lettre et ne doit pas contenir d’espace (par exemple $1var ou $var 1 est interdit, mais $var\_1 est correct).

## Chaînes de caractères

Une chaîne de caractères est un ensemble de caractères délimités par des signes. Les signes permettant de délimiter une chaîne de caractère en PHP sont « ' » (apostrophe ou appelé simple quote en PHP) ou « " » (guillemets ou encore double quote en PHP). La différence entre les deux réside dans le fait que PHP examinera ce que contient une chaîne entre « " », mais pas une chaîne qui est entre « ' » qu'il affichera directement. Il est donc souvent préférable d'utiliser les chaînes délimitées par « " ».

### Caractères d’échappement et spéciaux

Il existe un problème avec les chaînes de caractères, quand on veut afficher une chaîne contenant un « ' » et que celle-ci est délimitée par des « ' ». Exemple :

|  |  |
| --- | --- |
| Code PHP | Donne comme résultat à l'écran |
| echo 'j'utilise php';  | **Parse error:** parse error, unexpected T\_STRING, expecting ',' or ';' in votrefichier.php on line 2  |

Pour PHP la chaine s'arrête au deuxième « ' », (celui de j’) et ne peut pas interpréter la suite, ce qui se traduit par une erreur. La solution, c'est le caractère d’échappement antislash (\) qui permet de faire comprendre à PHP que le deuxième « ' » fait partie de la chaine de caractères (on dit qu’on échappe le « ' »). Ce qui donne :

|  |  |
| --- | --- |
| Code PHP | Donne comme résultat à l'écran |
| echo 'j\'utilise php';  | j'utilise php  |

Une autre solution consiste à délimiter les chaines avec « " », ce qui donne :

|  |  |
| --- | --- |
| Code PHP | Donne comme résultat à l'écran |
| echo "j'utilise php";  | j'utilise php  |

Il existe aussi d'autres caractères spéciaux :

* \t : tabulation
* \r : retour chariot
* \n : nouvelle ligne

Attention, ces caractères spéciaux ne fonctionnent que dans une chaîne délimitée par des « " », si vous les utilisez dans des chaînes délimitées par des « ' » vous verrez apparaître à l’écran \n par exemple ! Voici des exemples de codes utilisant ces caractères spéciaux :

* echo ' un texte '," \n ";
* echo " un texte\n ";

Notez bien que ces caractères s'appliquent aux sources html qui sont générées. Et comme vous le savez, un retour à la ligne dans un fichier html ne fait pas d'effet, il faut mettre un <br />. Néanmoins ils peuvent servir à clarifier le code source, ou pour d'autres utilisations que nous verrons plus loin.

**Attention :** A partir du moment où vous placez du code PHP dans un fichier \*.htm ou \*.html, vous devrez renommer ce fichier en \*.php ou encore \*.phtml, bien que le plus utilisé soit \*.php. Si vous ne faites pas cette manipulation, le code apparaîtra en toutes lettres dans le navigateur sans être exécuté par le serveur (n'ayant pas reconnu l'extension associée à PHP).

PHP s'intègre facilement dans du code HTML classique :

|  |  |
| --- | --- |
| Le code PHP/HTML | Donne comme résultat à l'écran |
| <html><body><p>HEURE :</p><?php// le code PHP ---------$heure = date("H\hi");?><!-- retour au code HTML --><p>Il est <?php echo **$heure**; ?></p></body></html> | HEURE :Il est 18h16 |

L'exemple ci-dessus démontre bien cette facilité à mélanger les deux langages.

A noter : En PHP si vous souhaitez ajouter des commentaires il suffit de faire suivre deux slashes **//** puis le commentaire de votre choix. Ils vous seront utiles pour vous y retrouver 6 mois plus tard, donc n'hésitez pas à en mettre, même sur des choses qui vous paraissent logiques sur l'instant. Si vous souhaitez mettre plusieurs lignes en commentaire, vous pouvez également utiliser le "slash étoile" puis "étoile slash" à la fin comme ceci : /\* le commentaire \*/.

## Fonction include()

Si le code de votre page HTML est long, il est parfois fouillis de rajouter des lignes de codes PHP en plus, une méthode couramment utilisée pour bien séparer le code HTML du code PHP est la fonction include().

|  |  |
| --- | --- |
| Le code HTML/PHP | Donne comme résultat à l'écran |
| <html><body><p>HEURE :</p><?phpinclude("toto.inc.php"); // on appelle le fichier toto.inc.php?></body></html> | HEURE :Il est 18h16. |
| Le code PHP de toto.inc.php |
| <?php$heure = date("H\hi");echo "<p>Il est$ **heure.**</p>");?>  |

Le code PHP est maintenant dans un fichier bien séparé mais est exécuté à l'appel du fichier principal. Vous aurez noté que les fichiers qui sont inclus portent l'extension \*.inc.php, ceci pour une meilleure lisibilité. Ainsi vous savez tout de suite si le fichier est exécuté directement ou bien s'il est uniquement appelé dans un ou plusieurs autres fichiers.

## La concaténation

Le point est utilisé pour concaténer des chaînes, variables etc. Prenons l'exemple d'une phrase où un texte doit être collé au bout d'une variable, (voyez ci-dessous), pour que PHP sache que le nom de la variable s'arrête à un endroit précis, nous utiliserons le point.

|  |  |
| --- | --- |
| Le code PHP  | Donne comme résultat à l'écran |
| <? $date = gmdate("H\hi"); print("Il est $date"."gmt."); ?>   | Il est 19h05gmt. |

Vous le voyez pour éviter que PHP pense que la variable porte le nom $dategmt , il faut refermer la double quote, mettre un point puis la rouvrir pour mettre le restant du texte (gmt). Notez également que le second point est lui placé entre les doubles quotes, donc, sera interprété comme du texte simple et non pas comme une demande de concaténation (c’est le point à la fin de la ligne).

Nous allons maintenant voir la différence entre du texte entre ' ', dites simples quotes et du texte entre double quotes " "

|  |  |
| --- | --- |
| Le code PHP  | Donne comme résultat à l'écran |
| <?  $nom = "Martin";  echo "Mon nom est $nom";?>   | Mon nom est Martin |
| <?  $nom = "Martin";  // affichage avec des simple quote echo 'Mon nom est $nom';?>   | Mon nom est $nom |
| <?  $nom = "Martin";  // affichage avec des simple quote echo 'Mon nom est '.$nom;?>   | Mon nom est Martin |

Vous l'aurez compris, PHP n’interprète pas ce qui se trouve entre simple quotes, ainsi, ce n'est pas la valeur de $nom qui est affiché, mais $nom. Il faut donc utiliser un opérateur de concaténation (le .) pour avoir l'affichage voulu.
Pensez aussi que si vous voulez afficher un ' dans un texte entre deux ' ', alors il faudra faire:
echo 'Aujourd\'hui'; ou echo "Aujoud'hui";
Ainsi le \ (backslash ou antislash) indique à PHP qu'il ne faut pas considérer le ' du milieu comme celui qui délimite la fin de la chaîne de caractères, mais juste un caractère comme un autre. Il en va de même pour afficher un " entre deux " ".

Ci-dessous vous allez voir qu'il est possible de concaténer directement une fonction et une chaîne de caractères.

|  |  |
| --- | --- |
| Le code PHP  | Donne comme résultat à l'écran |
| <? print('Nous sommes le '.gmdate('d-m-Y').'...'); ?>   | Nous sommes le 15-09-2000... |

Nous avons réduit le code d'une ligne, ce qui n'est pas négligeable pour les gros développements.

Dans certains exercices futurs, nous verrons comment appeler une page en passant quelques variables, dans ce cas la concaténation nous servira. Regardez le tableau ci-dessous :

|  |  |
| --- | --- |
| Ce qu'il faut éviter de faire : | Ce qu'il est conseillé de faire : |
| <? $fichier =  "fichier.php3?var=$var&data=$data"; ?>   | <? $fichier = 'fichier.php3?var='.$var.'&data='.$data; ?>   |

En d'autres termes, chaque fois que vous collez du texte et une variable (ou fonction), n'oubliez pas de mettre le point.

# Passage de valeurs par formulaire

Quand l'un de vos visiteurs entre les informations dans un formulaire, celle-ci sont récupérées sous forme de variables.
Le nom de ces variables dépend de la méthode d'envoi du formulaire.
Comme dans notre exemple suivant la méthode d'envoi est POST, il faut mettre comme nom **$\_POST['nom\_du\_champ']**.
Pour simplifier le nom des variables, dans notre exemple, on fait **$nom = $\_POST['nom']**
et **$prenom = $\_POST['prenom']** pour assigner la valeur de la variable **$\_POST['prenom']** a **$prenom** et idem pour **$\_POST['nom']**.

|  |  |
| --- | --- |
| Le code HTML du formulaire | Donne comme résultat à l'écran |
| <html><body> <form method="post" action="verif.php"> Nom : <input type="text" name="nom" size="12"><br> Prénom : <input type="text" name="prenom" size="12"> <input type="submit" value="OK"> </form></body></html>  | Nom:Prénom: |
| Le code PHP de verif.php | Donne comme résultat à l'écran après envoi "OK" |
| <?php $prenom = $\_POST['prenom']; $nom = $\_POST['nom']; print("<center>Bonjour $prenom $nom</center>"); ?>  | Bonjour Thaal Rasha  |

# Les structures de contrôle

Pour commencer voici un petit tableau bien utile sur les instructions les plus utilisées

|  |  |
| --- | --- |
| **if** | Si |
| **else** | Autrement |
| **elseif** | Autrement Si |
| **switch** | selon |
| **while** | Chaque fois que *(boucle)* |
| **for** | Tant que *(boucle)* |
| **==** | Strictement égal à |
| **!=** | Différent de |
| **<** | Plus petit que |
| **>** | Plus grand que |
| **<=** | Plus petit ou égal |
| **>=** | Plus grand ou égal |
| **and ou &&** | Et |
| **or ou ||** | Ou |

## If then else elseif

Pour illustrer les if, else et elseif, voici un exemple très simple à lire, nous définissons une variable à la valeur ' 512 ' puis nous allons tester si celle-ci est comprise entre 0 et 499 puis entre 500 et 999 et enfin supérieure à 999, ce qui nous donne :

|  |  |
| --- | --- |
| Le code PHP | Donne comme résultat à l'écran |
| <? $toto = 512; // on enchaîne les contrôles ci-dessous ---- if($toto>=0 && $toto<500)         //1er     {     echo $toto.' est compris entre 0 et 499';     } elseif($toto>=500 && $toto<1000) //2eme     {     echo $toto.' est compris entre 500 et 999';     } else                                  //3eme     {     echo $toto.' est plus grand que 999';     } ?>  | 512 est compris entre 500 est 999 |

1er contrôle : Si (512 est plus grand ou égal à 0 et que 512 est plus petit que 500) { on affiche le 1er message ; }
2e contrôle : Si (512 est plus grand ou égal à 500 et que 512 est plus petit que 1000) { on affiche le 2e message ; }
3e contrôle : Dans tous les autres cas { on affiche le 3e message ; }

Je vous invite à faire plusieurs tests en changeant à chaque fois la valeur de $toto pour vérifier que les messages respectifs s'affichent bien quand les conditions sont remplies.

Voici un autre exemple pour mieux comprendre, et qui nous permettra de voir la structure switch par la suite.

|  |  |
| --- | --- |
| Le code PHP | Donne comme résultat à l'écran |
| <? $medor = 'chien'; // on enchaîne les contrôles ci-dessous ---- if($medor == 'girafe')     {     echo 'Medor est une girafe !';     } elseif($medor == 'elephant')      {     echo 'Medor est un éléphant';     } elseif($medor == 'souris')     {     echo 'Medor est une souris';     } elseif($medor == 'chien')     {     echo 'Medor est un chien';     } elseif($medor == 'chat')     {     echo 'Medor est un chat';     } else      {     echo 'Peut être un hippopotame ? Qui sait ...';     } ?>  | Medor est un chien |

## switch

Cependant vous vous apercevez que cette structure est un peu lourde, et pas forcement facile à lire. Utiliser un switch permet de résoudre ce probleme. Le code suivant est exactement le même que le précédent, mais ecrit avec un switch

|  |  |
| --- | --- |
| Le code PHP | Donne comme résultat à l'écran |
| <? $medor = 'chien'; switch($medor)     {     case 'girafe':         echo 'Medor est un girafe !';         break;     case 'elephant':         echo 'Medor est un éléphant';         break;     case 'souris':         echo 'Medor est une souris';         break;     case 'chien':         echo 'Medor est un chien';         break;     case 'chat':         echo 'Medor est un chat';         break;     default:         echo 'Peut être un hippopotame ? Qui sait ...';     } ?>  | Medor est un chien |

Notez bien l'utilisation de break; . Cela permet de sortir de la boucle et donc de gagner en efficacité.

## while

Passons maintenant à la fameuse boucle while, je dis "fameuse" car elle est sujette à de petites galères quand on l'utilise pour la première fois. Nous allons reprendre notre variable $toto à laquelle nous allons donner la valeur 6, puis à l'aide de la boucle nous allons nous mettre dans le cas où nous ne connaissons pas la valeur de $toto et allons la rechercher. Ce qui donne :

|  |  |
| --- | --- |
| Le code PHP | Donne comme résultat à l'écran |
| <? $toto = 6; $i = 0;  //----------[DEBUT BOUCLE]------------ while($i != $toto)     {     echo 'Toto est different de '.$i.'<br>';     $i++;  // $i++ est équivalent à ($i+1)     } //------------[FIN BOUCLE]------------ echo 'Toto est égal à '.$i; ?>  | Toto est different de 0Toto est different de 1Toto est different de 2Toto est different de 3Toto est different de 4Toto est different de 5**Toto est égal à 6**  |

Par convention la variable $i fait toujours office de compteur dans une boucle elle a toujours la valeur "0" au début, vous notez que cette valeur prend +1 à la fin de la boucle ($i++) et chaque fois que la condition a été respectée (en l'occurrence que $i est différent de $toto) on retourne à WHILE et l'on fait un nouveau test, etc., ce qui donne en français :

$i = 0 , on teste si 0 est différent de toto = Oui, on affiche le message (echo) puis on ajoute 1 à $i ($i++) et on retourne à while.

$i = 1 , on teste si 1 est différent de toto = Oui, on affiche le message (echo) puis on ajoute 1 à $i ($i++) et on retourne à while.

$i = 2 , on teste si 2 est différent de toto = Oui, on affiche le message (echo) puis on ajoute 1 à $i ($i++) et on retourne à while.

etc...

$i = 6 , on teste si 6 est différent de toto = Non, on sort de la boucle (accolades), et on poursuit le code. En l'occurrence on affiche le message (Toto est égal à 6).

## for

Nous allons maintenant voir les boucle for. Elles permettent de réaliser la même chose que les boucles while, encore une fois c'est la syntaxe qui change. Au lieu de déclarer le compteur avant le debut de la boucle ($i=0;) et a chaque fin de tour d'incrémenter le compteur ($i++), on le fait directement dans la déclaration de la boucle. Voici le même code que précédemment avec une boucle for :

|  |  |
| --- | --- |
| Le code PHP | Donne comme résultat à l'écran |
| <? $toto = 6; //----------[DEBUT BOUCLE]------------ for($i=0; $i != $toto ; $i++)     {     echo 'Toto est different de '.$i.'<br>';     } //------------[FIN BOUCLE]------------ echo 'Toto est égal à '.$i; ?>  | Toto est different de 0Toto est different de 1Toto est different de 2Toto est different de 3Toto est different de 4Toto est different de 5**Toto est égal à 6** |

Notez : Il est vraiment très important de maîtriser les boucles car c'est là un élément systématiquement utilisé pour afficher des résultats venant d'une base de données et nous l'utiliserons beaucoup.

# Les fonctions utilisateurs

PHP propose de nombreuses fonctions, mais un autre avantage est de pouvoir créer les siennes à l'aide de function(). Ceci est vraiment très utile pour ne pas avoir à retaper des parties de code en entier.

|  |
| --- |
| Le code PHP de fonction.php |
| <?function Arial**(**$size**,**$color**,**$texte**)****{**  print**(**"<font face=Arial size="**.**$size**.**" color="**.**$color**.**">"**.**$texte**.**"</font>"**);****}**?> |
| Le code PHP de index.php  |
| <?Require**(**"fonction.php"**);** // on appelle la page contenant la fonction*// affichage -----------------------------------------*Arial**(**"2"**,**"red"**,**"Ici le texte ..."**);**Arial**(**"3"**,**"#0F74A3"**,**"Le second texte ..."**);**?> |
| On appelle index.php dans le navigateur, ce qui donne à l'écran |
| Ici le texte ...Le second texte ... |

Reprenons depuis le début. On distingue donc deux étapes lors de l'utilisation de fonctions. La première consiste à créer votre fonction en lui donnant un nom, en choisissant les paramètres dont elle aura besoin et déterminer ce qu'elle doit faire. Une fois cela fait, vous pouvez appeler votre fonction. Plus précisément : entre parenthèses ce sont les arguments que vous entrerez lors de l'utilisation de cette fonction. Ces arguments correspondent bien sûr à ceux qui se trouvent entre parenthèses à la création de la fonction. Ici, il s'agit juste de donner la taille "$size", la couleur "$color"et le contenu du texte "$texte". L'utilisation de vos fonctions se fait de la même manière que les fonctions intégrées de PHP (print et echo en l'occurrence). Dans Index.php, vous voyez que l'on appelle la fonction Arial() et qu'on lui donne les valeurs que l'on veut, pratique non ?

Pratique : Je vous conseille de créer un fichier fonction.php qui va contenir toutes vos fonctions de texte, tableau, etc. Il suffit juste ensuite de mettre un require() en entête de chacun de vos autres fichiers php pour pouvoir utiliser toutes les fonctions.

# Variables d’environnement

Ici ça n'est pas vraiment un exercice que je vous propose, même si vous allez pouvoir tester ces variables mais plutôt des informations que je vous livre. PHP propose toute une série de variables qui sont déjà implantées dans le langage sans que vous ayez à les créer, on les appelle les variables d'environnement.

Ces variables appartiennent à la famille des variables globales $\_SERVER

|  |  |
| --- | --- |
| Code PHP | Donne comme résultat à l'écran |
| <?print("Votre adresse IP est : **$\_SERVER['REMOTE\_ADDR']**");?> | Votre adresse IP est : 201.65.8.56 |

Voilà, rien de plus simple pour connaître l'adresse IP d'un visiteur :), il s'agit de la variable d'environnement $\_SERVER['REMOTE\_ADDR'].

Voici ci-dessous la liste exhaustive des variables d'environnement existantes :

|  |  |  |
| --- | --- | --- |
| Variables | Description | Donne comme résultat à l'écran (free.fr) |
| **$\_SERVER['DOCUMENT\_ROOT']** | Racine du serveur | /var/www/php.proxad.net |
| **$\_SERVER['HTTP\_ACCEPT\_LANGUAGE']** | Langage accepté par le navigateur | fr |
| **$\_SERVER['HTTP\_HOST']** | Nom de domaine du serveur | proxyphp3.free.fr |
| **$\_SERVER['HTTP\_USER\_AGENT']** | Type de navigateur | Mozilla/4.0 (compatible; MSIE 5.0; Windows 98; DigExt) |
| **$\_SERVER['PATH\_INFO']** | Chemin web du script | /d2expert.free.fr/ fichier.php |
| **$\_SERVER['PATH\_TRANSLATED']** | Chemin complet du script | /var/www/free.fr/fichier.php |
| **$\_SERVER['REQUEST\_URI']** | Chemin du script | /d2expert.free.fr/ fichier.php?var=23&data=ok |
| **$\_SERVER['REMOTE\_ADDR']** | Adresse IP du client | 195.132.7.201 |
| **$\_SERVER['REMOTE\_PORT']** | Port de la requête HTTP | 45039 |
| **$\_SERVER['QUERY\_STRING']** | Liste des paramètres passés au script | var=23&data=ok |
| **$\_SERVER['SERVER\_ADDR']** | Adresse IP du serveur | 212.27.32.44 |
| **$\_SERVER['SERVER\_ADMIN']** | Adresse de l'administrateur du serveur | email@email.com |
| **$\_SERVER['SERVER\_NAME']** | Nom local du serveur | php.proxad.net |
| **$\_SERVER['SERVER\_SIGNATURE']** | Type de serveur | ? |
| **$\_SERVER['REQUEST\_METHOD']** | Méthode d'appel du script | GET |

Pour finir (les puristes vont peut-être m'en vouloir d'en parler ici),  je voulais m'attarder quelques instants sur ce qui est une fonction et non pas une variable d'environnement. Je veux parler de phpinfo(). A l'aide de celle-ci et en 10 secondes vous aller pouvoir connaître la configuration et la version exacte de PHP qu'utilise le serveur où vous êtes hébergé.

# Création d’une base de données MySQL

## Concepts fondamentaux

Bases

Tables

Champs

Enregistrements

## Créer une table

1. CREATE TABLE : Le langage SQL (Structured Query Langage) permet d'interroger une base de données qui supporte ce langage, les plus connues sont MS Access, mSQL, MySQL et PostgreSQL. Comme vous le savez nous utiliserons MySQL  (attention mSQL n'est pas MySQL), nuance importante.

Comme je vous le disais en introduction nous allons travailler chez free.fr , vous pouvez donc vous rendre à <http://sql.free.fr/phpMyAdmin/> pour accéder à votre base de données MySQL via l'interface phpMyAdmin. Notez que 99% des hébergeurs proposent phpMyAdmin pour administrer votre base de données (cela facilite grandement la vie).

La première chose que nous allons devoir faire c'est de créer une table, c'est la commande CREATE TABLE, voyez la syntaxe ci-dessous qui permet de créer clients\_tbl. Il est important de savoir comment l'on crée une table en SQL avant de passer par l'interface phpMyAdmin pour le faire :

|  |
| --- |
| Syntaxe SQL pour créer la table : clients\_tbl |
| CREATE TABLE clients\_tbl (id INT not null AUTO\_INCREMENT, prenom VARCHAR (50) not null , nom VARCHAR (50) not null , ne\_le DATE not null , ville VARCHAR (90) not null , enfants INT not null , PRIMARY KEY (id)) |
| Commentons la table en la visualisant sous phpMyAdmin pour MySQL |
|

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Champ** | **Type** | **Null** | **Defaut** | **Extra** | **Commentaire** |
| id  | int  | Non  |   | auto\_increment  | **INT pour Integer (nombre entier)** : C'est l'id qui va nous permettre de classer nos enregistrements, l'auto-increment se charge d'affecter un nouveau numéro aux nouveaux enregistrements qui s'ajoutent dans la table. |
| prenom  | varchar(50)  | Non  |   |   | **Varchar pour Chaîne de caractères** : Nous l'utilisons quand nous souhaitons que le champ puisse recevoir des chaînes mélangeant texte, nombre, etc ... |
| nom  | varchar(50)  | Non  |   |   | **Idem que PRENOM**, la valeur placée entre parenthèses définit le nombre maximum de caractères que le champs accepte, ici 50. Si vous essayez d'insérer une chaîne de 55 caractères, les 5 derniers seront coupés.  |
| ne\_le  | date  | Non  | 0000-00-00  |   | **Date** : Permet de stocker des dates, mais attention TOUJOURS au format US soit : Année/mois/jour, si vous envoyez un format français J/M/A vous obtiendrez une date fausse dans la base. |
| ville  | varchar(90)  | Non  |   |   | **Idem que PRENOM**, ici nous avons spécifié 90 caractères car certaines villes comportent beaucoup de caractères (avec 90 nous sommes sûr) |
| enfants  | int  | Non  | 0  |   | **INT (nombre entier)** : Dans ce champs n'arriveront que des nombres entiers donc le INT voir SMALL INT et de rigeur. |

 |
| Ci-dessous la table : clients\_tbl une fois créée |
| +++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++  **id**  +   **prenom**  +    **nom**   +    **ne\_le**   +    **ville**    +  **enfants**  +++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++Bien sûr pour le moment cette table ne comporte aucun enregistrement ... |

Dès cet instant vous devez consulter [la documentation officielle de MySQL](http://www.nexen.net/docs/mysql/annotee/column-types.php) pour connaître tous les types de champs que vous pouvez utiliser dans une table, voici quelques exemples :

[TINYINT](http://www.nexen.net/docs/mysql/annotee/numeric-types.php) : Entier de 0 à 255 (unsigned)

[SMALLINT](http://www.nexen.net/docs/mysql/annotee/numeric-types.php) : Entier de 0 à 65535 (unsigned)

[MEDIUMINT](http://www.nexen.net/docs/mysql/annotee/numeric-types.php) : Entier de 0 à 16777215 (unsigned)

[INT](http://www.nexen.net/docs/mysql/annotee/numeric-types.php) : Entier de 0 à 4294967295 (unsigned)

[BIGINT](http://www.nexen.net/docs/mysql/annotee/numeric-types.php) : Entier de 0 à 18446744073709551615 (unsigned)

[DECIMAL](http://www.nexen.net/docs/mysql/annotee/numeric-types.php) : Un nombre à virgule flottante

[DATE](http://www.nexen.net/docs/mysql/annotee/date-and-time-types.php) : Une date, va de '1000-01-01' à '9999-12-31'

[DATETIME](http://www.nexen.net/docs/mysql/annotee/date-and-time-types.php) : Date et Heure, va de '1000-01-01 00:00:00' à '9999-12-31 23:59:59'

[TIMESTAMP](http://www.nexen.net/docs/mysql/annotee/date-and-time-types.php) : Date et Heure exprimée en secondes depuis le 1er janviers 1970. Va de '1970-01-01 00:00:00' à quelque part, durant l'année 2037

[TIME](http://www.nexen.net/docs/mysql/annotee/time.php) : Une mesure de l'heure, va de '-838:59:59' à '838:59:59'

[YEAR](http://www.nexen.net/docs/mysql/annotee/year.php) : Une annnée, va de 1901 à 2155

[CHAR](http://www.nexen.net/docs/mysql/annotee/char.php) : Chaîne de caractère de taille fixe, va de 1 à 255 caractères

[VARCHAR](http://www.nexen.net/docs/mysql/annotee/char.php) : Chaîne de caractère de taille variable, va de 1 à 255 caractères

[TINYTEXT ou TINYBLOB](http://www.nexen.net/docs/mysql/annotee/blob.php) : Un objet BLOB ou TEXT, longueur maximale de 255

[TEXT ou BLOB](http://www.nexen.net/docs/mysql/annotee/blob.php) : Un objet BLOB ou TEXT, longueur maximale de 65535

[MEDIUMTEXT ou MEDIUMBLOB](http://www.nexen.net/docs/mysql/annotee/blob.php) : Un objet BLOB ou TEXT, longueur maximale de 16777215

[LONGTEXT ou LONGBLOB](http://www.nexen.net/docs/mysql/annotee/blob.php) : Un objet BLOB ou TEXT, longueur maximale de 4294967295

A noter : Concernant la notion de UNSIGNED est importante pour les Integer (INT, SMALLINT, etc.) je vous propose un petit schéma simple pour vous expliquer la différence entre un Integer UNSIGNED ou pas:


Le **TINYINT** couvre **de 0 à 255** quand il est **UNSIGNED**, mais
il couvrira **de - 128 à 127** (soit toujours 255) si il ne l'est pas,
Ceci est valable pour tous les Integers (nombres entiers).

## Modifier une table

2. ALTER TABLE : Une fois que votre table est créée vous pourrez bien sûr la modifier en utilisant ALTER TABLE, voyez l'exemple ci-dessous pour ajouter un champs :

|  |
| --- |
| Syntaxe SQL pour ajouter le champ 'tel' à la table : clients\_tbl |
| ALTER TABLE clients\_tbl ADD tel INT not null |
| Ci-dessous la table clients\_tbl une fois modifiée |
| +++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++  **id**  +   **prenom**  +    **nom**   +    **ne\_le**   +    **ville**    +  **enfants**  + **tel** +++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++  |

  Voilà ci-dessus notre table a été modifiée grâce à ALTER TABLE puis ADD (comme ajouter).

Maintenant supprimons le champs tel de la table :

|  |
| --- |
| Syntaxe SQL pour supprimer le champ 'tel' de la table : clients\_tbl |
| ALTER TABLE clients\_tbl DROP tel |
| Ci-dessous la table clients\_tbl une fois modifiée |
| +++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++  **id**  +   **prenom**  +    **nom**   +    **ne\_le**   +    **ville**    +  **enfants**  +++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++ |

    Voilà ci-dessus notre table a été modifiée grâce à ALTER TABLE puis DROP. Cette commande DROP va également nous être utile pour supprimer une table complète.

## Supprimer une table

3. DROP TABLE : Il s'agit de la commande qui permet de supprimer une table complète, attention en supprimant une table vous perdez tout ce qu'elle contenait, donc à utiliser avec prudence !

|  |
| --- |
| Syntaxe SQL pour supprimer la table : clients\_tbl |
| DROP TABLE clients\_tbl |

Notre table clients\_tbl a été complètement effacée avec la commande DROP TABLE, dans notre cas elle ne contenait aucune information mais dans le cas contraire, cela signifie la perte de toutes les données de la table.

Pour finir : Nous avons vu ici la majeure partie des choses importantes à connaître en ce qui concerne la création et la modification des tables. Dans les prochains exercices nous verrons comment traiter les données d'une table avec les commandes SELECT, INSERT, UPDATE et DELETE.

## Insert et select

Pour commencer nous allons re-créer la table client\_tbl de l'exercice n°11.

|  |
| --- |
| Syntaxe SQL pour créer la table : clients\_tbl |
| CREATE TABLE clients\_tbl (id INT not null AUTO\_INCREMENT, prenom VARCHAR (50) not null , nom VARCHAR (50) not null , ne\_le DATE not null , ville VARCHAR (90) not null , enfants INT not null , PRIMARY KEY (id)) |
| Ci-dessous la table : clients\_tbl une fois créée |
| +++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++  **id**  +   **prenom**  +    **nom**   +    **ne\_le**   +    **ville**    +  **enfants**  +++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++Bien sûr pour le moment cette table ne comporte aucun enregistrement ... |

### La commande INSERT INTO

Cette commande permet d'insérer des enregistrements dans une table en l'occurrence clients\_tbl.

INSERT INTO clients\_tbl(id,prenom,nom,ne\_le,ville,enfants) VALUES('','Patrick','Martin','1965-10-08','Bordeaux','2')

ou bien cette autre requête qui aura le même résultat, mais que nous vous déconseillons fortement, car elle ne marchera plus si vous modifiez votre table, en particulier si vous ajoutez un champ :

INSERT INTO clients\_tbl VALUES('','Patrick','Martin','1965-10-08','Bordeaux','2')

|  |
| --- |
| Ci-dessous la table : clients\_tbl avec le nouvel enregistrement |
| +++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++  **id**  +   **prenom**  +    **nom**   +    **ne\_le**   +    **ville**    +  **enfants**  ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++   1  +   Patrick  +  Martin   + 1965-10-08 +  Bordeaux   +     2     +++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++ |

Dans le premier exemple nous avons spécifié les noms des champs (entre parenthèses) juste après le nom de la table. Dans ce cas il n'est pas obligatoire de le faire si dans les values vous spécifiez une valeur par champs, mysql affectera les valeurs dans l'ordre donné.

Par contre si vous ne donnez que deux valeurs, il sera important de toujours spécifier les champs dans lesquels elle doivent être insérées, exemple :

INSERT INTO clients\_tbl(id,nom) Values('','Dupond')

ou encore :

INSERT INTO clients\_tbl(nom,enfants) Values('Dupond','2')

Notez : La valeur du champ id est vide, je vous rappelle que ce champ est en auto-increment dans notre base, un nombre automatique sera donc attribué à chaque nouvel enregistrement, dans ce cas la valeur dans la requête peut rester vide !

Important : Si vous voulez n'insérer que quelques-unes des valeurs d'un enregistrement , vous devrez avoir spécifié lors de la création de la table que les autres champs peuvent rester vides ! (NULL).

###  La commande SELECT

Nous allons travailler à partir de la table ci-dessous qui comporte 5 enregistrements :

|  |
| --- |
| Ci-dessous le contenu final de la table : clients\_tbl |
| +++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++  **id**  +   **prenom**  +    **nom**   +    **ne\_le**   +    **ville**    +  **enfants**  ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++   1  +   Patrick  +  Martin   + 1965/10/08 +  Bordeaux   +     2     ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++   2  +   Julien   +  Lebreton + 1964/02/21 +  Paris      +     2     ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++   3  +   Marc     +  Richard  + 1958/04/15 +  Lille      +     4     ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++   4  +   Francis  +  Perrin   + 1982/12/05 +  Paris      +     0     ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++   5  +   Daniel   +  Bacon    + 1974/07/13 +  Reims      +     1     +++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++Ceci est une représentation en mode console et non pas sous phpMyAdmin |

Admettons que nous voulions afficher uniquement les personnes qui n'ont que 2 enfants, la requête SQL sera :

SELECT \* FROM clients\_tbl WHERE enfants='2'

|  |
| --- |
| Soit en français : |
| **SELECT** | Je Sélectionne |
| **\*** | Tous les champs |
| **FROM clients\_tbl** | Depuis la table client\_tbl |
| **WHERE enfants='2'** | Quand le champs enfants est égal à 2 |
|  |  |
| Ce qui donne comme résultat : |
| +++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++   1  +   Patrick  +  Martin   + 1965/10/08 +  Bordeaux   +     2     ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++   2  +   Julien   +  Lebreton + 1964/02/21 +  Paris      +     2     +++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++ |

Voici la même requête, mais cette fois nous n'allons demander l'affichage que des noms et prénoms :

SELECT nom,prenom FROM clients\_tbl WHERE enfants='2'

|  |
| --- |
| Soit en français : |
| **SELECT** | Je sélectionne |
| **nom, prenom** | Les champs nom et prenom |
| **FROM clients\_tbl** | Depuis la table client\_tbl |
| **WHERE enfants='2'** | Quand le champs enfants est égal à 2 |
|  |  |
| Ce qui donne comme résultat : |
| +++++++++++++++++++++++++++   Patrick  +  Martin   ++++++++++++++++++++++++++++   Julien   +  Lebreton +++++++++++++++++++++++++++ |

Reprenons la première requête et ajoutons d'autres conditions (WHERE), à savoir que nous allons demander l'affichage des personnes qui ont 1 ou 2 enfants et qui habitent la ville de Paris :

SELECT \* FROM clients\_tbl WHERE enfants='1' OR enfants='2' AND ville='Paris'

Vous devez faire la différence entre OR et AND, nous pouvons l'analyser ainsi :

Si le nombre d'enfant est 1 et que la ville est Paris , c'est OK.
Et si le nombre d'enfants est 2 et que la ville est Paris , c'est OK.

Il faut que les 2 conditions soient respectées pour que l'enregistrement soit affiché !

|  |
| --- |
| Voici les opérateurs possibles : |
| **+** | Addition |
| **-** | Soustraction |
| **\*** | Multiplication |
| **/** | Division |
| **<** | Plus petit que |
| **<=** | Plus petit ou égal à |
| **=** | Égal à |
| **!=** ou **<>** | N'est pas égal à |
| **>=** | Plus grand ou égal à |
| **and** | ET |
| **or** | OU |
| **not** | Négation |

**Pour finir voici quelques autres exemples de SELECT :**

|  |
| --- |
| SELECT \* FROM clients\_tbl WHERE ne\_le < "1978-01-01" |
| Sélection des personnes ayant une date de naissance plus petite que **1978/01/01**. |
|  |
| SELECT \* FROM clients\_tbl WHERE enfants != '0' |
| Sélection des personnes dont le nombre des enfants est différent de " **0** ". |
|  |
| SELECT \* FROM clients\_tbl WHERE nom LIKE 'le%' |
| Sélection des personnes dont le nom commence par  " **le** ". |
|  |
| SELECT \* FROM clients\_tbl WHERE nom LIKE '%ri%' |
| Sélection des personnes qui ont dans leur nom la syllabe " **ri** " . |

Il serait bien trop long d'énumérer tous les exemples possibles, mais n'hésitez pas à consulter
[la documentation en Français de Nexen.net sur MySQL](http://www.nexen.net/docs/mysql/annotee/manuel_tocd.php).

## update et delete

Ici nous continuons de travailler la table client\_tbl de l'exercice n°11.

### La commande UPDATE

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

UPDATE clients\_tbl SET prenom='Jacques' WHERE id=1

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

|  |  |
| --- | --- |
| UPDATE clients\_tbl | Mise à jour de la table **Clients\_tbl** |
| SET prenom='Jacques'  | Modifier le champ **prenom** pour la valeur **Jacques** |
| WHERE id=1 | Quand le champ **id** est **égal à 1** |

|  |
| --- |
| Ci-dessous l'enregistrement de la table : clients\_tbl une fois modifié. |
| +++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++ **id** + **prenom** + **nom** + **ne\_le** + **ville** + **enfants** ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++ 1 + Jacques + Martin + 1965/10/08 + Bordeaux + 2 +++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++ |

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

UPDATE clients\_tbl SET prenom='Jean-Pierre', nom='Papin', ville='Marseille', enfants=3 WHERE id=1

Vous le voyez, il suffit de séparer les "champs/valeurs" par une virgule, ce qui donne comme résultat dans la table :

|  |
| --- |
| Ci-dessous l'enregistrement de la table : clients\_tbl une fois modifié. |
| +++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++ **id** + **prenom** + **nom** + **ne\_le** + **ville** + **enfants** ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++ 1 + Jean-Pierre+ Papin + 1965/10/08 + Marseille + 3 +++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++ |

### La commande DELETE

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

DELETE FROM clients\_tbl WHERE id=1

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

|  |  |
| --- | --- |
| DELETE FROM clients\_tbl | Effacer de la table **Clients\_tbl** |
| WHERE id=1 | Quand l'id de l'enregistrement est **égal à 1** |

Notez : Pour finir notez que les opérateurs de l'exercices n°12 peuvent s'appliquer également dans le cadre d'un UPDATE ou d'un DELETE. En fait ils peuvent s'appliquer à n'importe quelle requête SQL.

Conclusion : Je n'ai abordé dans les 3 exercices sur SQL que des syntaxes simples, volontairement pour ne pas trop vous embrouiller. Nous aurons l'opportunité dans les futurs exercices de voir par exemple des requêtes UPDATE qui mettent à jour plusieurs tables en même temps, mais ne brûlons pas les étapes ;) ... Ceci dit si vous vous sentez à l'aise n'hésitez pas !

## Where

Comme tous les cours se suivent mais ne se ressemblent pas, nous allons ici comprendre les mécanismes du WHERE et ses possibilités grâce à un exemple comprenant deux tables (nous verrons par la suite pourquoi), une table Livres et une table Genres.
Tout d'abord il faut savoir que le mot clé WHERE peut être utilisé dans les requêtes SELECT, DELETE et UPDATE (son utilisation est cumulative grâce au AND ou OR) mais pas dans le INSERT car on ne peut pas insérer à un endroit précis de la table, l'insertion se fait toujours à la fin.
Ce cours ne sera basé que sur l'instruction SELECT.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|

|  |
| --- |
| ***Livres*** |
| **IDLivre** | **Prix** | **Titre** | **CodeGenre** |
| 1 | 40 | Le glaive magique | BD |
| 3 | 40 | Gaffes en gros | BD |
| 4 | 40 | Lagaffe nous gâte | BD |
| 5 | 45 | QRN sur Bretzelburg | BD |
| 6 | 80 | Tour de manège | RG |
| 7 | 45 | Le spectre aux balles d'or | BD |
| 8 | 30 | La bonne chanson | Po |
| 9 | 50 | La jeune Parque | Po |
| 10 | 50 | Michel Strogoff | Ro |
| 11 | 50 | La Serpe d'or | BD |
| 12 | 70 | Toujours aimer | RG |
| 13 | 70 | Toujours aimante | RG |
| 14 | 72 | Toujours aimé | RG |

 |

|  |
| --- |
| ***Genres*** |
| **CodeGenre** | **LibelleGenre** |
| BD | Bande Dessinée |
| Po | Poésie |
| RG | Roman de gare |
| Ro | Roman |
| SF | Science Fiction |

 |

1 - Sélection directe par l'ID ou le code
Ce cas vous permet de faire une sélection par exemple sur un livre dont vous connaissez le numéro, imaginons si vous faites un répertoire de tous les livres donc avec une instruction de sélection, lorsque le visiteur clique sur le titre de l'un des livres par exemple "Le spectre aux balles d'or", le détail de celui-ci apparaît.

SELECT \* FROM Livres WHERE IDLivre = 7;

Cette sélection va vous permettre d'avoir les informations concernant le livre :

|  |
| --- |
| ***Livres*** |
| **IDLivre** | **Prix** | **Titre** | **CodeGenre** |
| 7 | 45 | Le spectre aux balles d'or | BD |

SELECT \* FROM Livres **WHERE** IDLivre = 7 **OR** IDLivre = 8;

Bien sûr ce genre de sélection ne s'applique pas seulement sur les identifiants des tables.
Attention : il faut bien choisir entre le AND et le OR lorsque vous voulez faire des sélections multiples, par exemple ici le OR permet de sélectionner deux fois sur le même champ, le AND est utilisé lorsque vous faites une sélection sur différent champ de la table.

SELECT \* FROM Livres WHERE IDLivre = 7 AND Prix > 50;

La sélection dans cette requête prend le livre avec l'ID égal à 7 et d'un prix supérieur à 50. Donc aucun enregistrement ne sera retourné par la requête puisque le livre 7 coûte 45F.

SELECT \* FROM Livres WHERE IDLivre = 7 OR Prix > 50;

Ici le livre 7 et tous les autres livres d'un prix supérieur à 50F sont sélectionnés.

2 - Le mot clé LIKE
Le mot clé LIKE avec le WHERE va vous permettre de faire une sélection "approximative" dans vos tables, par exemple ici vous voulez tous les livres dont le titre commence par 'tou'.

SELECT \* FROM Livres WHERE Titre LIKE 'tou%';

|  |
| --- |
| ***Livres*** |
| **IDLivre** | **Prix** | **Titre** | **CodeGenre** |
| 6 | 80 | Tour de manège | RG |
| 12 | 70 | Toujours aimer | RG |
| 13 | 70 | Toujours aimante | RG |
| 14 | 72 | Toujours aimé | RG |

SELECT \* FROM Livres **WHERE** CodeGenre **LIKE** 'r\_';

|  |
| --- |
| ***Livres*** |
| **IDLivre** | **Prix** | **Titre** | **CodeGenre** |
| 6 | 80 | Tour de manège | RG |
| 10 | 50 | Michel Strogoff | Ro |
| 12 | 70 | Toujours aimer | RG |
| 13 | 70 | Toujours aimante | RG |
| 14 | 72 | Toujours aimé | RG |

Pour remplacer un seul caractère dans un LIKE il faut utiliser \_, pour remplacer une chaîne de caractère il faut utiliser %.
Le LIKE ne respecte pas la casse (minuscule - majuscule) contrairement au égal.

3 - Sélection selon une liste de valeur
Ici nous allons voir un mot clé permettant de faire la sélection à partir d'une liste de valeur. Le WHERE est ici utilisé avec un IN.

SELECT \* FROM Livres WHERE Prix IN (40, 50, 72);

Le mot clé IN permet de sélectionner les enregistrements dans la table Livres où le prix est de 40, 50 ou 72.

|  |
| --- |
| ***Livres*** |
| **IDLivre** | **Prix** | **Titre** | **CodeGenre** |
| 1 | 40 | Le glaive magique | BD |
| 3 | 40 | Gaffes en gros | BD |
| 4 | 40 | Lagaffe nous gâte | BD |
| 9 | 50 | La jeune Parque | Po |
| 10 | 50 | Michel Strogoff | Ro |
| 11 | 50 | La Serpe d'or | BD |
| 14 | 72 | Toujours aimé | RG |

De la même façon il y a le mot clé contraire au IN qui est le NOT IN qui va en fait sélectionner les enregistrements qui n'ont pas l'une des valeurs indiquées sur le champ demandé.

SELECT \* FROM Livres WHERE Prix NOT IN (40, 50, 72);

Cette requête sélectionne donc les livres qui ne valent pas 40, 50 ou 72F.

4 - Sélection selon une plage de valeurs
Nous allons lier dans cette partie le WHERE au BETWEEN...AND qui permet de faire une sélection en précisant une plage de données, ce mot clé n'est utilisé qu'avec des valeurs "numériques" (integer, date...).

SELECT \* FROM Livres WHERE Prix BETWEEN 40 AND 50;

|  |
| --- |
| ***Livres*** |
| **IDLivre** | **Prix** | **Titre** | **CodeGenre** |
| 1 | 40 | Le glaive magique | BD |
| 3 | 40 | Gaffes en gros | BD |
| 4 | 40 | Lagaffe nous gâte | BD |
| 5 | 45 | QRN sur Bretzelburg | BD |
| 7 | 45 | Le spectre aux balles d'or | BD |
| 9 | 50 | La jeune Parque | Po |
| 10 | 50 | Michel Strogoff | Ro |
| 11 | 50 | La Serpe d'or | BD |

S'il on veut par exemple avoir toutes les BD qui ont un prix entre 40 et 50F la requête est la suivante :

SELECT \* FROM Livres WHERE (Prix BETWEEN 40 AND 50) AND (CodeGenre = 'BD');

Les parenthèses ne sont bien sûr pas obligatoires mais c'est pour une question de clarté.

5 - Jointures
Ce système permet de joindre deux tables différentes qui ont un champ en relation, ici la table Livres et la table Genres ont le champ CodeGenre en commun, ces deux tables peuvent donc être jointes grâce au WHERE.

SELECT \* FROM Livres, Genres WHERE Livres.CodeGenre = Genres.CodeGenre;

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **IDLivre** | **Prix** | **Titre** | **Livres.CodeGenre** | **Genres.CodeGenre** | **LibelleGenre** |
| 1 | 40 | Le glaive magique | BD | BD | Bande Dessinée |
| 3 | 40 | Gaffes en gros | BD | BD | Bande Dessinée |
| 4 | 40 | Lagaffe nous gâte | BD | BD | Bande Dessinée |
| 5 | 45 | QRN sur Bretzelburg | BD | BD | Bande Dessinée |
| 6 | 80 | Tour de manège | RG | RG | Roman de gare |
| 7 | 45 | Le spectre aux balles d'or | BD | BD | Bande Dessinée |
| 8 | 30 | La bonne chanson | Po | Po | Poésie |
| 9 | 50 | La jeune Parque | Po | Po | Poésie |
| 10 | 50 | Michel Strogoff | Ro | Ro | Roman |
| 11 | 50 | La Serpe d'or | BD | BD | Bande Dessinée |
| 12 | 70 | Toujours aimer | RG | RG | Roman de gare |
| 13 | 70 | Toujours aimante | RG | RG | Roman de gare |
| 14 | 72 | Toujours aimé | RG | RG | Roman de gare |

Pour joindre deux tables il faut indiquer leur nom dans le FROM et utiliser la syntaxe avec le nom, le point et le champ de jointure.
Le problème avec cette jointure c'est que l'on se retrouve avec deux fois la "même" colonne, car elle vient une première fois de la table Livres et l'autre de la table Genres. Pour éviter cela et avoir quelque chose de plus sympa, la requête peut être modifiée comme ceci :

SELECT Titre, Livres.CodeGenre, LibelleGenre, Prix FROM Livres, Genres WHERE Livres.CodeGenre = Genres.CodeGenre;

|  |  |  |  |
| --- | --- | --- | --- |
| **Titre** | **Livres.CodeGenre** | **LibelleGenre** | **Prix** |
| Le glaive magique | BD | Bande Dessinée | 40 |
| Gaffes en gros | BD | Bande Dessinée | 40 |
| Lagaffe nous gâte | BD | Bande Dessinée | 40 |
| QRN sur Bretzelburg | BD | Bande Dessinée | 45 |
| Tour de manège | RG | Roman de gare | 80 |
| Le spectre aux balles d'or | BD | Bande Dessinée | 45 |
| La bonne chanson | Po | Poésie | 30 |
| La jeune Parque | Po | Poésie | 50 |
| Michel Strogoff | Ro | Roman | 50 |
| La Serpe d'or | BD | Bande Dessinée | 50 |
| Toujours aimer | RG | Roman de gare | 70 |
| Toujours aimante | RG | Roman de gare | 70 |
| Toujours aimé | RG | Roman de gare | 72 |

Pour différencier les deux tables dans le SELECT il faut indiquer le nom de la table sur les champs à afficher qui sont éventuellement en double à cause de la jointure.

6 - Application
Une fois n'est pas coutume, une petite application sur le même exemple ne fait pas de mal pour comprendre encore mieux les tutoriaux.
Nous allons faire une sélection de tous les LibelleGenre et avec cette sélection, en utilisant le WHERE on affichera les livres correspondant.

La requête de sélection des libellés est la suivante :

SELECT \* FROM Genres;

Le script permettant de créer le liste dans une boîte <select> :

|  |  |
| --- | --- |
| 1  | <select name="genre">  |
| 2  | <?php  |
| 3  | mysql\_connect($server, $user, $pass) or die('Erreur de connexion');  |
| 4  | mysql\_select\_db($db) or die('Base inexistante');  |
| 5  | $sql = 'SELECT \* FROM Genres;';  |
| 6  | $query = mysql\_query($sql) or die( 'Erreur' );  |
| 7  | $nb = mysql\_num\_rows($query);  |
| 8  | if ( !$nb[0] ) {  |
| 9  | echo '<option>Aucun genre</option>';  |
| 10  | } else {  |
| 11  | while ( $list = mysql\_fetch\_array( $query ) ) {  |
| 12  | echo '<option value="'.$list['CodeGenre'].'">'.$list['LibelleGenre'].'</option>';  |
| 13  | }  |
| 14  | }  |
| 15  | mysql\_close();  |
| 16  | ?>  |
| 17  | </select>  |

Lorsque le visiteur valide le formulaire on reçoit donc une variable passé en POST sur la page de résultat où l'on va pouvoir faire notre requête de sélection avec le WHERE. Pour l'exemple on va choisir les BD.

SELECT \* FROM Livres WHERE CodeGenre = 'BD';

Dans le script voici ce qui peut se passer :

|  |  |
| --- | --- |
| 1  | <?php  |
| 2  | mysql\_connect($server, $user, $pass) or die('Erreur de connexion');  |
| 3  | mysql\_select\_db($db) or die('Base inexistante');  |
| 4  | $sql = "SELECT \* FROM Livres WHERE CodeGenre = '".$\_POST['genre']."';";  |
| 5  | $query = mysql\_query($sql) or die( 'Erreur' );  |
| 6  | $nb = mysql\_num\_rows($query);  |
| 7  | if ( $nb>0 ) {  |
| 8  | echo 'Aucun livre dans ce genre';  |
| 9  | } else {  |
| 10  | while ( $list = mysql\_fetch\_array( $query ) ) {  |
| 11  | //traitement du résultat  |
| 12  | }  |
| 13  | }  |
| 14  | mysql\_close();  |
| 15  | ?>  |

Et voilà le résultat :

|  |
| --- |
| ***Livres*** |
| **IDLivre** | **Prix** | **Titre** | **CodeGenre** |
| 1 | 40 | Le glaive magique | BD |
| 3 | 40 | Gaffes en gros | BD |
| 4 | 40 | Lagaffe nous gâte | BD |
| 5 | 45 | QRN sur Bretzelburg | BD |
| 7 | 45 | Le spectre aux balles d'or | BD |
| 11 | 50 | La Serpe d'or | BD |

# Exploiter une base MySQL en PHP

## Interroger une table

Maintenant que nous nous sommes connectés à la base de données (exercice 14), nous allons interroger une table pour en extraire les résultats puis les ranger dans un ordre précis. Créons d'abord une table comme ci-dessous.

|  |
| --- |
| **Requête SQL :** CREATE TABLE famille\_tbl ( id int(11) NOT NULL auto\_increment, nom varchar(255) NOT NULL, prenom varchar(255) NOT NULL, statut varchar(255) NOT NULL, date date DEFAULT '0000-00-00' NOT NULL, PRIMARY KEY (id), KEY id (id), UNIQUE id\_2 (id) ); INSERT INTO famille\_tbl VALUES( '', 'Dupond', 'Grégoire', 'Grand-père', '1932-05-17');INSERT INTO famille\_tbl VALUES( '', 'Dupond', 'Germaine', 'Grand-mère', '1939-02-15');INSERT INTO famille\_tbl VALUES( '', 'Dupond', 'Gérard', 'Père', '1959-12-22');INSERT INTO famille\_tbl VALUES( '', 'Dupond', 'Marie', 'Mère', '1961-03-02');INSERT INTO famille\_tbl VALUES( '', 'Dupond', 'Julien', 'Fils', '1985-05-17');INSERT INTO famille\_tbl VALUES( '', 'Dupond', 'Manon', 'Fille', '1990-11-29'); |
|

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|

|  |
| --- |
|  |
| Structure de la table (PhpMyAdmin) : |
| **Champ** | **Type** | **Null** | **Defaut** | **Extra** |
| id  | int(11)  | Non  | 0  | auto\_increment  |
| nom  | varchar(255)  | Non  |   |   |
| prenom  | varchar(255)  | Non  |   |   |
| statut  | tinyint(255)  | Non  | 0  |   |
| date  | date  | Non  | 0000-00-00  |   |

 |
|

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

 |

 |

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

|  |
| --- |
| Code PHP  |
| <?php // on se connecte à MySQL $db = mysql\_connect('localhost', 'login', 'password'); // on sélectionne la base mysql\_select\_db('nom\_de\_la\_base',$db); // on crée la requête SQL $sql = 'SELECT nom,prenom,statut,date FROM famille\_tbl'; // on envoie la requête $req = mysql\_query($sql) or die('Erreur SQL !<br>'.$sql.'<br>'.mysql\_error()); // on fait une boucle qui va faire un tour pour chaque enregistrement while($data = mysql\_fetch\_assoc($req))     {     // on affiche les informations de l'enregistrement en cours     echo '<b>'.$data['nom'].' '.$data['prenom'].'</b> ('.$data['statut'].')';     echo ' <i>date de naissance : '.$data['date'].'</i><br>';     } // on ferme la connexion à mysql mysql\_close(); ?>  |
| Donne à l'écran  |
| **Dupond Grégoire** (Grand-père), *date de naissance : 1932-05-17***Dupond Germaine** (Grand-mère), *date de naissance : 1939-02-15***Dupond Gérard** (Père), *date de naissance : 1959-12-22***Dupond Marie** (Mère), *date de naissance : 1961-03-02***Dupond Julien** (Fils), *date de naissance : 1985-05-17***Dupond Manon** (Fille), *date de naissance : 1990-11-29* |

Explication :

Voici donc notre première boucle sur une table, champagne ! :). Plus sérieusement, vous vous apercevez que les résultats qui s'affichent sont exactement dans le même ordre que la table et pour cause, nous n'avons pas spécifié de condition dans notre requête (2), donc dans ce cas, la requête scanne la table de haut en bas.

Remarquez que la fonction mysql\_fetch\_assoc() renvoie un tableau dont les clés sont les noms des champs sélectionnés. On a aussi rajouté après mysql\_query() ceci:  or die('Erreur SQL !<br>' .$sql. '<br>'. mysql\_error()); . Cela veut dire qu'en cas d'erreur dans la requete vers mysql, ce qui arrive fréquemment, php va afficher un message indiquant l'erreur renvoyée par mysql (grace à mysql\_error()) ce qui fournit une aide précieuse pour comprendre le problème.

Vous notez également que les dates de naissance sont au format US, ceci est normal puisque nous avons défini un type DATE dans notre table, nous verrons plus bas comment convertir les dates US au format FR.

Nous allons maintenant faire plusieurs tests en ne changeant uniquement que la requête SQL (2). Le reste du code ne change pas.

 Affichage des résultats par ordre alphabétique de prénom.

|  |
| --- |
| Le code PHP de la requête |
| <?php // Gardez le code ci-dessus, changez juste la requête SQL !   $sql = 'SELECT nom,prenom,statut,date FROM famille\_tbl ORDER BY prenom'; // L'opérateur ORDER BY permet de classer soit alphabétiquement // soit numériquement suivant le type du champ. // Si l'on souhaite classer en décroissant (ex. de Z à A), nous // y ajouterons DESC soit : ORDER BY prenom DESC ?>  |
| Donne à l'écran |
| **Dupond Gérard** (Père), *date de naissance : 1959-12-22***Dupond Germaine** (Grand-mère), *date de naissance : 1939-02-15***Dupond Grégoire** (Grand-père), *date de naissance : 1932-05-17***Dupond Julien** (Fils), *date de naissance : 1985-05-17***Dupond Manon** (Fille), *date de naissance : 1990-11-29***Dupond Marie** (Mère), *date de naissance : 1961-03-02* |

Affichage des résultats par comparaison de date.

|  |
| --- |
| Le code PHP de la requête |
| <?php // Gardez le code ci-dessus, changez juste la requête !   $sql = "SELECT nom,prenom,statut FROM famille\_tbl WHERE date<'1960-01-01'"; // L'avantage d'avoir un type DATE dans notre base de données, c'est que // nous pouvons comparer des dates dans la requête SQL. // Ici nous ne souhaitons afficher que les membres de la famille qui sont // nés avant le 1er janvier 1960, soit : WHERE date<'1960-01-01' ?>  |
| Donne à l'écran |
| **Dupond Grégoire** (Grand-père), *date de naissance : 1932-05-17***Dupond Germaine** (Grand-mère), *date de naissance : 1939-02-15***Dupond Gérard** (Père), *date de naissance : 1959-12-22* |

 Affichage des résultats avec le commande LIKE.

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

|  |
| --- |
| Le code PHP de la requête |
| <?php // Gardez le code ci-dessus, changez juste la requête !   $sql = "SELECT nom,prenom,statut,date FROM famille\_tbl WHERE prenom LIKE 'G%'"; // Le signe pourcentage "%" placé après le G, indique que la lettre G peut // être suivie, mais pas précédée, d'autres caractères ! // Notez aussi que LIKE n'est pas sensible à la casse, cela veut dire que // la requête cherchera aussi bien des G majuscules que minuscules. ?>  |
| Donne à l'écran |
| **Dupond Grégoire** (Grand-père), *date de naissance : 1932-05-17***Dupond Germaine** (Grand-mère), *date de naissance : 1939-02-15***Dupond Gérard** (Père), *date de naissance : 1959-12-22*  |

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

|  |
| --- |
| Le code PHP de la requête |
| <?php // Gardez le code ci-dessus, changez juste la requête !   $sql = "SELECT \* FROM famille\_tbl WHERE prenom LIKE '%MA%'"; // Le signe pourcentage "%" placé avant et après MA indique que la syllabe // peut-être précédée ou suivie de caractères. // Une fois de plus notez que LIKE n'est pas sensible à la casse, la requête // cherchera aussi bien des MA majuscules que des ma en minuscules. ?>  |
| Donne à l'écran |
| **Dupond Marie** (Mère), *date de naissance : 1961-03-02***Dupond Manon** (Fille), *date de naissance : 1990-11-29* |

Bien sûr, il est possible de mettre plusieurs conditions dans la même requête, exemple :

 SELECT \* FROM famille\_tbl WHERE prenom LIKE '%MA%' AND date<'1960-01-01' ORDER BY prenom

Pour terminer voici comment vous allez pouvoir convertir la date du format US au format FR, une fois que vous avez récupéré l'information depuis la table. Nous vous donnons deux façons de faire, la deuxième est bien meilleure !

|  |
| --- |
| Le code PHP (déconseillé pour ce cas précis) |
| <?php  // on se connecte à MySQL $db = mysql\_connect('localhost', 'login', 'password');  // on sélectionne la base mysql\_select\_db('nom\_de\_la\_base',$db);  // on crée la requete SQL $sql = 'SELECT nom,prenom,statut,date FROM famille\_tbl';  // on envoie la requête $req = mysql\_query($sql) or die('Erreur SQL !<br>'.$sql.'<br>'.mysql\_error());  // on fait une boucle qui va faire un tour pour chaque enregistrement while($data = mysql\_fetch\_array($req))      {     $a = substr($data['date'], 0, 4);     // conversion     $m = substr($data['date'], 5, 2);     // de la date     $j = substr($data['date'], 8, 2);     // au format     $date = $j.'-'.$m.'-'.$a;             // Français            // on affiche les informations de l'enregistrement en cours     echo '<b>'.$data['nom'].' '.$data['prenom'].'</b> ('.$data['statut'].')';      echo ' <i>date de naissance : '.$date.'</i><br>';      }  // on ferme la connexion à mysql mysql\_close();  ?>   |
| Le code PHP (bien meilleur, utilisant les ressources de MySQL) |
| <?php  // on se connecte à MySQL $db = mysql\_connect('localhost', 'login', 'password');  // on seléctionne la base mysql\_select\_db('nom\_de\_la\_base',$db);  // on crée la requete SQL $sql = "SELECT nom,prenom,statut,DATE\_FORMAT(date, '%d-%m-%Y') as datefr FROM famille\_tbl";  // on envoie la requête $req = mysql\_query($sql) or die('Erreur SQL !<br>'.$sql.'<br>'.mysql\_error());  // on fait une boucle qui va faire un tour pour chaque enregistrement while($data = mysql\_fetch\_array($req))      {     // on affiche les informations de l'enregistrement en cours     echo '<b>'.$data['nom'].' '.$data['prenom'].'</b> ('.$data['statut'].')';      echo ' <i>date de naissance : '.$data['datefr'].'</i><br>';      }  // on ferme la connexion à mysql mysql\_close();  ?>   |
| Donne à l'écran, dans les deux cas |
| **Dupond Grégoire** (Grand-père), *date de naissance : 17-05-1932***Dupond Germaine** (Grand-mère), *date de naissance : 15-02-1939***Dupond Gérard** (Père), *date de naissance : 22-12-1959***Dupond Marie** (Mère), *date de naissance : 02-03-1961***Dupond Julien** (Fils), *date de naissance : 17-05-1985***Dupond Manon** (Fille), *date de naissance : 29-11-1990* |

## Alimenter une ou plusieurs tables

Dans cet exercice nous allons voir comment alimenter une ou plusieurs tables avec les données qui proviennent d'un même formulaire.

### Alimenter une table

  Pour commencer vous allez créer la table infos\_tbl dans phpMyAdmin comme suit :

requête SQL:
CREATE TABLE infos\_tbl (id INT (11) not null AUTO\_INCREMENT, nom VARCHAR (35) not null , prenom VARCHAR (35) not null , email VARCHAR (70) not null , icq INT (11) null , titre VARCHAR (70) not null , url VARCHAR (255) not null , PRIMARY KEY (id), INDEX (id), UNIQUE (id))

Ensuite nous allons utiliser le formulaire ci dessous qui va alimenter la table :

|  |  |
| --- | --- |
| Code HTML | Donne à l'écran |
| <html><form method="POST" action="add.php"><center><input type="text" name="nom" size="20" value="nom" maxlength="35"> <input type="text" name="prenom" size="20" value="prenom" maxlength="35"><br><input type="text" name="email" size="20" value="email" maxlength="70"><br><input type="text" name="titre" size="20" value="titre du site" maxlength="70"> <input type="text" name="url" size="20" value="url du site" maxlength="255"><br><input type="submit" value="Envoyer" name="envoyer"></center></form></html> | Haut du formulaireBas du formulaire |

Vous noterez les "maxlength" dans chacun des champs, ceci permet de brider le nombre de caractères maximum que le visiteur peut entrer dans le champ, bien sûr ici le "maxlength" correspond au nombre de caractères spécifié dans la création de la table infos\_tbl. Cela a un intérêt, celui d'être sûr que le visiteur ne tapera pas plus de caractères que prévu.

Voyons maintenant le script PHP en lui-même, celui-ci sera contenu dans le fichier add.php auquel fait référence le POST du formulaire :

|  |
| --- |
| Code PHP de "add.php" |
| <?php // On commence par récupérer les champs if(isset($\_POST['nom']))      $nom=$\_POST['nom'];else      $nom="";if(isset($\_POST['prenom']))      $prenom=$\_POST['prenom'];else      $prenom="";if(isset($\_POST['email']))      $email=$\_POST['email'];else      $email="";if(isset($\_POST['icq']))      $icq=$\_POST['icq'];else      $icq="";if(isset($\_POST['titre']))      $titre=$\_POST['titre'];else      $titre="";if(isset($\_POST['url']))      $url=$\_POST['url'];else      $url="";// On vérifie si les champs sont vides if(empty($nom) OR empty($prenom) OR empty($email) OR empty($titre) OR empty($url))     {     echo '<font color="red">Attention, seul le champs <b>ICQ</b> peut rester vide !</font>';     } // Aucun champ n'est vide, on peut enregistrer dans la table else          {        // connexion à la base$db = mysql\_connect('localhost', 'login', 'password')  or die('Erreur de connexion '.mysql\_error());// sélection de la base      mysql\_select\_db('nom\_de\_la\_base',$db)  or die('Erreur de selection '.mysql\_error());          // on écrit la requête sql     $sql = "INSERT INTO infos\_tbl(id, nom, prenom,email, icq, titre, url) VALUES('','$nom','$prenom','$email','$icq','$titre','$url')";          // on insère les informations du formulaire dans la table     mysql\_query($sql) or die('Erreur SQL !'.$sql.'<br>'.mysql\_error());     // on affiche le résultat pour le visiteur     echo 'Vos infos on été ajoutées.';     mysql\_close();  // on ferme la connexion     }  ?>  |

Explication :

La requête INSERT INTO permet donc l'insertion des champs du formulaire dans la table. Dans notre cas le premier champ reste vide car il s'agit de l'id (identifiant) qui s'incrémente automatiquement à chaque nouvelle requête INSERT.

Notez que dans le cas ou vous ne voudriez insérer que les champs Nom et Prénom dans la table vous pouvez très bien spécifier dans la requête le nom de chaque champ, comme suit :

$sql = "INSERT INTO infos\_tbl(nom,prenom) VALUES('$nom','$prenom')";

Mais attention dans ce cas les autres champs de la table devront avoir l'attribut NULL et non pas NOT NULL, Null indique au champ qu'il pourra rester vide. Et si vous essayez d'insérer un nombre de valeurs différent du nombre des champs, mysql vous fera un joli message d'errreur !

Bien sûr, l'idéal et de tester si l'URL existe dans la table pour éviter les doublons, si celle-ci existe déjà on indique au visiteur qu'il ne peut pas valider son formulaire. J'ai pris l'exemple ici de l'URL mais il va sans dire que vous pouvez vérifier les champs de votre choix, cela dépend des doublons que vous ne souhaitez pas avoir dans votre table. Voici le code :

|  |
| --- |
| Code PHP de "add.php" avec la vérification de doublon sur l'URL |
| <?php // On récupère les champs (voir plus haut)// On vérifie si les champs sont vides if(empty($nom) OR empty($prenom) OR empty($email) OR empty($titre) OR empty($url))     {     echo '<font color="red">Attention, seul le champs <b>ICQ</b> peut rester vide !</font>';     } // Aucun champ n'est vide, on peut enregistrer dans la table else          {     $db = mysql\_connect('localhost', 'login', 'password');  // connexion à la base     mysql\_select\_db('nom\_de\_la\_base',$db);                  // sélection de la base          // on regarde si l'url existe déjà     $sql = "SELECT id FROM infos\_tbl WHERE url='$url'";     $req = mysql\_query($sql) or die('Erreur SQL !'.$sql.'<br>'.mysql\_error());           // on compte le nombre de résultats     $res = mysql\_num\_rows($req);     if($res!=0)  // l'url existe déjà, on affiche un message d'erreur         {         echo '<font color="red">Désolé, mais cette URL existe déjà dans notre base.</font>';         }     else  // L'url n'existe pas, on insère les informations du formulaire dans la table         {            $sql = "INSERT INTO infos\_tbl(id, nom, prenom,email, icq, titre, url) VALUES('','$nom','$prenom','$email','$icq','$titre','$url')";         mysql\_query($sql) or die('Erreur SQL !'.$sql.'<br>'.mysql\_error());                   // on affiche le résultat pour le visiteur         echo 'Vos infos on été ajoutées.';         }     mysql\_close();  // on ferme la connexion     }  ?>  |

### Alimenter deux tables et créer une liaison

  Voyons maintenant comment ces mêmes informations peuvent être enregistrées dans deux tables différentes en gardant une liaison entre les deux. Le principe est en fait tout bête, admettons que nous ayons une première table qui va nous servir à stocker les coordonnées du visiteur ($nom , $prenom, $email, $icq) et une seconde dans laquelle ne seront sauvegardées que les informations du site ($titre et $url). Voyez ce petit schéma qui va vous éclairer sur la manière de créer une liaison entre 2 tables :



Examinons la méthode à employer :

1. On commence par insérer $titre et $url dans la table 2 (sites\_tbl).
2. Une fois l'insertion effectuée, on utilise la fonction PHP mysql\_insert\_id() pour connaître l'id\_site qui a été affecté à notre nouvel enregistrement.
3. On insère le reste du formulaire dans la table 1 (infos\_tbl), soit : $nom , $prenom, $email, $icq et $id\_site.

Nous avons maintenant un liaison entre la table 1 et 2 via l'id\_site. Ci-dessous le code PHP de cette manipulation :

|  |
| --- |
| Code PHP avec création de la liaison entre les 2 tables |
| <?php // On récupère les champs (voir plus haut)$db = mysql\_connect('localhost', 'login', 'password');  // connexion à la base mysql\_select\_db('nom\_de\_la\_base',$db);                  // sélection de la base // on regarde dans la table SITES\_TBL si l'url existe déjà $sql = "SELECT id FROM sites\_tbl WHERE url='$url'"; $req = mysql\_query($sql) or die('Erreur SQL !'.$sql.'<br>'.mysql\_error()); $res= mysql\_num\_rows($req); if($res!=0)  // l'url existe déjà, on affiche un message d'erreur     {     echo '<font color="red">Désolé, mais cette URL existe déjà dans notre base.</font>';     } else  // L'url n'existe pas, on insère d'abord les infos dans SITES\_TBL     {     $sql = "INSERT INTO sites\_tbl(id\_site, titre, url) VALUES('','$titre','$url')";     mysql\_query($sql) or die('Erreur SQL !'.$sql.'<br>'.mysql\_error());     // on récupère l'id\_site qui vient d'être généré     $id\_site = mysql\_insert\_id();     // ci-dessous on insère les infos dans INFOS\_TBL     $sql = "INSERT INTO infos\_tbl(id, nom, prenom,email, icq, id\_site) VALUES('','$nom','$prenom','$email','$icq','$id\_site')";     mysql\_query($sql) or die('Erreur SQL !'.$sql.'<br>'.mysql\_error());     } mysql\_close($db);  // on ferme la connexion ?>   |

Important : L'id\_site de la table 2 (sites\_tbl) doit être en auto\_incrément automatique au même titre que l'id de la table 1 (infos\_tbl). Je ne vous donne "volontairement" pas la requête SQL pour créer les deux tables, et ce pour vous faire travailler un peu tout de même ;).

Conclusion :

Vous pouvez créer autant de liaisons que vous le voulez entre vos tables, cela a pour but de les alléger ce qui permet une plus grande rapidité lorsque vous souhaitez les interroger.

# Les sessions

Beaucoup d'entre-vous se posent la question de savoir comment passer leur variables de pages en pages, ou encore comment conserver certaines informations pendant la durée d'une visite sur leur site. Les sessions devraient répondre à leurs attentes.

## Principe de fonctionnement

Une session est en fait un fichier conservé sur le serveur et accessible à vos scripts en fonction d'un identifiant généré à la création. Chaque fois qu'un de vos visiteurs génère une session, un identifiant lui est attribué. Tout ce qui est dans cette session est accessible de partout à vos scripts. On comprend dès lors très vite tout l'intérêt de la chose. En effet, si chaque session est propre à un visiteur, on peut personnaliser nos scripts en fonction du visiteur, ou encore alléger nos requêtes : plutôt que d'aller chercher un pseudonyme dans la base de donnée sur chaque page, vous le rapatriez à l'arrivée du visiteur sur le site, vous le stockez en session et c'est de là que vous y accéderez par la suite.

La session est finalement l'équivalent d'un cookie en plus sécurisé. En effet, étant stocké sur le serveur et non chez le client, elle est plus difficile d'accès aux éventuels pirates. Mais un risque demeure, prenez donc l'habitude de ne pas y stocker des informations trop sensibles. N'oubliez pas également que la session, à la différence du cookie, n'est valable qu'un temps limité (aux alentours des 30 minutes, mais cela dépend de la configuration de votre serveur), et est automatiquement détruite à la fermeture du navigateur du visiteur.

Vous pouvez enregistrer en session tout type de variable : du simple numérique au tableau en passant par la chaîne de caractères.

Il existe plusieurs fonctions liées aux sessions, mais deux seulement sont essentielles à leur fonctionnement, ainsi qu'un tableau :

session\_start();
Cette fonction sert à démarrer une session OU appeler la session existante. Elle doit donc être présente sur toutes les pages de votre site. A savoir que cette fonction ne tolère pas d'envoi au navigateur avant elle. Plus simplement il ne faut aucune sortie avant elle : pas de html, pas de echo. Prenez l'habitude de la placer au début du fichier, avant quoi que ce soit d'autre et tout se passera bien.

session\_destroy();
Cette fonction détruit la session en cours. Mais elle ne détruit pas les variables de sessions associées à la session courante. Nous verrons plus loin comment détruire une session complètement et proprement.

$\_SESSION
Il s'agit du tableau global contenant toutes les variables de sessions pour la session courante. Son utilisation est exactement identique aux tableaux habituels, à savoir :

|  |
| --- |
| Utilisation du tableau global $\_SESSION |
| **// Nous allons créer une variable de session appellée "nombre" et qui contient "1234" :$\_SESSION['nombre'] = 1234;****// Nous pouvons maintenant afficher la variable "nombre"echo $\_SESSION['nombre'];  // Ceci va afficher 1234** |

## Etude de cas

Nous allons maintenant étudier un cas concret à savoir une procédure de login. En effet c'est une cas répandu d'utilisation des sessions. Nous allons donc créer un formulaire qui permet au visiteur de se loguer, puis une page de vérification qui va consulter la base de données et vérifier que les informations de connexion sont correctes. Et enfin mettre les données utiles sur l'utilisateur en session afin de pouvoir les réutiliser sur d'autres pages du site.

|  |  |
| --- | --- |
| Le code HTML de la page de connexion  | Le rendu |
| <html> <head>  <title>Connexion au site</title> </head> <body>  <form method="post" action="verifLogin.php">   <table border="0" width="400" align="center">    <tr>     <td width="200"><b>Vôtre login</b></td>     <td width="200">      <input type="text" name="login">     </td>    </tr>    <tr>     <td width="200"><b>Vôtre mot de passe<b></td>     <td width="200">      <input type="password" name="password">     </td>    </tr>    <tr>     <td colspan="2">      <input type="submit" name="submit" value="login">     </td>    </tr>    </table>  </form> </body></html> | Haut du formulaire

|  |  |
| --- | --- |
| **Votre login** |  |
| **Votre mot de passe** |  |
|  |

Bas du formulaire |

Rien à dire ici, si vous ne comprenez pas cette étape, il va falloir commencer par apprendre le HTML (quoique là le code en question n'est pas fantastique, mais c'est pour l'exemple :p). Nous allons maintenant passer à la récupération des données et au test de celles-ci.

|  |
| --- |
| Code de la page verifLogin.php |
| **<?php// On démarre la sessionsession\_start();$loginOK = false;  // cf Astuce// On n'effectue les traitement qu'à la condition que // les informations aient été effectivement postéesif ( isset($\_POST) && (!empty($\_POST['login'])) && (!empty($\_POST['password'])) ) {  extract($\_POST);  // je vous renvoie à la doc de cette fonction  // On va chercher le mot de passe afférent à ce login  $sql = "SELECT pseudo, age, sexe, ville, mdp FROM user WHERE login = '".addslashes($login)."'";  $req = mysql\_query($sql) or die('Erreur SQL : <br />'.$sql);    // On vérifie que l'utilisateur existe bien  if (mysql\_num\_rows($req) > 0) {     $data = mysql\_fetch\_assoc($req);        // On vérifie que son mot de passe est correct    if ($password == $data['mdp']) {      $loginOK = true;    }  }}// Si le login a été validé on met les données en sessionsif ($loginOK) {  $\_SESSION['pseudo'] = $data['pseudo'];  $\_SESSION['age'] = $data['age'];     $\_SESSION['sexe'] = $data['sexe'];  $\_SESSION['ville'] = $data['ville'];}else {  echo 'Une erreur est survenue, veuillez réessayer !';** **}?>** |  |

Il n'y a pas grand chose à expliquer ici. Si vous ne comprenez pas quelque chose, c'est qu'il vous faut relire les tutos précédents. Donc si l'utilisateur est logué, nous avons ses informations à disposition partout sur le site.

Astuce : la petite astuce que j'ai utilisée est en fait l'inverse du principe d'optimisme. Ce dernier veut qu'on part d'un constat qui dit que tout va bien et en cas de problème on gère l'erreur. Ici c'est le contraire. On part de l'idée que le login ne fonctionne pas en initialisant une variable à false. Et c'est seulement lorsque le login est confirmé qu'on modifie cette variable à true. Il suffit ensuite de tester cette variable pour loguer l'utilisateur ou pas. Cela nous économise des else inutiles à chaque étape de test.
Note : Remarquez que je n'ai effectué que les tests minimums sur le login avant de lancer la requête. Il conviendrait d'effectuer plus de tests afin d'éviter d'éventuels tentatives d'intrusions dans votre application. Mais ce n'est pas le sujet étudié ici. De même que pour la base de données, je vous épargne la création de la table, il s'agit d'une bête table à 5 colonnes...

Imaginons maintenant que nous voulions afficher ces informations plus tard, sur une autre page. Sans les sessions nous aurions dû faire une requête qui retournerait chercher le tout, tandis que là, il nous suffit de faire ça :

|  |
| --- |
| Code de la page affichInfo.php |
| **<?php// On appelle la sessionsession\_start();// On affiche une phrase résumant les infos sur l'utilisateur courantecho 'Pseudo : ',$\_SESSION['pseudo'],'<br />     Age : ',$\_SESSION['age'],'<br />     Sexe : ',$\_SESSION['sexe'],'<br />     Ville : ',$\_SESSION['ville'],'<br />';****?>** |  |

Notez que le session\_start() est toujours présent. En effet il faut préciser au script que nous allons utiliser une session. C'est l'interpréteur qui va déterminer s'il faut en démarrer une ou sélectionner celle existante.
Bien entendu ce script n'a qu'une utilté réduite, à vous de voir quelles pourraient être les multiples façons de profiter des sessions.

# Déconnexion de session

Il s'agit là d'un point important. En effet une session reste ouverte tant que le visiteur ne ferme pas son navigateur. Vous devez lui offrir la possibilité de se déloguer d'une autre manière. Il vous suffit en fait de faire un lien appelé par exemple "déconnexion" qui pointe sur un script contenant ceci :

|  |
| --- |
| Code de la page logout.php |
| **<?php// On appelle la sessionsession\_start();// On écrase le tableau de session$\_SESSION = array();// On détruit la sessionsession\_destroy();****?>** |  |

Voilà vous connaissez l'essentiel sur les sessions. Faites-en bon usage...

# Librairies et API en PHP

Une librairie (ou bibliothèque ou script API) est un script PHP « tout fait » qui permet de gagner du temps dans le développement puisqu’il suffit de l’intégrer au projet en cours.

**Quelques bibliothèques :**

PDO : Interface de connexion aux bases de données (devrait être intégrée à PHP 6)

PHPGuru et PHPMailer : Gestion des e-mails

JPGraph et Artishow : Génération et manipulation de graphiques

PHPDig : Moteur de recherche

FPDF et ezPDF : Génération de fichiers PDF