
Support de cours PHP

Sommaire

- 0. Introduction
 - 1. Afficher une phrase ou une image
 - 2. Afficher la date et l'heure
 - 3. PHP dans du code HTML
 - 4. La concaténation
 - 5. Récupérer les valeurs d'un formulaire
 - 6. Les structures de contrôles
 - 7. Ecrire et lire dans un fichier texte
 - 8. Les fonctions utilisateurs
 - 9. Les variables d'environnement
 - 10. Quelques fonctions utiles
 - 11. SQL/MySQL (Create, Alter & Drop)
 - 12. SQL/MySQL (Insert et Select)
 - 13. SQL/MySQL (Delete et Update)
 - 14. SQL/MySQL (Where)
 - 15. Fonctions PHP pour mySQL
 - 16. Interroger une table MySQL
 - 17. Alimenter une ou plusieurs tables mySQL
 - 18. Les pseudos-frames
 - 19. Les sessions php4
 - 20. Affichage page par page
 - 21. Ca marche pas ?
 - 22. Variables globales à OFF

Introduction

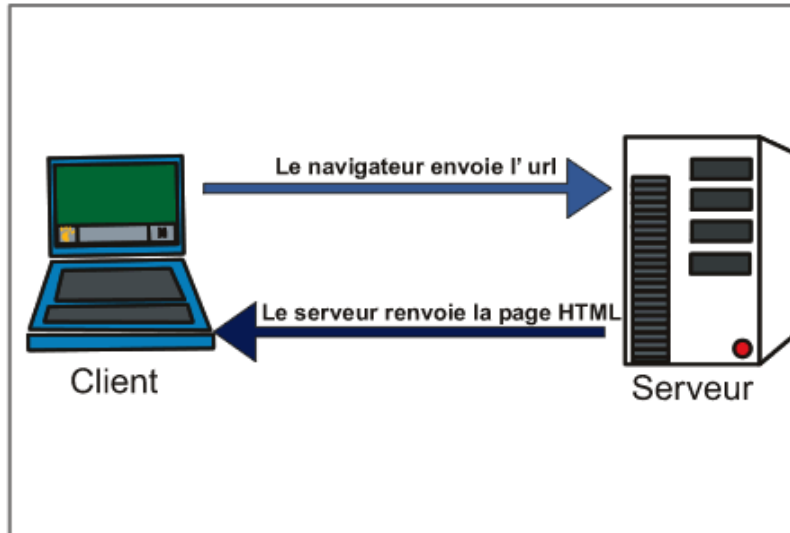
Qu'est-ce que PHP ?

PHP est un langage de programmation qui s'intègre dans vos pages HTML. Il permet entre autres de rendre automatiques des tâches répétitives, notamment grâce à la communication avec une base de données (utilisation la plus courante de PHP). Le but des exercices de *phpDébutant* est de vous apprendre à maîtriser les bases de ces deux outils

(PHP et base de données), afin que vous puissiez élaborer vos propres applications.

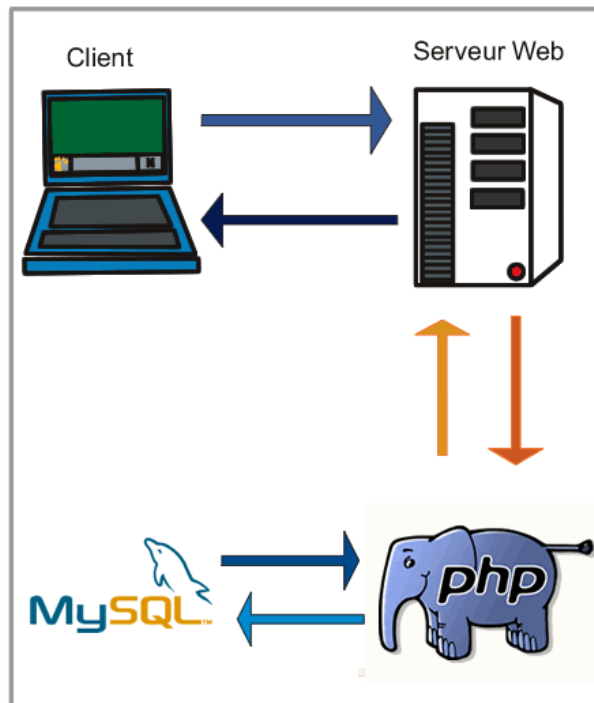
Mais, avant de continuer, il est bon d'expliquer comment se déroule une requête HTTP : en clair que se passe-t-il lorsque vous tapez une adresse dans votre navigateur, où interviennent PHP et MySQL (la base de données), et les conséquences que cela implique.

Voici, en simplifiant, ce qu'il se passe lorsque vous consultez une page html dite statique :



- Votre navigateur envoie l'adresse URL (Uniform Resource Locator) que vous avez tapée
- Le serveur web est un "ordinateur" présent sur l'Internet et qui héberge la page que vous demandez. Sur ce serveur on trouve Apache, logiciel apte à traiter les requêtes HTTP que vous envoyez lorsque vous demandez une page web. Apache va donc chercher le fichier demandé dans son arborescence et renvoie à votre navigateur la page HTML
- Votre navigateur interprète les différents langages se trouvant dans ce fichier (HTML, JavaScript, CSS, etc ...) et affiche la page.

Maintenant, voyons ce qui se passe lorsque votre page HTML contient du code PHP :



- Votre navigateur envoie l'adresse que vous avez tapée
- Le serveur web cherche dans son arborescence si le fichier existe, et si celui-ci porte une extension reconnue comme une application PHP (.PHP, .PHP3, .PHP4 par exemple). Si c'est le cas, le serveur web transmet ce fichier à PHP.
- PHP parse le fichier, c'est-à-dire qu'il va analyser et exécuter le code PHP qui se trouve entre les balises `<?PHP et ?>`. Si ce code contient des requêtes vers une base de données MySQL, PHP envoie la requête SQL. La base de données renvoie les informations voulues au script qui peut les exploiter (pour les afficher par exemple).
- PHP continue de parser la page, puis retourne le fichier dépourvu du code PHP au serveur web.
- Le serveur web renvoie donc un fichier ne contenant plus de PHP, donc seulement du HTML au

navigateur qui l'interprète et l'affiche.

Vous remarquez donc que PHP s'exécute côté serveur. Il n'y a plus aucune trace du code PHP lorsque vous regardez le code source de la page dans votre navigateur PHP.

La base de données la plus couramment utilisée avec PHP est sans aucun doute MySQL. A quoi sert une base de données ? Lorsque vous allez produire des informations dans votre script PHP, vous devez les stocker quelque part. Si ce n'est pas le cas, elles seront alors perdues lorsque le serveur renverra la page html au client (votre navigateur). Pour les stocker, il existe deux solutions: la première consiste à les enregistrer dans un fichier texte sur le serveur (quelque part dans l'arborescence de votre hébergement), la seconde à les enregistrer dans une base de données. La sauvegarde dans un fichier texte n'est pas l'idéal, notamment lorsque vous souhaitez chercher, modifier ou supprimer une partie de l'information que vous stockez. Les bases de données ont été conçues dans cette optique là. Vous verrez dans quelques exercices comment SQL permet de traiter l'information.

Utiliser PHP sur son ordinateur :

Pourquoi installer PHP sur son ordinateur ? Pour tester vos scripts PHP, vous allez être amené à les envoyer sur votre hébergeur, sur Internet. Cependant il devient vite très lourd de sans cesse renvoyer ces fichiers par FTP. C'est pourquoi installer un serveur web sur son ordinateur est utile, et permet de tester ses scripts plus simplement. Concrètement, votre ordinateur sera à la fois client et serveur. Ainsi vous pourrez programmer en PHP sans avoir besoin d'être connecté à Internet, ce qui peut être utile pour les personnes ne disposant pas de connexions illimitées.

Alors que pour tester des pages web html en local, il suffit d'ouvrir le fichier dans un navigateur, il faut un serveur web sur votre PC local pour tester une page PHP. Pour cela, il existe un utilitaire très pratique qui installera Apache, le serveur web le plus utilisé, PHP, MySQL Vous le trouverez sur www.easyPHP.org. Son installation et son utilisation sont très simples et détaillées ici. De plus une FAQ est disponible ici.

Si vous êtes sous linux, la plupart des distributions (RedHat, Mandrake, etc ...) installent par défaut Apache, PHP et MySQL. Un article sur Lea-linux indique comment installer Apache, PHP et Mysql si cela n'a pas été fait. Il se trouve ici.

De plus, une application similaire à EasyPHP existe pour Linux, il s'agit de Linux Easy Installer.

De plus, la documentation officielle (cf lien ci-après) contient une partie consacrée à l'installation.

Quel outil pour faire du PHP ?

Pour faire du PHP, il ne vous faut rien d'autre qu'un simple éditeur de texte. Vous en trouverez une liste ici ainsi qu'un comparatif plutôt complet ici.

Mais l'outil le plus indispensable est certainement la documentation officielle. Celle-ci est disponible ici. Elle contient la description complète des fonctions, la syntaxe de PHP, comment l'installer ... C'est un mine d'or, prenez le temps d'y jeter un oeil, et en cas de doute c'est ici que vous devez chercher en premier.

Enfin sachez utiliser ces merveilleux outils que sont les moteurs de recherche, et plus particulièrement Google. En cherchant bien vous trouverez facilement des ressources (documentation, tutorial, article, ...) dont vous aurez besoin. Le but pour vous est de pouvoir être autonome lorsque vous développerez et de savoir se débrouiller seul face à un problème.

Notions essentielles !

Il est très important de comprendre ce qui suit :

- Il faut bien distinguer le client et le serveur (imaginez tout bêtement la scène dans un bar). Votre navigateur est le client. C'est lui qui demande la page web que vous avez entré. Le serveur est l'ordinateur sur l'Internet qui héberge cette page web. PHP s'exécute donc côté serveur. Cependant, quand PHP envoie une requête SQL au serveur MySQL, il est alors client, vous saisissez ? :)
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 à 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
- PHPMyAdmin n'est pas une base de données ! Il s'agit simplement d'un script PHP qui permet d'administrer vos bases de données MySQL

Exemples et contre-exemples

Voyons maintenant des contre-exemples :

- *Je souhaite qu'une 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.*
Ici, PHP ne peut pas gérer cela car il s'exécute côté serveur.
- *Je souhaite faire un chat en php.*
En effet 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.
- *Je souhaite avoir mon café prêt tout les matins à 7 heures.*
Désolé, je crois que cela ne va pas être possible :)

En revanche, voici une liste non-exhaustive des cas où php pourra solutionner vos problèmes :

- Automatiser la gestion de news, d'article ou autre élément de votre site qui a un caractère répétitif.
- Réaliser des webmails afin de gérer ses mails partout.

- Mettre à disposition des autres sites automatiquement des informations de son site (via la 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.

Un petit tour sur des sites comme phpscripts-fr.net, phpapps.org ou encore hotscripts.com vous donnera un aperçu de ce qui peut être fait en PHP.

Le conseil gratuit

Quand vous essaieriez de comprendre les exemples du site et que vous voudriez faire un script similaire mais adapté à vos besoins, ne commencez pas à 0. Commencez par essayer l'exemple chez vous en le recopiant tel quel, vérifiez qu'il fonctionne, et ensuite modifiez-le petit bout par petit bout en testant à chaque fois, afin d'arriver à ce que vous souhaitez. C'est valable pour php mais pour n'importe quel autre langage ou problème auquel vous vous attaquez.

Le carnet d'adresse du débutant

- | Site officiel d'Apache
- | Site officiel de PHP
- | Site officiel de MySQL
- | Site officiel d'EasyPHP
- | Comment installer et utiliser easyPHP
- | Questions fréquemment posées sur easyPHP
- | Pour installer Apache, PHP, Mysql sous Linux
- | Linux Easy Installer, similaire à EasyPHP, mais pour Linux
- | Liste de quelques éditeurs de textes
- | Comparatif d'éditeurs de texte
- | La documentation PHP, l'outil indispensable !
- | Pour tout trouver sur le web : demander à Google
- | Le repertoire francais de dmoz.org sur PHP
- | phpscripts-fr.net, phpapps.org, hotscripts.com pour trouver des applications

Vous voilà fin prêt pour commencer à apprendre PHP, bon courage :)

Afficher une phrase ou une image

Créer le fichier

Ouvrez votre éditeur préféré (voir tutorial "Introduction"), et créez un nouveau fichier PHP. Le code PHP est toujours encadré par des balises le signalant. Les balises possibles sont :

- | `<?php ?>`
- | `<? ?>`
- | `<% %>`
- | `<script language="php"> </script>`

Les plus couramment utilisés sont `<? ?>`, que vous trouverez dans beaucoup de scripts, même si elles ne sont pas les plus correctes. L'idéal pour éviter des problèmes futurs est d'utiliser les plus correctes : `<?php ?>`. Celles en `<% %>` sont à fuir le plus souvent possible, sauf en cas de nécessité de compatibilité avec un éditeur d'asp.

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 une **PARSE ERROR** (voir tutorial "Ca marche pas ?") lors de l'exécution de votre fichier.

Code PHP	Donne comme résultat à l'écran
<code><?php echo 'Bonjour le monde !'; ?></code>	Bonjour le monde !


C'est la construction `echo` que nous utiliserons pour afficher du texte à l'écran. Ici on voit bien que la phrase n'est pas du tout formatée, voici donc comment l'on peut utiliser les balises HTML dans PHP (ci-dessous).

Code PHP	Donne comme résultat à l'écran
<code><?php echo 'Bonjour le monde !'; ?></code>	Bonjour le monde !

Voilà nous avons ajouté la balise font en HTML pour formater le texte. En fait PHP ne fait pas le formatage, il faut utiliser html pour ça. PHP ne génère que du texte, et en général on veut que ce texte soit une source html, mais aussi quelque fois du javascript, de la feuille de style, voir du xml, etc... Bref, tout ce qui est du texte :)

Affichons maintenant une image en plus du texte.

|

Code PHP	Donne comme résultat à l'éc
<pre><?php echo '<div align="center"> Bonjour le monde !
'; echo '</div>'; ?></pre>	<p>Bonjour le monde !</p> 

On utilise simplement une balise image () du html. On voit donc clairement qu'il est indispensable de maîtriser le HTML avant de vouloir coder en PHP.

Différences entre les navigateurs

Il est important de ne pas oublier que Internet Explorer n'est pas le seul navigateur, il existe aussi notamment Mozilla, Opera, Netscape, Galeon, Phoenix, etc... Il est clair que Internet Explorer est le plus permissif de tous, et affichera parfois correctement des pages qui n'apparaîtront carrément pas sous Netscape (qui est le plus strict).

Vous devrez donc faire particulièrement attention à bien refermer les balises, dans le bon ordre, ainsi que respecter la norme. Il est fortement conseillé que vous testiez vos scripts sous netscape plutôt que sous IE, même si l'idéal est de le faire en même temps sous les deux.

Différentes fonctions

Il existe deux fonctions pour l'affichage : `echo` et `print()`. La première est en fait une construction du langage php, et n'a pas besoin de parenthèses, contrairement à `print`, et peut prendre plusieurs paramètres, séparés par des ",". `echo` est donc légèrement plus rapide que `print`, c'est d'ailleurs pour ça que nous la privilégions. Voici un exemple des différentes formulations possibles :

```
| echo ' le texte ';
| echo ' le texte ' , ' le texte ' ;
| print ( ' le texte ' );
```

Chaîne 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 ' ou ". 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 préférable d'utiliser les chaînes délimitées par ' qui sont plus rapides. Nous reviendrons dans un autre tutorial sur les fonctions qui peuvent être utilisées sur les chaînes.

Caractères 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 '. En effet, cela donne une ligne comme :

Code PHP	Donne comme résultat à l'écran
<code>echo 'j'utilise php';</code>	Parse error: parse error, unexpected T_STRING, expecting ',' or ';' in votrefichier.php on line 2

et là forcément PHP croit qu'il faut s'arrêter au deuxième ', et ne comprend donc pas la suite, ce qui se traduit par un parse error. La solution, c'est l'antislash (\) qui permet de faire comprendre à PHP qu'il ne faut pas s'arrêter sur ce caractère-là. Ce qui donne :

Code PHP	Donne comme résultat à l'écran
<code>echo 'j\'utilise php';</code>	j'utilise php

Il faut utiliser la même chose avec les " dans des chaînes délimitées par des ". Mais du coup, pour mettre un \ dans une chaîne (ce qui est déjà plus rare), comment faire ? Eh bien c'est simple, il suffit de le faire précéder par un autre \, ce qui donne un \\.

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 ", 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
. Néanmoins ils peuvent servir à clarifier le code source, ou pour d'autres utilisations que nous verrons plus loin.

Afficher la date et l'heure

Avec PHP il est fort simple d'afficher la date du jour mais aussi de savoir quel jour nous serons dans 432 jours et réciproquement dans le passé. Voyons tout d'abord une date simple, nous allons en profiter pour utiliser notre première variable (les variables commencent toujours par le signe dollar \$).

Code PHP (ne copiez/collez pas ce code dans votre éditeur, retapez-le ou gare aux erreurs...)	Ce qui donne à l'écran
<pre><? \$date = date("d-m-Y"); \$heure = date("H:i");</pre>	

```
Print("Nous sommes le $date et il est
$heure");
?>
```

Nous sommes le 14-09-2000 et il est 15:10

C'est donc la fonction `date()` qui permet d'obtenir l'heure locale du serveur, mais attention l'heure locale est fonction de la situation géographique du serveur en lui-même. En effet un serveur situé au Canada vous donnera l'heure du Canada, en ce qui nous concerne les serveurs de Free.fr sont en France donc l'heure locale sera l'heure française :).

Dans le code ci-dessus nous générons la variable `$date` en lui donnant la valeur de ce que retourne la fonction `date("d-m-Y")` en l'occurrence : 14-09-2000. Les paramètres contenus entre les parenthèses `d-m-Y` peuvent être placés dans l'ordre que vous désirez, ainsi la date au format US sera écrite ainsi : `date("Y-m-d")`, il existe beaucoup de paramètres (extrait de la doc. en français de [Nexen.net](http://www.nexen.net)), je vous conseille de les tester pour vous rendre compte de ce que chaque paramètre retourne comme résultat :

- | **a** - "am" (matin) ou "pm" (après-midi)
- | **A** - "AM" (matin) ou "PM" (après-midi)
- | **d** - Jour du mois, sur deux chiffres (éventuellement avec un zéro) : "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** - ('L' minuscule) - 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

PHP dans du code HTML

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 `*.php3` ou encore `*.phtml`, bien que le plus utilisé soit `*.php3`. 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).

Comme je vous le disais en introduction, l'un des avantages du PHP c'est qu'il s'intègre facilement dans du code HTML classique. C'est d'ailleurs pour cela (en partie) qu'il connaît un fort succès sur les pages perso. En effet chacun peut à sa guise inclure quelques parties en PHP sans avoir à casser le site entièrement.

Le code PHP/HTML (ne copier/coller pas ce code dans votre éditeur, retapez-le ou gare aux erreurs...)	Donne comme résultat à l'écran
<pre><html> <body> Le texte en HTML <? // le code PHP -----</pre>	

```

$heure = date("H\hi");
print("<font size=\"2\" face=\"Arial\">
et celui en PHP.</font>");
?>
<!-- retour au code HTML -->
<br><font size="2" face="Arial">Il est
<? echo $heure; ?>.</font>
</body>
</html>

```

Le texte en HTML et celui en PHP.
Il est 18h16.

L'exemple ci-dessus démontre bien cette facilité à mélanger les deux langages. Notez que la seconde fois j'ai utilisé la fonction `echo` et non pas `print()`, c'est le plus souvent dans ce cas que je l'utilise pour une lecture plus simple du code dans les autres cas ce sera `print()`, mais bien sûr vous êtes libre sur ce coup-là :).

A noter : En PHP si vous souhaitez ajouter des commentaires il suffit de faire suivre deux Slashes `//` puis le commentaire de votre choix. Je mets l'accent sur les commentaires surtout lorsque que vous aurez des dizaines de lignes de code, 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 */`.

Quelques mots au passage sur la fonction `include()` : Si le code de votre page HTML est long, il est parfois fouillis de rajouter des lignes de codes PHP en plus, la méthode que j'utilise beaucoup pour bien séparer le code HTML du code PHP est la fonction `include()`.

Le code HTML/PHP <small>(ne copier/coller pas ce code dans votre éditeur, retapez-le ou gare aux erreurs...)</small>	Donne comme résultat à l'écran
<pre> <html> <body> Le texte en HTML <? include("toto.inc.php3"); // on appelle le fichier ?> </body> </html> </pre>	<p>Le texte en HTML et celui en PHP. Il est 18h16.</p>
<p>Le code PHP de toto.inc.php3</p> <pre> <? \$heure = date("H\hi"); print("<center> et celui en PHP. Il est \$heure.</center>"); ?> </pre>	

Et voilà le tour est joué, 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.php3`, ceci pour une meilleure lisibilité. Ainsi, en effet 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
<pre><? \$date = gmdate("H\hi"); print("Il est \$date"."gmt."); ?></pre>	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.

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
<pre><? \$nom = "Martin"; echo "Mon nom est \$nom"; ?></pre>	Mon nom est Martin
<pre><? \$nom = "Martin"; // affichage avec des simple quote echo 'Mon nom est \$nom'; ?></pre>	Mon nom est \$nom
<pre><? \$nom = "Martin"; // affichage avec des simple quote echo 'Mon nom est '.\$nom; ?></pre>	Mon nom est Martin

Vous l'aurez compris, PHP n'interprète pas ce qui se trouve entre simple quote, ainsi, ce n'est pas la valeur de

\$nom qui est affiché, mais \$nom. Il faut donc utiliser 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';
```

Ainsi le \ (backslash) indique à php qu'il ne faut pas considérer le ' du milieu comme celui qui délimite la fin de la chaîne de caractère, 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
<pre><? print('Nous somme le '.gmdate('d-m- Y').'...'); ?></pre>	Nous somme le 15-09-2000...

Nous avons réduit le code d'une ligne, ce qui n'est pas négligeable pour les gros développements, par contre j'admet que ceci est moins lisible pour quelqu'un qui débute totalement, c'est à vous de choisir.

Dans certains exercices futurs, nous verrons comment appeler une page en passant quelques variables, dans ce cas la concaténation nous servira je propose donc de regardez le tableau ci-dessous :

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

En d'autres termes, chaque fois que vous collez du texte et une variable (ou fonction), n'oubliez pas de mettre le point. Je ne dis pas que la première méthode ne fonctionne pas, mais elle n'est pas orthodoxe, et autant prendre les bonnes habitudes tout de suite :).

Récupérer les valeurs d'un formulaire

Quand l'un de vos visiteurs entre les informations dans un formulaire, celle-ci sont récupérées sous forme de variables.

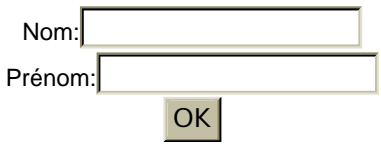
Le nom de ces variables dépend de la méthode d'envoi du formulaire.

Comme dans notre exemple suivant la méthode d'envoi est POST, il faut mettre comme nom `$_POST['nom_du_champ']`.

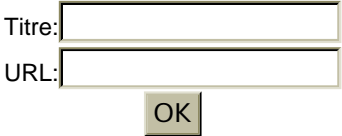
Pour les anciens qui exploitaient les variables de façon `$nom_du_champ` au lieu de `$_POST['nom_du_champ']`, je conseille de lire le tutoriel de flyingcow sur les variables globales à OFF.

Cette variable contient ce qu'a entré le visiteur dans le champ, oops :). Allez, un exemple me paraît plus simple, ci-dessous le `name="nom"` devient `$_POST['nom']` et `name="prenom"` devient `$_POST['prenom']`, il ne reste plus qu'à faire un `print()` des variables et le tour est joué !

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']` à `$prenom` et idem pour `$_POST['nom']` (attention une variable ne doit pas contenir d'espace et ne doit pas commencer par un chiffre alors n'en mettez pas dans vos nom de champ).

Le code HTML du formulaire (ne copier/coller pas ce code dans votre éditeur, retapez-le ou gare aux erreurs...)	Donne comme résultat à l'écran
<pre><html><body> <form method="post" action="verif.php3"> Nom : <input type="text" name="nom" size="12">
 Prénom : <input type="text" name="prenom" size="12"> <input type="submit" value="OK"> </form></body></html></pre>	
Le code PHP de verif.php3 (ne copier/coller pas ce code dans votre éditeur, retapez-le ou gare aux erreurs...)	Donne comme résultat à l'écran après envoi "OK"
<pre><? \$prenom = \$_POST['prenom']; \$nom = \$_POST['nom']; print ("<center>Bonjour \$prenom \$nom</center>"); ?></pre>	<p>Bonjour Thaal Rasha</p>

Il va bien sûr maintenant falloir contrôler les informations que rentre le visiteur pour éviter au maximum les erreurs. La première fonction que nous utiliserons est `empty()`, qui permet de contrôler si un champs est vide. Ensuite nous allons contrôler que `$_POST['url']` commence bien par `http://` à l'aide des deux fonctions `strtolower()` et `substr()`.

Le code HTML du formulaire (ne copier/coller pas ce code dans votre éditeur, retapez-le ou gare aux erreurs...)	Donne comme résultat à l'écran
<pre><html><body> <form method="post" action="verif.php3"> Titre : <input type="text" name="titre" size="12">
 URL : <input type="text" name="url" size="12" value="http://"></pre>	

<pre><input type="submit" value="OK"> </form></body></html></pre>	
Le code PHP de verif.php3 <small>(ne copier/coller pas ce code dans votre éditeur, retapez-le ou gare aux erreurs...)</small>	Donne comme résultat à l'écran après envoi "OK"
<pre><? \$titre = \$_POST['titre']; \$url = \$_POST['url']; if(empty(\$titre)) { print("<center>Le 'Titre' est vide !</center>"); exit(); } // vérification du début de l'url \$verif_url = strtolower(\$url); \$verif_url = substr ("\$verif_url", 0, 7); // on verifie les 7 premiers caractères if (\$verif_url!="http://") { print("L'URL doit commencée par http://"); exit(); } else { print("\$titre : \$url"); } ?></pre>	<p style="text-align: center;">Erreur n°1 : Le 'Titre' est vide !</p> <p style="text-align: center;">Erreur n°2 : L'URL doit commencée par http://</p> <p style="text-align: center;">Si pas d'erreur : phpdebutant : http://www.phpdebutant.org</p>

Avec cet exemple nous commençons à attaquer les conditions, c'est un aspect primordial dans tous les langages. La première vérification porte sur le champ '**titre**', la fonction `empty()` permet de contrôler si celui-ci est vide ou non. Ce qui nous donne :

- ▮ `if(empty($titre)){ 'erreur'; }` : Si la variable `$titre` est vide alors j'affiche le message : '**Le titre est vide**' (placé entre accolades) et j'arrête l'exécution du reste du code avec la commande `exit()`.
- ▮ Par contre si la variable n'est pas vide, l'exécution ne prend pas en compte ce qui se trouve entre accolades et continue.

La seconde vérification est plus fine puisqu'il s'agit de vérifier que les 7 premiers caractères qui ont été entrés par le visiteur sont bien **http://**. Pour commencer nous utilisons la fonction `strtolower()` qui permet de transformer tous les caractères en minuscules (ex. [HTTP://www.MONSite.CoM](http://www.MONSite.CoM) devient <http://www.monsite.com>). Puis à l'aide de la fonction `substr()`, nous sélectionnons les 7 premiers caractères (*0 est toujours le premier caractère d'une chaîne - le second chiffre '7' étant le nombre de caractères à sélectionner*), puis nous les comparons à ce que nous avons dans notre condition if :

- ▮ `if ($verif_url!="http://"){ 'erreur'; }` : Si les 7 premiers caractères sont différents (signe: **!=**) de **http://**, alors on exécute ce qui se trouve entre accolades (en l'occurrence on affiche un message d'erreur), puis nous arrêtons le reste du code avec la commande `exit()`.
- ▮ Par contre si le résultat est correct, PHP ignore ce qui se trouve entre accolades et exécute le reste du code.

Vous pourrez faire autant de tests que vous voudrez sur les champs, mais ne soyez pas trop draconien car les visiteurs n'aiment pas trop que l'on empiète sur leur liberté :). Les contrôles les plus fréquents s'effectuent sur les URL et email pour savoir si l'email comporte bien un "@" et un **point**.

Le code HTML du formulaire <small>(ne copier/coller pas ce code dans votre éditeur, retapez-le ou gare aux erreurs...)</small>	Donne comme résultat à l'écran
<pre><html><body></pre>	

<pre><form method="post" action="verif.php3"> Votre email : <input type="text" name="email" size="20"> <input type="submit" value="OK"> </form></body></html></pre>	<p>Votre email: <input type="text"/></p> <p><input type="button" value="OK"/></p>
<p>Le code PHP de verif.php3 (ne copier/coller pas ce code dans votre éditeur, retapez-le ou gare aux erreurs...)</p>	<p>Donne comme résultat à l'écran après envoi "OK"</p>
<pre><? \$email = \$_POST['email']; \$point = strpos(\$email, "."); \$aroba = strpos(\$email, "@"); if(\$point=='') { echo "Votre email doit comporter un point"; } elseif(\$aroba=='') { echo "Votre email doit comporter un '@"; } else { echo "Votre email est: '\$email'"; } ?></pre>	<p><i>Erreur n°1 :</i> Votre email doit comporter un point !</p> <p><i>Erreur n°2 :</i> Votre email doit comporter un '@' !</p> <p><i>Si pas d'erreur :</i> Votre email est : email@email.com</p>

Comme son nom l'indique, la fonction `strpos()` retourne la position d'un caractère dans une chaîne si celui-ci existe, autrement `strpos()` retourne "rien". C'est ce que nous utilisons pour savoir si les `point` et `@` sont bien présents dans l'email.

Exemple : Si `strpos()` retourne "10" cela veut dire que le premier caractère recherché est placé juste après les 10 premiers caractères donc en **11e position** dans la chaîne, puisque vous devez toujours vous rappeler que php commence à compter à **0** et non pas **1**.

Les structures de contrôles

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 (<i>boucle</i>)
for	Tant que (<i>boucle</i>)
==	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

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	Ce qui donne à l'écran
<pre> <? \$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'; } ?> </pre>	<p>512 est compris entre 500 est 999</p>

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	Ce qui donne à l'écran
<pre> <? \$medor = 'chien'; // on enchaîne les contrôles ci-dessous ---- if(\$medor == 'girafe') { echo 'Medor est un 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 ...'; } ?> </pre>	<p>Medor est un chien</p>

Cependant vous vous apercevez que cette structure est un peu lourde, et pas forcément facile à lire. Utilisez un switch permet de résoudre ce problème. Le code suivant est exactement le même que le précédent, mais écrit avec un switch

Le code PHP	Ce qui donne à l'écran
<pre> <? \$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 ...'; } ?> </pre>	<p>Medor est un chien</p>

Notez bien l'utilisation de break; . Cela permet de ne pas passer aux case suivant, sans quoi tout les messages s'afficheraient.

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	Ce qui donne à l'écran
<pre> <? \$toto = 6; \$i = 0; //-----[DEBUT BOUCLE]----- while(\$i != \$toto) { echo 'Toto est different de '.\$i.'
'; \$i++; // \$i++ est équivalent à (\$i+1) } //-----[FIN BOUCLE]----- echo 'Toto est égal à '.\$i; ?> </pre>	<p>Toto est different de 0 Toto est different de 1 Toto est different de 2 Toto est different de 3 Toto est different de 4 Toto est different de 5 Toto est égal à 6</p>

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 :

1. **\$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**.
2. **\$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**.
3. **\$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**.
4. etc...
5. **\$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).

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 meme code que précédemment avec une boucle for :

Le code PHP	Ce qui donne à l'écran
<pre> <? \$toto = 6; //-----[DEBUT BOUCLE]----- for(\$i=0; \$i != \$toto ; \$i++) { echo 'Toto est different de '.\$i.'
'; } //-----[FIN BOUCLE]----- echo 'Toto est égal à '.\$i; ?> </pre>	<p>Toto est different de 0 Toto est different de 1 Toto est different de 2 Toto est different de 3 Toto est different de 4 Toto est different de 5 Toto est égal à 6</p>

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.

Ecrire et lire dans un fichier texte

Nous allons voir ici comment l'on peut écrire et lire depuis un fichier se trouvant sur un serveur FTP, le vôtre en l'occurrence. Pour commencer vous créez un fichier de type *.txt (vous pouvez mettre l'extension que vous voulez, voire pas du tout) et nous placerons le fichier dans le même répertoire que le script PHP.

Contenu du fichier data.txt	Donne comme résultat à l'écran
1523	Le fichier contient : 1523
Le code PHP data.php3 (ne copier/coller pas ce code dans votre éditeur, retapez-le ou gare aux erreurs...)	
<pre><? \$fp = fopen("data.txt","r"); // (1) \$donnees = fgets(\$fp,255); // (2)</pre>	

```
fclose($fp); // (3)
// Affichage du résultat -----
echo "Le fichier contient : $donnees";
?>
```

Vous le voyez il est relativement simple de lire ce qui se trouve dans un fichier :

- 1 . On ouvre le fichier "data.txt" en lecture seule "r" avec la fonction `fopen()`.
- 2 . La lecture s'effectue avec la fonction `fgets()` et on spécifie le nombre de caractères (ici 255 soit la première ligne).
- 3 . Ensuite il ne reste plus qu'à refermer le fichier texte c'est la fonction `fclose()`.
- 4 . Enfin on affiche le résultat, c'est la variable `$donnees` qui contient "1523".

Revenons à la première ligne. La commande "r" indique que l'on ouvre le fichier uniquement en lecture seule. Nous allons voir ci-dessous que pour l'ouvrir en lecture/écriture, il suffit de mettre "r+". Concernant la seconde fonction `fgets()`, on spécifie le nombre de caractères que l'on veut lire dans le fichier (255). Dans notre cas nous aurions très bien pu mettre (`$fp,4`); puisque 1523 ne comporte que 4 caractères = logique. Ceci dit, le fait de mettre systématiquement 255 n'engendre pas de problème dans notre cas, sachez tout de même que 255 est le nombre maximum de caractères par ligne, le 256e passera à la ligne automatiquement à l'affichage.

Et devant vous yeux englués de bonheur :), voici le code php qui va vous permettre de réaliser un compteur de page. Notez qu'ici le fichier texte s'appelle `compteur.txt`.

Le code PHP de `compteur.php3`

(ne copier/coller pas ce code dans votre éditeur, retapez-le ou gare aux erreurs...)

```
<?
$fp = fopen("compteur.txt","r+"); // 1.On ouvre le fichier en lecture/écriture
$nbvisites = fgets($fp,11); // 2.On récupère le nombre dans le fichier
$nbvisites++; // 3.On incrémente le nombre de visites (+1)
fseek($fp,0); // 4.On se place en début de fichier
fputs($fp,$nbvisites); // 5.On écrit dans le fichier le nouveau nb
fclose($fp); // 6.On ferme le fichier
print("$nbvisites visiteurs"); // 7.On affiche le compteur à l'écran
?>
```

Il vous suffit de placer ce code dans la page un `index.php3` de votre site. Ici la fonction `fseek()` permet de se replacer où l'on veut en l'occurrence "0", donc au début, ensuite avec `fputs()` on écrit dans le fichier à l'endroit spécifié.

Pour finir avec les fichiers, n'oubliez pas que votre fichier texte qui se trouve sur votre FTP doit avoir tous les droits (chmod 777) pour que le script puisse écrire dedans.

A savoir : PHP permet également de créer et effacer des fichiers sur un serveur distant (FTP), je vous conseille de lire la documentation NEXEN ci-dessous pour en savoir encore plus.

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.

<p>Le code PHP de fonction.php3</p> <p>(ne copier/coller pas ce code dans votre éditeur, retapez-le ou gare aux erreurs...)</p> <pre> <? function Arial(\$size,\$color,\$texte) { print("" . \$texte . ""); } ?> </pre>
<p>Le code PHP de index.php3</p> <pre> <? Require("fonction.php3"); // on appelle la fonction // affichage ----- Arial("2","red","Ici le texte ..."); Arial("3","#0F74A3","Le second texte ..."); ?> </pre>
<p>On appelle index.php3 dans le navigateur, ce qui donne à l'écran</p>
<p>Ici le texte ...Le second texte ...</p>

Reprenons depuis le début. Vous pouvez donner le nom que vous voulez à vos fonctions, ensuite entre parenthèses ce sont les arguments que vous entrerez lors de l'utilisation de cette fonction. Ces arguments correspondent bien sûr à celles qui se trouvent entre accolades. 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 font de la même manière que les fonctions intégrées de PHP (print et echo en l'occurrence). Dans Index.php3, 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.php3` qui va contenir toutes vos fonctions de texte, tableau, etc. Il suffit juste ensuite de mettre un `require()` en tête de chacun de vos autres fichiers php pour pouvoir utiliser toutes les fonctions.

Je pense que vous voyez de suite l'intérêt de créer ses propres fonctions. Cela permet une bien meilleure lecture du code et un gain de temps à la programmation.

Dans un premier temps nous en resterons là, mais nous reviendrons à des fonctions plus complètes au cours

des exercices futurs.

Les 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 s'écrivent toujours en **MAJUSCULES** et bien sûr précédées du signe dollar \$, vous pouvez les utiliser n'importe où dans vos scripts comme ci-dessous :

Code PHP	Ce qui donne à l'écran
<pre><? print("Votre adresse IP est : \$REMOTE_ADDR"); ?></pre>	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 **\$REMOTE_ADDR**.

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

Variables	Description	Résultat à l'écran (Free.fr)
\$DOCUMENT_ROOT	Racine du serveur	/var/www/php.proxad.net
\$HTTP_ACCEPT_LANGUAGE	Langage accepté par le navigateur	fr
\$HTTP_HOST	Nom de domaine du serveur	proxyp3.free.fr
\$HTTP_USER_AGENT	Type de navigateur	Mozilla/4.0 (compatible; MSIE 5.0; Windows 98; DigExt)
\$PATH_INFO	Chemin web du script	/d2expert.free.fr/phpdebutant/fichier.php3
\$PATH_TRANSLATED	Chemin complet du script	/var/www/free.fr/3/d/2/e/x/d2expert/phpdebutant/fichier.php3
\$REQUEST_URI	Chemin du script	/d2expert.free.fr/phpdebutant/fichier.php3
\$REMOTE_ADDR	Adresse IP du client	195.132.7.201
\$REMOTE_PORT	Port de la requête HTTP	45039
\$QUERY_STRING	Liste des paramètres passés au script	?var=23&data=ok
\$SERVER_ADDR	Adresse IP du serveur	212.27.32.44
\$SERVER_ADMIN	Adresse de l'administrateur du serveur	email@email.com
\$SERVER_NAME	Nom local du serveur	php.proxad.net
\$SERVER_SIGNATURE	Type de serveur	?
\$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 allez pouvoir connaître la configuration et la version exacte de PHP qu'utilise le serveur où vous êtes hébergé. Faites un petit fichier comme ci-dessous ou [cliquez ici](#) pour tout savoir du serveur FREE.

Code PHP	Ce qui donne à l'écran
<pre><? phpinfo(); ?></pre>	Cela serait trop long :)

Quelques fonctions utiles

Fonction	Description	Code PHP	Rés.
addslashes()	Ajoute des anti-slashes devant les caractères spéciaux	<code>\$res = addslashes("L'a");</code>	L\'a
stripslashes()	Retire les anti-slashes devant les caractères spéciaux.	<code>\$res = stripslashes("L'a");</code>	L'a
dechex()	Retourne la valeur Hexadécimale d'un nombre (ici 2548).	<code>\$res = dechex("2548");</code>	9f4
ceil()	Retourne le nombre entier supérieur ici (12,1).	<code>\$res = ceil("12.1"); *</code>	13
chunk_split()	Permet de scinder une chaîne en plusieurs morceaux.	<code>\$res = chunk_split("DGDFEF", "2", "-");</code>	DG- DF- EF-
htmlentities()	Remplace les caractères par leur équivalent HTML (si ils existent).	<code>\$res = htmlentities("&");</code>	&
strstr()	Recherche le premier caractère 'p' dans la chaîne et affiche le reste de la chaîne y compris le 'p'.	<code>\$res = strstr("webmaster@phpdebutant.com", "p");</code>	phpdebutant.c
strlen()	Retourne la longueur de la chaîne	<code>\$res = strlen("lachainedecaracteres");</code>	20
strtolower()	Passe tous les caractères en minuscules.	<code>\$res = strtolower("LA CHAINE de caRactERes");</code>	la chaîne de caracteres
strtoupper()	Passe tous les caractères en MAJUSCULES.	<code>\$res = strtoupper("LA CHAINE de caRactERes");</code>	LA CHAINE DE CARACTERE
str_replace()	Remplace un caractère par un autre dans une chaîne. Tiens compte de la casse.	<code>\$res = str_replace("a", "o", "Lalala");</code>	Lololo
trim()	Efface les espaces blancs (\n, \r, etc) au début et à la fin d'une chaîne (pas au milieu).	<code>\$res = trim(" Salut le monde ");</code>	Salut le monde
ucfirst()	Met la première lettre de chaque chaîne en Majuscule.	<code>\$res = ucfirst("salut le monde. ca va ?");</code>	Salut le monde. ca va ?
ucwords()	Met la première lettre de chaque mot d'une chaîne en Majuscule.	<code>\$res = ucwords("salut le monde");</code>	Salut Le Monde
strpos()	Recherche la position du premier caractères trouvé. Retourne le nombre de caractères placés avant lui (ici 4).	<code>\$res = strpos("abcdef", "e");</code>	4
ereg()	Recherche si une chaîne de caractère est contenue dans une autre (ex. recherche si "ABCDE" contient "BCD").	<code>if(ereg("BCD", "ABCDE")) {echo "oui";} else {echo "non";}</code>	oui

* La virgule sous PHP est représentée par un point ".", ainsi 12,1 s'écrit : 12.1 !

SQL/MySQL (Create, Alter & Drop)

1. CREATE TABLE : Le langage SQL (Structured Query Language) 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](http://sql.free.fr/phpMyAdmin/), vous pouvez donc vous rendre à <http://sql.free.fr/phpMyAdmin/> pour accéder à votre base de données MySQL via l'interface [phpMyAdmin](http://sql.free.fr/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 :

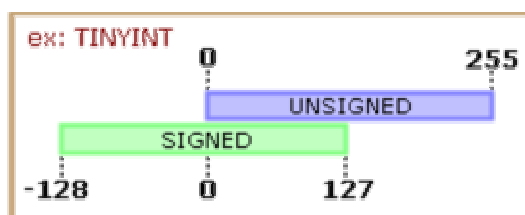
Syntaxe SQL pour créer la table : clients_tbl					
<pre>CREATE TABLE clients_tbl (id INT not null AUTO_INCREMENT, prenom VARCHAR (50) not null , nom VARCHAR (50) not null , ne_le DATE not null , ville VARCHAR (90) not null , enfants INT not null , PRIMARY KEY (id))</pre>					
Commentons la table en la visualisant sous phpMyAdmin pour MySQL					
Champ	Type	Null	Defaut	Extra	Commentaire
id	int(11)	Non	0	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 champs puisse recevoir aussi bien des Chaîne 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 fautive dans la base.
ville	varchar(90)	Non			Idem que PRENOM , ici nous avons spécifier 90 caractères car certaines villes comportent beaucoup de caractères (avec 90 nous sommes sûr)
enfants	int(11)	Non	0		INT (nombre entier) : Dans ce champs n'arriveront que des nombres entiers donc le INT voir SMALL INT et de rigueur.
Ci-dessous la table : clients_tbl une fois créée					
<pre>+++++ + id + prenom + nom + ne_le + ville + enfants + +++++</pre>					
Bien sûr pour le moment cette table ne comporte aucun enregistrement ...					

Dès cet instant vous devez consulter [la documentation officielle de MySQL](#) pour connaître tous les types de champs que vous pouvez utiliser dans une table, voici quelques exemples :

- **TINYINT** : Entier de 0 à 255 (unsigned)
- **SMALLINT** : Entier de 0 à 65535 (unsigned)

- **MEDIUMINT** : Entier de 0 à 16777215 (unsigned)
- **INT** : Entier de 0 à 4294967295 (unsigned)
- **BIGINT** : Entier de 0 à 18446744073709551615 (unsigned)
- **DECIMAL** : Un nombre à virgule flottante
- **DATE** : Une date, va de '1000-01-01' à '9999-12-31'
- **DATETIME** : Date et Heure, va de '1000-01-01 00:00:00' à '9999-12-31 23:59:59'
- **TIMESTAMP** : Date et Heure exprimée en secondes depuis le 1er janvier 1970. Va de '1970-01-01 00:00:00' à quelque part, durant l'année 2037
- **TIME** : Une mesure de l'heure, va de '-838:59:59' à '838:59:59'
- **YEAR** : Une année, va de 1901 à 2155
- **CHAR** : Chaîne de caractère de taille fixe, va de 1 à 255 caractères
- **VARCHAR** : Chaîne de caractère de taille variable, va de 1 à 255 caractères
- **TINYTEXT** ou **TINYBLOB** : Un objet BLOB ou TEXT, longueur maximale de 255
- **TEXT** ou **BLOB** : Un objet BLOB ou TEXT, longueur maximale de 65535
- **MEDIUMTEXT** ou **MEDIUMBLOB** : Un objet BLOB ou TEXT, longueur maximale de 16777215
- **LONGTEXT** ou **LOB** : 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).

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 modifié 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						
<pre> +++++ + id + prenom + nom + ne_le + ville + enfants + +++++</pre>						

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

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 aucunes informations 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 les plus 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.

SQL/MySQL (Insert et Select)

Pour commencer nous allons re-cr  er la table client_tbl de l'exercice n  11.

Syntaxe SQL pour cr��er la table : clients_tbl	
<pre>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))</pre>	
Ci-dessous la table : clients_tbl une fois cr��e	
<pre>+++++ + id + prenom + nom + ne_le + ville + enfants + +++++ Bien s��r pour le moment cette table ne comporte aucun enregistrement ...</pre>	

La commande INSERT INTO

Cette commande permet d'ins  rer 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 requ  te qui aura le m  me r  sultat:

```
| INSERT INTO clients_tbl VALUES
('','Patrick','Martin','1965/10/08','Bordeaux','2')
```

Ci-dessous la table : clients_tbl avec le nouvel enregistrement	
<pre>+++++ + id + prenom + nom + ne_le + ville + enfants + +++++ + 1 + Patrick + Martin + 1965/10/08 + Bordeaux + 2 + +++++</pre>	

Dans le premier exemple nous avons sp  cifier les noms des champs (entre parenth  ses) juste apr  s le nom de la table. Dans ce cas il n'est pas obligatoire de le faire si dans les valeurs vous sp  cifiez une valeur par champs, mysql affectera les valeurs dans l'ordre donn  es.

Par contre si vous ne donnez que deux valeurs, il sera important de toujours sp  cifier les champs dans lesquels elle doivent   tre ins  r  es, exemple :

```
| INSERT INTO clients_tbl(id,nom) Values('','Dupond')
| ou encore :
| INSERT INTO clients_tbl(nom,enfants) Values('Dupond','2')
```

Notez : La valeur du champs **id** est vide, je vous rappelle que ce champs est en **auto-increment** dans notre base, un nombre automatique sera donc attribu      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 spécifier 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	+
+	1	+	Patrick	+	Martin	+
+	2	+	Julien	+	Lebreton	+
+	3	+	Marc	+	Richard	+
+	4	+	Francis	+	Perrin	+
+	5	+	Daniel	+	Bacon	+

Ceci est une représentation en mode console et non pas sous phpMyAdmin

Admettons que nous voulions affiché 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='0' 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 0 et que la ville est Paris , c'est OK.
Et si le nombre d'enfant 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 dont la syllabe "ri" existe dans leur nom.

Il serait bien trop long de lister tous les exemples possibles, mais n'hésitez pas à consulter

la documentation en Français de Nexen.net sur MySQL.

SQL/MySQL (Delete et Update)

La commande **UPDATE**

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

```
I 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 champs **prenom** pour la valeur **Jacques**
WHERE id='1' Quand le champs **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 !

SQL/MySQL (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** et l'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					
ID	Livre	Prix	Titre	Code	Genre
1	40	Le glaive magique	BD		
3	40	Gaffes en gros	BD		
4	40	Lagaffe nous gêne	BD		
5	45	QRN sur Bretzelburg	BD		

	Code	Genre
	Code	Libelle

6	80	Tour de manège	RG	BD	Bande Dessinée
7	45	Le spectre aux balles d'or	BD	Po	Poésie
8	30	La bonne chanson	Po	RG	Roman de gare
9	50	La jeune Parque	Po	Ro	Roman
10	50	Michel Strogoff	Ro	SF	Science Fiction
11	50	La Serpe d'or	BD		
12	70	Toujours aimer	RG		
13	70	Toujours aimante	RG		
14	72	Toujours aimé	RG		

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 valeur

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 clareté.

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 modifier 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 passée 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

Fonctions PHP pour mySQL

PDF

----- Liste des exercices -----

Les fonctions PHP pour MySQL commence toujours par " **MYSQL_** ", en voici ci-dessous la liste exhaustive. En règle générale je n'utilise pas plus de 6 ou 7 fonctions, c'est d'ailleurs sur celles-ci que je mettrai l'accent plus bas dans cet exercice, ceci dit, je vous conseille tout de même les tester toutes.

Fonctions	Descriptions
<code>mysql_affected_rows()</code>	Retourne le nombre de rangée affectées par la dernière requête faite sur la base de données.
<code>mysql_close()</code>	Ferme la connexion à une base de données.
<code>mysql_connect()</code>	Établit une connexion vers la base de données spécifiée dans les arguments. Si cette base se trouve sur un port différent, faites suivre le nom de la machine par (:) puis le numéro du port (ex. : 8080). Cette connexion est automatiquement fermée à la fin de l'exécution du script sauf si une fonction mysql_close() intervient auparavant.
<code>mysql_create_db()</code>	Permet de créer une nouvelle base de données.
<code>mysql_data_seek()</code>	Déplace le pointeur interne de la rangée du résultat vers la rangée spécifiée. Utilisez cette fonction avec mysql_fetch_row() pour passer à la rangée spécifiée et récupérer les données.
<code>mysql_db_query()</code>	Permet d'exécuter une requête SQL sur une ou plusieurs tables d'une base de données.
<code>mysql_drop_db()</code>	Permet de supprimer une base de données. Dans ce cas toutes les données sont perdues.
<code>mysql_errno()</code>	Retourne le numéro de l'erreur générée lors de la dernière action sur la base de donnée.
<code>mysql_error()</code>	Retourne la description textuelle d'une erreur générée par la dernière action sur une base de données.
<code>mysql_fetch_array()</code>	Retourne un tableau qui représente tous les champs d'une rangée dans le résultat. Chaque appel récupère la prochaine rangée et ce jusqu'à ce qu'il n'y en ait plus. Chaque valeur de champ est stockée de deux façons: elle est indexée par un offset qui commence par '0', et indexée par le nom du champ.
<code>mysql_fetch_field()</code>	Récupère l'information attachée à un champs du résultat. Ces champs sont numérotés à partir de zéro.
<code>mysql_fetch_lengths()</code>	Retourne un tableau d'une longueur spécifiée pour chaque champ de résultat.
<code>mysql_fetch_object()</code>	Cette fonction est semblable à mysql_fetch_array() et à mysql_fetch_row() . Mais à la place, elle retourne un objet. Chaque champ du résultat correspond à une propriété dans l'objet renvoyé. Chaque appel à mysql_fetch_object() retourne la rangée suivante, ou False s'il ne reste plus de rangée.
	Retourne un tableau qui représente tous les champs d'une rangée du résultat. Chaque appel

<code>mysql_fetch_row()</code>	récupère la prochaine rangée et ce jusqu'à ce qu'il n'y en ait plus. C'est la méthode la plus rapide pour obtenir des résultats à partir d'une requête.
<code>mysql_field_flags()</code>	Permet d'obtenir une description des options rattachées au champs spécifié.
<code>mysql_field_len()</code>	Retourne la longueur maximale du champ spécifié.
<code>mysql_field_name()</code>	Retourne le nom d'une colonne. L'argument champ correspond à un offset numéroté à partir de zéro.
<code>mysql_field_seek()</code>	Déplace le pointeur interne du champ vers le champs spécifié. Le prochain appel vers <code>mysql_field_seek()</code> retournera l'information de ce champs.
<code>mysql_field_table()</code>	Retourne le nom de la table pour le champ spécifié.
<code>mysql_field_type()</code>	Retourne le type d'un champ particulier dans le résultat obtenu.
<code>mysql_free_result()</code>	Libère la mémoire associée au résultat spécifié. Elle n'est toutefois pas strictement nécessaire, car cette mémoire est automatiquement vidée lorsqu'un script termine son exécution.
<code>mysql_insert_id()</code>	Après l'insertion d'un nouvel enregistrement avec un champ <code>auto_increment</code> , la fonction <code>mysql_insert_id()</code> retourne l' ID qui vient d'être affecté au nouvel enregistrement.
<code>mysql_list_dbs()</code>	Interroge le serveur pour obtenir une liste de bases de données. Elle retourne un pointeur de résultat qui pourra être exploité avec <code>mysql_fetch_row()</code> et d'autres fonctions similaires.
<code>mysql_list_fields()</code>	Retourne un pointeur de résultat correspondant à une requête sur une liste de champs pour la table spécifiée. Ce pointeur pourra être exploité par toutes les fonctions qui recherchent des rangées à partir d'un résultat. Notez que l'argument lien reste optionnel.
<code>mysql_list_tables()</code>	Retourne le pointeur de résultat d'une liste de tables pour la base de données spécifiée. Ce pointeur pourra être exploité par toutes les fonctions qui recherchent des rangées à partir d'un résultat. Notez que l'argument lien reste optionnel.
<code>mysql_num_fields()</code>	Retourne le nombre de champs dans un résultat.
<code>mysql_num_rows()</code>	Retourne le nombre de rangées dans un résultat.
<code>mysql_pconnect()</code>	Cette fonction opère de la même manière que <code>mysql_connect()</code> , sauf que la connexion ne se referme pas à la fin de l'exécution du script sauf si un <code>mysql_close()</code> se trouve en fin de script.
<code>mysql_query()</code>	Permet d'exécuter une requête SQL sur une ou plusieurs tables d'une base de données. Si la requête exécute une instruction: INSERT , DELETE ou UPDATE , une valeur booléenne sera retournée (0 ou 1). Dans le cas d'une requête de type SELECT , vous obtiendrez un identifiant de résultat.
<code>mysql_result()</code>	Retourne la valeur du champ spécifié dans la rangée concernée. L'argument champ peut être un numéro et, dans ce cas, il sera considéré comme un champ offset. Il peut également désigner le nom de la colonne, avec éventuellement celui de la table. Enfin, il peut également renvoyer à un alias.
<code>mysql_select_db()</code>	Sélectionne la base de données par défaut.

Fonctions : `mysql_connect()`, `_select_db()`, `_query()`, `_numrows()`, `_close()`

Ces fonctions sont le minimum que vous utiliserez à chaque fois que vous interrogerez une base de données MySQL, voyez le code ci-dessous (nous avons repris notre table `clients_tbl` de l'exercice n°11).

Code PHP
<pre><? \$db = mysql_connect('sql.free.fr', 'login', 'password'); // 1 mysql_select_db('nom_de_la_base', \$db); // 2 \$req = mysql_query('SELECT * FROM clients_tbl'); // 3 \$res = mysql_numrows(\$req); // 4 echo 'Il y a '.\$res.' enregistrement(s) dans la table Clients.'; // 5 mysql_close(\$db); // 6 ?></pre>
Donne à l'écran
Il y a 5 enregistrement(s) dans la table Clients.

Explication du code ci-dessus :

1. On se connecte à la base de données :
Host : Dans notre cas le HOST est "sql.free.fr", vous pouvez également utiliser "[localhost](#)" par défaut.
Login : Ensuite vous devez mettre le "[login](#)" pour accéder à la base (chez les hébergeurs gratuits c'est souvent le même login que l'accès FTP).
Password : Et pour finir le "[mot de passe](#)", là aussi il s'agit très souvent du même password que l'accès FTP.
2. On sélectionne la base de données, en effet je vous rappelle que [MySQL](#) est un [serveur de bases de données](#), donc il peut contenir plusieurs bases. Bien sûr dans votre cas si vous êtes chez un hébergeur gratuit, vous n'avez en général droit qu'à une seule base, mais MySQL ne le sait pas ;), il faut donc lui spécifier sur quelle base vous souhaitez vous connecter.
Notez : Chez [Free.fr](#) et [Nexen](#) le nom de la base est souvent le même que le login de connection.
3. La fonction [Mysql_query\(\)](#) permet de passer une requête SQL vers la base de données, c'est évidemment l'un des attraits intéressants de PHP (notez que nous initialisons au passage la variable [\\$req](#) qui contient la requête).
4. La fonction [Mysql_numrows\(\)](#) permet de compter le nombre d'enregistrements que retourne la requête "[\\$req](#)" dans notre cas : 5 puisqu'il s'agit d'un simple "select " sur la table sans aucune condition, nous initialisons donc une variable [\\$res](#) qui contient : 5.
5. Il ne reste plus qu'à afficher le nombre de résultat avec un [echo\(\)](#) de [\\$res](#).
6. Et pour finir on referme la connexion - [ouverte avec mysql_connect](#) - avec la fonction [Mysql_close\(\)](#). Cette fonction n'est pas vraiment obligatoire avec un [Mysql_connect\(\)](#), car par défaut la connexion sera coupée automatiquement à la fin de l'exécution du script.

Interroger une table MySQL

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	Default	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

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

Code 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 créer la requete SQL et on l'envoie
$sql = 'SELECT nom,prenom,statut,date FROM famille_tbl';

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

// on fait une boucle qui va faire un tour pour chaque enregistrements
while($data = mysql_fetch_array($req))
{
    // on affiche les informations de l'enregistrements 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_array()` renvoie un tableau dont les clés sont les noms des champs sélectionnés. On a aussi rajouté après `mysql_query()` ceci: `or die('Erreur SQL !
'.$sql.'
'.mysql_error());`. Cela veut dire qu'en cas d'erreur dans la requête vers mysql, ce qui arrive fréquemment, php va afficher un message indiquant l'erreur renvoyé par mysql (grâce à `mysql_error()`) ce qui fournit une aide précieuse pour comprendre le problème.

Vous notez également que les dates de naissances 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 testes en ne changeant uniquement que la requête SQL (2). **Le reste du code ne change pas.**

Affichages des résultats par ordre alphabétique de prénom.

Le Code PHP de la requête

```
<?
// 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 classé 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

Affichages des résultats par comparaison de date.

Le Code PHP de la requête
<pre><? // 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 ?></pre>
Donne à l'écran
<p>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</p>

Affichages 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
<pre><? // Gardez le code ci-dessus, changez juste la requête ! \$sql = "SELECT nom,prenom,statut,date FROM famille_tbl WHERE prenom LIKE 'M%'"; // Le signe pourcentage "%" placé après le M, indique que la lettre M peut // être suivie d'autres caractères, mais pas précédée ! // Notez aussi que LIKE n'est pas sensible à la casse, cela veut dire que // la requête cherchera aussi bien des M majuscules que minuscules. ?></pre>
Donne à l'écran
<p>Dupond Marie (Mère), date de naissance : 1961-03-02 Dupond Manon (Fille), date de naissance : 1990-11-29</p>

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
<pre><? // 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. ?></pre>
Donne à l'écran
<p>Dupond Germaine (Grand-mère), date de naissance : 1939-02-15 Dupond Marie (Mère), date de naissance : 1961-03-02</p>

Dupond Manon (Fille), <i>date de naissance</i> : 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.

Le Code 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 créer la requete SQL et on l'envoie
$sql = 'SELECT nom,prenom,statut,date FROM famille_tbl';

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

// on fait une boucle qui va faire un tour pour chaque enregistrements
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'enregistrements 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();
?>
```

Donne à l'écran

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

Alimenter une ou plusieurs tables mySQL

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					
<pre><html> <form method="POST" action="add.php3"> <center> <input type="text" name="nom" size="20" value="nom" maxlength="35"> <input type="text" name="prenom" size="20" value="prenom" maxlength="35">
 <input type="text" name="email" size="20" value="email" maxlength="70"> <input type="text" name="icq" size="20" value="icq" maxlength="11">
 <input type="text" name="titre"</pre>	<table><tr><td>nom</td></tr><tr><td>prenom</td></tr><tr><td>email</td></tr><tr><td>icq</td></tr><tr><td>titre du site</td></tr></table>	nom	prenom	email	icq	titre du site
nom						
prenom						
email						
icq						
titre du site						

```

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>

```

url du site

Soumettre la requête

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.php3** auquel fait référence le **POST** du formulaire :

Comme vous le savez maintenant la variable de chacun des champs reprend le "**name**" précédé du signe dollar (\$), dans notre cas voici les 6 variables :
\$nom , **\$prenom** , **\$email** , **\$icq** , **\$titre** et **\$url**.

Code PHP de "add.php3"

```

<?
// On commence par vérifier 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 ecrit la requete sql
    $sql = "INSERT INTO infos_tbl VALUES
    ('','$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 champs 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 devriez spécifier dans la requête le nom de chaque champs, 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.

Bien sûr, l'idéal est 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 c

Code PHP de "add.php3" avec la vérification de doublon sur l'URL

```
<?
// On commence par vérifier 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
());

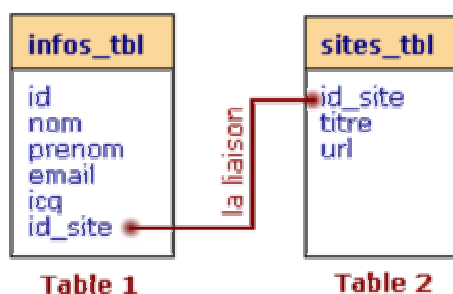
// on compte le nombre de résultat
$res = mysql_numrows($req);

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
{
$sql = "INSERT INTO infos_tbl VALUES
('','$nom','$prenom','$email','$icq','$titre','$url)";
mysql_query($sql) or die('Erreur SQL !'.$sql.'<br>'.mysql_error());

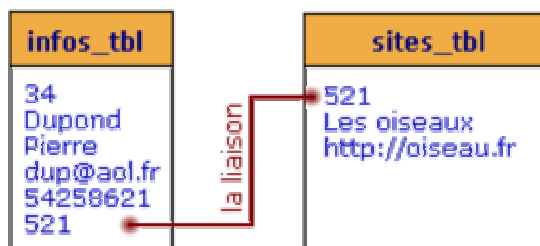
// on affiche le résultat pour le visiteur
echo 'Vos infos on été ajoutées.';
}
mysql_close(); // on ferme la connexion
?>
```

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 :



Un exemple concret :



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 lien 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

```

<?
$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());
$res = mysql_numrows($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
{
    $sql = "INSERT INTO sites_tbl VALUES('','$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
    $sql = "INSERT INTO infos_tbl VALUES
('','$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 travaillé un peu tout de même ;).

Les pseudos-frames

Suite à un grand nombre de questions postées sur le forum du genre "comment faire, comme sur le site, ? page=forum_index ", je vais tenter, dans cette exercice, de vous faire comprendre le système des pseudos frames.

Le premier exemple vous montre comment on peut procéder:

Lien hypertexte

```
<a href="index.php3?page=news">les news</a>
```

Vous devez avoir créé auparavant une page s'appelant news.php3 et une default.php3 pour l'exemple.

Le code PHP

```

La page index.php3
<?
// On regarde si la variable page est déclarée
if(isset($page))
{
    // Si oui on affiche la page correspondante
    include($page.'.php3');
}
else
{
    // Sinon on affiche la page par défaut
    include('default.php3');
}
?>

```

Notez que vous ne devez pas laisser le code de cet exemple sur votre site, il comporte un trou de sécurité (n'importe qui peut inclure n'importe quel fichier, qui ne contiendra pas forcément un code très catholique). Il n'est là que pour illustrer le principe, et les exemples suivant ne comportent pas le trou de sécurité.

Vous me direz, pourquoi une page par défaut ? Pour qu'il y ait une page visible lorsque la variable n'est pas déclarée ! Ca aurait pu être votre code html.

Maintenant, je vais vous montrer comment créer votre site avec des pseudos frames.

Voici le code de la page index.php3 de votre site

Code de "index.php3"

```

<html>
<head>
<title>index du site</title>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
</head><body bgcolor="#FFFFFF" text="#000000">
<table width="100%" border="0" cellspacing="0" cellpadding="0">
<tr>
<td colspan="2"><? include('haut.php3'); // L'entete de votre site ?>
</td>
</tr>
<tr>
<td width="19%"><? include('menu.php3'); // la page menu de votre site ?>
</td>
<td width="81%"><? include('centre.php3'); // la page centrale ?>
</td>
</tr>
</table>
</body>
</html>

```

Explication :

Nous avons donc créé la page index de votre site, comprenant un cadre de 2 lignes.

La page haut.php3 comprendra votre logo, nom du site et autres informations, elle ne devra pas, comme toutes les autres pages, comporter les balises html standards (<html> <title><head><body>), puisque ces informations sont dans la page index.

La page menu.php3 contiendra tous vos liens hypertextes.

La page centre.php3 pour le contenu du site.

Maintenant nous allons détailler la page centre.php3 , cette page est celle qui contiendra les chemins correspondants à la requête de la variable page.

Code PHP de "centre.php3"

```
<?
// On continu l'exemple avec notre page de news, et on ajoute une seconde page
if ($page=='news') {include('news.php3');}
elseif ($page=='formulaire') {include('formulaire.php3');} // Une page
supplémentaire nommée formulaire.php3
else {include('defaut.php3');} // Sinon on affiche la page par défaut
?>
```

Explication :

Pour détailler la page centre .php3 si la page s'appelle news , on inclut news.php3 etc....

Vous pouvez y ajouter les chemins des répertoires bien évidemment, comme ceci

```
if($page=="news"){ include("news/news.php3"); }
```

Là votre répertoire s'appelle news et la page news.php3

Maintenant passons à la page menu.php3 , la page qui contiendra vos liens

Code PHP de "menu.php3"

```
<?
print "<a href=\"index.php3?page=news\">les news</a>\n";
print "<a href=\"index.php3?page=formulaire\">formulaire</a>\n ";
?>
```

Je ne vous mets pas d'explications pour cette page, je pense que cela est inutile.! ;-)

Les sessions php4

Nous allons encore repousser l'exercice sur le moteur de recherche, car, je pense, le plus important est de se baser sur les questions que vous postez sur le forum. Il y'a eu plusieurs messages concernant le passage des variables de pages en pages, donc voici comment procéder. **Attention, les sessions ne fonctionnent que sous php4**

1 . Les noms à retenir

Pour travailler correctement avec les sessions, vous devez connaître les noms d'appels pour les différentes exécutions.

Le noms des fonctions
<pre>// le démarrage d'une session session_start(); // enregistrer une variable de session session_register(); // effacer une variable de session session_unregister(); // vérifier si une variable est déclaré pour la session en cours session_is_registered(); // le numéro de session en cours session_id(); // le nom de la session par défaut (souvent PHPSESSID) session_name(); // destruction de la session en cours session_destroy(); // destruction de toutes les variables pour la session en cours session_unset();</pre>

Vous devez mémoriser le nom des fonctions qui se rapportent aux sessions, elles sont peu nombreuses et surtout utiles.

Passons maintenant à l'application , et voyons comment travailler avec nos sessions.

Voici le code d'une page d'enregistrement classique de session
Le code
<pre><? session_start(); // on démarre la session (pas de html avant le session start !!!) \$id_de_session = session_id(); // on récupère l'id de session \$nom_session = session_name(); // on récupère le nom de la session</pre>


```
print 'L\'id de session est ' . $id_de_session . ' et le nom de session est ' . $nom_session;
?>

// donne à l'écran
L'id de session est dg25Hukf52zkn et le nom de session est PHPSESSID
```

Bon, c'était juste l'échauffement, passons maintenant à ce qui vous intéresse.

Nous allons créer un formulaire pour logger un visiteur, utile pour une section membre par exemple.

Code PHP du formulaire

```
<html>
<head><title>Formulaire</title>
</head>
<body bgcolor=#ffffff>
<form method="post" action="login.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>
```

Pas besoin d'explications, c'est le code html ;-)

Maintenant passons à la page login.php

Code PHP de "login.php"

```
<?
$login_defini = "phpdebutant"; // on définit un login pour la démo
$password_defini = "session"; // on définit un password pour la démo

// On vérifie les valeurs du formulaire
if ( $login == $login_defini && $password == $password_defini )
{
// Si les password et login sont valides

session_start(); // on démarre une session
// On enregistre les variables login et password dans la session en cours
session_register("login"); // Attention, pas de signe $ dans le
session_register
session_register("password");

header('location: afficher.php'); // Redirection sur une page pour afficher le
résultat
}
else{
// Si les password et login ne sont pas valides, on affiche un message
d'erreur
print 'Password ou login non valide';
```

```
}  
?>
```

Explications :

Nous avons commencé par vérifier la validité du login et du password que vous avez défini à l'avance. Vous devez respecter certaines règles avec les sessions, vous avez pu voir que le `session_start` n'est pas déclaré de suite, cela ne sert à rien d'ouvrir une session avant la vérification des infos ;-), mais vous ne devez dans aucuns cas, mettre du html pour personnaliser cette page, sinon PHP se mettra en erreur, vous voyez de quoi je veux parler, header ! Si vous voulez personnaliser le message d'erreur, mettez votre code html entre `else{ et }` uniquement.

Si les infos envoyés par le formulaire sont ok, on enregistre les variables dans la session en cours, juste après avoir déclaré cette même session avec le `session_start()`; dans le if .

On enregistre donc nos variables avec `session_register()`; notez bien l'absence du signe \$, ne me demandez pas pourquoi, c'est comme ça ;-)

En dernier, le header location, pour envoyer le visiteur sur la page afficher et voir le résultat.

Maintenant l'affiche du résultat

Code PHP de "afficher.php"

```
<?  
session_start();// on démarre la session  
// On affiche les variables enregistrées dans la sessions  
print 'Vôtre login est '.$login.' et votre password '.$password.'<br>';  
  
// Simple non ? , maintenant fessons quelques vérifications  
if ( !session_is_registered("password") )  
{  
print 'Vôtre mot de passe n\'est pas enregistré <br>';  
}  
// on affiche un lien pour terminer une session  
print '<a href="logout.php">Détruire la session</a><br>';  
?>
```

Code PHP de "logout.php"

```
<?  
session_start();// on démarre la session  
  
// Pour le fun on supprime une variable de session  
session_unregister("login");  
  
// maintenant on détruit la session en cours, je vous conseil d'utiliser unset  
et destroy, ce n'est pas obligatoire, mais plus sécurisé dirons nous.  
  
session_unset(); // on efface toutes les variables de session  
session_destroy(); // on detruit la session en cours.  
  
// On renvoi sur la page afficher pour voir le résultat, uniquement pour ce  
test, si tout s'est effectué normalement, le login et password ne  
s'afficheront pas, car ils n'existent plus  
header('location: afficher.php') ;  
?>
```

Affichage page par page

Présentation de l'exercice.

Comme la question revient très fréquemment sur le forum et sur le chat. Je vais vous apprendre à créer simplement un affichage page par page à partir du résultat d'une requête SQL.

Cet exo est en quelque sorte une extension de l'exo n°15

<mode type="hors sujet" voix="voix d'hôtesse">

Ding ding dong ding !

Le capitaine vous souhaite un bon voyage sur notre ligne.

Le tuto est bien long... Alors respirez un bon coup, concentrez-vous! On y va!

</mode>

La table SQL.

Pour notre exercice, nous allons utiliser une table assez simple.

Requête de création et d'insertion des données

```
CREATE TABLE vaches (  
  id int(11) NOT NULL auto_increment,  
  surnom varchar(50) NOT NULL default '',  
  prenom varchar(20) NOT NULL default '',  
  PRIMARY KEY (id)  
) TYPE=MyISAM;  
  
INSERT INTO vaches VALUES (1, 'Qui aime les frites', 'Margueritte');  
INSERT INTO vaches VALUES (2, 'Belle clochette', 'Roussette');
```

```

INSERT INTO vaches VALUES (3, 'La starlette', 'Blanchette');
INSERT INTO vaches VALUES (4, 'Petite Pataude', 'Noireau');
INSERT INTO vaches VALUES (5, 'Toujours chouette', 'Paquerette');
INSERT INTO vaches VALUES (6, 'Tête de linotte', 'Cowpilot');
INSERT INTO vaches VALUES (7, 'Qui saute partout', 'Kancowroo');
INSERT INTO vaches VALUES (8, 'La gentille', 'Cowchenille');
INSERT INTO vaches VALUES (9, 'Le plus bizarre des cows', 'Haricow');

```

Première étape : afficher tous les enregistrements dans une table HTML

Comme c'est une question qui revient aussi très souvent, je vais vous expliquer comment faire pour afficher des résultats dans une table HTML.

Normalement, avant d'avoir posé les yeux sur ce tuto, vous êtes censés avoir lu, compris et assimilé les tutos précédents. Si c'est le cas, le script ci-dessous ne devrait pas vous poser trop de problèmes.

Affichage des enregistrements dans une table HTML.

```

<html>
<body>
<?
// information pour la connection à la DB
$host = 'localhost';
$user = 'root';
$pass = '';
$db = 'test';

// connection à la DB
$link = mysql_connect ($host,$user,$pass) or die ('Erreur :
'.mysql_error() );
mysql_select_db($db) or die ('Erreur :'.mysql_error());

// requête SQL qui compte le nombre total d'enregistrement dans la
table et qui
//récupère tous les enregistrements
$select = 'SELECT prenom,surnom FROM vaches';
$result = mysql_query($select,$link) or die ('Erreur : '.mysql_error
() );
$total = mysql_num_rows($result);

// si on a récupéré un résultat on l'affiche.
if($total) {
    // debut du tableau
    echo '<table bgcolor="#FFFFFF">'. "\n";
    // première ligne on affiche les titres prénom et surnom dans
2 colonnes
    echo '<tr>';
    echo '<td bgcolor="#669999"><b><u>Prénom</u></b></td>';
    echo '<td bgcolor="#669999"><b><u>Surnom</u></b></td>';
    echo '</tr>'. "\n";
    // lecture et affichage des résultats sur 2 colonnes, 1 résultat
par ligne.
    while($row = mysql_fetch_array($result)) {
        echo '<tr>';
        echo '<td bgcolor="#CCCCCC">'. $row['prenom']. '</td>';
        echo '<td bgcolor="#CCCCCC">'. $row['surnom']. '</td>';
        echo '</tr>'. "\n";
    }
    echo '</table>'. "\n";
    // fin du tableau.
}
else echo 'Pas d\'enregistrements dans cette table...';

// on libère le résultat

```

```
mysql_free_result($result);

?>
</body>
</html>
```

Le résultat de ce code donnera ceci à l'écran :

<u>Prénom</u>	<u>Surnom</u>
Margueritte	Qui aime les frites
Roussette	Belle clochette
Blanchette	La starlette
Noireau	Petite Pataude
Paquerette	Toujours chouette
Cowpilot	Tête de linotte
Kancowroo	Qui saute partout
Cowchenille	La gentille
Haricow	Le plus bizarre des cows

Explications

Juste avant notre boucle, nous affichons une ligne de la table HTML.

Elle est constituée de 2 colonnes dans lesquelles nous mettons le titre de chacune d'elles sous cette forme :

Affichage des titres dans des colonnes d'une table HTML

```
<tr>
<td bgcolor="#669999"><b><u>Prénom</u></b></td>
<td bgcolor="#669999"><b><u>Surnom</u></b></td>
</tr>;
```

Dans la boucle, nous allons afficher chaque enregistrement dans une ligne du tableau.

Le prénom dans une cellule et le surnom dans une autre, sous cette forme :

Affichage des données.

```
<tr>
<td bgcolor="#CCCCCC"><b><u>Roussette</u></b></td>
<td bgcolor="#CCCCCC"><b><u>Belle clochette</u></b></td>
</tr>;
```

Explication d'une option d'un select SQL : Limit

Bien pratique, cette option nous permet de sélectionner une partie des enregistrements de la table en lui donnant le nombre d'enregistrements à récupérer à partir d'une position donnée.

Voici quelques exemples :

Syntaxe d'un SELECT avec une limite - Exemple 1

```
SELECT prenom,surnom FROM vache LIMIT 0,4;
```

Cette requête retournera 4 enregistrements à partir du premier de la liste (0 + 1).

Dans le cas de notre table :

<u>Prénom</u>	<u>Surnom</u>
Margueritte	Qui aime les frites
Roussette	Belle clochette
Blanchette	La starlette
Noireau	Petite Pataude

Syntaxe d'un SELECT avec une limite - Exemple 2

```
SELECT prenom,surnom FROM vache ORDER BY prenom ASC LIMIT 0,4;
```

Cette requête retournera 4 enregistrements à partir du premier de la liste dans l'ordre alphabétique. Donc, la requête trie les enregistrements et ensuite en prend 4 à partir du premier (0 + 1).

Dans le cas de notre table :

<u>Prénom</u>	<u>Surnom</u>
Blanchette	La starlette
Cowchenille	La gentille
Cowpilot	Tête de linotte
Haricow	Le plus bizarre des cows

Syntaxe d'un SELECT avec une limite - Exemple 2

```
SELECT prenom,surnom FROM vache ORDER BY prenom ASC LIMIT 5,4;
```

Cette requête retournera 4 enregistrements à partir du 6ème (on en saute 5 , 5 +1) dans l'ordre alphabétique. Voilà ce que donne le résultat :

<u>Prénom</u>	<u>Surnom</u>
Margueritte	Qui aime les frites
Noireau	Petite Pataude
Paquerette	Toujours chouette
Roussette	Belle clochette

Si vous voulez comparer avec ce que donnerait un select sans limit avec un tri alphabétique :

Comparaison de l'exemple 2 sans limit

```
SELECT prenom,surnom FROM vache ORDER BY prenom ASC;
```

Voilà le résultat :

<u>Prénom</u>	<u>Surnom</u>
Blanchette	La starlette
Cowchenille	La gentille
Cowpilot	Tête de linotte
Haricow	Le plus bizarre des cows
Kancowroo	Qui saute partout
Margueritte	Qui aime les frites
Noireau	Petite Pataude
Paquerette	Toujours chouette
Roussette	Belle clochette

Notre affichage page par page.

Je vais reprendre chaque partie du script de base pour le modifier afin de créer un affichage page par page.

Ouvrez bien vos yeux, c'est parti !

Première partie du script.

Première partie du script.

```
// information pour la connection à le DB
$host = 'localhost';
$user = 'root';
$pass = '';
$db = 'test';

// connection à la DB
$link = mysql_connect ($host,$user,$pass) or die ('Erreur :
```

```
'.mysql_error() );
mysql_select_db($db) or die ('Erreur :'.mysql_error());
```

Explications

Cette partie-là ne change pas et ne comporte que des choses que vous êtes censés connaître.

- **\$host**, **\$pass**, **\$user** et **\$db** contiennent les informations pour se connecter à la base de données.
- Ensuite on se connecte à la base de données.

De quelles variables avons nous besoin ?

Pour notre affichage page par page, nous avons besoin de connaître certaines données que l'on va stocker dans des variables.

- **\$nombre** : Nombre d'enregistrements que l'on veut afficher par page.
- **\$total** : Le nombre total d'enregistrements de la table .
- **\$limite** : O? en sommes-nous dans notre affichage.

Nous allons reprendre notre script de base et y ajouter ce dont nous avons besoin pour travailler.

Au début du script nous allons initialiser les variables **\$nombre** et **\$limite**.

Initialisation des variables.

```
$nombre = 5; // on va afficher 5 résultats par page.
if (!$limite) $limite = 0; // si on arrive sur la page pour la
première fois
// on met limite à 0.

$path_parts = pathinfo($PHP_SELF);
$page = $path_parts["basename"];
```

Explications :

- **\$nombre** est initialisé à 5 (5 enregistrements affichés à la fois)
- Si **\$limite** n'existe pas on l'initialise à 0.
- Les 2 lignes de codes suivantes permettent de récupérer le nom de la page qui sera utilisée dans les liens.
- La variable **\$total** contient le nombre d'enregistrement à afficher au total.

Compte le nombre d'enregistrements total de la table.

```
$select = 'SELECT count(id) FROM vaches';

$result = mysql_query($select,$link) or die ('Erreur : '.mysql_error
() );

$row = mysql_fetch_row($result);

$total = $row[0];
```

Ajout de la requête select avec le limit

Maintenant, nous avons besoin de faire notre requête de sélection qui va afficher **\$nombre** enregistrements à partir de **\$limite**.

Selectionne la portion d'enregistrements à afficher.

```
// requête SQL qui ne prend que le nombre d'enregistrement necessaire  
à l'affichage.
```

```
$select = 'select prenom,surnom FROM vaches ORDER BY prenom ASC limit  
'. $limite. ',' . $nombre;
```

```
$result = mysql_query($select,$link) or die ('Erreur : '.mysql_error  
( ) );
```

Explications

On utilise l'opérateur de concaténation pour compléter notre requête de base par le tri et la limite de sélection. Notre requête donnera donc au premier affichage de la page (**\$limite** étant initialisé à 0 la première fois):

Requête de sélection de la première page

```
SELECT prenom,surnom FROM vaches ORDER BY prenom ASC limit 0,5
```

Qui donnera ceci à l'écran :

<u>Prénom</u>	<u>Surnom</u>
---------------	---------------

Blanchette	La starlette
------------	--------------

Cowchenille	La gentille
-------------	-------------

Cowpilot	Tête de linotte
----------	-----------------

Haricow	Le plus bizarre des cows
---------	--------------------------

Kancowroo	Qui saute partout
-----------	-------------------

Calcul de la nouvelle limite d'affichage

Imaginons que nous avons affiché la page pour la première fois. Notre **\$limite** est à Zéro, et notre script affiche donc 5 enregistrements à partir du début.

C'est pas mal, mais maintenant il faut savoir s'il reste d'autres enregistrements à afficher. Si oui il nous faut mettre un lien ou un bouton permettant de les afficher.

Et pour cela, il faut recalculer la limite d'affichage.

Nous voulons afficher les **\$nombre** enregistrements suivants et/ou les **\$nombre** enregistrements précédents:

Il s'agit d'un calcul très simple :

Calcul des limites précédente et suivante

```
$limitesuivante = $limite + $nombre;  
$limiteprecedente = $limite - $nombre;
```

Voilà à quoi notre requête SQL ressemblera si on est sur la première page et que l'on clique sur le lien suivant.

Requête de sélection de la deuxième page

```
SELECT prenom,surnom FROM vaches ORDER BY prenom ASC limit 5,5
```

Qui donnera ceci à l'écran :

<u>Prénom</u>	<u>Surnom</u>
---------------	---------------

Margueritte	Qui aime les frites
-------------	---------------------

Noireaude	Petite Pataude
-----------	----------------

Paquerette	Toujours chouette
------------	-------------------

Roussette	Belle clochette
-----------	-----------------

Pour avoir la limite d'affichage de la portion suivante, on ajoute **\$nombre** à **\$limite**

Pour avoir la limite d'affichage de la portion précédente, on retire **\$nombre** à **\$limite**

Affichage des boutons ou liens page précédente , page suivante

Maintenant, il va nous falloir afficher les liens ou les boutons pour afficher le reste des enregistrements.

Pour cela nous allons devoir réfléchir à 2 choses.

- Sommes nous sur la première page de notre affichage ? C'est à dire est ce que **\$limite** vaut 0 .
 - oui : il ne faut donc pas afficher de bouton ou de lien 'page précédente'.
 - non : il faut afficher un bouton ou un lien 'page précédente'.

Affichage du lien précédent si besoin

```
if($limite != 0) {
    echo '<a href="'. $page. '?limite=' . $limiteprecedente. '>Page
précédente</a>';
}
```

Ou bien affichage d'un bouton formulaire

```
if($limite != 0) {
    echo '<form action="'. $page. '" method="post">';
    echo '<input type="submit" value="précédents">';
    echo '<input type="hidden" value="'. $limiteprecedente. '"
name="limite">';
    echo '</form>';
}
```

- Y a-t-il encore une page à afficher après la page courante? C'est à dire : est-ce que notre limite recalculée (**\$limitesuivante**) est plus petite que notre **\$total** ?
 - oui : il faut donc afficher un bouton ou un lien 'page suivante'.
 - non : il ne faut pas afficher de bouton ou de lien 'page suivante' car on est arrivé à la fin de notre affichage.

Affichage du lien suivant si besoin

```
if($limitesuivante < $total) {
    echo '<a href="'. $page. '?limite=' . $limitesuivante. '>Page
Suivante</a>';
}
```

Ou bien affichage d'un bouton formulaire

```
//Ou bien affichage d'un bouton formulaire
if($limitesuivante < $total) {
    echo '<form action="'. $page. '" method="post">';
    echo '<input type="submit" value="suivants">';
    echo '<input type="hidden" value="'. $limitesuivante. '"
name="limite">';
    echo '</form>';
}
```

Et voilà le tour est joué !

Maintenant, Si vous voulez afficher + de 5 enregistrements par page , il vous suffit de changer la valeur de **\$nombre**.

<mode type="hors sujet" voix="voix d'hôtesse">

Ding ding dong ding !

Nos hôtesse vont passer vous proposer des pizzas et du coca ou des jus de fruits.

C'est le moment de faire une petite pause avant d'aller plus loin.

</mode>

Comment utiliser ce que nous venons d'apprendre de façon intelligente.

Il serait particulièrement malin de créer des fonctions pour notre affichage page/page pour pouvoir les réutiliser dans tous nos scripts.

```
<mode type="hors sujet" voix="voix d'hôtesse">
```

Ding ding dong ding !

Le capitaine vous prie de bien vouloir attacher vos ceintures, nous passons à la vitesse supérieure !.

```
</mode>
```

Valider la \$limite passée par l'url.

Comme pour tous les scripts, il est important de penser à la sécurité, particulièrement quand des variables sont passées par URL ou par un formulaire.

Nous devons tester si la valeur de \$limite :

- I est bien numérique.
- I se trouve dans une fourchette de valeurs possibles (de 0 à \$max)
- I est un multiple de \$nombre.

Pourquoi faire tout ces tests ? Parce qu'il ne faut jamais faire confiance à l'utilisateur, ni aux données qu'il peut entrer dans un formulaire ou via l'URL !

Fonction qui vérifie la validité de \$limite

```
function verifLimite($limite,$total,$nombre) {

    // je verifie si limite est un nombre.

    if(is_numeric($limite)) {

        // si $limite est entre 0 et $total, $limite est ok

        // sinon $limite n'est pas valide.

        if(($limite >=0) && ($limite <= $total) && (($limite%$nombre)
==0)) {

            // j'assigne 1 à $valide si $limite est entre 0 et $max

            $valide = 1;

        }

        else {

            // sinon j'assigne 0 à $valide

            $valide = 0;

        }

    }

    else {

        // si $limite n'est pas numérique j'assigne 0 à $valide

        $valide = 0;

    }

    // je renvoie $valide

    return $valide;
```

```
}
```

Explication des arguments passés en paramètres à la fonction `verifLimite`

- **\$limite** : la limite passée en paramètre.
- **\$total** : le nombre total d'enregistrements à afficher.
- **\$nombre** : nombre total d'enregistrements à afficher par page.

Fonction qui affiche les boutons précédent/suivant

```
<?
function displayNextPreviousButtons($limite,$total,$nb,$page) {
$limiteSuivante = $limite + $nb;
$limitePrecedente = $limite - $nb;
echo '<table><tr>'. "\n";
if($limite != 0) {
    echo '<td valign="top">'. "\n";
    echo '<form action="'. $page. '" method="post">'. "\n";
    echo '<input type="submit" value="précédents">'. "\n";
    echo '<input type="hidden" value="'. $limitePrecedente. '"
name="limite">'. "\n";
    echo '</form>'. "\n";
    echo '</td>'. "\n";
}
if($limiteSuivante < $total) {
    echo '<td valign="top">'. "\n";
    echo '<form action="'. $page. '" method="post">'. "\n";
    echo '<input type="submit" value="suivants">'. "\n";
    echo '<input type="hidden" value="'. $limiteSuivante. '"
name="limite">'. "\n";
    echo '</form>'. "\n";
    echo '</td>'. "\n";
}
echo '</tr></table>'. "\n";
}
```

Explication des arguments passés en paramètres à la fonction `displayNextPreviousButtons`.

- **\$limite** : la limite d'affichage courante.
- **\$total** : nombre total d'enregistrements retournés par le query
- **\$nb** : nombre d'enregistrements à afficher par page.
- **\$page** : nom de la page php.

Explication de la fonction `displayNextPreviousButtons`.

Elle se comporte exactement comme les bouts de script expliqués ci-dessus.

On calcule la limite précédente et suivante, et, selon la page sur laquelle on se trouve ainsi que le nombre d'enregistrements total à afficher, on affiche ou pas les boutons next/previous.

Comment afficher les numéros de page.

Vous vous demandez sûrement comment faire pour afficher des liens vers les différentes pages à afficher. Cette petite fonction va vous aider.

Fonction qui affiche les liens vers les pages

```
function affichePages($nb,$page,$total) {
$nbpages=ceil($total/$nb);
$numeroPages = 1;
$compteurPages = 1;
$limite = 0;
echo '<table border = "0" ><tr>'. "\n";
while($numeroPages <= $nbpages) {
echo '<td ><a href = "'. $page. '?'
```

```

    limite='.$limite.'">'.$numeroPages.'</a></td>'. "\n";
    $limite = $limite + $nb;
    $numeroPages = $numeroPages + 1;
    $compteurPages = $compteurPages + 1;
    if($compteurPages == 10) {
        $compteurPages = 1;
        echo '<br>'. "\n";
    }
}
echo '</tr></table>'. "\n";
}

```

Explication des arguments passés en paramètres à la fonction affichePages

- **\$nb** : \$nb : nombre d'enregistrements à afficher par page.
- **\$page** : nom de la page php.
- **\$total** : nombre total d'enregistrements retournés par le query

Explication de la fonction affichePages.

1. On calcule le nombre de pages en divisant le nombre total d'enregistrements (retournés par le query) par le nombre d'enregistrements à afficher par page.
Si nous avons 20 enregistrements à afficher il nous faudra $20/5 = 4$ pages. Si nous avons 21 enregistrements à afficher, comme $21/5 = 4,2$, cinq pages seront nécessaires. On doit donc arrondir à l'entier supérieur grâce à la fonction ceil();
2. On initialise **\$numeroPages** à 1 (première page)
3. On fait une boucle qui va faire varier **\$numeroPages** tant qu'il est inférieur ou égal à **\$nbpages**.
4. A chaque passage dans la boucle, on incrémente **\$numeroPages**.
5. Pour ne pas se retrouver avec trop de numéros de pages sur la même ligne, on met en place un compteur **\$compteurPages** qui va varier dans la boucle lui aussi.
6. Une fois que sa valeur arrivera à 10, on ajoute un retour à la ligne et on remet notre compteur à 1.

Modification de notre script de base afin qu'il utilise nos 2 nouvelles fonctions.

Si le nombre total d'enregistrements à afficher est plus grand que le nombre d'enregistrements à afficher par page, on appelle la fonction qui affiche les numéros de page ainsi que celle qui affiche les boutons

Modification du script de base pour utiliser nos 2 fonctions

```

<html>

<body>

<?

//=====

// includes du fichier fonctions

//=====

require 'fonctions.php';

//=====

// information pour la connection à la DB

//=====

$host = 'localhost';

$user = 'root';

$pass = '';

```

```
$db = 'test';

//=====

// initialisation des variables

//=====

// on va afficher 5 résultats par page.

$nombre = 5;

// si limite n'existe pas on l'initialise à Zéro
if (!$limite) $limite = 0;

// on cherche le nom de la page.

$path_parts = pathinfo($PHP_SELF);
$page = $path_parts["basename"];

//=====

// connection à la DB

//=====

$link = mysql_connect ($host,$user,$pass) or die ('Erreur :
'.mysql_error() );

mysql_select_db($db) or die ('Erreur :'.mysql_error());

//=====

// requête SQL qui compte le nombre total
// d'enregistrement dans la table.

//=====

$select = 'SELECT count(id) FROM vaches';

$result = mysql_query($select,$link) or die ('Erreur : '.mysql_error
() );

$row = mysql_fetch_row($result);

$total = $row[0];

//=====

// vérifier la validité de notre variable
```

```
// $limite;

//=====

$verifLimite= verifLimite($limite,$total,$nombre);

// si la limite passée n'est pas valide on la remet à Zéro

if(!$verifLimite) {

    $limite = 0;

}

//=====

// requête SQL qui ne prend que le nombre

// d'enregistrement necessaire à l'affichage.

//=====

$select = 'select prenom,surnom FROM vaches ORDER BY prenom ASC limit
' . $limite . ',' . $nombre;

$result = mysql_query($select,$link) or die ('Erreur : ' . mysql_error
() );

//=====

// si on a récupéré un resultat on l'affiche.

//=====

if($total) {

    // debut du tableau

    echo '<table bgcolor="#FFFFFF">' . "\n";

    // première ligne on affiche les titres prénom et surnom dans
    2 colonnes

    echo '<tr>';

    echo '<td bgcolor="#669999"><b><u>Prénom</u></b></td>';

    echo '<td bgcolor="#669999"><b><u>Surnom</u></b></td>';

    echo '</tr>' . "\n";

    // lecture et affichage des résultats sur 2 colonnes

    while($row = mysql_fetch_array($result)) {

        echo '<tr>';

        echo '<td bgcolor="#CCCCCC">' . $row['prenom'] . '</td>';

        echo '<td bgcolor="#CCCCCC">' . $row['surnom'] . '</td>';
```

```

        echo '</tr>'."\n";
    }

    echo '</table>'."\n";
}

else echo 'Pas d\'enregistrements dans cette table...';
mysql_free_result($result);

//=====

// si le nombre d'enregistrement à afficher
// est plus grand que $nombre
//=====

if($total > $nombre) {
    // affichage des liens vers les pages
    affichePages($nombre,$page,$total);

    // affichage des boutons
    displayNextPreviousButtons($limite,$total,$nombre,$page);
}

?>

</body>

</html>

```

Que pourriez-vous ajouter?

- Signaler sur quelle page l'utilisateur se trouve.
"vous êtes sur la page n°1", "vous êtes sur la page n°2",...
- Changer la couleur du numéro de page active.
- Afficher où on en est dans l'affichage des enregistrements :
Affichage des 5 premiers enregistrements sur / 32.
Affichage des 10 premiers enregistrements sur / 32.
etc...

<mode type="hors sujet" voix="voix d'hôtesse">

Ding ding dong ding !

J'espère que vous avez bon voyage en notre compagnie. L'équipe de PHPdébutant espère vous revoir très bientôt sur ses lignes :p.

Le capitaine du tuto :flyingcow.

</mode>

Ca marche pas ?

Ca marche pas?

"Ca marche pas !!!" vous connaissez? Tout ces petits bugs qui nous rendent fous, peuvent être facilement repérés si on utilise une méthode organisée pour les détecter.

Vu les questions qui reviennent en permanence sur le chat et le forum ne me dites pas non :) et comme visiblement un seul tuto sur le debuguage ne suffit pas, voici un tuto complémentaire "ca marche pas".

Les messages d'erreur.

Première chose importante, il faut comprendre le message d'erreur correctement et ne pas tenter une interprétation personnelle de l'erreur. Pour vous aider, voici une liste des erreurs les plus fréquemment rencontrées. Bien sûr, cette liste ne contient pas TOUS les messages d'erreur :).

Message	Signification
Parse error: parse error in xxxx.php on line y	Typique d'une erreur de syntaxe. Vérifiez si vous n'avez pas oublié un ; (point virgule) en fin de ligne. Ou encore un \$ (dollar) devant le nom d'une variable. N'hésitez pas à contrôler les lignes avoisinantes. L'erreur se trouve souvent juste au-dessus.
Warning: Oops, php_SetCookie called after header has been	Vous avez tenté d'initialiser un cookie après que l'entête HTTP soit envoyé au client. Vérifiez si une sortie (echo,

sent in xxxx.php on line y	print, message d'erreur, ligne blanche avant les tags php) ne se fait quelques temps avant.
Warning: MySQL Connection Failed: Access denied for user:	Erreur de connexion à la base MySQL. Vérifiez si host, user et password sont corrects.
Warning: Unable to create [chemin] No such file or directory in your script on line [numero]	Le chemin vers le répertoire sensé contenir le fichier ou bien le chemin du répertoire dans lequel le fichier doit être créé n'est pas bon (n'existe pas)
Warning: 0 is not a MySQL result index in xxxx.php on line y	Erreur probable au niveau de la requête SQL. Vérifiez votre requête SQL, en particulier les champs manipulés, le nom de ou des tables impliquées, etc. Un petit truc : il est souvent pratique de stocker ses requêtes dans un chaîne.
Warning: Variable \$zzzz is not an array or string in xxxx.php on line y	Vous tentez de manipuler une valeur numérique avec une fonction dédiée aux chaînes ou aux tableaux.
Warning: Variable \$zzzz is not an array or object in xxxx.php on line y	Vous tentez de manipuler une valeur numérique avec une fonction dédiée aux tableaux ou aux objets.
Warning: Cannot add header information headers already sent in xxxx.php on line y	Vous avez tenté d'effectuer un Header après que l'entête HTTP ait envoyé au client. Vérifiez si une sortie (echo, print, message d'erreur) ne se fait quelques temps avant.
Fatal error: Maximum execution time exceeded in xxxx.php on line y	PHP dispose d'un mécanisme permettant de se prémunir des scripts susceptibles d'engendrer un temps d'exécution trop important pouvant saturer un serveur. Par défaut, ce temps est de 30 secondes.
Fatal error: Allowed memory size of 8388608 bytes exhausted (tried to allocate x bytes) in yyyy.php on line z	PHP dispose d'un mécanisme permettant de se prémunir des scripts susceptibles d'engendrer une consommation mémoire trop importante pouvant saturer un serveur. Par défaut, une limite est fixée à environ 8 Mo (8388608 octets).
Fatal Error: Call to undefined function: xxxx() in yyy.php on line z	La fonction que vous appelez n'existe pas. Ce peut-être une fonction liée à une librairie externe (GD, Zlib, PDF, etc.). Dans ce cas, un simple phpinfo() vous renseignera sur les paramètres de compilation de votre version de PHP. Peut-être s'agit-il sinon d'une de vos propres fonctions. Vérifiez alors qu'elle existe. Et dans tous les cas, contrôlez de plus près le nom de la fonction appelée (orthographe, etc.). Une erreur de frappe est vite arrivée.
Fatal Error: Cannot redeclare xxxx() in yyy.php on line z	Vous avez certainement déclaré plusieurs fois la même fonction. Contrôlez à nouveau l'ensemble des fonctions que vous avez créées. Et n'hésitez pas à vérifier également dans les éventuels fichiers inclus. C'est souvent dans un script secondaire que vous trouverez le doublon. Veillez aussi à ne pas utiliser le nom d'une fonction propre à PHP ou à l'une de ses librairies.
Fatal error: Input in flex scanner failed in xxxx on line y	Vérifiez vos include et require. Il y a fort à croire que vous avez indiqué un chemin incomplet (genre /usr/local/ sans préciser de fichier).
Fatal error: Input in flex scanner failed in xxxx on line y	Vérifiez vos include et require. Il y a fort à croire que vous avez indiqué un chemin incomplet (genre /usr/local/ sans préciser de fichier).
Failed opening '%s' for inclusion (include_path='%s')	Le fichier n'a pas pu être inclus dans votre script, car PHP n'a pas pu y accéder : vérifiez les droits (utilisateur PHP, droits du fichier), les noms et chemins du fichier inclus.
file("%s") - Bad file descriptor	Problème d'accès à un fichier avec la fonction file(). Vérifiez bien que l'URL est valide. (l'URL "http://www.super.php") est invalide alors qu'une erreur de type 404 sera valide.
Wrong parameter count for %s()	La fonction est appelée avec un nombre insuffisant de paramètre, ou bien avec trop de paramètres. Certaines fonctions ont besoin d'un minimum de paramètres (array()), et généralement d'un maximum.

stat failed for %s
(errno=%d - %s)

Impossible d'accéder au fichier (problème de droits ou de chemin d'accès).

A noter que si vous avez une parse error à la ligne 30, l'erreur peut se trouver à la ligne de code précédente (par exemple il peut manquer un ; à la fin de la ligne)

Vous pouvez aussi obtenir une parse error "bizarre" en fin de page. Vérifiez bien si vous avez fermé toutes vos accolades.

Problèmes de variables

Parfois, vous pouvez avoir certains problèmes à cause de variables extérieures au script, elles ne semblent pas passer. D'abord assurez vous de ce qui passe vraiment, en imprimant vos variables en début de script, vous verrez ainsi d'un coup d'oeil, si certaines (ou toutes les) variables sont vides. Vous gagnerez au moins 30 minutes, plutôt que de chercher ce qui ne va pas et qui devrait marcher :)

Rappelez vous notre chapitre sur les variables globales (register_globals) et vérifiez bien que vous récupérez les variables par les bons tableaux (d'après la version de PHP sur le serveur).

Vous pouvez utiliser var_dump() ou print_r() pour afficher d'un coup le contenu des tableaux.

Problèmes avec mysql

Ca ne sert à rien de tourner en rond et de jurer qu'une DB ou une table existe, si PHP vous dit le contraire. Avec Mysql il y a quelques bonnes habitudes à prendre.

die()

Cette fonction permet d'afficher un message et d'arrêter le script. Vous devriez toujours l'utiliser.

```
mysql_connect('localhost','root','')  
or die('Problème de connection à mysql');
```

En cas d'erreur, le script s'arrêtera et affichera 'Problème de connection à mysql', de cette façon vous saurez exactement où se trouve le problème dans le script.

```
mysql_connect('localhost','root','')  
or die('Problème de connection à mysql');
```

```
$link = mysql_select_db('tes')  
or die('Prob de DB');
```

Construire votre requête dans une variable.

Prenez l'habitude de construire votre requête dans une variable à part, afin de pouvoir afficher votre requête et débogger facilement.

```
$select = 'select * from aches';  
$result = mysql_query($select,$link)  
or die('requete =>'.$select.'<br>');
```

Utiliser mysql_error()

Cette fonction retournera une erreur mysql sous la forme d'un texte, vous pouvez l'utiliser avec la fonction die()

```
$select = 'select * from aches';  
$result = mysql_query($select,$link)  
or die('requete =>'.$select.'<br>');
```

```
error->' .mysql_error());
```

Résultat

```
requete =>select * from aches
error->Table 'test.aches' doesn't exist
```

Eviter l'affichage si pas de résultat

Si vous tentez d'afficher le résultat d'un select, alors que celui ci ne retourne pas de résultat, vous serez confronté à un message d'erreur. Il vaut mieux donc, prendre l'habitude de vérifier si il y a un résultat à afficher. Pour cela vous pouvez utiliser la fonction `mysql_num_rows()`

```
$select = 'select * from vaches';
$result = mysql_query($select,$link)
or die('requete =>' . $select . '<br> error->' . mysql_error());

$compte = 0;
$compte = mysql_num_rows($result);

if($compte) {
    while($row = mysql_fetch_row($result)) {
        echo $row[0] . '<br>';
    }
}
else {
    echo 'aucun résultats';
}
```

Instruction conditionnelles

If...else

Votre script n'a pas le comportement attendu, il passe dans un if alors qu'il ne devrait pas, ou il passe dans un else alors que vous vous attendiez à ce qu'il passe dans le if.

Ne tournez pas autour du pot! Affichez les variables utilisées dans le test avant le if afin d'être certain qu'elles ont les valeurs attendues.

Etes vous sûrs d'avoir utilisé l'opérateur de comparaison (==) et non l'opérateur d'assignation (=) ?

Les espaces dans les chaînes peuvent être une source de perte de temps également. Imaginez que vous compariez 2 variables de type string dans votre if, mais que l'une d'elles, pour l'une ou l'autre raison ait un espace en début ou en fin de chaîne. Ce qui la rend non égale à l'autre chaîne.

Afin de vérifier ce genre de petites erreurs bêtes. Faites un écho de vos 2 variables et entourez les avec des *, de cette façon, du premier coup d'oeil, vous verrez si un espace s'est glissé dans votre chaîne.

```
$var1 = 'lala';
$var2 = 'lala ';

echo '*' . $var1 . '*<br>';
echo '*' . $var2 . '*<br>';

if($var1 == $var2) {
    echo 'identiques';
}
else {
    echo 'pas identiques';
}
```

Les boucles

Une boucle n'est jamais exécutée? Vérifiez la valeur de départ de la variable sensée déclencher la boucle.

Mon script devient fou et ne s'arrête pas! Vous avez sans doute un problème avec la condition d'arrêt de la boucle (elle n'est jamais atteinte). Dans le cas d'une boucle while, êtes vous certain de ne pas avoir oublié l'itération de votre variable ?

Pour voir le nombre de passages dans la boucle (afin de contrôler un peu) vous pouvez ajouter un echo '.', ; cela vous permettra de voir combien de fois la boucle est exécutée et voir si c'est normal. Rien ne vous empêche d'utiliser une variable \$debug servant "d'interrupteur" au debug. Exemple :

```
$i = 0;
$debug = 1;
while($i < 100) {
    if($debug) {
        echo '.';
    }
    $i = $i + 1;
}
```

Si vous passez \$debug à 0, les echo de debug ne seront pas affichés, si vous avez besoin d'afficher des variables afin d'en contrôler la valeur, vous n'avez plus qu'à réactiver votre interrupteur.

Les fichiers

Une erreur d'include, de require ou de fopen peut avoir plusieurs causes.

- | Le chemin vers le fichier n'est pas correct.
- | Les permissions sur le fichier ne sont pas suffisantes.
- | Certaines fonctions d'information sur les fichiers ne fonctionnent pas sous windows (ce qui est en général est indiqué dans la doc).

La fonction mail()

Vous devez bien vous douter que les mails ne s'envoient pas par vaches volantes (non? même pas un petit peu?).

Pour envoyer un mail, vous avez besoin ou bien :

- | d'avoir un serveur SMTP sur le serveur qui host votre script
- | ou bien de configurer votre php.ini avec un serveur SMTP que vous pouvez utiliser (celui de votre FAI par exemple)
- | Et surtout d'avoir configuré le php.ini en conséquence

```
[mail function]
; pour windows
SMTP = smtp.monfai.fr

; pour windows
sendmail_from = test@moi.fr

; pour unix.
;sendmail_path =
```

Si vous obtenez cette erreur en utilisant la fonction mail(), rappelez vous de mon histoire de vaches volantes et vérifiez la configuration de PHP. N'oubliez pas de redémarrer Apache pour que les infos soient prises en compte (merci oreil)

```
Warning: Failed to Connect in [chemin] on line [nombre]
```

Pas à pas

Rien ne fonctionne comme vous voulez? vous devenez fou à essayer de corriger votre script ? :) Restez Zen! Deboggez votre script partie par partie, prenez le temps de tracker vos variables en les affichant avant et après certains traitements.

Si ça ne va toujours pas, allez prendre l'air, ou reprenez votre travail le lendemain, vous verrez qu'une bonne nuit de sommeil peut faire des miracles :)

Variables globales à OFF

Travailler avec les variables globales à OFF

Depuis peu, peut être avez vous rencontré certains problèmes avec les versions récentes de PHP pour la manipulation de variables.

Avant, pour récupérer vos variables, vous pouviez le faire directement directement par leur nom, quelque soit le type de variables(passées par la méthode POST/GET,cookie,session,....

Cela nous permettait de faire ceci :

```
Récupération directe de variables
un appel au script :
http://www.mondomaine.fr/monscript.php
?truc=coucou
<?
    echo 'Ma variable passée par url =>'.$truc;
?>
affichant :
Ma variable passée par url =>coucou
```

La variable \$truc, passée par URL est directement accessible depuis le script PHP. Cette façon de travailler à l'air, certes bien pratique, elle laisse néanmoins la porte ouverte à des trous de sécurités.

Ce qui nous permettait d'utiliser nos variables de cette façon là est en fait une option se trouvant dans le fichier PHP.ini qui s'appelle : register_globals et qui est par défaut initialisée à ON dans ce fichier.

Depuis la version 4.2.0 de PHP, cette option est par défaut initialisée à OFF.

C'est pourquoi il est important de prendre dès maintenant l'habitude de travailler avec register_globals à OFF. De plus, beaucoup d'hébergeurs ayant upgradé PHP avec la nouvelle version, laissent register_globals à OFF...Et puis, cette méthode permet de résoudre pas mal de trous de sécurité dans les scripts.

Une autre option importante de la configuration PHP : track_vars, positionnée à ON, celle ci nous permet de récupérer les variables passées par formulaire, url et cookies dans des tableaux prédéfinis.

Avec ces 2 options configurées telles que décrites ci-dessus, nous pouvons utiliser les tableaux associatifs suivants pour récupérer nos variables :

Tableaux associatifs

\$HTTP_GET_VARS	Permet de récupérer les variables passées par url ou par la méthode GET d'un formulaire
\$HTTP_POST_VARS	Permet de récupérer les variables envoyées par la méthode POST d'un formulaire
\$HTTP_POST_FILES	Permet de récupérer les variables de fichiers envoyés par un formulaire
\$HTTP_COOKIE_VARS	Permet de récupérer les cookies
\$HTTP_SESSION_VARS	Permet de récupérer les variables de session
\$HTTP_ENV_VARS	Permet de récupérer les variables d'environnement données par PHP
\$HTTP_SERVER_VARS	Permet de récupérer les variables serveur envoyées par le serveur HTTP

A partir de PHP 4.1.0, ces variables changent de nom pour prendre des noms plus courts (et plus facile à retenir :p).

Avant 4.1.0

```
$HTTP_GET_VARS
$HTTP_POST_VARS
$HTTP_POST_FILES
$HTTP_COOKIE_VARS
$HTTP_SESSION_VARS
$HTTP_ENV_VARS
$HTTP_SERVER_VARS
```

A partir de 4.1.0

```
$_GET
$_POST
$_FILES
$_COOKIE
$_SESSION
$_ENV
$_SERVER
```

Mise en pratique

Ok..ok, mais comment ça marche? Nous allons voir quelques petits exemples dans lesquels nous allons manipuler ces tableaux magiques...

1. Formulaires

Récupération d'informations provenant d'un formulaire utilisant la méthode POST.

Récupération d'un formulaire exemple1.php

```
<?
$bouton = $_POST['send'];
if($bouton) {
    $nom = trim($_POST['nom']);
    $prenom = trim($_POST['prenom']);
    if($nom && $prenom) {
        echo 'Bonjour, '.$prenom.' '.$nom;
    }
    else {
        echo 'vous n\'avez pas rempli tous
        les champs';
    }
}
?>
```

Formulaire utilisé form.html

```
<form action="exemple1.php" method="post">
<table>
<tr>
<td>Nom :</td>
<td>
<input type="text" name="nom"
value="">
</td>
</tr>
<tr>
<td>Prénom :</td>
<td>
<input type="text" name="prenom"
value="">
</td>
</tr>
</table>
<input type="submit" value="envoyer"
name="send">
</form>
```

Comme vous le voyez, nous récupérons la valeur des variables grâce au tableau associatif \$_POST dans lequel une entrée a été créée avec le nom de chaque variable du formulaire.

Pour ceux qui se demandent à quoi sert la fonction trim(), elle retire les espaces en début et fin de chaîne, cela nous

permet de ne pas traiter des informations vides (un utilisateur pourrait très bien n'entrer que des espaces dans le formulaire)

Récupération d'un fichier.

Bien, maintenant voyons comment récupérer les informations concernant l'envoi de fichiers par une formulaire

Voici comment récupérer toutes les informations du fichier envoyé avec le tableau associatif \$_FILES

Entrée	Explications
\$_FILES['variable']['name']	Le nom original du fichier qui provient de la machine de l'utilisateur
\$_FILES['variable']['type']	Le type mime du fichier
\$_FILES['variable']['size']	Le taille du fichier en bytes
\$_FILES['variable']['tmp_name']	Le nom temporaire du fichier stocké sur le serveur
\$_FILES['variable']['error']	le code erreur associé à l'upload (attention cette option à été ajoutée en PHP 4.2.0)

Cet exemple ne fait qu'afficher les informations concernant le fichier que l'on a uploadé.

Récupération de fichiers

```
<?
$bouton = $_POST['bouton'];
if($bouton) {
    $fichier = $_FILES['fichier']['name'];
    $size = $_FILES['fichier']['size'];
    $tmp = $_FILES['fichier']['tmp_name'];
    $type = $_FILES['fichier']['type'];
    $error = $_FILES['fichier']['error'];
    $max = $_POST['max'];
    Récupération de fichiers
    if($fichier) {
        if($size <= $max) {
            echo 'Nom d\'origine => '.$fichier.'  
';
            echo 'Taille => '.$size.'  
';
            echo 'Nom sur le serveur => '.$tmp.'  
';
            echo 'Type de fichier => '.$type.'  
';
            echo 'Code erreur => '.$error.'  
';
        }
        else {
            echo 'Le fichier est trop volumineux';
        }
    }
    else {
        echo 'aucun fichier envoyé';
    }
}
?>
<form enctype="multipart/form-data"
action="exemple2.php"
method="post">
<input name="fichier" type="file">
<input type="hidden" name="max" value="10000">
<input type="submit" value="send" name="bouton">
```

```
</form>
```

Récupérer les informations d'un formulaire facilement.

Imaginons que vous vouliez simplement afficher les informations entrées dans un formulaire. Vous pouvez parcourir le tableau \$_POST afin d'afficher tout ce qu'il contient.

La variable \$key contient le nom de la variable formulaire et la variable \$val contient la valeur entrée dans le formulaire.

```
Afficher toutes les infos d'un formulaire
<?
if($_POST) {
foreach($_POST as $key=>$val) {
    echo $key.'=>'.$val.'<p>';
  }
}
else {
    echo 'le formulaire n\'a pas été envoyé';
}
?>
<form action="exemple3.php" method="post">
Nom :<input type="text" name="nom"
value="">
<br />
Prénom:<input type="text" name="prenom"
value="">
<br />
Adresse:<input type="text" name="adresse"
value="">
<br />
Ville:<input type="text" name="ville"
value="">
<br />
Pays:<input type="text" name="pays"
value="">
<br />
<input type="submit" value="envoyer">
</form>
```

2. La Méthode get et le passage par URL

Rien de bien compliqué, il suffit de récupérer les variables passées en get par le tableau \$_GET. Un petit exemple pour illustrer quand même.

```
Passage par URL
http://www.monsite.fr/exemple4.php?nom=cow
&prenom=flying
<?
$nom = $_GET['nom'];
$prenom = $_GET['prenom'];
echo 'Bonjour, '.$prenom.' '.$nom;
?>
```

3. Les cookies

là nom plus rien de particulier

```
Récupération d'un cookie
<?
$cookie = $_COOKIE['moncookie'];
if($cookie){
    echo 'Valeur du cookie=>'.$cookie;
```



```
}
?>
```

4. Les variables session

En récupérant vos variables sessions par le tableau associatif \$_SESSION vous êtes sûrs que la valeur est bien une variable session. Voyons un exemple ou nous récupérerons une variable session directement.

Le fichier session.php, débute une session si il n'en existe pas encore pour l'utilisateur.

Fichier ou l'on crée une entrée en session si le login et mot de passe entrés sont corrects, auquel cas, on initialise la variable session \$login, avec la chaîne 'ok'.

```
Variables session
<?
include 'session.php';
$bouton = $_POST['bouton'];
if($bouton) {
$log = trim($_POST['log']);
$pass = trim($_POST['pass']);
if ($log=='jenny' && $pass == 'foo') {
    $login = 'ok';
    session_register('login');
    echo '<a href="exemple7.php"> >> suite >></a>';
}
else {
    echo 'mauvais login/pass';
}
}
?>
<form action="exemple6.php" method="post">
Login :<input type="text" name="log"
value=""><br />
Pass :<input type="text" name="pass"
value=""><br />
<input type="submit" value="envoyer"
name="bouton">
</form>
```

Voici comment récupérer les variables session par le bon tableau.

```
Récupération d'une session
<?
include 'session.php';
if($_SESSION['login'] == 'ok') {
    echo 'vous êtes bien connecté';
}
else {
    die('pas en session');
}
?>
```

5. Les variables d'environnement

utilisation du tableau \$_ENV pour récupérer les variables d'environnement en voici quelques unes utiles.

Les variables d'environnement

```
<?
echo 'Nombre de process actifs=>';
```

```

echo $_ENV[ 'NUMBER_OF_PROCESSORS' ].'

';
echo 'Système d\'exploitation=>';
echo $_ENV[ 'OS' ].'

';
echo 'Chemin du répertoire temporaire=>';
echo $_ENV[ 'TMP' ].'

';
echo 'Chemin du profil utilisateur=>';
echo $_ENV[ 'USERPROFILE' ].'

';
?>

```

6. Les variables serveurs

utilisation du tableau \$_SERVER pour récupérer les variables serveur, voici quelques exemples.

Les variables serveur

```

<?
echo 'Chemin du script courant=>';
echo $_SERVER[ 'PHP_SELF' ].'

';
echo 'Nom du serveur=>';
echo $_SERVER[ 'SERVER_NAME' ].'

';
echo 'Variables passées au script=>';
echo $_SERVER[ 'QUERY_STRING' ].'

';
echo 'Document root=>';
echo $_SERVER[ 'DOCUMENT_ROOT' ].'

';
echo 'Référéant=>';
echo $_SERVER[ 'HTTP_REFERER' ].'

';
echo 'Adresse ip de l\'utilisateur=>';
echo $_SERVER[ 'REMOTE_ADDR' ].'

';
?>

```

Vous pouvez avoir la liste des variables serveurs et d'environnement en créant le petit fichier PHP ci-dessous et en l'exécutant.

```

<?
phpinfo();
?>

```

7. Récupérer facilement les variables

Pour récupérer facilement les variables, vous pouvez utiliser la fonction PHP extract. Elle va exporter votre tableau associatif et créer une variable pour chaque clé du tableau.

```
extract($_POST, EXTR_OVERWRITE);
```

Cette fonction va créer une variable pour chaque clé du tableau associatif \$_POST. Si on a :

```
| $_POST['nom']
| $_POST['prenom']
| $_POST['age']
```

La fonction extract() va créer les variables suivantes :

```
| $nom
| $prenom
| $age
```

Le second argument sert à gérer les collision de variables. Donc sert à dire ce que l'on fait si une variable existe déjà. Par défaut les variables qui existent déjà seront écrasées.

Type	signification
EXTR_OVERWRITE	Ecrase les variables existantes
EXTR_SKIP	N'écasse pas les variables existantes
EXTR_PREFIX_SAME	Si une variable existe déjà, une nouvelle variable est créée avec un préfix donné en 3ème argument à la fonction
EXTR_PREFIX_ALL	Crée de nouvelles variables avec le préfix passé en 3ème argument pour toutes les clés du tableau
EXTR_PREFIX_INVALID	Crée de nouvelles variables avec le préfix passé en 3ème argument pour les noms de variable invalides (par exemple \$1)

Voilà un petit exemple qui montre comment fonctionne la fonction extract. La variable "\$nom" existant déjà, une nouvelle variable va être créée "\$new_nom".

```
<?php
$nom = 'blabla';
$_POST['nom'] = 'jenny';

extract($_POST, EXTR_PREFIX_SAME, 'new');

echo 'variable "nom" =>' . $nom . '<br>';
echo 'variable "new_nom" =>' . $new_nom . '<br>';
?>
```

Résultat de cet exemple :

```
variable "nom" =>blabla
variable "new_nom" =>jenny
```