14.1. Le langage PHP - MySql

1 Introduction - 2. Utilisations du PHP. - 3. Utilitaires

Dans la partie précédente de la formation Internet, nous avons créé un site Web en HTML avec FrontPage, donc statique. Ces notes de cours en ligne reprennent le développement en **PHP et MySQL** (sites dynamiques), elles font partie de la formation YBET "Création de site" donnée en nos locaux ou en entreprise et permettent de donner des bases sur le langage PHP couplé à une base de donnée MySQL. Cette formation parle de programmation, elle est donc plutôt compliquée, mais j'essaye de suffisamment dissocier les différentes procédures dans ces notes. Ce cours est corrigé et mise à jour en fonction des difficultés rencontrées lors des séances dans les salles de formation: exercices, notes complémentaires, ...

Si l'apprentissage du langage et des procédures est important, une partie de cette formation va également permettre de créer un site simple ou modifier des sites en GNU/GPL (portals, OScommerce, livres d'or, forum, ...). L'exercice lié à ce cours est la création d'un site de petites annonces. Le site associé, créé en partie par les "élèves" est dahut.be.

Commençons notre formation PHP - MySQL par quelques rappels sur les langages utilisés sur créer des sites INTERNET, leurs différents avantages et inconvénients, leur mise en ligne. Avec l'HML et l'ASP, le PHP est un des principal langage utilisé pour créer un site INTERNET. Une page en HTML est directement décodée par votre navigateur (Internet explorer ou Firefox par exemple). Par contre, l'ASP et le PHP sont décodées par le serveur qui transfère le résultat vers le navigateur. Dans ce sens, ces 2 langages permettent une programmation de vos pages Internet et même une interactivité entre le site et l'utilisateur (choix des couleurs, identifiant, ...). Le langage PHP est généralement installé sur les serveurs WEB Linux associé à une base de donnée Mysql (même si les serveurs sous Windows l'acceptent). Le langage ASP, associé à une base de donnée SQL, est réservé aux serveurs sous Windows.

Dans la suite de ce cours les tableaux en fond gris reprendront les commandes

Lignes de commandes

Les tableaux en fond jaune reprendront des aides:

Aides, fonctionnement des procédures.

2. Utilisations de PHP

Le langage PHP est directement décodé par le serveur Internet qui renvoie le résultat au format HTML vers le navigateur. Pour l'utilisateur, cette fonctionnalité est complètement transparente. Comme les pages du site sont "programmées", les possibilités sont presque illimitées. Les pages créées peuvent afficher des paramètres personnalisés (l'heure, les indications spécifiques au visiteur), créer des formulaires personnalisés avec une vérification des données, afficher des fichiers textes remaniés, ... PHP est généralement couplé à une base de donnée MySQL. Ce n'est pas obligatoire mais permet des application étendues comme la

création d'un forum, d'un site de vente en ligne, ... Vous trouverez de multiples exemples téléchargeable gratuitement sur Internet.

Créer des pages en PHP n'oblige pas à créer tout le site dans ce langage. Les pages peuvent être mélangées avec d'autres en html sur le même site. La seule manière de vérifier le type de page est de vérifier l'extension de la page Internet, et encore puisque des pages en HTML peuvent être simplement renommée en PHP pour des développements futurs.

Les extensions suivantes sont liées aux pages PHP suivant les versions utilisées par votre hébergeur:

- **phtml**: version Php/FI 2.0
- **PHP3**: version PHP 3.0
- **PHP** pour les versions 3.0, 4.0 (sortie officiellement en 2000), 4.11, 4.3, ... La version 5.0 est sortie en 2004, la 6 est en cours de développement.

Pour vérifier quelle version utilise votre hébergeur, tapez les lignes suivantes dans un éditeur:

```
<?php

phpinfo();

?>
```

enregistrez le fichier comme info.php (par exemple et attention à l'extension txt ajoutée automatiquement par NotePad si vous n'affichez pas les extensions sous Windows) et transférez ce fichier sur le serveur. En exécutant le fichier sur votre navigateur (par exemple http://www.mon-site.com/info.php), vous recevrez une fenêtre équivalente à ceci (même si le résultat est généralement beaucoup plus long) qui affiche la configuration sur le serveur.

PHP Version 4.3.10-16

System	Linux frontal1 2.6.8-2-386 #1 Tue Aug 16 12:46:35 UTC 2005 i686
Build Date	Aug 24 2005 20:03:04
Configure Command	'../configure' '--prefix=/usr' '--enable-force-cgi-redirect' '--enable-fastcgi' '--with-config-file-path=/etc/php4/cgi' '--enable-memory-limit' '--disable-debug' '--with-regex=php' '--disable-rpath' '--disable-static' '--with-pic' '--with-layout=GNU' '--with-pear=/usr/share/php' '--enable-calendar' '--enable-sysvsem' '--enable-sysvshm' '--enable-sysvmsg' '--enable-track-vars' '--enable-trans-sid' '--enable-bcmath' '--with-bz2' '--enable-ctype' '--with-db4' '--with-iconv' '--enable-exif' '--enable-filepro' '--enable-ftp' '--with-gettext' '--enable-mbstring' '--with-pcre-regex=/usr' '--enable-shmop' '--enable-sockets' '--enable-wddx' '--disable-xml' '--with-expat-dir=/usr' '--with-xmlrpc' '--enable-yp' '--with-zlib' '--without-pgsql' '--with-kerberos=/usr' '--with-openssl=/usr' '--with-zip=/usr' '--enable-dbx' '--with-mime-magic=/usr/share/misc/file/magic.mime' '--with-exec-dir=/usr/lib/php4/libexec' '--without-mm' '--without-mysql' '--without-sybase-ct'
Server API	CGI/FastCGI
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/etc/php4/apache/php.ini

Nous venons finalement de programmer notre première commande en PHP. Analysons les lignes de ce fichier:

<?php démarre une liste de commande PHP

phpinfo(); est une commande du langage de programmation qui affiche la configuration du serveur. Remarquez que la ligne de commande est terminée par **;** En cas d'omission, vous recevez un message d'erreur.

?> termine une liste de commandes.

Vous pouvez utilisez plusieurs séries de commandes (chaque fois avec les mêmes délimiteurs) dans une même page, entrecoupée de parties en html. L'extension du fichier sera néanmoins .php.

3. Utilitaires.

Programmer en PHP nécessite quelques programmes et utilitaires spéciaux pour débuter:

- **Editeur PHP**: Vous pouvez créer des pages avec Notepad, mais des logiciels gratuits et plus professionnels sont téléchargeables sur Internet. Personnellement, j'utilise ConText, un freeware (gratuit) utilisable en plusieurs langues. HydraPHP, PHPedit, ... sont aussi possibles.

- **Serveur personnel:** Une page en PHP doit être décodée par un serveur et ... votre ordinateur sous Windows ne fonctionne pas comme serveur php. Pour tester vos programmes en PHP - MySQL, vous pouvez télécharger logiciel **easyphp**. Il fonctionne sous Windows et permet non seulement de tester des pages en local, mais également de créer des bases de données locales Mysql. EasyPhp n'est pas seulement un logiciel de test, c'est une réelle application serveur qui peut vous permettre d'héberger votre site Internet sur votre ordinateur en local, mais **attention à la sécurité**. Toutes les commandes ne fonctionnent néanmoins pas comme les envois de mails qui nécessitent un serveur pop ou quelques fonctions sur les images.

Ces 2 utilitaires vous permettent finalement d'utiliser votre PC sous Windows pour créer et tester des pages programmées en PHP, pour les retransférer vers votre site Internet ensuite

14.2. Installer EasyPhP.

1 Introduction - 2. Installation d'EasyPhp. - 3. Utilisation en php

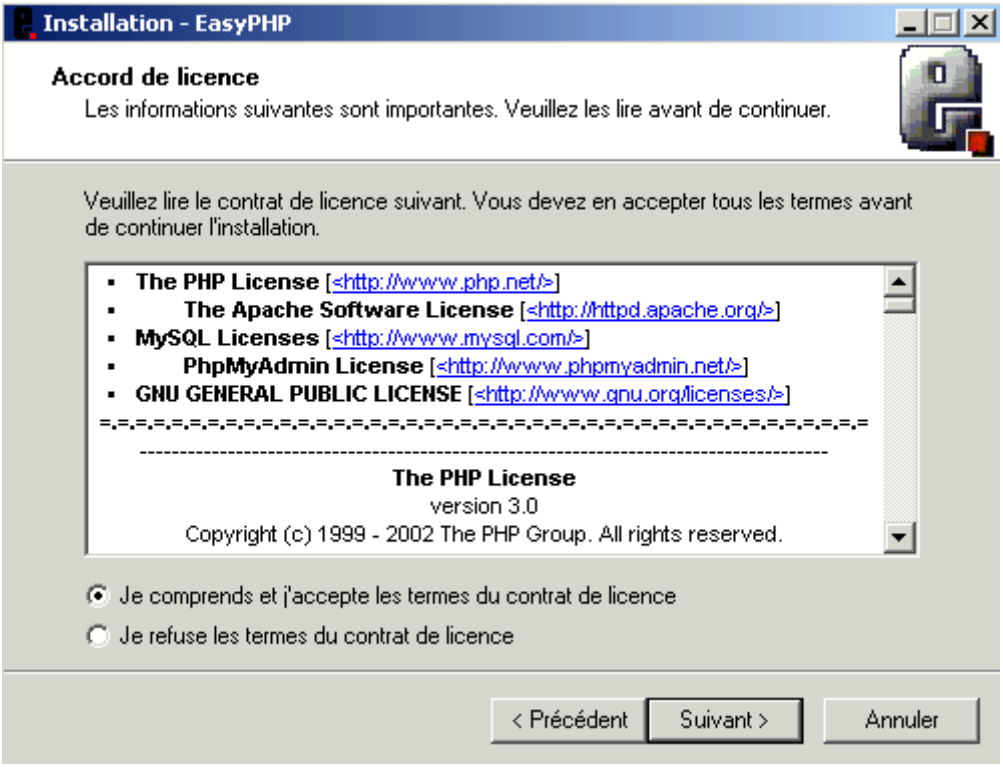
Le langage PHP nécessite un serveur pour décoder les instructions. De même une base de donnée MySQL nécessite un gestionnaire. La majorité des hébergements Internet proposent ces possibilités (sauf quelques serveurs sous Windows). Tester les applications directement sur un hébergeur nécessite de taper les lignes de commandes dans l'éditeur, de le transférer via un serveur FTP, de le tester, ... et en cas d'erreur, de recommencer. Dans le cas d'une base de donnée MySql, le problème est identique sauf que la majorité des hébergements ne proposent qu'une seule base de donnée (mais une base de donnée peut fonctionner pour plusieurs applications). Cette solution risquerait de corrompre la base et de rendre inaccessible un site "en production". **Bref, ce n'est pas la solution idéale.**

La solution standard est d'installer le logiciel gratuit (en fait en GNU) **EasyPHP** sur votre ordinateur en local. EasyPhp émule un serveur apache sous Windows compatible avec PHP qui permet via PHPMyadmin de gérer des bases de données MySQL. Attention, ce logiciel est un réel gestionnaire réseau, d'où quelques risques de sécurité. Il ne doit être démarré que pour les tests de vos programmes en PHP.

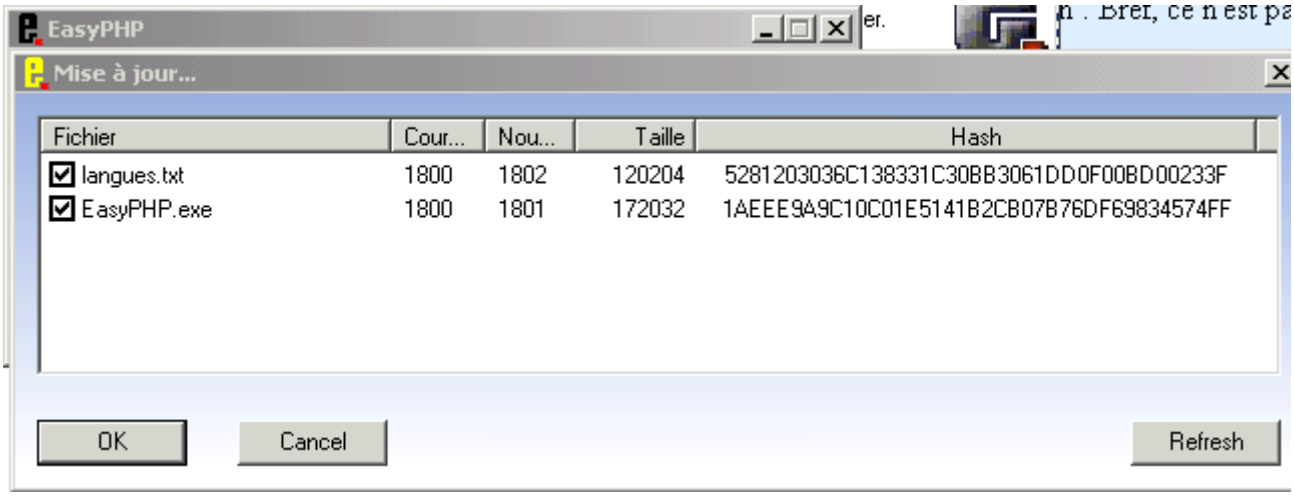
Quelques commandes spécifiques ne fonctionnent pas sur EasyPhp par rapport à un hébergement Internet (mail, gestion des images, ...) mais cette solution est largement assez complète pour tester la majorité des applications.

2. Installation d'EasyPhp

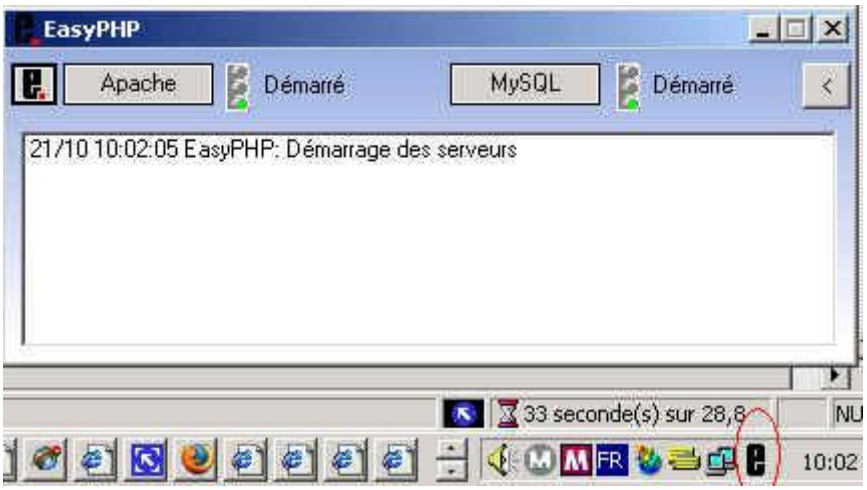
La première chose est de le télécharger sur le site Internet d'easyphp.org. Sélectionnez la version 2.0 (la dernière en date). Cette version est compatible PHP 4 et 5. La version 1.8 est compatible PHP 3 et 4. Ceci peut poser quelques problèmes de compatibilités entre vos programmes et l'hébergement, vérifiez la version sur votre serveur avant. Sélectionnez le site miroir le plus adapté pour le téléchargement. Commencez l'installation.



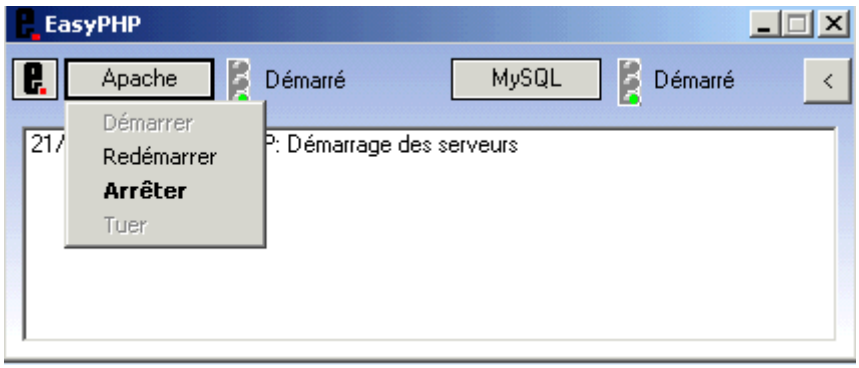
La fenêtre suivant rappelle les conditions d'utilisation. Sélectionnez le dossier par défaut pour l'installation: C:\Program Files\EasyPHP1-8. Le reste de l'installation se fait sans problèmes.



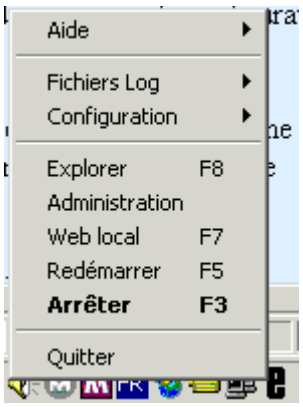
La fenêtre suivante propose le téléchargement des mises à jour. Une fois téléchargées, fermez Easy Php et redémarrez-le via Démarrer -> Programmes -> EasyPhp ->EasyPhp. Deux indications vont vous permettre de vérifier si le serveur est démarré sur votre ordinateur sous Windows en local: une fenêtre et un petit logo dans la barre des tâches à coté de l'heure comme ci-dessous.



Par la fenêtre, vous pouvez arrêter le PHP ou la gestion MySQL:



Par l'icône, vous pouvez récupérer la fenêtre ci-dessus en double cliquant ou accéder au menu de configuration avec la touche contextuelle de la souris (click droit). Ce menu est la base du fonctionnement d'EasyPhp:

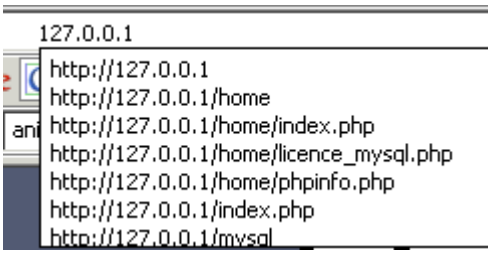


- **Aide** renvoie à différentes aides sur le fonctionnement ou sur le langage PHP, généralement via des sites.
- **Log** permet de vérifier les messages d'erreurs Apache, PHP ou MySQL
- **Configuration** permet de configurer le logiciel (démarrage automatique, ...)
- **Explorer** donne l'accès à l'explorateur Windows
- **Administration** permet d'administrer les différents composants, notamment de gérer les bases de données.
- **Web** local permet d'accéder à votre navigateur Internet en local
- **Redémarrer** et **arrêter** permet de quitter EasyPhp.

3. Utilisation en php.

L'utilisation de nos programmes en php (en html) va passer par un sous dossier Windows d'EasyPhp, soit C:\Program Files\EasyPHP1-8**www**. Les fichiers à tester doivent obligatoirement se trouver dans ce dossier ou dans un de ses sous-dossier.

Démarrez Internet explorer (ou Firefox) et dans la barre d'adresse, tapez 127.0.0.1, c'est l'adresse utilisée automatiquement.



Au premier démarrage, la page d'accueil va afficher la page 127.0.0.1/index.php, en fait le le fichier index.php inclus dans le dossier Windows. Pour tester vos programmes, il vous suffit de les insérer dans ce dossier et d'utiliser l'adresse 127.0.0.1/nom_fichier.

ATTENTION: Windows ne fait pas la distinction entre les minuscules et majuscules, Linux oui. Si votre fichier s'appelle INDEX.php, en local, index.php, INdex.php, ... seront identiques. Par contre, si vous transférer le fichier sur un hébergement, ils seront vus comme des fichiers différents. Par facilité, **tapez tous les noms de fichiers en minuscule**.

L'utilisation d'EasyPhP en mode MySql sera vu dans le chapitre Créer une base de donnée en local. Pour EasyPhp, le nom du serveur est toujours localhost, l'utilisateur root (éventuellement root@localhost), mot de passe vide. Ceci peut-être modifié via les options.

14.3. Premières commandes en PHP

Pour débuter ce cours de développement de sites Web, nous allons nous inspirer d'un page internet écrite en HTML pour le modifier en PHP. Nous en profiterons pour voire quelques commandes simples comme l'affichage et les formats de dates.

2. Premier fichier PHP

Partons du fichier htm suivant (par exemple écrit avec FrontPage et en copiant la partie HTML dans l'éditeur PHP):

- <HEAD>
- <TITLE>Création d'un fichier PHP</TITLE>
- </HEAD>
- <BODY>
- <H1>Ma première commande</H1>
- Ceci fait partie de la formation INTERNET YBET
- </BODY>

Ce fichier HTM est relativement simple, il ne fait que mettre un titre dans l'entête et afficher un texte dans un paragraphe. Modifions la programmation pour le créer en PHP. Le header n'est pas modifié. Nous allons simplement remplacer Ceci fait partie de la formation INTERNET YBET par :

```
<?php
print('Ceci fait partie de la formation INTERNET YBET');
```

```
?>
```

La commande **print()** est insérée entre deux balises qui délimitent le PHP. Elle permet d'afficher un texte sur l'écran. Les autres parties en HTML ne sont pas modifiées. Chaque ligne est terminée par**;** Si vous l'oubliez vous aurez un message d'erreur de type: **Parse error**: parse error, unexpected T_PRINT in **/home/clients/www/ybet.be/www/tests.php** on line **5**

Nous aurions pu utiliser la commande **ECHO** similaire. Les lignes de commandes deviennent:

```
<?php
echo"Ceci fait partie de la formation INTERNET YBET';
?>
```

Remarquez qu'à la différence de la commande Print(), nous n'utilisons pas les parenthèse. Par contre, pour afficher du texte, nous utilisons également les guillemets (simples ' ou doubles " au choix, mais les doubles sont préférables pour la suite du cours).

Et si nous désirons afficher **un texte comprenant un guillemets**? Vous devez insérer le caractère \devant le guillemet.

Par exemple **print('J\'ai réussi ma première commande en PHP');**

- Les caractères de contrôle html
, <hr>, .. Par exemple sont également acceptés: print("
"); insère un saut de ligne.
- Pour **afficher une URL**, echo"Le site YBET";
- Pour afficher un texte et une variable (ou plus) dans la même ligne de commande: print($ligne." Lignes affichées"); Le **.** sert de séparation.

3. Les variables

Le langage PHP permet de manipuler des variables et constantes. Le PHP accepte 6 types de constantes:

- Booléen: True, False (vrai ou faux)
- Entier: un nombre entier (sans virgules)
- Chiffre en virgule flottante: (c'est automatiquement un nombre décimal de type double), nombre avec des chiffrée derrière la virgule comme 0.23, 45.2369, ... Remarquez le point comme séparateur.
- Chaîne de caractères: du texte encadré par ', exemple: 'Ceci est un texte'
- Tableaux: array(2,2,2)
- Objets (image, lien texte, ...)

PHP ne demande pas de spécifications du type de variable préalable, ni même de les déclarer (sauf les tableaux que nous verrons au chapitre 5).

Les noms de variable doivent:

- commencer par $
- inclure des lettres, des chiffres et le caractère _
- commencer par une lettre ou _

Les noms de variables **ne doivent pas**:

- inclure les caractères réservés - @ , . ; : /<\>
- inclure des espaces

Quelques **remarques** pour le nom:

- Il peut inclure des caractères accentués mais ce n'est pas souhaitable.
- le nom de la variable est spécifique à la casse (majuscules, minuscules)

L'exemple suivant alloue la date du jour à la variable jour_creation et l'affiche.

```
<?php
$jour_creation= '22/12/2008';
Print($jour_creation);
?>
```

Comme nous imprimons une variable, les guillemets ne sont pas utilisés.

Vous pouvez également afficher plusieurs chaînes sur une même ligne en séparant les parties par un . L'addition de 2 variables textes entre elles utilise également le . Par exemple:

```
<?php
$jour_creation= '22/12/2008';
Print("La date de création de ce fichier est ".$jour_creation);
$texte="Bonjour";
$prenom="Patrick";
$message=$texte." ".$prenom;
print($message);
// affiche Bonjour Patrick
?>
```

4. Les constantes.

Une variable va changer de contenu suivant le programme mais parfois nous allons utiliser la même valeur pour tous le programme. Une constante se définit par la commande DEFINE()

```
<?php
define("VALEUR_FIXE","valeur");
define("VALEUR_FIXE_2",19);
?>
```

Même si ce n'est pas obligatoire, les constantes sont souvent nommées en majuscules, ceci facilite la lecture du programme. Pour les afficher ou les utiliser

```
<?php
Print ("La valeur de la constante VALEUR_FIXE est ".VALEUR_FIXE);
ECHO "et VALEUR_FIXE_2 prend la valeur ".VALEUR_FIXE_2;
?>
```

Remarquez que nous avons directement utilisé son nom, sans $ comme pour les variables.

5. Les formats de dates

Le langage PHP inclut une fonction gérant les dates et heures. La fonction **date(format,optionnel)** renvoie une chaîne de caractère suivant le format choisi.

Principaux opérateurs pour la fonction date()	
j	Jour du mois, sans les 0, soit de 1 à 31
d	Jour du mois avec 2 chiffres, soit de 01 à 31
D	Jour de la semaine sur 3 lettres, en anglais
l	Jour de la semaine (en anglais)
m	Mois, de 01 à 12
n	Mois, sans les 0 initiaux (de 1 à 12)
M	trois premières lettres du mois en anglais
F	non du mois complet
Y	année en 4 chiffres
y	année en 2 chiffres
z	jour de l'année (de 0 à 365)
h	heure, de 0 à 12
H	heure, de 0 à 24
s	secondes avec les 0, de 00 à 59
S	secondes, sans les zero.
a	am pour le matin, pm pour l'après-midi
A	AM pour le matin, PM pour l'après-midi

Quelques exemples de la commande date():

- date('d/m/Y') affiche 08/09/2006

- date('D d F Y h:s') affiche Fri 08 september 2007 17:56

L'utilisation de la partie optionnelle permet par exemple d'afficher la date de modification de la page, comme **filemtime**.

```
<?php
print('Créé le 25/02/2007');
print(', modifié le ');
print(date('d/m/y',filemtime('index.php')));
?>
```

Affiche le message: Créé le 25/02/2008 , modifié le 28/12/2008.

Si vous souhaitez afficher ce texte sur deux lignes, insérez la commande print('
'), soit

```
<?php
print('Créé le 25/02/2006<br>');
print("modifié le ".date('d/m/y',filemtime('index.php')));
?>
```

Nous pouvons également combiné avec des caractères de contrôle, comme par exemple mettre en gras la date avec texte . Ceci nous donne:

```
<?php
print('Créé le 19/10/2006<br>');
print("modifié le <b>".date('d/m/y',filemtime('index.php'))."</b>");
?>
```

6. Commentaires

Un commentaire permet d'insérer des lignes dans un programme qui ne seront pas exécutées. Il y a 2 méthodes pour insérer des commentaires dans un programme PHP:

- débuter la ligne par //
- insérer un ensemble de lignes entre /* et */

Exemple:

```
<?php
/*
Cette partie rassemble différentes lignes de commentaires PHP
*/
Print('Nous avons inséré plusieurs lignes de commentaires');
// une ligne de commentaire seulement .
?>
```

Le seul résultat sera l'affichage du message.

7. A retenir, exercice

Même si nous débutons, les quelques parties de ce chapitre nous ont déjà permises de

- afficher un message de type texte
- d'afficher la date et l'heure
- d'utiliser des variables.
- d'insérer des commentaires pour faciliter la maintenance du code de notre page

Le chapitre suivant va nous permettre de créer des fonctions en PHP.

Exercice: créer un script en PHP qui affiche votre prénom suivi de "nous sommes le " date du jour.

14.4. Fonctions en PHP

Dans le chapitre précédant de cette formation, nous avons utilisé les fonctions **print()** et **echo**. Cette partie du cours PHP va nous permettre de créer nos propres fonctions et d'utiliser des fonctions plus complexes pour la création de notre site Internet.

2. Créer ses propres fonctions

Créer une fonction en PHP va nous permettre de définir des fonctionnalités supplémentaires dans nos pages Internet et de les réutiliser au sein d'une même page. En php3, les fonctions doivent être établies au préalable (en début de page), plus en PHP4. La syntaxe d'une fonction est:

```
function nom-fonction($argument1, $argument2) {
// code php
return $variable;
}
```

FUNCTION (mot réservé) permet de spécifier que les lignes suivantes (encadrées par { et }) reprennent les arguments d'une fonction.

$argument1 et $argument2 sont des variables transférées à la fonction, $variable, le résultat (précédé de return).

Comme exemple, créons la fonction qui calcule et affiche un prix TVA comprise suivant un prix de départ et un taux de TVA

```
<?php
function prix_tvac($prix_htva,$taux_tva)
{
return ($prix_htva*(1+$taux_tva/100));
}
print(prix_tvac(19.23,21));
?>
```

Ces lignes affichent le prix TVAc d'un produit de 19,23 € pour un taux de 21 %, soit 23,2683

Remarques:

1. Si vous transférez **trop d'arguments**, les arguments excédentaires sont ignoré.
2. **Si le nombre d'arguments** est insuffisant, la fonction renvoie la valeur booléen FALSE
3. Un **seul paramètre** peut-être renvoyé par return mais vous pouvez utiliser une matrice return array(1,2,3), à condition que le nombre de valeurs soit défini (fixe).

3. Fonction conditionnelle IF

Comme beaucoup d'autres langages de programmation, PHP utilise la fonction IF pour les opérations conditionnelles. Sa structure:

```
<?php
if (condition1) {
// fonction 1
}elseif (condition2) {
//fonction 2
}elseif (condition3) {
//fonction 3
}else {
//fonction 4
}
?>
```

Si la condition 1 est vraie, le programme exécute la fonction 1, sinon, elle vérifie la condition 2. Si cette condition est remplie, elle exécute la fonction 2. Sinon, elle vérifie la condition 3. Si la condition 3 est remplie elle exécute la fonction 3. Sinon, c'est la fonction 4 qui est exécutée.

Le tableau suivant reprend les comparaisons possibles.

condition	exemple	nom	remarque
= =	$valeur1==$valeur2	égal	vrai si $valeur1 est égal à $valeur2

= = =	$valeur1===$valeur2	identique	vrai si $valeur1 est égal à $valeur2 et s'ils sont de même type (à partir de PHP version 4)
<>	$valeur1<>$valeur2	différent	vrai si $valeur1 est différente de la $valeur2
<	$valeur1<$valeur2	inférieur	vrai si $valeur1 est inférieure à $valeur2
>	$valeur1>$valeur2	supérieur	vrai si $valeur1 est supérieure à $valeur2
<=	$valeur1<=$valeur2	inférieur ou égal	vrai si $valeur1 est inférieure ou égale à $valeur2
>=	$valeur1>=$valeur2	supérieur ou égal	vrai si $valeur1 est supérieure ou égale à $valeur2

Le sigle égal est réservé pour introduire des données dans une variable. Dans les versions inférieures à PHP 4, "1" == 1 renvoie la valeur vraie (même s'ils sont de types différents).

4. La fonction While

Cette fonction permet de faire une boucle tant que la condition est vraie. Attention de mettre une valeur qui varie dans le temps (par exemple un compteur qui s'incrémente à chaque boucle), sinon, le programme tourne sur lui même à l'infini (en boucle).

```
While(condition)
{
 // a effectuer
}
```

En PHP, la fonction While peut par exemple afficher les nombre de 0 à 9 par le script:

```
$i=0;
while($i<10){
print($i."<br>");
$i=$i+1;
}
```

5. Opérateurs logiques

Comme dans la majorité des langages de programmations, vous pouvez utiliser les fonctions Et et Ou dans les fonctions conditionnelles Si:

Comparateur	exemple	Nom	description
AND	$valeur1 AND $valeur2	ET	Vrai si toutes les conditions sont exactes
OR	$valeur1 OR $valeur2	Ou	Vrai si une des condition est exacte
XOR	$valeur1 XOR $valeur2	Ou exclusif	Vrai si une des condition est exacte, mais pas toutes en même temps
!	! $valeur1	Non	Vrai si $valeur1 est fausse
&&	$valeur1 && $valeur2	ET	Vrai si toutes les conditions sont exactes
||	$valeur1 || $valeur2	Ou	Vrai si une des condition est exacte

6. La commande Switch

La commande Switch permet de définir une liste de commande en fonction d'une variable.

```
switch($variable){
 case valeur1:
   commmande 1
   break;
case valeur2:
   commande 2
   break;
default:
   commande avec valeur
   break;
}
```

- Pour chaque possibilité, on utilise CASE suivi de la valeur. La liste de commande se termine par BREAK;.
- La ligne DEFAULT (optionnelle) permet de définir une liste de commande si aucune autre condition n'est remplie. Si vous omettez cette option et que la valeur de la variable $variable n'est pas reprise, aucune commande n'est exécutée, aucun message d'erreur n'est non plus affiché.

Exemple:

```
<?php
$variable=1;
switch($variable){
```

```
case 1:
 echo '1';
 break;
case 2:
 echo'2';
 break;
default:
 echo'autre valeur';
 break;
}
?>
```

Ce petit programme affiche 1 si $variable vaut 1, 2 si la variable est 2 et autre dans tous les autres cas.

7. La commande Isset()

La commande ISSET() est régulièrement utilisée dans les fonctions conditionnelles, elle vérifie si une variable est définie et renvoie true (Vrai). Dans le cas contraire, la fonction renvoie False (faux).

Prenons un exemple:

```
<?PHP
$a="valeur";
print(isset($a));
print(" - la valeur a est bien définie<br>");
print(isset($b));
?>
```

Affichera "1 - La valeur a est bien définie". Remarquez que la fonction ISSET($b) n'est pas affichée. Nous pouvons modifier nos lignes de commandes comme:

```
<?php
If (isset($b)==FALSE){
print ("Valeur PHP b non définie");
}
?>
ou
<?php
if (!isse($b)
{
print ("variable PHP b non définie");
}
```

Nous avons inséré la commande print("
"); qui envoie un saut de ligne et utilisé une ligne de commande Si pour afficher "Valeur PHP b non définie". Remarquez que pour tester

l'égalité, nous utilisons ==. Le **signe =** est utilisé uniquement pour **associer une valeur à une variable**. La deuxième méthode utilise la négation ! et donne un résultat identique.

8. Informations sur le visiteur

PHP inclut en standard quelques fonctions permettant de mieux connaître les visiteurs d'un site. Elles sont automatiquement récupérées par PHP.

Les données sont intégrées dans un tableau $_SERVER["ligne"]="valeur" ($HTTP_SERVER_VARS["ligne"]= "Valeur" dans les anciennes versions PH3, et PH4). Les valeurs possibles sont:

Fonction	description	exemple de valeur
$_SERVER["REMOTE_ADDR"]	adresse IP de l'internaute	62.32.56.269
$_SERVER["HTTP_USER_AGENT"]	Type de navigateur utilisé	• Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.0; .NET CLR 1.1.4322): **Windows 2000 sous Internet explorer 6.0** • Mozilla/5.0 (Windows; U; Windows NT 5.0; fr; rv:1.8.0.1) Gecko/20060111 Firefox/1.5.0.1: **Win 2000 sous Firefox1.5**
$_SERVER["SCRIPT_FILENAME"]	le nom du fichier PHP en cours	index.php
$_SERVER["SCRIPT_URl"]	L'adresse complète de la page précédente sans le nom de domaine	/index.php
$_SERVER["REQUEST_METHOD"]	procédure pour obtenir la page	Par défaut GET, sauf pour les formulaires ou l'on peut également utiliser la fonction POST
$_SERVER["HTTP_REFERER"]	nom complet de la page Internet précédente	L'adresse complète de la page qui fait un lien (la valeur est vide si le navigateur n'a pas suivi de lien). http://www.ybet.be/vente/index.php
$_SERVER["HTTP_HOST"]	le nom de domaine qui envoie le lien	http://www.ybet.be

L'affichage de ces valeurs passent par les fonctions en PHP Print() ou Echo.

Exemple d'utilisation:

```
<?PHP print('Nous sommes le ');
print(date('d/m/Y')); print('. Il est actuellement '); print(date('H:i:s')); print('.');
Print(' Ton adresse Internet IP ');
Print($_SERVER['REMOTE_ADDR']);
print(" ");
Print($_SERVER['HTTP_USER_AGENT']);
print(" ");
print($_SERVER['SCRIPT_URI']); print(' ');
print("Ta page d'arrivée est: ");
print($_SERVER['HTTP_REFERER']);
?>
```

Nous sommes le 12/02/2010. Il est actuellement 00:45:59 Ton adresse Internet IP 77.195.64.121 Mozilla/5.0 (Windows; U; Windows NT 6.0; fr; rv:1.9.2) Gecko/20100115 Firefox/3.6 (.NET CLR 3.5.30729)
Ta page d'arrivée est: http://www.ybet.be/internet14/php-3.php

9. Exercice

1. Affichez les nombres de 1 à 10 dans un tableau de 2 colonnes (1 seule ligne) les 5 premiers dans la première cellule, les 5 suivants dans la seconde.
2. Déclarez une variable $texte="exercice PHP". Faites une fonction conditionnelle qui affiche "Félicitation, vous avez réussi le test" si $texte="exercice PHP", qui affiche 'l'exercice est difficile' si la variable $texte est différent de exercice PHP. Essayez en donnant une autre valeur à $texte.

14.5. Variables tableaux en PHP

Comme dans tous les langages de programmation, un tableau (matrice) permet d'insérer des valeurs dans des cases. Ceci facilité le rassemblement de données, surtout que de nombreuses fonctions en PHP permettent le tri et le filtre sur les données incluses dans ce type de variables. Dans un chapitre suivant, nous les remplaceront par une base de donnée Mysql, mais les tableaux sont souvent utilisées pour de faibles quantités d'informations, surtout que le nombre de bases de données possibles sur un hébergement mutualisé est souvent limité à 1 base, en plus d'une limitation éventuelle de taille.

2. Initialisation d'un tableau

Contrairement aux autres variables de PHP, vous devez initialiser les tableaux avant de les utiliser. Cette fonction se fait par:

```
<?php
```

```
// $tableau est la variable de type matrice

$tableau = array();

?>
```

Remarquez que nous n'avons pas donné le nombre de colonnes ou de lignes. Un élément du tableau peut être de tout type, y compris un tableau en utilisant différents niveaux d'accolades. Prenons quelques exemple:

3. Utilisation d'un tableau

$tableau[1] est l'élément 1 du tableau.

$tableau["a"] est l'élément a du tableau.

$tableau["jour-du-mois"] est l'élément jour-du-mois du tableau.

Si l'opérante est une chaîne de texte, elle doit être entre guillemets.

Nous allons par exemple gérer un tableau reprenant le type d'appareil et sont numéro de série.

```
<?php
$appareil = array();
$appareil [1] [1] = "PC bureau";
$appareil [1]["serie"] = "ABV25620";
if (!isset($appareil[1][1])){
return FALSE;
}else{
print($appareil[1][1]);
print("<br>");
print($appareil[1]["serie"]);
}
?>
```

La première partie définit les valeurs du tableau.

La fonction conditionnelle si vérifie si les coordonnées de la variable 1 existe (**!**isset() est la négation). Si la cellule n'existe pas, il renvoie la valeur FALSE, sinon, il affiche les valeurs.

Voici le résultat:

```
PC bureau
ABV25620
```

La variable $_SERVER vue au chapitre précédant est également un tableau.

Cette partie sera utilisée notamment pour récupérer des valeurs d'une table Mysql

4. Nombres de valeur d'un tableau

Pour compter le nombre de cellules d'un tableau (**en fait le nombre de lignes**), nous pouvons utiliser la fonction **count()**

```
$tableau=array();
$tableau[1]="valeur 1";
$tableau[2]="valeur 2";
$tableau[2]['tests']="valeur 2";
print(count($tableau));
```

Affichera le nombre 2.

Une remarque, si votre tableau est à 2 dimensions, cette commande affiche uniquement le nombre de lignes.

14.6. Fonctions sur les variables en PHP

Cette partie va nous permettre d'étudier quelques fonctions PHP principales spécifiques au traitement des chaînes de caractères et nombres.

2. Fonction STRLEN()

La **fonction PHP STRLEN()** renvoie la longueur de la chaîne de caractères.

```
<?PHP
$a=strlen("erer");
print($a);
?>
```

affiche le chiffre 4, le nombre de caractères de la chaîne erer.

3. Fonctions Trim, Rtrim, ltrim

Ces fonctions permettre de supprimer les blancs dans une chaîne de caractères.

Trim($variable) efface les blancs devant et derrière dans la chaîne de caractères, pas les espaces au milieu.

RTRIM($variable) supprime les blancs à droite de la chaîne de caractères. A partir de PHP 4.0.1, une opérante optionnelle a été rajoutée: **rtrim($text,caractères à supprimer)**, par exemple: rtrim("Bonjour","jour") donne "Bon".

LTRIM($variable) supprime les blancs à gauche de la chaîne de caractères. A partir de PHP 4.0.1, une opérante optionnelle a été rajoutée: **ltrim($text,caractères à supprimer)**, par exemple: ltrim("Bonjour","bon") donne "jour".

4. Minuscules - majuscules

- **STRTOLOWER($variable)** transforme tous les caractères de la chaîne en minuscule
- **STRTOUPPER($variable)** transforme tous les caractères en majuscule
- **UCWORDS($variable)** transforme la première lettre chaque mot en majuscule
- **UCFIRST($chaîne-caractère)** transforme la première lettre de la chaîne en majuscule

5. Manipulation et modification de chaînes

Ces fonctions PHP vont modifier le contenu d'une chaîne de caractères.

ADDSLASHES($variable): ajoute les anti-slashes \ devant les caractères spéciaux. Cette fonction est utilisée pour les fonctions Print("") et ECHO"" et lors d'introduction de valeurs via un formulaire. exemple: **ADDSLASHES("L'entreprise")** donne "L\'entreprise".

STRIPSLASHES($variable): supprime les anti-slashes, notamment utilisée lors de version imprimable.

CHUNK_SPLIT($variable,nb caractères, caractère de séparation): permet de scinder une chaîne de caractère. Exemple: CHUNK_SPLIT($bonbon,"3","-") donne bon-bon-

STRSTR($variable,"caractère") recherche le caractère et affiche le reste de la chaîne, y compris le caractère. Exemple: STRSTR("YBET informatique","i") affiche "informatique".

STR_replace("caractère à remplacer","caractère de remplacement",$variable) remplace dans la chaîne $variable les caractères à replacer par le caractère de remplacement et l'assigne à une variable. Cette fonction PHP tient compte de la casse.

Exemple: $texte=STR_REPLACE("i","y'","ibet"). Print($texte); donne ybet.

SUBSTR (chaine, numero_depart [,longueur]) récupère longueur caractères sans la chaîne à partir du numero de départ.

Exemple: Il n'y a pas de fonction **left** en PHP. Pour récupérer les 5 premiers caractères d'une chaîne $chaine, la commande est SUBSTR ($chaine,0,5);

HTMlentities($variable) remplace le caractère par son équivalent HTML si possible. Exemple: HTMlentities(" ") affiche

EREG($variable1,$variable2) recherche si la chaîne $variable1 est contenue dans $variable2, renvoie une valeur logique. Exemple, vérification du pays suivant le numéro de TVA

```
<?php
if (ereg("BE","BE718409912")){
echo"Belgique";}
else{
Echo"Hors Belgique";
}
?>
```

Explode($caractere,$chaine[,int limit]) coupe la variable texte $chaine en deux en supprimant le caractère de séparation $caractere. Le résultat est renvoyé sous forme de tableau. L'option limit (à partir de la version 4.0.1, une valeur entière) permet de définir le nombre maximum de lignes du tableau renvoyée par la fonction. La dernière ligne contient alors le reste de la chaîne.

```
<?php
$tableau=array[];
$chaîne="Ma maman est invitée";
$tableau=Explode("a",$chaine,2);
print($tableau[0]);
print($tableau[1]);
?>
```

STRPOS($variable,$variable1): revoie le nombre de caractère devant la lettre $variable1. Exemple: STRPOS ("YBET informatique,"i") renvoie le nombre 5. Si le résultat est 0, le caractère n'est pas repris dans la chaîne.

nl2br($chaîne) affiche les sauts de lignes en remplaçant les /n par
. Nous en reparlerons

6. Spécifiques nombres

DECHEX($valeur): renvoie la valeur hexadécimale d'un nombre.

CEIL($valeur): renvoie le nombre entier supérieur.

FLOOR($valeur): renvoie le nombre entier inférieur

Round($valeur,$nb): renvoie l'arrondit de $valeur avec $nb chiffres derrière la virgule. Si $nb est omis, il est considéré comme 0 chiffres derrière la virgule.

is_int() renvoie true si le contenu est un entier, false sinon.

intval() convertit une chaîne en variable entière.

rand(min,max): ou min est la valeur minimale, max, la valeur maximale. Cette fonction renvoie une variable entière.

Exemple: $i=rand(0,5) $i sera compris entre 0 et 5 (6 valeurs possibles). Quelques hébergements limitent la valeur maximum à 32768. A défaut des paramètres RAND() renvoie un nombre compris entre 0 et le nombre maximum.

Nous verrons d'autres fonctions PHP dans les chapitres suivants mais ceux-ci sont les principaux.

7. Fonctions sur les dates

checkdate (int mois, int jour, int année): vérifie si une date est valide. mois doit être compris entre 1 et 12, jour entre 1 et 31 et année entre 1 et 32767. La fonction tient compte des années bissextiles. Remarquez que le format est anglophone. Le résultat est une valeur logique.

mktime ([int heure [, int minute [, int seconde [, int mois [, int jour [, int année [, int is_dst]]]]]]]) renvoie une valeur entière correspondant à la différence par rapport au 1er Janvier 1970 00:00:00 GMT en secondes. On appelle cette méthode le Timestamp. Cette commande permet de calculer des différences de dates. Pour le résultat inverse, utilisez la fonction **date (format,int timestamp).**

Exemple: mktime(0,0,0,12,1,2006).

14.8. Créer une base de donnée MySQL

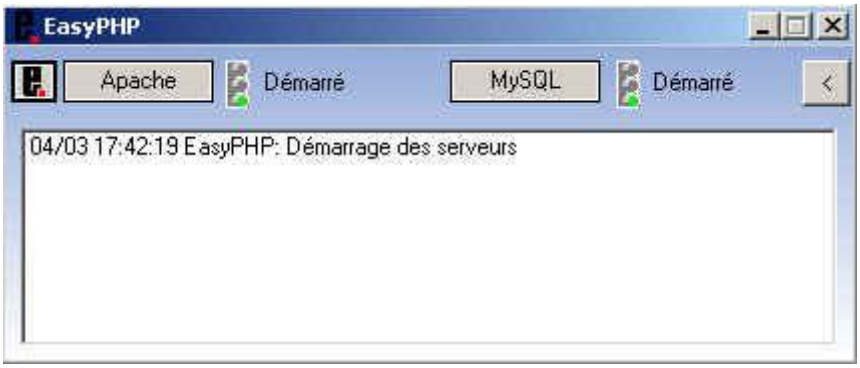
1 Introduction - 2. Démarrer EasyPHP. - 3. Création d'une table - 4. Types de champs

La base de donnée Mysql, couplée à une base de donnée PHP permet de créer quasiment tous les sites Internet inimaginables. Mysql a de nombreux avantages, elle est sous licence GPL (donc libre d'utilisation), son nombre de développeurs est élevé (aide facile sur les forums informatiques), relationnelle, ... En plus, de nombreuses applications sont téléchargeables gratuitement sur Internet: forum, livre d'or, gestion photos, vente en ligne, ... (mais merci de respecter les règles d'utilisation - un lien ne coûte ... rien)

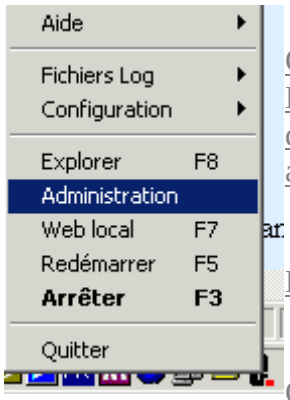
Cette partie de la formation création de site va nous permettre de comprendre le fonctionnement de MySQL, couplé au PHP. A partir de ce chapitre, nous allons commencer à développer une véritable application comme exercice: **un petit site d'annonces en ligne.**

Si le prochain chapitre va permettre de créer des tables Mysql par programmation, nous allons débuter par **créer la base de donnée sous EasyPHP en local**. Pourquoi choisir cette méthode? Tout simplement pour pouvoir tester l'application avant de la transférer sur un site "en production". D'ailleurs, cette formation sera principalement "en local".

Pour une introduction aux bases de données, vous pouvez utiliser la première partie du cours Access: les notions sur les tables, champs sont équivalentes.



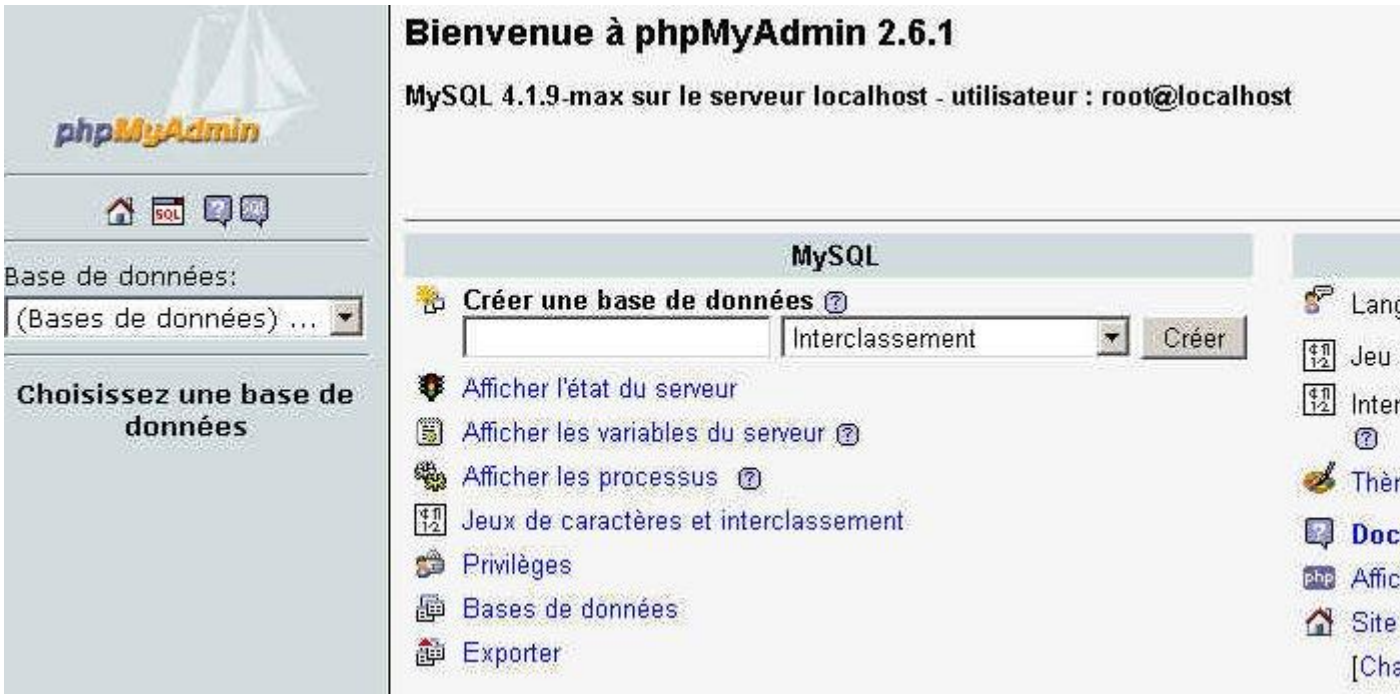
2. Démarrer EasyPhp.



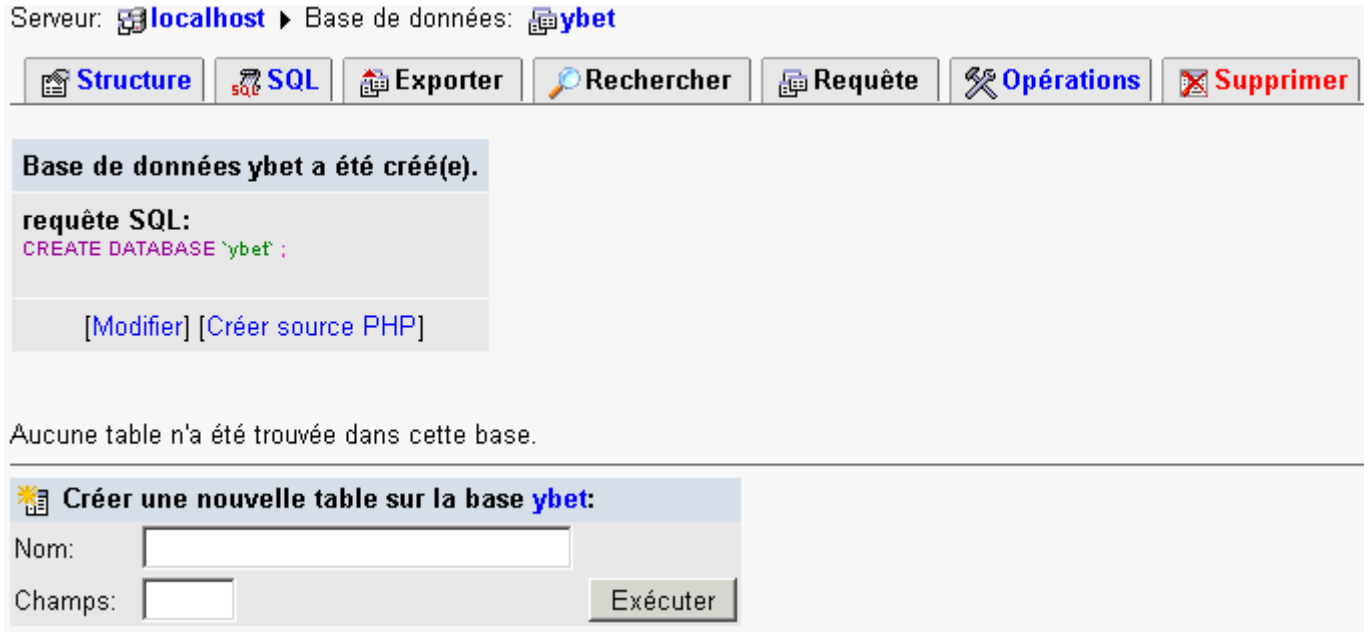
Si vous ne l'avez pas encore installer, télécharger le logiciel GNU/GPL sur le site. Dans Démarrer ->Programmes, sélectionnez EasyPhp. Une petite icône apparaît à coté de l'horloge signalant le démarrage du serveur PHP-mysql. Une petite fenêtre doit également apparaître vous signalant le démarrage des serveurs.

Cette fenêtre permet de démarrer ou d'arrêter Apache (le serveur PHP) et MySQL (la base de donnée) à l'aide des boutons.

A l'aide du menu contextuel de l'icône, sélectionnez administration. Ceci va nous afficher la fenêtre d'EasyPHP. Dans la fenêtre, en PHPMYADMIN, sélectionnez Gestion BD. La fenêtre qui va vous permettre de créer, gérer, ... les bases de données en mode local. La fenêtre ressemble à ceci. La partie de gauche permet d'administrer les bases de données existantes sur votre PC, celle de droite de créer une nouvelle base de donnée ou de gérer celles existantes si elles sont sélectionnées à gauche.



Commençons par créer une base de donnée que nous appellerons YBET par exemple (laisser le type en Interclassement par défaut, normalement, cette possibilité sélectionne Latin1_swedish_ci, le code de caractère par défaut des européens).



Cette partie va nous permettre de créer ensuite des tables dans la base de donnée MySQL. La création peut également se faire par programme (durant l'installation de l'application par exemple).

3. Création d'une table

Les tables en PHP sont identiques à celle d'une table d'Access. Elle intègrent les informations brutes, non traitées. Rentrez le nom d'une table ("annonce par exemple) et le nombre de champs (rubriques) souhaités, à ce stade, sélectionnez 2.

La fenêtre suivante va permettre de personnaliser nos champs. Vous devez obligatoirement rentrer toutes les données

Champ	Type	Taille/Valeurs*	Interclassement	Attributs	N
	VARCHAR				not
	VARCHAR				not

- Le **nom du champ** ne peut être nul, ni comporter de caractères accentués, espaces, -, ... par contre, il accepte _
- Le type de champ est sélectionné dans une liste de choix. Pour l'instant, sélectionnons VARCHAR qui correspond à du texte.
- la **Taille/valeurs** nous permet de déterminer la taille maximum des données à rentrer dans les champs.
- **Interclassement** représente le jeux de caractères par défaut si vous ne sélectionnez rien (préférable). Sinon vous pouvez sélectionner Latin1_swedish_ci. Si le jeu de caractère n'est pas démarré dans PHPMYADMIN, vous recevrez néanmoins un code d'erreur
- L'**attribut** peut-être rien, unsigned ou Unsigned zerofill. Les 2 dernières valeurs s'utilisent uniquement pour des types de champs spéciaux.
- **Null** permet de définir si le champ peut-être null (vide) ou non.
- **Défaut** permet de donner une valeur par défaut au champ.
-
- permet de déterminer si le champ est ou non une **clé primaire**. Dans ce cas, la valeur du champ sera unique dans la table. Aucun n'enregistrement n'aura la même valeur dans ce champ, les tris sont également plus rapides
- **index** permet d'augmenter les vitesses de traitement (tri) dans la table. La clé primaire est également un index.
- unique permet de spécifier que le contenu est unique dans l'ensemble du champ. Contrairement à l'index, elle n'augmente pas la vitesse de traitement.
- permettent d'indexer des textes complets. Cette possibilité ralentit la table pour de longs champs.

Pour l'instant nous allons créer 2 champs de type VARCHAR (caractères de taille variables)

- code, de taille 10 et non null. Unique
- titre, de taille 30 et non null.

N'oubliez pas de sauvegarder la table. La fenêtre suivante donne ceci

	Champ	Type	Interclassement	Attributs	Null	Défaut	Extra	Action
☐	code	varchar(10)	latin1_swedish_ci		Non			
☐	TITRE	varchar(30)	latin1_swedish_ci		Non			

- permet d'éditer (modifier) la table.

- Le bouton  permet de supprimer le champ.

4. Les types de champs

Nous venons de créer 2 champs. Dans notre exemple, nous avons choisi le type VARCHAR (type caractère à taille variable). Cette formation va nous permettre d'étudier les autres types sous MYSQL.

4.1. Les champs de types caractères

CHAR	Chaîne de caractères fixe, nombre de caractères obligatoires. Les caractères non inclus seront remplacés par des espaces.	255 char. max
VARCHAR	Chaîne de caractères de longueur variable	255 char. max
TINYBLOB	Petite zone de texte, sensible à la casse	255 char. max
TINYTEXT	Petite zone de texte, insensible à la casse	
BLOB	Zone de texte standard (mais accepte toute sortes de données comme des images), sensibles aux majuscules / minuscules	65K char. max
TEXT	Zone de texte standard (mais accepte toute sortes de données comme des images), insensible à la casse	
MEDIUMBLOB	Zone de texte moyenne, différentie majuscule /minuscules	16 millions ch
MEDIUMTEXT	Zone de texte moyenne, insensible à la casse	16 millions ch
LONGBLOB	Grande zone de texte, différentie majuscule / minuscule	4 milliards ch
LONGTEXT	Grande zone de texte, ne différentie pas majuscule / minuscule	4 milliards ch
ENUM	Liste de choix	65535 valeurs
SET	Liste de choix multiple	64 valeurs ma

Les types CHAR et CHARVAR ne sont pas sensibles aux majuscules et minuscules: "Table"="table". Pour rendre ces données sensibles à la casse, utilisez l'option [Binary] lors de la création du champs.

Les variables type TEXTE sont insensibles à la casse, les variables type BLOG les différentie.

4.2. Les champs numériques

Champs numériques

TINYINT	Entier très petit	256 valeurs différentes, de -128 à +127 ou 0 à 255 (1 octet)
SMALLINT	Entier petit	65.536 valeurs différentes (2 octets)
MEDIUMINT	Entier moyen	16.777.216 valeurs différentes (3 octets)
INT	Entier standard	4.294.967.296 valeurs différentes (4 octets)
BIGINT	Entier grand	8 octets
FLOAT	Décimal de simple précision	4 octets
DOUBLE, REAL	Décimal de double précision	8 octets
DECIMAL, NUMERIC	Décimal sous forme de chaîne	variable

Pour les champs de type entier, vous pouvez utiliser l'option **Signed** (par défaut) pour déterminer des valeurs négatives ou positives ou Unsigned (uniquement des valeurs positives).

L'option ZEROFILL assure l'affichage des zéro.

4.3. Dates et heures

DATE	Date. Par exemple 2008-03-06, format anglophone	3 octets
TIME	Heure. Par exemple 10:25:59	3 octets
DATETIME	Date et heure. Par exemple 2006-03-06 10:25:59	8 octets
TIMESTAMP	Date et heure sans les séparations, par exemple 20060306102559. Attention, le format a évolué avec les versions d'où un risque d'erreur en changeant d'hébergement Internet.	4 octets
YEAR	Année, exemple 2008	1 octet

14.9. Gérer la base de donnée MySQL

Dans le précédant chapitre, nous avons créé une base de donnée MYSQL et une table manuellement avec PHPMyADMIN. Nous aurions également pu rentrer des données ou

modifier des données existantes mais le but de cette formation est de gérer des bases de données MySql à l'aide de programmes écrits dans le langage PHP. Cette partie va reprendre les commandes nécessaires. L'ensemble des parties suivantes de ce cours nécessitent le démarrage de EasyPhp.

2. Principe.

Pour utiliser une base de donnée écrite en MySQL, il faut:

1. **Connexion au serveur**

Création d'une ressource
Authentification
Sélection de la base de donnée

2. **Utilisation des tables MySQL (via PHP)**

Lecture
Ecriture
Modification

3. **Déconnexion du serveur**

> Toutes les fonctions PHP pour gérer une base ou une table MySQL commencent par **mysql_**.

La **connexion au serveur** se fait par la commande: **mysql_connect($serveur,$login,$mot-passe)**. Dans le cas d'une base de donnée créée avec EasyPhp, par défaut, le serveur est **localhost**, le **login** est root, **mot de passe** vide. Pour une base de donnée hébergée sur un serveur Internet, ces paramètres sont fournis par votre hébergeur.

La fonction mysql_connect() renvoie une ressource si la connexion a réussie, sinon, elle renvoie le message d'erreur False. Une fois que la ressource est établie, elle est sauvegardée et utilisable par les autres fonctions utilisant la base de donnée. Par défaut, le script PHP utilise la dernière connexion ouverte. Cette commande ne doit donc être utilisée qu'une seule fois.

Pour **fermer la connexion**, la commande à utiliser est **mysql_close**.

Exemple de connexion:

```
<?php
if(!mysql_connect('localhost','root')){
 Echo 'Connexion Impossible';
 exit();
} else{
 Echo 'Connexion réussie';
}
// instructions et requêtes sur la base de donnée
mysql_close();
?>
```

Dans le script ci-dessus, le programme PHP affichera Connexion impossible si la connexion à la base de donnée échoue. Si elle est établie, PHP affichera "Connexion réussie". La fonction Exit() (ou son équivalente **die("Message")**) permet d'arrêter le script si la connexion n'est pas réussie.

L'étape suivante va sélectionner la base de donnée que nous avons créé.

Mysql_select_db('ybet') sélectionne la base de données (DB) YBET que nous avons créé au chapitre précédant manuellement. Les tables sont elles sélectionnées dans la requête.

3. Commandes MySQL

C'est maintenant que ça va se compliquer. MySQL est une véritable base de donnée qui travaille par requêtes (interrogations des tables). PHP ne va servir que comme application client pour envoyer ces requêtes. La première partie est donc de créer les requêtes en SQL et de les transférer ensuite par PHP. En gros, on va créer une variable en PHP qui reprend la ligne de commande. Comme le but de ce tutorial n'est pas de créer des applications SQL, mais bien d'utiliser les bases de données via une interface PHP, la suite explique quelque commandes utiles que nous transférerons par PHP ensuite

3.1 Requête d'insertion

Il y a plusieurs méthodes de rentrer des données (d'insérer). Toutes utilisent la **commande INSERT nom_table**. Différentes options sont également proposées.

1. Une liste de colonnes et une liste de valeurs : Insert nom_table(nom colonne 1,nom colonne 2) Values(valeur colonne 1,valeur colonne 2)

Dans notre cas, la table MySQL s'appelle annonce avec comme nom de champs code et titre (remarque: SQL ne fait pas de distinctions majuscules / minuscules dans les nom des champs. La commande SQL est par exemple: Insert annonce(code,TITRE) Values("users0001","Maison à Vendre Pin").

2. Une liste de valeurs noms - valeurs: INSERT nom_table SET nom colonne1=valeur1, nom colonne 2=valeur 2, ...

INSERT annonce SET code='users0003',titre='magasin informatique YBET'

3.2. Sélectionner la base de donnée.

Use nom_base_ donnée permet de sélectionner une base de donnée par défaut. Sans cette commande, vous pouvez dans certaines fonctions nommer la table comme: base.table

3.3. Sélection d'enregistrements

La commande SELECT permet de sélectionner des enregistrements dans une table. Sa syntaxe complète est

SELECT [distinct] liste_colonne FROM table

[WHERE definition]

[ORDER BY [ASC | DESC], ...]

[Limit [offset,] lignes]

La version la plus simple: **Select Liste_colonne FROM table** permet de sélectionner l'ensemble des lignes d'une table suivant la liste des colonnes si vous avez utilisé la commande USE au préalable. Dans le cas contraire, utilisez la commande **Select Liste_colonne FROM base.table**

Pour sélectionner toutes les colonnes, utilisez la commande **SELECT * From TABLE**

L'option Where permet de filtrer les enregistrements.

Select * FROM table Where Titre='YBET' sélectionne tous les enregistrements ou le champ titre est égal à YBET dans la table "table".

Vous pouvez utiliser les opérateurs de comparaison suivants pour la condition.

!	Fonction logique Non, inverse
||	Fonction logique OU (OR)
&&	Fonction logique ET, toutes les conditions doivent être remplies (AND)
=	égal
<> ou !=	Différent
<=	Inférieur ou égal
>=	Supérieur ou égal
>	Supérieur
<	Inférieur
Like	permet de prendre le mot à l'intérieur de la chaîne. % remplace n'importe quel nombre de caractères, y compris 0. _ remplace 1 seul caractère.

ORDER BY permet de trier les résultats de la requête MySQL.

Select * FROM table ORDER BY code récupère tous les enregistrements de la table et les trie suivant le code. Par défaut, le tri est par ordre croissant (ou avec l'option ASC). Pour trier par ordre décroissant, utiliser ORDER BY DESC

LIMIT permet de limiter le nombre de lignes renvoyées, par exemple LIMIT 50 limite aux 50 premiers résultats. Vous pouvez également utiliser 2 arguments, par exemple LIMIT 2,20. Le 2 spécifie de ne pas prendre les 2 premières lignes, mais les 20 suivantes.

3.4. Modification d'enregistrements.

La commande SQL Update permet de modifier le contenu d'un enregistrement. La syntaxe complète est:

UPDATE nom_table SET nom_colonne1=contenu1, [nom_colonne2=contenu2, ...]

[WHERE conditions]

[LIMIT]

Les options Where et Limit sont équivalentes à celles ci-dessus. Vous pouvez également incrémenter directement une valeur.

Par exemple: UPDATE annonce SET code, code+1 incrémente le code de 1

UPDATE annonce SET code, code*2 multiplie le code par 2

La commande UPDATE renvoie automatiquement le nombre de colonnes qui ont été modifiées.

3.5. Effacement d'enregistrements

Cette fonction se fait par la commande DELETE.

La commande complète est de type: **DELETE FROM table [WHERE condition] [LIMIT]**

4. Liaison MySQL - PHP

L'envoie de commandes MySQL par un programme php se fait à l'aide de la commande **mysql_query($requête SQL)**. La première partie est donc l'enregistrement de la requête dans une variable. La deuxième est l'envoi de la variable.

En cas d'erreur, la fonction mysql_query($requête) renvoie la valeur FALSE. Pour récupérer le message d'erreur, on utilise la commande SQL mysql_error(). Ceci va nous amener à modifier immédiatement notre scritp PHP.

4.1. Sélection de la base de donnée

Pour pouvoir accéder aux données d'une table, vous devez sélectionner la base de donnée au préalable par la commande:

Mysql_select_db(nom_base);

4.2. Insertion de données dans une table

Nous allons adapter les requêtes SQL sous forme d'une valeur pour la transmettre comme variable à la base de donnée. Les données à insérer doivent être entourées de **simples guillemets**, y compris les variables PHP. L'utilisation de variables sous formes de tableaux est déconseillée.

```
<?php
if(!mysql_connect('localhost','root')){
Echo'Connection Impossible';
exit();
} else{
Echo'Connexion réussie';
}
Mysql_select_db('ybet');
$requete="Insert annonce(code,TITRE) Values('users0002','Maison à Vendre Florenville')";
$valeur=mysql_query($requete);
$erreur=mysql_error();
print($erreur);
?>
```

Analysons ces lignes de commandes. La première partie vue plus haut nous permet de nous connecter à la base de donnée YBET.

Nous allouons ensuite à la variable $requete la valeur de la requête SQL d'insertion qui transmet la valeur users0002 au champ code et la valeur Maison à vendre Florenville au champ TITRE.

La commande suivante va transférer la requête via la commande mysql_query et retourner une valeur à la variable PHP $erreur (True ou False).

Dans le cas d'erreur, la fonction mysql_error() renvoie le message d'erreur que nous affichons à l'écran par la commande Print, la variable $valeur ne renvoie rien (false, comme la requête n'a pas été exécutée). Dans le cas où la requête s'est bien passée, la variable $valeur renvoit 1 (True) et $erreur, rien. Nous n'affichons donc que $erreur, la seule intéressante.

Dans le cas où nous utilisons la commande INSERT table SET colonne1=valeur1, colonne2=valeur2, le script PHP est presque identique. La fonction SET est insérée dans la même ligne.

```
<?php
if(!mysql_connect('localhost','root')){
Echo'Connection Impossible';
exit();
} else{
Echo'Connexion réussie';
}
Mysql_select_db('ybet');
$requete="INSERT annonce SET code='users0004',TITRE='magasin informatique YBET'";
$valeur=mysql_query($requete);
$erreur=mysql_error();
print($erreur);
?>
```

4.3. Recherches d'enregistrements

La requête SQL associée est **SELECT FROM**. Dans le cas de la fonction mysql_query(), la réponse est la valeur trouvée. Reprenons les enregistrements insérés ci-dessus et affichons les enregistrements

```
<?php
...
$requete="SELECT * FROM annonce";
$valeur=mysql_query($requete);
print($valeur);
...
mysql_close()
?>
```

N'affichera rien, tout simplement parce que notre table contient effectivement des valeurs mais nous demandons d'afficher en fait 2 variables (puisque la table contient les champs code ET titre). $valeur, le résultat de la commande mysql_query($requete) est une matrice (une variable tableau).

La commande utilisée est finalement **mysql_fetch_array($valeur)** qui va créer un table reprenant chaque fois les noms des colonnes. Lorsqu'il ne reste plus de lignes à lire, mysql_fetch_array() retourne FALSE. Le script PHP devient:

```
<?php
// ouverture de la base de donnée

$requete="SELECT * FROM annonce WHERE code='users0004'";
$valeur=mysql_query($requete);
print(mysql_error());
$tableau=mysql_fetch_array($valeur);
print($tableau['code']);
print($tableau['TITRE']);
mysql_close()
```

```
?>
```

Ce script **ne fonctionne que s'il n'y a qu'une seule ligne correspondant à notre filtrage**. La solution passe par la commande **WHILE**. Nous allons faire tourner cette fonction en boucle tant que des données sont transmises.

La commande **mysql_num_rows()** permet de connaître le nombre de lignes affichées (**mysql_affected_rows()**, identique, est utilisée pour les requêtes d'effacement ou de modification d'enregistrement). En utilisant la commande en bouche WHILE:

```
<?php
if(!mysql_connect('localhost','root')){
   Echo'Connection Impossible<br>';
   exit();
} else{
   Echo'Connexion réussie <br>';
}
Mysql_select_db('ybet');
$requete="SELECT * FROM annonce WHERE code='users0004'";
$valeur=mysql_query($requete);
print(mysql_num_rows($valeur).' ligne(s)<br>');
while ($tableau=mysql_fetch_array($valeur)){
   print("<b>".$tableau["code"]."<br>".$tableau["TITRE"]."</b>\n");
}
mysql_close()
?>
```

Voici le résultat dans notre base de donnée actuelle

Connexion réussie
1 ligne(s)
users0004
magasin informatique YBET

14.10. Créer une table dans une base de donnée MySQL avec PHP

1 Introduction - 2. Syntaxe - 3. Exemple de création - 4. Supression d'une table

Dans les 2 chapitres précédents, nous avons créé une base de donnée MYSQL en local et commencer à apprendre les commandes MySQL. Cette partie va nous permettre de créer de manière automatique une base de donnée. Ces commandes sont nécessaires pour une installation automatique, mais aussi parce que tous les hébergeurs Internet n'autorise pas une gestion complète de la base de donnée par une interface propre.

La base de donnée MySQL doit être préalablement créée de manière manuelle sur votre hébergement qui vous donnera les renseignements nécessaires: nom du serveur, le login et le mot de passe d'accès. A la création, cette base de données est vide. Vous pouvez utiliser une seule base de donnée pour des applications différentes le cas échéant, à condition de ne pas insérer 2 tables sous le même nom. Cette solution est à utiliser pour la majorité des hébergements qui n'autorisent qu'une seule base de donnée, même si cette solution ralentit le traitement.

2. Syntaxe

La commande SQL associée est **CREATE TABLE [IF NOT EXIST] nom_table (definition-colonne, Index)**

Si une table du même nom existe dans la base de donnée, un message d'erreur est renvoyé. Si vous utilisez la commande optionnelle IF NOT EXIST, non seulement il n'y a pas de message d'erreur, mais la table n'est pas non plus créée.

la partie définition-colonne reprend:

- nom-colonne type [NOT NULL ou NUL] [DEFAULT valeur-par-defaut)
- [AUTO_INCREMENT]
- [PRIMARY KEY]

Dans Index, on retrouve

- Primary Key (index_nom-colonne)
- INDEX [Nom-index] (index-nom-colonne, ...)
- UNIQUE [INDEX] (index-nom-colonne, ...)

Ces notions ont déjà été vues dans le chapitre "créer une base de donnée en local"

3. Exemple de création

Nous allons créer une table dans notre base de donnée YBET dont le nom est CONTENU. Voici la liste des champs à insérer:

	types de variable	primary-key, Index	remarque	Instruction SQL
code	numérique, integer	Primary-key	auto incrémentation	code int primary key NOT NULL auto_increment
titre	VARCHAR(120)		titre de l'annonce limitée à 120 caractères, insensible à la casse	titre varchar(120) not null

description	TEXT		texte limité à 65535 caractères	description blob not null
photo	varchar(255)		adresse et nom de la photo, 255 caractères	photo varchar(255)
Ville	VARCHAR(40)	index	caractère limité à 40 caractères	ville varchar(40), index(ville)
Pays	type enum		choix entre Belgique, France, Luxembourg	pays enum('Belgique','France','Luxembourg')
prix	decimal(10,2)		décimal avec 2 chiffres derrière la virgule	prix decimal(8,2) not null
dateinsertion	date		date courte	dateinsertion date
telephone	varchar(15)			telephone varchar(15)
mail	varchar(30)			mail varchar(30)

Nous pourrons éventuellement créer des champs supplémentaires ou modifier les attributs des champs par la suite. Dans notre version locale sous EasyPhp, le serveur est localhost, le login est root, sans mot de passe.

```
<?php
if(!mysql_connect('localhost','root')){
Echo'Connection Impossible';
exit();
} else{
   Echo'Connexion réussie';
}
Mysql_select_db('ybet');
// cette partie ouvre la base de donnée
$requete="CREATE TABLE if not exists contenu (code int primary key NOT NULL auto_increment,titre varchar(120) not null,description TEXT not null,photo varchar(255), ville varchar(40),pays enum('Belgique','France','Luxembourg'),prix decimal(8,2) not null,dateinsertion date,telephone varchar(15),mail varchar(30),index(ville))";
$erreur=mysql_query($requete);
$erreur1=mysql_error();
print($erreur."<br>");
print($erreur1);
mysql_close();
?>
```

L'ensemble des commandes ont déjà étés vues dans les chapitres précédents. La difficulté est de créer la requête SQL. Chaque champ est séparé par une virgule.

Dans le cas de l'index, il peut être inséré juste derrière le champs ou à la fin de la requête: Index(ville(10)) indexe seulement sur les 10 premiers caractères du champ ville.

4. Suppression d'une table

Vous pouvez supprimer une table dans une base de donnée existante par la commande SQL: **DROP table [if exists] nom_table (, nom_table1, ...)**

Par exemple, pour supprimer la table contenu ci-dessus, utilisez les lignes de commandes PHP

```
<?php
 if(!mysql_connect('localhost','root')){
  Echo'Connection Impossible';
 exit();
 } else{
    Echo'Connexion réussie';
}
Mysql_select_db('ybet');
 // cette partie ouvre la base de donnée
 $requete="DROP table if exists contenu";
 $valeur=mysql_query($requete);
 $erreur1=mysql_error();
 print($valeur."<br>");
 print($erreur1);
 mysql_close();
?>
```

Il vous reste à copier les codes ci-dessus dans le dossier www de Easyphp et d'exécuter les différentes requêtes pour essayer ces commandes et les adapter à votre propre utilisation. Remarquez que ces commandes sont automatiquement exécutées par le programme d'installation lorsque vous installez un forum, un portal, un livre d'or, site de vente en ligne, ... sous licence GPL. Les commandes DROP sont également utilisables dans la console MYPHPADMIN de votre hébergement Internet.

10.A. Exercice PHP: un petit formulaire de contact

1 Introduction - 2. Le formulaire - 3. Vérification des données entrées - 4. Transfert dans la base de donnée MySQL - 5. Envoi à votre adresse mail - 6. Le développement complet.

Les différentes précédentes parties de cette formation nous permettent maintenant de commencer quelques exercices comme un simple formulaire de contacts en PHP - MySQL

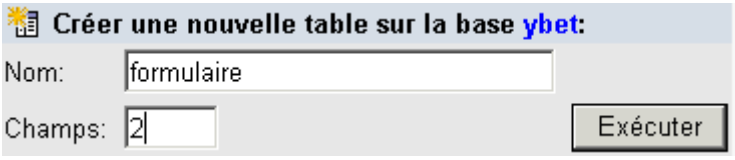
(mises à jours, inscriptions aux news, ...) à insérer sur votre site Internet. Il est volontairement "dépouillé", vous pourrez le compléter.

L'énoncé:

- **Créer un formulaire de contact** reprenant le nom et l'adresse mail.
- **Vérification des données entrées**, un nom et une adresse mail valide. Ce formulaire est auto-invocant.
- Si les données sont correctes, elles doivent être **sauvegardées dans une table MySQL** que nous appellerons formulaire reprenant ces 2 données.
- les données doivent également être envoyées dans votre **boîte mail**.

J'ai volontairement limité le nombre de champs à 2. Il n'y a pas de listes de choix, ... ceci va faciliter la création du formulaire

La table doit être au préalable créée dans EasyPhp (mais la fonction mail ne fonctionne pas) ou directement sur votre hébergement Internet.



Les champs

Nous n'utilisons pas de clés primaires, index, ...

2. Le formulaire

Comme le fichier contact.php doit être auto-invocant, le formulaire, méthode POST, doit être affiché en PHP (mais ce n'est pas obligatoire ici). Pour cela, nous allons attribuer à une variable $form l'ensemble du contenu, sans oublier les \ devant les " et ensuite affichée ce texte.

```
<php

$form="

<form method=\"POST\">
<p>Nom: <input type=\"text\" name=\"nom\" size=\"30\" value=\"Nom\"></p>
<p>Votre adresse mail: <input type=\"text\" name=\"mail\" size=\"30\"
value=\"Votre adresse mail\"></p>
<p><input type=\"submit\" value=\"Envoyer\" name=\"B1\"></p>
</form>";
```

```
echo $form;

?>
```

Remarquez que nous lui donnons une valeur par défaut via l'option value: la zone NOM reprend le texte nom, la zone mail reprend la valeur "Votre adresse mail". Ce n'est pas obligatoire, sauf que les tests ci-dessous seront modifiés.

3. Vérification des valeurs entrées.

La partie ci-dessus s'insère au-dessus du formulaire. Les valeurs entrées par l'utilisateur sont directement modifiées pour supprimer les blancs et insérer le caractère \ devant les guillemets et apostrophes par la fonction addslashes.

Si la valeur envoyée est égale à la valeur par défaut, un message d'erreur s'affiche. Les tests sur l'adresse mail vérifient simplement si elle n'est pas nulle et si **le caractère @ existe**, suivi de caractères.

```
if (isset($_POST['B1']))
{
// teste les valeurs.
$nom=trim(addslashes($_POST['nom']));
$mail=trim(addslashes($_POST['mail']));
$tableau=array();
$tableau=Explode("@",$mail,2);
// echo $tableau[0];
if ($nom=="" || $nom=="Nom")
{
echo "Veuillez rentrer un nom";
}elseif ($mail==""){
echo "Rentrez une adresse mail";
}elseif (!isset($tableau[1])) {
echo"Rentrez une adresse valide";
}else{
// entrée des valeurs dans la table MySQL, envoi des données à une adresse mail.

}
}
```

4. Transfert dans la base de donnée

Le transfert des données vers la base de données ne se fait que si les données sont validées. Rien de bien spécial, nous ne vérifions même pas si l'utilisateur existe déjà (mais vous pouvez le faire comme exercice).

```
if(!mysql_connect('localhost','root')){
Echo'Connection Impossible';
exit();
} else{
// Echo'Connexion réussie';
}
Mysql_select_db('ybet');
$requete="insert formulaire SET nom='$nom', mail='$mail'" ;
$resultat=mysql_query($requete);
```

5. Envoi dans votre boîte mail

La dernière fonction est d'envoyer les données par mail. Elle utilise la fonction PHP **mail(destinataire, sujet, message, headers)**. Cette fonction renvoie true si le message est bien envoyé, false sinon. Les différents composants doivent au préalable être définis dans les variables.

```
// envoi par mail
$date=date('Ymd');
$message=date("d/m/Y H:m:s\n")."\n Nouveau contact: "."\n nom: ".$nom."\n
Adresse mail: ".$mail;
$entetes = "From: contact@site.be\nReply-to: contact@site.be\n";
$sujet="Nouveau contact ".$nom;
$webmaster=('contact@site.be');
mail($webmaster,$sujet,$message,$entetes);
```

Elle vient à la suite de l'entrée des données dans la base de donnée.

Remarques: La fonction mail **ne fonctionne pas** sous Easyphp en local et doit se faire sur un serveur. Les différentes adresses mail doivent être corrigées selon votre propre adresse mail.

6. Le fichier complet

Vous pouvez le télécharger ici.

11. Première partie de notre exercice: la mise en page de notre site de petites annonces.

Dans les précédents chapitres de cette formation PHP - MySQL en ligne, nous avons déjà vu une partie des commandes qui vont nous permettre d'utiliser une base de donnée pilotée par PHP. Avant de commencer le développement de **notre site de petites annonces** (la création de la base de donnée est déjà faite au chapitre précédant), nous allons faire la mise en page, la présentation de notre site et utiliser quelques fonctions spécifiques pour l'utilisation simultanée de plusieurs fichiers PHP. Cette partie va être nécessaire pour tous le développement futur. Nous allons utiliser une structure de page Internet standard:

Header		
Colonne gauche	Corps de la page (contenu)	colonne droite
Footer		

12. Créer la table utilisateur en PHP - MySQL

Pour continuer notre formation, nous allons créer des tables MySql directement par des commandes PHP. Cette partie va nous permettre de créer une table utilisateur. Ces utilisateurs doivent pouvoir créer un profil et se connecter pour créer leurs annonces. Ils doivent également récupérer leur mot de passe, modifier leurs coordonnées, ... A ce stade, nous autorisons toutes les connexions, seule quelques vérifications sont faites au niveau des données rentrées.

La partie "entrée des coordonnées" va être la plus difficile, nous devons:

1. Créer le formulaire pour entrer des données
2. Vérifier la validité des données entrées
3. Permettre de corriger les données si elle sont non valides
4. Vérifier si l'utilisateur n'existe pas (dans cette partie via l'adresse mail)
5. Insérer les données dans la base de donnée

2. Table utilisateur

Dans les précédents chapitres, nous avons créés des tables dans la base de donnée pour les annonces. Comme une base de donnée MySql peut inclure plusieurs tables (MySql est même nativement relationnelle), nous allons créer une nouvelle table reprenant les utilisateurs.

Les champs à inclure dans la table sont:

1. uid (numéro d'utilisateur)
2. titre (Monsieur, Madame, Mademoiselle, ...)
3. nom
4. prenom

5. adresse
6. codepostal
7. ville
8. pays
9. telephone
10. email (adresse mail)
11. username (login)
12. password
13. type

Les 8 premiers permettent de retrouver le visiteur, mais sont dans certains cas facultatifs.

- L'adresse mail est obligatoire
- L'username et le password permettront à l'utilisateur de se connecter
- Le type permettra dans le futur de déterminer des rangs d'utilisateurs comme administrateur ou simple utilisateur.

Pour suivre certaines législations ou personnaliser l'accès au site, nous allons ajouter quelques champs comme la date de naissance et la date d'enregistrement

13. bday (jour de naissance)
14. bmonth (mois de naissance)
15. byear (année de naissance)
16. regdate (date d'enregistrement du membre)

Nous en rajouterons par la suite pour mieux satisfaire le "client", en profitant des multiples possibilités de la programmation.

3. Création de la table utilisateurs du site

Dans les chapitres précédents, nous avons créer notre mise en page et créer de manière automatique une table. Cette partie va utiliser ces 2 ressources pour permettre une certaine automatisation (ce n'est qu'un début) de l'installation et de l'utilisation. Appelons cette table members. La création de la table doit:

1. Ouvrir la base de donnée (et vérifier si elle existe)
2. Créer la table "member" et les champs associés.
3. Fermer la base de donnée.

La première et la dernière partie vont utiliser les fichier start.php et stop.php du chapitre précédant. Commençons par créer le fichier install_member.

```
<?php
require ('includes/start.php');
// cette partie ouvre la base de donnée
/*
Création de la requête
*/
require ('includes/stop.php');
```

```
echo 'Base de donnée fermée';
?>
```

4. La requête MySQL

Il nous reste à créer la requête SQL. Dissocions le problème en différentes parties Examinons d'abord les champs et les types de données associées. Pour les types de champs, vous pouvez revoir le chapitre 8.

Nom	Type	Remarque	requête assosciée
uid	smallint(6)	auto-increment, clé primaire	**uid smallint(6) primary key NOT NULL auto_increment**
titre	type enum	liste choix, null interdit	titre enum('Monsieur','Madame','Mademoiselle') Not Null
nom	varchar(25)	null interdit	nom varchar(25) Not Null
prenom	varchar(25)	null interdit	prenom varchar(25) Not Null
adresse	varchar(40)		adresse varchar(40)
codepostal	varchar(10)		codepostal varchar(10)
ville	varchar(25)		ville varchar(25)
pays	varchar(20)	null interdit	pays varchar(20) Not Null
telephone	varchar(20)	null interdit	telephone varchar(20) Not Null
email	varchar(60)	null interdit	email varchar(60) Not null
username	varchar(25)	null interdit	username varchar(25) Not Null
password	varchar(40)	null interdit	password varchar(40) Not Null
type	char(2)	null interdit	type char(2) Not null
bday	smallint(2)		bday smallint(2)
bmonth	smallint(2)		bmonth smallint(2)
byear	smallint(4)		byear smallint(2)
regdate	int(10)	unsigned, null interdit, la date d'inscription	regdate int(10) unsigned not null

Les contraintes sont données dans cet exercice, vous pourrez en créer d'autres (ou en supprimer si nécessaire). La requête se définit comme suit (il suffit d'insérer les requêtes SQL, séparées par des virgules):

```
$requete="CREATE TABLE if not exists member (uid smallint(6) primary key NOT NULL auto_increment,titre enum('Monsieur','Madame','Mademoiselle') Not null,nom varchar(25) Not Null,prenom varchar(25) Not Null,adresse
```

```
varchar(40),codepostal varchar(10),ville varchar(25),pays varchar(20) Not
Null,telephone varchar(20) Not Null, email varchar(60) Not null,username
varchar(25) Not Null,password varchar(40) Not Null,type char(2) Not null,bday
smallint(2),bmonth smallint(2),byear smallint(4),regdate int(10) unsigned not null
)";
```

Le code de création de la table MySQL devient:

```
<?php
require ('includes/start.php');
// cette partie ouvre la base de donnée
// création de la table
$requete="CREATE TABLE if not exists member (uid smallint(6) primary key
NOT NULL auto_increment,titre enum('Monsieur','Madame','Mademoiselle') Not
null,nom varchar(25) Not Null,prenom varchar(25) Not Null,adresse
varchar(40),codepostal varchar(10),ville varchar(25),pays varchar(20) Not
Null,telephone varchar(20) Not Null, email varchar(60) Not null,username
varchar(25) Not Null,password varchar(40) Not Null,type char(2) Not null,bday
smallint(2),bmonth smallint(2),byear smallint(4),regdate date )";
$erreur=mysql_query($requete);
$erreur1=mysql_error();
print($erreur."<br>");
print($erreur1);
// fin création table
require ('includes/stop.php');
echo 'Base de donnée fermée';
?>
```

Dans PHPMyAdmin, ça doit donner ceci:

☐	uid	smallint(6)			Non		auto_increment	✎	✕	
☐	titre	enum('Monsieur', 'Madame', 'Mademoiselle')	latin1_swedish_ci		Non	Monsieur		✎	✕	
☐	nom	varchar(25)	latin1_swedish_ci		Non			✎	✕	
☐	prenom	varchar(25)	latin1_swedish_ci		Non			✎	✕	
☐	adresse	varchar(40)	latin1_swedish_ci		Oui	*NULL*		✎	✕	
☐	codepostal	varchar(10)	latin1_swedish_ci		Oui	*NULL*		✎	✕	
☐	ville	varchar(25)	latin1_swedish_ci		Oui	*NULL*		✎	✕	
☐	pays	varchar(20)	latin1_swedish_ci		Non			✎	✕	
☐	telephone	varchar(20)	latin1_swedish_ci		Non			✎	✕	
☐	email	varchar(60)	latin1_swedish_ci		Non			✎	✕	
☐	username	varchar(25)	latin1_swedish_ci		Non			✎	✕	
☐	password	varchar(40)	latin1_swedish_ci		Non			✎	✕	
☐	type	char(2)	latin1_swedish_ci		Non			✎	✕	
☐	bday	smallint(2)			Oui	*NULL*		✎	✕	
☐	bmonth	smallint(2)			Oui	*NULL*		✎	✕	
☐	byear	smallint(4)			Oui	*NULL*		✎	✕	
☐	regdate	date			Oui	*NULL*		✎	✕	

Tout cocher / Tout décocher *Pour la sélection :*

Il vous reste à copier les codes ci-dessus dans le dossier www de easyphp et d'exécuter les différentes requêtes pour créer la table. Vous pouvez les adapter à votre propre utilisation mais dans le cadre de cette formation webmaster, ce n'est pas conseillé. Ces commandes sont automatiquement exécutées lorsque vous installez un forum, un portal, un livre d'or, site de vente en ligne, ... sous licence GPL. Les commandes DROP (supprimer) sont également utilisables dans la console MYPHPADMIN de votre hébergement Internet éventuellement.

14.13. Premier formulaire d'entrée des petites annonces

Maintenant que nous avons créé la table utilisateur MySQ, nous allons créer le formulaire et envoyer les données vers la base de donnée. Deux méthodes sont utilisables avec leurs avantages et leurs défauts.

1. La **première méthode** utilise un formulaire en HTML et un autre fichier en PHP qui vérifie les données. C'est la méthode la plus simple. Par contre, en cas de données

incorrectes, elle nécessite que l'utilisateur reviennent en arrière avec le navigateur pour corriger.

2. La **deuxième méthode** utilise un seul fichier PHP qui insère et vérifie les données (comme dans l'exercice précédant). On appelle de type de formulaire auto-invocant.

Par facilité dans ce cours, nous utilisons ici la méthode HTML. L'autre méthode sera utilisée plus tard.

2. Création du formulaire

La première partie va créer le formulaire en HTM (inséré dans 1 fichier php) reprenant les différents champs à créer:

Nom d'utilisateur (*)

Mot de passe (*) (minimum 5 caractères)

Adresse mail (*)

Titre Monsieur

Nom (*) Prénom (*)

Adresse (minimum 5 caractères)

Code Postal Ville (*) (minimum 3 caractères)

Pays (*)

Téléphone

Date de naissance: Jour mois Année

Remarque, ce formulaire n'envoie aucune données

Envoyer

Les champs uid (numéro d'utilisateur), type (utilisateur, administrateur, ...) et regdate (date d'inscription) ne sont pas reprises dans le formulaire, elle seront insérées automatiquement en PHP. Ce formulaire peut éventuellement être créer par FrontPage (ce qui est plus facile), mais voici le codage en html.

```
<form method="POST" action="mail.php" name="creer-utilisateur">
```

```
<p>Nom d'utilisateur (*) <input type="text" name="username" size="25"></p>

<p>Mot de passe (*) <input type="password" name="password" size="40">
(minimum 5 caractères)</p>

<p>Adresse mail (*) <input type="text" name="email" size="60"></p>

<p>Titre <select size="1" name="titre">

<option selected>Monsieur</option>

<option>Madame</option>

<option>Mademoiselle</option>

</select></p>

<p>Nom (*) <input type="text" name="nom" size="25"> Prénom (*) <input
type="text" name="prenom" size="25"></p>

<p>Code Postal <input type="text" name="codepost" size="10"> Ville (*) <input
type="text" name="ville" size="25"> (minimum 3 caractères)</p>

<p>Pays (*) <input type="text" name="pays" size="20"></p>

<p>Téléphone <input type="text" name="telephone" size="20"></p>

<p>Date de naissance: Jour <input type="text" name="bday" size="2"> mois <input
type="text" name="bmonth" size="2">

Année <input type="text" name="byear" size="4"></p>

<p> <p> <input type="submit" name="ok" value="Envoyer" /></p>

</form>
```

3. Méthode POST et GET

Deux méthodes sont utilisée pour transférer un message, la méthode POST et la méthode GET. GET est la méthode par défaut.

1. **GET** envoie les informations via l'URL comme:
http://www.ybet.be/mail.php?ok=envoyer&titre='Monsieur'
Toutes les variables sont reprises dans l'URL comme nom-variable=valeur. C'est un avantage en sites dynamiques au niveau référencement des différentes pages dans les moteurs de recherche.

2. **POST** par contre n'utilise par l'URL fait passer les informations via l'en-tête HTTP , en même temps que les informations de navigation (adresse IP, version navigateur, résolution écran, ...). Cette méthode est donc plus sécurisée.

La méthode GET ne permet que de transférer quelques octets, la méthode post permet d'envoyer des variables de grande taille.

La méthode de réception des données est également différente.

- **$_GET** (PHP4 compris) rassemble les données transmises par la méthode GET (dans les versions 4 et inférieures, on utilisait **HTTP_GET_VARS)**
- **$_POST** (PHP4 compris) récupère les données par la méthode POST (PHP 4 et inférieures: **HTTP_POST_VARS)**

Pour savoir si les données sont transmises, il suffit de vérifier si des valeurs sont incluses dans la variable tableau.

$_GET['nom-variable']='valeur-variable'

est équivalent à

$_POST['nom-variable']='valeur-variable'

Dans cette formation, **nous utiliserons POST**. Elle se retrouve dans l'en-tête du formulaire: <form method="POST" action="mail.php" name="creer-utilisateur"> qui demande envoie le résultat au fichier mail.php avec comme titre (optionnel) creer-utilisateur.

4. Traitement de l'information.

Une fois les données envoyées avec le bouton OK, le fichier suivant que nous appellerons **mail.php** par facilité va:

1. Récupérer les données (si elles existent)
2. Vérifier les informations obligatoires et la taille des informations transmises
3. Transmettre un message d'erreur en cas de données non valides
4. Vérifier si l'adresse mail et le nom d'utilisateur n'existent pas déjà dans la table "member"
5. Transmettre un message d'erreur en cas de duplicate
6. Si l'ensemble est correct, le fichier va transmettre les données dans la table "member"

Chaque variable de notre formulaire va correspondre à une variable php. Par facilité, nous allons faire correspondre le nom du champ (titre par exemple) avec le même nom de variable PHP ($titre).

Les vérifications à effectuer sont:

- Champs obligatoires
- Longueur minimum et maximum des champs.
- Vérification d'une adresse mail valide.

Nom	Variable PHP correspondante	Valeur nulle autorisée	valeur minimum, maximum
titre	$titre	Non	Liste: Monsieur, Madame, Mademoiselle. (Cette partie est faite par le formulaire)
nom	$nom	Non	3,25
prenom	$prenom	Non	3,25
codepostal	$codepostal	Oui	4,10
ville	$ville	Oui	3,25
pays	$pays	Non	3,20
telephone	$telephone	Non	8,20
email	$email	Non	8,60, caractère @, serveur mail
username	$username	Non	2,25
password	$password	Non	5,40
bday	$bday	Oui	chiffre entre 1 et 31
bmonth	$bmonth	Oui	chiffre entre 1 et 12
byear	$byear	Oui	chiffre entre 1900 et 2050

5. Récupération des données.

L'ensemble des variables vont être récupérées dans notre fichier mail.php par la commande HTTP_POST_VARS, soit

```
<?php
$titre=$_POST['titre'];
echo $titre;
?>
```

Mais est-ce nécessaire. Finalement nous pouvons directement utiliser le résultat de la variable tableau $_POST['nom-variable'].

6. Vérification des données

La première partie va vérifier le nombre de caractères des champs. Pour cela, nous allons créer une fonction comme vu au chapitre 4 de ce cours en ligne. Ceci va éviter de longues lignes de codes.

```
function taille_variable($variable,$taille_min=0,$taille_max=0){
```

```
    global $_POST;
    if(!isset($_POST[$variable])){
    // valeur non définie
    return false;
    }elseif (strlen($_POST[$variable])<$taille_min){
    return False;
    }elseif(strlen($_POST[$variable])>$taille_max){
    return FALSE;
    }
    return True;
    }
```

Analysons cette fonction.

- La première partie demande 3 arguments: le nom de la variable, la taille minimum et la taille maximum. En ajoutant une valeur pour les paramètres maximum et minimum, nous donnons une valeur par défaut si ces 2 variables ne sont pas transmises à la fonction.
- La deuxième ligne **global $_POST**; déclare la variable. C'est nécessaire pour toutes fonctions. Par contre, une fois ouverte dans une fonction, elle est accessible pour l'ensemble de script.
- La suite utilise la fonction STRLEN() qui vérifie le nombre de caractères. Nous l'avons déjà étudiée au chapitre 6.

Certains champ sont optionnels. Par contre, si une valeur est rentrée, nous devons vérifier si les informations de taille sont correctes. Remarquez que dans ce cas, nous utilisons une nouvelle fonction if. Certaines versions PHP n'acceptent pas l'imbrications de fonctions IF.

```
if (strlen($_POST['codepost'])<>0){
  if(!taille_variable('codepost',4,10)){
    echo 'Code Postal invalide, entre 4 et 10 caractères';
    echo'<br>';
  }
}
```

Au total, si le contenu est correct, la valeur renvoyée est TRUE, False sinon. La partie complète de vérification de la taille devient:

```
    <?php
if(!taille_variable('username',2,25)){
echo'Login invalide, rentrez minimum 2 caractères';
echo'<br>';

}elseif (!taille_variable('password',5,40)){
echo'Mot de passe invalide, rentrez minimum 5 caractères, 40 caractères maximum';
echo'<br>';
}elseif (!taille_variable('email',8,60)){
echo'email invalide, rentrez minimum 8 caractères';
echo'<br>';
```

```
}elseif(!taille_variable('nom',3,25)){
echo'Nom invalide, rentrez votre nom (minimum 3 lettres)';
echo'<br>';
}elseif (!taille_variable('prenom',3,25)){
echo'Prénom invalide, rentrez votre nom';
echo'<br>';
}
}
if (strlen(_POST['codepost'])<>0){
  if(!taille_variable('codepost',4,10)){
   echo'Code Postal invalide, entre 4 et 10 caractères';
   echo'<br>';
 }
}
if (!taille_variable('ville',3,25)){
 echo'Ville invalide, rentrez minimum 3 caractères';
 echo'<br>';
}elseif (!taille_variable('pays',3,20)){
 echo'Pays invalide, rentrez minimum 3 caractères';
 echo'<br>';
}elseif (strlen($_POST['telephone'])<>0){
 if (!taille_variable('telephone',5,20)){
  echo'Numéro de téléphone non valide';
  echo'<br>';
 }
}
if (strlen($_POST['bday'])<>0){
 if (!taille_variable('bday',1,2)){
  echo'Jour de naissance invalide, rentrez 2 chiffres';
  echo'<br>';
 }if (($_POST['bday'])<"1"){
echo'Jour inférieur à 1';
echo'<br>';
}elseif (($_POST['bday'])>"31"){
echo'Jour supérieur à 31';
echo'<br>';
}
}
if (strlen($_POST['bmonth'])<>0){
if (!taille_variable('bmonth',2,2)){
echo'Mois de naissance invalide, rentrez 2 chiffres';
echo'<br>';
}
if ($_POST['bmonth']<1){
echo'Mois de naissance >0';
echo'<br>';
}elseif ($_POST['bmonth']>12){
echo'Mois de naissance <12';
echo'<br>';
}
```

```
}
if (strlen($_POST['byear'])<>0){
if (!taille_variable('byear',2,2)){
echo'Année de naissance invalide, rentrez 2 chiffres';
echo'<br>';
}
if ($_POST['byear']<1900){
echo'Année de naissance >1900';
echo'<br>';
}elseif ($_POST['byear']>2020){
echo'Année de naissance <2020';
echo'<br>';
}
}
?>
<p>Recommencez, revenez en arrière avec votre navigateur</a></p>
<?php
function taille_variable($variable,$taille_min=0,$taille_max=0){
global $_POST;
if(!isset($_POST[$variable])){
// valeur non définie
return false;
}elseif (strlen($_POST[$variable])<$taille_min){
return False;
}elseif(strlen($_POST[$variable])>$taille_max){
return FALSE;
}
return True;
}
?>
```

7. Vérifier l'adresse Mail.

Il nous faut encore corriger certaines parties, notamment au niveau de l'adresse mail où nous ne vérifions que la taille. Deux vérifications sont possibles: le caractères @ doit être inclut dans l'adresse, nous pouvons également vérifier si le DNS accepte les mails.

La vérification du caractère @ dans la chaîne mail va utiliser la fonction strpos() que nous avons vu au chapitre 6 de cette formation. Comme rappel

STRPOS($variable,$variable1): revoie le nombre de caractère devant la lettre $variable1. Exemple: STRPOS ("YBET informatique,"i") renvoie le nombre 5. Si la réponse est false, le caractère n'est pas trouvé dans la chaîne.

```
if (!strpos($_POST['email'],'@')){
  echo 'Adresse mail non valide';
  // vérifie si le caractère @ est inclus dans la chaîne
}
```

Nous aurions également pu vérifier la position du caractère dans la chaîne. En premier, l'adresse mail n'est pas valide, moins de 5 positions avant la fin de la chaîne est également une adresse de messagerie invalide (exemple: ybet@a.b n'est pas valide puisque les nom de domaine incluent minimum 2 caractères pour l'extension, le point plus au moins 1 lettre de nom de domaine).

Nous allons simplement interroger le serveur. Pour accepter des adresses mails, le serveur DNS doit être configurer spécifiquement. Ceci n'est pas réellement une preuve que l'adresse de messagerie est effectivement valide, le nom utilisateur peut être faux avec un serveur valide.

Cette vérification se fait en 2 parties: la récupération du nom de domaine et l'envoi de la vérification au serveur. Commençons par récupérer le nom utilisateur et le nom de domaine par la fonction PHP explode(). Cette fonction renvoie les 2 valeurs dans un tableau. Nous allons également utiliser la fonction **list** pour récupérer les 2 valeurs sous forme de variable.

La fonction de récupération devient **list($user,$dns)=explode("@",$_POST['email']**

checkdnsrr($dns) renvoie true si le nom de domaine est configuré, false sinon. Cette fonction n'est pas normalement pas utilisable sous windows. Vous ne pourrez donc pas la tester directement avec EasyPhp.

If (!checkdnsrr($dns)){

  echo'L\'adresse mail n'est pas valide';

}

Au total:

```
list($user,$dns)=explode("@",$_POST['email']
If (!checkdnsrr($dns)){
   echo'L\'adresse mail n'est pas valide';
}
```

L'ensemble de la procédure devient:

```
<?php
if(!taille_variable('username',2,25)){
echo'Login invalide, rentrez minimum 2 caractères';
echo'<br>';

}elseif (!taille_variable('password',5,40)){
echo'Mot de passe invalide, rentrez minimum 5 caractères, 40 caractères maximum';
echo'<br>';
}elseif (!taille_variable('email',8,60)){
   echo'email invalide, rentrez minimum 8 caractères';
   echo'<br>';
}
```

```
if (!strpos($_POST['email'],'@')){
   echo'Adresse mail non valide';
}
list($user,$dns)=explode("@",$_POST['email']);
If (!checkdnsrr($dns)){
  echo'L\'adresse mail n\'est pas valide';
}
if(!taille_variable('nom',3,25)){
echo'Nom invalide, rentrez votre nom (minimum 3 lettres)';
echo'<br>';
}elseif (!taille_variable('prenom',3,25)){
echo'Prénom invalide, rentrez votre nom';
echo'<br>';
}
}
if (strlen($_POST['codepost'])<>0){
if(!taille_variable('codepost',4,10)){
echo'Code Postal invalide, entre 4 et 10 caractères';
echo'<br>';
}
}
if (!taille_variable('ville',3,25)){
echo'Ville invalide, rentrez minimum 3 caractères';
echo'<br>';
}elseif (!taille_variable('pays',3,20)){
echo'Pays invalide, rentrez minimum 3 caractères';
echo'<br>';
}elseif (strlen($_POST['telephone'])<>0){
if (!taille_variable('telephone',5,20)){
echo'Numéro de téléphone non valide';
echo'<br>';
}
}
if (strlen($_POST['bday'])<>0){
if (!taille_variable('bday',1,2)){
echo'Jour de naissance invalide, rentrez 2 chiffres';
echo'<br>';
}if (($_POST['bday'])<"1"){
echo'Jour inférieur à 1';
echo'<br>';
}elseif (($_POST['bday'])>"31"){
echo'Jour supérieur à 31';
echo'<br>';
}
}
if (strlen($_POST['bmonth'])<>0){
if (!taille_variable('bmonth',2,2)){
echo'Mois de naissance invalide, rentrez 2 chiffres';
echo'<br>';
}
```

```
if ($_POST['bmonth']<1){
echo'Mois de naissance >0';
echo'<br>';
}elseif ($_POST['bmonth']>12){
echo'Mois de naissance <12';
echo'<br>';
}

}
if (strlen($_POST['byear'])<>0){
if (!taille_variable('byear',2,2)){
echo'Année de naissance invalide, rentrez 2 chiffres';
echo'<br>';
}
if ($_POST['byear']<1900){
echo'Année de naissance >1900';
echo'<br>';
}elseif ($_POST['byear']>2020){
echo'Année de naissance <2020';
echo'<br>';
}
}
?>
<p>Recommencez, revenez en arrière avec votre navigateur</a></p>
<?php
function taille_variable($variable,$taille_min=0,$taille_max=0){
global $_POST;
if(!isset($_POST[$variable])){
// valeur non définie
return false;
}elseif (strlen($_POST[$variable])<$taille_min){
return False;
}elseif(strlen($_POST[$variable])>$taille_max){
return FALSE;
}
return True;
}
?>
```

Cette partie ne fait que vérifier si les données rentrées dans le formulaire sont correctes. Le chapitre suivant va rentrer les données dans la base MySql si les données sont correctes.

14.14. Rentrée des données utilisateur dans la table Mysql

1 Introduction - 2. La méthode - 3. formulaire correct

1. Introduction.

Dans le chapitre précédant, nous avons créé le formulaire d'inscription et vérifié si les données étaient valides. Cette partie va:

1. Vérifier si l'adresse mail et le nom d'utilisateur sont déjà dans la base de donnée
2. Inscrire l'utilisateur
3. Envoyer un mail de confirmation

2. La méthode

La méthode va simplement utiliser une variable. Cette variable va être mise à TRUE en début de traitement. A chaque message d'erreur, la variable va être mise à false. Si le formulaire est rempli correctement, la variable va rester à True (vrai). Dans le cas où la variable reste vraie, nous allons vérifier si le nom utilisateur et l'adresse mail sont déjà utilisés. Si un de ces 2 paramètres est déjà utilisé, un autre message le signalera à l'utilisateur. Dans le cas contraire, les données validés seront rentrées dans la base de données, en même temps que les valeurs systèmes tel que la date d'inscription (en fait la date système du serveur Internet).

3. Formulaire correct

Nous allons commencer par modifier la vérification des données du précédant chapitre en utilisant la variable booléenne $correcte.

- La première partie va être de récupérer une seule variable pour l'adresse mail et l'username.

```
if ($correct){
$mail=$_POST["email"];
$utili= $_POST["username"];
```

- L'ouverture de la table se fait le plus tard possible. Ceci pour (essayer de) réduire le nombre de message d'erreur liés à un nombre important d'utilisateurs simultanés. Nous utilisons de nouveau le fichier stop.php.

```
require('includes/start.php');
// on ouvre la table
```

- La vérification se fait d'abord sur l'username en vérifiant le nombre de lignes renvoyées par la requête de recherche select.

```
$requete="SELECT * FROM member where username='$utili'";
$valeur=mysql_query($requete);
if (mysql_affected_rows()<>0){
echo 'Username existe déjà';
echo '<br>Rentrez un autre nom';
// retour en arrière
}else{
```

- Si le nombre de ligne renvoyé est différent de 0, on vérifie si l'adresse mail existe déjà

```
$requete= "SELECT * FROM member where email='$mail'";
$valeur= mysql_query($requete);
if (mysql_affected_rows()<>0){
echo 'Adresse mail existe déjà';
// procédure pour récupérer le mot de passe
}else{
// nom utilisateur et adresse mail inconnue, on peut rentrer l'utilisateur
echo 'Inscription utilisateur';
}
}
require ('includes/stop.php');
}
```

- Dans le cas où l'adresse mail n'existe pas, on rentre seulement les données dans la table.

La partie suivante va être d'insérer les données dans la base mysql. Première partie, nous allons récupérer les variables du formulaire sous une forme simple (remplacer $_POST['titre'] par $titre par exemple). quelques remarques pour cette partie:

- **uid** est créé automatiquement par mysql, vous ne devez donc pas rentrer de valeurs
- la **date d'inscription** insérée dans la base de donnée est celle de l'ordinateur, elle utilise la variable $regdate=date('Ymd'); qui utilise la fonction php date() avec le format Ymd. Ceci renvoie la valeur 20060523 par exemple, soit le format reconnu par mysql. remarquez que nous n'utilisons par de caractères de séparation.

```
// on commence par vérifier si le formulaire est correctement rentré
if ($correct){
$mail=$_POST["email"];
$utili= $_POST["username"];
$titre=$_POST['titre'];
$nom=$_POST['nom'];
$prenom=$_POST['prenom'];
```

```
$adresse=$_POST['adresse'];
$codepost=$_POST['codepost'];
$ville=$_POST['ville'];
$pays=$_POST['pays'];
$telephone=$_POST['telephone'];
$mail=$_POST["email"];
$utili= $_POST["username"];
$password=$_POST["password"];
$type="US";
$bday=$_POST["bday"];
$bmonth=$_POST["bmonth"];
$byear=$_POST["byear"];
$regdate=date('Ymd');
echo'<br>';
echo'Valeurs correctes';
echo'<br>';
require('includes/start.php');
// on ouvre la table
$requete="SELECT * FROM member where username='$utili'";
$valeur=mysql_query($requete);
if (mysql_affected_rows()<>0){
echo 'Username existe déjà';
echo'<br>Rentrez un autre nom';
// retour en arrière
}else{
$requete="SELECT * FROM member where email='$mail'";
$valeur=mysql_query($requete);
if (mysql_affected_rows()<>0){
echo 'Adresse mail existe déjà';
// procédure pour récupérer le mot de passe
}else{
// nom utilisateur et adresse mail inconnue, on peut rentrer l'utilisateur
echo 'Je rentre utilisateur';
$requete="INSERT member SET
titre='$titre',nom='$nom',prenom='$prenom',adresse='$adresse',codepost='$codepost',ville
='$ville',pays='$pays',
telephone='$telephone', email='$mail',username='$utili',
password='$password',type='$type',bday='$bday',
bmonth='$bmonth',byear='$byear',regdate='$regdate'";
$résultat=mysql_query($requete);
$erreur=mysql_error();
print ($erreur);
}
}
require('includes/stop.php');
}
```

Cette partie est insérée à la suite de la vérification des données. Il nous restera dans le futur à créer la fonction de récupération du mots de passe si l'adresse mail existe déjà et

l'envoi automatique d'un mail à l'administrateur en cas d'erreur de la base de donnée (on est jamais trop prudent).

15. Les catégories d'annonces

Avant de leur permettre aux utilisateurs de rentrer leurs annonces, nous allons créer les catégories. Par facilité, nous allons créer 3 niveaux de catégories. Un seul serait suffisant, mais cette méthode va nous permettre dans le futur d'utiliser cette programmation pour d'autres types de sites, pensez à un annuaire par exemple.

Une solution serait de permettre une liste de choix directement dans le formulaire. Cette méthode a déjà été utilisée pour entrer les utilisateurs (Monsieur, Madame, Mademoiselle). Malheureusement, l'ajout, la modification ou la suppression d'une catégorie nécessiterait de modifier ... la programmation, pas très intéressant pour un développeur de site standard. Nous allons donc créer une table dans notre base de donnée que nous appellerons **CATEGORIE**.

Cette formation est donc le premier niveau de l'administration de notre site d'annonces en ligne.

Remarque: cette partie a été complètement modifiée dans le développement actuel.

2. Création de la table.

La table peut-être crée avec la console PHPMyadmin d'EasyPhp, mais nous allons la créer à l'aide d'une requête MySQL. Pour un rappel sur les commandes sur les tables MySQL.

```
<?php
require('includes/start.php');
// cette partie ouvre la base de donnée
/*
Création de la requête
*/
require('includes/stop.php');
echo'Base de donnée fermée';
?>
```

Les champs à créer dans la table MySql:

nom	type	description
uid	int(10), clé primaire	numéro de la catégorie

Nom	varchar(25)	Nom de la catégorie
Description	Varchar(80)	Description de la catégorie
Attachement	smallint(5) default '0'	Sous catégorie de la catégorie supérieure numéro, 0 si c'est une catégorie
Image	varchar(40)	adresse de l'image associée
Actif	char(1)	actif (oui ou non), O par défaut

- uid va nous permettre de numéroter nos catégories
- Attachement permet de créer les sous catégories. Cette manière de procéder va nous permettre de créer autant de niveau de catégories que nous souhaitons.
- image permet d'associer une image à la catégorie
- Actif (O / N) permet éventuellement de désactiver une catégorie d'annonces.

```
<?php
require('includes/start.php');
$requete="CREATE TABLE if not exists categorie (uid int primary key NOT NULL
auto_increment,nom varchar(25) not null,description varchar(80) not null,attachement
smallint(5) default '0',image tinytext NOT NULL, actif char(1))";
$erreur=mysql_query($requete);
$erreur1=mysql_error();
print($erreur."<br>");
print($erreur1);
require('includes/stop.php');
?>
```

Enregistrez ce fichier dans votre dossier principal sous le nom creation-cat.php et exécutez-le, la table est directement créée.

3. Insertion d'une catégorie

Pour permettre d'insérer les catégories, nous allons de nouveau un formulaire. Cette fois, nous n'allons pas utiliser la méthode fichier html -> fichier php mais directement utiliser un seul fichier PHP. Cette méthode est plus élégante puisqu'elle n'oblige pas l'utilisateur à faire un retour en arrière avec le navigateur pour corriger des données.

```
<?php

// test d'exécution du sommaire
if (isset($_POST['go'])){
```

```
$nom = $_POST['nom'];

$erreur_nom = "";

$description = $_POST['description'];

$erreur_description = "";

$attachement = $_POST['attachement'];

$erreur_attachement = "";

$image = $_POST['image'];

$erreur_image = "";

$actif = $_POST['actif'];

$erreur_actif = "";

    if ($_POST['nom'] == ''){

      $erreur_nom = "Le nom de la catégorie doit être renseigné<br>";

    } elseif ($_POST['description'] == ''){

      $erreur_description = "Entrez une description<br>";

    } elseif ($_POST['image'] == ''){

      $erreur_image = "Entrez une adresse image valide<br>";

    } else {
      $ajout = "";
     echo"Connexion réussie";
     // entrée des données dans la table

    }

} else {

$nom = "Nom de la catégorie";

$erreur_nom = "";

$description = "Description de la catégorie";

$erreur_description = "";
```

```
    $attachement = "Numéro de la catégorie supérieure";

    $erreur_attachement = "";

    $image = "adresse de l'image";

    $erreur_image = "";

    $actif ="O";

    $erreur_actif = "";
}
$form = "<table>
<form METHOD=\"POST\">
<tr><td>Nom de la catégorie</td><td><input type=text name=nom value=\"$nom\"
size=\"25\"><br>$erreur_nom</td></tr>
<tr><td>Description</td><td><input type=text name=description
value=\"$description\" size=\"40\"><br>$erreur_description</td></tr>
<tr><td>Numéro de la catégorie supérieure</td><td><input type=text
name=attachement value=\"$attachement\"><br>$erreur_attachement</td></tr>
<tr><td>Adresse image</td><td><input type=text name=image value=\"$image\"
size=\"100\"><br>$erreur_image</td></tr>
<tr><td>Actif (O/N)</td><td><input type=text name=actif value=\"$actif\"
size=\"1\"><br>$erreur_actif</td></tr>
<tr><td><input type=submit name=go value=sauve></td><td><input type=reset
name=reset value=\"recommencer\"></td></tr>
</form>
</table>";
print($form);
?>
```

Vous pouvez vérifier le fonctionnement de cette partie ici. Examinons cette partie de code.

La partie finale reprend le formulaire. Il est quasiment identique à un formulaire en HTML, sauf qu'il est enregistré sous forme d'une variable PHP ($form). Il est affiché juste après la fonction conditionnelle If de départ.

Si la condition **if (isset($_POST['go'])){** n'est pas correcte (donc si le bouton go n'est pas cliqué), il n'y a aucun tests. Par contre, si le **bouton go** est utilisé, le test de condition est effectivement vérifié. Par le "else" de la première condition (go), donc au démarrage, les variables sont initialisées. Si le bouton OK est utilisé mais les valeurs fournies sont non valides, seul les messages d'erreur équivalents sont mises à jour. En schématique, ceci devient:

Condition go

- **Oui**
  - récupération des données du formulaire

- Mise à zéro des erreurs
- tests variables
  - OK
    - Données correctes, la catégorie peut-être enregistrée
  - Données non correctes
    - On récupère l'erreur par $erreur_variable
- **Non**
  - On initialise les variables.

Fin condition: affichage du formulaire

Remarquez que nous n'avons pas (encore) vérifier le numéro de l'attachement en cas de sous-catégorie. Pour vérifier si une catégorie existe, nous devons ouvrir la table. De plus, il est préférable d'afficher les catégories existantes, ce sera la prochaine partie. Nous développerons cette fonctionnalité en fin de chapitre.

Maintenant que nous avons vérifié la validité de la catégorie, il nous reste à insérer la catégorie dans la table. La requête MySQL prend la forme:

$requete="INSERT categorie SET nom='$nom',description='$description',attachement='$attachement',image='$image',actif='$actif'";

Un conseil, à ce stade, évitez le caractère ' dans vos descriptions et messages d'erreurs.

4. Affichage des catégories

Récupérer les catégories dans la table ne va pas être trop compliqué, c'est une simple requête SQL de type $requete='SELECT * FROM "valeur"'. Par contre, l'affichage est plus délicat. Nous devons afficher les catégories les unes en-dessous des autres (par ordre alphabétique ...) mais en plus insérer chaque fois les sous-catégories.

La méthode va d'abord rechercher tous les fichier dont le numéro d'attachement est 0. Chaque fois qu'on en rencontre 1, on va récupérer toutes les catégories dont le numéro d'attachement est justement ce numéro de catégorie. Un test sur le nombre total de catégorie va réduire le temps d'accès à la base de donnée, ceci pour éviter de rechercher des catégories inexistantes. Nous allons effectuer le tests via la commande While

Pour arriver à notre tests, nous allons devoir imbriquer plusieurs boucles entre elles.

- La première boucle va vérifier le nombre de catégories dans la table MySQL.
- La deuxième boucle va vérifier les catégories
- La troisième boucle va vérifier les sous-catégories
- La quatrième boucle va vérifier les sous-catégories niveau 2.

L'affichage complet ne pose pas de réelle difficulté. Il prend la forme:

```
<?php
```

```
require('includes/start.php');
$requete='SELECT * FROM categorie';
$valeur=mysql_query($requete);
while ($tableau=mysql_fetch_array($valeur)){
  print("<b>".$tableau["uid"]." ".$tableau["nom"]." ".$tableau["description"]."
".$tableau["attachement"]." ".$tableau["image"]." ".$tableau["actif"]."<br>\n");
}
require('includes/stop.php');
```

Cette méthode affiche tous les enregistrements de la table ligne par ligne. Ce qui nous intéresse est de faire un tri. La présentation n'est pas non plus très esthétique utilise un tableau. Les catégories sont affichées les unes à la suite de l'autre suivant le numéro de uid, mais les catégories ne sont pas dissociées des sous-catégories

```
<?php
?>
<table border="1">
<?php
require('includes/start.php');
$requete='SELECT * FROM categorie';
$valeur=mysql_query($requete);
while ($tableau=mysql_fetch_array($valeur)){
?>
<tr>
<td><?php print($tableau["uid"]);?></td>
<td><?php print($tableau["nom"]);?></td>
<td><?php print($tableau["description"]);?></td>
<td><?php print($tableau["attachement"]);?></td>
<td><?php print($tableau["image"]);?></td>
<td><?php print($tableau["actif"]);?></td>
</tr>
<?php
}

?>
</table>
<?php
print('<br>');
require('includes/stop.php');
?>
```

La première partie de notre affichage est de vérifier le nombre d'enregistrement de notre table, ceci nous évitera de vérifier des lignes qui n'existent pas. Si le nombre de catégories (et sous catégories) est de 10 par exemple, la recherche n'est plus nécessaire dès que nous avons

trouvé 10 catégories.. Nous allons lui attribuer la variable $ligne. Le fichier suivant affiche le nombre de lignes (et donc de catégories) dans la table "categorie".

```
<?php
require('includes/start.php');
$requete='SELECT * FROM categorie';
$valeur=mysql_query($requete);
$ligne=mysql_affected_rows();
print($ligne);
require('includes/stop.php');
?>
```

La variable $ligne va nous servir pour le premier tests.

```
$i=0
  while ($i<=$ligne){
  // boucles internes
}
```

Chaque fois que nous trouvons une nouvelle catégorie, nous incrémentons la variable $i. Le tests suivant va également utiliser une variable à incrémenter. Nous utiliserons $j. Cette variable va prendre toutes les valeurs possibles de 1 à 99.999.999.999 (la taille maximum de notre champ uid). La variable $i ne sert finalement qu'à réduire la recherche au nombre de catégories effectives dans la table.

Commençons par afficher les différentes catégories de la table Mysql à l'aide de ce script PHP:

```
<?php
require('includes/start.php');
$requete='SELECT * FROM categorie';
$valeur=mysql_query($requete);
$ligne=mysql_affected_rows();
print($ligne." catégories <br>");

$result= mysql_query("SELECT MAX(uid) AS LAST_ID FROM categorie");
$result = mysql_fetch_array($result);
$uidlast=$result['LAST_ID'];
print("uid maximum :".$uidlast);
$i=0 ;
$j=1;

---

while ($j<=lastuid){
```

```
 $requete="SELECT * FROM categorie where uid='$j'";
 $valeur=mysql_query($requete);
 if (mysql_affected_rows()<>0){
  $i=$i+1;
  // on vient de trouver une catégorie ou une sous-catégorie
 $tableau=mysql_fetch_array($valeur)
 ?>
```

```
  <table border="1">
    <tr>
        <td><?php print($tableau["uid"]);?></td>
        <td><?php print($tableau["nom"]);?></td>
        <td><?php print($tableau["description"]);?></td>
         <td><?php print($tableau["attachement"]);?></td>
         <td><?php print($tableau["image"]);?></td>
         <td><?php print($tableau["actif"]);?></td>
    </tr>
  </table>
<?php
```

```
}

$j=$j+1;
if($i>=$ligne){
$j=100000000000;
}
print($j." ".$i."<br>");
}
require('includes/stop.php');
?>
```

- La première partie sert à récupérer le nombre de lignes dans la table et initialise les variables $i (nombre de catégorie trouvée) et $j (numéro uid). Elle est suivie par la récupération du numéro uid maximum avec la requête "SELECT MAX(uid) AS LAST_ID FROM categorie". Cette valeur est transmise à la variable $uidlast.
- La deuxième partie débute la boucle. Tant que $j est inférieur ou égal à 99.999.999.999.999, la boucle tourne et recherche la ligne correspondant à uid =$j. Si une valeur est trouvée, on incrémente le nombre de catégories trouvées ($i), sinon (partie 4), on incrémente $j .
- L'affichage se fait dans la partie 3, nous l'avons déjà vu plus haut.
- La partie 4 va vérifier si le nombre de catégories trouvées est supérieures au nombre de lignes dans la table. Si c'est le cas, on donne la valeur 100.000.000.000 à $j, ce qui fait quitter directement la boucle principale. print($j." ".$i."
"); n'est inséré à ce stade que comme vérification et peut-être supprimé.

Au niveau affichage, il est identique à celui ci-dessus (mais en plus compliqué au niveau programmation). Ce qui nous manque, c'est de dissocier les catégories des sous catégories. Remplaçons la partie 1 par ceci:

```
while ($j<=$uidlast){
$requete="SELECT * FROM categorie where uid='$j'";
$valeur=mysql_query($requete);
$tableau=mysql_fetch_array($valeur);
$i=$i+1;
if ((mysql_affected_rows()<>0)&&($tableau["attachement"]=="0")){
// on vient de trouver une catégorie et négliger les sous catégories
?>
```

Nous avons juste déplacé l'incrémentation de la variable $i qui ne tient plus compte que des catégories principales, mais surtout mis 2 conditions (tests logique && - ET). De cette manière, nous n'affichons plus que les catégories. La dernière étape va être pour chaque catégorie de récupérer les sous-catégories.

Une fois une catégorie trouvée, nous allons vérifier dans les enregistrements de la table si le champ attachement est identique à cette catégorie. Si nous insérons les lignes suivantes à la fin de la partie ci-dessous, les sous-catégorie sont également affichées avec un décalage. Nous récupérons les différentes valeurs dans un tableau et les imprimons simplement.

```
// on recherche les sous-catégorie de la catégorie trouvée.

$requete="SELECT * FROM categorie where attachement='$j'";
$valeur=mysql_query($requete);
$tableau=mysql_fetch_array($valeur);
// on vient de trouver une sous catégories de la catégorie $uid
?>
<table border="1">
<tr>
<td width="20"> </td>
<td><?php print($tableau["uid"]);?></td>
<td><?php print($tableau["nom"]);?></td>
<td><?php print($tableau["description"]);?></td>
<td><?php print($tableau["attachement"]);?></td>
<td><?php print($tableau["image"]);?></td>
<td><?php print($tableau["actif"]);?></td>
</tr>
</table>
<?php
```

Malheureusement, cette méthode ne marche qu'avec 2 niveaux de catégories et nous en utilisons 3 ... Nous allons créer une boucle pour les sous-catégories 1 et utiliser cette partie pour les catégories de niveau 3. Finalement, l'affichage complet donne:

```
<?php
require('includes/start.php');
$requete='SELECT * FROM categorie';
$valeur=mysql_query($requete);
$ligne=mysql_affected_rows();
print($ligne." catégories <br>");
$result= mysql_query("SELECT MAX(uid) AS LAST_ID FROM categorie");
$result = mysql_fetch_array($result);
$uidlast=$result['LAST_ID'];
print("uid maximum :".$uidlast);
$i=0 ;
$j=1;
while ($j<=$uidlast){
$requete="SELECT * FROM categorie where uid='$j'";
$valeur=mysql_query($requete);
$tableau=mysql_fetch_array($valeur);
$i=$i+1;
if ((mysql_affected_rows()<>0)&&($tableau["attachement"]=="0")){
// on vient de trouver une catégorie et négliger les sous catégories
?>
<table border="1">
<tr>
<td><?php print($tableau["uid"]);?></td>
<td><?php print($tableau["nom"]);?></td>
<td><?php print($tableau["description"]);?></td>
<td><?php print($tableau["attachement"]);?></td>
<td><?php print($tableau["image"]);?></td>
<td><?php print($tableau["actif"]);?></td>
</tr>
</table>
<?php
// on recherche les sous-catégorie de la catégorie trouvée.
$k=1;
while ($k<$uidlast){
$requete="SELECT * FROM categorie where attachement='$j' && uid='$k'";
$valeur=mysql_query($requete);
$tableau=mysql_fetch_array($valeur);
// début du tests si une sous-catégorie existe
if (mysql_affected_rows()<>0){
// on affiche les sous-catégories 1
?>
```

```
<table border="1">
<tr>
<td width="20"> </td>
<td><?php print($tableau["uid"]);?></td>
<td><?php print($tableau["nom"]);?></td>
<td><?php print($tableau["description"]);?></td>
<td><?php print($tableau["attachement"]);?></td>
<td><?php print($tableau["image"]);?></td>
<td><?php print($tableau["actif"]);?></td>
</tr>
</table>

<?php
//début des catégories 2
$requete="SELECT * FROM categorie where attachement='$k'";
$valeur=mysql_query($requete);
if (mysql_affected_rows()<>0){
while ($tableau=mysql_fetch_array($valeur)){
// on vient de trouver une sous catégories 2
?>
<table border="1">
<tr>
<td width="40"> </td>
<td><?php print($tableau["uid"]);?></td>
<td><?php print($tableau["nom"]);?></td>
<td><?php print($tableau["description"]);?></td>
<td><?php print($tableau["attachement"]);?></td>
<td><?php print($tableau["image"]);?></td>
<td><?php print($tableau["actif"]);?></td>
</tr>
</table>

<?php
}
}
//fin catégories niveau 2
}
// fin tests sous-catégories 1
$k=$k+1;
}
}
$j=$j+1;
if($i>=$ligne){
$j=199999999999;
```

```
}
}
require('includes/stop.php');
?>
```

L'affichage est nettement perfectible, nous l'affinerons avec la mise en page de nos différents scripts php.

5. Insertion des sous-catégories

La dernière partie va être de permettre d'insérer des sous catégories avec une vérification des catégories existantes. Nous reprenons:

- affichage des catégories (partie 4).
- afficher le formulaire d'entrée
- Si un attachement est inséré (différent de 0), vérifier si la catégorie supérieure existe.

Pour l'affichage, nous allons utiliser une fonction par facilité. Nous allons également utiliser une autre fonction (redirect) pour rediriger l'utilisateur vers cette même page lors de l'insertion ou la suppression d'une catégorie qui rappelle le formulaire. Ceci fera la mise à jour de notre page dans ce cas mais d'autres URL sont possibles.

```
function redirect($url, $redir = false) {
global $headmess;

$headmess['redirjs'] = '<script type="text/javascript">function redirect()
{window.location="' . $url . '";} setTimeout("redirect()", 1000);</script>';
if($redir) {
return $headmess['redirjs'];
}
}
```

Pour vérifier si la catégorie au-dessus existe, nous allons utiliser une fonction appelée dans la partie vérification.

```
} elseif (souscategorie($attachement)<0){
// vérification des sous catégories
echo'categorie supérieure inconnue, pas inséré';
// fin de vérification des catégories

et la fonction:

function souscategorie($categorie){
```

```
global $requete,$valeur,$faux;
require('includes/start.php');
$requete="SELECT * FROM categorie where uid='$categorie'";
$valeur=mysql_query($requete);
if (mysql_affected_rows()<>0){
require('includes/stop.php');
// categorie existe, insertion dans la base de donnée
echo'catégorie existe';
return 1;
}else{
// catégorie inconnue, pas insérée.
echo'catégorie inconnue, pas insérée';
require('includes/stop.php');
return -1;
}
}
```

Si la catégorie existe, la fonction renvoie 1. Si la catégorie n'existe pas, on renvoie -1. Cette fonction ne gère pas le niveau de catégories. A ce niveau de développement, nous n'en tenons pas compte.

16. Catégories dans un liste déroulante à partir de la table MySQL

Nous allons maintenant permettre aux utilisateurs d'insérer leurs annonces dans la table annonce. Nous devons non seulement permettre l'insertion d'une annonce mais également vérifier que l'utilisateur permet l'utilisation des cookies. Cette formation étant dédiée à des débutants en PHP - MySQL, nous ferons l'ensemble du développement sur plusieurs parties.

La première partie va être de **créer une liste déroulante** reprenant les catégories dans le formulaire d'entrée. Les catégories sont récupérées à partie de la table MySql (y compris une description reprenant le nom des catégories supérieures). Avant de gérer les utilisateurs, nous afficherons les annonces (ce sera la partie suivante) en les intégrant dans notre lay-out du chapitre 11.

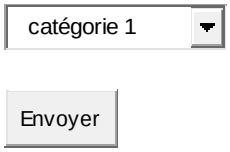
L'insertion d'annonces reprendra un **formulaire auto-appelé** comme dans le chapitre précédant, un seul fichier php reprenait le formulaire et les vérifications.

2. La liste déroulante

Une liste déroulante se code en HTML de la manière suivante. Elle va être insérée dans le formulaire d'entrée de nos petites annonces.

```
<form method="POST" name="catégorie">

<p><select size="1" name="uid">

<option selected value="valeur 1">catégorie 1</option>

<option value="valeur 2">catégorie 2</option>

</select></p>

<p><input type="submit" value="Envoyer" name="B1"></p>

</form>
```

Ce qui affiche:



La difficulté est de reprendre non pas des textes et des valeurs fixes comme ci-dessus mais bien des valeurs reprises dans la table.

La première partie va reprendre l'ensemble des données dans une **variable php sous forme de tableaux**. Commençons par créer notre formulaire auto-invoquant:

```
<?php
if (isset($_POST['go'])){
$uid = $_POST['uid'];
echo$uid;
}
$form = "<table>
<form METHOD=POST>
<form method=\"POST\">
<p><select size=\"1\" name='uid'>
<option selected value=\"valeur 1\">catégorie 1</option>
<option value=\"valeur 2\">catégorie 2</option>
</select></p>
<p><input type=\"submit\" value=\"Envoyer\" name=\"go\"></p>
</form>
</table>
<HR>";
print($form);
?>
```

Ce petit programme ne fait qu'afficher la valeur correspondant à la catégorie. Si vous sélectionnez catégorie 1, il affiche donc valeur 1. Nous allons modifier ce formulaire plus tard pour qu'il affiche le numéro uid en fonction de l'**affichage de la catégorie**. Pour l'instant commençons par récupérer les catégories avec leur description dans une variable PHP sous forme de tableau que nous appellerons **$categorie-tab**

3. Récupération de la catégorie - sous catégorie dans un tableau.

1. La première partie du fichier php ne fait que créer la requête MySQL qui récupère les données.
2. La deuxième partie va remplir un tableau ($categorie_tab) avec les mêmes valeur correspondante de la table MySQL. Seule distinction, nous ne récupérons pas les catégories non actives, seulement toutes celles dont le contenu est différent de "N". A ce stade, les manipulations sur la table sont terminées et nous pouvons la fermer.
3. La troisième partie est le coeur de notre affichage. Nous recherchons dans une boucle les catégories des sous catégories. Pour une catégorie, la variable 'attachement' de la table MySQL =0
4. Si c'est une catégorie (en dessous), la cat1 devient le numéro uid., CAT2 et CAT 3 ne sont pas remplis. 'complet' reprend simplement le nom de la catégorie et nous affichons le nom, et les différent niveaux de catégorie

```
$categorie_tab[$i]['cat1']=$categorie_tab[$i]['uid'];
 $categorie_tab[$i]['cat2']=0;
$categorie_tab[$i]['cat3']=0;
$categorie_tab[$i]['complet']=$categorie_tab[$i]['nom'];
 print("<br>".$categorie_tab[$i]['complet']." ".$categorie_tab[$i]['cat1']."-
".$categorie_tab[$i]['cat2']." ".$categorie_tab[$i]['cat3']);
```

5. La difficulté se situe au niveau des sous catégorie. Nous distinguons les sous-catégorie niveau 1 (en dessous d'une catégorie) des sous catégories niveau 2. Dans le cas d'une sous catégorie niveau 1, 'attachement' de la catégorie supérieure doit être à 0. Ce tests se fait par la boucle.

```
 while ($j<$ligne){
    if(($categorie_tab[$j]['uid']==$k)&&($categorie_tab[$j]['attachement']=="0")){
     $l=1;
     $categorie_tab[$i]['complet']= $categorie_tab[$j]['complet'];
    }
  $j=$j+1;
  }
// si $l = 1, il s'agit bien d'une sous-catégorie de niveau 1
```

Nous remplissons ensuite simplement les valeurs. Remarquez que pour "additionner" deux chaînes ne caractères, nous n'utilisons pas + mais bien . (comme pour les commandes print() et echo de l'affichage).

6. Dans le cas d'une sous catégorie niveau 2, nous reprenons deux boucles, une pour récupérer la catégorie, 1 pour la sous catégorie.

En schématique, la procédure se résume à:

```
Récupération des valeurs de la table dans le tableau $categorie_tab
```

Rentrée des valeurs uniquement si la catégorie est active

Si sous-catégorie

  Si sous-catégorie niveau 1

    Boucle de récupération du nom de la catégorie supérieure

  sinon catégorie niveau 2

    Boucle de récupération de la sous catégorie supérieure

    Boucle de récupération de la catégorie

sinon catégorie

Dans tous les cas, nous remplissons $categorie_tab['complet'] du nom de la catégorie

```
<?php
$categorie_tab=array();
$requete="SELECT * FROM categorie";
$valeur=mysql_query($requete);
$i=0;
$ligne=0;
while ($tableau=mysql_fetch_array($valeur)){
// on récupère les données de la table
// print($ligne);
if($tableau['actif']<>"N"){
$categorie_tab[$i]['uid']=$tableau['uid'];
$categorie_tab[$i]['nom']=$tableau['nom'];
$categorie_tab[$i]['description']=$tableau['description'];
$categorie_tab[$i]['attachement']=$tableau['attachement'];
$i=$i+1;
}else{
print('<br>Catégorie non active');
}
}
require('includes/stop.php');
// début du traitement des catégories - sous catégories dans le tableau
$ligne=$i;
print($ligne." lignes dans le tableau");
$i=0;
while ($i<$ligne){
if ($categorie_tab[$i]['attachement']<>0){
// vérification des sous catégories 1: ce n'est pas une catégorie
```

```
$k=$categorie_tab[$i]['attachement'];
$nom=$categorie_tab[$i]['nom'];
// on récupère attachement et vérifie si uid est effectivement avec attachement à 0
$j=0;
$l=0;
//
while ($j<$ligne){
if(($categorie_tab[$j]['uid']==$k)&&($categorie_tab[$j]['attachement']=="0")){
$l=1;
$categorie_tab[$i]['complet']= $categorie_tab[$j]['complet'];
}
$j=$j+1;
}
// si $l = 1, il s'agit bien d'une sous-catégorie de niveau 1
if ($l==1){
$categorie_tab[$i]['cat1']=$k;
$categorie_tab[$i]['cat2']=$categorie_tab[$i]['uid'];
$categorie_tab[$i]['cat3']="0";
$categorie_tab[$i]['complet']= $categorie_tab[$i]['complet']." - ".$categorie_tab[$i]['nom'];
}else{
// c'est une catégorie de niveau 2, il faut retrouver les 2 catégories supérieures
$j=0;
$l=0;
while ($j<$ligne){
if(($categorie_tab[$j]['uid']==$k)&&($categorie_tab[$j]['attachement']<>"0")){
$l=$j;
$j=$ligne;
}
$j=$j+1;
}

$categorie_tab[$i]['cat1']=$categorie_tab[$l]['attachement'];
$categorie_tab[$i]['cat2']=$categorie_tab[$l]['uid'];
$categorie_tab[$i]['cat3']=$categorie_tab[$i]['uid'];
// on recherche ensuite la catégorie
$m=0;
while ($m<$ligne){
if
(($categorie_tab[$m]['uid']==$categorie_tab[$i]['cat1'])&&($categorie_tab[$m]['attachement']==0)){
$categorie_tab[$i]['complet']=$categorie_tab[$m]['nom']." - ".$categorie_tab[$l]['nom']." -
".$categorie_tab[$i]['nom'];
$m=$ligne;
}
$m=$m+1;
```

```
}
}
// fin de vérification des catégories 1
}else{
// c'est une catégorie }else{
$categorie_tab[$i]['cat1']=$categorie_tab[$i]['uid'];
$categorie_tab[$i]['cat2']=0;
$categorie_tab[$i]['cat3']=0;
$categorie_tab[$i]['complet']=$categorie_tab[$i]['nom'];
}
$i=$i+1;
}
if (isset($_POST['go'])){
$choix = $_POST['uid'];
echo"<br><b>".$choix."</b>";
}
$i=0;
$liste="<table>
<form method=\"POST\">
<p><select size=\"1\" name='uid'>";
while ($i<$ligne){
$uid= $categorie_tab[$i]['uid'];
$complet= $categorie_tab[$i]['complet'];
$liste=$liste."<option value=".$uid.">".$complet."</option>";
$i=$i+1;
}
$liste=$liste."<p><input type=\"submit\" value=\"Envoyer\" name=\"go\"></p>
</form>
</table>
<HR>";
echo$liste;
?>
```

Nous avons maintenant dans un tableau avec un nombre de lignes = $ligne reprenant non seulement le 'uid', mais également le nom réel de la catégorie. Il nous reste maintenant à intégrer cette fonction dans notre formulaire.

4. Liste déroulante des différentes catégories dans un formulaire.

Maintenant que nous avons créer le tableau reprenant les différentes catégories, il nous reste à insérer ces données dans un formulaire pour afficher le nom complet de la catégorie et faire passe comme variable son numéro.

```
if (isset($_POST['go'])){
$choix = $_POST['uid'];
echo"<br><b>".$choix."</b>";
}
$i=0;
$liste="<table>

<form method=\"POST\">
<p><select size=\"1\" name='uid'>";
while ($i<$ligne){
$uid= $categorie_tab[$i]['uid'];
$complet= $categorie_tab[$i]['complet'];
$liste=$liste."<option value=".$uid.">".$complet."</option>";
$i=$i+1;
}
$liste=$liste."<p><input type=\"submit\" value=\"Envoyer\" name=\"go\"></p>
</form>
</table>
<HR>";
echo$liste;
```

En insérant ces quelques lignes en fin de notre programme PHP, nous obtenons la liste des catégorie complète dans une liste déroulante de choix reprise de notre tableau. Il affiche également le numéro correspondant à notre choix si vous appuyez sur le bouton envoyer.

$liste est la variable utilisée pour afficher le formulaire. La première partie insère le début du formulaire. Dans la boucle While, nous reprenons les différentes valeurs comme options. Une fois que nous avons terminé de reprendre toutes les options, nous terminons notre formulaire et affichons la variable $liste contenant l'ensemble du formulaire. Pour l'instant, nous ne faisons qu'afficher le numéro de la catégorie choisie (echo"
".$choix."";) en gras.

La partie suivante va nous permettre de créer le formulaire complet pour insérer les annonces, même si nous ne gérerons pas encore les utilisateurs.

17. Modification d'une table MySql par PHP

1 Introduction - 2. La commande ALTER - 3. Les champs à remplir

L'affichage de nos différentes catégories du précédent chapitre va maintenant nous permettre de créer le formulaire d'entrée des annonces. La structure de la table a été crée avec les champs suivants. Vous l'avez remarqué, le champ reprenant la catégorie du bien n'est pas repris dans la table, ni le code utilisateur de celui qui a inséré l'annonce, il va nous falloir le

rajouter. Ce chapitre va nous permettre d'étudier les modifications sur une table MySql existante.

	type	primary-key, Index	remarque	Instruction SQ
code	numérique, integer	Primary-key	auto incrémentation	code int primary key NOT NU auto_increment
titre	VARCHAR(120)		titre de l'annonce limitée à 120 caractères, insensible à la casse	titre varchar(120) not null
description	BLOB		texte limité à 65535 caractères	description blob not null
photo	varchar(255)		adresse et nom de la photo, 255 caractères	photo varchar(255)
Ville	VARCHAR(40)	index	limité à 40 caractères	ville varchar(40), index(ville)
Pays	type enum		choix entre Belgique, France, Luxembourg	pays enum('Belgique','France','Lux
prix	decimal(10,2)		décimal avec 2 chiffres derrière la virgule	prix decimal(8,2) not null
dateinsertion	date		date courte	dateinsertion date
telephone	varchar(15)			telephone varchar(15)
mail	varchar(30)			mail varchar(30)

2. La commande Alter

La commande ALTER TABLE permet d'insérer, de modifier ou de supprimer un champ dans une table existante. Nous ne verrons pas toutes les commandes dans ce cours, mais voici l'ensemble des syntaxes.

ALTER [IGNORE] TABLE Nom_table alter_spec [, alter_spec ...]

où: Alter_specification=
ADD [COLUMN] create_definition [FIRST | AFTER column_name]: ajouter une colonne après (AFTER) une autre colonne
ou **ADD INDEX [Nom_index] (index_nom_col,...)**: ajout d'un index à la

colonne Index_nom_col.
ou **ADD PRIMARY KEY (index_nom_col,...)**: ajoute une clé primaire à la colonne index_nom_col
ou **ADD UNIQUE [Nom_index] (index_Nom_col,...)**: index unique pour le champ Nom_index
ou **ALTER [COLUMN] Nom_col {SET DEFAULT literal | DROP DEFAULT}**
ou **CHANGE [COLUMN] old_Nom_col create_definition**
ou **MODIFY [COLUMN] create_definition**
ou **DROP [COLUMN] Nom_col:** supprime la colonne Nom_col
ou **DROP PRIMARY KEY:** supprime la clé primaire
ou **DROP INDEX nom_cle**: supprime l'index nom_cle
ou **RENAME [AS] new_Nom_table**: renome la table

Plusieurs modifications peuvent être faite sur la même table avec la même requête SQL, chaque modification étant séparée par une ,.

IGNORE est une extension MySQL de SQL et contrôle la manière dont Alter Table gère les clés dupliquées: Si l'option est omise, la modification est abordée. Dans le cas où cette option est utilisée, seule la première colonne est conservée, les autres colonnes en conflit sont supprimées.

Le champ à insérer est:

uid-cat	smallint(6)	numéro de la catégorie
uid-util	int(10)	numéro utilisateur
tel	char(1)	téléphone utilisateur affiché (O ou N)

Notre requête MySQL d'insertion de champ devient:

```
<?php
require('includes/start.php');

Mysql_select_db('ybet');
$requete="alter table contenu add uid_cat smallint(6) NOT NULL, add uid_util int(10) NOT NULL, add tel char(1)";
$erreur=mysql_query($requete);
require('includes/stop.php');
?>
```

3. Les champs à remplir.

1. **Code** sera rempli automatiquement (auto-indent)

2. **Titre** est le titre de l'annonce, limité à 120 caractères
3. **Description** est limité à 65535 caractères
4. **Photo** nécessite l'adresse de la photo, mais également de télécharger la photo directement sur le site
5. **Ville** peut être inséré par l'utilisateur ou repris de fiche utilisateur. Dans notre table MEMBER, nous n'avons pas obligé à rentrer une valeur. Nous proposerons donc par défaut celle de l'utilisateur.
6. **Pays** est une liste de choix (Belgique, France, Luxembourg). Obligatoire pour le membre, nous reprendrons celle-ci
7. **Prix** est la valeur du bien, elle peut-être omise le cas échéant.
8. **dateinsertion** est directement reprise par le système
9. **Téléphone**, obligatoire dans la fiche du membre est repris dans l'annonce mais peut-être masqué par l'utilisateur (TEL)
10. **TEL** permet à l'utilisateur d'afficher ou non son numéro de téléphone
11. **Mail** est repris de la fiche du membre mais pourra être supprimé puisqu'il apparaîtra dans la fiche du membre
12. **uid-cat** est un choix au moment de la création de l'annonce avec la méthode du chapitre précédant par liste déroulante
13. **uid-util** est le numéro de l'utilisateur qui dépose l'annonce.

L'ensemble ne pose pas trop de problèmes, seul le téléchargement de l'image n'as pas encore été étudié. Il sera alors sauvegardé directement sur le serveur. Cette solution, intéressante, nécessite quand même quelques précautions, notamment au niveau de la taille du fichier: pour l'espace disque mais aussi pour la sécurité du système. La commande HTML de téléchargement ne limite pas les fichiers au format images. Un internaute mal-honnête pourrais également insérer un virus ou même un script php. La majeur partie sera les tests de téléchargements (type de fichier, taille, ...). La programmation du téléchargement d'image sera le chapitre suivant.

18. Insérer une image dans une table MySQL avec PHP

Pour notre site de petites annonces, une petite photo peut être insérée avec chaque entrée. Cette partie va permettre à un utilisateur de télécharger une image directement sur le serveur, en fait n'importe quel fichier mais nous n'autoriserons que les images par sécurité. Nous allons en plus redimensionner l'image téléchargée et en créer une deuxième de format plus petit. La première sera affichée avec l'annonce, l'image réduite sera utilisée pour les listes. Pour enregistrer une image dans une base de donnée MySQL, deux solutions sont possibles:

- soit directement **enregistrer l'image dans un champ de la base de donnée**

- soit **enregistrer seulement l'adresse** et sauvegarder l'image dans un dossier spécifique.

La première solution va très vite alourdir la table et augmenter les temps d'accès à la base de donnée. Nous utiliserons la deuxième solution. Permettre de télécharger un fichier a des risques, notamment si le fichier téléchargé est un programme php ou un fichier exécutable (virus). Nous devrons obligatoirement vérifier si le fichier est bien une photo. En pratique, ceci ne pose pas de problèmes puisque nous allons également modifier la taille de la photo insérée lors du traitement, et même créer des miniatures.

Examinons le formulaire suivant:

```
<FORM ACTION="insertion.php" method="POST" ENCTYPE="multipart/form-data">
<input type="hidden" name=\"max_file_size" value="50000">
image:<input TYPE="file" NAME="image"><br>
<INPUT TYPE="submit" NAME="telecharger" VALUE="envoyer">
</form>
```

Une large partie est connue, je n'analyse que les nouvelles:

- **ENCTYPE** ne peut être utilisé qu'avec ma méthode POST, il spécifie l'encodage utilisé pour la forme que prendra le contenu du formulaire. les valeurs possibles de la commande ENCTYPE sont:
    - ENCTYPE="application/x-www-form-urlencoded" valeur par défaut; encode le contenu du formulaire selon une forme URL, difficilement lisible par le destinataire;
    - **ENCTYPE="text/plain"** le contenu du formulaire sera retourné en format texte lisible par le destinataire, généralement utilisé avec ACTION=mailto:
    - **ENCTYPE="multipart/form-data"** permet d'expédier un fichier attaché au message.

      Comme nous souhaitons transférer un fichier, c'est l'option **"multipart/form-data" qui sera utilisé**
- **type="hidden"**: permet de masquer (hidden) un paramètre
- Il est suivit de **name=\"max_file_size" value="50000"** qui spécifie la taille maximum du fichier à télécharger. C'est une variable cachée qui est transférée. Elle ne bloque pas le transfert si la taille du fichier est supérieure. Elle **peut être omise** si vous ne récupérez pas la variable dans les tests (par $_POST['max_file_size']).
- <input TYPE="file" NAME="image">: spécifie que le contenu est un fichier (à ce niveau, le type de fichier n'est pas vérifié.

Le résultat devrais nous donner ceci:

image:

envoyer

Voyons maintenant la procédure.

- Ce formulaire va télécharger le fichier sur le serveur WEB sous forme temporaire.
- Une fois enregistré, PHP initialise la variable $_FILES. Si à la fin de l'exécution le fichier temporaire na pas été déplacé dans le dossier "image", il est automatiquement effacé.

- Notre script vérifiera si le fichier est correct. Au moindre doute, l'image sera laissée telle qu'elle et supprimée automatiquement. La taille maximum d'un fichier est par défaut de 8 MB et est fixée directement sur le serveur donc pas modifiable.En cas de dépassement, un message d'erreur est affiché.

La variable transférée est fournie sous forme de tableau où (à partir de PHP 4):

- **$_FILES['image']['name']** est le nom du fichier original sur le PC de l'internaute.
- **$_FILES['image']['type']** est le type de fichier transféré, de type 'image/gif' par exemple
- **$_FILES['image']['size']** est la taille du fichier envoyé (en octet).
- **$_FILES['image']['tmp_name']** est le nom du fichier temporaire.

Si nous permettons de télécharger plusieurs fichiers images dans le même formulaire, le nombre de tableau sera équivalent: 1 tableau par variable (ici 'image' pour notre formulaire).

C'est la fonction **move_uploaded_file($_FILES['image']['tmp_name'],nom destination)** qui transfèrera le fichier s'il est valide du dossier temporaire vers un dossier de notre choix.

2. La commande preg_match

La première chose va être de vérifier le nom du fichier. Le nom du fichier inclut également son adresse, par exemple C:\document\logo.gif. Nous allons utiliser la fonction preg_match:

preg_match (string pattern, string subject [, array matches])

preg_match() analyse ***subject*** pour trouver l'expression dans ***pattern***. Le résultat envoyé est TRUE (valeur trouvée) ou False, notamment en cas d'erreur.

Si **[array matches]** est utilisé, il sera rempli par les résultats de la recherche. $matches[0] contiendra le texte qui satisfait le masque complet, $matches[1] contiendra le texte qui satisfait la première parenthèse capturante, ...

La grosse difficulté de cette commande est de créer la pattern. Essayons les scripts suivants:

```
<?php
$subject = "abcdef";
$pattern = '/dej/';
$matches=preg_match($pattern, $subject);
print($matches);
?>
```

Nous vérifions ici si dej est inclus dans la chaîne abcdef. La valeur est affichée est 0 (False). Si nous remplaçons dej par def, la valeur affichée sera 1 (True) puisque dej est bien inclus dans la chaîne.

Plusieurs options sont possibles comme:

- **\bchaîne\b**: dans l'exemple ci dessus, \bdef\b qui vérifie si def est inclus comme mot complet et pas inclus dans un mot
- **/i** ne fait pas de distinctions entre les minuscules et les majuscules. Dans l'exemple ci-dessus: $pattern = '/DEF/i' renverra 1 (True);
- l'utilisation des **parenthèse** et du caractère ? est également utilisé. Dans l'exemple ci-dessus: $pattern = '/(ab)?(ef)/i'; renverra vrai puisque la chaîne de caractère reprend ab et ef.
- **^c** oblige que le caractère soit en début de chaîne. Dans notre exemple: $pattern = '/^c/i'; renverra faux.
- **f$** oblige le caractère f en fin de chaîne.
- la mise **entre crochet** reprend un ensemble de valeur. Dans notre exemple: $pattern = '/[a-z]/i'; vérifiera si un des caractère entre a, b, ... z est inclus dans la chaîne. Par contre, $pattern = '/[0-9]/i'; renverra faux (pas de chiffres dans notre message) et $pattern = '/[a-z0-9]/i'; renverra vrai, remarquez que les 2 chaînes sont collées. $pattern = "/[^a-z0-9_\-.]/i";, plus complet vérifie également les caractères \, . , _ et . en début de chaîne
- Le | (pour la fonction OU) est également accepté. Dans notre exemple, $pattern = "/(.gif$)|(.jpg$)|(.png$)/i"; vérifie que la fin de la chaîne est .gif, .jpg ou .png (les formats d'images). $pattern = "/(^c)|(^d)|(^e)/i"; vérifie si l'adresse commence par la lettre c ou la lettre d ou la lettre e

L'utilisation des pattern est suffisamment complexe pour que nous en restions ici au niveau de ce cours en ligne. Dans notre téléchargement nous pourrions utiliser **$pattern = '/^[cdef].*.(gif|jpg|png)$/i'**; qui oblige une extension + de débuter la chaîne par c, d, e ou f mais ca pourrait poser des problèmes aux utilisateurs de Linux ou Mac. Nous n'utiliserons que:

```
<?php
$subject = "c:\Image.gif";
$pattern = '/(gif|jpg|png)$/i';
$matches=preg_match($pattern, $subject);
print($matches);
?>
```

qui n'accepte (renvoie TRUE) que pour les fichiers terminant par jpg, gif ou png à l'exception de tous autres (y compris image.gif.exe, un virus en perspective). Par contre, ça nous obligera à mettre une image par défaut au cas où l'utilisateur ne rentre pas de photo. Pour la liste de ces différentes syntaxes. Attention que c'est assez complexe.

Nous pourrions également utiliser une fonction conditionnelle **$_FILES['image']['type']**. Ces pattern seront également utilisé pour **l'Url Rewriting** dans le fichier .HTACCESS, quoique modifiées.

3. Vérification si le nom existe déjà

Autre problème, si le fichier existe déjà, la nouvelle image remplacera le fichier existant. Nous allons commencer par créer un dossier que nous appellerons images dans le dossier www d'EasyPhp. Ceci nous évitera de mélanger nos propres images de celles des annonces.

Pour vérifier si le fichier existe déjà dans notre dossier images, nous pouvons utiliser la fonction file_exists.

file_exists(dossier-image-serveur.nom_fichier)

Remarquez le point qui n'est pas nécessaire si le fichier est dans le même dossier. Par contre, nous l'utilisons pour modifier notre fonction en

file_exists("images/".$_files['image']['name'])

Une autre solution est d'utiliser comme nom d'image enregistrée sur le serveur code+nom, le code étant le uid (champ autoindexé) de chaque annonce. Dans ce cas la vérification si l'image existe ne sert plus à rien puisqu'elles seront toutes uniques.

4. Taille de l'image.

Même s'il votre configuration serveur est configuré pour limiter la taille des fichiers téléchargés 8MB par défaut), mieux vaut faire sa propre limitation, au moins pour réduire le temps de traitement.

Nous allons utiliser la commande:

filesize(nom_fichier) et la comparer par rapport à notre taille maximum acceptée.

Dans notre cas, ceci deviendra par exemple pour une limitation de 50 KB:

```
<?php
if (filesize($_FILES['image'][tmp_name]) >50000){
 echo 'image trop grande, limitée à 50 KB';
}
?>
```

ou en récupérant la variable cachée

```
<?php
$max=$_POST['max_file_size'];
if (filesize($_FILES['image'][tmp_name]) >$max){
  echo 'image trop grande, limitée à '.$max;
}
```

```
?>
```

5. Le fichier est-il une image?

Nous avons tester le type de fichier par une fonction getimagesize(). Cette fonction renvoie en fait un tableau. mais surtout **False** si l'image n'est pas reconnue ou l'accès au fichier impossible. Certaines images en JPG ne sont pas parfois pas reconnues sur les serveurs.

getimagesize (filename , *array imageinfo*)

La fonction gère les images de type GIF, JPG, PNG, SWF, PSP et BMP.

Le tableau est constitué de 4 éléments:

array(0): la longueur en pixels

array(1): la largeur en pixels

array(2): le type de l'image 1 = GIF ,2 = JPG , 3 = PNG , 5 = PSD , 6 = BMP , 7 = TIFF (Ordre des octets Intel), 8 = TIFF (Ordre des octets Motorola), 9 = JPC , 10 = JP2 , 11 = JPX , 12 = JB2, 13 = SWC, 14 = IFF.

array(3): la chaîne à placer dans les balises html ("height=xxx width=xxx").

La première vérification va être de vérifier si la taille de l'image n'est pas trop grande en largeur et en hauteur:

```
$taille=getimagesize("images/php-111.gif");
if ($taille[0]>300){
echo'Image trop large, maximum 300 pixels';
}
if ($taille[1]>400){
echo'Image trop haute, supérieure à 400 pixels';
}
```

6. Créer une image réduite

Les commandes suivantes utilisent la librairie GD inclue dans PHP. Cette librairie accepte les fichiers de type JPG, BMP et PNG pour toutes les versions de php. Par contre après avoir accepté les fichiers GIF, ils ont été enlevés, puis réintroduits. Cette fonction pourrait ne pas fonctionner sur tous les sites. De plus, son utilisation en local nécessite d'ajouter une extension qui **ne marche pas en local avec easyPhp 1.8.0.1 sur mon ordinateur,** mais bien sur l'hébergement.

La librairie inclut de nombreuses fonctionnalités, y compris de dessiner, fusionner des images, ... Nous ne verrons que redimensionner une photo.

Le but est de créer une image miniature en gardant l'image d'origine. Pour cela, nous utiliserons 2 fichiers distincts. Les commandes sont spécifiques au type d'image (uniquement JPG, GIF, PNG et BMP). Ca tombe bien, avec la commande **getimagesize** , tableau[2] nous avons déjà récupéré le type d'image. Comme en plus nous avons déjà interdit les autres types d'images par l'extension (au passage, y compris BMP un peu lourd à charger), il nous suffira de sélectionner la commande en fonction du type d'image retournée et de ne pas en créer sinon.

La procédure (l'image doit être au préalablement transférée sur l'hébergement

1. récupérer l'image en mémoire par la commande
   imagecreatefromgif(adresse de l'image) pour un fichier gif
   imagecreatefromjpeg(adresse de l'image) pour un fichier jpg
   imagecreatefrompnp(adresse de l'image) pour un fichier png
2. Nous créons une image vide aux dimensions voulues, 60 * 60 par exemple avec la commande **imagecreate(largeur en pixel,hauteur en pixel)**
3. nous redimensionnons l'image avec la commande **imagecopyresized(destination,image d'origine,coordonnée X de destination, coordonnée y de destination,coordonnée X de départ, coordonnée Y de départ, largeur finale, hauteur finale, largeur de départ, hauteur de départ)**. Si nécessaire, l'image sera étirée. Dans la majorité des cas, la coordonne de départ et d'arrivée est 0.
4. enregistrons l'image miniaturisé dans le fichier vide en respectant le format avec les commandes:
   imagejpeg (resource image [, string filename])
   imagegif (resource image [, string filename])
   imagepng (resource image [, string filename])

Reste à mettre cette procédure en pratique. Essayons ceci en utilisant l'image php-111.gif préalablement sauvegardée dans le même dossier que le programme.

```
<?php
$taille=getimagesize("php-111.gif");
if ($taille[0]>100){
echo'Image trop large, maximum 300 pixels';
}
if ($taille[1]>100){
echo'Image trop haute, supérieure à 400 pixels';
}
If ($taille[2]==1){
// ceci est une image GIF
$image1=imagecreatefromgif('php-111.gif');
$image2=imagecreate(60,60);
imagecopyresized($image2,$image1,0,0,0,0,60,60,$taille[0],$taille[1]);
```

```
imagegif($image2,"mini-php-111.gif");
}elseif ($taille[2]==2){
// ceci est une image JPG
$image1=imagecreatefromjpeg('php-111.jpg');
$image2=imagecreate(60,60);
imagecopyresized($image2,$image1,0,0,0,0,60,60,$taille[0],$taille[1]);
imagejpeg($image2,"mini-php-111.jpg");

}elseif ($taille[2]==3){
// ceci est une image png
$image1=imagecreatefrompng('php-111.png');
$image2=imagecreate(60,60);
imagecopyresized($image2,$image1,0,0,0,0,60,60,$taille[0],$taille[1]);
imagepng($image2,"mini-php-111.png");
}else{
echo'Format non accepté';
}
?>
```

L'image mini-php-111.gif est automatiquement créé même si cet exemple va déformer l'image puisque nous ne tenons pas compte des **proportions hauteur / largeur**. La solution passe par un paramètre de rapport comme ci-dessous:

```
// nous récupérons la hauteur / largeur de l'image téléchargée dont le nom est contenue dans $image=$_FILES['image']['name'].
// $rapport permet de redimensionner l'image en 350 de large. L'image a été préalablement transférée via la commande
move_uploaded_file($_FILES['image']['tmp_name'],$image)

$taille=getimagesize($image);
$rapport=round($taille[1]/$taille[0]*350,0);

// le transfert (ici dans le cas d'une image jpg) donne par exemple

elseif ($taille[2]==2)
$image1=imagecreatefromjpeg('$image);
$image2=imagecreatetruecolor(350,$rapport);
imagecopyresized($image2,$image1,0,0,0,0,350,$rapport,$taille[0],$taille[1]);
imagejpeg($image2,$ref."-".$image);
```

L'étape suivante va être de mettre le formulaire en place (en auto-invocant) pour

permettre 19. Formulaire d'insertion d'annonces.

1 Introduction - 2. Le formulaire d'entrée - 3. Tests du fichier et requêtes d'entrées

Pas un chapitre agréable puisque finalement nous avons déjà quasiment tout vu, juste (presque) des lignes de code en PHP et des requêtes MySQL. Nous reprenons finalement l'ensemble des développements des 3 précédents chapitres: affichage de la liste des catégories sous forme de liste déroulante, la liste des champs à remplir dans la table et le téléchargement d'une image via un formulaire.

Attention que les tests d'image et le redimentionnement ne fonctionne pas sous EasyPhp (même en cochant l'option php_gd2). Il faudra probablement le tester directement sur un site en ligne avec une base de donnée.

2. Première partie, le formulaire d'entrée

Nous allons d'abord commencer par créer notre formulaire auto-vérifié. Attention, ce programme utilise la base de donnée MySQL, vous devez le testez en local avec easyphp.

```
<?php
require('includes/start.php');
$categorie_tab=array();
$requete="SELECT * FROM categorie";
$valeur=mysql_query($requete);
$i=0;
$ligne=0;
while ($tableau=mysql_fetch_array($valeur)){
// on récupère les données de la table
// print($ligne);
if($tableau['actif']<>"N"){
$categorie_tab[$i]['uid']=$tableau['uid'];
$categorie_tab[$i]['nom']=$tableau['nom'];
$categorie_tab[$i]['description']=$tableau['description'];
$categorie_tab[$i]['attachement']=$tableau['attachement'];
$i=$i+1;
}else{
// categorie non active
}
}
require('includes/stop.php');
// début du traitement des catégories - sous catégories dans le tableau
$ligne=$i;
$i=0;
```

```
while ($i<$ligne){
if ($categorie_tab[$i]['attachement']<>0){
// vérification des sous catégories 1: ce n'est pas une catégorie
$k=$categorie_tab[$i]['attachement'];
$nom=$categorie_tab[$i]['nom'];
// on récupère attachement et vérifie si uid est effectivement avec attachement à 0
$j=0;
$l=0;
//
while ($j<$ligne){
if(($categorie_tab[$j]['uid']==$k)&&($categorie_tab[$j]['attachement']=="0")){
$l=1;
$categorie_tab[$i]['complet']= $categorie_tab[$j]['complet'];
}
$j=$j+1;
}
// si $l = 1, il s'agit bien d'une sous-catégorie de niveau 1
if ($l==1){
$categorie_tab[$i]['cat1']=$k;
$categorie_tab[$i]['cat2']=$categorie_tab[$i]['uid'];
$categorie_tab[$i]['cat3']="0";
$categorie_tab[$i]['complet']= $categorie_tab[$i]['complet']."->".$categorie_tab[$i]['nom'];
}else{
// c'est une catégorie de niveau 2, il faut retrouver les 2 catégories supérieures
$j=0;
$l=0;
while ($j<$ligne){
if(($categorie_tab[$j]['uid']==$k)&&($categorie_tab[$j]['attachement']<>"0")){
$l=$j;
$j=$ligne;
}
$j=$j+1;
}

$categorie_tab[$i]['cat1']=$categorie_tab[$l]['attachement'];
$categorie_tab[$i]['cat2']=$categorie_tab[$l]['uid'];
$categorie_tab[$i]['cat3']=$categorie_tab[$i]['uid'];
// on recherche ensuite la catégorie
$m=0;
while ($m<$ligne){
if
(($categorie_tab[$m]['uid']==$categorie_tab[$i]['cat1'])&&($categorie_tab[$m]['attachement']==0)){
$categorie_tab[$i]['complet']=$categorie_tab[$m]['nom']."->".$categorie_tab[$l]['nom']."-
>".$categorie_tab[$i]['nom'];
```

```
$m=$ligne;
}
$m=$m+1;
}
}
// fin de vérification des catégories 1
}else{
// c'est une catégorie }else{
$categorie_tab[$i]['cat1']=$categorie_tab[$i]['uid'];
$categorie_tab[$i]['cat2']=0;
$categorie_tab[$i]['cat3']=0;
$categorie_tab[$i]['complet']=$categorie_tab[$i]['nom'];
}
$i=$i+1;
}
```

```
// test d'exécution du sommaire
if (isset($_POST['go'])){
$titre = $_POST['titre'];
$erreur_titre = "";
$description = $_POST['description'];
$erreur_description = "";
$ville = $_POST['ville'];
$erreur_ville = "";
$pays = $_POST['pays'];
$erreur_pays = "";
$prix = $_POST['prix'];
$erreur_prix = "";
$telephone = $_POST['telephone'];
$erreur_telephone = "";

$dateinsertion=date('Ymd');
if ($_POST['titre'] == ''){
$erreur_titre = "Entrez un nom<br>";
} elseif ($_POST['description'] == ''){
$erreur_description = "Entrez une description<br>";
} elseif ($_POST['ville'] == ''){
$erreur_ville = "Entrez une ville<br>";
} else {
$ajout = "";
echo"Connexion réussie";
// entrée des données dans la table
}
} else {
$nom = "Nom de la catégorie";
$erreur_titre = "";
```

```
$description = "Description de la catégorie";
$erreur_description = "";
$attachement = "Numéro de la catégorie supérieure";
$erreur_attachement = "";
$actif ="O";
$erreur_actif = "";
}
$form = "<form method=\"POST\" form action=\"$PHP_SELF\"
ENCTYPE=\"multipart/form-data\">
<p>Titre annonce <input type=\"text\" name=\"titre\" size=\"120\"><br><font
color=\"#FF0000\">$erreur_titre</font></p>
<p><select size=\"1\" name='uid'>";
$i=0;
while ($i<$ligne){
$uid= $categorie_tab[$i]['uid'];
$complet= $categorie_tab[$i]['complet'];
$form=$form."<option value=".$uid.">".$complet."</option>";
$i=$i+1;
}
```

// Formulaire

```
$form=$form."</select>
<p>Description de votre annonce</p>

<P><TEXTAREA NAME=\"description\" ROWS=\"8\" COLS=\"50\"
WRAP=virtual></TEXTAREA></p><p><font
color=\"#FF0000\">$erreur_description</font></P>
<p>Ville: <input type=\"text\" name=\"ville\" size=\"40\"><font
color=\"#FF0000\">$erreur_titre</font> Pays: <select name=\"pays\">
<option selected>Belgique</option>
<option>France</option>
<option>Luxembourg</option>
</select><P>
Téléchargez une Photo : <input TYPE=\"file\" NAME=\"photo\">
Prix : <input type=\"text\" name=\"prix\" size=\"13\"><P>
Téléphone : <input type=\"text\" name=\"telephone\" size=\"15\">
Tel. affiché: <input type=\"checkbox\" name=\"tel\" value=\"ON\" checked><P>
<input type=\"submit\" value=\"Envoyer\" name=\"go\">
</form>
<HR>";
print($form);
?>
```

La première partie récupère nos catégories sous forme d'un tableau comme vu précédemment. La deuxième partie est finalement la vérification de quelques données. Pour l'instant, nous ne vérifions pas grand chose (juste si le titre, la description et la ville sont

correct, nous ajouterons les vérifications à fait. De plus, nous ne remplissons pas encore les données provenant de l'utilisateur

La dernière partie affiche le formulaire. C'est simplement une adaptation de nos différentes parties de formulaires (liste déroulante et transfert d'image).

3. Tests du fichier et requêtes SQL

Insérons le test de chargement du fichier du chapitre précédant avec les modifications du formulaire. Pour cela, nous allons créer une fonction vérifiant l'extension de l'image (imageok()). D'autres vérifications sont faites sur la taille uniquement si le fichier est reconnu: quelques images JPG sont refusées.

De même si l'utilisateur ne rentre pas de photo, l'image par défaut logo-min.gif est chargé et aucun test n'est effectué.

L'entrée des données se fait en 2 parties: la partie 1 est standard

```
require('includes/start.php');
if($_FILES['photo']['name']<>"logo-min.gif"){
$photo=$_FILES['photo']['name'];
}else{
$photo="logo-min.gif";
}
$requete="INSERT annonce SET
titre='$titre',description='$description',photo='$photo',ville='$ville',pays='$pays',prix='$prix',dateinse
rtion='$dateinsertion',
telephone='$telephone', mail='$mail',uid_cat='$uid_cat',uid_util='$uid_util',tel='$tel'";
$valeur=mysql_query($requete);
```

Par contre, une fois l'annonce rentrée et si l'image n'est pas l'image par défaut, on récupère le numéro de l'annonce via la commande MySQL **mysql_insert_id()** qui récupère la dernière valeur entrée dans un champ auto_increment.

Une fois le code récupéré, on update le nom de la photo en code-nom_photo. Par exemple, si l'image téléchargée est maison.gif pour l'annonce 271, elle est enregistrée dans le dossier images sous le nom: 271-moison.gif. Il n'y a donc pas de risque de retrouver 2 fois la même photo et ça permet éventuellement de supprimer les vieilles photos plus facilement. De même, la miniature est codée sous la forme code-mini-nom_image: (**"images/".$code."-mini-".$_FILES['photo']['name']**)

```
$code=mysql_insert_id();
// on vient de récupérer le code de l'annonce et on uddate la photo par code-nom photo
if ($_FILES['photo']['name']<>"logo-min.gif"){
 // on modifie le nom du fichier image uniquement si l'image n'est pas celle par défaut
```

```
 $requete="update annonce set photo='$photo' where code='$code'";
 $valeur=mysql_query($requete);
 // transfert du fichier avec le code de l'annonce en début du nom.
 move_uploaded_file($_FILES['photo']['tmp_name'],"images/".$code."-
".$_FILES['photo']['name']);

chmod ("images/".$code."-".$_FILES['photo']['name'],0644);

 // fin de transfert du fichier
  // création de la miniature.
$taille=getimagesize("images/".$code."-".$_FILES['photo']['name']);
If ($taille[2]==1){
// ceci est une image GIF
$image1=imagecreatefromgif('images/'.$photo);
$image2=imagecreate(60,60);
imagecopyresized($image2,$image1,0,0,0,0,60,60,$taille[0],$taille[1]);
imagegif($image2,"images/".$code."-mini-".$_FILES['photo']['name']);
}elseif ($taille[2]==2){
// ceci est une image JPG
$image1=imagecreatefromjpeg('images/'.$photo);
$image2=imagecreate(60,60);
imagecopyresized($image2,$image1,0,0,0,0,60,60,$taille[0],$taille[1]);
imagejpeg($image2,"images/".$code."-mini-".$_FILES['photo']['name']);
}elseif ($taille[2]==3){
// ceci est une image png
$image1=imagecreatefrompng('images/'.$photo);
$image2=imagecreate(60,60);
imagecopyresized($image2,$image1,0,0,0,0,60,60,$taille[0],$taille[1]);
imagepng($image2,"images/".$code."-mini-".$_FILES['photo']['name']);
}else{
echo'Format non accepté pour miniaturiser';
}
}
```

Voici notre fichier définitif, il suffit juste de recoller la partie création de la catégorie. Remarque, **ceci le véritable fichier utilisé en production qui récupère également l'adresse IP du visiteur**.

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=windows-1252">
<meta http-equiv="Content-Language" content="fr">
<meta name="description" content="Entrez votre petite annonce, tests de développement">
<title>Entrer sa petite annonce</title>

</head>
<?php
```

```
require('includes/header.php');
require('includes/fonctions.php');
?>
<div align="center">
<table border="1" width="1000">
<tr>
<td width="20%" valign="top" bgcolor="#cccccc">
<?php
include('includes/colon-left.php');
?>
</td>
<td width="60%" valign="top">
<?php
// début affichage du formulaire
require('includes/start.php');
$categorie_tab=array();
$requete="SELECT * FROM categorie order by nom_complet";
$valeur=mysql_query($requete);
$i=0;
$ligne=0;
while($tableau=mysql_fetch_array($valeur)){
if($tableau['actif']<>"N"){
if($tableau['uid']<>$tableau['attachement'])
{
$categorie_tab[$i]['uid']=$tableau['uid'];
$categorie_tab[$i]['nom']=$tableau['nom'];
$categorie_tab[$i]['description']=$tableau['description'];
$categorie_tab[$i]['attachement']=$tableau['attachement'];
$categorie_tab[$i]['nom_complet']=$tableau['nom_complet'];
$i=$i+1;
}
}else{
// categorie non active
}
}
require('includes/stop.php');
// début du traitement des catégories - sous catégories dans le tableau
$ligne=$i;
$i=0;

// test d'execuction du sommaire
if (isset($HTTP_POST_VARS['go'])and $login<>""){
$ip=$_SERVER["REMOTE_ADDR"];
echo "Votre adresse IP est enregistrée: ".$ip."<br>";
```

```
$titre = ADDSLASHES($HTTP_POST_VARS['titre']);
$erreur_titre = "";
$description =ADDSLASHES($HTTP_POST_VARS['description']);
$erreur_description = "";
$ville = ADDSLASHES($HTTP_POST_VARS['ville']);
$erreur_ville = "";
$pays = ADDSLASHES($HTTP_POST_VARS['pays']);
$erreur_pays = "";
$prix = ADDSLASHES($HTTP_POST_VARS['prix']);
$erreur_prix = "";
$telephone = ADDSLASHES($HTTP_POST_VARS['telephone']);
$erreur_telephone = "";
$erreur_file = "";
$tel=$HTTP_POST_VARS['tel'];
$mois=$HTTP_POST_VARS['mois']*2678400;
$uid_cat=$HTTP_POST_VARS['uid'];
$dateinsertion=date('Ymd');
$dateexpiration= date('Ymd',mktime(date('m,d,Y'))+$mois);
// tests des mots interdits
$correct="O";
$mots_interdits=array();
$mots_interdits=motsinterdits();
$mots_remplacement=array();
$mots_remplacement=liste_remplacement();

// vérification mots interdits, suppression liens, images, remplacement caractères
$correct=inclusinterdits($titre,$mots_interdits,$correct);
$correct=inclusinterdits($description,$mots_interdits,$correct);
$correct=inclusinterdits($ville,$mots_interdits,$correct);
$correct=inclusinterdits($pays,$mots_interdits,$correct);
$correct=inclusinterdits($prix,$mots_interdits,$correct);
$correct=inclusinterdits($telephone,$mots_interdits,$correct);
$titre=modif_texte($titre,$mots_remplacement);
$description=modif_texte($description,$mots_remplacement);
$ville=modif_texte($ville,$mots_remplacement);
$pays=modif_texte($pays,$mots_remplacement);
$prix=modif_texte($prix,$mots_remplacement);
$telephone=modif_texte($telephone,$mots_remplacement);

if ($_FILES['photo']['name']==""){
// nécessaire si l'utilisateur ne met pas de photo, alors photo par défaut
$_FILES['photo']['name']="pas-photo.jpg";
}
$taille=getimagesize($_FILES['photo']['tmp_name']);
```

```
if ($HTTP_POST_VARS['titre'] == ''){
$erreur_titre = "Entrez un nom<br>";
} elseif ($HTTP_POST_VARS['description'] == ''){
$erreur_description = "Entrez une description<br>";
} elseif ($HTTP_POST_VARS['ville'] == ''){
$erreur_ville = "Entrez une ville<br>";
} elseif (!imageok($_FILES['photo']['name'])){
// image incorrecte
echo'Image incorrecte';
$erreur_file = "<br>Seuls sont acceptés les formats GIF, JPG et PNG";
}elseif($taille[0]>350){
$erreur_file ="<font color=\"#FF0000\">Image trop large, maximum 350 pixels</font>";
}elseif (($_FILES['photo']['tmp_name']=="")&&($_FILES['photo']['name']<>"pas-photo.jpg")){
$erreur_file ="<font color=\"#FF0000\">Photo non reconnue, essayez une autre photo, en GIF par
exemple</font><br>";
}elseif($taille[1]>400){
$erreur_file ="<font color=\"#FF0000\">Image trop haute, supérieure à 400 pixels</font>";
} else {
// on entre les données
$rapport=round($taille[1]/$taille[0]*120,0);
require('includes/start.php');
if($_FILES['photo']['name']<>"pas-photo.jpg"){

}else{
$photo= "pas-photo.jpg";
}
// insertion dans la base de donnée en fonction de $correct
if ($correct=="O")
{
$requete="INSERT contenu SET
titre='$titre',description='$description',photo='$photo',ville='$ville',pays='$pays',prix='$prix',
dateinsertion='$dateinsertion',dateexpiration='$dateexpiration',telephone='$telephone',mail='$mail'
,
uid_cat='$uid_cat',uid_util='$uid_util',tel='$tel',ip='$ip'";
}else{
$requete="INSERT invalide SET
titre='$titre',description='$description',photo='$photo',ville='$ville',pays='$pays',prix='$prix',dateinse
rtion='$dateinsertion',
dateexpiration='$dateexpiration',
telephone='$telephone',mail='$mail',uid_cat='$uid_cat',uid_util='$uid_util',tel='$tel',ip='$ip'";
}

$valeur=mysql_query($requete);
```

```
// Mise à jour du nombre d'annonce
if ($correct=="O")
{
$requete= "UPDATE categorie SET nb_annonce=nb_annonce+1 where uid='$uid_cat'";
$valeur=mysql_query($requete);
}
// -- on code la photo et on crée la miniature si ce n'est pas celle par défaut
if ($photo<>"pas-photo.jpg"){
// nécessaire si le nom du fichier contient un tiret
$photo=STR_replace("-","",$_FILES['photo']['name']);
$photo=STR_replace(" ","",$_FILES['photo']['name']);
$photo=STR_replace("é","e",$_FILES['photo']['name']);
$photo=STR_replace("è","e",$_FILES['photo']['name']);
$photo=STR_replace("à","a",$_FILES['photo']['name']);
$photo=STR_replace("ç","c",$_FILES['photo']['name']);
$requete="select * from contenu where titre ='$titre' and dateinsertion='$dateinsertion'";
$valeur=mysql_query($requete);
while ($tableau=mysql_fetch_array($valeur)){
// $photo=$tableau["code"]."-".$photo;
$code= $tableau["code"];
}
// insertion dans la base de donnée en fonction de $correct
if ($correct=="O")
{
$requete="update contenu set photo='$photo' where code='$code'";
}else{
$requete="update invalide set photo='$photo' where code='$code'";
}
$valeur=mysql_query($requete);
// transfert du fichier avec le code de l'annonce en début du nom.
move_uploaded_file($_FILES['photo']['tmp_name'],"image-annonce/".$code."-".$photo);
chmod ("image-annonce/".$code."-".$photo,0644);
// fin de transfert du fichier
// création de la miniature.
$taille=getimagesize("image-annonce/".$code."-".$photo);
If ($taille[2]==1){
// ceci est une image GIF
$image1=imagecreatefromgif('image-annonce/'.$code."-".$photo);
$image2=imagecreatetruecolor(120,$rapport);
imagecopyresized($image2,$image1,0,0,0,0,120,$rapport,$taille[0],$taille[1]);
/*
$image2=imagecreate(100,100);
imagecopyresized($image2,$image1,0,0,0,0,100,100,$taille[0],$taille[1]);
*/
```

```
imagegif($image2,"image-annonce/".$code."-mini-".$photo);
}elseif ($taille[2]==2){
// ceci est une image JPG
$image1=imagecreatefromjpeg('image-annonce/'.$code."-".$photo);
/*$image2=imagecreate(100,100);
imagecopyresized($image2,$image1,0,0,0,0,100,100,$taille[0],$taille[1]);
*/
$image2=imagecreatetruecolor(120,$rapport);
imagecopyresized($image2,$image1,0,0,0,0,120,$rapport,$taille[0],$taille[1]);
imagejpeg($image2,"image-annonce/".$code."-mini-".$photo);
}elseif ($taille[2]==3){
// ceci est une image png
$image1=imagecreatefrompng('image-annonce/'.$code."-".$photo);
/*
$image2=imagecreate(100,100);
imagecopyresized($image2,$image1,0,0,0,0,100,100,$taille[0],$taille[1]);
*/
$image2=imagecreatetruecolor(120,$rapport);
imagecopyresized($image2,$image1,0,0,0,0,120,$rapport,$taille[0],$taille[1]);
imagepng($image2,"image-annonce/".$code."-mini-".$photo);
}else{
echo 'Format non accepté pour miniaturiser';
}
}
// fin de la création de la miniature
print('<br><b>Votre annonce a bien été enregistrée<BR> Merci</b>');
die('<meta http-equiv="refresh" content="3; URL=index.php">');
// fin du programme de création d'une nouvelle annonce.
}
} else {
$erreur_file = "";
$erreur_titre = "";
$description = "Description de la catégorie";
$erreur_description = "";
$erreur_file = "";
$erreur_ville="";
}
if ($login=="")
{
echo"<h1>Vous ne pouvez pas insérer une activité, merci de vous <a href=\"connexion.php\">
connecter</a>
ou de vous <a href=\"inscription.php\">inscrire</a></h1>";
}else{
echo"<b>Bonjour ".$login."</b>: Vous pouvez entrer votre annonce.";
```

```
}
require('includes/start.php');
$categorie_tab=array();
$requete="SELECT * FROM cat3 where actif3='O' && level<>'1' order by nom_complet3";
$valeur=mysql_query($requete);
$i=0;
$ligne=0;
while($tableau=mysql_fetch_array($valeur)){
$categorie_tab[$i]['uid']=$tableau['uid_cat3'];
$categorie_tab[$i]['nom']=$tableau['nom_cat3'];
$categorie_tab[$i]['description']=$tableau['description'];
$categorie_tab[$i]['attachement']=$tableau['attachement3'];
$categorie_tab[$i]['nom_complet']=$tableau['nom_complet3'];
$i=$i+1;

}
require('includes/stop.php');
// début du traitement des catégories - sous catégories dans le tableau
$ligne=$i;
$i=0;
$form = "<form method=\"POST\" ENCTYPE=\"multipart/form-data\">
<p>Remarque: Les liens et adresses mail sont automatiquement désactivés, cases en gras
obligatoires.<p>
<p><b>Titre annonce</b> <input type=\"text\" name=\"titre\" size=\"50\"><br><font
color=\"#FF0000\">$erreur_titre</font></p>
<p><b>Catégorie</b> <select size=\"1\" name='uid'>";
$i=0;
while ($i<$ligne){
$uid= $categorie_tab[$i]['uid'];
$complet= $categorie_tab[$i]['nom_complet'];
$form=$form."<option>".$complet."</option>";

$i=$i+1;
}
$form=$form."</select>
<p><b>Description de votre annonce</b><br>

<TEXTAREA NAME=\"description\" ROWS=\"8\" COLS=\"50\"
WRAP=virtual></TEXTAREA></p><p><font color=\"#FF0000\">$erreur_description</font></P>
<p><b>Ville</b>: <input type=\"text\" name=\"ville\" size=\"30\"><font
color=\"#FF0000\">$erreur_ville</font> Pays: <select name=\"pays\">
<option selected>Belgique</option>
<option>France</option>
<option>Luxembourg</option>
```

```
</select>
<input type=\"hidden\" name=\"max_file_size\" value=\"400000\">Téléchargez une Photo (formats
JPG, GIF, PNG: taille 100 KB. max , 350 l. et 400 H. max.): <input TYPE=\"file\"
NAME=\"photo\"><font color=\"#FF0000\">".$erreur_file."</font>
Prix : <input type=\"text\" name=\"prix\" size=\"13\"><P>
Téléphone : <input type=\"text\" name=\"telephone\" size=\"15\"> Tel. affiché: <input
type=\"checkbox\" name=\"tel\" value=\"ON\" checked>
Mois affichés: <select size=\"1\" name=\"mois\">
<option selected>3</option>
<option>2</option>
<option>1</option>
</select></p>
<input type=\"submit\" value=\"Envoyer\" name=\"go\">
</form>";
print($form);
function imageok($image_ok){
// on teste d'abord l'extension du fichier
$subject =$image_ok ;
$pattern = '/(gif|jpg|png)$/i';
$matches=preg_match($pattern,$subject);
return $matches;
}
//Fin affichage entrée des données>
?>
</td>
<td width="20%" valign="top" bgcolor="#CEEFD1">
<?php
include('includes/colon-right-annonce.php');
?>
</td>
</tr>
</table>
</div>
<?php
// require('includes/stop.php');
//echo'Base de donnée fermée';
include('includes/footer1.php');
?>
</BODY>
</HTML>
```

Je vous avait prévenu en début de chapitre, cette partie est assez difficile et reprend de nombreuses lignes de programmation mais elle est nécessaire. Elle sera remodifiée lorsque

nous utiliserons les codes utilisateurs pour rentrer les annonces. La partie suivante va nous permettre d'afficher l'annonce créée.

20. Affichage des annonces.

Dans le chapitre précédant, nous avons inséré les annonces. Même si les procédures ne sont pas complètes, elles nous permettent maintenant de travailler directement sur des annonces insérées dans la base de donnée.

Par facilité, nous allons créer 3 affichages différents de nos annonces:

1. un **affichage en liste** reprenant le titre et l'image miniature, dans les catégories ou en recherche.
2. un **affichage complet** sur une page reprenant toutes les informations de l'annonce.
3. un **affichage simple**, uniquement l'image miniature, le titre et la description. Cette méthode est utilisée dans de nombreux sites d'annonces. Dans notre cas, elle sera surtout utilisée plus tard comme bannières sur le site (genre dernières annonces postées) en renvoyant vers l'annonce effective par exemple

2. L'affichage en liste simple

Cette première partie va nous permettre d'afficher toutes les annonces reprises sur le site. Elle ne sera normalement pas utilisée pour les visiteurs standards mais nous utiliserons cette programmation dans le cas d'une recherche.

Comme d'habitude, nous allons travailler étape par étape.

1. la requête MySQL de sélection n'est pas très complexe, seule rajout, nous avons ajouté **order by 'dateinsertion' DESC**. Ceci va renvoyer les résultats par ordre décroissant sur la date d'insertion. Pour un ordre ascendant, DESC est remplacé par ASC.
2. Nous récupérons finalement les données sous forme standard, avec la variable $tableau. Quelques nouvelles commandes sont néanmoins utilisées. L'**affichage de l'image** se fait à l'aide d'une commande HTML récupérant le nom de l'image: <img border="0" src="images\<?php print($tableau["photo"]);?>"> .
3. La deuxième modification vient de l'affichage de la description
$description= $tableau['description'];
print("
".nl2br($description)."
");
On récupère d'abord la valeur sous forme d'une simple variable, plus nous l'affichons en utilisant la fonction **nl2br()**. Même si nous avons rencontré cette fonction, nous ne l'avons pas encore utilisée. Elle permet d'imprimer les sauts de lignes inclut dans la valeur de la table MySQL. Sans cette commande, PHP affiche tout le texte sur une même ligne.
4. En dernier, nous n'affichons le numéro de téléphone que si la case **tel** est effectivement oui.

5. La mise en page sommaire se contente d'insérer une ligne entre chaque annonces.

```
<?php
require('includes/start.php');
$requete="select * from annonce order by 'dateinsertion' DESC";
$valeur=mysql_query($requete);
$tableau=array();
while ($tableau=mysql_fetch_array($valeur)){
echo$tableau['code']."-".$tableau['titre']."-<br>";
print($tableau['uid-cat']);
?>
<p><img border="0" src="images/<?php print($tableau["photo"]);?>"></p>
<?php

$description= STRIPSLASHES($tableau['description']);

print("<br>".nl2br($description)."<br>");
print("- ".$tableau['dateinsertion']."<br>");
Print($description= $tableau['ville']." (".$tableau['pays']." )");
Print("Prix :".$tableau['Prix']);
if ($description= $tableau['tel']=="O"){
Print("<br>Téléphone: ".$tableau['telephone']);
}
print("<hr>");
}
require('includes/stop.php');
?>
```

Rien de bien sérieux ici mais quelques erreurs ou omissions sont incluses.

1. Un affichage correct utilise des tableaux ou une feuille de style
2. Nous affichons le numéro de la catégorie, son titre serait plus explicite.
3. L'image affichée est l'image normale, l'image miniature serait préférable.
4. Nous ne l'avons pas affiché mais un lien vers l'utilisateur serait préférable.

Nous aborderons cette partie en vérifiant les paramètres id de l'utilisateur dans un prochain chapitre.

MySQL est relationnel, mais à ce stade du tutorial, nous allons juste récupérer les variables par de simple recherche.

Commençons par **récupérer le titre de la catégorie**, nous connaissons déjà son numéro par $tableau['uid-cat']. Il nous suffit juste de créer une requête sur la table categorie pour récupérer la description. Par contre si, dans le cas d'une sous-catégorie, nous souhaitons récupérer toute la description, nous devons également récupérer la description des catégories supérieures. Ce développement a déjà été fait, il faut juste l'adapter. Nous utilisons une fonction appelée catégorie qui renvoie simplement la description. Remarquez que nous avons

laissé les valeurs du tableau catégorie. Ceci nous permettra de récupérer d'autres valeurs éventuellement.

La partie suivante est de **récupérer l'image sous forme de miniature**. Dans notre base de donnée, l'adresse de l'image est codée sous la forme code_annonce-nom_image. Par contre la miniature est codée sous le nom: code_annonce-mini-nom_image. J'avoue que ça aurait été plus facile si la miniature était codée sous la forme mini-code_annoncenom_image, il suffisait de supprimer "mini-". Nous allons utiliser les fonctions de recherche dans une chaîne vues au chapitre 6.

Nous devons d'abord rechercher les caractères "mini-" et les remplacer par rien. Cette partie utilise simplement la fonction **STR_replace("caractère à remplacer","caractère de remplacement'",$variable).**

La partie affichage de l'image <img border="0" src="images\<?php print($tableau["photo"]);?>"> devient: <img border="0" src="images\/<?php print(STR_replace("-","-mini-",$tableau["photo"]));?>"> .

Notre fichier à ce stade devient simplement:

```
<?php
 require('includes/start.php');
  require('includes/fonctions.php');
 $requete="select * from annonce order by 'dateinsertion' DESC";
 $valeur=mysql_query($requete);
 $tableau=array();
 while ($tableau=mysql_fetch_array($valeur)){
 echo$tableau['code']."-".$tableau['titre']."-<br>";
 print($tableau['uid_cat']);
 $categorie= categorie($tableau['uid_cat']);
 print("Catégorie: ".$categorie);
?>
<p><img border="0" src="images\/<?php print(STR_replace("-","-mini-
",$tableau["photo"]));?>"></p>
<?php
$description= $tableau['description'];
print(nl2br($description)."<br>");
print("- ".$tableau['dateinsertion']."<br>");
Print($description= $tableau['ville']." (".$description= $tableau['pays']." )");
Print("Prix :".$description= $tableau['Prix']);
if ($description= $tableau['tel']=="O"){
Print("<br>Téléphone: ".$description= $tableau['telephone']."<br>");
}
print("<hr>");
}
?>
```

Remarque: cette partie nécessite 2 logos dans le dossier images, un logo standard et une version réduite: logo-mini.gif

L'affichage sous forme de tableau va prendre ce lay-out:

<table>
<tr><td>Titre de l'annonce</td><td>description de la catégorie</td><td>code annonce</td></tr>
<tr><td rowspan="2">image miniature</td><td colspan="2">Description de l'annonce</td></tr>
<tr><td colspan="2">Ville (Pays)</td></tr>
<tr><td>Prix</td><td></td><td>Téléphone</td></tr>
<tr><td>Date insertion</td><td></td><td>utilisateur</td></tr>
</table>

Il va falloir simplement adapter l'affichage des données à partir d'ici:

```
<?php
require('includes/start.php');
require('includes/fonctions.php');
$requete="select * from annonce order by 'dateinsertion' DESC";
$valeur=mysql_query($requete);
$tableau=array();
while ($tableau=mysql_fetch_array($valeur)){
?>
<div align="center">
<center>
<table border="1" width="90%">
<tr>
<td width="33%"><b><font size="4"><?php Print($tableau['titre']);?></font></b></td>
<td width="33%"><b>
<?php
$categorie= categorie($tableau['uid_cat']);
print("Catégorie: ".$categorie);
?>
</b></td>
<td width="34%" align="right"><font color="#0000FF"><b><?php
echo$tableau['code'];?></b></font></td>
</tr>
<tr>
<td width="33%" rowspan="2"><img border="0" src="images\<?php print(STR_replace("-
","-mini-",$tableau["photo"]));?>">
</td>
<td width="67%" colspan="2">
```

```
<?php
$description= $tableau['description'];
print(nl2br($description)."<br>");
?>
</td>
</tr>
<tr>
<td width="67%" colspan="2" align="right"><?php Print($description= $tableau['ville']."
(".$description= $tableau['pays']." )"); ?>
</td>
</tr>
<tr>
<td width="33%"><b><?php Print("Prix :".$description= $tableau['prix']); ?></b></td>
<td width="33%"> </td>
<td width="34%" align="center"><b>
<?php
if ($description= $tableau['tel']=="O"){
Print("Téléphone: ".$description= $tableau['telephone']);
}
?>
</b></td>
</tr>
<tr>
<td width="33%" align="center"><b>
<?php
list($year, $month, $day) = explode("-", $tableau['dateinsertion']);
echo $day."/".$month."/".$year;
?>
</b></td>
<td width="33%"> </td>
<td width="34%" align="right"><b>
<?php
print($tableau['uid_util']);
?>
</b></td>
</tr>
</table>
</center>
</div>
<hr color="#0000FF">
<?php
}
?>
```

Les lignes

- list($year, $month, $day) = explode("-", $tableau['dateinsertion']);
- echo $day."/".$month."/".$year;

permettent d'afficher la date au format européens. Vous pouvez voire le résultat ici avec un en-tête ajouté.

Nous pouvons également modifier le tri en sélectionnant un tri multiple dans l'option ORDER BY de la requête MySQL SELECT, par exemple trié par catégorie, ensuite par date d'insertion.

```
$requete="select * from annonce order by 'uid_cat' DESC, 'dateinsertion' DESC";
```

3. Affichage complet

La deuxième partie va nous permettre d'afficher une page par annonce. Nous pourrions utiliser une construction similaire à celle ci-dessus mais au niveau référencement, ce serait irréaliste. La solution passe par une page récupérant le numéro de l'annonce, de type affichage-fiche?numéro où le numéro est en fait code de l'annonce. La méthode de transfert sera GET, le paramètre sera donc envoyé directement dans l'en-tête du fichier.

Modifions notre affichage-liste en remplaçant:

```
Print($tableau['titre']);
```

Par:

```
$code=$tableau['code'];
$titre=$tableau['titre'];
print("<a href=\"affichage-fiche.php?id=$code\">$titre</a>");
```

et

```
Print($tableau['code']);
```

Par:

```
print("<a href=\"affichage-fiche.php?id=$code\">$code</a>");
```

(la variable $code est déjà définie ci-dessus).

Pour comprendre cette partie, commençons par étudier le code html d'un lien:

```
<a href="adresse du lien">texte du lien</a>
```

Nous remplaçons finalement l'adresse du lien par: "<a href=\"affichage-fiche.php?id=$code\" où affichage-fiche.php est le nom du fichier qui reçoit le lien avec ?id=paramètre à transférer, dans notre cas, $code. Attention au caractères \ devant les guillemets. Le texte du lien est remplacé par le titre de l'annonce ($titre) ou le code de l'annonce ($code).

L'affichage de la page de l'annonce donne:

```
<?php
 require('includes/start.php');
 // cette partie ouvre la base de donnée
 include('includes/header.php');
 require('includes/fonctions.php');
 ?>
 <div align="center">
 <table border="1" width="90%">
 <tr>
  <td width="20%" align="center">
   <?php
   include('includes/colon-left.php');
   ?>
  </td>
 <td width="60%" align="center">
  <?php
    // La fiche de la petite annonce.
   if (!isset($_GET['id'])){
     Print("Cette annonce n'existe pas");
  }else{
     $code = ($_GET['id']);
     $requete="select * from annonce where code='$code'";
     $valeur=mysql_query($requete);
     $tableau=array();
     while ($tableau=mysql_fetch_array($valeur)){
       ?>
      <font size="4"><?php Print($tableau['titre']."<br>");?></font>
      <?php
      $categorie= categorie($tableau['uid_cat']);
      print("<hr>Catégorie: <b>".$categorie."</b> Numéro: ");
       ?>
 </b><font color="#0000FF"><b><?php echo$tableau['code']."<br>";
```

```
?></b></font>
<img border="0" src="images\<?php print($tableau["photo"]);?>">
<?php
$description= $tableau['description'];
print("<br>".nl2br($description)."<br>");
Print("<br>".$tableau['ville']." (".$tableau['pays']." )");
if ($tableau['prix']==0){
print("<br>Prix: Non communiqué");
}else{
Print("<br><b>Prix :".$tableau['prix']."</b>");
}
if ($tableau['tel']=="O"){
  Print("<br>Téléphone: ".$tableau['telephone']);
}
list($year, $month, $day) = explode("-", $tableau['dateinsertion']);
echo "<br>".$day."/".$month."/".$year;
print("<br>".$tableau['uid_util']."</b>");
}
}
?>
</td>
<td width="20%" align="center">
<?php
include('includes/colon-right.php');
?>
</td>
</tr>
</table>
</div>
<?php
include('includes/footer.php');
?>
```

Vous pouvez tester cette partie par ici

4. Affichage bannière.

La méthode suivant va finalement reprendre l'affichage global de la page mais sous forme restreinte: titre, début de description, prix. Ce format sera dédié aux annonces à droite de nos pages, reprenant les dernières annonces en ligne par exemple.

```
<?php
require('includes/start.php');
// cette partie ouvre la base de donnée
```

```
require('includes/fonctions.php');
// La fiche de la petite annonce.
if (!isset($_GET['id'])){
Print("Cette annonce n'existe pas");
}else{
$code = ($_GET['id']);
$requete="select * from annonce where code='$code'";
$valeur=mysql_query($requete);
$tableau=array();
while ($tableau=mysql_fetch_array($valeur)){
$code=$tableau['code'];
$titre=$tableau['titre'];
print("<a href=\"affichage-fiche.php?id=$code\">$titre</a>");
?>
<p><img border="0" src="images\/<?php print(STR_replace("-","-mini-
",$tableau["photo"]));?>">
</p>
<?php
$description=substr($tableau['description'],0,30);
print("<br>".nl2br($description)."...<br>");

if ($tableau['prix']<>0){
Print("<br><b>Prix :".$tableau['prix']."</b>");
}
}
}
?>
```

Dans un chapitre suivant, nous verrons comment insérer les annonces dans notre colonne de droite de manière aléatoire.

5. Affichage des annonces d'une catégorie uniquement.

Cette dernière partie va permettre d'afficher toutes les annonces mais seulement d'une catégorie. Pour celà, nous allons utiliser la même méthode que pour l'affichage des annonces: une fonction GET qui envoie le numéro de la catégorie vers un fichier **affichage-catégorie.php**. Seule réelle différence, le numéro de la catégorie est sélectionné à l'aide du menu déroulant développée au chapitre 16.

Cette liste de choix sera insérée à gauche de nos page par exemple (donc dans le fichier includes\column-left.php).

Nous avons utilisé la formule suivante pour envoyer un lien de l'affichage-liste vers la fiche de chaque publicité suivant le code.

```
print("<a href=\"affichage-fiche.php?id=$code\">$code</a>");
```

Dans notre précédant développement, nous utilisions les lignes de codes suivantes insérées dans un formulaire auto-invoquant (en plus de la partie récupération des différentes catégories dans un tableau).

```
<table>
<form method=\"POST\">
<p><select size=\"1\" name='uid'>";
while ($i<$ligne){
$uid= $categorie_tab[$i]['uid'];
$complet= $categorie_tab[$i]['complet'];
$liste=$liste."<option value=".$uid.">".$complet."</option>";
$i=$i+1;
}
$liste=$liste."</select><p><input type=\"submit\" value=\"Envoyer\" name=\"go\"><input
type=\"reset\" value=\"Rétablir\" name=\"B2\"></p>
</form>
</table>
<HR>
```

Toute l'astuce va être d'utiliser cette fois la méthode GET et de transférer le résultat vers le fichier affichant les différentes annonces de la catégorie souhaitée.

Vous le voyez, peu de modifications finalement pour notre formulaire. La partie affichage ne pose pas de problèmes particulier. Le formulaire envoie directement au fichier PHP affichage catégories ci-dessous. Attention que l'affichage ne fonctionne que si vous passez une variable. Je laisse à chacun le soin d'adapter son affichage.

```
<?php
require('includes/start.php');
// cette partie ouvre la base de donnée
include('includes/header.php');
require('includes/fonctions.php');
?>
<div align="center">
<table border="1" width="90%">
<tr>
<td width="20%" align="center">
<?php
include('includes/colon-left.php');
?>
</td>
<td width="60%" align="center">
```

```
<?php
// La fiche de la petite annonce.
if (!isset($_GET['uid'])){
Print("Cette annonce n'existe pas");
}else{
$code = ($_GET['uid']);
$requete="select * from annonce where uid_cat='$code' order by 'dateinsertion' DESC";
$valeur=mysql_query($requete);
$tableau=array();
while ($tableau=mysql_fetch_array($valeur)){
?>

<div align="center">
<center>
<table border="1" width="100%">
<tr>
<td bgcolor="#FFFFCC">
<img border="0" src="images\<?php print(STR_replace("-","-mini-
",$tableau["photo"]));?>" align="right">
<font size="4">
<?php
$code=$tableau['code'];
$titre=$tableau['titre'];
print("<a href=\"affichage-fiche.php?id=$code\">$titre</a>");
?>
</font>
</td>
</tr>
</table>
</center>
</div>
<?php
$categorie= categorie($tableau['uid_cat']);
print("Catégorie: <b>".$categorie."</b> ");
if ($tableau['prix']==0){
print("Prix: Non communiqué");
}else{
Print("<b>Prix :".$tableau['prix']."</b>");
}
$description= $tableau['description'];
print("<br>".nl2br($description)."<br>");
Print("<br>".$tableau['ville']." (".$tableau['pays']." )");

if ($tableau['tel']=="O"){
```

```
Print("<br>Téléphone: ".$tableau['telephone']);
}
list($year, $month, $day) = explode("-", $tableau['dateinsertion']);
echo "<br>".$day."/".$month."/".$year;
print(" - Posteur: ".$tableau['uid_util']."</b>");
}
}
?>
</td>
<td width="20%" align="center">
<?php
include('includes/colon-right.php');
?>
</td>
</tr>
</table>
</div>
<?php
include('includes/footer.php');
?>
```

Un chapitre suivant de cette formation va nous permettre d'utiliser cette partie pour la recherche d'annonces. Nous utiliserons 2 niveaux de recherche, simple et avancées. Mais d'abord, nous allons ajouter quelques "précautions" dans l'insertion des annonces, notamment la mise à l'écart d'annonces reprenant des mots interdits, liens hypertextes, adresses mails, ..

21. Corrections des entrées d'annonces

Au chapitre 19, nous avons créé le formulaire pour insérer nos petites annonces. Si vous l'avez un peu testé, il a (normalement) fonctionné sans problèmes. Pourtant, nous ne pouvons pas le mettre sur un site "en production" tel quel. Aucune sécurité n'est associée aux données rentrées. Un utilisateur standard ne posera pas de problèmes, quelques spécimens, oui.

En premier, l'utilisation des **2 caractères réservés ' et "** dans les annonces va envoyer un messages d'erreur au niveau des requêtes MySQL. Cette solution va utiliser la fonction ADDSLASHES($variable) dans l'insertion des annonces et STRIPSLASHES($variable) dans l'affichage. Ces 2 fonctions vont être utilisés dans TOUS les champs utilisateurs.

En deuxième, une spécialité de quelques webmasters est de **spamsmer** les sites de ce type (forum, livres d'or et ... **sites de petites annonces**). Les annonces risquent vite de reprendre des publicités pour des médicaments, casinos ou autres, liens sauvages sans rapport. D'autres termes peuvent également apparaître comme des termes injurieux. Certaines

annonces peuvent également être vue comme de la calomnie vis à vis d'un site, d'une personne ou d'une marque. Au niveau juridique, le webmasters responsable du contenu de votre site.

1. Dans le premier cas, nous allons créer **une table reprenant les mots interdits**. Cette solution n'est pas parfaite puisque les spamsmeurs vont utiliser toutes les possibilités pour essayer d'insérer leurs mots clés, y compris avec des caractères de séparation entre les lettres des mots. A l'insertion des annonces, nous allons juste tester si ces mots sont repris. Si le mot apparaît, l'annonce est automatiquement supprimée.
2. Le deuxième cas est difficile à repérer en automatique. Selon certains termes, nous enverrons un mail à l'administrateur reprenant l'annonce pour vérification manuelle. Un texte de type : "Vend ordinateur de marque ZZZZ, la poubelle des PC" est un terme injurieux pour la firme ZZZZ. Par contre, "Poubelles contenance 1000 litres à vendre" est une annonce standard.

En troisième, les spamsmers vont insérer des liens dans 1 ou plusieurs case à remplir (y compris le prix ...). Un lien Internet en HTML est codé sous forme:

- **YBET informatique**** pour un lien texte texte
- **<a href=**"http://www.ybet.be"><img border="0" src="http://www.ybet.be/logo.gif" width="326" height="48" **alt="**Magasin informatique"><**/a**> pour un lien image (avec la balise ALT optionnelle ici)

Nous allons systématiquement supprimer ces liens en gardant le contenu. Ceci peut-être nécessaire pour certaines annonces. De toute façon, les liens sauvages reprennent dans la majorité des cas des termes ... de la liste interdite et les annonces seront automatiquement supprimées

En dernier, on va retrouver dans le corps du message des **adresses mail, même si elle est reprise dans la fiche de l'utilisateur**. Ces adresses "sauvages" sont généralement insérées par celui qui poste l'annonce par facilité. Par contre, ces adresses sont utilisées par d'autres robots pour envoyer des courriers indésirables. Deux solutions sont possibles, soit supprimer automatiquement l'adresse, soit remplacer le caractère @ par un autre mot.

Pour chaque cas, un mail va être envoyé à l'administrateur du site reprenant les coordonnées de celui qui poste. Ceci pour permettre par exemple de lui interdire manuellement de poster de nouvelles annonces.

L'ensemble de ces fonctions vont être reprises dans l'insertion d'annonces vues au chapitre 19. Par facilité, nous découperons cette partie.

2. Caractères réservés.

Dans entree-annonce, nous allons modifier

```
$titre = $_POST['titre'];
$erreur_titre = "";
$description = $_POST['description'];
$erreur_description = "";
```

```
$ville = $_POST['ville'];
$erreur_ville = "";
$pays = $_POST['pays'];
$erreur_pays = "";
$prix = $_POST['prix'];
$erreur_prix = "";
$ville = $_POST['telephone'];
$erreur_telephone = "";
$erreur_file = "";
$uid_cat=$_POST['uid_cat'];

$dateinsertion=date('Ymd');
```

par

```
$titre = ADDSLASHES($_POST['titre']);
$erreur_titre = "";
$description =ADDSLASHES($_POST['description']);
$erreur_description = "";
$ville = ADDSLASHES($_POST['ville']);
$erreur_ville = "";
$pays = ADDSLASHES($_POST['pays']);
$erreur_pays = "";
$prix = ADDSLASHES($_POST['prix']);
$erreur_prix = "";
$telephone = ADDSLASHES($_POST['telephone']);
$erreur_telephone = "";
$erreur_file = "";
$uid_cat=$_POST['uid'];
$dateinsertion=date('Ymd');
```

Nous avons simplement inséré la fonction ADDSLASHES pour chaque variable.

Dans la partie affichage liste, nous utiliserons la fonction inverse: STRIPSLASHES($variable)

```
<?php
require('includes/start.php');
$requete="select * from annonce order by 'dateinsertion' DESC";
$valeur=mysql_query($requete);
$tableau=array();
while ($tableau=mysql_fetch_array($valeur)){
echoSTRIPSLASHES($tableau['code'])."-".STRIPSLASHES($tableau['titre'])."-<br>";
print($tableau['uid-cat']);
?>
```

```
<p><img border="0" src="images\<?php print(STRIPSLASHES($tableau["photo"]));?>"></p>
<?php

$description= STRIPSLASHES($tableau['description']);

print("<br>".nl2br($description)."<br>");
print("- ".$tableau['dateinsertion']."<br>");
Print(STRIPSLASHES($tableau['ville'])." (".STRIPSLASHES($tableau['pays'])."
)");
Print("Prix :".STRIPSLASHES($tableau['Prix']));
if ($description= $tableau['tel']=="O"){
Print("<br>Téléphone: ".STRIPSLASHES($tableau['telephone']));
}
print("<hr>");
}
require('includes/stop.php');
?>
```

Cette partie du développement ne pose normalement pas de problèmes particuliers, nous avons simplement ajouter les fonctions sur les variables.

3. Administration Termes interdits

La première chose est de créer la partie administration pour cette liste de mots. Nous appellerons cette table "interdit". Nous allons utiliser simplement les notions vues au chapitre 14. Cette table ne reprend qu'un seul champ: interdit de type varchar(30).

3.1 Création de la table Interdit

La requête de création de table MYSQL reprend la commande **CREATE TABLE [IF NOT EXIST] nom_table (definition-colonne, Index)**. Le champ interdit est une clé primaire, la valeur nulle est interdite.

```
<?php
require('../includes/start.php');

$requete="CREATE TABLE if not exists interdit (interdit varchar(30) primary key NOT
NULL)";
$erreur=mysql_query($requete);
$erreur1=mysql_error();
print($erreur."<br>");
print($erreur1);
mysql_close();
?>
```

Nous aurions pu également utiliser les fichiers stop.php et start.php

3.2. Insertion des mots interdits.

Nous allons simplement créer un petit formulaire d'entrée comme exercice. Une version plus complète sera vue juste en-dessous

```
<form METHOD=\"POST\">
<p>Mot ou suite de mots interdits <input type="text" name="interdit" size="20"></p>

<p><input type="submit" value="Envoyer" name="go"></p>

</form>
```

Comme d'habitude, ce formulaire est auto-invocant. Le codage PHP complet de l'insertion devient:

```
<?php
$interdit =$_POST['interdit'];
$erreur_interdit = "";
//
if ($interdit == '')
{
$erreur_interdit = "La suite de mots interdits doit être renseignée<br>";
} else {
$interdit = ADDSLASHES($interdit);
require('../includes/start.php');
$requete="insert into interdit Set interdit='$interdit'";
$erreur=mysql_query($requete);
require('../includes/stop.php');
}
$form = "<table>
<form METHOD=\"POST\">
<p>Mot ou suite de mots interdits <input type=\"text\" name=\"interdit\"
size=\"30\"><font color=\"#FF0000\">$erreur_interdit</font></p>
<p><input type=\"submit\" value=\"Insérer\" name=\"go\"></p>
</table>
</form>";
print($form);
?>
```

Comme cette partie fait partie de l'administration, elle sera insérée dans un sous-dossier (typiquement admin) protégé. Ceci nous conduit à faire quelques simplification. La première, il n'y a pas de tests pour vérifier si l'utilisateur utilise le bouton "Insérer". Le deuxième est lié au dossier includes qui devient require('**..**/includes/stop.php'); Comme le champ interdit est une clé primaire, il n'est pas non plus nécessaire de tester si le mot existe déjà, il sera automatiquement supprimé de la liste par MySQL.

3.3. Afficher les mots interdits

Pour pouvoir vérifier les mots insérés, nous allons simplement afficher en liste les mots que nous avons insérés.

```
<?php
require('../includes/start.php');
$requete="select * from interdit";
$valeur=mysql_query($requete);
$tableau=array();
while ($tableau=mysql_fetch_array($valeur))
{
$interdit=STRIPSLASHES($tableau['interdit']);
print($interdit."<br>");
}
require('../includes/stop.php');
?>
```

Ce script ne fait qu'afficher les mots dans une liste, les uns après les autres. La partie suivante va les afficher dans un formulaire en liste déroulante.

3.4. Ajouter, modifier, supprimer les mots interdits.

Dans les 2 parties ci-dessus, nous avons découpé les fonctionnalités. Pourtant, le plus facile serait de rassembler toutes les fonctions. La modification ou la suppression d'un mot dans la liste utilise un formulaire pour rentrer le mot (ou la liste de mots) dans une liste déroulante qui va nous servir également à afficher les mots interdits.

Une deuxième liste de choix permet de supprimer, modifier ou insérer.

Une troisième zone de texte permet de modifier le choix sélectionner en 1 (cas modifier) ou d'insérer (cas Insérer).

Le plus facile reste d'afficher les valeurs déjà rentrer dans une liste déroulante. Ceci facilite la sélection. Commençons par créer la liste déroulante. Un formulaire standard est de type:

```
<form method=POST>

<p>Mots <select size=1 name=mot>

<option>liste déroulante</option>

<option>liste déroulante 2</option>

<option>...</option>
```

```
</select>

<select name=action>

<option>Insérer</option><option>Modifier</option><option>Effacer</option>

</select>

nouvelle valeur: <input type=text name=nouveau size=30>

<input type=submit value=Envoyer name=go></p></form>";
```

Ceci affiche:

Mots liste déroulante Insérer nouvelle valeur:

Envoyer

Ce type de développement a déjà été effectué pour les catégories. Dans le cadre de cette formation, nous allons le refaire. **Décomposons le travail**:

```
<?php
$mots_interdits=array();
$tableau=array();

// début du tests auto-invocant
if (isset($_POST['go']))
{

// variable reprenant les 3 paramètres du formulaire
$choix =$_POST['mot'];
$action=$_POST['action'];
$nouveau=ADDSLASHES($_POST['nouveau']);
// vérification du deuxième bouton
if ($action=="Effacer"){

// cas effacer, on efface le choix sélectionné dans la liste déroulante
require ('../includes/start.php');
$requete = "delete from interdit where interdit='$choix'";
$valeur = mysql_query($requete);
}elseif ($action=="Modifier"){

// Cas modifier: on remplace la valeur dans la liste déroulante par celle de la zone texte Nouvelle valeur
if ($_POST['nouveau']=="")
{

  // on teste si la valeur de remplacement est nulle. Dans ce cas aucune action
```

```
  Echo"Aucune valeur de remplacement sélectionnée, recommencez";
}else{

  // sinon on effectue le remplacement
  require('../includes/start.php');

  $requete="UPDATE interdit SET interdit='$nouveau' WHERE interdit='$choix'";
  $valeur=mysql_query($requete);
}

// fin de modifier
}elseif (($action=="Insérer")and($nouveau<>")){

// insertion d'une nouvelle valeur si elle n'est pas nulle
require('../includes/start.php');
$requete="insert into interdit Set interdit='$nouveau'";
$erreur=mysql_query($requete);
}
}

// fin de la boucle, début de récupération des mots
require('../includes/start.php');
$requete="SELECT * FROM interdit";
$valeur=mysql_query($requete);
$i=1;
$ligne=0;

while ($tableau=mysql_fetch_array($valeur)){
// on récupère les données de la table
// print($ligne);
$mots_interdits[$i]=STRIPSLASHES($tableau['interdit']);
$i=$i+1;
}
require('../includes/stop.php');
// Affichage du formulaire
$ligne=$i;
$i=1;
$formulaire="<form method=\"POST\"><p>Mots <select size=\"1\" name=\"mot\">";
while ($i<$ligne){
$formulaire=$formulaire."<option>".$mots_interdits[$i]."</option>";
$i=$i+1;
}
$formulaire=$formulaire."</select> <select
name=\"action\"><option>Insérer</option><option>Modifier</option><option>Effacer</opti
on></select> nouvelle valeur: <input type=\"text\" name=\"nouveau\" size=\"30\"><input
type=\"submit\" value=\"Envoyer\" name=\"go\"></p></form>";
print($formulaire);
?>
```

Quelques remarques, nous reprenons la liste des valeurs en fin de programme. Ceci permet à la liste d'être remise à jour à chaque fois. L'**utilisation du bouton Envoyer** est obligatoire

4. Tests et enregistrements des petites annonces avec mots interdits.

Nous pourrions simplement supprimer les annonces reprenant les mots interdits, mais nous allons les conserver. Nous avons 2 solutions:

- utiliser une autre table identique à la table annonce mais qui n'enregistre les annonces valides.
- mettre un paramètres dans la table annonce qui signale que ce message est invalide. A l'affichage, un test permet d'afficher uniquement les annonces valides.

La deuxième méthode est plus facile à mettre en oeuvre mais nécessite de revoir toute la programmation. De plus, elle permet d'afficher à quelques utilisateurs privilégiés également les tables non valides. Par contre, avec la première méthode. L'affichage, la suppression définitive, ... ne peut se faire que via le panneau d'administration.

Dans ce développement, nous allons créer une table équivalente que nous nommerons invalide. Voici le programme de création de la table. Il est parfaitement identique à celle de la table annonce (y compris avec les commandes ALTER pour rajouter les champs.

```
// création table contenu des annonces invalides

<?php

$requete="CREATE TABLE if not exists invalide (code int primary key NOT
NULL auto_increment,titre varchar(120) not null,description blob not null,photo
varchar(255), ville varchar(40),pays enum('Belgique','France','Luxembourg'),prix
decimal(8,2) not null,dateinsertion date,telephone varchar(15),mail
varchar(30),index(ville(10)))";
$erreur=mysql_query($requete);
$erreur1=mysql_error();
$requete="alter table invalide add uid_cat smallint(6) NOT NULL, add uid_util
int(10) NOT NULL, add tel char(1)";
$erreur=mysql_query($requete);

?>
```

Cette solution permettra de reprendre les mêmes requêtes MySQL, en changeant uniquement le nom de la table.

4.1. Vérification de la présence des mots interdits dans un texte

Pour vérifier si un mot (ou une suite de mots) est présent dans un texte, nous utilisons:

EREG($variable1,$variable2) recherche si la chaîne $variable1 est contenue dans $variable2, renvoie une valeur logique.

Cette fonction PHP a été utilisée comme exercice pour détecter un numéro de TVA belge. Dans notre cas, nous allons utiliser comme variable1 un ensemble de variables reprises dans une table. Ceci nécessite une boucle pour vérifier l'ensemble des valeurs interdites.

Dans la partie insertion des données du chapitre 19, aucune vérification n'est faite sur aucun champ (excepté pour la manipulation des photos). Nous allons simplement tester tous les champs avec les mots interdits. Dans la cas où un des mots de la liste est détecté, on va donner au paramètre (variable) $accepté la valeur"O". Si le mot est détecté dans un des champs, la valeur du paramètre devient "N". Au niveau de l'enregistrement, si le paramètre est OUI, il est enregistré dans la table "annonce". Dans le cas contraire, il est enregistré dans la table "invalide" préalablement créé. Charge à l'administrateur de supprimer cette annonce.

A la suite de cette partie du programme, nous allons insérer les différents tests

```
$titre = ADDSLASHES($_POST['titre']);
$erreur_titre = "";
$description =ADDSLASHES($_POST['description']);
...
$dateinsertion=date('Ymd');
```

Nous allons créer une fonction personnelle PHP comme vu au chapitre 4. Cette fonction demande en entrée un texte, en sortie, elle renvoie "O" si un des mot n'est pas inclus et "N" dans le cas contraire. Commençons par créer cette fonction personnelle. Un petit rappel, les fonctions personnelles sont de la forme:

```
function nom-fonction($argument1, $argument2) {
// code php
return $variable;
}
```

Les paramètres à envoyer à la fonction sont le textes de l'annonce tout simplement. Le paramètre de retour est simplement vrai ou faut. Nous allons décomposer le fonctionnement du programme par facilité. Commençons par un développement simple, sans nous occuper des mots interdits reprenant la table.

```
<?php
$texte="Cette phrase va être analysée pour détecter des mots interdits";
$correct=motinterdit($texte);
print($correct);
```

```
function motinterdit($texte){
$interdit="mots";
$correct="1";

if (EREG($interdit,$texte))
{
$correct="0";
}
return $correct;
}
?>
```

Dans ce petit programme, nous définissons le texte à tester au préalable via la variable $texte. La valeur $correct est déterminée par la fonction motinterdit. Celle-ci renvoie "1" si le mot est inclus dans $texte, "0" sinon. C'est justement ce que nous souhaitons. La seule difficulté va être de récupérer les mots à tester dans la table invalide et de faire le texte en boucle tant que tous les mots interdits ne sont pas testés.

Nous allons décomposer cette fonctions en 2: une partie pour reprendre la liste des mots sous forme de variable tableau (déjà faite) et une partie pour la détection. Cette méthode est logique puisque le tests va être fait sur plusieurs champs. Autant que la récupération se fasse une seule fois et les tests à chaque fois.

La partie récupération a déjà été développée.

```
require('../includes/start.php');
$requete="SELECT * FROM interdit";
$valeur=mysql_query($requete);
$i=1;
$ligne=0;

while ($tableau=mysql_fetch_array($valeur)){
// on récupère les données de la table
// print($ligne);
$mots_interdits[$i]=STRIPSLASHES($tableau['interdit']);
$i=$i+1;
}
require('../includes/stop.php');
```

Toute la difficulté va être d'envoyer et et de récupérer plusieurs variables via une fonction. Hors, si une fonction accepte plusieurs paramètres en entrée (avec un nombre fixé en plus), elle n'accepte pas de renvoyé plusieurs paramètres via la commande return. La solution passe par une matrice (une variable de type tableau). Essayons le petit programme ci-dessus.

```
<?php
```

```
$tableau1=array();
$tableau1[1]="rien";
$tableau1[2]="rien";
tests($tableau1);
// envoit tableau
$rien1=array();
$rien1=renvoit();
print("<br>".count($rien1)." valeurs récupérées de la fonction renvoit()");
function renvoit()
{
// cette fonction renvoie un tableau
$rien=array();
$rien[1]="1";
$rien[2]="2";
return $rien;
}
function tests($tableau)
// cette fonction reçoit une variable tableau et affiche le nombre de cellules.
{
print(count($tableau)." valeurs transférées à la fonction tests");
}
?>
```

Nous commençons par créer le tableau $tableau1 et nous l'envoyons dans la fonction tests. Celle ne fait que compter le nombre de cellule dans la variable transférée et afficher le nombre (count($tableau)). La ligne suivante définit la matrice $rien1 et récupère les données de la fonction renvoit(). Cette fonction ne fait finalement que de créer un tableau $rien à 2 entrées et le renvoi comme résultat de la fonction. C'est justement ce que nous voulons.

Dans le premier cas, nous souhaitons récupérer le nombre de mots interdits et les insérer dans une variable tableau (la fonction renvoit() dans l'exemple ci-dessus. Une fois les mots déterminés, nous devons renvoyer ces mots dans une fonction pour les tester (c'est la fonction tests() ci-dessus).

Le petit script en PHP ci-dessus récupère les variables de notre table MySQL via la fonction motsinterdits() et les affichent. Cette fonction doit être insérée dans le dossier include, ce qui explique que require('start.php'); ne tient pas compte du répertoire

```
<?php
$mots_interdits=array();
$mots_interdits=motsinterdits();
$ligne=count($mots_interdits);
print($ligne." lignes <br>");
$i=1;
while($i<=$ligne)
```

```
{
print($mots_interdits[$i]."<br>");
$i=$i+1;
}

function motsinterdits()
{
require('start.php');
$requete="SELECT * FROM interdit";
$valeur=mysql_query($requete);
$i=1;
while ($tableau=mysql_fetch_array($valeur)){
// on récupère les données de la table
$mots_interdits[$i]=STRIPSLASHES($tableau['interdit']);
$i=$i+1;
}
require('stop.php');
print("<br>termine <hr>");
return $mots_interdits;
}
```

La deuxième fonction va être de tester si un des mots existe dans une variable $texte via la fonction EREG() de l'exemple ci-dessus. Le petits programme ci dessus récupère les mots interdits via la fonction ci-dessus, assigne à $texte le texte à tester. Dans notre cas, nous lui avons défini la valeur "tests sur les mots interdits". Si la chaîne contient un des mots refusé, la fonction renvoie "N", sinon, "O" (valeurs correctes). Remarque si la liste contient une suite de mots, elle ne vérifie pas dans le désordre. Il faut donc mieux insérer dans la table des mots que des suites de mots.

```
<?php
$mots_interdits=array();
$mots_interdits=motsinterdits();
$ligne=count($mots_interdits);
print($ligne." lignes <br>");
$i=1;
while($i<=$ligne)
{
print($mots_interdits[$i]."<br>");
$i=$i+1;
}
$texte="tests sur les mots interdits";
$correct=inclusinterdits($texte,$mots_interdits);
print($correct);
function inclusinterdits($texte,$mots_interdits)
```

```
{
$i=1;
$ligne=count($mots_interdits);
$correct="O";
while ($i<=$ligne)
{
if (EREG($mots_interdits[$i],$texte))
{
$correct="N";
}
$i=$i+1;
}
return $correct;
}

function motsinterdits()
{
require('start.php');
$requete="SELECT * FROM interdit";
$valeur=mysql_query($requete);
$i=1;
while ($tableau=mysql_fetch_array($valeur)){
// on récupère les données de la table
$mots_interdits[$i]=STRIPSLASHES($tableau['interdit']);
$i=$i+1;
}
require('stop.php');
print("<br>termine <hr>");
return $mots_interdits;
}
```

Il ne nous reste plus qu'à **insérer ces 2 fonctions dans le fichiers fonctions.php** inclus dans le dossier includes, appelée avec le début des procédures.

4.2. Correction de notre programme d'insertion de petites annonces.

Pour éviter que les spamsmeurs ne détectent trop la méthode, rien ne préviendra que le message est mis de coté. Tous les tests standards sont donc effectuer. Nous allons simplement effectuer les tests ci-dessus à la récupérations des données via la formulaire et modifier la requête d'insertion si la variable $correct="N" pour l'insérer dans l'autre table. Nous pourrons alors comme administrateur vérifier ces annonces, les supprimer, les modifier ou même les renvoyer vers la table standard après vérification.

Reprenons notre programme d'entrée des annonces et insérons les lignes suivantes (en rouge).

```
<?php
require('includes/header.php');
require('includes/fonctions.php');
?>

<body vlink="#0000FF" bgcolor="#D8EAFA" topmargin="0" leftmargin="0">
<div align="center">
<table border="1" width="90%">
<tr>
<td width="20%" align="center">
<?php
include('includes/colon-left.php');
?>
</td>
<td width="60%">
<?php
// début affichage du formulaire
$uid_util="120";
require('includes/start.php');
$categorie_tab=array();
$requete="SELECT * FROM categorie";
$valeur=mysql_query($requete);
$i=0;
$ligne=0;
while($tableau=mysql_fetch_array($valeur)){
if($tableau['actif']<>"N"){
$categorie_tab[$i]['uid']=$tableau['uid'];
$categorie_tab[$i]['nom']=$tableau['nom'];
$categorie_tab[$i]['description']=$tableau['description'];
$categorie_tab[$i]['attachement']=$tableau['attachement'];
$i=$i+1;
}else{
// categorie non active
}
}
require('includes/stop.php');
// début du traitement des catégories - sous catégories dans le tableau
$ligne=$i;
$i=0;
while ($i<$ligne){
if ($categorie_tab[$i]['attachement']<>0){
// vérification des sous catégories 1: ce n'est pas une catégorie
$k=$categorie_tab[$i]['attachement'];
$nom=$categorie_tab[$i]['nom'];
```

```
// on récupère attachement et vérifie si uid est effectivement avec attachement à 0
$j=0;
$l=0;
//
while ($j<$ligne){
if(($categorie_tab[$j]['uid']==$k)&&($categorie_tab[$j]['attachement']=="0")){
$l=1;
$categorie_tab[$i]['complet']= $categorie_tab[$j]['complet'];
}
$j=$j+1;
}
// si $l = 1, il s'agit bien d'une sous-catégorie de niveau 1
if ($l==1){
$categorie_tab[$i]['cat1']=$k;
$categorie_tab[$i]['cat2']=$categorie_tab[$i]['uid'];
$categorie_tab[$i]['cat3']="0";
$categorie_tab[$i]['complet']= $categorie_tab[$i]['complet']."->".$categorie_tab[$i]['nom'];
}else{
// c'est une catégorie de niveau 2, il faut retrouver les 2 catégories supérieures
$j=0;
$l=0;
while ($j<$ligne){
if(($categorie_tab[$j]['uid']==$k)&&($categorie_tab[$j]['attachement']<>"0")){
$l=$j;
$j=$ligne;
}
$j=$j+1;
}

$categorie_tab[$i]['cat1']=$categorie_tab[$l]['attachement'];
$categorie_tab[$i]['cat2']=$categorie_tab[$l]['uid'];
$categorie_tab[$i]['cat3']=$categorie_tab[$i]['uid'];
// on recherche ensuite la catégorie
$m=0;
while ($m<$ligne){
if
(($categorie_tab[$m]['uid']==$categorie_tab[$i]['cat1'])&&($categorie_tab[$m]['attachement']==0)){
$categorie_tab[$i]['complet']=$categorie_tab[$m]['nom']."->".$categorie_tab[$l]['nom']."-
>".$categorie_tab[$i]['nom'];
$m=$ligne;
}
$m=$m+1;
}
}
```

```
// fin de vérification des catégories 1
}else{
// c'est une catégorie }else{
$categorie_tab[$i]['cat1']=$categorie_tab[$i]['uid'];
$categorie_tab[$i]['cat2']=0;
$categorie_tab[$i]['cat3']=0;
$categorie_tab[$i]['complet']=$categorie_tab[$i]['nom'];
}
$i=$i+1;
}
// test d'exécution du sommaire
if (isset($_POST['go'])){
$titre = ADDSLASHES($_POST['titre']);
$erreur_titre = "";
$description =ADDSLASHES($_POST['description']);
$erreur_description = "";
$ville = ADDSLASHES($_POST['ville']);
$erreur_ville = "";
$pays = ADDSLASHES($_POST['pays']);
$erreur_pays = "";
$prix = ADDSLASHES($_POST['prix']);
$erreur_prix = "";
$telephone = ADDSLASHES($_POST['telephone']);
$erreur_telephone = "";
$erreur_file = "";
$tel=$_POST['tel'];
// pour tests
$mail="ybet-skynet";
// pour tests :$uid_cat=$_POST['uid'];
$uid_cat="1";
$dateinsertion=date('Ymd');
// tests des mots interdits
$correct="O";
$mots_interdits=array();
$mots_interdits=motsinterdits();
$correct=inclusinterdits($titre,$mots_interdits,$correct);
$correct=inclusinterdits($description,$mots_interdits,$correct);
$correct=inclusinterdits($ville,$mots_interdits,$correct);
$correct=inclusinterdits($pays,$mots_interdits,$correct);
$correct=inclusinterdits($prix,$mots_interdits,$correct);
$correct=inclusinterdits($telephone,$mots_interdits,$correct);


if ($_FILES['photo']['name']==""){
```

```
// nécessaire si l'utilisateur ne met pas de photo, alors photo par défaut
$_FILES['photo']['name']="logo-min.gif";
}
// echo$_FILES['photo']['name']."-";
// echo"<br>".$_FILES['photo']['tmp_name']."<br>";
// vérification fichier
$taille=getimagesize($_FILES['photo']['tmp_name']);
// echo "<br>".$taille[0].'-'.$taille[1]."pixels";
if ($_POST['titre'] == ''){
$erreur_titre = "Entrez un nom<br>";
} elseif ($_POST['description'] == ''){
$erreur_description = "Entrez une description<br>";
} elseif ($_POST['ville'] == ''){
$erreur_ville = "Entrez une ville<br>";
} elseif (!imageok($_FILES['photo']['name'])){
  echo'Image incorrecte';
  $erreur_file = "<br>Seuls sont acceptés les formats GIF, JPG et PNG";
}elseif($taille[0]>300){
$erreur_file ="<font color=\"#FF0000\">Image trop large, maximum 300 pixels</font>";
}elseif (($_FILES['photo']['tmp_name']=="")&&($_FILES['photo']['name']<>"logo-min.gif")){
$erreur_file ="<font color=\"#FF0000\">Photo non reconnue, essayez une autre photo, en GIF par
exemple</font><br>";
}elseif($taille[1]>400){
$erreur_file ="<font color=\"#FF0000\">Image trop haute, supérieure à 400 pixels</font>";
} else {
// on entre les données
require('includes/start.php');
if($_FILES['photo']['name']<>"logo-min.gif"){
$photo=$valeur.$_FILES['photo']['name'];
}else{
$photo="logo-min.gif";
}
// insertion dans la base de donnée en fonction de $correct
if ($correct=="O")
{
$requete="INSERT contenu SET
titre='$titre',description='$description',photo='$photo',ville='$ville',pays='$pays',prix='$prix',dateinse
rtion='$dateinsertion',
telephone='$telephone',mail='$mail',uid_cat='$uid_cat',uid_util='$uid_util',tel='$tel'";
}else{
$requete="INSERT invalide SET
titre='$titre',description='$description',photo='$photo',ville='$ville',pays='$pays',prix='$prix',
dateinsertion='$dateinsertion',telephone='$telephone',mail='$mail',uid_cat='$uid_cat',uid_util='$uid
_util',tel='$tel'";
```

```
}

$valeur=mysql_query($requete);
// -- on code la photo et on crée la miniature si ce n'est pas celle par défaut
if ($_FILES['photo']['name']<>"logo-min.gif"){
$requete="select * from contenu where titre ='$titre' and dateinsertion='$dateinsertion'";
$valeur=mysql_query($requete);
while ($tableau=mysql_fetch_array($valeur)){
$photo=$tableau["code"]."-".$_FILES['photo']['name'];
$code= $tableau["code"];
}
// insertion dans la base de donnée en fonction de $correct
if ($correct=="O")
{
$requete="update contenu set photo='$photo' where code='$code'";
}else{
$requete="update invalide set photo='$photo' where code='$code'";
}
$valeur=mysql_query($requete);
// transfert du fichier avec le code de l'annonce en début du nom.
move_uploaded_file($_FILES['photo']['tmp_name'],"images/".$code."-".$_FILES['photo']['name']);
// fin de transfert du fichier
// création de la miniature.
$taille=getimagesize("images/".$code."-".$_FILES['photo']['name']);
If ($taille[2]==1){
// ceci est une image GIF
$image1=imagecreatefromgif('images/'.$photo);
$image2=imagecreate(60,60);
imagecopyresized($image2,$image1,0,0,0,0,60,60,$taille[0],$taille[1]);
imagegif($image2,"images/".$code."-mini-".$_FILES['photo']['name']);
}elseif ($taille[2]==2){
// ceci est une image JPG
$image1=imagecreatefromjpeg('images/'.$photo);
$image2=imagecreate(60,60);
imagecopyresized($image2,$image1,0,0,0,0,60,60,$taille[0],$taille[1]);
imagejpeg($image2,"images/".$code."-mini-".$_FILES['photo']['name']);
}elseif ($taille[2]==3){
// ceci est une image png
$image1=imagecreatefrompng('images/'.$photo);
$image2=imagecreate(60,60);
imagecopyresized($image2,$image1,0,0,0,0,60,60,$taille[0],$taille[1]);
imagepng($image2,"images/".$code."-mini-".$_FILES['photo']['name']);
}else{
echo'Format non accepté pour miniaturiser';
```

```
}
}
// fin de la création de la miniature
print('<br>Votre annonce a bien été enregistrée<BR> Merci');
// fin du programme de création d'une nouvelle annonce.
}
} else {
$erreur_file = "";
$erreur_titre = "";
$description = "Description de la catégorie";
$erreur_description = "";
$erreur_file = "";
$erreur_ville="";
}
$form = "<form method=\"POST\" ENCTYPE=\"multipart/form-data\">
<p>Titre annonce <input type=\"text\" name=\"titre\" size=\"120\"><br><font
color=\"#FF0000\">$erreur_titre</font></p>
<p><select size=\"1\" name='uid'>";
$i=0;
while ($i<$ligne){
$uid= $categorie_tab[$i]['uid'];
$complet= $categorie_tab[$i]['complet'];
$form=$form."<option value=".$uid.">".$complet."</option>";
$i=$i+1;
}
$form=$form."</select>
<p>Description de votre annonce</p>

<P><TEXTAREA NAME=\"description\" ROWS=\"8\" COLS=\"50\"
WRAP=virtual></TEXTAREA></p><p><font color=\"#FF0000\">
$erreur_description</font></P>
<p>Ville: <input type=\"text\" name=\"ville\" size=\"40\"><font
color=\"#FF0000\">$erreur_ville</font> Pays: <select name=\"pays\">
<option selected>Belgique</option>
<option>France</option>
<option>Luxembourg</option>
</select>
<P>Téléchargez une Photo : <input TYPE=\"file\" NAME=\"photo\"><font
color=\"#FF0000\">$erreur_file;</font>
Prix : <input type=\"text\" name=\"prix\" size=\"13\"><P>
Téléphone : <input type=\"text\" name=\"telephone\" size=\"15\"> Tel. affiché: <input
type=\"checkbox\" name=\"tel\" value=\"ON\" checked><P>
<input type=\"submit\" value=\"Envoyer\" name=\"go\">
</form>";
```

```
print($form);
function imageok($image_ok){
// on teste d'abord l'extension du fichier
$subject =$image_ok ;
$pattern = '/(gif|jpg|png)$/i';
$matches=preg_match($pattern,$subject);
return $matches;
}
//Fin affichage entrée des données>
?>
</td>
<td width="20%" align="center">
<?php
include('includes/colon-right.php');
?>
</td>
</tr>
</table>
</div>
<?php
// require('includes/stop.php');
//echo'Base de donnée fermée';
include('includes/footer.php');
?>
```

Nous devons également modifier la fonction inclusinterdits (dans fonctions.php). La valeur $correct précédente lui est directement envoyée.

```
function inclusinterdits($texte,$mots_interdits,$correct)
{
$i=1;
$ligne=count($mots_interdits);
while ($i<=$ligne)
{
if (EREG($mots_interdits[$i],$texte))
{
$correct="N";
}
PRINT($correct.$mots_interdits[$i]);
$i=$i+1;
}
return $correct;
}
```

```
?>
```

La modification ne fait qu'insérer les 2 fonctions à la fin de la récupération par post des variables pour chaque valeur entrée par l'utilisateur. La valeur $correct est mise à "O" en début de déclaration et l'envoie également vers la fonction. A chaque appel de la fonction, elle renvoie la valeur $correct, égale à celle rentrée s'il n'y a pas de problème, N si un mot interdit est détecté.

5. Conclusion

Nous venons déjà de bloquer certaines rentrées. Le chapitre suivant va tester les annonces au niveau des liens inclus (une autre spécialité des webmasters indélicats).

Une remarque des participants au cours: "De toute façon, l'utilisateur doit rentrer son adresse mail avant de poster une annonce". Effectivement, lorsque tout le développement sera fini, ce sera une des conditions pour poster une petite annonce. Malheureusement, ce n'est pas une sécurité. Si cette méthode permet de bloquer les logiciels automatiques, elle ne bloque pas le spams manuel (créer une adresse mail est relativement facile et par principe, je ne bloque pas les adresses mail gratuites). Et puis, un habitué peut toujours décidé de vendre un produit spécial sans but de spams.

A ce stade, je sépare en 2 parties. Le site dahut.be reprend la suite du développement sur cette page: Aide sur le développement petites annonces. YBET.be reprenant la suite de la formation PHP - MySQL

23. Exercice: gérer des news en PHP - MySQL

Cette va servir d'exercice pour les précédents chapitres. Nous allons simplement développer un système de gestions des news à insérer sur un site existant. une page reprendra l'affichage des dernières nouvelles sous forme d'un titre, d'une description et d'une date. Nous changeons de page pour l'affichage toutes les 10 news pour ne pas surcharger l'écran. Les données utilisent une base de donnée MySQL.

Le développement reprend:

1. La création de la table
2. Le formulaire d'entrée des nouvelles
3. Un formulaire de modification (suppression, modification, ...)
4. La page d'affichage
5. Un système de tri.

Pour faciliter le tri des nouvelles, nous allons utiliser également un champ catégorie qui permettra de différencier les types de nouvelles affichées: nous créerons une deuxième table reprenant les catégories et utiliserons une liste déroulante dans l'insertion ou pour le tri par catégories. Ceci permettre au final de filtrer les annonces sur une date (période) ou par types de nouvelles. Ce système pourrait également être utilisé pour créer un blog sur un site.

Toute les fichiers PHP reprenant la partie gestion sera insérée dans un sous dossier admin à protéger. Rien de bien difficile puisque la majorité de ces fonctionnalités a déjà été vu dans les chapitres précédents.

2. Les tables MySQL des news.

La table **news** va reprendre les champs suivants:

Nom	Type	Remarque	requête associée
uid	smallint(6)	auto-increment, clé primaire	**uid smallint(6) primary key NOT NULL auto_increment**
titre	varchar(100)	titre de votre nouvelle	titre varchar(100) Not Null
news	TEXT	texte limité à 65535 caractères	news blob Not Null
datenews	date	date du jour, rentrée automatiquement	datenews date
categorie	varchar(30)		categorie varchar (30) Not Null

La table **categorie_news** est on ne peut plus simple:

Nom	Type	Remarque	Requête associée
categorie	Varchar(30)		categorie varchar(30) primary key not null

La création automatique des tables ne va pas poser trop de problèmes. **fichier news-table.php**

```
<?php
```

```
/* remplacez dans les lignes ci-dessus:
localhost par le nom du serveur (fourni par l'hébergeur)
root par le nom utilisateur de la base de donnée (fourni par l'hébergeur)
password par le mot de passe (fourni par l'hébergeur)
ybet par le nom de la base de données (fourni par l'hébergeur)
*/
if(!mysql_connect('localhost','root','password')){
Echo'Connection Impossible';
exit();
}
Mysql_select_db('ybet');

// création des tables
$requete="CREATE TABLE if not exists news (uid smallint(6) primary key NOT NULL auto_increment,titre varchar(100) Not null,news TEXT not null, datenews date, categorie varchar(30))";
$resultat=mysql_query($requete);
$erreur=mysql_error();
print($erreur);
$requete="CREATE TABLE if not exists categorie_news (categorie varchar(30) primary key)";
$resultat=mysql_query($requete);
$erreur=mysql_error();
print($erreur);
mysql_close();
```

3. Fichiers includes.

Comme dans les précédents développements, nous utilisons 2 fichiers inclus dans un sous- dossier includes: start.php et stop.php.

start.php

```
<?php
/* remplacez dans les lignes ci-dessus:
localhost par le nom du serveur (fourni par l'hébergeur)
root par le nom utilisateur de la base de donnée (fourni par l'hébergeur)
password par le mot de passe (fourni par l'hébergeur)
ybet par le nom de la base de données (fourni par l'hébergeur)
*/
if(!mysql_connect('localhost','root','password')){
Echo'Connection Impossible';
exit();
}
Mysql_select_db('ybet');
```

```
?>
```

stop.php

```
mysql_close();
```

4. Gestion les catégories

La première partie va nous permettre de gérer les catégories, nous allons faire le plus simple possible.

4.1. Affichage des catégories.

C'est une simple requête de sélection, sans filtre, juste classée par ordre alphabétique sur le nom de la catégorie.

news-affichage-cat.php

```
<?php
require('../includes/start.php');
$requete="select * from categorie_news order by categorie asc";
$resultat=mysql_query($requete);
$valeur=mysql_query($requete);
$tableau=array();
if (mysql_num_rows($valeur)<>0)
{
while ($tableau=mysql_fetch_array($valeur)){
echo stripslashes($tableau['categorie'])."<br>";
}
}
?>
```

4.2. Insertion catégorie

Nous allons simplement créer un petit formulaire d'entrée. Comme le champ categorie est de type primary key, rentrer 2 fois le même mot est impossible (sauf majuscules / minuscules). De toute façon, nous affichons les catégories déjà créée au dessus.

Pour éviter 2 fichiers, nous allons le faire en auto-invocant, en utilisant la méthode POST.

news-insertion-cat.php

```
<?php
```

```
require('..\includes\start.php');
if (isset($_POST['B1'])){
$categorie=addslashes($_POST['categorie']);
$requete="insert categorie_news set categorie='$categorie'";
$resultat=mysql_query($requete);
}
require('news-affichage-cat.php');
$form="<form method=\"POST\">
<p><input type=\"text\" name=\"categorie\" size=\"30\"></p>
<p><input type=\"submit\" value=\"Insérer\" name=\"B1\"></p>
</form>";
echo$form;
?>
```

Remarque, en insérant l'affichage après le test, la nouvelle catégorie est également affichée.

4.3. Suppression de catégorie.

Pour pouvoir supprimer une catégorie, nous devons obligatoirement vérifier si aucune news n'utilise cette catégorie. Cette partie sera fait après l'insertion des news.

5. Gestion des news.

Pour pouvoir gérer les news, nous devons d'abord les afficher. Pourtant, dans la partie administration, nous allons les afficher simplement en liste, en tronquant le contenu en lui-même.

5.1. Affichage simple en liste

Dans cette partie, nous allons simplement afficher les news les unes à la suite de l'autre, sans trop s'occuper de la mise en page. Nous n'affichons également que les 20 premiers caractères de la description de la nouvelle.

```
<?php
require('.../includes/start.php');
$requete="select * from news order by uid asc";
$resultat= mysql_query($requete);
$tableau=array();
while ($tableau=mysql_fetch_array($resultat)){
echo $tableau['uid'].": ".stripslashes($tableau['titre'])."
".stripslashes(substr($tableau['news'],0,40))."... ".$tableau['datenews']."-
".$tableau['categorie']."<br>";
}
```

```
?>
```

5.2. Insertion de la news.

De nouveau, nous allons utiliser un formulaire auto-invocant: peu de difficulté, sauf récupérer les catégories de la table catégorie dans une table déroulante.

Le fichier **news-insertion.php**

```
<?php
require('..\includes\start.php');
// récupération des catégories
$requete="select * from categorie_news order by categorie asc";
$resultat=mysql_query($requete);
$valeur=mysql_query($requete);
$tableau=array();
$cat=array();
$i=0;
if (mysql_num_rows($valeur)<>0)
{
while ($tableau=mysql_fetch_array($valeur)){
$cat[$i]=$tableau['categorie']."<br>";
$i=$i=1;
}
}
$nb_cat=count($cat);
//echo $nb_cat;
$i=0;
if (isset($_POST['B1'])){
$titre=addslashes($_POST['titre']);
$news=nl2br(addslashes($_POST['news']));
$categorie=addslashes($_POST['categorie']);
$datenews=date('Ymd');
if ($titre<>"" and $news<>"")
{
$requete="insert news set
titre='$titre',news='$news',categorie='$categorie',datenews='$datenews'";
$entrees=mysql_query($requete);
echo"Votre news a bien été enregistrée";
die('<meta http-equiv="refresh" content="3; URL=news-insertion.php">');
}
}
require('news-affichage.php');
$form="<form method=\"POST\">
```

```
<p>Titre: <input type=\"text\" name=\"titre\" size=\"100\"></p>
<p>Le contenu: <textarea rows=\"2\" name=\"news\" cols=\"50\"></textarea></p>
<p><select size=\"1\" name=\"categorie\">";
while($i<$nb_cat)
{
$form=$form."<option>".$cat[$i]."</option>";
$i=$i+1;
}
$form=$form."</select></p><p><input type=\"submit\" value=\"Envoyer\"
name=\"B1\"></p>
</form>";
echo$form;
?>
```

La fonction DIE renvoie automatiquement vers une nouvelle insertion et permet de ne pas insérer plusieurs fois la même annonce automatiquement.

5.3. Suppression d'une news.

Nous allons simplement afficher les news et demander via un formulaire le numéro de la news (uid) à effacer.

```
<?php

if (isset($_POST['go']))
{
$uid=$_POST['numero'];
require('../includes/start.php');
$requete="DELETE From news where uid='$uid'";
$valeur=mysql_query($requete);
mysql_close();
}

require('news-affichage.php');
$form="<form method=\"POST\">
<p>Numéro de la news: <input type=\"text\" name=\"numero\" size=\"20\"></p>
<p><input type=\"submit\" value=\"Envoyer\" name=\"go\"></p>
</form>";
echo$form;
?>
```

5.4. Modification d'une news.

La procédure est équivalente à celle d'effacement, sauf qu'une fois la nouvelle sélectionnée, nous devons récupérer les données dans un formulaire pour pouvoir la modifiée. Nous utilisons 2 fichiers distincts. Le premier news-modification.php affiche les nouvelles et permet de sélectionner l'annonce à modifier. Le deuxième: news-modification1.php permet la modification en elle-même de la news. Les 2 fichiers utilisent un formulaire en méthode POST.

news-modification.php

```
<?php
require('news-affichage.php');
?>
<form name='modification' action='news-modification1.php' method='post'>
<input name='numero' type='text' size='20'><br>
<input name='go' type='submit' value='modifier'><br>
</form>
```

news-modification1.php

```
<?php
if (isset($_POST['go']))
{
$uid=$_POST['numero'];
include('../includes/start.php');
$requete="select * from news where uid='$uid'";
$resultat=mysql_query($requete);
// récupération des anciennes valeurs
while ($tableau=mysql_fetch_array($resultat))
{
$titre=stripslashes($tableau['titre']);
echo$titre;
$news=stripslashes($tableau['news']);
$categorie=stripslashes($tableau['categorie']);
echo$categorie;
//
}
// récupération des catégories
$requete="select * from categorie_news order by categorie asc";
$resultat=mysql_query($requete);
$valeur=mysql_query($requete);
$tableau=array();
$cat=array();
$i=0;
```

```
if (mysql_num_rows($valeur)<>0)
{
while ($tableau=mysql_fetch_array($valeur)){
$cat[$i]=$tableau['categorie']."<br>";
$i=$i=1;
}
$nb_cat=count($cat);
//echo $nb_cat;
}
}
if (isset($_POST['B1']))
{
$titre1=addslashes($_POST['titre']);
$news1=nl2br(addslashes($_POST['news']));
$categorie1=addslashes($_POST['categorie']);
$uid=$_POST['uid'];
if ($titre1<>"" and $news1<>"")
{
require('../includes/start.php');
$requete="update news set titre='$titre1',news='$news1',categorie='$categorie1' where
uid='$uid'";
$valeur=mysql_query($requete);
echo"News modifée";
die('<meta http-equiv="refresh" content="3; URL=news-modification.php">');

}else{
echo"vous ne pouvez pas rentrer de valeurs nulles";
die('<meta http-equiv="refresh" content="3; URL=news-modification.php">');
}
}
$i=0;
$form="<form method=\"POST\">
<p>Titre: <input type=\"text\" name=\"titre\" size=\"100\" value=\"".$titre."\"></p>
<p>Le contenu: <textarea name=\"news\" rows=\"2\"
cols=\"50\">".$news."</textarea></p>
<input type=\"hidden\" name=\"uid\" value=\"".$uid."\">
<p><select size=\"1\" name=\"categorie\">
<p><option selected>".$categorie;
while($i<$nb_cat)
{
$form=$form."<option>".$cat[$i]."</option>";
$i=$i+1;
}
$form=$form."</select></p><p><input type=\"submit\" value=\"Envoyer\"
```

```
name=\"B1\"></p>
</form>";
echo$form;
?>
```

6. Suppression des catégories.

Nous allons développer le suppression des catégories. Cette partie n'a rien de bien compliqué, nous devons juste vérifié si aucune news ne reprend comme catégorie celle que nous souhaitons supprimer.

```
<?php
require('..\includes\start.php');
if (isset($_POST['B1'])){
$categorie=addslashes($_POST['categorie']);
// vérification dans la table news si cette catégorie est inutilisée
$requete="select * from news where categorie='$categorie'";
$valeur=mysql_query($requete);
$i=mysql_num_rows($valeur);
if ($i<>0)
{
// catégorie utilisée
echo"<b>Cette catégorie est utilisée dans les news et ne peut être effacée</b>";
die('<meta http-equiv="refresh" content="3; URL=news-supprimer-cat.php">');

}else{
$requete="delete from categorie_news where categorie='$categorie'";
$resultat=mysql_query($requete);
echo "<b>".mysql_affected_rows()." catégorie supprimée</b>";
die('<meta http-equiv="refresh" content="3; URL=news-supprimer-cat.php">');
}
}
require('news-affichage-cat.php');
$form="<form method=\"POST\">
<p><input type=\"text\" name=\"categorie\" size=\"30\"></p>
<p><input type=\"submit\" value=\"Supprimer\" name=\"B1\"></p>
</form>";
echo$form;
?>
```

7. Affichage complet des news

Nous allons simplement mettre en page les nouvelles insérées dans la table. Remarquez que dans l'insertion des news, nous n'avons strictement fait aucune modification des données entrées (à part ajouter les caractères de contrôle spécifiques au php). Vous pouvez donc insérer du code HTLM dans les news sans problèmes (lien, images, ...).

L'affichage va simplement reprendre dans un tableau la liste classée par date d'insertion descendante, de la dernière à la première). Commençons par créer l'affichage des nouvelles. Dans cette partie, nous ne découpons pas les résultats, elles seront toutes affichées.

```
<?php
// affichages des nouvelles sans découpage.
require('includes/start.php');
$requete="select * from news order by datenews desc";
$resultat=mysql_query($requete);
while ($tableau=mysql_fetch_array($resultat)){
echo "<h2><font color=\"#0000FF\">".stripslashes($tableau['titre'])."</font></h2>";
echo"<p>".stripslashes($tableau['news'])."</p>";
echo"<p>".$tableau['datenews']." - ".$tableau['categorie']."</p>";
echo"<hr>";
}
?>
```

Ce développement ne fait que d'afficher toutes les nouvelles les unes à la suite de l'autre. Seule une petite mise en page est utilisée (titre 1 en gras et en bleu pour le titre). Il est nettement perfectible. La première chose est de limiter l'affichage à 10 news (ou un nombre à déterminer). La deuxième partie va utiliser une feuille de style, plus facile à modifier pour un débutant.

8. Affichage par tranches

L'affichage est finalement identique à ci-dessus. Nous allons juste utiliser l'option LIMIT dans la requête de sélection. Ceci va également créer différentes pages avec un paramètre dans l'URL en remplaçant par les lignes ci-dessus, nous n'affichons que les 10 premières lignes de news:

```
    $i=0;

    $l=10;

    $requete="select * from news order by datenews desc limit $i,$l";
```

La première difficulté est de déterminer le nombre de pages totales de news. Nous allons créer 2 requêtes sur la table. La première va simplement déterminer le nombre de lignes de résultats et permettre de créer les différentes pages. La deuxième va afficher les résultats par tranches de 10 nouvelles. Le passage du numéro de page se fait à l'aide de la méthode GET. L'URL sera de type news-affichage.php?page=1, ...

Le fichier **news-affichage.php**

```
<?php
// affichages des nouvelles sans découpage.
require('includes/start.php');
if (!isset($HTTP_GET_VARS['page']))
{
$i=0;
$l=10;
}else{
$i=10*$HTTP_GET_VARS['page'];
$l=10;
}
$requete="select * from news order by datenews";
$resultat=mysql_query($requete);
$r=mysql_num_rows($resultat);
$t=round($r/10)+1;
// fin de récupération du nombre de résultat: $r est le nombre de nouvelles, $t le nombre
de pages.
$requete="select * from news order by datenews desc limit $i,$l";
$resultat=mysql_query($requete);
while ($tableau=mysql_fetch_array($resultat)){
echo "<h2><font color=\"#0000FF\">".stripslashes($tableau['titre'])."</font></h2>";
echo"<p>".stripslashes($tableau['news'])."</p>";
echo"<p>".$tableau['datenews']." - ".$tableau['categorie']."</p>";
echo"<hr>";
}
$j=1;
echo"<a href=\"news-affichage-lim.php\">Dernières news</a>";
while ($j<$t)
{
echo" - <a href=\"news-affichage-lim.php?page=".$j."\">page".$j."</a></p>";
$j=$j+1;
}
?>
```

Les liens pour passer d'une page à l'autre sont insérés en-dessous. Remarquez que la page d'entrées (les 10 dernières news) n'utilise pas de paramètres, même si l'adresse news-affichage.php?page=0 fonctionne (ceci évite le duplicate content pour les moteurs de recherche, néfaste au référencement des pages).

9. Affichage par catégories

Nous allons permettre l'affichage des news suivant certains filtres, par exemple par date, par catégories, ... Nous développerons également un petit moteur de recherche dans la partie suivante.

Relativement simple, cette partie va juste demander la catégorie dans un petit formulaire, faire passer le paramètre par la méthode GET et le transmettre au fichier news-cat.php.

Le formulaire à insérer dans vos pages (il est repris dans le fichier news-form-cat.php):

```
<?php
// récupération des catégories
require('includes/start.php');
$requete="select * from categorie_news order by categorie asc";
$resultat=mysql_query($requete);
$valeur=mysql_query($requete);
$tableau=array();
$cat=array();
$i=0;
if (mysql_num_rows($valeur)<>0)
{
while ($tableau=mysql_fetch_array($valeur)){
$cat[$i]=$tableau['categorie']."<br>";
$i=$i+1;
}
$nb_cat=count($cat);
$i=0;
$form="<form method=\"GET\" action=\"news-categorie.php\">
<p><select size=\"1\" name=\"categorie\">";
while($i<$nb_cat)
{
$form=$form."<option>".$cat[$i]."</option>";
$i=$i+1;
}
$form=$form."</select></p><p><input type=\"submit\" value=\"Envoyer\"></p>
</form>";
echo$form;
}
?>
```

Le fichier **news-categorie.php**

```
<?php
// affichages des nouvelles par catégorie
```

```
require('includes/start.php');
if (!isset($HTTP_GET_VARS['categorie']))
{
//pas de catégories sélectionnées
die('<meta http-equiv="refresh" content="0; URL=news-affichage.php">');
}else{
$categorie=$HTTP_GET_VARS['categorie'];
echo$categorie;
$requete="select * from news where categorie='$categorie'";
$resultat=mysql_query($requete);
$r=mysql_num_rows($resultat);
$t=round(($r-1)/10,0);
echo$t."<br>";
// fin de récupération du nombre de résultat: $r est le nombre de nouvelles, $t le nombre
de pages.
if (!isset($HTTP_GET_VARS['page']))
{
$i=0;
$l=10;
}else{
$i=10*$HTTP_GET_VARS['page'];
$l=10;
}
$requete="select * from news where categorie='$categorie' order by datenews desc limit
$i,$l";
$resultat=mysql_query($requete);
}

while ($tableau=mysql_fetch_array($resultat)){
echo "<h2><font color=\"#0000FF\">".stripslashes($tableau['titre'])."</font></h2>";
echo"<p>".stripslashes($tableau['news'])."</p>";
echo"<p>".$tableau['datenews']." - ".$tableau['categorie']."</p>";
echo"<hr>";
}
// affichage des liens
$j=0;
?>
<p>
<?php
while ($j<$t)
{
echo" <a href=\"news-
categorie.php?page=".$j."&categorie=".$categorie."\">".$categorie."-".$j."</a>";
$j=$j+1;
```

```
}
?>
</p>
```

Le procédé est identique à celui-ci dessus pour les news complète. Si une catégorie n'est pas sélectionnée, cette page reprend automatiquement la page standard news-affichage.php.

10. Affichage par date.

Cette partie va permettre à un utilisateur de sélectionner les news dans une période donnée (de date à date). Le premier développement va être le formulaire permettant de sélectionner les dates. Nous devons également vérifier si les dates sont valides. Nous n'utilisons pas ici de script purs, juste des fonctions en PHP. C'est là le problème.

Deux problèmes à régler:

- date rentrée existe (valide)
- date de départ supérieure à la date finale et vis versa, à nous de convertir pour reprendre la même programmation.

La première est la plus difficile. Elle va utiliser la fonction **checkdate (int mois, int jour, int année)** où: mois égale de 1 à 12, jour de 1 à 31 et année de 1 à 32767. La fonction tient compte des années bissextiles. Le formulaire doit donc être auto-invocant avant d'envoyer le formulaire suivant la méthode GET.

Le fichier de rentrée des dates et de vérification des données est repris sous **news-form-date.php.**

```
<?php
if (isset($_POST['B1']))
{
$date_init=$_POST['date_init'];
$date_fin=$_POST['date_fin'];
// vérification de la date initiale
$day_init=substr($date_init,0,2);
$mois_init=substr($date_init,3,2);
$annee_init=substr($date_init,6,4);
$day_fin=substr($date_fin,0,2);
$mois_fin=substr($date_fin,3,2);
$annee_fin=substr($date_fin,6,4);
// vérification de la date la plus ancienne. Version PHP<5.0, erreur en cas de date
indérieure au 01/01/1970
if ($annee_init < 1970){
$annee_init=1970;
}
```

```
if ($annee_fin < 1970){
$annee_fin=1970;
}
if((!is_numeric($day_init))|| (!is_numeric($mois_init)) ||
(!is_numeric($annee_init))||(!checkdate($mois_init,$day_init,$annee_init)))
{
echo"Date initiale non valide";
$date_init=date('d/m/Y');
}elseif ((!is_numeric($day_fin))|| (!is_numeric($mois_fin)) ||
(!is_numeric($annee_fin))||(!checkdate($mois_fin,$day_fin,$annee_fin))){
echo"Date finale non valide";
$date_fin=date('d/m/Y');
}else{
// vérification date la plus ancienne
if (mktime(0,0,0,$mois_init,$day_init,$annee_init)>
mktime(0,0,0,$mois_fin,$day_fin,$annee_fin)){
$day1=($annee_fin.$mois_fin.$day_fin);
$day2=($annee_init.$mois_init.$day_init);
}else{
$day2=($annee_fin.$mois_fin.$day_fin);
$day1=($annee_init.$mois_init.$day_init);
}
//requête de sélection
require ('includes\start.php');
$requete="select * from news where (datenews>='$day1') and (datenews<='$day2')";
$resultat=mysql_query($requete);
while ($tableau=mysql_fetch_array($resultat)){
echo "<h2><font color=\"#0000FF\">".stripslashes($tableau['titre'])."</font></h2>";
echo"<p>".stripslashes($tableau['news'])."</p>";
echo"<p>".$tableau['datenews']." - ".$tableau['categorie']."</p>";
echo"<hr>";
}
}
}else{
$date_init=date('d/m/Y');
$date_fin=date('d/m/Y');
}
// formulaire

$form="<form method=\"POST\"><p>Date de départ: <input type=\"text\"
name=\"date_init\" size=\"10\" value=\"".$date_init."\"></p>
<p>Date finale: <input type=\"text\" name=\"date_fin\" size=\"10\"
value=\"".$date_fin."\"></p>
<p><input type=\"submit\" value=\"Sélectionner\" name=\"B1\"></p>
```

```
</form>";
echo$form;
?>
```

Remarque: l'affichage est complet, il n'y a pas de pages séparées toutes les 10 news.

11. Mise en forme

Il ne nous reste plus qu'à mettre en forme nos différents fichiers pour la présentation. Nous utiliserons simplement un tableau comme ci-dessus. Les différentes parties sont reprises dans le dossier includes sous les noms header.php, footer.php, column_right.php (colonne de droite) et column_left.php (colonne de gauche)

Header		
column_left	Corps de la page (contenu)	column_right
Footer		

Dans le pack d'installation, les colonnes de gauche et de droites sont déjà remplies. Il vous suffit de modifier le header (en-tête) et le footer (pied de page) selon le design de votre site. Différentes parties de navigation peuvent également être insérées dans vos différentes pages.

Ce développement est téléchargeable sur le site dahut.be.

24. Utilisation des cookies avec PHP

Diverses applications nécessitent de différencier les utilisateurs: compteurs de visites et statistiques diverses (même si une solution externe gratuite ou payante est souvent préférable) ou le suivi d'utilisateurs inscrits. Dans les 2 cas, la distinction entre deux visiteurs différents se fait soit par l'adresse IP du visiteur, soit par de petits fichiers textes appelés cookies ou même par un paramètre de session (déconseillé).

Dans le premier cas, plusieurs visiteurs provenant d'une même connexion interne (via un routeur) seront vus comme un seul utilisateur. Par contre, les cookies sont insérés sur l'ordinateur du visiteur (donc unique par visiteur) et lus par le serveur Internet. Cette solution est donc nettement plus intéressante puisqu'elle distincte les visiteurs des visites. Par contre, l'utilisateur peut interdire les cookies sur son ordinateur.

Une autre utilisation est liée aux utilisateurs enregistrés. Ceci permet de différencier les utilisateurs sur tous les forums, logiciels de vente en ligne, ... Un autre système utilise des ID de session, un nombre unique qui suit le visiteur durant sa visite sur le site mais est affiché dans l'adresse de la page. Comme le numéro d'identification change à chaque nouvelle connexion, cette solution est déconseillée pour le référencement dans les moteurs de recherche.

La gestion utilise les en-têtes HTTP, d'une manière similaire à la méthode POST des formulaires. La fonction qui permet de créer des cookies en PHP:
setcookie($nom,$valeur,$expiration)

- $nom est le nom du cookies
- $valeur est la chaîne de caractères à transmettre
- $expiration est la date d'expiration, sous forme d'un timeStamp.

La commande doit débuter le script PHP, avant tout affichage en utilisant les commandes print(), echo(), ... et avant le code HTML pour les premières versions de PHP

Si le cookies ne peut être envoyé (placé après le début), le message d'erreur est "Cannot send cookies -headers alredy sent by ..."

Sans date d'expiration, il est automatiquement supprimé en fermant la fenêtre du navigateur.

2. Créer les cookies

Si vous placez un cookie à l'aide de la fonction Setcookie, le prochain passage de l'utilisateur sur la page créera une variable tableau $_COOKIE, similaire aux constantes $_GET, $_POST ou $_SETVER déjà rencontrées.

Par exemple, si la commande est setcookie('ybet','site'); au prochain passage, la variable $_COOKIES['ybet']='site'.

Les compteurs utiliseront cette variable pour distinguer les visiteurs, les sessions utilisateurs permettront de suivre un utilisateur une fois celui-ci connecté ou même de permettre la connexion automatique.

```
<?php
if(!isset($_COOKIE['ybet']))
{
Setcookie('ybet','site');
Print ("Première visite");
}else{
Print ("Vous avez déjà visité la page");
}
?>
```

Analysons le code ci-dessus. Lors du démarrage du fichier, nous testons l'existence de la variable $_COOKIE['ybet']. Si elle n'existe pas, nous créons le cookies. Sinon, nous affichons le message "Vous avez déjà visité la page". Remarquez que nous n'avons pas mis de date d'expiration. Dès lors, une fois le visiteur passé, il sera toujours renseigné comme "déjà visité".

Attention, le **script suivant ne fonctionne pas.**

```
<?php
if(!isset($_COOKIE['ybet']))
{
Setcookie('ybet','site');
Print ("Première visite");
}else{
Print ("Vous avez déjà visité la page");
Setcookie('ybet','site');
}
?>
```

Renvoie le message: Warning: Cannot modify header information - headers already sent by (output started at c:\program files\easyphp1-8\www\cookie.php puisque l'affichage est déjà démarré par une commande Print();.

La méthode la plus courante consiste à mettre une date d'expiration du cookies. C'est un paramètre optionnel de la commande.

```
Setcookie("ybet","site",time()+3600);
```

Cette option commence par la fonction time() pour récupérer la date courante au format timestamp (nombre de secondes depuis le 01/01/1970) et d'ajouter la durée du cookies en secondes. Dans le cas ci-dessus, l'expiration du cookies se fait après 1 heures.

3. Récupérer les cookies.

Pour récupérer les cookies, nous utilisons la variable $_COOKIES (version PHP 4.1.0 et supérieure ou $HTTP_COOKIES_VARS (version inférieures). C'est une variable tableau. Nous allons afficher sa valeur via le petit programme ci-dessus:

```
<?php
if(!isset($_COOKIE['ybet']))
{
Setcookie('ybet','site',time()+3600);

Print ("Première visite sur le ");
```

```
echo $_COOKIE['ybet']."<br>";
}else{
Setcookie('ybet','avant',time()+60);
Print ("Vous avez déjà visité la page ");
echo $_COOKIE['ybet']."<br>";
}
?>
```

Si c'est la première visite, on insère un cookies et on l'affiche (avec sa valeur: site). Il est valable 1 heure. Dans le cas contraire, on affiche un texte suivant d'une autre variable (avant). Le fichier expire après 60 secondes.

4. Exemple d'utilisation: login - mot de passe

Nous allons utiliser les cookies pour sauvegarder un nom d'utilisateur suivi d'un mot de passe. Cette fonctionnalité est utilisée dans la majorité des forum, portals, ... L'utilisateur se connecte avec son login - mot de passe. Une case à cocher lui permet d'être reconnu directement par le site Internet à sa prochaine connexion.

Débutons par un petit formulaire permettant à l'utilisateur de se connecter. Pour l'instant, nous ne vérifions pas la validité de la connexion.

```
<form method="POST">

<p>Nom utilisateur: <input type="text" name="login" size="20"></p>

<p>Mot de passe: <input type="password" name="password" size="20"></p>

<p>Mémoriser la connexion: <input type="checkbox" name="memo[1]"
value="ON"> </p>

<p><input type="submit" value="Envoyer" name="B1"></p>

</form>
```

Remarquez que le mots de passe n'est pas de type texte, mais password. Les caractères affichés seront "cachés" lors de la frappe. En deuxième, nous utilisons un champ de formulaire de type "**checkbox**". PHP les analyse comme des variables de type tableau, ils ne sont pas passez par $_POST mais bien comme valeur de matrice. En plus, si la case n'est pas cochée, la variable tableau n'est pas envoyée.

Nous allons l'insérer dans un fichier php auto-invocant.

```
<?php
if (isset($_POST['B1']))
{
```

```
// à ce stade, nous ne vérifions pas si l'utilisateur existe
$login=addslashes($_POST['login']);
$password= addslashes($_POST['password']);
if (isset($memo[0]))
{
$memo=$memo[0];
}else{
$memo="";
}
echo $login." ".$password." ".$memo;
}

$form="<form method=\"POST\"><p>Nom utilisateur: <input type=\"text\"
name=\"login\" size=\"20\"></p>
<p>Mot de passe: <input type=\"password\" name=\"password\" size=\"20\"></p>
<p>Mémoriser la connexion: <input type=\"checkbox\" name=\"memo[]\" value=\"ON\">
</p>
<p><input type=\"submit\" value=\"Envoyer\" name=\"B1\"></p>
</form>";
echo $form;
?>
```

La partie **checkbox** vérifie si effectivement la matrice est définie. Dans le cas contraire, on lui donne valeur nulle.

Il nous reste à créer le cookies utilisateur. Si la case est cochée, nous lui donnons une durée de 10 jours (c'est un exemple), sinon, une heure, le temps de la navigation. Nous utilisons une simple fonction conditionnelle si.

```
<?php
if (isset($_POST['B1']))
{
// à ce stade, nous ne vérifions pas si l'utilisateur existe
$login=addslashes($_POST['login']);
$password= addslashes($_POST['password']);
if (isset($memo[0]))
{
$memo=$memo[0];
}else{
$memo="";
}
if ($memo=="")
{
Setcookie('ybet',$login."-pw-".$password,time()+3600);
```

```
}else{
Setcookie('ybet',$login."-pw-".$password,time()+864000);
}
}
$form="<form method=\"POST\"><p>Nom utilisateur: <input type=\"text\"
name=\"login\" size=\"20\"></p>
<p>Mot de passe: <input type=\"password\" name=\"password\" size=\"20\"></p>
<p>Mémoriser la connexion: <input type=\"checkbox\" name=\"memo[]\" value=\"ON\">
</p>
<p><input type=\"submit\" value=\"Envoyer\" name=\"B1\"></p>
</form>";
echo $form;
?>
```

Remarque: la partie affichage va remplacer le cookies à chaque passage du nom utilisateur dans la partie suivante.

Il nous reste à récupérer le cookies sur chaque page pour vérifier l'utilisateur.

5. Mise à jour du cookie.

Cette partie du développement va mettre à jour la durée de validité du cookies à chaque passage de l'utilisateur sur une page.

Avant de créer le programme en PHP, quelques rappels sur les cookies. Un utilisateur peut interdire les cookies. De même, les robots d'indexation ne les acceptent pas. Dans certains cas, on va détecter si l'utilisateur les accepte et "forcer" les cookies via un fichier temporaire sur l'hébergement qui suit le visiteur. C'est utilisé notamment pour suivre les visiteurs sur les pages (statistiques des visites) Dans notre cas, nous n'allons pas les forcer. Si l'utilisateur ne les accepte pas, nous ne le suivons pas, tout simplement. Il n'est utilisé ici que pour les utilisateurs inscrits.

Testez ce petit programme sur votre navigateur. Il envoie un cookies et juste après vérifie s'il peut être lu mais ce programme a un petit problème, le cookies est vérifié uniquement **au passage sur une page suivante**. La détection se fait en javascript, pas en PHP.

```
<?php
// ce programme vérifie si le navigateur accepte les cookies.
Setcookie('cookies',"tests",time()+60);
if(!isset($_COOKIE['cookies']))
{
echo "N'accepte pas";
}else{
echo "Accepte";
}
```

```
?>
```

Nous allons pourtant utiliser cette partie pour suivre notre visiteur. Le visiteur doit préalablement s'inscrire. Sur chaque page, nous vérifions si le cookies existe. S'il n'est inscrit, nous passons la détection. Par contre, s'il est inscrit, nous modifions le cookies pour remettre la durée à 1 mois. Le cookies reprend le login et le mot de passe. Ce mot de passe est défini dans le formulaire d'inscription utilisateur.

6. Vérification de l'utilisateur

Avant d'envoyer le cookies, nous allons vérifier si l'utilisateur existe. Pour cela, nous allons utiliser la table member créée dans le chapitre 14. Nous ne vérifions finalement que le login (champ username) et le mot de passe (champ password). Cette partie va s'insérer juste avant l'envoi des cookies.

De nouveau, nous utilisons le fichier start.php pour débuter la connexion sur la base de donnée. Un utilisateur doit être préalablement créé dans la table.

```
<?php
/* vérification login - mot de passe.

$login reprend le nom utilisateur
$password, le mot de passe
$memo, sauvegarde cookies 10 jours.

*/
$login="YBET";
$password="ybet";
require('includes/start.php');
$requete="SELECT * FROM member where username='$login' and
password='$password'";
$valeur=mysql_query($requete);
$ligne=mysql_num_rows($valeur);
if ($ligne==0)
{
echo"Utilisateur inconnu";
}else{
echo"Utilisateur connu";
}
?>
```

Ce petit programme ne fait qu'afficher "Utilisateur connu" s'il est déjà enregistré dans la table MySql et "Utilisateur inconnu s'il n'est pas encore enregistré. Il nous reste à l'insérer

dans le programme ci-dessus de gestion des cookies. Remarque, nous utilisons également une variable Memo suivi d'un timestamp pour la durée de connexion.

```
<?php
if (isset($_POST['B1']))
{
// à ce stade, nous ne vérifions pas si l'utilisateur existe
$login=addslashes($_POST['login']);
$password= addslashes($_POST['password']);
if (isset($memo[0]))
{
$memo=$memo[0];
}else{
$memo="";
}
// echo $login." ".$password." ".$memo;
// Vérification dans la table member
if(!mysql_connect('localhost','root')){
Echo'Connection Impossible';
exit();
}
Mysql_select_db('ybet');
$requete="SELECT * FROM member where username='$login' and
password='$password'";
$valeur=mysql_query($requete);
$ligne=mysql_num_rows($valeur);
if ($ligne==0)
{
echo"Utilisateur inconnu";
die('<meta http-equiv="refresh" content="3; URL=inscription.php">');
}else{
//echo"Utilisateur connu";
if ($memo=="")
{
Setcookie('ybet',$login."-pw-".$password."memo".$memo,time()+3600);
}else{
Setcookie('ybet',$login."-pw-".$password."memo".$memo,time()+864000);

}
echo"Bonjour ".$login.", vous êtes maintenant connecté";
die('<meta http-equiv="refresh" content="3; URL=index.php">');
}
}
$form="<form method=\"POST\"><p>Nom utilisateur: <input type=\"text\"
name=\"login\" size=\"20\"></p>
```

```
<p>Mot de passe: <input type=\"password\" name=\"password\" size=\"20\"></p>
<p>Mémoriser la connexion: <input type=\"checkbox\" name=\"memo[]\" value=\"ON\">
</p>
<p><input type=\"submit\" value=\"Envoyer\" name=\"B1\"></p>
</form>";
echo $form;
?>
```

Si l'utilisateur est connu, nous insérons le cookies et le redirigeons vers la page d'entrée. Par contre, en cas d'erreur, il est renvoyé vers la page inscription. **Remarque**, le code inséré dans le cookies reprend le login, mot de masse et champ MEMO. C'est solution n'est pas très sécurisée, n'importe quel utilisateur pourra récupérer les 2 en lisant les cookies directement sur un ordinateur.

7. Gestion des utilisateurs dans les pages.

Dernier petit développement, suivre l'utilisateur avec ses cookies. Pour cela, nous devons vérifier sur chaque page si le cookies est déjà implanté. S'il ne l'est pas, nous l'empêchons simplement d'exécuter quelques fonctions (comme inscrire une annonce). Si l'utilisateur est connu et suivi, nous mettons le cookies à jour et lui permettons différentes possibilités, comme la modification de son compte et l'inscription de nouvelles annonces.

Dans cette formation, nous nous contentons de suivre l'utilisateur sur la partie enregistrement des annonces. Le développement complet est repris sur le site officiel du développement.

Nous allons tout simplement insérer le code suivante au début de notre fichier d'insertion des annonces

```
<?php
//Vérification de l'utilisateur
if(!isset($_COOKIE['ybet']))
{
echo"Vous ne pouvez pas poster d\'annonce, merci de vous connecter";
$login="";
}else{
$cookies=$_COOKIE['ybet'];
$cookies_array=array();
$cookies_array=Explode("-pw-",$cookies);
$login=$cookies_array[0];
$l=$cookies_array[1];
$cookies_array=explode("memo",$l);
$password=$cookies_array[0];
$memo=$cookies_array[1];
if(!mysql_connect('localhost','root')){
```

```
//Echo'Connection Impossible';
exit();
}
Mysql_select_db('ybet');
$requete="SELECT * FROM member where username='$login' and
password='$password'";
$valeur=mysql_query($requete);
$ligne=mysql_num_rows($valeur);

if ($ligne==0)
{
echo"Vous ne pouvez pas rentrer d'annonce, merci de vous connecter";
die('<meta http-equiv="refresh" content="3; URL=inscription.php">');
}else{
//echo"Utilisateur connu";
if ($memo=="")
{
Setcookie('ybet',$login."-pw-".$password."memo".$memo,time()+3600);
}else{
Setcookie('ybet',$login."-pw-".$password."memo".$memo,time()+864000);
}
while ($tableau=mysql_fetch_array($valeur)){
$mail=$tableau['email'];
$uid_util=$tableau['uid'];
}
}
}

?>
<html>
```

Quelques remarques:

- nous récupérons diverses informations de la table member comme son adresse mail et son numéro d'utilisateur.
- cette partie doit être insérée avant <HTML>

8. Pour conclure

Nous venons d'utiliser les cookies pour vérifier un utilisateur. Cette partie ne doit normalement pas être insérée dans un site de cette manière. Ceci est lié à l'envoi des coordonnées (login et mot de passe) dans le cookies qui peut-être récupéré directement sur l'ordinateur en claire.

Ca nous a déjà permis d'envoyer un le fichier de suivi, de le récupérer et de vérifier s'il est correct (en fonction d'une table utilisateur). C'est déjà pas si mal. Cette méthode oblige néanmoins l'utilisateur à les accepter pour insérer une annonce (par exemple). Si c'est correct dans le cas d'un login utilisateur, cette méthode ne peut pas être utilisée pour des fonctions de suivi sur les pages (statistiques par exemple). Le mot de passe n'est pas non plus crypté, nous en reparlerons dans un prochain chapitre.

14.25. Fonctions FTP et dossiers avec PHP

Nous avons déjà utilisé le protocole de transfert FTP pour transférer nos fichiers de notre ordinateur (en local) vers le site internet. Ceci utilise un simple client FTP (un logiciel).

Dans cette partie, nous allons lier FTP à PHP, pour nous permettre de transférer automatiquement un fichier d'un serveur (ou d'un site) vers un autre, éventuellement un site complet ou même un fichier texte MySQL à importer. Nous ne rentrerons pas trop dans toutes les commandes, juste permettre un transfert simple de données entre serveurs.

Un transfert nécessite que le serveur qui reçoit les fichiers soit configuré comme serveur FTP. Tous les serveurs sur Internet sont configurés pour cette fonction (à quelques rares exceptions, pour des ordinateurs hébergés en interne dans l'entreprise et reliés à Internet via un serveur démilitarisé DMZ. C'est loin d'être le plus courant. Par contre, EasyPhp n'inclus pas un serveur de ce type. La suite se fait donc directement sur un serveur INTERNET.

Pour créer une connexion, vous devez connaître trois paramètres:

- le nom du serveur (souvent désigné par **HOST**), par exemple ftp.ybet.be (sans //). un nom de serveur, éventuellement une adresse IP.
- un login (nom d'utilisateur) qui vous permet de vous connecter.
- un mot de passe (password) qui garantit que vous êtes l'utilisateur effectif.

Dans la suite du cours (même s'ils ne fonctionnent pas), nous utiliserons ftp.ybet.be comme Host, YBET comme login et PS comme mot de passe. Ces paramètres sont fournis par votre hébergeur.

2. Base de connexion FTP

Le mécanisme est finalement équivalent à une connexion sur une base de donnée MySQL:

- Ouverture de la communication
- login en utilisant les paramètres fournis par l'hébergeur Internet.

Mais forcément, les commandes seront différentes.

La première chose est d'établir une **liaison avec le serveur**. Ceci se fait pas la commande PHP:

```
ftp_connect ( HOST [, numero port ], temps de connexion ])
```

HOST est le nom du serveur FTP, le numéro de port est le numéro utilisé pour se connecter. S'il est omis, c'est le 21 utilisé par défaut. Le temps de connexion est la durée de connexion avant la fermeture s'il n'y a pas de transfert. Il est exprimé en secondes (90 secondes par défaut).

```
<?php
$ftp_server = "ftp.ybet.be";
$connect = ftp_connect($ftp_server);
?>
```

Ensuite, connexion comme utilisateur en reprenant le nom utilisateur et le mot de passe

```
ftp_login($connect,login,mot de passe)
$connect est le paramètre récupéré de ftp_connect
```

Ceci donne pour la connexion en langage PHP:

```
$login="YBET";
$password="PS";
$login_result = ftp_login($connect, $login, $password);
```

La partie suivante va vérifier si la connexion est effective, cette méthode ci-dessous est la plus facile.

```
if (ftp_login($connect, $login, $password)) {
echo "Connecté en tant que $login sur $ftp_server<br>";
} else {
echo "Connexion impossible en tant que ".$login."<br>";
}
```

Une fois la communication établie, vous pouvez la fermer par la commande

```
ftp_close($connect)
$connect est le paramètre récupéré de ftp_connect
```

3. Commandes liées aux fichiers et répertoires.

Le but n'est pas trop de rentrer dans les détails, mais de voire les fonctions les plus utilisées pour les transferts FTP. Cette partie de reprend quasiment **que le serveur de destination**, pas votre serveur de départ (qui sera vu au chapitre 4).

3.1. Transfert d'un fichier.

Commençons par le plus simple, un transfert de fichier depuis un site vers un autre. La première opération est de télécharger le fichier ci-dessous dans le dossier source. Supposons que ce fichier s'appelle tests.php. Nous allons le transférer sur le site ybet.be

```
<?php
$ftp_server = "ftp.ybet.be";

$login="YBET";
$password="PS";
$connect = ftp_connect($ftp_server);
if (ftp_login($connect, $login, $password)) {
echo "Connecté en tant que $login sur $ftp_server<br>";
} else {
echo "Connexion impossible en tant que ".$login."<br>";
}
$destination_file="/www/tests.php";
$source_file="tests.php";
$upload = ftp_put($connect, "$destination_file", "$source_file", FTP_ASCII);
if (!$upload) {
echo "Le transfert Ftp a échoué!";
} else {
echo "Téléchargement de ". $source_file." sur ". $ftp_server." en ".$destination_file;
}
?>
```

La commande principale est FTP_PUT qui renvoit True en cas de succès et False en cas d'erreur. Dans notre cas, nous envoyons le fichier du dossier courant (où est téléchargé le programme) vers le dossier /www/tests.php. Ceci est lié à la structure des dossiers des serveurs Linux (mais peut être paramétré sur la console d'administration du serveur).

```
$upload = ftp_put($connect, "$destination_file", "$source_file", FTP)
```

FTP doit être renseigné, il prend la valeur FTP_BINARY ou FTP_ASCII.

La variable de retour permet de vérifier si le fichier a bien été téléchargé.

3.2. Créer un dossier

Nous pouvons également créer un dossier sur le serveur de destination. Remarquez de nouveau le /www/ spécifique à un serveur Linux mais paramétrable.

```php
<?php
$directory="/www/directory";
$dossier=ftp_mkdir ($connect,$directory);
?>
```

3.3. Modifie le répertoire de destination courant.

Une fois le dossier créé (ou même avant de le créer, cas d'un sous-dossier), vous pouvez modifier le dossier courant de destination (pas celui depuis le quel vous transférez des fichiers).

```php
<?php
$directory="/www/hardware";
$dossier=ftp_chdir ($connect,$directory);
?>
```

3.4. Dossier courant de destination.

La solution de transfert de fichier ci-dessus permet de transférer un fichier vers un autre serveur (ou dossier) mais le nom du fichier doit être connu. En plus, vous ne savez pas où est ce dossier. Cette commande nécessite au préalable de modifier le répertoire courant, sinon elle ne renvoit rien.

```php
<?php
$dossier=ftp_pwd($connect);
?>
```

3.5. Fichiers dans le dossier courant de destination.

Cette fonction renvoie une variable sous forme de tableau. Comme vu dans le cours sur les variables de type matrice du cours PHP-MySQL, pour récupérer le nombre de valeur (fichiers et dossiers), nous pouvons utiliser la fonction count($tableau) dans le cas d'une matrice à une entrée. Le script ci-dessus permet également via la boucle While d'afficher les fichiers et dossiers présents dans le répertoire de destination.

```php
<?php
$tableau=ftp_nlist($connect,$repertoire);
$nombre=count($tableau);
echo "nombre: ".$nombre."<br>";
$i=0;
while ($i<21)
{
echo $tableau[$i]."<br>";
$i=$i+1;
```

```
}
?>
```

La commande **FTP_RAWLIST()** est identique mais renvoie également les informations sur le fichier sous forme de tableau.

3.6. Créer un dossier

```
$dir="directory";
$valeur=ftp_mkdir($connect, $dir));
```

Cette commande permet de créer un dossier dans le dossier courant en cours.

3.7. Suppression d'un dossier

```
$dir="directory";
$valeur=ftp_rmdir($connect, $dir));
```

3.8. Taille d'un fichier.

La fonction **ftp_size ($connect,fichier)** permet de récupérer la taille d'un fichier (en octet), mais elle peut aller plus loin puisque la taille d'un dossier est de 0 (en fait c'est -1 qui est renvoyé), permettant de dissocier les fichiers de dossiers. Jetons un oeil sur le programme ci-dessous.

```
$directory="/www/hardware";
$dossier=ftp_chdir ($connect,$directory);

$dossier=ftp_pwd($connect);
echo "dossier courant ".$dossier;
$tableau=ftp_nlist($connect,$repertoire);
$nombre=count($tableau);
echo "nombre: ".$nombre."<br>";
$i=0;
while ($i<$nombre)
{
echo $tableau[$i]." - ".ftp_size ($connect,$tableau[$i])."<br>";
$i=$i+1;
}
```

La première partie va positionner le curseur dans le dossier /www/hardware/ (commande FTP_CHDIR) et l'afficher via FTP_PWD.

La partie suivante va récupérer les fichiers dans ce dossier via FTP_NLIST, afficher le nombre puis dans la boucle While afficher le nom et la taille associée. Ceux qui ont une taille de -1 sont automatiquement des répertoires.

3.9. Suppression d'un fichier.

Pour supprimer un fichier sur le dossier distant (de destination),

```
$file="tests.php";
$valeur=FTP_DELETE ($connect,$file);
```

La variable $valeur est booléenne, false si la commande a échouée, True si elle a réussit.

3.10. CHMOD

Dernière commande de cette liste (il y en a d'autres), celle qui permet de modifier les propriétés CHMOD d'un fichier.

```
$fichier="tests.php";
$int=0644;
// la valeur est constituée de 4 chiffres en octal (de 0 à 7).
$valeur=FTP_CHMOD($connect,$int,$fichier);
```

4. Commande du dossier source.

Dans la partie précédente, nous avons vu les commandes permettant de gérer la **partie destination**. Il nous faut également gérer les dossiers et fichiers de départs (**source**). Ces commandes ne sont pas liées à FTP, mais bien à PHP et à Apache.

4.1. Le dossier courant.

La première question est de connaître le dossier courant du pointeur. Au départ, c'est celui où s'exécute le programme PHP, mais il sera modifié par la suite par une autre fonction.

```
echo getcwd();

// par exemple /home/ybet/www. Remarquez que c'est le dossier de type Linux, pas de l'hébergement, home étant le dossier reprenant les dossiers utilisateurs, ybet, le nom utilisateur. En pratique, ça ne gênera pas trop.
```

4.2. Changer le dossier courant.

Une fois le dossier courant connu (par défaut, celui où s'exécute le programme PHP), vous pouvez modifier le dossier courant pour copier des fichiers par exemple.

```
CHDIR ('directory');
// descend dans le sous-dossier directory
CHDIR ('..');
// remonte au niveau de la root
CHDIR('../');
// remonte d'un niveau de dossier.
CHDIR('directory/include');
// descend dans le sous-dossier directory, puis include.
```

4.3. Ouverture, lecture et fermeture d'un dossier.

Pour lire le contenu d'un dossier, vous devez d'abord ouvrir le dossier avec la fonction **opendir('dossier')**. OPENDIR('.') ouvre le dossier courant.

La partie lecture utilise la fonction **readdir()** qui reprend les fichiers 1 à 1. Elle nécessite donc une boucle pour lire le contenu complet d'un répertoire.

En dernier, vous devez fermer le dossier par la commande **closedir().**

```
$dir = opendir('.');
while (false !== ($file = readdir($dir))) {
echo $file."<br>";
}
closedir($handle);

// cette fonction affiche le contenu du dossier en cours, fichiers et sous-dossiers.
```

Une **nouvelle fonction avec PHP5** permet directement d'afficher le contenu. Elle renvoie une variable sous forme de tableau.

```
$tableau=scandir ("dossier");
// par ordre alphabétique croissant
$tableau=scandir ("dossier",1);
// par ordre alphabétique décroissant
```

4.4. Créer ou supprimer un dossier

Deux commandes permettent de créer ou de supprimer un dossier. De nouveaux, elles renvoient une commande booléenne pour le résultat de la fonction.

bool=mkdir('répertoire'[,CHMOD]) permet de créer un sous-dossier. Le code CHMOD peut également être inséré automatiquement.

RMDIR('répertoire') permet de supprimer un sous-dossier.

Par exemple, créons le dossier verif dans le dossier courant

```
if (!mkdir ('verif'))
{
echo 'erreur';
}else
{
echo "réussi";
}
// la partie suivante positionne le curseur dans le dossier verif.
CHDIR ('verif');
echo getcwd();
```

4.5. Fichier ou dossier?

Deux commandes permettent finalement de déterminer si c'est un fichier ou un dossier. Elles renvoient de nouveau une variable booléenne.

```
$variable=is_dir ('dossier' )
// renvoie true si le fichier existe et si c'est un dossier, faux sinon
$variable=is_file ('fichier' )
// renvoie true si le fichier existe et si c'est bien un fichier (pas un dossier), faux sinon
```

4.6. Est-ce que le fichier existe?

```
$variable=file-exist ('fichier' )
// renvoie true si le fichier existe et faux (false) sinon
```

4.7. Effacer un fichier

Deux options sont possibles, la commande unlink() et la fonction delete (qui renvoie en fait vers la première). Elle n'efface pas un dossier, réservé à RMDIR() vue plus haut.

```
$variable=unlink ('fichier' );
// renvoie true si le fichier existe et faux (false) sinon
```

4.8. Copier un fichier

Cette commande utilise COPY. Le fichier de destination est écrasé s'il existe déjà. De nouveau, le résultat est une valeur booléenne indiquant si le transfert a réussi. Il n'y a pas de caractères de remplacement.

```
$variable=copy('source','destination')
// ou
```

```
if (!copy('tests.php','tests.sav'))
{
echo "raté";
} else
{
echo "La copie s'est bien passée";
}
```

4.9. Déplacer ou renommer un fichier

La commande MOVE n'existe pas en PHP, c'est la commande rename qui permet de donner un nouveau nom au fichier ou de le déplacer.

```
if (!rename('tests.php','tests.sav'))
{echo "raté";
}else
{
echo "réussi";
}
// déplacer dans un sous-dossier par exemple (ici verif) et renommer le fichier.
if (!rename('tests.php','verif/tests.sav'))
{echo "raté";
}else
{
echo "réussi";
}
```

4.10 Contenu d'un fichier.

Via la **commande FILE**, vous pouvez lire et afficher le contenu d'un fichier (où plutôt le résultat HTML, pas les lignes de commandes en PHP). Elle permet via des commandes d'ouverture et de modification du contenu de fichier de modifier le contenu. Nous ne verrons que celle-ci.

```
<?php
// Lit une page web dans un tableau.
$lines = file ('http://www.ybet.be/');
// Affiche toutes les lignes du fichier en code HTML, avec numéro de ligne
foreach ($lines as $line_num => $line)
{
echo 'Ligne: ' . $line_num . ' : ' . htmlspecialchars($line) . '<br />'."\n";
}
?>
```

4.11. Pour terminer.

Voici pour les principales. Les autres commandes ne sont pas vues ici, notamment les commandes qui permettent d'ouvrir un fichier pour ajouter du contenu. Pour la liste des commandes de dossiers, fichiers, vous pouvez vérifier directement dans les manuels en ligne. Les commandes permettant de télécharger un fichier avec un dossier ont déjà été vues dans la partie téléchargement image.

5. Quelques exemples pratiques

5.1. Déplacé tous les fichier d'un dossier vers un sous-dossier sur votre serveur.

Nous ouvrons d'abord le dossier en cours, récupérons les fichiers dans une boucle et les transférons vers le sous-dossier verif (en fait un fichier appelé 'verif/'.$file. Les dossiers ne sont pas copiés mais bien affichés. Vous pouvez utiliser la fonction is_dir() pour éventuellement ne pas les afficher dans la boucle (ou pour tous autres traitements).

```
$dir = opendir('.');
while (false !== ($file = readdir($dir))) {
echo $file."<br>";
$file1="verif/".$file;
copy ($file,$file1);
}
closedir($handle);
```

5.2. Déplacer tous les fichiers d'un dossier source vers un dossier à créer sur un autre serveur.

Nous allons mélanger les fonctions source et FTP (destination) pour cette partie. Le programme ci-dessous ne déplace pas les sous-dossiers, juste les fichiers.

La première chose est de créer la connexion FTP sur le serveur de destination et de créer le répertoire. Dans notre cas, nous allons utiliser

```
$ftp_server = "ftp.ybet.be";
$login="YBET";
$password="PS";
$connect = ftp_connect($ftp_server);
if (ftp_login($connect, $login, $password)) {
echo "Connecté en tant que $login sur $ftp_server<br>";
} else {
echo "Connexion impossible en tant que ".$login."<br>";
}
// fin de connexion FTP, ouvert par défaut pendant 90 secondes
// Créer le dossier directory et déplacer le pointeur vers le dossier créé.
```

```
$directory="/www/directory";
$dossier=ftp_mkdir ($connect,$directory);
$dossier=ftp_chdir ($connect,$directory);
// récupération des fichiers sur le dossier source (en cours) comme ci-dessus et les envoyer vers le dossier de destination.

// lorsque le fichier source est un dossier, il n'est pas créé en destination et le transfert échoue

while (false !== ($file = readdir($dir))) {
// début transfert
$destination_file=$file;
$source_file=$file;
$upload = ftp_put($connect,$destination_file,$source_file, FTP_ASCII);
if (!$upload) {
echo "Le transfert Ftp a échoué!<br>";
} else {
echo $file." Téléchargement de ". $source_file." sur ". $ftp_server." en ".$destination_file."<br>";
}
// fin transfert
}
closedir($handle);
// on ferme la connexion FTP.
```

26. Le cryptage des données en PHP.

Lorsque nous avons utilisé les Cookies, nous n'avons pas coder le mot de passe. Ceci pose quelques problèmes de sécurité. Cette partie va nous permettre d'étudier quelques fonctions. J'essayerais d'être le plus simple possible. Pourtant, les fonctions PHP permettant de crypter et de décrypter un texte sont suffisamment nombreuses pour en rebuter plus d'un. En plus, les fonctions PHP permettent une multitude de possibilités liées à la méthode de cryptage (encore une bonne raison d'essayer de faire simple en n'en voyant que quelques unes).

Reprenons la gestion des cookies comme vu précédemment lors de la connexion d'un utilisateur (sans la partie vérification dans la base de donnée). Dans l'exemple ci-dessous, le nom du cookies est YBET. Le contenu reprend le login suivi de -pw- puis seulement le mot de passe. Nous avons volontairement supprimé le timestamp pour ne pas compliquer. Cette partie vérifie donc la présence du cookies ybet, et récupère le login et mot de passe s'il n'existe pas.

```
if(!isset($_COOKIE['ybet']))
{
```

```
echo"Vous n'êtes pas encore inscrit ou connecté";
}else{
$cookies=$_COOKIE['ybet'];
$cookies_array=array();
$cookies_array=Explode("-pw-",$cookies);
$login=$cookies_array[0];
$password=$cookies_array[1];
)
```

A l'inverse, lorsque nous récupérons le login et le mot de passe dans la base de donnée, la création du cookie est

```
Setcookie('ybet',$login."-pw-".$password);
```

2. Méthodes d'encryptage.

Dans le cas général, le cryptage utilise une clé. C'est une suite de lettres (ou une phrase) qui est connue des deux parties. Dans notre cas, la clé est uniquement connue par le serveur qui crypte le mot de passe lors de la création et le décrypte lors de la récupération par le cookies.

D'un autre coté, il y a différentes méthodes, on parle d'algorithme, du plus simple au plus compliqué.

En dernier, il y a la méthode de gestion du login et du mot de passe (vous pouvez éventuellement ne crypter que le mot de passe).

- Soit, vous crypter dans le cookies et dans la table utilisateur. Dans ce cas, vous pouvez directement faire correspondre les deux.
- Dans l'autre cas (non conseillé), vous ne cryptez que le contenu du cookies. Cette solution permet de voire le mot de passe en clair dans la table.

La fonction (ou plutôt la bibliothèque) **mcrypt** accessible depuis PHP 4.0.2 permet une large panoplie d'algorithmes de chiffrement, tels que DES, DES, TripleDES, Blowfish (par défaut), 3-WAY, SAFER-SK64, SAFER-SK128, TWOFISH, TEA, RC2 et GOST en modes CBC, OFB, CFB et ECB. Elle accepte également RC6 et IDEA qui ne sont pas considérés comme libres de droits. La majorité utilisent un chiffrement sur 64 bits.

La commande de chiffrement est:

$texte_chiffre = mcrypt_encrypt($algo, $clef, $texte, $mode, $iv);

où:

- $algo est l'algorithme utilisé, différentes variables sont associées (voire ci-

dessous)

- $clef est la clé de chiffrement, connue uniquement par le serveur. C'est une phrase de 64 bits (lettres) maximum.
- $texte, le texte (mot de passe, login, ...) à chiffrer.
- $mode, est spécifique à chaque méthode de chiffrement (algorithme).
- $iv est un vecteur d'initialisation utilisé pour quelques modes (CBC, CFB, OFB, et dans certains algorithmes de mode STREAM). Si ce vecteur est nécessaire, la fonction renvoie une alerte et utilise une suite de caractère \0

La commande de déchiffrement est:

```
$texte_dechiffre = mcrypt_decrypt($algo,$clef,$texte_chiffre,$mode,$iv);
```

où:

- $algo est l'algorithme utilisé
- $clef est la clé de chiffrement, connue uniquement par le serveur. C'est une phrase de 64 bits (lettres) maximum.
- $texte_chiffre est le texte à décrypter.
- $mode, est spécifique à chaque méthode de chiffrement (algorithme).
- $iv est le vecteur d'initialisation.

Les modes de chiffrement et de déchiffrage doivent forcément être les mêmes!.

Des variables sont préprogrammées pour les différentes méthodes de cryptage comme par exemple:**:**

- MCRYPT_MODE_CBC pour le mode CBC
- MCRYPT_BLOWFISH
- MCRYPT_CAST_256
- MCRYPT_DES
- MCRYPT_TRIPLEDES
- .....

Au niveau du mode de cryptage, on en retrouve 4 utilisables avec la commande Mcrypt:

- **cfb**, associé à MCRYPT_MODE_CFB, chiffrement par flux de clés qui est ensuite appliqué sur le texte, le flux suivant est obtenu à partir du précédant bloc, iv obligatoire
- **ofb**, associé à MCRYPT_MODE_OFB, similaire au mode cfb, iv obligatoire
- **cbc**, associé à MCRYPT_MODE_CBC, enchaînement des blocs avec une fonction OU exclusif avec le chiffrage du bloc précédant, iv optionnel
- **ecb**, associé à MCRYPT_MODE_ECB, la méthode la plus simple. Le texte est découpé en blocs cryptés les uns après les autres.

- **nofb**, associé à la variable MCRYPT_MODE_NOFB, identique à OFB mais plus sécurisé, avec iv.

3. Exemples d'utilisation.

Une remarque, **certaines erreurs** sont liées au paramétrage du fichier php.ini et tous les algorithme ne sont pas implantés sur tous les serveurs. **EasyPhp** n'accepte normalement aucune méthode de chiffrement. Ces tests doivent dont être exécutés directement sur un serveur Internet.

Commençons par un cryptage simple, algorythme BLOWFICH en mode CBC (donc sans vecteur d'initialisation)

```
<?php
echo "------------ mode CBC";

$algo = MCRYPT_BLOWFISH;
$mode = MCRYPT_MODE_CBC;

$key_size = mcrypt_module_get_algo_key_size($algo);

// choix d'une clé secrète de cryptage/décryptage et mise à longueur
$cle= "Ceci est la clé du cryptage utilisée par ybet informatique";
$cle= substr($cle, 0, $key_size);
// Phrase à crypter et cryptage
$texte= "texte à crypter";
$chiffre= mcrypt_encrypt($algo, $cle, $texte, $mode);
// Décryptage de contrôle
$dechiffre = mcrypt_decrypt($algo, $cle, $chiffre, $mode);
// affichage de contrôle
echo "<p>Texte à crypter: <b>".$texte. "</b></p><p> Texte chiffre: <b>" .$chiffre.
"</b></p>
<p> Mot de passe décrypter: <b>" .$dechiffre."</b><br />";

?>
```

Une autre méthode utilisant un vecteur d'initialisation, plus sécurisée mais plus complexe à mettre en oeuvre:

```
<?php
echo "------------ BLOWFISH avec mode NOFB -----------";

// choix d'un algo, mode
```

```
$algo = MCRYPT_BLOWFISH;
$mode = MCRYPT_MODE_NOFB;
// calcul des longueurs max de la clé et de l'IV
$key_size = mcrypt_module_get_algo_key_size($algo);
$iv_size = mcrypt_get_iv_size($algo, $mode);
// création d'un IV de la bonne longueur débutant par exemple de iv suivi de 0
$iv= mcrypt_create_iv($iv_size, substr("exemple de iv",0,$iv_size));
// choix d'une clé secrète de chiffrement et mise à longueur suivant l'algorithme et le mode associé
$cle= "Ceci est la clé du cryptage utilisée par ybet informatique";
$cle= substr($cle, 0, $key_size);
// Phrase à crypter et cryptage
$texte= "Mon mot de passe à crypter";
$chiffre= mcrypt_encrypt($algo, $cle, $texte, $mode, $iv);
// Décryptage
$dechiffre = mcrypt_decrypt($algo, $cle, $chiffre, $mode, $iv);
// affichage
echo "

Mot de passe à crypter: ".$texte. "


Mot de passe crypté: " .$chiffre. "



Mot de passe décrypté: " .$dechiffre."

";
?>
```

Le résultat nous donne:

---------- BLOWFISH mode CBC ----------

Mot de passe à chiffrer: **texte à crypter**

Mot de passe chiffre: **¯vÔà¥®FÊ——————————————šÆy“&**

Mot de passe déchiffrer: **texte à crypter**

---------- BLOWFISH avec mode NOFB ----------

Mot de passe à crypter: **Mot de passe à crypter**

Mot de passe crypté: **Ì™ËTA«v"µ'>=(ÆjÒZà´3R**

Mot de passe décrypté: **Mot de passe à crypter**

Pas très difficile en théorie, tant que vous avez correctement sélectionner les paramètres. Il ne reste plus qu'à l'implanter dans la gestion des cookies et la relier à la table utilisateurs. Mais çà c'est une autre histoire.

27. Base de donnée relationnelle MySQL

Avant de voire les requêtes spécifiques, voyons d'abord ce qu'est une base de donnée relationnelle. Il y a d'abord "relation", le terme devrait déjà aider mais voyons plutôt une petite explication plus concrète:

Supposons un module de facturation. Il reprend d'un coté des clients, de l'autre côté des factures. A chaque facture est attachée un client. Dès lors, plutôt que d'encoder à chaque fois les coordonnées du client pour chaque nouvelle livraison, on va créer deux tables MySQL, l'une reprend la facture en elle même, l'autre la fiche du client. L'astuce est de créer une liaison entre les deux. Elle se fait généralement via un champ commun, par exemple le code. Celui-ci est repris dans les deux tables, ce qui permet de relier les deux tables entre-elles lorsque c'est nécessaire. De même, on peut créer une relation entre les codes des produits et leur fiche.

Le gros avantage est lié non seulement au gain de temps lors de l'encodage, mais aussi au gain de place dans les tables ou même à la possibilité de changer l'adresse du client sans devoir retaper toutes les anciens documents. Par contre, la clé de la relation doit être unique, pas question que la même facture soit reliée à deux clients différents.

MySQL est nativement relationnel (au même titre qu'Access, DBase, ... ou même MsSQL de Microsoft) mais nous avons déjà vu que PHP et MySQL ne sont pas en fait liés. Le langage de programmation a simplement créer des fonctions spécifiques lui permettant d'exécuter des requêtes, une sorte de symbiose plus ou moins réussie entre deux logiciels du monde "libre".

Les relations étant relativement complexes, avec de nombreuses options, cette partie de la formation est volontairement épurée de bon nombre de cas spéciaux. Si elle ne sera pas tout à fait complète, elle devrait permettre de créer et d'utiliser rapidement des commandes et des tables relationnelles

2. Petit exemple de départ

Cette partie va reprendre deux tables à créer sous easyPhp (PhPMyadmin) ou dans la base de donnée fournie avec votre hébergement. En local, créez la base de donnée RELATION.

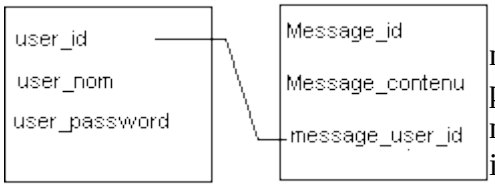
La première table reprend l'utilisateur:

Table USER

- user_id: int, 10, **clé primaire, auto increment**
- user_nom: texte, 40
- user_password: texte, 40

la deuxième reprend ses messages.

- message_id: int, 10, **clé primaire, auto increment**
- message_contenu: texte, 50
- message_user_id: int 10, idex



La relation ressemble à ceci. Remarquez que nous n'utilisons pas le champ message_user_id comme clé primaire mais comme index. Ceci n'est pas obligatoire mais va augmenter la vitesse de traitement puisque les index et clés primaires sont déjà pré-organisés par la base de donnée. Par contre, un index permet d'entrer la même valeur dans différents enregistrements.

Entrez manuellement quelques valeurs (3 minimum) dans les 2 champs et mettant au moins un nombre égal dans user_id et message_user_id.

L'entre d'un message par un utilisateur va donc remplir 3 champs dans la table MESSAGE, par contre l'affichage va permettre d'afficher également le nom de l'utilisateur. Le premier avantage, si l'utilisateur change de nom (ou de pseudo), ses messages lui seront toujours associés. Comme nous avons créé user_id comme clé primaire, nous savons déjà que la valeur de ce champ **sera unique** dans la table user.

L'ouverture de la base de donnée se fait par la liste de commandes ci-dessous, elle peut éventuellement être inscrite dans un fichier en include:

```
<?php

if(!mysql_connect('localhost','root')){
Echo'Connection Impossible';
exit();
} else{
// Echo'Connexion réussie';
}
Mysql_select_db('relation');

?>
```

Deux méthodes peuvent au départ être utilisée, la mauvaise (disons plutôt l'ancienne donc à éviter):

```
Select user_nom, message from user, message  where
user.user_id=message.message_user_id
```

L'autre méthode va utiliser le JOIN et la requête va être de type

```
SELECT ...
FROM <table de gauche> LEFT JOIN <table de droite>
ON <Condition de liaison>
```

Par principe, mais surtout pour éviter des problèmes si des champs ont le même nom dans les différentes tables liées et pour augmenter la vitesse de traitement, sélectionnez les champs à utiliser et nom pas "SELECT *".

3. La méthode Where

Cette méthode n'est pas la meilleure, notamment parce que la clause **WHERE** va mélanger des conditions de liaisons de tables et des conditions de valeurs des champs mais débutons par celle-ci. Elle ne doit plus être utilisée. Reprenons nos tables ci-dessus.

```
$requete="select user_nom,message_id,message_contenu from message,user where
user.user_id=message.message_user_id";
$valeur=mysql_query($requete);
while ($tableau=mysql_fetch_array($valeur)){
echo$tableau['user_nom']."- numéro message: ".$tableau['message_id']." -
contenu:".$tableau['message_contenu']."<br />";
}
```

La première partie est finalement le plus important, c'est la requête qui va afficher toutes les messages de la table message en affichant le nom de l'utilisateur repris dans la table user via la relation **user.user_id=message.message_user_id.** Remarquez que le pour **délimiter les champs en fonction de leur table**, nous avons ajouter le **"nom de la table." devant**. Ceci peut également être fait dans les noms des champs si deux noms de champs sont identiques dans les différentes tables.

Quoique intéressante, cette requête affiche toutes les valeurs. Supposons que nous voulons uniquement les messages de l'utilisateur numéro 1, la requête va reprendre une deuxième condition dans le where.

```
$requete="select user_nom,message_id,message_contenu from message,user where
user.user_id=message.message_user_id && user.user_id=1";
```

C'est le gros problème de cette méthode au niveau lecture de la requête, les conditions de jointures sont complètement mélangées avec les conditions restrictives des valeurs, mais elle fonctionne.

4. La méthode JOIN

C'est la méthode la plus propre, elle va séparer les liaisons des conditions de valeurs. Plusieurs types de jointures existent, notamment LEFT JOIN (par la gauche), RIGHT JOIN (par la droite) et INNER JOIN. Avant de commencer, nous allons créer une troisième table:

Table renseignement

- renseignement_id, INT 10, autoincrement, clé primaire
- renseignement_nom, VARCHAR 50, index
- renseignement_date, date
- renseignement_texte, VARCHAR 100

Elle reprend finalement des renseignements généraux sur l'utilisateur et est liée à la table user par le champ renseignement_nom.

4.1. INNER JOIN

C'est la jointure par défaut qui compare deux tables et retourne tous les enregistrement comportant une concordance de liaison.

Reprenons notre exemple ci-dessus et remplaçons la requête par:

```
$requete = "select user_nom, message_id,message_contenu from message INNER JOIN
user ON user.user_id = message.message_user_id";
```

Ceci est exactement équivalent à la méthode where sans conditions de champs. Si nous exécutons en plus une condition de valeur (par exemple, users_id=1, la requête SQL en PHP devient:

```
$requete = "select user_nom, message_id, message_contenu from message INNER JOIN
user ON user.user_id = message.message_user_id where user.user_id=1";
```

Effectivement, la méthode INNER JOIN est finalement équivalente à la condition Where standard, si ce n'est la séparation des conditions de valeurs et de liaison.

Les deux méthodes suivantes vont nous permettre de peaufiner les requêtes en signalant les directions des jointures. Elles seront plus précises, mais plus complexes. Par contre, elles vont permettre de créer des liaisons entre 3 tables avec comme relations table1 <-> table2 <-> table3, avec aucune liaisons entre les tables 1 et 2.

4.2. LEFT JOIN

Pour une liaison par la gauche, nous allons utiliser la commande LEFT Join. En plus des concordances, la commande va afficher **toutes les lignes de la table de gauche**, même s'il n'y a pas de concordances. Dans notre cas, les utilisateurs sans messages seront donc également affichées.

```
$requete = "select user_nom, message_id, message_contenu from user LEFT JOIN
message ON user.user_id = message.message_user_id";
```

Un petit exemple d'affichage donnerait:

users 1- numéro message: 1 - contenu:message 1
users 1- numéro message: 2 - contenu:message2 de 1
utilisateur 2- numéro message: 3 - contenu:message 3
users 3- numéro message: - contenu:

4.3. Left Join sur trois tables.

Nous allons cette fois compliquer un peu en utilisant les données de 3 tables.

```
Le principe est d'utiliser

SELECT * FROM message
LEFT JOIN user ON (instruction)
LEFT JOIN renseignement ON (instruction)
```

Message est la table principale (tout affiché à cause du Left Join), elle est reliée à la table user, elle même reliée à la table renseignement.

```
$requete ="select * from message
LEFT JOIN user ON user.user_id = message.message_user_id
LEFT JOIN renseignement on user.user_nom=renseignement.renseignement_nom";
```

Pour une table de 3 utilisateurs (users1,2 et 3 avec 2 messages pour le premier et 1 pour l'utilisateur 2 (pas pour le troisième), ceci donne:

users 1- numéro message: 1 - contenu:message 1
users 1- numéro message: 2 - contenu:message2 de 1
users 2- numéro message: 3 - contenu:message 3

4.4. Mélange.

Pour cette partie, nous allons relier nos trois tables en utilisant les parties LEFT et RIGHT JOIN. Avant toute chose, des tables complétées ou non vont afficher des résultats tout à fait différents: le contenu des liaisons sont être complètement complété. En plus, inverser la commande RIGHT avec LEFT va aussi changer les résultats.