

PHP MySQL Tutorial

By
Khairul

PHP MySQL Tutorial

Soon you will know just how easy it is to build dynamic web pages using PHP and MySQL. Really, PHP and MySQL is easy to learn, and I hope this tutorial will help you realise that :-)

I'm sure that by the end of this tutorial you will have enough knowledge to build your own database driven website using PHP & MySQL. Of course this tutorial **is not** perfect. So if you have any critiques, question, problem or suggestion please [let me know](#).

For a fast start just go straight to the fourth section where you can learn how to [connect to MySQL using PHP](#). However, if you haven't installed PHP and MySQL go visit the first section. It explains how to install PHP, MySQL and Apache. And if you are a newcomer in PHP go to the second section to learn the basics of PHP.

Ok, here is the list of tutorials you can find in this website. Feel free to browse around.. .

1. [Installing Apache PHP and MySQL](#)

In case you haven't installed the trio yet **don't skip** this section. This page explains about installing Apache, PHP and MySQL on Windows plus some images to make things clearer. It also covers modifying Apache configuration and PHP configuration so the two can work together.

2. [PHP Tutorial](#)

Give you enough to get started. First, you will learn how to open and close PHP blocks, continued with using comments, a brief explanation about PHP variables and types. Then you'll learn about manipulating strings, control structures, functions and how to use web forms.

3. [MySQL Tutorial](#)

You will learn about starting MySQL, adding new MySQL user, creating a database and tables. Then you'll learn about the SQL queries to insert data, get the data, update and delete.

4. [Connecting to MySQL database](#)

This is where you start to put PHP and MySQL together. This page explains how to open and close MySQL connection with PHP.

5. [Creating a MySQL database](#)

Obviously you will need to create your database first. This part explains how to create MySQL database and table through PHP

6. [Insert Data To MySQL Database](#)

After you have the database and tables ready it's time to learn how to insert your data into the

database.

7. [Getting The Data](#)
Once you have the data stored in the database surely you want to get it back. This page explains how to get your data out of MySQL plus how to convert your query result into Excel format
8. [Using Paging](#)
This one explains how to show your query result into multiple pages and how to create the navigation link. Also show the problem that might happen when using paging and the solution.
9. [Update and Delete](#)
Explains how to update and delete your data and how to use table locking to prevent violation of data integrity.
10. [Using PHP To Backup MySQL Database](#)
In this page you can learn three different ways to backup your MySQL database
11. [Form Validation](#)
This one explains how to validate HTML form on server side using PHP plus client side form validation using Javascript to make your form more user friendly.
12. [Creating a Guestbook](#)
Guestbook is one of the most common feature for a website and this tutorial will teach you how to create your own guestbook using PHP and MySQL. It also explain how to use PHP functions to prevent code injection and the use of paging.
13. [Uploading Files to MySQL](#)
This page describe how to upload a file to MySQL database and how to download it back.
14. [Creating a Content Management System \(CMS\)](#)
Content Management System is getting more and more popular by the day. This tuto rial explains how to create a simple CMS, how to add, modify and delete content using web form.
15. [User Authentication](#)
This part explain three methods of authenticating a user. T he first is hardcoding the user info in the script itself. The second one check for the user id and password in database. The third one add an random number verification.
16. [Image Gallery](#)
Just another tutorial on making an image gallery
17. [Finding Web Hosting for PHP and MySQL](#)
Just some tips for choosing the right host.
18. [Freelance PHP and MySQL jobs](#)

Internet is the best place to find freelance jobs. This page show one of them and also some list you need to think about before going freelancing.

19. [Q & A](#)

I put some of the questions that i received here (plus the answers). Also added the common queries regarding php and mysql

20. [PHP MySQL Bookstore](#)

Since this tutorial doesn't cover everyt hing about php and mysql programming I decided to add a book store on this site. Here you can find several great books about programming with PHP and MySQL. And if you're thinking about replacing your old computer checkout the [computer store](#)

21. [Shopping Cart Tutorial](#)

I finally complete this tutorial. It's kindof big so i put it in it's own domain. Because it's ne w i'm begging you to send me your critiques. The demo site is working with the administration stuff disabled (like add/modify/delete product) but you can download the source code so you can try it on your computer.

22. [Online resources](#)

Some website related to PHP and MySQL. Actually a couple of these resources are not related to PHP or MySQL but I put them there anyway because they are useful. Check it out, you may find some that interest st you

NOTE:

This tutorial currently only covers PHP4. When my web host support PHP5 I'll start adding some PHP5 specific features. Stay tuned for more php mysql tutorial

Installing Apache PHP and MySQL

PHP and MySQL are usually associated with LAMP (Linux, Apache, MySQL, PHP). However, most PHP developer (including me) are actually using Windows when developing the PHP application. So this page will only cover the WAMP (Windows, Apache, MySQL, PHP). You will learn how to install Apache, PHP, and MySQL under Windows platform.

The first step is to download the packages :

- Apache : www.apache.org
- PHP : www.php.net
- MySQL : www.mysql.com

You should get the latest version of each packages. As for the example in this tutorial i'm using

Apache 2.0.50 (apache_2.0.50-win32-x86-no_ssl.msi), PHP 4.3.10 (php-4.3.10-Win32.zip) and MySQL 4.0.18 (mysql-4.0.18-win.zip).

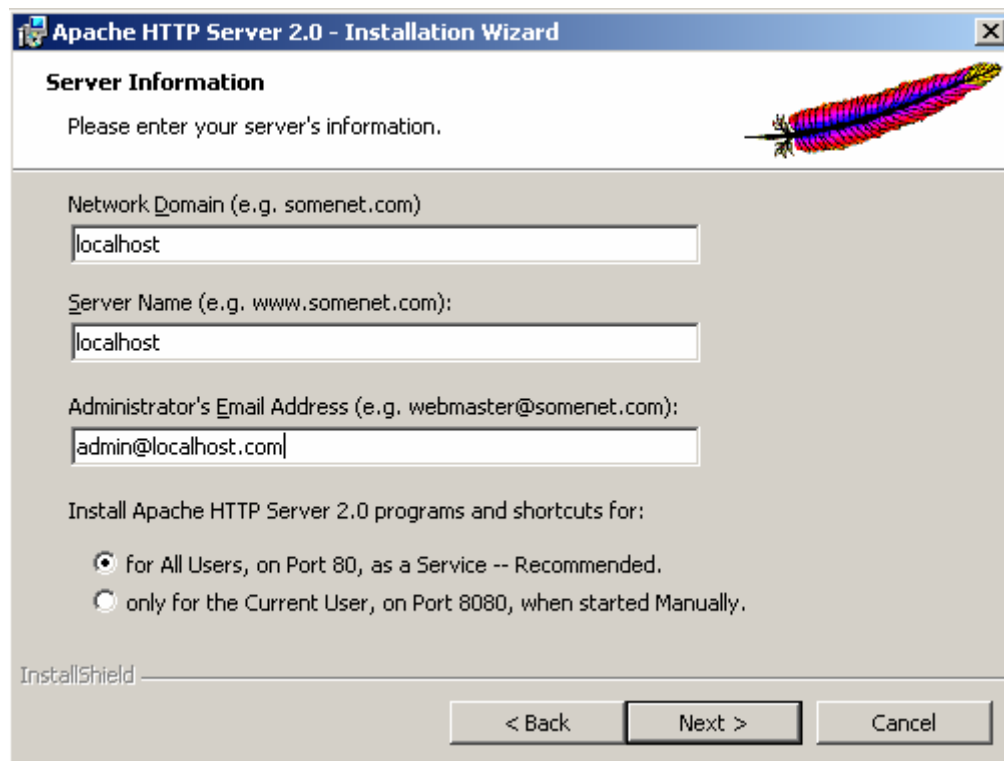
Now let's start the installation process one by one.

- [Installing Apache](#)
- [Installing PHP](#)
- [Modifying Apache Configuration](#)
- [Installing MySQL](#)
- [Modifying PHP Configuration File](#)

Installing Apache

Installing apache is easy if you download the Microsoft Installer (.msi) package. Just double click on the icon to run the installation wizard. Click next until you see the Server Information window. You can enter localhost for both the Network Domain and Server Name. As for the administrator's email address you can enter anything you want.

I'm using Windows XP and installed Apache as Service so everytime I start Windows Apache is automatically started.



Click the Next button and choose Typical installation. Click Next one more time and choose where

you want to install Apache (I installed it in the default location C:\Program Files\Apache Group). Click the Next button and then the Install button to complete the installation process.

To see if you Apache installation was successful open up you browser and type `http://localhost` in the address bar. You should see something like this :



By default Apache's **document root** is set to **htdocs** directory. The document root is where you must put all your PHP or HTML files so it will be process by Apache (and can be seen through a web browser). Of course you can change it to point to any directory you want. T he configuration file for Apache is stored in C:\Program Files\Apache Group\Apache2\conf\httpd.conf (assuming you installed Apache in C:\Program Files\Apache Group) . It's just a plain text file so you can use Notepad to edit it.

For example, if you want to put all your PHP or HTML files in C:\www just find this line in the httpd.conf :

```
DocumentRoot "C:/Program Files/Apache Group/Apache2/htdocs"
```

and change it to :

```
DocumentRoot "C:/www"
```

After making changes to the configuration file you have to restart A pache (Start > Programs >

Apache HTTP Server 2.0.50 > Control Apache Server > Restart) to see the effect.

Another configuration you may want to change is the **directory index**. This is the file that Apache will show when you request a directory. As an example if you type <http://www.php-mysql-tutorial.com/> without specifying any file the [index.php](#) file will be automatically shown.

Suppose you want apache to use index.html, index.php or main.php as the directory index you can modify the DirectoryIndex value like this :

DirectoryIndex index.html index.php main.php

Now whenever you request a directory such as <http://localhost/> Apache will try to find the index.html file or if it's not found Apache will use index.php. In case index.php is also not found then main.php will be used.

Installing PHP

First, extract the PHP package (php-4.3.10-Win32.zip). I extracted the package in the directory where Apache was installed (C:\Program Files\Apache Group\Apache2). Change the new created directory name to php (just to make it shorter). Then copy the file php.ini-dist in PHP directory to you windows directory (C:\Windows or C:\Winnt depends on where you installed Windows) and rename the file to php.ini. This is the PHP configuration file and we'll take a look what's in it later on.

Next, move the php4ts.dll file from the newly created php directory into the sapi subdirectory. Quoting from php installation file you can also place php4ts.dll in other places such as :

- In the directory where apache.exe is start from (C:\Program Files\Apache Group\Apache2\bin)
- In your %SYSTEMROOT%\System32, %SYSTEMROOT%\system and %SYSTEMROOT% directory.
Note: %SYSTEMROOT%\System32 only applies to Windows NT/2000/XP)
- In your whole %PATH%

Side Note : Thanks to Shannon Tang for pointing this out

Modifying Apache Configuration

Apache doesn't know that you just install PHP. We need to tell Apache about PHP and where to find it. Open the Apache configuration file in C:\Program Files\Apache Group\Apache2\conf\httpd.conf and add the following three lines :

```
LoadModule php4_module php/sapi/php4apache2.dll
AddType application/x-httpd-php .php
AddType application/x-httpd-php-source .phps
```


The first line tells Apache where to load the dll required to execute PHP and the second line means that every file that ends with .php should be processed as a PHP file. You can actually change it to anything you want like .html or even .asp! The third line is added so that you can view your php file source code in the browser window. You will see what this mean when you browse this tutorial and click the link to the example's source code like [this one](#).

Now restart Apache for the changes to take effect (Start > Programs > Apache HTTP Server 2.0.50 > Control Apache Server > Restart). To check if everything is okay create a new file, name it as test.php and put it in document root directory (C:\Program Files\Apache Group\Apache2\htdocs). The content of this file is shown below.

```
<?php
phpinfo();
?>
```


phpinfo() is the infamous PHP function which will spit out all kinds of stuff about PHP and your server configuration. Type <http://localhost/test.php> on your browser's address bar and if everything works well you should see something like this :

ss <http://localhost/test.php>

PHP Version 4.3.10 

System	Windows NT DAKU 5.1 build 2600
Build Date	Dec 14 2004 17:46:48
Server API	Apache 2.0 Handler
Virtual Directory Support	enabled
Configuration File (php.ini) Path	C:\WINDOWS\php.ini
PHP API	20020918
PHP Extension	20020429
Zend Extension	20021010
Debug Build	no
Thread Safety	enabled
Registered PHP Streams	php, http, ftp, compress.zlib

This program makes use of the Zend Scripting Language Engine:
 Zend Engine v1.3.0, Copyright (c) 1998-2004 Zend Technologies



Installing MySQL

First extract the package (mysql-4.0.18-win.zip) to a temporary directory, then run setup.exe. Keep clicking the next button to complete the installation. By default MySQL will be installed in C:\mysql.

Open a DOS window and go to C:\mysql\bin and then run `mysqld-nt --console` , you should see some messages like these :

C:\mysql\bin>mysqld-nt --console

InnoDB: The first specified data file .\ibdata1 did not exist:

InnoDB: a new database to be created!

040807 10:54:09 InnoDB: Setting file .\ibdata1 size to 10 MB

InnoDB: Database physically writes the file full: wait...

040807 10:54:11 InnoDB: Log file .\ib_logfile0 did not exist: new to be created

InnoDB: Setting log file .\ib_logfile0 size to 5 MB

InnoDB: Database physically writes the file full: wait...
040807 10:54:12 InnoDB: Log file .\ib_logfile1 did not exist: new to be created

InnoDB: Setting log file .\ib_logfile1 size to 5 MB
InnoDB: Database physically writes the file full: wait...
InnoDB: Doublewrite buffer not found: creating new
InnoDB: Doublewrite buffer created
InnoDB: Creating foreign key constraint system tables
InnoDB: Foreign key constraint system tables created
040807 10:54:31 InnoDB: Started
mysqld-nt: ready for connections.
Version: '4.0.18-nt' socket: " port: 3306

Now open another DOS window and type C:\mysql\bin\mysql

if your installation is successful you will see the MySQL client running :

C:\mysql\bin>mysql

Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 1 to server version: 4.0.18-nt

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql>

Type exit on the mysql> prompt to quit the MySQL client.

Now let's **install MySQL as a Service**. The process is simple just type mysqld-nt --install to install the service and net start mysql to run the service. But make sure to shutdown the server first using mysqladmin -u root shutdown

C:\mysql\bin>mysqladmin -u root shutdown

C:\mysql\bin>mysqld-nt --install

Service successfully installed.

C:\mysql\bin>net start mysql

The MySQL service was started successfully.

C:\mysql\bin>mysql

Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 1 to server version: 4.0.18-nt

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql>

Modifying PHP Configuration File (php.ini)

PHP stores all kinds of configuration in a file called `php.ini`. You can find this file in the directory where you installed PHP. Sometimes you will need to modify this file for example to use a PHP extension. I won't explain each and every configuration available just the ones that often need modification or special attention.

Some of the configurations are :

1. `register_globals`
2. `error_reporting` and `display_errors`
3. `extension` and `extension_path`
4. `session.save_path`
5. `max_execution_time`

register_globals

Before PHP 4.2.0 the default value for this configuration is **On** and after 4.2.0 the default value is **Off**. The reason for this change is because it is so easy to write **insecure** code with this value on. So make sure that this value is Off in `php.ini`.

error_reporting and display_errors

Set the value to `error_reporting = E_ALL` during development but after production set the value to `error_reporting = E_NONE` .

The reason to use `E_ALL` during development is so you can catch most of the nasty bugs in your code. PHP will complain just about any errors you make and spit out all kinds of warning (for example if you're trying to use an uninitialized variable).

However, after production you should change the value to `E_NONE` so PHP will keep quiet even if there's an error in your code. This way the user won't have to see all kinds of PHP error message when running the script.

One important thing to note is that you will also need to set the value of `display_errors` to On. Even if you set `error_reporting = E_ALL` you will not get any error message (no matter how buggy our script is) unless `display_errors` is set to On.

extension and extension_path

PHP4 comes with about 51 extensions such as GD library (for graphics creation and manipulation), CURL, PostgreSQL support etc. These extensions are not turned on automatically. If you need to use the extension, first you need to specify the location of the extensions and then uncomment the extension you want.

The value of `extension_path` must be set to the directory where the extension is installed which is `PHP_INSTALL_DIR/extensions`, with `PHP_INSTALL_DIR` is the directory where you install PHP. For example I installed PHP in `C:\Program Files\Apache Group\Apache2\php` so the extensions path is :

```
extension_path = C:/Program Files/Apache Group/Apache2/php/extensions/
```

Don't forget to add that last slash or it won't work

After specifying the `extension_path` you will need to uncomment the extension you want to use. In `php.ini` a comment is started using a semicolon (;). As an example if you want to use GD library then you must remove the semicolon at the beginning of `;extension=php_gd2.dll` to `extension=php_gd2.dll`

session.save_path

This configuration tells PHP where to save the session data. You will need to set this value to an existing directory or you will not be able to use session. In Windows you can set this value as `session.save_path = c:/windows/temp/`

max_execution_time

The default value for `max_execution_time` is 30 (seconds). But for some scripts 30 seconds is just not enough to complete it's task. For example a database backup script may need more time to save a huge database.

If you think your script will need extra time to finish the job you can set this to a higher value. For example to set the maximum script execution time to 15 minutes (900 seconds) you can modify the configuration as `max_execution_time = 900`

PHP have a convenient function to modify PHP configuration in runtime, `ini_set()`. Setting PHP configuration using this function **will not** make the effect permanent. It last only until the script ends.

That's it, now you have Apache, PHP and MySQL installed and running smoothly. It's time to move on to the next section, [PHP Tutorial](#).

PHP Tutorial

What is PHP? PHP is a web programming language used to write dynamic webpages. In this tutorial you will learn basics of PHP. I will assume you already know a bit about programming language so I won't cover the whole thing.

This PHP Tutorial will explain the followings :

- [Opening and ending PHP tags](#)
- [Comments](#)
- [Variables](#)
- [Types](#)
- [Playing with strings](#)
- [Control structures](#)
- [Functions](#)
- [Using Forms](#)

First, make sure you already have PHP installed on your computer. If you don't have it installed visit the first part of this tutorial, [Install Apache PHP and MySQL](#). By the way this tutorial only covers PHP 4 not the old PHP 3 (I wonder if anyone still use it ?).

By the way, if you don't already use a suitable editor for coding [check out this page](#). If you already have your favorite editor just skip it and move on to the next part : [Opening and ending PHP tags](#)

PHP Editors

Because PHP files are just plain text files you can use any editor to create and edit PHP files. But even though you can use simple text editor like Windows Notepad it is alot easier if you use an editor capable of syntax highlighting and advance text manipulation. For a simple file like hello.php syntax highlighting will do nothing good, but when you write long pages it's alot easier to find mistakes just by looking at the text color.

Some editors you can use are :

- [Editplus](#)
- Macromedia Dreamweaver
- Frontpage
- [Eclipse](#) (with [PHP plugin](#))

Personally I like using Dreamweaver and Eclipse. I like Dreamweaver because my PHP code is usually mixed with HTML, and creating HTML tables and forms are just so easy in Dreamweaver. However, Dreamweaver doesn't support CVS (Concurrent Versioning System) which is critical when running projects with many programmers. So currently I'm using Eclipse for creating PHP

files.

By the way Eclipse is a free software and the PHP plugin is also free. If you have tight budget but in need of a good editor you should try Eclipse.

Opening & Ending PHP Tags

To open a block of PHP code in a page you can use one of these four sets of opening and closing tags

Opening Tag	Closing Tag
<?	?>
<?php	?>
<%	%>
<script language="php">	</script>

The first pair (<? and ?>) is called short tags. You should avoid using short tags in your application especially if it's meant to be distributed on other servers. This is because short tags are not always supported (though I never seen any web host that don't support it). Short tags are only available only explicitly enabled setting the short_open_tag value to On in the PHP configuration file php.ini.

So, for all PHP code in this website I will use the second pair, **<?php and ?>**.

Now, for the first example create a file named hello.php (you can use NotePad or your favorite text editor) and put it in your web servers root directory. If you use Apache the root directory is APACHE_INSTALL_DIR\htdocs, with APACHE_INSTALL_DIR is the directory where you install Apache. So if you install Apache on C:\Program Files\Apache Group\Apache2\htdocs then you put hello.php in C:\Program Files\Apache Group\Apache2\htdocs

Example : [hello.php](#)

Source code : [hello.phps](#)

```
<html>
<head>
<title>My First PHP Page</title>
</head>
<body>
<?php
echo "<p>Hello World, How Are You Today?</p>";
?>
</body>
</html>
```

To view the result start Apache then open your browser and go to `http://localhost/hello.php` or `http://127.0.0.1/hello.php`. You should see something like [this](#) in your browser window.

The example above shows how to insert PHP code into an HTML file. It also shows the `echo` statement used to output a string. See that the `echo` statement ends with a semicolon. Every command in PHP must end with a semicolon. If you forget to use semicolon or use colon instead after a command you will get an error message like this

Parse error: parse error, unexpected ':', expecting ',' or ';' in c:\Apache\htdocs\examples\get.php on line 7

However in the `hello.php` example above omitting the semicolon won't cause an error. That's because the `echo` statement was immediately followed by a closing tag.

Using Comments

Comment is a part of your PHP code that will not be translated by the PHP engine. You can use it to write documentation of your PHP script describing what the code should do. A comment can span only for one line or span multiple line.

PHP support three kinds of comment tags :

1. `//`
This is a one line comment
2. `#`
This is a Unix shell-style comment. It's also a one line comment
3. `/* */`
Use this multi line comment if you need to.

Example : [comments.php](#)

Source code : [comments.phps](#)

```
<?php
echo "First line <br>"; // You won't see me in the output
// I'm a one liner comment
```

```
/*
Same thing, you won't
see this in the output file
*/
```

```
echo "The above comment spans two lines <br>";

# Hi i'm a Unix shell-style comment
# Too bad you can't see me
echo "View the source of this file, you'll see no comments here <br>";
?>
```

PHP Variables

Variables in PHP are represented by a dollar sign followed by the name of the variable. The variable name is **case-sensitive**, so \$myvar is different from \$myVar.

A valid variable name starts with a letter or underscore, followed by any number of letters, numbers, or underscores.

Example :

```
<?php
$myvar      = "Hello";    // valid
$yourVar_is-123 = "World"; // valid
$123ImHere   = "Something"; // invalid, starts with number
?>
```

Variable Scope

The scope of a variable is the context within which it is defined. Basically you can not access a variable which is defined in different scope.

The script below will not produce any output because the function Test() declares no \$a variable. The echo statement looks for a local version of the \$a variable, and it has not been assigned a value within this scope. Depending on error_reporting value in php.ini the script below will print nothing or issue an error message.

Example : [scope.php](#)
Source code : [scope.phps](#)

```
<?php
```



```
$a = 1; // $a is a global variable
```

```
function Test()
{
    echo $a; // try to print $a, but $a is not defined here
}

Test();
?>
```

If you want your global variable (variable defined outside functions) to be available in function scope you need to use the `$global` keyword. The code example below shows how to use the `$global` keyword.

Example : [global.php](#)
Source code : [global.phps](#)

```
<?php
$a = 1; // $a is defined in global scope ...
$b = 2; // $b too

function Sum()
{
    global $a, $b; // now $a and $b are available in Sum()
    $b = $a + $b;
}

Sum();
echo $b;
?>
```

PHP Superglobals

Superglobals are variables that available anywhere in the program code. They are :

- **\$_SERVER**
Variables set by the web server or otherwise directly related to the execution environment of the current script. One useful variable is `$_SERVER['REMOTE_ADDR']` which you can use to know you website visitor's IP address

Example : [ip.php](#)
Source code : [ip.phps](#)

Your computer IP is
<?php
echo \$_SERVER['REMOTE_ADDR'];

?>

- **\$_GET**

Variables provided to the script via HTTP GET. You can supply GET variables into a PHP script by appending the script's URL like this : <http://www.php-mysql-tutorial.com/./examples/get.php?name=php&friend=mysql> or set the a form method as method="get"

Example: [get.php](#)

Source code : [get.phps](#)

```
<?php
echo "My name is {$_GET['name']} <br>";
echo "My friend's name is {$_GET['friend']}";
?>
```

Note that I put \$_GET['name'] and \$_GET['friend'] in curly brackets. It's necessary to use these curly brackets when you're trying to place the value of an [array](#) into a string.

You can also split the string like this :

echo "My name is " . {\$_GET['name']} . "
"; but it is easier to put the curly brackets.

- **\$_POST**

Variables provided to the script via HTTP POST. These comes from a form which set method="post"

- **\$_COOKIE**

Variables provided to the script via HTTP cookies.

- **\$_FILES**

Variables provided to the script via HTTP post file uploads. You can see the example in [Uploading files to MySql Database](#)

- **\$_ENV**

Variables provided to the script via the environment.

- **\$_REQUEST**

Variables provided to the script via the GET, POST, and COOKIE input mechanisms, and which therefore cannot be trusted. It's use the appropriate \$_POST or \$_GET from your script instead of using \$_REQUEST so you will always know that a variable comes from POST or GET.

- **\$GLOBALS**

Contains a reference to every variable which is currently available within the global scope of the script.

You usually won't need the last three superglobals in your script.

Variable Types

PHP supports eight primitive types.

Four scalar types:

- boolean : expresses truth value, TRUE or FALSE. Any non zero values and non empty string are also counted as TRUE.
- integer : round numbers (-5, 0, 123, 555, ...)
- float : floating-point number or 'double' (0.9283838, 23.0, ...)
- string : "Hello World", 'PHP and MySQL, etc

Two compound types:

- array
- object

And finally two special types:

- resource (one example is the return value of `mysql_connect()` function)
- NULL

In PHP an array can have numeric key, associative key or both. The value of an array can be of any type. To create an array use the `array()` language construct like this.

Example : [array.php](#)

Source code : [array.phps](#)

```
<?php
$numbers = array(1, 2, 3, 4, 5, 6);
$age = array("mom" => 45, "pop" => 50, "bro" => 25);
$mixed = array("hello" => "World", 2 => "It's two");

echo "numbers[4] = {"$numbers[4]} <br>";
echo "My mom's age is {"$age['mom']} <br>";
echo "mixed['hello'] = {"$mixed['hello']} <br>";
echo "mixed[2] = {"$mixed[2]}";
?>
```

When working with arrays there is one function I often used. The `print_r()` function. Given an array

this function will print the values in a format that shows keys and elements

Example : [printr.php](#)

Source code : [printr.phps](#)

```
<?php
$myarray = array(1, 2, 3, 4, 5);
$myarray[5] = array("Hi", "Hello", "Konnichiwa", "Apa Kabar");

echo '<pre>';
print_r($myarray);
echo '</pre>';
?>
```

Don't forget to print the preformatting tag `<pre>` and `</pre>` before and after calling `print_r()`. If you don't use them then you'll have to view the page source to see a result in correct format.

Type Juggling

In PHP you don't need to explicitly specify a type for variables. A variable's type is determined by the context in which that variable is used. That is to say, if you assign a string value to variable `$var`, `$var` becomes a string. If you then assign an integer value to `$var`, it becomes an integer.

An example of PHP's automatic type conversion is the addition operator `+`. If any of the operands is a float, then all operands are evaluated as floats, and the result will be a float. Otherwise, the operands will be interpreted as integers, and the result will also be an integer. Note that this does NOT change the types of the operands themselves; the only change is in how the operands are evaluated.

Example :

```
<?php
$myvar = "0"; // $myvar is string (ASCII 48)
$myvar += 2; // $myvar is now an integer (2)
$myvar = $foo + 1.3; // $myvar is now a float (3.3)
$myvar = 5 + "10 Piglets"; // $foo is integer (15)
?>
```

Type Casting

To cast a variable write the name of the desired type in parentheses before the variable which is to be cast.

Example : [casting.php](#)

Source code : [casting.phps](#)

```
<?php
$abc = 10; // $abc is an integer
$xyz = (boolean) $abc; // $xyz is a boolean

echo "abc is $abc and xyz is $xyz <br>";
?>
```

The casts allowed are:

- (int), (integer) - cast to integer
- (bool), (boolean) - cast to boolean
- (float), (double), (real) - cast to float
- (string) - cast to string
- (array) - cast to array
- (object) - cast to object

Playing With Strings

Strings are probably what you will use most in your PHP script. From concatenating, looking for patterns, trim, chop etc. So I guess it's a good idea to take a better look at this creature. We will also take a peek at some string functions that you might find useful for everyday coding.

Creating a string

To declare a string in PHP you can use double quotes (") or single quotes ('). There are some differences you need to know about using these two.

If you're using double-quoted strings variables will be expanded (processed). Special characters such as line feed (\n) and carriage return (\r) are expanded too. However, with single-quoted strings none of those things happen. Take a look at the example below to see what I mean.

Note that browsers don't print newline characters (\r and \n) so when you open [string.php](#) take a look at the source and you will see the effect of these newline characters.

Example : [string.php](#)

Source code : [string.phps](#)

```
<?php
$fruit = 'jamblang';
```

```

echo "My favourite fruit is $fruit <br>";
echo 'I lied, actually I hate $fruit <br>';

echo "\r\n My first line \r\n and my second line <br>\r\n";
echo ' Though I use \r\n this string is still on one line <br>';
?>

```

String Concatenation

To concat two strings you need the dot (.) operator so in case you have a long string and for the sake of readability you have to cut it into two you can do it just like the example below.

Actually if you need to write a loong string and you want to write it to multiple lines you don't need concat the strings. You can do it just like the second example below where \$quote2 is split into three lines.

Example : [concat.php](#)
Source : [concat.phps](#)

```

<?php

$quote1 = "Never insult Dumbledore " .
    "in front of me!";

$quote2 = "Nami,
you are
my nakama!";

echo $quote1 . "<br>";
echo $quote2;
?>

```

String Functions

substr(\$string, \$start, \$end) : get a chunk of \$string

```

<?php

// print '12'
echo substr('123456789', 0, 2);

// print '56789'

```

```

echo substr('123456789', 4);

// print '89'
echo substr('123456789', -2);

// print '456'
echo substr('123456789', 3, -4);
?>

```

str_repeat(\$string, \$n) : repeat \$string \$n times

For example if you want to print a series of ten asteriks (*) y ou can do it with a for loop like this :

```

<?php
for ($i = 0; $i < 10; $i++) {
    echo '*';
}
?>

```

Or you can go the easy way and do it like this :

```

<?php
echo str_repeat('*', 10);
?>

```

strrchr(\$string, \$char) : find the last occurence of the character \$char in \$string

For example: you want to get the file extension from a file name. You can use this function in conjunction with substr()

```

<?php
$ext = substr(strrchr($filename, '.'), 1);
?>

```

What the above code do is get a chunk of \$filename starting from the **last dot** in \$filename then get the substring of it starting from the second character (index 1).

To make things clearer suppose \$filename is 'tutorial.php'. Using strrchr('tutorial.php', '.') yield '.php' and after substr('.php', 1) we get the file extension; 'php'

trim(\$string) : remove extra spaces at the beginning and end of \$string

```
<?php
// print 'abc def'
echo trim(' abc def ');
?>
```

addslashes(\$string) : adding backslashes before characters that need to be quoted in \$string

This function is usually used on form values before being used for database queries. You will see this function used a lot in this tutorial ([like this one](#)) so there's no need to present an example here.

explode(\$separator, \$string) : Split \$string by \$separator

This function is commonly used to extract values in a string which are separated by a certain separator string. For example, suppose we have some information stored as comma separated values. To extract each values we can do it like shown below

```
<?php
// extract information from comma separated values
$csv = 'Uzumaki Naruto,15,Konoha Village';

$info = explode(',', $csv);
?>
```

Now, \$info is an array with three values :

```
Array
(
    [0] => Uzumaki Naruto
    [1] => 15
    [2] => Konoha Village
)
```

We can further process this array like displaying them in a table, etc.

implode(\$string, \$array) : Join the values of \$array using \$string

This one do the opposite than the previous function. For example to reverse back the \$info array into

a string we can do it like this :

```
<?php
$info = array('Uzumaki Naruto', 15, 'Konoha Village');

$csv = implode(',', $info);
?>
```

Another example : Pretend we have an array containing some values and we want to print them in an ordered list. We can use the implode() like this :

```
<?php
// print ordered list of names in array

$names = array('Uchiha Sasuke', 'Haruno Sakura', 'Uzumaki Naruto', 'Kakashi');

echo '<ol><li>' . implode('</li><li>', $names) . '</li></ol>';
?>
```

The result of that code is like an ordered list just like shown below

1. Uchiha Sasuke
2. Haruno Sakura
3. Uzumaki Naruto
4. Kakashi

By the way, i did write the above php code to print that list instead of writing the list directly

number_format(\$number): display a number with grouped thousands

When displaying numbers it's usually more readable if the numbers is properly formatted like 1,234,567 instead of 1234567. Using this function is very simple like shown below

```
<?php
// display 15,120,777
echo number_format(15120777);

?>
```

Control Structures

The next examples will show you how to use control structures in PHP. I won't go through all just the ones that i will use in the code examples in this site. The control structures are

- if
- else
- while
- for

If Else

The if statement evaluates the truth value of it's argument. If the argument evaluate as TRUE the code following the if statement will be executed. And if the argument evaluate as FALSE and there is an else statement then the code following the else statement will be executed.

Example : [visitor- info.php](#)

Source code : [visitor-info.phps](#)

```
<?php
$ip  = $_SERVER['REMOTE_ADDR'];
$agent = $_SERVER['HTTP_USER_AGENT'];

if(strpos($agent, 'Opera') !== false)
    $agent = 'Opera';
else if(strpos($agent, "MSIE") !== false)
    $agent = 'Internet Explorer';

echo "Your computer IP is $ip and you are using $agent";
?>
```

The strpos() function returns the numeric position of the first occurrence of it's second argument ('Opera') in the first argument (\$agent). If the string 'Opera' is found inside \$agent, the function returns the position of the string. Otherwise, it returns FALSE.

When you're using Internet Explorer 6.0 on Windows XP the value of \$_SERVER['HTTP_USER_AGENT'] would be something like:

Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)

and if you're using Opera the value the value may look like this :

Mozilla/4.0 (compatible; MSIE 6.0; MSIE 5.5; Windows NT 5.1) Opera 7.0 [en]

So if i you use Opera the strpos() function will return value would be 61. Since 61 !== false then the first if statement will be evaluated as true and the value of \$agent will be set to the string 'Opera'.

Note that I use the !== to specify inequality instead of != The reason for this is because if the string is found in position 0 then the zero will be treated as FALSE, which is not the behaviour that I want.

While Loop

The while() statement is used to execute a piece of code repeatedly as long as the while expression evaluates as true. For example the code below will print the number one to nine.

Example : [while.php](#)

Source code : [while.phps](#)

```
<?php
$number = 1;

while ($number < 10)
{
    echo $number . '<br>';
    $number += 1;
}
?>
```

You see that I make the code \$number += 1; as bold. I did it simply to remind that even an experienced programmer can sometime forget that a loop will happily continue to run forever as long as the loop expression (in this case \$number < 10) evaluates as true. So when you're creating a loop please make sure you already put the code to make sure the loop will end in timely manner.

Break

The break statement is used to stop the execution of a loop. As an example the while loop below will stop when \$number equals to 6.

Example : [break.php](#)

Source code : [break.phps](#)

```
<?php
$number = 1;
```

```

while ($number < 10)
{
    echo $number . '<br>';

    if ($number == 6)
    {
        break;
    }

    $number += 1;
}
?>

```

You can stop the loop using the break statement. The break statement however will only stop the loop where it is declared. So if you have a cascading while loop and you put a break statement in the inner loop then only the inner loop execution that will be stopped.

Example : [break2.php](#)

Source code : [break2.phps](#)

```

<?php
$floor = 1;

while ($floor <= 5)
{
    $room = 1;

    while ($room < 40)
    {
        echo "Floor : $floor, room number : $floor". "$room <br>";

        if ($room == 2)
        {
            break;
        }

        $room += 1;
    }
    $floor += 1;

    echo "<br>";
}
?>

```

If you run the example you will see that the outer loop, while (\$floor <= 5), is executed five times and the inner loop only executed two times for each execution of the outer loop. This proves that the break statement only stops the execution of the inner loop where it's declared.

For

The for loop syntax in PHP is similar to C. For example to print 1 to 10 the for loop is like this

```
<?php
for ($i = 1; $i <= 10; $i++) {
    echo $i . '<br>';
}
?>
```

A more interesting function is to print this number in a table with alternating colors. Here is the code

Example : [alternate-colors.php](#)

Source : [alternate-colors.phps](#)

```
<table width="200" border="0" cellspacing="1" cellpadding="2">
<tr>
<td bgcolor="#CCCCFF">Alternating row colors</td>
</tr>
<?php
for ($i = 1; $i <= 10; $i++) {
    if ($i % 2) {
        $color = '#FFFFCC';
    } else {
        $color = '#CCCCCC';
    }
}
?>
<tr>
<td bgcolor="<?php echo $color; ?>"><?php echo $i; ?></td>
</tr>
<?php
}
?>
</table>
```

This code displays different row colors depending on the value of \$i. If \$i is not divisible by two it prints yellow otherwise it prints gray colored rows.

Using Functions

Real world applications are usually much larger than the examples above. It has been proven that the best way to develop and maintain a large program is to construct it from smaller pieces (functions) each of which is **more manageable** than the original program.

A function may be defined using syntax such as the following:

```
<?php
function addition($val1, $val2)
{
    $sum = $val1 + $val2;
    return $sum;
}
?>
```

Using Default Parameters

When calling a function you usually provide the same number of argument as in the declaration. Like in the function above you usually call it like this :

```
$result = addition(5, 10);
```

But you can actually call a function without providing all the arguments by using default parameters.

Example : [default-param.php](#)

Source code : [default-param.phps](#)

```
<?php

function repeat($text, $num = 10)
{
    echo "<ol>\r\n";
    for($i = 0; $i < $num; $i++)
    {
        echo "<li>$text </li>\r\n";
    }
    echo "</ol>";
}
```

```
// calling repeat with two arguments
repeat("I'm the best", 15);
```

```
// calling repeat with just one argument
repeat("You're the man");
```

?>

Function repeat() have two arguments \$text and \$num. The \$num argument has a default value of 10. The first call to repeat() will print the text 15 times because the value of \$num will be 15. But in the second call to repeat() the second parameter is omitted so repeat() will use the default \$num value of 10 and so the text is printed ten times.

Returning Values

Applications are usually a sequence of functions. The result from one function is then passed to another function for processing and so on. Returning a value from a function is done by using the return statement.

Example : [return.php](#)

Source code : [return.phps](#)

```
<?php
$myarray = array('php tutorial',
                 'mysql tutorial',
                 'apache tutorial',
                 'java tutorial',
                 'xml tutorial');

$rows = buildRows($myarray);
$table = buildTable($rows);

echo $table;

function buildRows($array)
{
    $rows = '<tr><td>' .
        implode('</td></tr><tr><td>', $array) .
        '</td></tr>';

    return $rows;
}

function buildTable($rows)
{
    $table = "<table cellpadding='1' cellspacing='1'          bgcolor='#FFCC00'
border='1'>$rows</table>";

    return $table;
}
?>
```

You can return any type from a function. An integer, double, array, object, resource, etc.

Notice that in `buildRows()` I use the built in function `implode()`. It joins all elements of `$array` with the string `'</td></tr><tr><td>'` between each element. I also use the `'.'` (dot) operator to concat the strings.

You can also write `buildRows()` function like this.

```
<?php
...

function buildRows($array)
{
    $rows = '<tr><td>';
    $n    = count($array);
    for($i = 0; $i < $n - 1; $i++)
    {
        $rows .= $array[$i] . '</td></tr><tr><td>';
    }

    $rows .= $array[$n - 1] . '</td></tr>';

    return $rows;
}

...
?>
```

Of course it is more convenient if you just use `implode()`.

Using Form

Using forms in a web based application is very common. Most forms are used to gather information like in a sign-up form, survey / polling, guestbook, etc.

A form can have the method set as post or get. When using a form with `method="post"` you can use `$_POST` to access the form values. And when the form is using `method="get"` you can use `$_GET` to access the values. The `$_REQUEST` superglobal can be used to access form values with `method="post"` and `method="get"` but it is recommended to use `$_POST` or `$_GET` instead so you will know from what method did the values come from.

Here is an example of HTML form :

Example : [form.php](#)

Source code : [form.phps](#)

```
<form method="post" action="<?php echo $_SERVER['PHP_SELF'];?>">
Name : <input name="username" type="text"><br>
Password : <input name="password" type="password"><br>
<input name="send" type="submit" value="Send!">
</form>
```

The form above use `$_SERVER['PHP_SELF']` as the action value. It is not required for the form to perform correctly but it's considered good programming practice to use it.

Below is the PHP code used to access form values :

Example : [form.php](#)

Source code : [form.phps](#)

```
<?php
if(isset($_POST['send']))
{
    echo "Accessing username using POST : " . $_POST['username'] . "<br>";
    echo "Accessing username using REQUEST : " . $_REQUEST['username'] . "<br>";

    $password = $_POST['password'];
    echo "Password is $password";
}
?>
```

The if statement is used to check if the send variable is set. If it is set then the form must have been submitted. The script then print the value of username using `$_POST` and `$_REQUEST`

Using Array As Form Values

Take a look at the code example below. The form have five input with the same name, `language[]`. Using the same input name is common for checkboxes or radio buttons.

Example : [form-array.php](#)

Source code : [form-array.phps](#)

```
<form method="post" action="<?php echo $_SERVER['PHP_SELF'];?>">
Select the programming languages you can use<br>
<input name="language[]" type="checkbox" value="C++">
C++<br>
<input name="language[]" type="checkbox" value="Java">
Java<br>
```

```

<input name="language[]" type="checkbox" value="PHP">
PHP<br>
<input name="language[]" type="checkbox" value="ASP">
ASP<br>
<input name="language[]" type="checkbox" value="Delphi">
Delphi<br>
<input name="send" type="submit" id="send" value="Send!">
</form>

```

The PHP code below print the value of language after the form is submitted. Go ahead and try the [example](#). Try checking and unchecking the options to see the effect.

Example : [form-array.php](#)

Source code : [form-array.phps](#)

```

<?php
if(isset($_POST['language']))
{
    $language = $_POST['language'];
    $n      = count($language);
    $i      = 0;

    echo "The languages you selected are \r\n" .
        "<ol>";
    while ($i < $n)
    {
        echo "<li>{ $language[$i] }</li> \r\n";
        $i++;
    }
    echo "</ol>";
}
?>

```

From the above code you will notice that \$language is an array. This is because in the form code language is repeated several times. When you specify the same input name in a form, PHP will treat it as an array.

There is one important issue you should know when using forms, that is form validation. The code above does not check whether the input is correct such as checking if the user is entering any value into the textbox or is the user selecting any checkboxes. This is actually a bad practice because you should never put a web form without any validation. To learn about validating form input you can go to this page : [Form Validation With PHP](#)

That's it. You just completed the basics of PHP programming. When you're ready for more advance programming in PHP check out the [book store](#). You can find plenty good books in there with deeper coverage on PHP programming

As is said earlier if you haven't got the PHP manual you should download it now from [php.net](#). You can find lots of interesting (and important) stuff in it. You can get the manual in various format but I recommend that you get the **compiled HTML (CHM)** version. It's easier to read plus it has search capability and loads of user contributed notes, very useful.

Now let's move on the next section : [MySQL Tutorial](#)

Reading a Remote File Using PHP

To read a remote file from php you have at least four options :

1. Use `fopen()`
2. Use `file_get_contents()`
3. CURL
4. Make your own function using php's socket functions.

First i need to warn you about something. You can only use `fopen()` and `file_get_contents()` when **fopen wrappers is enabled**. This parameter is specified in the `php.ini` file and cannot be changed at run time using `ini_set()`, To know whether you can use these two or not you can use the following code to check the value of `fopen` wrapper setting.

```
if (ini_get('allow_url_fopen') == '1') {  
    // use fopen() or file_get_contents()  
} else {  
    // use curl or your custom function  
}
```

1 . Using `fopen()`

If you use `fopen()` to read a remote file the process is as simple as reading from a local file. The only difference is that you will specify the URL instead of the file name. Take a look at the example below :

```
// make sure the remote file is successfully opened before doing anything else  
if ($fp = fopen('http://www.google.com/', 'r')) {  
    $content = "";  
    // keep reading until there's nothing left  
    while ($line = fread($fp, 1024)) {  
        $content .= $line;  
    }  
}
```

```

}

// do something with the content here
// ...
} else {
    // an error occurred when trying to open the specified url
}

```

Now, the code above use `fread()` function in the while loop to read up to 1024 bytes of data in a single loop. That code can also be written like this :

```

// make sure the remote file is successfully opened before doing anything else
if ($fp = fopen('http://www.google.com/', 'r')) {
    $content = "";
    // keep reading until there's nothing left
    while ($line = fgets($fp, 1024)) {
        $content .= $line;
    }

    // do something with the content here
    // ...
} else {
    // an error occurred when trying to open the specified url
}

```

instead of `fread()` we use `fgets()` which reads **one line** of data up to 1024 bytes. The first code is much more preferable than the second though. Just imagine if the remote file's size is 50 kilobytes and consists of 300 lines. Using the first code will cause the loop to be executed about fifty times but using the second the loop will be executed **three hundred times**.

If you consider the cost to call a function plus the time required to make 300 requests compared to just 5 then clearly the first one is the winner.

2. Using `file_get_contents()`

This is my favorite way of reading a remote file because it is very simple. Just call this function and specify a url as the parameter. But make sure you remember to check the return value first to determine if it return an error before processing the result

```

$content = file_get_contents('http://www.google.com/');
if ($content !== false) {
    // do something with the content
} else {
    // an error happened
}

```

```
}
```

3. **CURL**

Unlike the two methods above using CURL cannot be said as straightforward. Although this library is very useful to connect and communicate with many different protocols (not just http) it requires more effort to learn. And another problem is that not all web hosts have this library in their php installation. So we better make sure to check if the library is installed before trying to use it.

Here is a basic example on fetching a remote file

```
// make sure curl is installed
if (function_exists('curl_init')) {
    // initialize a new curl resource
    $ch = curl_init();

    // set the url to fetch
    curl_setopt($ch, CURLOPT_URL, 'http://www.google.com');

    // don't give me the headers just the content
    curl_setopt($ch, CURLOPT_HEADER, 0);

    // return the value instead of printing the response to browser
    curl_setopt($ch, CURLOPT_RETURNTRANSFER, 1);

    // use a user agent to mimic a browser
    curl_setopt($ch, CURLOPT_USERAGENT, 'Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.7.5) Gecko/20041107 Firefox/1.0');

    $content = curl_exec($ch);

    // remember to always close the session and free all resources
    curl_close($ch);
} else {
    // curl library is not installed so we better use something else
}
```

In some cases using CURL is faster than using `file_get_contents()` or `fopen()`. This is because CURL handles compression protocols by default (for example gzip). Many sites, big and small, use gzip compression to compress their web pages in order to save bandwidth. This site, for example, also uses gzip compression which cuts the bandwidth used into half. So if you're the type who just can't wait CURL will fit you most.

4. Custom functions

In the worst case your server will have fopen wrappers disabled and don't have CURL library installed. In this sad situation you just have to make your own way.

Our function shall be named `getRemoteFile()` which takes only one parameter, the url for the remote file. The skeleton for this function is shown below

```
function getRemoteFile($url)
{
    // 1. get the host name and url path

    // 2. connect to the remote server

    // 3. send the necessary headers to get the file

    // 4. retrieve the response from the remote server

    // 5. strip the headers

    // 6. return the file content
}
```

To extract the host name and url path from the given url we'll use `parse_url()` function. When given a url this function will spit out the followings :

- scheme
- host
- port
- user
- pass
- path
- query
- fragment

For example, if the url is `http://www.php-mysql-tutorial.com/somepage.php` then `parse_url()` will return :

```
Array
(
    [scheme] => http
    [host]   => www.php-mysql-tutorial.com
    [path]   => /somepage.php
)
```

and if the url is `http://myusername:mypassword@www.php-mysql-tutorial.com/somepage.php?q=whatsthis#ouch` then `parse_url()` will return this :

```
Array
(
```

```

[scheme] => http
[host]   => www.php-mysql-tutorial.com
[user]   => myusername
[pass]   => mypassword
[path]   => /somepage.php
[query]  => q=whatsthis
[fragment] => ouch
)

```

For our new function we only care about the host, port, path and query.

To establish a connection to a remote server we use `fsockopen()`. This function requires five arguments, the hostname, port number, a reference for error number, a reference for the error message and timeout

```

function getRemoteFile($url)
{
    // get the host name and url path
    $parsedUrl = parse_url($url);
    $host = $parsedUrl['host'];
    if (isset($parsedUrl['path'])) {
        $path = $parsedUrl['path'];
    } else {
        // the url is pointing to the host like http://www.mysite.com
        $path = '/';
    }

    if (isset($parsedUrl['query'])) {
        $path .= '?' . $parsedUrl['query'];
    }

    if (isset($parsedUrl['port'])) {
        $port = $parsedUrl['port'];
    } else {
        // most sites use port 80
        $port = '80';
    }

    $timeout = 10;
    $response = "";

    // connect to the remote server
    $fp = @fsockopen($host, $port, $errno, $errstr, $timeout );

    if( !$fp ) {
        echo "Cannot retrieve $url";
    } else {
        // send the necessary headers to get the file
        fputs($fp, "GET $path HTTP/1.0\r\n" .

```

```

        "Host: $host\r\n" .
        "User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.1; en -US; rv:1.8.0.3)
Gecko/20060426 Firefox/1.5.0.3\r\n" .
        "Accept: */*\r\n" .
        "Accept-Language: en-us,en;q=0.5\r\n" .
        "Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7\r\n" .
        "Keep-Alive: 300\r\n" .
        "Connection: keep-alive\r\n" .
        "Referer: http://$host\r\n\r\n");

```

```

// retrieve the response from the remote server
while ( $line = fread( $fp, 4096 ) ) {
    $response .= $line;
}

fclose( $fp );

// strip the headers
$pos    = strpos($response, "\r\n\r\n");
$response = substr($response, $pos + 4);
}

// return the file content
return $response;
}

```

The code above sends nine lines of headers but only the first two is mandatory. So even if you send only these

```

fputs($fp, "GET $path HTTP/1.0\r\n" .
        "Host: $host\r\n\r\n");

```

the function will likely be working correctly. Not always though. Since the file is stored in a remote server It really up to that server to reply to your request or not. Some people code their page to block any request without the proper referer header. Some will only accept a specific user agent. Other will require cookies set in the header.

If you want to see what headers should be sent to successfully fetch a specific remote file try using [firefox](#) plus [live http headers plugin](#). It's really a useful little tool

MySQL Tutorial

This mysql tutorial covers the basic stuff that you can do with MySQL. But before we go any further make sure you already have MySQL installed, if you don't have it installed che ck out this page :

[Installing PHP and MySQL \(plus Apache\).](#)

This part of tutorial covers the following :

- [Starting MySQL](#)
How to start mysql server and client from the command line (DOS)
- [Adding New Users To MySQL](#)
How to add new MySQL users. Skip this part if you don't intend to add more user other than the default
- [Create a new database](#)
- [Create tables](#)
- [Insert data](#)
- [Get the data](#)
- [Update & Delete data](#)

Starting MySQL

Before starting the MySQL client make sure the server is turned on. If you run Windows 9x start the **mysqld.exe** or if you run Windows 2000/XP run the **mysqld-nt.exe**



```
C:\WINDOWS\System32\cmd.exe - mysqld-nt --c...
C:\Apache\mysql\bin>mysqld-nt --console
mysqld-nt: ready for connections
```

The --console option tells mysqld-nt not to remove the console window. if you are running Windows NT/2000/XP it's better to install MySQL as a service. That way mysql server will be automatically started when you start Windows.

Use mysqld-nt --install to install mysqld as a service and mysqld-nt --remove to remove mysqld from the service list

Once the server is on, open another DOS window and type mysql, you should see something like this

C:\>**mysql**

Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 3 to server version: 4.0.18 -nt

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql>

Or if you need to specify a user name and password you can start mysql like this :

C:\>**mysql -h localhost -u root -p**

Enter password:

Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 4 to server version: 4.0.18 -nt

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql>

You can also specify the database you want to use. If you already have a database named petstore you can start mysql as

C:\>**mysql petstore**

Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 5 to server version: 4.0.18 -nt

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql>

Once you're done you can end mysql client by typing **quit** or **exit** at the mysql> prompt

mysql> **exit**

Bye

C:\>

Note : If you get this kind of error message when trying to run mysql from the DOS window

C:\>mysql

'mysql' is not recognized as an internal or external command,
operable program or batch file.

that means you haven't set the path to mysql bin directory. To solve the problem you can add the following line to your autoexec.bat file :

```
path=%path%;c:\mysql\bin
```

assuming that you install MySQL in c:\mysql. Or if you're using Windows XP you can go to :

Start->Settings->Control Panel->System->Advanced->Environment Variables

Chose Edit on the System Variables section and add c:\mysql\bin to the path environment variable.

Add New MySQL User

For adding a new user to MySQL you just need to add a new entry to user table in database mysql. Below is an example of adding new user phpcake with SELECT, INSERT and UPDATE privileges with the password mypass the SQL query is :

```
mysql> use mysql;  
Database changed
```

```
mysql> INSERT INTO user (host, user, password, select_priv, insert_priv, update_priv) VALUES  
( 'localhost', 'phpcake', PASSWORD('mypass'), 'Y', 'Y', 'Y');  
Query OK, 1 row affected (0.20 sec)
```

```
mysql> FLUSH PRIVILEGES;  
Query OK, 1 row affected (0.01 sec)
```

```
mysql> SELECT host, user, password FROM user WHERE user = 'phpcake';  
+-----+-----+-----+  
| host   | user   | password      |  
+-----+-----+-----+  
| localhost | phpcake | 6f8c114b58f2ce9e |  
+-----+-----+-----+  
1 row in set (0.00 sec)
```

When adding a new user remember to encrypt the new password using P ASSWORD() function provided by MySQL. As you can see in the above example the password mypass is encrypted to 6f8c114b58f2ce9e.

Notice the the FLUSH PRIVILEGES statement. This tells the server to reload the grant tables. If you don't use it then you won't be able to connect to mysql using the new user account (at least until the server is reloaded).

You can also specify other privileges to a new user by setting the values of these columns in user table to 'Y' when executing the INSERT query :

- Select_priv
- Insert_priv
- Update_priv
- Delete_priv
- Create_priv
- Drop_priv
- Reload_priv
- Shutdown_priv
- Process_priv
- File_priv
- Grant_priv
- References_priv
- Index_priv
- Alter_priv

I think you can guess what those privileges serve by reading it's name

Create New MySQL Database

You need to use mysqladmin to create MySQL database. The command is simple just write **mysqladmin** in a dos window followed by the database name you want to create

```
C:\>mysqladmin create petstore
```

```
C:\>mysql
```

Welcome to the MySQL monitor. Commands end with ; or \g.

Your MySQL connection id is 3 to server version: 4.0.18 -nt

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

```
mysql> SHOW databases;
```

```
+-----+
| Database |
+-----+
| mysql   |
| petstore |
| test    |
+-----+
```

2 rows in set (0.00 sec)

mysql>

You can also type the query in mysql> prompt like this

```
mysql> CREATE database petstore;  
Query OK, 1 row affected (0.00 sec)
```

To show available databases in mysql use the command `show databases` on mysql> prompt. Now use the database by typing **USE petstore** and then type **SHOW tables** to see what tables are available in the database

```
mysql> USE petstore;  
Database changed  
mysql> SHOW tables;  
Empty set (0.00 sec)
```

Next I will show you how to create table in mysql database

Create New Table

For this example we'll create two tables. The first one describe the species of animals available in a pet store and the second will store the data of a pet in the store. The table names will be **species** and **pet**

The species table will consist of the id and the animal species, and the pet table will consist of animal id, species, sex, and price

```
mysql> CREATE TABLE species (id INT NOT NULL AUTO_INCREMENT, species varchar(30)  
NOT NULL, primary key(id));  
Query OK, 0 rows affected (0.02 sec)
```

```
mysql> CREATE TABLE pet(id INT NOT NULL AUTO_INCREMENT, sp_id INT NOT NULL,  
sex CHAR(1) NOT NULL, price DECIMAL(4,2) NOT NULL, primary key(id));  
Query OK, 0 rows affected (0.03 sec)
```

```
mysql> SHOW tables;  
+-----+  
| Tables_in_petstore |  
+-----+  
| pet                |
```

```
| species      |
+-----+
2 rows in set (0.00 sec)
```

mysql> **DESCRIBE species;**

```
+-----+-----+-----+-----+-----+
| Field | Type      | Null | Key | Default | Extra      |
+-----+-----+-----+-----+-----+
| id    | int(11)   |      | PRI | NULL    | auto_increment |
| name  | varchar(30) |      |     |         |               |
+-----+-----+-----+-----+-----+
2 rows in set (0.05 sec)
```

mysql> **DESC pet;**

```
+-----+-----+-----+-----+-----+
| Field | Type      | Null | Key | Default | Extra      |
+-----+-----+-----+-----+-----+
| id    | int(11)   |      | PRI | NULL    | auto_increment |
| sp_id | int(11)   |      |     | 0        |               |
| sex   | char(1)   |      |     |          |               |
| price | decimal(4,2) |      |     | 0.00     |               |
+-----+-----+-----+-----+-----+
4 rows in set (0.00 sec)
```

The SQL syntax to create table is : **CREATE TABLE** <tablename> (<list of fields>)

The DESCRIBE or DESC statement is used to show a description of a table. You can also use EXPLAIN or SHOW COLUMNS

mysql> **EXPLAIN pet;**

```
+-----+-----+-----+-----+-----+
| Field | Type      | Null | Key | Default | Extra      |
+-----+-----+-----+-----+-----+
| id    | int(11)   |      | PRI | NULL    | auto_increment |
| sp_id | int(11)   |      |     | 0        |               |
| sex   | char(1)   |      |     |          |               |
| price | decimal(4,2) |      |     | 0.00     |               |
+-----+-----+-----+-----+-----+
4 rows in set (0.00 sec)
```

mysql> **SHOW COLUMNS FROM pet;**

```
+-----+-----+-----+-----+-----+
| Field | Type      | Null | Key | Default | Extra      |
+-----+-----+-----+-----+-----+
| id    | int(11)   |      | PRI | NULL    | auto_increment |
| sp_id | int(11)   |      |     | 0        |               |
```

sex	char(1)					
price	decimal(4,2)			0.00		

4 rows in set (0.00 sec)

Insert Data To MySQL

You can insert data to the tables directly from `mysql>` prompt or by loading a file containing the data. The values will be tab separated one line represents one record.

To insert the data directly from mysql use the INSERT statement. The format for INSERT is `INSERT INTO <table name> (column1, column2, ...) values ('value1', 'value2', ...)`. For example to insert a species name into the table species the command is like this :

```
mysql> INSERT INTO species (name) values ('Cat');
Query OK, 1 row affected (0.00 sec)
```

Remember to put single quotes around a value it is a string.

Notice that i don't have to set the value of id because id have AUTO_INCREMENT attribute. Whenever you insert a new record to the table the value of id is set automatically by mysql with increasing values. The AUTO_INCREMENT attribute is commonly used to create unique identity for new rows

Next example will show how to insert data from a text file, you can get the file [here](#) and try in on your computer.

```
mysql> LOAD DATA LOCAL INFILE "insert.txt" INTO TABLE species;
Query OK, 3 rows affected (0.00 sec)
```

You can also run SQL queries directly from the DOS prompt. Assuming insert.txt is in C:\ you can run the query in insert.txt like this

```
C:\mysql < insert.txt
```

Get Data From MySQL

Retrieving the table data is easy, just use the SELECT statement like this

```
mysql> SELECT * FROM species;
```

```
+----+-----+
| id | name |
+----+-----+
| 1 | Cat  |
| 2 | Bird |
| 3 | Fish |
| 4 | Turtle |
+----+-----+
4 rows in set (0.00 sec)
```

The * from the SELECT statement means select all columns. If you only want the names you can write

```
mysql> SELECT name FROM species;
```

```
+-----+
| name |
+-----+
| Cat  |
| Bird |
| Fish |
| Turtle |
+-----+
4 rows in set (0.00 sec)
```

To select only the records that interest you, you can use WHERE statement followed by the definition. For example to select a record from species table where id equals 4 you can do this :

```
mysql> SELECT * FROM species WHERE id = 4;
```

```
+----+-----+
| id | name |
+----+-----+
| 4 | Turtle |
+----+-----+
1 row in set (0.59 sec)
```

If you want to order the returned rows by a criteria you can use ORDER BY like this :

```
mysql> SELECT * FROM species ORDER BY name;
```

```
+----+-----+
| id | name |
+----+-----+
```



```
| 2 | Bird |
| 1 | Cat |
| 3 | Fish |
| 4 | Turtle |
+----+-----+
4 rows in set (0.00 sec)
```

By default the result is sorted in ascending order. So this query will give the same result :

```
mysql> SELECT * FROM species ORDER BY name ASC;
+----+-----+
| id | name |
+----+-----+
| 2 | Bird |
| 1 | Cat |
| 3 | Fish |
| 4 | Turtle |
+----+-----+
4 rows in set (0.00 sec)
```

The ASC means ascending order. To get a descending order you just change the ASC with DESC like this :

```
mysql> SELECT * FROM species ORDER BY name DESC;
+----+-----+
| id | name |
+----+-----+
| 4 | Turtle |
| 3 | Fish |
| 1 | Cat |
| 2 | Bird |
+----+-----+
4 rows in set (0.00 sec)
```

MySQL Update And Delete

The statement UPDATE is used to change the value in a table. For example to change to species name from Turtle to Dog

```
mysql> UPDATE species SET name = 'Dog' WHERE name = 'Turtle';
```

Query OK, 1 row affected (0.02 sec)
Rows matched: 1 Changed: 1 Warnings: 0

```
mysql> SELECT * FROM species;
```

```
+----+-----+  
| id | name |  
+----+-----+  
| 1 | Cat  |  
| 2 | Bird |  
| 3 | Fish |  
| 4 | Dog  |  
+----+-----+
```

4 rows in set (0.00 sec)

Delete Data

The DELETE statement deletes rows from a table that satisfy the condition given by the WHERE clause. For example to delete a record from species table where id equals 1

```
mysql> DELETE FROM species WHERE id = 1;
```

Query OK, 1 row affected (0.00 sec)

```
mysql> SELECT * FROM species;
```

```
+----+-----+  
| id | name |  
+----+-----+  
| 2 | Bird |  
| 3 | Fish |  
| 4 | Dog  |  
+----+-----+
```

3 rows in set (0.01 sec)

Now that you already know the basics of MySQL it's time to continue the tutorial. You will start learning how to [connect to MySQL database using PHP](#).

By the way this is a good time for you to download the [MySQL reference manual](#) (if you haven't done so). It covers in detail about everything you need to know about MySQL. As with the PHP manual you should download the compiled HTML version of MySQL manual because it's easier to browse than other formats.

Connect to MySQL Database

Opening a connection to MySQL database from PHP is easy. Just use the `mysql_connect()` function like this

```
<?php
$dbhost = 'localhost';
$dbuser = 'root';
$dbpass = 'password';
```

```
$conn = mysql_connect($dbhost, $dbuser, $dbpass) or die           ('Error connecting to  
mysql');
```

```
$dbname = 'petstore';
mysql_select_db($dbname);
?>
```

`$dbhost` is the name of MySQL server. When your webserver is on the same machine with the MySQL server you can use `localhost` or `127.0.0.1` as the value of `$dbhost`. The `$dbuser` and `$dbpass` are valid MySQL user name and password. For adding a user to MySQL visit this page : [MySQL Tutorial](#)

Don't forget to select a database using `mysql_select_db()` after connecting to mysql. If no database selected your query to select or update a table will not work.

Sometimes a web host will require you to specify the MySQL server name and port number. For example if the MySQL server name is `db.php-mysql-tutorial.com` and the port number is `3306` (the default port number for MySQL) then you can modify the above code to :

```
<?php
$dbhost = 'db.php-mysql-tutorial.com:3306';
$dbuser = 'root';
$dbpass = 'password';
```

```
$conn = mysql_connect($dbhost, $dbuser, $dbpass) or die           ('Error connecting to  
mysql');
```

```
$dbname = 'petstore';
mysql_select_db($dbname);
?>
```

It's a common practice to place the routine of opening a database connection in a separate file. Then

everytime you want to open a connection just include the file. Usually the host, user, password and database name are also separated in a configuration file.

An example of config.php that stores the connection configuration and opendb.php that opens the connection are :

Source code : [config.php](#) , [opendb.php](#)

```
<?php
// This is an example of config.php
$dbhost = 'localhost';
$dbuser = 'root';
$dbpass = 'password';
$dbname = 'phpcake';
?>
```

```
<?php
// This is an example opendb.php
$conn = mysql_connect($dbhost, $dbuser, $dbpass) or die      ('Error connecting to
mysql');
mysql_select_db($dbname);
?>
```

So now you can open a connection to mysql like this :

```
<?php
include 'config.php';
include 'opendb.php';

// ... do something like insert or select, etc

?>
```

Closing the Connection

The connection opened in a script will be closed as soon as the execution of the script ends. But it's better if you close it explicitly by calling `mysql_close()` function. You could also put this function call in a file named `closedb.php`.

Source code : [closedb.phps](#)

```
<?php
// an example of closedb.php
// it does nothing but closing
// a mysql database connection
```

```
mysql_close($conn);
?>
```

Now that you have put the database configuration, opening and closing routines in separate files your PHP script that uses mysql would look something like this :

```
<?php
include 'config.php';
include 'opendb.php';

// ... do something like insert or select, etc

include 'closedb.php';
?>
```

Create MySQL Database With PHP

To create a database use the `mysql_query()` function to execute an SQL query like this

```
<?php
include 'config.php';
include 'opendb.php';

$query = "CREATE DATABASE phpcake";
$result = mysql_query($query);

include 'closedb.php';
```

?>

Please note that the query **should not** end with a semicolon.

PHP also provide a function to create MySQL database, `mysql_create_db()`. This function is deprecated though. It is better to use `mysql_query()` to execute an SQL CREATE DATABASE statement instead like the above example.

If you want to create MySQL database using PHP `mysql_create_db()` function you can do it like this :

```
<?php
include 'config.php';
include 'opendb.php';

mysql_create_db('phpcake');

include 'closedb.php';
?>
```

If you want to create tables in the database you just created don't forget to call `mysql_select_db()` to access the new database.

Note: some webhosts require you to create a MySQL database and user through your website control panel (such as CPanel). If you get an error when trying to create database this might be the case.

Creating the Tables

To create tables in the new database you need to do the same thing as creating the database. First create the SQL query to create the tables then execute the query using `mysql_query()` function.

Example : `contact.php`

Source code : [contact.phps](#)

```
<?php
include 'config.php';
include 'opendb.php';

$query = 'CREATE DATABASE phpcake';
$result = mysql_query($query);

mysql_select_db('phpcake') or die('Cannot select database');

$query = 'CREATE TABLE contact( '
        'cid INT NOT NULL AUTO_INCREMENT, '
        'cname VARCHAR(20) NOT NULL, '
        'cemail VARCHAR(50) NOT NULL, '
        'csubject VARCHAR(30) NOT NULL, '
        'cmessage TEXT NOT NULL, '
        'PRIMARY KEY(cid))';

$result = mysql_query($query);

include 'closedb.php';
?>
```

Of course when you need to create lots of tables it's a good idea to read the query from a file then save in `$query` variable instead of writing the query in your script.

```
<?php
include 'config.php';
include 'opendb.php';

$queryFile = 'myquery.txt';

$fp = fopen($queryFile, 'r');
$query = fread($fp, filesize($queryFile));
fclose($fp);
$result = mysql_query($query);

include 'closedb.php';
```

?>

Deleting a Database

As with creating a database, it is also preferable to use `mysql_query()` and to execute the SQL DROP DATABASE statement instead of using `mysql_drop_db()`

```
<?php
include 'config.php';
include 'opendb.php';

// ... do something here

$query = 'DROP DATABASE phpcake';
$result = mysql_query($query);

// ... probably do something here too

include 'closedb.php';
?>
```

After the database and the tables are ready it's time to put something into the database.

Insert Data To MySQL Database

Inserting data to MySQL is done by using `mysql_query()` to execute INSERT query. Note that the query string **should not** end with a semicolon. Below is an example of adding a new MySQL user by inserting a new row into table `user` in database `mysql` :

Example : `insert.php`

Source code : [insert.php](#)

```
<?php
include 'library/config.php';
include 'library/opendb.php';

mysql_select_db($mysql);
$query = "INSERT INTO user (host, user, password, select_priv, insert_priv, update_priv)
VALUES ('localhost', 'phpcake', PASSWORD('myp ass'), 'Y', 'Y', 'Y')";

mysql_query($query) or die('Error, insert query failed');
```



```
$query = "FLUSH PRIVILEGES";  
mysql_query($query) or die('Error, insert query failed');
```

```
include 'library/closedb.php';  
?>
```

In the above example `mysql_query()` was followed by `die()`. If the query fail the error message will be printed and the script's execution is terminated. Actually you can use `die()` with any function that might not execute properly. That way you can be sure that the script won't continue to run when an error occurred.

In a real application the values of an `INSERT` statement will be form values. As a safe precaution always escape the values using `addslashes()` if `get_magic_quotes_gpc()` returns false. Below is an example of using form values with `INSERT`. It's the same as above except that the new username and password are taken from `$_POST` :

Example : `adduser.php`

Source code : [adduser.phps](#)

```
<html>  
<head>  
<title>Add New MySQL User</title>  
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">  
</head>  
  
<body>  
<?php  
if(isset($_POST['add']))  
{  
include 'library/config.php';  
include 'library/opendb.php';  
  
$username = $_POST['username'];  
$password = $_POST['password'];  
  
$query = "INSERT INTO user (host, user, password, select_priv, insert_priv, update_priv)  
VALUES ('localhost', '$username', PASSWORD('$password'), 'Y', 'Y', 'Y')";  
mysql_query($query) or die('Error, insert query failed');  
  
$query = "FLUSH PRIVILEGES";  
mysql_query($query) or die('Error, insert query failed');  
  
include 'library/closedb.php';  
echo "New MySQL user added";  
}  
else
```

```

{
?>
<form method="post">
<table width="400" border="0" cellspacing="1" cellpadding="2">
<tr>
<td width="100">Username</td>
<td><input name="username" type="text" id="username"></td>
</tr>
<tr>
<td width="100">Password</td>
<td><input name="password" type="text" id="password"></td>
</tr>
<tr>
<td width="100">&nbsp;</td>
<td>&nbsp;</td>
</tr>
<tr>
<td width="100">&nbsp;</td>
<td><input name="add" type="submit" id="add" value="Add New User"></td>
</tr>
</table>
</form>
<?php
}
?>
</body>
</html>

```

Get Data From MySQL Database

Using PHP you can run a MySQL SELECT query to fetch the data out of the database . You have several options in fetching information from MySQL. PHP provide several functions for this. The first one is `mysql_fetch_array()` which fetch a result row as an associative array, a numeric array, or both.

Below is an example of fetching data from MySQL, the table contact have three columns, name, subject and message.

Example : `select.php`

Source code : [select.phps](#), [contact.txt](#)

```

<?php
include 'config.php';

```

```

include 'opendb.php';

$query = "SELECT name, subject, message FROM contact";
$result = mysql_query($query);

while($row = mysql_fetch_array($result, MYSQL_ASSOC))
{
    echo "Name : { $row['name']} <br>" .
        "Subject : { $row['subject']} <br>" .
        "Message : { $row['message']} <br><br>";
}

include 'closedb.php';
?>

```

The while() loop will keep fetching new rows until mysql_fetch_array() returns FALSE, which means there are no more rows to fetch. The content of the rows are assigned to the variable \$row and the values in row are then printed. Always remember to put curly brackets when you want to insert an array value directly into a string.

In above example I use the constant MYSQL_ASSOC as the second argument to mysql_fetch_array(), so that it returns the row as an associative array. With an associative array you can access the field by using their name instead of using the index . Personally I think it's more informative to use \$row['subject'] instead of \$row[1].

PHP also provide a function called mysql_fetch_assoc() which also return the row as an associative array.

```

<?php
include 'config.php';
include 'opendb.php';

$query = "SELECT name, subject, message FROM contact";
$result = mysql_query($query);

while($row = mysql_fetch_assoc($result))
{
    echo "Name : { $row['name']} <br>" .
        "Subject : { $row['subject']} <br>" .
        "Message : { $row['message']} <br><br>";
}

include 'closedb.php';
?>

```

You can also use the constant MYSQL_NUM, as the second argument to mysql_fetch_array(). This

will cause the function to return an array with numeric index.

```
<?php
include 'config.php';
include 'opendb.php';

$query = "SELECT name, subject, message FROM contact";
$result = mysql_query($query);

while($row = mysql_fetch_array($result, MYSQL_NUM))
{
    echo "Name :{ $row[0]} <br>" .
        "Subject : { $row[0]} <br>" .
        "Message : { $row[0]} <br><br>";
}

include 'closedb.php';
?>
```

Using the constant MYSQL_NUM with mysql_fetch_array() gives the same result as the function mysql_fetch_row().

There is another method for you to get the values from a row. You can use list(), to assign a list of variables in one operation.

```
<?php
include 'config.php';
include 'opendb.php';

$query = "SELECT name, subject, message FROM contact";
$result = mysql_query($query);

while(list($name,$subject,$message)= mysql_fetch_row($result))
{
    echo "Name :$name <br>" .
        "Subject : $subject <br>" .
        "Message : $row <br><br>";
}

include 'closedb.php';
?>
```

In above example, list() assign the values in the array returned by mysql_fetch_row() into the variable \$name, \$subject and \$message.

Of course you can also do it like this

```
<?php
include 'config.php';
include 'opendb.php';

$query = "SELECT name, subject, message FROM contact";
$result = mysql_query($query);

while($row = mysql_fetch_row($result))
{
    $name = $row[0];
    $subject = $row[1];
    $message = $row[2];

    echo "Name :$name <br>" .
        "Subject : $subject <br>" .
        "Message : $row <br><br>";
}

include 'closedb.php';
?>
```

So you see you have lots of choices in fetching information from a database. Just choose the one appropriate for your program

Freeing the memory ?

In some cases a query can return large result sets. As this results are stored in memory there's a concern about memory usage. However you do not need to worry that you will have to call this function in all your script to prevent memory congestion. In PHP all results memory is automatically freed at the end of the script's execution.

But you are really concerned about how much memory is being used for queries that return large result sets you can use `mysql_free_result()`. Calling this function will free all memory associated with the result identifier (`$result`).

Using the above example you can call `mysql_free_result()` like this :

```
<?php
include 'config.php';
```

```
include 'opendb.php';

$query = "SELECT name, subject, message FROM contact";
$result = mysql_query($query);

while($row = mysql_fetch_row($result))
{
    ...
}

mysql_free_result($result);

include 'closedb.php';
?>
```

Convert MySQL Query Result To Excel

Using PHP to convert MySQL query result to Excel format is also common especially in web based finance applications. The finance data stored in database are downloaded as Excel file for easy viewing. There is no special functions in PHP to do the job. But you can do it easily by formatting the query result as tab separated values or put the value in an HTML table . After that set the content type to application/vnd.ms-excel

Example : [convert.php](#)

Source : [convert.php](#), [students.txt](#)

```
<?php
include 'library/config.php';
include 'library/opendb.php';

$query = "SELECT fname, lname FROM students";
$result = mysql_query($query) or die('Error, query failed');

$tsv = array();
$html = array();
while($row = mysql_fetch_array($result, MYSQL_NUM))
{
    $tsv[] = implode("\t", $row);
    $html[] = "<tr><td>" . implode("</td><td>", $row) . "</td></tr>";
}

$tsv = implode("\r\n", $tsv);
$html = "<table>" . implode("\r\n", $html) . "</table>";

$fileName = 'mysql-to-excel.xls';
header("Content-type: application/vnd.ms-excel");
header("Content-Disposition: attachment; filename=$fileName");

echo $tsv;
//echo $html;

include 'library/closedb.php';
?>
```

In the above example \$tsv is a string containing tab separated values and \$html contain an HTML table. I use implode() to join the values of \$row with tab to create a tab separated string.

After the while loop implode() is used once again to join the rows using newline characters. The headers are set and the value of \$tsv is then printed. This will force the browser to save the file as

mysql-to-excel.xsl

Try running the script in your own computer then try commenting `echo $tsv` and uncomment `echo $html` to see the difference.

Next, how to split your query result to multiple pages by applying paging.

Using Paging

Ever use a Search Engine? I'm sure you have, lots of time. When Search Engines found thousands of results for a keyword do they spit out all the result in one page? Nope, they use paging to show the result little by little.

Paging means showing your query result in **multiple pages** instead of just put them all in one long page. Imagine waiting for five minutes just to load a search page that shows 1000 result. By splitting the result in multiple pages you can save download time plus you don't have much scrolling to do.

To show the result of a query in several pages first you need to know **how many rows you have** and **how many rows per page** you want to show. For example if I have 295 rows and I show 30 rows per page that mean I'll have ten pages (rounded up).

For the example I created a table named `randoms` that store 295 random numbers. Each page shows 20 numbers.

Example: [paging.php](#)

Source code : [paging.phps](#)

```
<?php
include 'library/config.php';
include 'library/opendb.php';

// how many rows to show per page
$rowsPerPage = 20;

// by default we show first page
$pageNum = 1;

// if $_GET['page'] defined, use it as page number
if(isset($_GET['page']))
{
    $pageNum = $_GET['page'];
```



```

}

// counting the offset
$offset = ($pageNum - 1) * $rowsPerPage;

$query = " SELECT val FROM randoms " .
        " LIMIT $offset, $rowsPerPage";
$result = mysql_query($query) or die('Error, query failed');

// print the random numbers
while($row = mysql_fetch_array($result))
{
    echo $row['val'] . '<br>';
}

// ... more code here
?>

```

Paging is implemented in MySQL using LIMIT that take two arguments. The first argument specifies the offset of the first row to return, the second specifies the maximum number of rows to return. The offset of the first row is 0 (not 1).

When [paging.php](#) is called for the first time the value of \$_GET['page'] is not set. This caused \$pageNum value to remain 1 and the query is :

```
SELECT val FROM randoms LIMIT 0, 20
```

which returns the first 20 values from the table. But when paging.php is called like this <http://www.php-mysql-tutorial.com/examples/paging/paging.php?page=4> the value of \$pageNum becomes 4 and the query will be :

```
SELECT val FROM randoms LIMIT 60, 20
```

this query returns rows 60 to 79.

After showing the values we need to print the links to show any pages we like. But first we have to count the number of pages. This is achieved by dividing the number of total rows by the number of rows to show per page :

```

$maxPage = ceil($numrows/$rowsPerPage);
<?php
// ... the previous code

// how many rows we have in database
$query = "SELECT COUNT(val) AS numrows FROM randoms";
$result = mysql_query($query) or die('Error, query failed');

```

```

$row = mysql_fetch_array($result, MYSQL_ASSOC);
$numrows = $row['numrows'];

// how many pages we have when using paging?
$maxPage = ceil($numrows/$rowsPerPage);

// print the link to access each page
$self = $_SERVER['PHP_SELF'];
$nav = "";

for($page = 1; $page <= $maxPage; $page++)
{
    if ($page == $pageNum)
    {
        $nav .= " $page "; // no need to create a link to current page
    }
    else
    {
        $nav .= " <a href=\"\$self?page=$page\">$page</a> ";
    }
}

// ... still more code coming
?>

```

The mathematical function `ceil()` is used to round up the value of `$numrows/$rowsPerPage`.

In this case the value of total rows `$numrows` is 295 and `$rowsPerPage` is 20 so the result of the division is 14.75 and by using `ceil()` we get `$maxPage = 15`

Now that we know how many pages we have we make a loop to print the link. Each link will look something like this:

```
<a href="paging.php?page=5">5</a>
```

You see that we use `$_SERVER['PHP_SELF']` instead of `paging.php` when creating the link to point to the paging file. This is done to avoid the trouble of modifying the code in case we want to change the filename.

We are almost complete. Just need to add a little code to create a 'Previous' and 'Next' link. With these links we can navigate to the previous and next page easily. And while we at it let's also create a 'First page' and 'Last page' link so we can jump straight to the first and last page when we want to.

```

<?php
// ... the previous code

// creating previous and next link
// plus the link to go straight to
// the first and last page

if ($pageNum > 1)
{
    $page = $pageNum - 1;
    $prev = " <a href=\"\$self?page=$page\">[Prev]</a> ";

    $first = " <a href=\"\$self?page=1\">[First Page]</a> ";
}
else
{
    $prev = '&nbsp;'; // we're on page one, don't print previous link
    $first = '&nbsp;'; // nor the first page link
}

if ($pageNum < $maxPage)
{
    $page = $pageNum + 1;
    $next = " <a href=\"\$self?page=$page\">[Next]</a> ";

    $last = " <a href=\"\$self?page=$maxPage\">[Last Page]</a> ";
}
else
{
    $next = '&nbsp;'; // we're on the last page, don't print next link
    $last = '&nbsp;'; // nor the last page link
}

// print the navigation link
echo $first . $prev . $nav . $next . $last;

// and close the database connection
include '../library/closedb.php';

// ... and we're done!
?>

```

Making these navigation link is actually easier than you may think. When we're on the fifth page we just make the 'Previous' link point to the fourth. The same principle also apply for the 'Next' link, we

just need to add one to the page number.

One thing to remember is that we don't need to print the 'Previous' and 'First Page' link when we're already on the first page. Same thing for the 'Next' and 'Last' link. If we do print them that would only confuse the one who click on it. Because we'll be giving them the exact same page.

We got a problem here...

Take a look at [this slightly modified version](#) of paging.php. Instead of showing 20 numbers in a page, I decided to show just three.

See the problem already?

Those page numbers are running across the screen! Yuck!

This call for a little modification to the code. Instead of printing the link to each and every page we will just saying something like "Viewing page 4 of 99 pages".

Than means we havel remove these code :

```
<?php
// ... the previous code

$nav = "";

for($page = 1; $page <= $maxPage; $page++)
{
    if ($page == $pageNum)
    {
        $nav .= " $page "; // no need to create a link to current page
    }
    else
    {
        $nav .= " <a href=\"\$self?page=$page\">$page</a> ";
    }
}

// ... the rest here
?>
```

And then modify this one

```
<?php
// ...

// print the navigation link
echo $first . $prev . $nav . $next . $last;

// ...
?>
```

Into this

```
<?php
// ...

// print the navigation link
echo $first . $prev .
" Showing page $pageNum of $maxPage pages " . $next . $last;

// ...
?>
```

Take a look at the at the result [here](#), and you can get the source code [here](#)

Using Paging (Part 2)

When there's more than one column involved in paging there isn't much that we need to modify. We only need to decide how to count the total number of rows we have in the table. Consider the student table. This table have five columns as shown in the SQL below.

Source : [examples/source/student.sql](#)

```
CREATE TABLE student(
  id INTEGER UNSIGNED NOT NULL AUTO_INCREMENT,
  name VARCHAR(30) NOT NULL,
  address VARCHAR(50) NOT NULL,
  age TINYINT UNSIGNED NOT NULL,
  register_date DATE NOT NULL,

  PRIMARY KEY (id)
);
```

In the select query we just select all the columns. You can also use `SELECT *` instead of mentioning all the column names (`SELECT id, name, address, age, register_date`). But personally i prefer writing the column names in the query so that by reading the code i know what the column names in a table without having to check the database.

Example: [paging4.php](#)

Source code : [paging4.phps](#)

```
<?php
include 'library/config.php';
include 'library/opendb.php';

// how many rows to show per page
$rowsPerPage = 3;

// by default we show first page
$pageNum = 1;

// if $_GET['page'] defined, use it as page number
if(isset($_GET['page']))
{
    $pageNum = $_GET['page'];
}

// counting the offset
$offset = ($pageNum - 1) * $rowsPerPage;

$query = "SELECT id, name, address, age, register_date
        FROM student
        LIMIT $offset, $rowsPerPage";
$result = mysql_query($query) or die('Error, query failed');

// print the student info in table
echo '<table border="1"><tr><td>Student
Id</td><td>Name</td><td>Address</td><td>Age</td><td>Register Date</td></tr>';
while(list($id, $name, $address, $age, $regdate) = mysql_fetch_array($result))
{
    echo "<tr><td>$id</td><td>$name</td><td>$address</td>
    <td>$age</td><td>$regdate</td></tr>";
}
echo '</table>';
echo '<br>';

// ... more code here
?>
```

In this example we print the result in table. Before looping through the array we just echo the starting table code and the header which displays the column names. Then in the loop we just print the values in a HTML table row.

The next thing is finding out the total number of rows. There are several ways to do it. The first one

is shown below. It's the same method used in previous examples. We just use the COUNT() function

Example: [paging4.php](#)

Source code : [paging4.phps](#)

```
<?php
// ... previous code here

// how many rows we have in database
$query = "SELECT COUNT(id) AS numrows FROM student";
$result = mysql_query($query) or die('Error, query failed');
$row = mysql_fetch_array($result, MYSQL_ASSOC);
$numrows = $row['numrows'];

// ... just the same code that prints the prev & next link
?>
```

You can also count any other columns since they all yield the same result. So your query can be rewritten into this :

```
<?php
// ...
$query = "SELECT COUNT(name) AS numrows FROM student";
// ...
?>
```

Or this :

```
<?php
// ...
$query = "SELECT COUNT(age) AS numrows FROM student";
// ...
?>
```

There is another way to count the total rows. Instead of using COUNT() function in the query you use a simple SELECT <column> and use mysql_num_rows() to see how many rows returned.

Take a look at the code below. We now separate the query into two parts. One is the normal SELECT query and the second is the SQL that performs the paging. After finish printing the result you can reuse the first part of the query to find the total number of rows.

Example: [paging5.php](#)

Source code : [paging5.phps](#)

```
<?php
```



```
// ... same old code to get the page number and counting the offset

$query = "SELECT id, name, address, age, register_date
        FROM student ";



$pagingQuery = "LIMIT $offset, $rowsPerPage";


$result = mysql_query($query . $pagingQuery) or die('Error, query failed');

// ... the code that prints the result in a table

// how many rows we have in database
$result = mysql_query($query) or die('Error, query failed');
$numrows = mysql_num_rows($result);

// ... and here is the code that print the prev & next links
?>
```

There is another advantage in separating the original query with the paging query. In case you only wish to list all student whose age is older than 15. You just need to modify the original query and you don't have to worry about changing the query to find the total number of rows. The example is shown below :

```
<?php
// ... same old code to get the page number and counting the offset

$query = "SELECT id, name, address, age, register_date
        FROM student
        WHERE age > 15";

$pagingQuery = "LIMIT $offset, $rowsPerPage";
$result = mysql_query($query . $pagingQuery) or die('Error, query failed');

// ... the code that prints the result in a table

// how many rows we have in database
$result = mysql_query($query) or die('Error, query failed');
$numrows = mysql_num_rows($result);

// ... and here is the code that print the prev & next links
?>
```

The disadvantage of this method is that the second execution of `mysql_query()` will retrieve all columns from the database. This is very useless since we're not going to use them. We only interested in finding the total rows returned by that query. In the end it's really up to you to decide

which method you prefer.

To get all the source for this paging examples [click here](#)

MySQL Update and Delete

There are no special ways in PHP to perform update and delete on MySQL database. You still use `mysql_query()` to execute the UPDATE or DELETE statement.

For instance to update a password in mysql table for username phpcake can be done by executing an UPDATE statement with `mysql_query()` like this:

Example : `update.php`

Source code : [update.phps](#)

```
<?php
include 'library/config.php';
include 'library/opendb.php';

mysql_select_db('mysql')
or die('Error, cannot select mysql database');
```

```
$query = "UPDATE user SET password = PASSWORD('newpass')". "WHERE user = 'phpcake'";
```

```
mysql_query($query) or die('Error, query failed');
include 'library/closedb.php';
?>
```

There is one important thing that you should be aware of when updating and deleting rows from database. That is **data integrity**.

If you're using **InnoDB** tables you can leave the work of maintaining data integrity to MySQL . However when you're using other kind of tables you need to enforce the data integrity manually.

To make sure that your update and delete queries will not break the data integrity. You have to make appropriate update and delete queries for all tables referencing to the table you update or delete.

For example, suppose you have two tables, Class and Student. The Student table have a foreign key column, `cid` which references to the `class_id` column in table Class. When you want to update a `class_id` in Class table you will also need to update the `cid` column in Student table to maintain data integrity.

Suppose i want to change the `class_id` of Karate from 3 to 10. Since there is a row in Student table

with cid value of 3, I have to update that row too.

```
$query = "UPDATE Class SET class_id = 10 WHERE class_id = 3";
```

```
mysql_query($query);
```

```
$query = "UPDATE Student SET cid = 10 WHERE cid = 3";
```

```
mysql_query($query);
```

Below are the data in Table and Student class before an update query :

Table Class

class_id	class_name
1	Silat
2	Kungfu
3	Karate
4	Taekwondo

Table Student

student_id	student_name	cid
1	Uzumaki Naruto	1
2	Uchiha Sasuke	3
3	Haruno Sakura	2

Now the content of Table and Student class after the update query are :

Table Class

class_id	class_name
1	Silat
2	Kungfu
10	Karate
4	Taekwondo

Table Student

student_id	student_name	cid
1	Uzumaki Naruto	1
2	Uchiha Sasuke	10
3	Haruno Sakura	2

You can go as far as creating your own functions in PHP to ensure the data integrity. I have done this before and I hope you don't do it. Save yourself the headache and just write appropriate queries to maintain your data integrity whenever you update / delete rows from a table.

This means that whenever you make a query to update / delete **always** consult your database design

to see if you need to update / delete another table to maintain data integrity. Your code will be more portable like this.

Using LOCK TABLES

When your web application is used by more than one user using LOCK TABLES before any update / delete query is a safe bet. This will make sure that only one user change the table at a time.

Using the above update code examples again, suppose there are two users. The first one want to update one row in Class table and the second want to delete it

```
$query = "LOCK TABLES Class WRITE, Student WRITE";  
mysql_query($query);  
$query = "DELETE FROM Class WHERE class_id = 3";  
mysql_query($query);  
$query = "DELETE FROM Student WHERE class_id = 3";  
mysql_query($query);
```

```
$query = "UNLOCK TABLES";  
mysql_query($query);
```

The update queries above can be rewritten as :

```
$query = "LOCK TABLES Class WRITE, Student WRITE";  
mysql_query($query);  
  
$query = "UPDATE Class SET class_id = 10 WHERE class_id = 3";  
mysql_query($query);  
$query = "UPDATE Student SET cid = 10 WHERE cid = 3";  
mysql_query($query);  
  
$query = "UNLOCK TABLES";  
mysql_query($query);
```

By issuing the LOCK TABLES all other users are blocked from reading and writing to the tables. So you're update / delete query will continue to completion without any worries that the intended table already changed by another user

Using PHP To Backup MySQL Database

There are at least three ways to backup your MySQL Database :

1. Execute a database backup query from PHP file.
2. Run mysqldump using system() function.
3. Use phpMyAdmin to do the backup.

Execute a database backup query from PHP file

Below is an example of using SELECT INTO OUTFILE query for creating table backup :

```
<?php
include 'config.php';
include 'opendb.php';

$tableName = 'mypet';
$backupFile = 'backup/mypet.sql';
$query = "SELECT * INTO OUTFILE '$backupFile' FROM $tableName";
$result = mysql_query($query);

include 'closedb.php';
?>
```

To restore the backup you just need to run LOAD DATA INFILE query like this :

```
<?php
include 'config.php';
include 'opendb.php';

$tableName = 'mypet';
$backupFile = 'mypet.sql';
$query = "LOAD DATA INFILE '$backupFile' INTO TABLE $tableName";
$result = mysql_query($query);

include 'closedb.php';
?>
```

It's a good idea to name the backup file as tablename.sql so you'll know from which table the backup file is

Run mysqldump using system() function

The system() function is used to execute an external program. Because MySQL already have built in

tool for creating MySQL database backup (mysqldump) let's use it from our PHP script

```
<?php
include 'config.php';
include 'opendb.php';

$backupFile = $dbname . date("Y-m-d-H-i-s") . '.gz';
$command = "mysqldump --opt -h $dbhost -u $dbuser -p $dbpass $dbname | gzip > $backupFile";
system($command);

include 'closedb.php';
?>
```

Use phpMyAdmin to do the backup

This option as you may guessed doesn't involve any programming on your part. However I think i mention it anyway so you know more options to backup your database.

To backup your MySQL database using phpMyAdmin click on the "export" link on phpMyAdmin main page. Choose the database you wish to backup, check the appropriate SQL options and enter the name for the backup file.

Form Validation With PHP

Whenever you make a form you should not leave it alone without any form validation. Why? Because there is no guarantee that the input is correct and processing incorrect input values can make your application give unpredictable result.

You can validate the form input on two places, **client side** and **server side**.

Client side form validation usually done with javascript. Client side validation makes your web application respond '**faster**' while server side form validation with PHP can act as a **backup** just in case the user switch off javascript support on her browser. And since different browsers can behave differently there is always a possibility that the browser didn't execute the javascript code as you intended.

Some things you need to check :

- empty values
- numbers only
- input length

- email address
- strip html tags

To show form validation with php in action I'll use the contact form in this website. Click [here](#) to see the contact form and then take a look at the [source code](#).

This contact form requires four input :

- sender name
- sender email
- message subject
- message body

First let's focus on the client side validation. On the "Send Message" button I put this javascript code : `onClick="return checkForm();"`, which is triggered when you click on it. Clicking the button will run the function `checkForm()`. Every input is checked to see whether they are valid input. When an invalid input is found the function returns false so the form is not submitted. When you insert valid input the function will return true and the form is submitted.

Go ahead and play around with the form. Try entering only spaces for the input value or enter gibberish string as email address.

The code snippet below shows the client part of contact form.

Example : [contact.php](#)

Source code : [contact.phps](#)

```
<html>
<head>
<title>Contact Form</title>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
<style type="text/css">
```

// CSS goes here

```
</style>
<script language="JavaScript">
function checkForm()
{
    var cname, cemail, csubject, cmessage;
    with(window.document.msgform)
    {
        cname    = sname;
        cemail    = email;
        csubject  = subject;
        cmessage  = message;
    }
}
```

```

if(trim(cname.value) == "")
{
    alert('Please enter your name');
    cname.focus();
    return false;
}
else if(trim(cemail.value) == "")
{
    alert('Please enter your email');
    cemail.focus();
    return false;
}
else if(!isEmail(trim(cemail.value)))
{
    alert('Email address is not valid');
    cemail.focus();
    return false;
}
else if(trim(csubject.value) == "")
{
    alert('Please enter message subject');
    csubject.focus();
    return false;
}
else if(trim(cmessage.value) == "")
{
    alert('Please enter your message');
    cmessage.focus();
    return false;
}
else
{
    cname.value = trim(cname.value);
    cemail.value = trim(cemail.value);
    csubject.value = trim(csubject.value);
    cmessage.value = trim(cmessage.value);
    return true;
}
}

```

```

function trim(str)
{
    return str.replace(/^\s+|\s+$/g,'');
}

```



```

function isEmail(str)
{
    var regex = /^[_-.a-z0-9]+@(([_-.a-z0-9]+\.)+(ad|ae|aero|af|ag|
ai|al|am|an|ao|aq|ar|arpa|as|at|au|aw|az|ba|bb|bd|be|bf|bg|
bh|bi|biz|bj|bm|bn|bo|br|bs|bt|bv|bw|by|bz|ca|cc|cd|cf|cg|
ch|ci|ck|cl|cm|cn|co|com|coop|cr|cs|cu|cv|cx|cy|cz|de|dj|dk|
dm|do|dz|ec|edu|ee|eg|eh|er|es|et|eu|fi|fj|fk|fm|fo|fr|ga|gb|
gd|ge|gf|gh|gi|gl|gm|gn|gov|gp|gq|gr|gs|gt|gu|gw|gy|hk|hm|hn|
hr|ht|hu|id|ie|il|in|info|int|io|iq|ir|is|it|jm|jo|jp|ke|kg|
kh|ki|km|kn|kp|kr|kw|ky|kz|la|lb|lc|li|lk|lr|ls|lt|lu|lv|ly|
ma|mc|md|mg|mh|mil|mk|ml|mm|mn|mo|mp|mq|mr|ms|mt|mu|museum|
mv|mw|mx|my|mz|na|name|nc|ne|net|nf|ng|ni|nl|no|np|nr|nt|nu|
nz|om|org|pa|pe|pf|pg|ph|pk|pl|pm|pn|pr|pro|ps|pt|pw|py|qa|
re|ro|ru|rw|sa|sb|sc|sd|se|sg|sh|si|sj|sk|sl|sm|sn|so|sr|st|
su|sv|sy|sz|tc|td|tf|tg|th|tj|tk|tm|tn|to|tp|tr|tt|tv|tw|tz|
ua|ug|uk|um|us|uy|uz|va|vc|ve|vg|vi|vn|v|u|wf|ws|ye|yt|yu|za|
zm|zw)|((([0-9][0-9]?[0-1][0-9][0-9]|[2][0-4][0-9]|[2][5][0-5]))\.){3}([0-9][0-9]?[0-1][0-9][0-9]|[2][0-4][0-9]|[2][5][0-5]))$/i;

```

```

return regex.test(str);

```

```

}
</script>
</head>
<body>
<form method="post" name="msgform">
<table width="500" border="0" align="center" cellpadding="2" cellspacing="1" class="maincell">
<tr>
<td width="106">Your Name</td>
<td width="381"><input name="sname" type="text" class="box" id="sname" size="30"></td>
</tr>
<tr>
<td>Your Email</td>
<td>
<input name="email" type="text" class="box" id="email" size="30">
</td></tr>
<tr>
<td>Subject</td>
<td><input name="subject" type="text" class="box" id="subject" size="30"></td>
</tr>
<tr>
<td>Message</td>
<td><textarea name="message" cols="55" rows="10" wrap="OFF" class="box"
id="message"></textarea></td>
</tr>
<tr align="center">
<td colspan="2"><input name="send" type="submit" class="bluebox" id="send" value="Send

```

```

Message" onClick="return checkForm();" ></td>
</tr>
<tr align="center">
<td colspan="2">&nbsp;   </td>
</tr>
</table>
</form>
</body>
</html>

```

Now we'll take a better look at checkForm() function :

```

function checkForm()
{
  var cname, cemail, csubject, cmessage;
  with(window.document.msgform)
  {
    cname   = sname;
    cemail  = email;
    csubject = subject;
    cmessage = message;
  }

  // ... the rest of the code

}

```

In the beginning of the function I use the keyword **var** to declare four variables to reference the form input . They are cname, cemail, csubject and cmessage. These variables will reference the form input sname, email, subject and message respectively.

Javascript treats a document and it's element as object. The message form is named msgform so to access it we use window.document.msgform and to access the sname input text we can use window.document.msgform.sname.

To avoid the hassle of writing the window.document.msgform part whenever we want to access a form object I use the **with**() keyword. Without it the checkForm() function would look like :

```

function checkForm()
{
  var cname, cemail, csubject, cmessage;

  cname   = window.document.msgform.sname;

```

```
cemail = window.document.msgform.email;  
csubject = window.document.msgform.subject;  
cmessage = window.document.msgform.message;
```

```
// ... the rest of the code
```

```
}
```

Next we'll validate each form input.

```
function checkForm()  
{  
  // variable declarations goes here ...  
  
  if(trim(cname.value) == "")  
  {  
    alert('Please enter your name');  
    cname.focus();  
    return false;  
  }  
  else if(trim(cemail.value) == "")  
  {  
    alert('Please enter your email');  
    cemail.focus();  
    return false;  
  }  
  else if(!isEmail(trim(cemail.value)))  
  {  
    alert('Email address is not valid');  
    cemail.focus();  
    return false;  
  }  
  // The rest of validation code goes here ...  
}
```

To access the value of the name input box we use `cname.value`. The name value is trimmed to remove extra spaces from the beginning and end of the name. If you do not enter your name or only entering spaces then an alert box will pop up. Using `cname.focus()` the cursor will be placed to the name input box and then `checkForm()` return false which cancel the form submit.

The code above uses `trim()` function. This is not a built in javascript function. I can't understand why there is no `trim()` function in javascript, even VBScript has it. Anyway it's not a big deal because we can just make our own `trim()` function. The solution here uses **regular expression** to replace any

spaces in the beginning and end of a string with blank string.

```
function trim(str)
{
    return str.replace(/^s+|s+$/g,"");
}
```

The forward slash (/) is used to create a regular expression. Note that it **is not** a string, you don't have to use quotes and it won't work if you use quotes. Let's chop the regular expression notation so we can understand it better :

- ^ : the beginning of a string
- \$: end of string.
- \s : single whitespace character (tab also count as whitespace)
- + : one or more
- | : conditional (OR)
- g : global, mainly used for search and replace operation

So in english the search replace function above can be read as :

"Replace one or more whitespace character from the beginning or ending of a string with blank character"

As for the email input, we need to double check it. First, check if the email is entered and second check if the input is in a valid email format. For the second check we'll use isEmail() function. This function also uses regular expression.

A valid email format can be described as :

[a string consisting of alphanumeric characters, underscores, dots or dash] @ ([a valid domain name] DOT [a valid TLD]) OR [a valid IP address]

In case you're wondering TLD means Top Level Domain such as com, net, org, biz, etc.

When you see the [source code](#) you will see that the regular expression in isEmail() function is actually written in one line. I have to break them into multiple lines just to fit the space. The PHP Manual explains the regular expression syntax for PHP in depth, but if you want to learn regular expression for javascript you can go to : <http://www.regular-expressions.info>

Finally, if all input are considered valid checkForm() returns true and the form will be submitted. This will set the \$_POST['send'] variable and now we start validating the input on the server side using PHP.

```
<?php
```

```
$errmsg = ""; // error message
```

```

$name = ""; // sender's name
$email = ""; // sender's email address
$subject = ""; // message subject
$message = ""; // the message itself

if(isset($_POST['send']))
{
    $name = $_POST['name'];
    $email = $_POST['email'];
    $subject = $_POST['subject'];
    $message = $_POST['message'];

    if(trim($name) == "")
    {
        $errmsg = 'Please enter your name';
    }
    else if(trim($email) == "")
    {
        $errmsg = 'Please enter your email address';
    }
    else if(!isEmail($email))
    {
        $errmsg = 'Your email address is not valid';
    }
    else if(trim($subject) == "")
    {
        $errmsg = 'Please enter message subject';
    }
    else if(trim($message) == "")
    {
        $errmsg = 'Please enter your message';
    }

    // ... more code here
?>

```

The PHP validation is doing the same thing as the javascript validation. It check each value to see if it's empty and if it is we consider that as an error. We also recheck the validity of the email address.

When we find an error we set the value of \$errmsg. We will print this value so the user can fix the error.

If everything is okay the value of \$errmsg will be blank. So we continue processing the input.

```
<?php
```

```
// ... previous validation code

if($errmsg == '')
{
    if(get_magic_quotes_gpc())
    {
        $subject = stripslashes($subject);
        $message = stripslashes($message);
    }

    $to = "email@yourdomain.com";
    $subject = "[Contact] : ' . $subject;
    $msg = "From : $sname \r\n " . $message;
    mail($to,
        $subject,
        $msg,
        "From: $email\r\nReturn-Path: $email\r\n");

// ... more code here
?>
```

Some web host set the PHP directive `magic_quotes_gpc` to 'on' which runs `addslashes()` to every GET, POST, and COOKIE data so we got an extra work to strip the slashes from the input.

Because the `addslashes()` function only add slashes before single quote ('), double quote ("), backslash (\) and NULL, we only need to worry about the `$subject` and `$message`. This is because (usually) only these two can contain such characters. However, we can't be sure if `magic_quotes_gpc` is On or Off so we have to check it's value first using the `get_magic_quotes_gpc()` function

After finishing all that boring job of validating the input we finally come to the last, and the most important step, sending the message using the `mail()` function.

The first parameter we pass to the `mail()` function is the receiver's email address. The second is the email subject. The third is the message itself and the fourth is an additional headers.

I'm sure you already understand the purpose of the first three parameters so I'll just discuss about the fourth one, the additional parameter (additional headers)

```
"From: $email\r\nReply-To: $email\r\nReturn-Path: $email\r\n"
```

Each headers are separated by the `"\r\n"` (newline) characters. The first two (From and Reply -To) is self explanatory. But what about the third one (Return -Path)?

The reason is some spam filter will check the Return -Path header and compare it with the From header. If these two don't match then the email is considered as spam and your email won't get

delivered (or sent to the spam folder). So it's better to play safe and put Return -Path header when we want to send an email to make sure it gets delivered.

That's it. I hope this tutorial can give you a clear idea on validating form, both on client side and server side. But if doesn't, then just use the [contact form](#) and let me know about it :-)

Creating A Guestbook Using PHP and MySQL

You've seen it at least once right? Guestbook is one of the most common thing to find in a website. In this tutorial we'll create a guestbook using PHP and MySQL.

I have split this tutorial into two section, each covering a specific feature of the guestbook.

- [Creating The Sign-Guestbook Form](#)
This part will cover creating the database tables, the guestbook form and the process of saving the entry to database
- [Viewing The Entries](#)
You want to see the guestbook entries of course. This section covers fetching the entries from database and put into an HTML table. You will also learn to show the entries in multiple pages using MySQL paging.

I think you should take a quick look what the finished guestbook look like. Just click [here](#) to see it.

Creating The Sign-Guestbook Form

We start by creating the table to store the data, guestbook. There are six fields in the guestbook table:

1. id : the unique identifier for an entry in the guestbook
2. name : the visitor's name
3. email : visitor's email address
4. url : visitor's website url, if she has one
5. message : the message
6. entry_date : when did this entry added

I have put the SQL query needed to create the table in [guestbook.txt](#).

Below is the HTML form code. It's pretty simple, we have text box for name, email and url plus a textarea to hold the message. The submit button is attached with a javascript function because we want to check the input values before the page is submitted.

Example : [guestbook.php](#)

Source code : [guestbook.phps](#), [guestbook.txt](#)

```
<form method="post" name="guestform">
<table width="550" border="0" cellpadding="2" cellspacing="1">
<tr>
<td width="100">Name *</td> <td>
<input name="txtName" type="text" size="30" maxlength="30"></td>
</tr>
<tr>
<td width="100">Email</td>
<td>
<input name="txtEmail" type="text" size="30" maxlength="50"></td>
</tr>
<tr>
<td width="100">Website URL</td>
<td>
<input name="txtUrl" type="text" value="http://" size="30" maxlength="50"></td>
</tr>
<tr>
<td width="100">Message *</td> <td>
<textarea name="mtxMessage" cols="80" rows="5"></textarea></td>
</tr>
<tr>
<td width="100">&nbsp;</td>
<td>
<input name="btnSign" type="submit" value="Sign Guestbook" onClick="return
checkForm();"></td>
</tr>
</table>
</form>
```

Below is the javascript code to check the input form. The checkForm() function is called when the "Sign Guestbook" button is clicked.

The mandatory fields are name and message so if either is empty we pop an alert box to tell the visitor to enter the name and message. Email is not a mandatory field so we only check if in an email address is entered but we won't complain if there's none .

```
function checkForm()
{
    // the variables below are assigned to each
    // form input
```



```

var gname, gemail, gurl, gmessage;

with(window.document.guestform)
{
    gname  = txtName;
    gemail = txtEmail;
    gurl   = txtUrl;
    gmessage = mtxMessage;
}

// if name is empty alert the visitor
if(trim(gname.value) == "")
{
    alert('Please enter your name');
    gname.focus();
    return false;
}
// alert the visitor if email is empty or
// if the format is not correct
else if(trim(gemail.value) != "" && !isEmail(trim(gemail.value)))
{
    alert('Please enter a valid email address or leave it blank');
    gemail.focus();
    return false;
}
// alert the visitor if message is empty
else if(trim(gmessage.value) == "")
{
    alert('Please enter your message');
    gmessage.focus();
    return false;
}
else
{
    // when all input are correct
    // return true so the form will submit
    return true;
}
}

/*
Strip whitespace from the beginning and end of a string
*/
function trim(str)
{
    return str.replace(/^s+|s+$/g,"");
}

```

```

}

/*
Check if a string is in valid email format.
*/
function isEmail(str)
{
var regex = /^[-_a-z0-9]+@(([-a-z0-9]+\.)+(ad|ae|aero|af|ag|
ai|al|am|an|ao|aq|ar|arpa|as|at|au|aw|az|ba|bb|bd|be|bf|bg|bh|
bi|biz|bj|bm|bn|bo|br|bs|bt|bv|bw|by|bz|ca|cc|cd|cf|cg|ch|ci|
ck|cl|cm|cn|co|com|coop|cr|cs|cu|cv|cx|cy|cz|de|dj|dk|dm|do|dz|
ec|edu|ee|eg|eh|er|es|et|eu|fi|fj|fk|fm|fo|fr|ga|gb|gd|ge|gf|gh|
gi|gl|gm|gn|gov|gp|gq|gr|gs|gt|gu|gw|gy|hk|hm|hn|hr|ht|hu|id|ie|
il|in|info|int|io|iq|ir|is|it|jm|jo|jp|ke|kg|kh|ki|km|kn|kp|kr|
kw|ky|kz|la|lb|lc|li|lk|lr|ls|lt|lu|lv|ly|ma|mc|md|mg|mh|mil|mk|
ml|mm|mn|mo|mp|mq|mr|ms|mt|mu|museum|mv|mw|mx|my|mz|na|name|nc|
ne|net|nf|ng|ni|nl|no|np|nr|nt|nu|nz|om|org|pa|pe|pf|pg|ph|pk|
pl|pm|pn|pr|pro|ps|pt|pw|py|qa|re|ro|ru|rw|sa|sb|sc|sd|se|sg|sh|
si|sj|sk|sl|sm|sn|so|sr|st|su|sv|sy|sz|tc|td|tf|tg|th|tj|tk|tm|
tn|to|tp|tr|tt|tv|tw|tz|ua|ug|uk|um|us|uy|uz|va|vc|ve|vg|vi|vn|
vu|wf|ws|ye|yt|yu|za|zm|zw)|(([0-9][0-9]?[0-1][0-9][0-9][2]
[0-4][0-9][2][5][0-5])\.){3}([0-9][0-9]?[0-1][0-9][0-9][2]
[0-4][0-9][2][5][0-5]))$/i;
return regex.test(str);
}

```

After the form is submitted our job turns to saving the input into the database.

In the code below I include [config.php](#) and [opendb.php](#) which contain the database configuration and the code needed to open a connection to MySQL. It's a good practice to put these actions in separate file. That way everytime you need to connect to MySQL you can include these files instead of rewriting the code. Also you can change the database information from just one file instead of changing it in every file that use MySQL. To see what the content of config.php, opendb.php and closedb.php go to : [Connecting to MySQL database](#)

```

<?php

include 'library/config.php';
include 'library/opendb.php';

if(isset($_POST['btnSign']))
{
    include 'library/config.php';
    include 'library/opendb.php';
}

```

```

$name  = trim($_POST['txtName']);
$email = trim($_POST['txtEmail']);
$url   = trim($_POST['txtUrl']);
$message = trim($_POST['mtxMessage']);

if(!get_magic_quotes_gpc())
{
    $message = addslashes($message);
}

// if the visitor do not enter the url
// set $url to an empty string
if ($url == 'http://')
{
    $url = "";
}

$query = "INSERT INTO guestbook (name,
                                email,
                                url,
                                message,
                                entry_date)
VALUES ('$name',
        '$email',
        '$url',
        '$message',
        current_date)";

mysql_query($query) or die('Error, query failed');

header('Location: ' . $_SERVER['REQUEST_URI']);
exit;
}
?>

```

The script check if the `$_POST['btnSign']` variable is set. If it is then the "Sign Guestbook" button must have been clicked and now we can read name, email, url and message from the `$_POST` global variable. After that we create an INSERT query string and execute the query using `mysql_query()`.

Sometimes a message can contain single quotes, we need to escape these single quotes (replacing it with `\`) otherwise MySQL will think that it's the end of a string and the query will fail. We use the `addslashes()` function to escape the string.

Unfortunately some web hosts set the `magic_quotes_gpc` setting on. This will make values

containing single-quotes in \$_GET, \$_POST and \$_COOKIE will be automatically escaped. If we use addslashes() when the string is already escaped the result would be a mess.

To check if magic_quotes_gpc is On use get_magic_quotes_gpc(). If it returns true then we don't have to call addslashes().

Ok, now after all input is ready we can build the query string to enter the name, email, url, message and entry date. Note that for the entry_date field we use current_date. This is not a PHP variable or function, it's a built in MySQL function that returns (guess what?) the current date.

You also see that I didn't explicitly insert the value of id field. This is because id is set as auto_increment so when we insert a new row into the table a new value for id is automatically generated (incremented for each new row).

After inserting the new guestbook entry the next thing we do is redirect back to current page using header('Location: ' . \$_SERVER['REQUEST_URI']);

Why?

The redirect is just to prevent double submission. Suppose we don't use the redirect and the visitor hit the refresh button after signing up the guestbook then the form will be submitted again.

Note : If you get this kind of error message

Warning: Cannot modify header information - headers already sent by (output started at C:\webroot\guestbook\library\config.php:7) in C:\webroot\guestbook\guestbook.php on line 43

this mean the redirect failed because you already sent something to the browser. I got the error message above because i "accidentally" have a space right after the closing tag (?>) in [config.php](#). By removing this space the error is fixed.

This kind of error is actually very common to see when your code is sending headers and fixing it is easy like the example above. Just check the file pointed by the error message and see if you accidentally sent (print) anything to the browser.

Pheww, we just finished the first part of our guestbook. Now it's time to create the script which will show the guestbook entries. We'll also try to split the entries into multiple pages when they are too much to be shown in one page.

Creating A Guestbook Using PHP and MySQL (Part 2)

Welcome to the second part of this guestbook tutorial. In case you haven't read the first section then go [here](#) to read it.

In this second part we'll add some code to our previous guestbook script which will allow us to view the entries. Without further ado let's start working on it.

Viewing the entries

Example : [guestbook.php](#)

Source code : [guestbook.phps](#)

```
<?php
include 'library/config.php';
include 'library/opendb.php';

// ... the code to save guestbook entries

}
?>
<html>
<head>
<title>Guestbook</title>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
<script language="JavaScript">

// ... the rest of javascript code goes here

</script>
</head>
<body>

<!-- this is where we put the guestbook form -->

<?php

// prepare the query string
$query = "SELECT id,
            name,
            email,
            url,
            message,
            DATE_FORMAT(entry_date, '%d.%m.%Y') ".
"FROM guestbook ".
"ORDER BY id DESC ";

$result = mysql_query($query) or die('Error, query failed');

// if the guestbook is empty show a message
if(mysql_num_rows($result) == 0)
{
?>
```

```

<p><br><br>Guestbook is empty </p>
<?php
}
else
{
// get the entries
while($row = mysql_fetch_array($result))
{
// list() is a convenient way of assign a list of variables
// from an array values
list($id, $name, $email, $url, $message, $date) = $row;

// change all HTML special characters,
// to prevent some nasty code injection
$name = htmlspecialchars($name);
$message = htmlspecialchars($message);

// convert newline characters to HTML break tag ( <br> )
$message = nl2br($message);
?>
<table width="550" border="1" cellpadding="2" cellspacing="0">
<tr>
<td width="80" align="left">
<a href="mailto:<?=$email;?>"> <?=$name;?> </a> </td>
<td align="right"><small><?=$date;?></small></td>
</tr>
<tr>
<td colspan="2"> <?=$message;?>
<?php

if($url != "")
{
// make the url clickable by formatting it as HTML link
$url = "<a href='$url' target='_blank'>$url</a>";
?>
<br> <small>Homepage : <?=$url;?></small>
<?php
}
?>
</td>
</tr>
</table>
<br>
<?php
} // end while

```

When you just created the guestbook, there are no entry in guestbook table. We use

mysql_num_rows() to check how many guestbook entries we have. If mysql_num_rows() returns 0 that means the table is empty and we can print a message saying that the guestbook is empty.

If there are already entries in the guestbook we then loop to get all the rows. I use list() to extract the values of a row into the variables \$id, \$name, \$email, \$url and \$message.

An additional step is needed for the \$name and \$message. For these two we use htmlspecialchars() before printing their value. This function will convert all special characters to HTML entities.

As an example suppose I enter the string `I am a wizard` in the message textarea. After applying htmlspecialchars() it will be converted to `I am a wizard`

What's the point of using htmlspecialchars()?

Well, the answer is because some people may try to abuse your guestbook. Some will enter a simple HTML bold formatted message like the example above but some may even try to input a javascript code in the message. As an example I could enter a script like this :

```
<script>
while(true)
{
    window.open("http://www.google.com");
}
</script>
```

If I don't use htmlspecialchars() and show it as is then when we view the guestbook entries this code will continuously open a new window of www.google.com. Won't do any harm if you have a popup blocker ready. But for those unlucky people who haven't got it installed will have their desktop filled with new windows in no time. Very annoying indeed.

One more thing added for \$message. We also use the function nl2br() to convert any newline characters (that's \r OR \n OR both) into HTML break tags (
). Because web browser "ignores" newline characters, we need nl2br() to preserve the message formatting. This way if you explicitly enter a three line message it will also be shown as a three line message.

Ok, now that we have the values ready we just need to put them in the HTML table. In above example I use `<?=$name;?>` to print the value of \$name. I can also use `<?php echo $name; ?>`, but it's easier to use the first form.

Now we're one step closer to finishing the guestbook. We just need to add a little more code for paging. Surely you don't want to show all the entries in one page. If you have a hundred entries the page will take forever to load. So let's add that little code to split the result into multiple pages.

Showing the entries in multiple pages

Example : [guestbook.php](#)

Source code : [guestbook.phps](#)

```
<?php

// how many guestbook entries to show per page
$rowsPerPage = 10;

// by default we show first page
$pageNum = 1;

if(isset($_GET['page']))
{
    $pageNum = $_GET['page'];
}

$offset = ($pageNum - 1) * $rowsPerPage;

// prepare the query string
$query = "SELECT id,
            name,
            email,
            url,
            message,
            DATE_FORMAT(entry_date, '%d.%m.%Y') ".
"FROM guestbook ".
"ORDER BY id DESC ".
"LIMIT $offset, $rowsPerPage";

// ... the rest of the code
?>
```

First we set how many entries we want to show per page (`$rowsPerPage`). We will use this value with the `LIMIT` keyword in our query so the query will only get a chunk of all entries available.

The logic flow is like this. When the page is first loaded the `$_GET['page']` is not yet initialized so we use the default `$pageNum = 1`. We then use `$pageNum` to count the offset (the index of the first result we want to show).

As an example, if `$pageNum = 1`, `$offset` will be $(1 - 1) * 10 = 0$. Our limit query will be `"LIMIT 0, 10"`. This will select the first ten entries from our guestbook table.

Another example . When `$pageNum = 3`, `$offset = 20`, limit query is `"LIMIT 20, 10"` which will

select ten result starting from the 20th index

Now that we have the query ready we need to create the navigation link so our visitor can easily move from the first page to other pages. We simply print the page number as a hyperlink. So when a visitor click on a page number the script will show the entries for the specified page.

The code needed is shown below

```
<?php
// .... previous code

$query = "SELECT COUNT(id) AS numrows FROM guestbook";
$result = mysql_query($query) or die('Error, query failed');
$row = mysql_fetch_array($result, MYSQL_ASSOC);
$numrows = $row['numrows'];

$maxPage = ceil($numrows/$rowsPerPage);
$nextLink = "";

if($maxPage > 1)
{
    $self = $_SERVER['PHP_SELF'];

    $nextLink = array();

    for($page = 1; $page <= $maxPage; $page++)
    {
        $nextLink[] = "<a href=\"\$self?page=$page\">$page</a>";
    }

    $nextLink = "Go to page : " . implode(' &raquo; ', $nextLink);
}

include 'library/closedb.php';
?>
<table width="550" border="0" cellpadding="2" cellspacing="0">
<tr>
<td align="right" class="text">
<?=$nextLink;?>
</td>
</tr>
</table>
<?php
}
?>
```

First we count the total number of entries we have (\$numrows) then we find the maximum page

numbers. To do this we just need the `ceil()` function to round the number up.

For example, suppose we have 34 entries in our guestbook database and we want to show 10 entries per page. From these numbers we know that we will split the result in $\text{ceil}(34 / 10) = 4$ pages.

If the entries span in more than one page we do a loop to create the links. The link will look something like this :

`guestbook.php?page=3`

Note that in above code we use `$_SERVER['PHP_SELF']` instead of using the filename itself, `guestbook.php`. This is done to save the trouble of modifying the code if someday we want to change the filename.

We temporarily put the links in an array, `$nextLink`. Once we get all the links in there we just join them all using `implode()`. And now our guestbook is done. Congratulations to you : -).

If you want the source code for this guestbook tutorial just click [here](#) . The zip file contain all the files required but **dont' forget** to modify `library/config.php` to match your own settings.

Room For Improvements

Our guestbook script is actually very simple. You can really make lots of improvements, such as :

- Flood prevention
Prevent the visitor from signing the guestbook over and over again. You can log the visitor's IP and before saving the entry check the database if there's already an entry from such IP in the past hour (or minute). You can also use cookie for this
- Bad words filtering
Before saving the message strip out any bad words. You can create an array listing the words you want to omit and then check the message against the list
- Message size limitation
This is to prevent the visitor to enter a very long message. Spammers usually do this. Advertising their website in guestbooks.
- Emoticons
You simply need to replace some special set of characters like :) or :(into an image tag. For example changing :) into ``
- Mail notification of new entry
Just use the `mail()` function after saving the message
- Allow a specific set of HTML tags

This also can be achieved by simply searching and replacing unwanted HTML tags.

Uploading Files To MySQL Database

Using PHP to upload files into MySQL database sometimes needed by some web application. For instance for storing pdf documents or images to make some kind of online briefcase (like Yahoo briefcase).

For the first step, let's make the table for the upload files. The table will consist of.

1. id : Unique id for each file
2. name : File name
3. type : File content type
4. size : File size
5. content : The file itself

For column content we'll use BLOB data type. BLOB is a binary large object that can hold a variable amount of data. MySQL have four BLOB data types, they are :

- TINYBLOB
- BLOB
- MEDIUMBLOB
- LONGBLOB

Since BLOB is limited to store up to 64 kilobytes of data we will use MEDIUMBLOB so we can store larger files (up to 16 megabytes).

Example : [upload.txt](#)

```
CREATE TABLE upload (  
id INT NOT NULL AUTO_INCREMENT,  
name VARCHAR(30) NOT NULL,  
type VARCHAR(30) NOT NULL,  
size INT NOT NULL,  
content MEDIUMBLOB NOT NULL,  
PRIMARY KEY(id)  
);
```

Uploading a file to MySQL is a two step process . First you need to upload the file to the server then read the file and insert it to MySQL.

For uploading a file we need a form for the user to enter the file name or browse their computer and select a file. The input type="file" is used for that purpose.

Example : upload.php

Source code : [upload.php](#)

```
<form method="post" enctype="multipart/form-data">
<table width="350" border="0" cellpadding="1" cellspacing="1" class="box">
<tr>
<td width="246">
<input type="hidden" name="MAX_FILE_SIZE" value="2000000">
<input name="userfile" type="file" id="userfile">
</td>
<td width="80"><input name="upload" type="submit" class="box" id="upload" value=" Upload
"></td>
</tr>
</table>
</form>
```

An upload form **must** have enctype="multipart/form-data" otherwise it won't work at all. Of course the form method also need to be set to method="post". Also remember to put a hidden input MAX_FILE_SIZE **before** the file input. It's to restrict the size of files.

After the form is submitted the we need to read the autoglobal \$_FILES. In the example above the input name for the file is userfile so the content of \$_FILES are like this :

`$_FILES['userfile']['name']`

The original name of the file on the client machine.

`$_FILES['userfile']['type']`

The mime type of the file, if the browser provided this information. An example would be "image/gif".

`$_FILES['userfile']['size']`

The size, in bytes, of the uploaded file.

`$_FILES['userfile']['tmp_name']`

The temporary filename of the file in which the uploaded file was stored on the server.

`$_FILES['userfile']['error']`

The error code associated with this file upload. ['error'] was added in PHP 4.2.0

Example : upload.php

Source code : [upload.phps](#)

```
<?php
if(isset($_POST['upload']) && $_FILES['userfile']['size'] > 0)
{
    $fileName = $_FILES['userfile']['name'];
    $tmpName = $_FILES['userfile']['tmp_name'];
    $fileSize = $_FILES['userfile']['size'];
    $fileType = $_FILES['userfile']['type'];

    $fp = fopen($tmpName, 'r');
    $content = fread($fp, filesize($tmpName));
    $content = addslashes($content);
    fclose($fp);

    if(!get_magic_quotes_gpc())
    {
        $fileName = addslashes($fileName);
    }

    include 'library/config.php';
    include 'library/opendb.php';

    $query = "INSERT INTO upload (name, size, type, content ) ".
    "VALUES ('$fileName', '$fileSize', '$fileType', '$content')";

    mysql_query($query) or die('Error, query failed');
    include 'library/closedb.php';

    echo "<br>File $fileName uploaded<br>";
}
?>
```

Before you do anything with the uploaded file. You **should not** assume that the file was uploaded successfully to the server. Always check to see if the file was successfully uploaded by looking at the file size. If it's larger than zero byte then we can assume that the file is uploaded successfully.

PHP saves the uploaded file with a temporary name and save the name in `$_FILES['userfile']['tmp_name']`. Our next job is to read the content of this file and insert the content to database. Always make sure that you use `addslashes()` to escape the content. Using `addslashes()` to the file name is also recommended because you never know what the file name would be.

That's it now you can upload your files to MySQL. Now it's time to write the script to download those files.

Downloading Files From MySQL Database

When we upload a file to database we also save the file type and length. These were not needed for uploading the files but is needed for downloading the files from the database.

The download page list the file names stored in database. The names are printed as a url. The url would look like download.php?id=3. To see a working example click [here](#). I saved several images in my database, you can try downloading them.

Example : [download.php](#)

Source code : [download.phps](#)

```
<html>
<head>
<title>Download File From MySQL</title>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
</head>

<body>
<?php
include 'library/config.php';
include 'library/opendb.php';

$query = "SELECT id, name FROM upload";
$result = mysql_query($query) or die('Error, query failed');
if(mysql_num_rows($result) == 0)
{
echo "Database is empty <br>";
}
else
{
while(list($id, $name) = mysql_fetch_array($result))
{
?>
<a href="download.php?id=<?php=$id;?>"><?php=$name;?></a> <br>
<?php
}
}
include 'library/closedb.php';
?>
</body>
</html>
```

When you click the download link, the `$_GET['id']` will be set. We can use this id to identify which files to get from the database. Below is the code for downloading files from MySQL Database.

Example : [download.php](#)

Source code : [download.phps](#)

```
<?php
if(isset($_GET['id']))
{
// if id is set then get the file with the id from database

include 'library/config.php';
include 'library/opendb.php';

$хid  = $_GET['id'];
$query = "SELECT name, type, size, content " .
        "FROM upload WHERE id = '$хid'";

$result = mysql_query($query) or die('Error, query failed');
list($name, $type, $size, $content) =          mysql_fetch_array($result);

header("Content-length: $size");
header("Content-type: $type");
header("Content-Disposition: attachment; filename=$name");
echo $content;

include 'library/closedb.php';
exit;
}

?>
```

Before sending the file content using `echo` first we need to set several headers. They are :

1. `header("Content-length: $size")`
This header tells the browser how large the file is. Some browser need it to be able to download the file properly. Anyway it's a good manner telling how big the file is. That way anyone who download the file can predict how long the download will take.
2. `header("Content-type: $type")`
This header tells the browser what kind of file it tries to download.
3. `header("Content-Disposition: attachment; filename=$name");`
Tells the browser to save this downloaded file under the specified name. If you don't send this header the browser will try to save the file using the script's name (download.php).

After sending the file the script stops executing by calling `exit`.

NOTE :

When sending headers the most common error message you will see is something like this :

Warning: Cannot modify header information - headers already sent by (output started at C:\Webroot\library**config.php:7**) in C:\Webroot\download.php on line 13

This error happens because some data was already sent before we send the header. As for the error message above it happens because i "accidentally" add one space right after the PHP closing tag (`?>`) in config.php file. So if you see this error message when you're sending a header just make sure you don't have any data sent before calling `header()`. Check the file mentioned in the error message and go to the line number specified

Next we will use a different method for uploading files. Instead of saving the file content to the database we will store the file in the server and just save the file path.

Uploading Files To File Server Using PHP

Now, we will make another upload script. But this time we won't save the file in the database. We will only store the file info there but the real file is stored in the file server. We need a little modification to the upload table. Instead of using BLOB datatype we just use VARCHAR to store the file path.

Example : [upload2.txt](#)

```
CREATE TABLE upload2 (  
id INT NOT NULL AUTO_INCREMENT,  
name VARCHAR(30) NOT NULL,  
type VARCHAR(30) NOT NULL,  
size INT NOT NULL,  
path VARCHAR(60) NOT NULL,  
PRIMARY KEY(id)  
);
```

The HTML form we use is no different with the previous one since the real changes will take place in the PHP codes.

Example : [upload2.php](#)

Source code : [upload2.phps](#)

```

<form method="post" enctype="multipart/form-data">
<table width="350" border="0" cellpadding="1" cellspacing="1" class="box">
<tr>
<td width="246">
<input type="hidden" name="MAX_FILE_SIZE" value="2000000">
<input name="userfile" type="file" id="userfile">
</td>
<td width="80"><input name="upload" type="submit" class="box" id="upload" value=" Upload
"></td>
</tr>
</table>
</form>

```

Okay, now let's take a look at the upload process. First we need to specify the directory to store the uploaded files. We store the directory name in \$uploadDir. Note that PHP **must have write access** to \$uploadDir or else the upload will fail. If you're web host using a Linux server you may need to set the permission for the upload directory to 777.

Example : upload2.php

Source code : [upload2.phps](#)

```

<?php
$uploadDir = 'C:/webroot/upload/';

if(isset($_POST['upload']))
{
$fileName = $_FILES['userfile']['name'];
$tmpName = $_FILES['userfile']['tmp_name'];
$fileSize = $_FILES['userfile']['size'];
$fileType = $_FILES['userfile']['type'];

$filePath = $uploadDir . $fileName;

$result = move_uploaded_file($tmpName, $filePath);
if (!$result) {
echo "Error uploading file";
exit;
}

include '../library/config.php';
include '../library/opendb.php';

if(!get_magic_quotes_gpc())
{
$fileName = addslashes($fileName);
$filePath = addslashes($filePath);

```

```

}

$query = "INSERT INTO upload2 (name, size, type, path ) ".
"VALUES ('$fileName', '$fileSize', '$fileType', '$filePath')";

mysql_query($query) or die('Error, query failed : ' . mysql_error());

include '../library/closedb.php';

echo "<br>Files uploaded<br>";

}
?>

```

The key here is the `move_uploaded_file()` function. This function will move the uploaded files from the temporary upload directory to the location that we earlier (`$uploadDir . $fileName`). If for some reason the function cannot move the file it will return false and we exit the script because continuing the script is no use.

Downloading

For listing the download files we just need to copy from the previous script. The real difference start when you click on the download link.

Example : [download2.php](#)

Source code : [download2.phps](#)

```
if(isset($_GET['id']))
{
include '../library/config.php';
include '../library/opendb.php';

$Id = $_GET['id'];
$query = "SELECT name, type, size, path FROM upload2 WHERE id = '$Id'";
$result = mysql_query($query) or die('Err or, query failed');
list($name, $type, $size, $filePath) = mysql_fetch_array($result);

header("Content-Disposition: attachment; filename=$name");
header("Content-length: $size");
header("Content-type: $type");

readfile($filePath);

include '../library/closedb.php';
exit;
}
```

After fetching the file info from the database and sending the required headers the next thing we need to do is read the file content from the server and send it to the browser. We can accomplish this by using readfile() function.

The Problems

When using this method of uploading files there are two problems that we need to take care of. They are :

1. Preventing direct access to the uploaded files
2. Handling duplicate file names

Preventing direct access

For this example the upload directory where the files are stored is `/home/arman198/public_html/examples/upload/files/`. Using your browser you see the upload directory by clicking [here](#). This is (usually) a bad thing because anyone can see directly the file list and download them all. If you don't want to prevent people from seeing the content of the upload directory you could create an empty file, name it `index.html` then put that file to the upload directory. This is certainly not the optimal solution because maybe some people will try guessing the files names.

A better approach is to move the upload directory away from your web root. For example, the web root for this site is: `/home/arman198/public_html/` to prevent direct listing i can set the upload directory to `/home/arman198/upload/`.

This way an outsider cannot see directly what's inside the upload directory. For example, even if you go to this url : <http://www.php-mysql-tutorial.com/..../upload/> you can't see the upload directory

Handling duplicate file names

When saving the files into the MySQL database we don't have to worry about this. The table for saving the files uses an id as the primary key so even we put ten files with the same name there won't be any problem since we access the files using that unique id.

The problem arise when saving the files to file server. The `move_uploaded_file()` function will overwrite a file with the same name and this is certainly not a desired behaviour.

To prevent this we just need to modify the file name. In this example the file names are changed into a random string, 32 characters long.

Example : `upload3.php`

Source code : [upload3.phps](#)

```
<?php
// ... same code as before

// get the file extension first
$ext = substr(strchr($fileName, "."), 1);

// make the random file name
$randName = md5(rand() * time());

// and now we have the unique file name for the upload file
$filePath = $uploadDir . $randName . '.' . $ext;
```

```
$result = move_uploaded_file($tmpName, $filePath);  
  
// ... same code as before  
  
}  
?>
```

First we extract the file extension from the file name using `strchr()` function combined with `substr()`. Then using `md5()` we generate the 32 characters long of random string. It will look something like 7d1d1da5aac5ad72b293165e8e6fe89b. After we join them up we get the new unique file name. This way the chance of stumbling upon a duplicate name problem is very very slim.

As for the download part there's no change required. All we did was change the file name on the server so the previous download script ([download2.phps](#)) will do just fine

That's it. We're done. If you need the source codes for this php upload tutorial you can download them as zip file [here](#). Make sure you change the library/config.php file to match your own settings and change the `$uploadDir` too if necessary.

Content Management System (CMS) Using PHP And MySQL

A Content Management System (CMS) is used to add, edit, and delete content on a website. For a small website, such as this, adding and deleting a page manually is fairly simple. But for a large website with lots of pages like a news website adding a page manually without a content management system can be a headache.

A CMS is meant to ease the process of adding and modifying new content to a webpage. The pages content are stored in database, not in the file server.

This tutorial will present an example of a simple content management system. You will be able to add, edit and delete articles using HTML forms.

For the database table we'll call it the news table. It consist of three columns :

- id : The article's id
- title : The title of an article
- content : The article itself

First we need to create a script to add an article. It is just a form where a user can enter the article's title and content.

Example : [cms-add.php](#)

Source code : [cms-add.php](#) , [cms.txt](#)

```
<form method="post">
<table width="700" border="0" cellpadding="2" cellspacing="1" align="center">
<tr>
<td width="100">Title</td>
<td><input name="title" type="text"></td>
</tr>
<tr>
<td width="100">Content</td>
<td><textarea name="content" cols="50" rows="10"></textarea> </td>
</tr>
<tr>
<td width="100">&nbsp;</td>
<td>&nbsp;</td>
</tr>
<tr>
<td colspan="2" align="center"><input name="save" type="submit" value="Save Article"></td>
</tr>
</table>
</form>
```

When an article is added the script just inserts the article into the database. An article id is automatically generated by MySQL because the id column was created with AUTO_INCREMENT parameter.

```
<?php
if(isset($_POST['save']))
{
    $title = $_POST['title'];
    $content = $_POST['content'];

    if(!get_magic_quotes_gpc())
    {
        $title = addslashes($title);
        $content = addslashes($content);
    }
    include 'library/config.php';
    include 'library/opencv.php';

    $query = " INSERT INTO news (title, content) " .
        " VALUES ('$title', '$content')";
    mysql_query($query) or die('Error ,query failed');
```

```

include 'library/closedb.php';

echo "Article '$title' added";
}
?>

```

Now that we have the script to add articles let's create another script to view those articles. The script will list the title of articles available in database as clickable links. The article link will have the article id appended like this

<http://www.php-mysql-tutorial.com/examples/cms/article1.php?id=3>

One possible implementation of article1.php is presented below :

Example : [article1.php](#)

Source code : [article1.phps](#)

```

<?php
include 'library/config.php';
include 'library/opendb.php';

// if no id is specified, list the available articles
if(!isset($_GET['id']))
{
    $self = $_SERVER['PHP_SELF'];

    $query = "SELECT id, title FROM news ORDER BY id";
    $result = mysql_query($query) or die('Error : ' . mysql_error());

    // create the article list
    $content = '<ol>';
    while($row = mysql_fetch_array($result, MYSQL_NUM))
    {
        list($id, $title) = $row;
        $content .= "<li><a href=\"\$self?id=$id\">$title</a></li>\r\n";
    }

    $content .= '</ol>';

    $title = 'Available Articles';
} else {
    // get the article info from database
    $query = "SELECT title, content FROM news WHERE id=".$_GET['id'];
    $result = mysql_query($query) or die('Error : ' . mysql_error());
}

```



```

$row = mysql_fetch_array($result, MYSQL_ASSOC);

$title = $row['title'];
$content = $row['content'];
}

include 'library/closedb.php';
?>

// ... more code here

```

When [article1.php](#) is first called the `$_GET['id']` variable is not set and so it will query the database for the article list and save the list in the `$content` variable as an ordered list. The variable `$title` and `$content` will be used later when we print the result page. Take a look at the code below :

Example : [article1.php](#)
Source code : [article2.phps](#)

```

<?php

// ... previous code

?>
<html>
<head>
<title>
<?php echo $title; ?>
</title>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
<style type="text/css">

// ... some css here to make the page look nicer

</style>
</head>
<body>
<table width="600" border="0" align="center" cellpadding="10" cellspacing="1"
bgcolor="#336699">
<tr>
<td bgcolor="#FFFFFF">
<h1 align="center"><?php echo $title; ?></h1>
<?php
echo $content;

// when displaying an article show a link
// to see the article list

```

```

if(isset($_GET['id']))
{
?>
<p>&nbsp;</p>
<p align="center"><a href="<?php echo $_SERVER['PHP_SELF']; ?>">Article List</a></p>
<?php
}
?>
</td>
</tr>
</table>
</body>
</html>

```

If you click on an article link the script will fetch the article's title and content from the database, save it to \$title and \$content variable and print the HTML file . At the bottom of the page we place a code to show the link to the article list which is the file itself without any query string (\$_SERVER['PHP_SELF'])

With this implementation each article request involve one database query. For a heavy load website with lots of articles using the above implementation can cause a very high amount of database - request. So we need a better cms solution to reduce the load.

One feasible solution is to implement caching (cache) which load an article from the database only once when the article was first requested. The article is then saved to a cache directory as a regular HTML file. Subsequent request to the article will no longer involve any database request. The script just need to read the requested article from the cache d irectory.

Example : [article2.php](#)

Source code : [article2.phps](#)

```

<?php
include 'library/config.php';
include 'library/opendb.php';

$cacheDir = dirname(__FILE__) . '/cache/';

if (isset($_GET['id'])) {
    $cacheFile = $cacheDir . '_' . $_GET['id'] . '.html';
} else {
    $cacheFile = $cacheDir . 'index.html';
}

if (file_exists($cacheFile))
{
    header("Content-Type: text/html");

```

```

    readfile($cacheFile);
    exit;
}

// ... more code coming

?>

```

First we need to specify the cache directory where all cache files are located. For this example the cache directory is located in the same place as the article2.php script. I mean if article2.php is stored in C:/webroot then the cache dir is in C:/webroot/cache/

The script then check if the article was already in the cache. An article is saved into the cache directory using a filename generated from it's id. For example if you request the article using a link like this :

<http://www.php-mysql-tutorial.com/examples/cms/article2.php?id=3>

Then the cache file for the article is

_3.html

This filename is just an underscore (_) followed by the article id. In case article2.php is called like this :

<http://www.php-mysql-tutorial.com/examples/cms/article2.php>

no id is defined so we make the cache file name as index.html

If the cache file is found , the content is read and printed using readfile() and the script terminate. When the article is not found in the cache then we need to look in the database and get the page content from there.

Example : [article2.php](#)

Source code : [article2.phps](#)

```

<?php

// ... previous code

if(!isset($_GET['id']))
{
    $self = $_SERVER['PHP_SELF'];

```

```

$query = "SELECT id, title FROM news ORDER BY id";
$result = mysql_query($query) or die('Error : ' . mysql_error());

$content = '<ol>';
while($row = mysql_fetch_array($result, MYSQL_NUM))
{
    list($id, $title) = $row;
    $content .= "<li><a href=\"\$self?id=$id\">$title</a></li>\r\n";
}

$content .= '</ol>';

$title = 'Available Articles';
} else {
    // get the article info from database
    $query = "SELECT title, content FROM news WHERE id=".$_GET['id'];
    $result = mysql_query($query) or die('Error : ' . mysql_error());
    $row = mysql_fetch_array($result, MYSQL_ASSOC);

    $title = $row['title'];
    $content = $row['content'];
}

include 'library/closedb.php';

// ... still more code coming

?>

```

As you can see above the process of fetching the article list and content is the same as `article1.php`. But before showing the page we have to start output buffering so we can save the content of the generated HTML file.

See the code below. Just before printing the html we call `ob_start()` to activate output buffering. From this point no output is sent from the script to the browser. So in the code example below anything between `<html>` and `</html>` tag is not sent to the browser but stored in an internal buffer first.

After the closing html tag we use `ob_get_contents()` to get the buffer content and store it in a temporary variable, `$buffer`. We then call `ob_end_flush()` which stops the output buffering (so the page is now sent to the browser).

Example : [article2.php](#)
Source code : [article2.phps](#)

```
<?php
```

```
// ... previous code

ob_start();
?>
<html>


// ... same html code as article1.php


</html>
<?php


// get the buffer
$buffer = ob_get_contents();


// end output buffering, the buffer content
// is sent to the client
ob_end_flush();


// now we create the cache file
$fp = fopen($cacheFile, "w");
fwrite($fp, $buffer);
fclose($fp);
?>
```

Now that we have the file content we can write the cache file using the filename generated earlier (using underscore plus the article id). From now on any request to the article will no longer involve a database query. At least until the article is updated.

Next we will need an admin page for our content management system. It is where we can edit and delete the articles.

Admin Page For Content Management System (CMS)

Now we start creating the administration page for our content management system. This page will list all the articles we have in database and also a menu for article maintenance including add, view, update and delete articles.

The view command is just a link to the article2.php so I won't explain about it anymore and we can focus on deleting and modifying an article.

Delete Article

Go the [admin page](#), and try to delete an article (if there's no article create one first). When you want to delete an article, a little javascript confirmation window will pop up. You should always put this kind of confirmation when trying to delete something. Just to make sure you don't delete by accident.

Example : [cms-admin.php](#)

Source code : [cms-admin.phps](#)

```
<?php
include 'library/config.php';
include 'library/opendb.php';

if(isset($_GET['del']))
{
    $query = "DELETE FROM news WHERE id = '{$_GET['del']}'";
    mysql_query($query) or die('Error : ' . mysql_error());

    // then remove the cached file
    $cacheDir = dirname(__FILE__) . '/cache/';
    $cacheFile = $cacheDir . '_' . $_GET['id'] . '.html';

    @unlink($cacheFile);

    // and remove the index.html too because the file list
    // is changed
    @unlink($cacheDir . 'index.html');

    // redirect to current page so when the user refresh this page
    // after deleting an article we won't go back to this code block
    header('Location: ' . $_SERVER['PHP_SELF']);
    exit;
```

```
}  
  
// ... more code here  
?>
```

Deleting an article is a simple process. When you click on an 'Delete' link. The admin page will reload to something like :

`cms-admin.php?del=3`

where 3 is the article id that we want to delete.

Once we have the article id we just need to execute a delete query and article will be deleted from the database. Then we delete the cache file (if it exists) and the index.html file too because the file list already changed. In the code above we use an @ before the unlink() command. This is to suppress any error message in case the cache file is not found (i.e. the article is never requested before / recently updated)

After that we reload the admin page (omitting the 'del' query string). We need to do this because if the admin is refreshing the page we will go back executing the delete query again. It's a waste of database resource.

Here is the rest of admin page code. We just fetch the article list from the database, print the article title and add some link to view, edit, delete and add article.

Example : [cms-admin.php](#)
Source code : [cms-admin.php](#)

```
// ... previous code  
  
<html>  
<head>  
<title>Admin Page For Content Management System (CMS)</title>  
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">  
<script language="JavaScript">  
function delArticle(id, title)  
{  
    if (confirm("Are you sure you want to delete '" + title + "'"))  
    {  
        window.location.href = 'cms-admin.php?del=' + id;  
    }  
}  
</script>  
</head>
```

```

<body>
<?php
$query = "SELECT id, title FROM news ORDER BY id" ;
$result = mysql_query($query) or die('Error : ' . mysql_error());
?>
<table width="600" border="0" align="center" cellpadding="5" cellspacing="1"
bgcolor="#999999">
<tr align="center" bgcolor="#CCCCCC">
<td width="500"><strong>Title</strong></td>
<td width="150"><strong>Action</strong></td>
</tr>
<?php
while(list($id, $title) = mysql_fetch_array($result, MYSQL_NUM))
{

?>
<tr bgcolor="#FFFFFF">
<td width="500">
<?php echo $title;?>
</td>
<td width="150" align="center">
<a href="article2.php?id=<?php echo $id;?>" target="_blank">view</a>
| <a href="cms-edit.php?id=<?php echo $id;?>">edit</a>
| <a href="javascript:delArticle('<?php echo $id;?>',
'<?php echo $title;?>');">delete</a></td>
</tr>
<?php
}

include 'library/closedb.php';
?>
</table>
<p align="center"><a href="cms-add.php">Add an article</a> </p>
</body>
</html>

```


Edit Article

To complete our content management system we'll make the edit page. Its interface is basically the same with cms-add.php but we just need to add some code to fetch the article information from database.

One important thing to remember when editing a page is that the page content may contain html tags or even javascript tags. If we put the content as is in the textarea wrong stuff could happen. To prevent this we need to use htmlspecialchars(). This function will change special HTML characters like < and > into **<** and **>**.

If you're curious what could happen when the content are put as is, go ahead and try removing the htmlspecialchars() code and see the result..

Example : [cms-edit.php](#)

Source code : [cms-edit.phps](#)

```
<html>
<head>
<title>Edit An Article</title>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
<style type="text/css">
<!--
.box {
font-family: Arial, Helvetica, sans-serif;
font-size: 12px;
border: 1px solid #000000;
}
-->
</style>
</head>

<body>
<?php
include 'library/config.php';
include 'library/opendb.php';

if(isset($_GET['id']))
{
    $query = "SELECT id, title, content " .
            "FROM news " .
            "WHERE id = '{$_GET['id']}'";
    $result = mysql_query($query) or die('Error : ' . mysql_error());
    list($id, $title, $content) = mysql_fetch_array($result,
                                                    MYSQL_NUM);
```

```

    $content = htmlspecialchars($content);
}
else if(isset($_POST['save']))
{
    $id = $_POST['id'];
    $title = $_POST['title'];
    $content = $_POST['content'];

    if(!get_magic_quotes_gpc())
    {
        $title = addslashes($title);
        $content = addslashes($content);
    }

    // update the article in the database
    $query = "UPDATE news "
        . "SET title = '$title', content = '$content' "
        . "WHERE id = '$id'";
    mysql_query($query) or die('Error : ' . mysql_error());

    // then remove the cached file
    $cacheDir = dirname(__FILE__) . '/cache/';
    $cacheFile = $cacheDir . '_' . $_GET['id'] . '.html';

    @unlink($cacheFile);

    // and remove the index.html too because the file list
    // is changed
    @unlink($cacheDir . 'index.html');

    echo "Article '$title' updated";

    // now we will display $title & content
    // so strip out any slashes
    $title = stripslashes($title);
    $content = stripslashes($content);
}

include 'library/closedb.php';
?>
<form method="post">
<input type="hidden" name="id" value="<?=$id;?>">
<table width="700" border="0" cellpadding="2" cellspacing="1" class="box">
<tr>
<td width="100">Title</td>

```

```

<td><input name="title" type="text" class="box" id="title" value="<?=$title;?>"></td>
</tr>
<tr>
<td width="100">Content</td>
<td><textarea name="content" cols="50" rows="10" class="box"
id="content"><?=$content;?></textarea></td>
</tr>
<tr>
<td width="100">&nbsp;</td>
<td>&nbsp;</td>
</tr>
<tr>
<td colspan="2" align="center"><input name="update" type="submit" class="box" id="update"
value="Update Article"></td>
</tr>
</table>
<p align="center"><a href="cms-admin.php">Back to admin page</a></p>
</form>
</body>
</html>

```

When the form is submitted we begin the updating process. First we use an UPDATE query to modify the article in the database. Then we remove the cache file and the index.html because the content of these two files may no longer accurate. We also use an @ before the unlink command to suppress any error message.

You may already notice this but we already use the \$cacheDir variable in three scripts (article2.php, cms-admin.php, cms-edit.php). It's a good idea to put this variable in the config.php file. So if we want to specify a different cache directory we only need to change the variable in one file instead of three.

Okay, now that we have completed the tutorial maybe you start thinking that this system is too simple. If you do think so, you are absolutely right : -).

There are a whole bunch of great content management system solutions out there. [Mambo](#) and [Website Baker](#) are two of them. These two are really cool (and free). Mambo has lots of features, it is really a complete content management system solution. Website Baker is similar to Mambo but it's simpler and easier to use and learn.

I suggest you get one of those two, use it and learn from the source code. You really can learn a lot from it. By the way all the files needed for this cms tutorial can be downloaded [here](#)

User Authentication

Here are some authentication method that we'll discuss

1. [Basic authentication](#)
We hard code the username and password combination in the login script itself. Suitable for simple application
2. [User & password stored in database](#)
A very common method. We store all the user name and password information in the database
3. [User authentication with image verification](#)
This is a more advance method of user authentication. We can prevent any automatic login by a robot (script) by using this method

All three of these methods will use the session. Since session is supported by PHP by default you don't need to configure anything to use it **but** you do need to use a recent version of PHP. By recent i mean greater than PHP 4.3.2. Using lesser than that version may cause the script to work only occasionally (sometimes it won't work at all). Don't ask me what cause that cause i don't about the reason too :-)

For the third method you will need GD library which is already bundled in PHP but you will need to enable the GD support before using the library.

To see if GD library is already enabled save the following code, execute it, and see what the result is

Source : [gdtest.phps](#)

```
<?php
if (function_exists('imagecreate')) {
    echo "GD Library is enabled <br>\r\n<pre>";
    var_dump(gd_info());
    echo "</pre>";
} else {
    echo 'Sorry, you need to enable GD library first';
}
?>
```

If GD is not enabled yet open up php.ini and search for "extension=php_gd2.dll" (without the quotes). Uncomment the line by removing the semicolon (;). After restarting your webserver try running the test script again. You should see the message saying that GD is enabled.

Basic User Authentication

With this basic authentication method we store the user information (user id and password) directly in the script. This is only good if the application only have one user since adding more user means we must also add the new user id and password in the script.

Let's start by making the login form first. You can see the code below.

Example : [basic/login.php](#)

Source : [basic/login.phps](#)

```
<?php

// ... we will put some php code here

?>
<html>
<head>
<title>Basic Login</title>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
</head>

<body>
<?php
if ($errorMessage != "") {
?>
<p align="center"><strong><font color="#990000"><?php echo $errorMessage;
?></font></strong></p>
<?php
}
?>
<form method="post" name="frmLogin" id="frmLogin">
<table width="400" border="1" align="center" cellpadding="2" cellspacing="2">
<tr>
<td width="150">User Id</td>
<td><input name="txtUserId" type="text" id="txtUserId"></td>
</tr>
<tr>
<td width="150">Password</td>
<td><input name="txtPassword" type="password" id="txtPassword"></td>
</tr>
<tr>
<td width="150">&nbsp;</td>
```

```

<td><input type="submit" name="btnLogin" value="Login"></td>
</tr>
</table>
</form>
</body>
</html>

```

Nothing sophisticated in that form. It's just a basic login form with two input for entering the user id and password. Make sure that the form method is set to **post** since we certainly don't want to show up the user id and password in the address bar.

Right before the login form we there's a code for printing an error message. We can ignore this for now since we'll be talking about it shortly.

Once we submit the form we can start the authentication process. We simply check if the user id and password exist in \$_POST and check if these two match the hardcoded user id and password.

Example : <basic/login.php>

Source : <basic/login.phps>

```

<?php
// we must never forget to start the session
session_start();

$errorMessage = "";
if (isset($_POST['txtUserId']) && isset($_POST['txtPassword'])) {
// check if the user id and password combination is correct
if ($_POST['txtUserId'] === 'theadmin' && $_POST['txtPassword'] === 'chumbawamba') {
// the user id and password match,
// set the session
$_SESSION['basic_is_logged_in'] = true;

// after login we move to the main page
header('Location: main.php');
exit;
} else {
$errorMessage = 'Sorry, wrong user id / password';
}
}
?>

// ... here is the login form shown previously

```

But before we start matching the user id and password. We **must** start the session first. Never forget to start the session before doing anything to the session since it won't work.

You can see above that the hardcoded user id and password are "theadmin" and "chumbawamba". If the submitted user id and password match these two then we set the value of `$_SESSION['basic_is_logged_in']` to true. After that we move the application's main page. In this case it's called main.php

If the user id and password don't match we set the error message. This message will be shown on top of the login form.

Note : When starting the session you may stumble upon this kind of error :

Warning: session_start(): Cannot send session cache limiter - headers already sent (output started at C:\Webroot\examples\user-authentication\basic\login.php:1) in C:\Webroot\examples\user-authentication\basic\login.php on line 3

PHP will spit this error message if the script that call session_start() already send something (a blank space, newline, etc). The error above happen when i add a single space on the **first line** right before the php opening tag (<?php). Thankfully the error message shows where the output started so fixing this kind of error is simple. After removing that extra space the error is fixed.

Checking if the user is logged in or not

Since the application main page, main.php, can only be accessed by those who already authenticated themselves we must check that before displaying the page.

The checking process is fairly simple. We just see if `$_SESSION['basic_is_logged_in']` is set or not. If it is set we check if the value is true. If either of this condition is not met then the one accessing this page haven't login yet. And so we redirect to the login page and quit the script.

If `$_SESSION['basic_is_logged_in']` is set and it's value is true then we can continue showing the rest of the page.

Here is the code for main.php

Example : [basic/main.php](#)

Source : [basic/main.phps](#)

```
<?php
// like i said, we must never forget to start the session
session_start();

// is the one accessing this page logged in or not?
if (!isset($_SESSION['basic_is_logged_in'])
    || $_SESSION['basic_is_logged_in'] !== true) {
```

```

    // not logged in, move to login page
    header('Location: login.php');
    exit;
}

?>
<html>
<head>
<title>Main User Page</title>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
</head>

<body>
<p>This is the main application page. You are free to play around here since you
are an authenticated user :-) </p>
<p>&nbsp;</p>
<p><a href="logout.php">Logout</a> </p>
</body>
</html>

```

A little note about naming a session variable. As you can see the session that we used to mark whether a user is logged in or not is named 'basic_is_logged_in'. When setting a name for a session variable it's a good thing to use the application name as the prefix. In this case the prefix is 'basic_'. This is especially important when you have **multiple application** on one site where each requires **different** login information.

For example, suppose we have a cms application and a link exchange application where each have their own user authentication system. In both application we use the session variable \$_SESSION['is_logged_in']. In this case if we already logged in in the cms application we will no longer be required to login in the link exchange application since both are using the same session name. This is usually **not** an intended feature. To avoid that kind of thing we can instead use \$_SESSION['cms_is_logged_in'] and \$_SESSION['exchange_is_logged_in']

The Logout Script

No login script is complete without the logout script right? So let's start making the logout script now.

The process of logging out a user is actually depends on how we check if a user is logged in or not. In our case we check if \$_SESSION['basic_is_logged_in'] is already set or not and check whether it's value is true. Using this information we can build the logout script to simply unset this session **or** set the session value to false.

The logout script below use the first method (unset the session). Here is the code :

Example : <basic/logout.php>

Source : <basic/logout.php>

```
<?php
// i will keep yelling this
// DON'T FORGET TO START THE SESSION !!!
session_start();

// if the user is logged in, unset the session
if (isset($_SESSION['basic_is_logged_in'])) {
    unset($_SESSION['basic_is_logged_in']);
}

// now that the user is logged out,
// go to login page
header('Location: login.php');
?>
```

Before we unset the session we first check if the session is actually exist or not. In case you access the logout script before using the login form then this session variable won't exist yet.

Unsetting a variable is done simply by using the `unset()` statement. After we unset the session the next thing we do is simply moving to the login page. Pretty simple huh ?

Another note : You may already notice this but in each script i keep repeating about not to forget to start the session. The reason is that it is a very very very common error to forget about it when handling session. Once i spent a lot of time debugging a script and it was all because i forgot to add that one line.

So please remember this : **DON'T FORGET TO START THE SESSION !!!**

Next we will make a better login method where the user info is stored in the database

Better User Authentication : Storing user id & Password In Database

A more common method of authenticating a user is by checking the database to see if the submitted user id and password combination exist. To use this kind of authentication we must first build the database table. The sql code to build it is shown below. We also add two user accounts for

testing the login script

Source : [database/tbl_auth_user.sql](#)

```
CREATE TABLE tbl_auth_user (  
user_id VARCHAR(10) NOT NULL,  
user_password CHAR(32) NOT NULL,  
  
PRIMARY KEY (user_id)  
);
```

```
INSERT INTO tbl_auth_user (user_id, user_password) VALUES ('theadmin',  
PASSWORD('chumbawamba'));  
INSERT INTO tbl_auth_user (user_id, user_password) VALUES ('webmaster',  
PASSWORD('webmistress'));
```

We will use the same html code to create login form created in [previous example](#). We will only need to modify the login process a bit. The login script's content is shown below :

Example : [database/login.php](#)

Source : [database/login.phps](#)

```
<?php  
// we must never forget to start the session  
session_start();  
  
$errorMessage = "";  
if (isset($_POST['txtUserId']) && isset($_POST['txtPassword'])) {  
    include 'library/config.php';  
    include 'library/openssl.php';  
  
    $userId = $_POST['txtUserId'];  
    $password = $_POST['txtPassword'];  
  
    // check if the user id and password combination exist in database  
    $sql = "SELECT user_id  
        FROM tbl_auth_user  
        WHERE user_id = '$userId'  
            AND user_password = PASSWORD('$password')";  
  
    $result = mysql_query($sql)  
        or die('Query failed. ' . mysql_error());  
  
    if (mysql_num_rows($result) == 1) {  
        // the user id and password match,  
        // set the session
```

```

$_SESSION['db_is_logged_in'] = true;

// after login we move to the main page
header('Location: main.php');
exit;
} else {
    $errorMessage = 'Sorry, wrong user id / password';
}

include 'library/closedb.php';
}
?>

```

// ... same html login form as previous example

Instead of checking the user id and password against a hardcoded info we query the database if these two exist in the database using the SELECT query. If we found a match we set the session variable and move to the main page. Note that the session name is prefixed by 'db_' to make it different than the previous example.

For the next two scripts (main.php and logout.php) the code is similar to previous one. The only difference is the session name. Here is the code for these two

Example : [database/main.php](#)

Source : [database/main.phps](#)

```

<?php
session_start();

// is the one accessing this page logged in or not?
if (!isset($_SESSION['db_is_logged_in'])
    || $_SESSION['db_is_logged_in'] !== true) {

    // not logged in, move to login page
    header('Location: login.php');
    exit;
}

?>

// ... some html code here

```

Example : [database/logout.php](#)

Source : [dabase/logout.php](#)s

```
<?php
session_start();

// if the user is logged in, unset the session
if (isset($_SESSION['db_is_logged_in'])) {
    unset($_SESSION['db_is_logged_in']);
}

// now that the user is logged out,
// go to login page
header('Location: login.php');
?>
```

The next part is the most interesting one of the authentication methods. We will display an image and the user must enter the random number displayed in the image to complete the login process.

User Authentication With Image Verification

In some cases you may want your logging form to be able to prevent automatic login by a robot (script). To achieve this we can create a login form which displays an image showing random numbers. The login form will have an extra input field to enter the values shown.

Take a look at the [login form](#). The numbers shown there will change everytime you refresh the page. Go ahead and try refreshing that page you will see the numbers always change.

Before working on the login form we must take care of the script that create the verification image first. Here is the code :

Example : [image-verification/randomImage.php](#)
Source : [image-verification/randomImage.php](#)s

```
<?php
session_start();

// generate 5 digit random number
$rand = rand(10000, 99999);

// create the hash for the random number and put it in the session
$_SESSION['image_random_value'] = md5($rand);
```

```

// create the image
$image = imagecreate(60, 30);

// use white as the background image
$bgColor = imagecolorallocate ($image, 255, 255, 255);

// the text color is black
$textColor = imagecolorallocate ($image, 0, 0, 0);

// write the random number
imagestring ($image, 5, 5, 8, $rand, $textColor);

// send several headers to make sure the image is not cached
// taken directly from the PHP Manual

// Date in the past
header("Expires: Mon, 26 Jul 1997 05:00:00 GMT");

// always modified
header("Last-Modified: " . gmdate("D, d M Y H:i:s") . " GMT");

// HTTP/1.1
header("Cache-Control: no-store, no-cache, must-revalidate");
header("Cache-Control: post-check=0, pre-check=0", false);

// HTTP/1.0
header("Pragma: no-cache");

// send the content type header so the image is displayed properly
header('Content-type: image/jpeg');

// send the image to the browser
imagejpeg($image);

// destroy the image to free up the memory
imagedestroy($image);
?>

```

To create a five digit random number we use `rand()` function and specify that the random number must be between 10000 and 99999. We put the hash value of this random number in the session. This hash value will be used by the login script to check if the entered number is correct.

Next we create a small image, 60 x 30 pixels, using `imagecreate()`. We set the background color to white (RGB = 255, 255, 255) using `imagecolorallocate()` function. Note that the first call to `imagecolorallocate()` will always set the background color for the image. Then we set the text color

as black (RGB = 0, 0, 0). Feel free to change the color text to your liking.

To print the random number to the image we use the function `imagestring()`. In the script above we call this function like this : `imagestring ($image, 5, 5, 8, $rand, $textColor);`

The first argument passed to this function is the image handler (`$image`). The second one (`5`) is the font. You can choose from one to five where one is the smallest font. The third and fourth parameter is the horizontal and vertical coordinate where we will print the image. The top left corner is defined as 0, 0. The last one is the text color which is black as mentioned earlier.

After we got the image ready we can now send it to the browser. But before doing that we must set several headers to make sure that the image is not cached. If the image is cached then the login form will show the same image even if you refresh it. That will cause a problem since the random number is always different.

Finally after everything is set we send the image to the browser using `imagejpeg()` and to free the memory we use `imagedestroy()`.

The Login Form

The login form is pretty much the same but only have extra field to enter the displayed number.

Example : <image-verification/login.php>

Source : <image-verification/login.phps>

```
<?php
// ... the login script is up here
?>
<html>
<head>
<title>Basic Login</title>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
</head>

<body>
<?php
if ($errorMessage != "") {
?>
<p align="center"><strong><font color="#990000"><?php echo $errorMessage;
?></font></strong></p>
<?php
}
?>
<form action="" method="post" name="frmLogin" id="frmLogin">
```

```

<table width="500" border="1" align="center" cellpadding="2" cellspacing="2">
<tr>
<td width="150">User Id</td>
<td><input name="txtUserId" type="text" id="txtUserId"></td>
</tr>
<tr>
<td width="150">Password</td>
<td><input name="txtPassword" type="password" id="txtPassword"></td>
</tr>
<tr>
<td width="150">Enter Number</td>
<td><input name="txtNumber" type="text" id="txtNumber" value="">
&nbsp;&nbsp;&nbsp;</td>
</tr>

<tr>
<td width="150">&nbsp;&nbsp;&nbsp;</td>
<td><input name="btnLogin" type="submit" id="btnLogin" value="Login"> </td>
</tr>
</table>
</form>
</body>
</html>

```

To check if the login information is correct we first check if the entered number is the same one as displayed in the image. To do this we check the **hash** of the entered number and see if it match the one saved in the session. If the number don't match we just set an error message.

If the number do match we continue checking the given user id and password just like the previous example. If the userid and password combination is correct we set \$_SESSION['image_is_logged_in'] to true and move on to the main page

Example : <image-verification/login.php>

Source : <image-verification/login.phps>

```

<?php
// we must never forget to start the session
session_start();

$errorMessage = "";
if (isset($_POST['txtUserId']) && isset($_POST['txtPassword'])) {
    // first check if the number submitted is correct
    $number = $_POST['txtNumber'];

    if (md5($number) == $_SESSION['image_random_value']) {
        include 'library/config.php';
    }
}

```

```

include 'library/opendb.php';

$userId = $_POST['txtUserId'];
$password = $_POST['txtPassword'];

// check if the user id and password combination exist
$sql = "SELECT user_id
        FROM tbl_auth_user
        WHERE user_id = '$userId'
        AND user_password = PASSWORD('$password')";

$result = mysql_query($sql) or
        die('Query failed. ' . mysql_error());

if (mysql_num_rows($result) == 1) {
    // the user id and password match,
    // set the session
    $_SESSION['image_is_logged_in'] = true;

    // remove the random value from session
    $_SESSION['image_random_value'] = "";

    // after login we move to the main page
    header('Location: main.php');
    exit;
} else {
    $errorMessage = 'Sorry, wrong user id / password';
}

include 'library/closedb.php';
} else {
    $errorMessage = 'Sorry, wrong number. Please try again';
}
}
?>

```

We don't need to discuss about main.php and logout.php since they are the same as previous example except the session name is now called `$_SESSION['image_is_logged_in']`. So instead of working on those two files let's move on to a more interesting stuff...

Improving The Verification Image

We can improve the verification image in at least two ways. They are :

1. Using alphanumeric characters as the verification code instead of numbers
2. Using background images

For the first improvement the only thing we need to change is the way we generate the code. Take a look at the code below

Example : image-verification/randomImage2.php

Source : image-verification/randomImage2.phps

```
<?php
session_start();

$alphanum = "ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789";





// generate the verification code
$rand = substr(str_shuffle($alphanum), 0, 5);

// ... no changes after this point
?>
```

We start by defining the characters that we want to use in the verification code. For this example we use upper case alphabet plus numbers. The code is generated using the combination of `str_shuffle()` and `substr()` function. Using `str_shuffle()` we jumble all the characters in `$alphanum` and then using `substr()` we take just the first five characters. The result will look something like "D79ZG". Just [run the example](#) and see it for yourself.

The second improvement is by using background images. Maybe you already know this but there are software/scripts that can extract the characters displayed as images . And if the verification image only use plain background color identifying the characters will be quite easy.

For this reason we will make the verification code displayed on a background image. In this tutorial we will only use four different background images. You can add as many background images as you want in your own code. Here are the background images :

- background image #1 : 
- background image #2 : 
- background image #3 : 
- background image #4 : 

Note : When you want to create a background image make sure the code will still be readable on

them. For instance it's quite hard (at least for me) to recognize the code when the code is displayed on image #1 and image #4. Go [take a look](#) if you don't believe me.

Here is the code for this second improvement

Example : [image-verification/randomImage3.php](#)

Source : [image-verification/randomImage3.phps](#)

```
<?php
session_start();

$alphanum = "ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789";

// generate the verification code
$rand = substr(str_shuffle($alphanum), 0, 5);

// choose one of four background images
$bgnNum = rand(1, 4);

$image = imagecreatefromjpeg("background$bgnNum.jpg");

$textColor = imagecolorallocate ($image, 0, 0, 0);

// write the code on the background image
imagestring ($image, 5, 5, 8, $rand, $textColor);

// ... no changes after this point
?>
```

After making the verification code we randomly pick one background image. Then we create an image object from the chosen background using `imagecreatefromjpeg()` and draw the code on the background. The rest of the code is the same as `randomImage1.php` and `randomImage2.php` so no need to explain it here.

Okay that is it. The three method of user authentication. Just pick one that fit your application. Btw, all the code for the three methods can be downloaded [here](#)

PHP MySQL Image Gallery

On the [previous tutorial](#) you already know how to upload and download files to the server. Now we're gonna reuse the codes to build an image gallery. It's a simple image gallery where the admin (that's you) is the only one who can add/modify/delete the album and images. The "normal folks" can

only browse around the image gallery checking out the images from one album to another

The admin section contain the following :

- [Add New Album](#)
- [Album List](#)
- [Edit & Delete Album](#)
- [Add Image](#)
- [Image List](#)
- [Edit & Delete Image](#)

And the visitor page contain these :

- [Display Album List](#)
- [Display Image List](#)
- [Display Image Detail](#)

Now, before we go straight to the codes we need to talk about the database design, directory layout, and configurations.

Database Design

For a simple image gallery like this we only need two tables, tbl_album and tbl_image. Here is the SQL to create these tables

Source code : [image-gallery.sql](#)

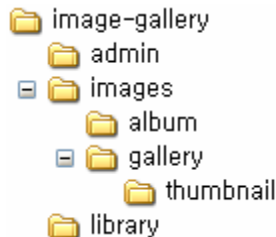
```
CREATE TABLE tbl_album (  
  al_id INT NOT NULL AUTO_INCREMENT,  
  al_name VARCHAR(64) NOT NULL,  
  al_description TEXT NOT NULL,  
  al_image VARCHAR(64) NOT NULL,  
  al_date DATETIME NOT NULL,  
  PRIMARY KEY(al_id)  
);
```

```
CREATE TABLE tbl_image (  
  im_id INT NOT NULL AUTO_INCREMENT,  
  im_album_id INT NOT NULL,  
  im_title VARCHAR(64) NOT NULL,  
  im_description TEXT NOT NULL,  
  im_type VARCHAR(30) NOT NULL,  
  im_image VARCHAR(60) NOT NULL,  
  im_thumbnail VARCHAR(60) NOT NULL,  
  im_date DATETIME NOT NULL,
```

PRIMARY KEY(im_id)
);

Directory Layout

The image below show the file organization for the image gallery



The images directory is where we kept all of the images. The image icons are stored in the thumbnail sub-directory under the gallery. Please remember **to set write access** to the album, gallery, and thumbnail directories otherwise the gallery script will not be able to save the images.

Configurations

There are some constants in the [config file](#) that you should change :

1. ALBUM_IMG_DIR, GALLERY_IMG_DIR
These are the absolute path to the images directories
2. THUMBNAIL_WIDTH
The PHP script will create a thumbnail (icons) for each image that you upload. In addition when you add an album image that image will also resized automatically.

One more note. If you want to test this gallery on your own computer please make sure you already have **GD library installed**. To check if GD library is installed on your system save the following code and run it.

```
<?php
if (function_exists('imagecreate')) {
    echo 'OK, you already have GD library installed';
} else {
    echo 'Sorry, it seem that GD library is not installed/enabled';
}
?>
```

After taking care of the configurations we'll take a quick tour on the [admin page](#)

Image Gallery Administration Page

Login and logout

Before you can access the admin pages you will need to login first. The login method is very basic we just check the given userid and password against the hardcoded value in the login script. Here's the code :

Example : [admin/login.php](#)

Source : [admin/login.phps](#)

```
$errMsg = "";
if (isset($_POST['txtUserId'])) {
    if ($_POST['txtUserId'] == 'bigbadwolf'
        && $_POST['txtUserpw'] == 'huffnpuff') {

        $_SESSION['isLogin'] = true;
        header('Location: index.php?page=list-album');
        exit;
    } else {
        $errMsg = "Wrong Id/Password";
    }
}
```

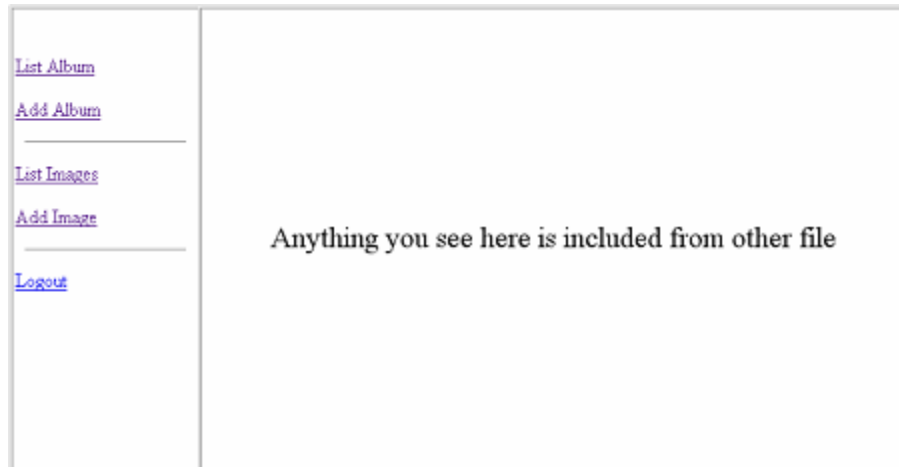
The userid is **bigbadwolf** and the password is **huffnpuff** . If the user enter the matching userid and password we set the session variable `$_SESSION['isLogin']` to true. Then we send a header to redirect the user to the album list.

To see if the user has logged in or not we call the function `checkLogin()` from the admin pages. It merely check the existence of the session variable we speak of earlier and it's value. If the session variable exist and it's value is set to true then we can say that the user has logged in. If you think that this login method is not very good you could easily plug other login method from this [login tutorial](#)

The [logout](#) part is fairly simple also. We just need to set the value of `$_SESSION['isLogin']` to false then redirect the user to the login page. That's it.

Admin Page Layout

If you see the [source code for the admin main page](#) you can see that it act more like a "frame" meaning that it's content is included from other files. Check out the [album list page](#) (you may need to login first).The tables containing the album list are included from [this file](#). Now click the "Add Album" button. The form is included from [this one](#). Here's the snapshot to make it clearer



Take a look at the code snippet below.

Source code : [admin/index.php](#)

```
// ... some code here
```

```
$page = (isset($_GET['page']) && $_GET['page'] != '') ? $_GET['page'] : 'list -album';
```

```
$allowedPages = array('list-album', 'add-album', 'album-detail', 'modify-album', 'list-image', 'add-image', 'image-detail', 'modify-image');
```

```
if (in_array($page, $allowedPages)) {
```

```
    include $page . '.php';
```

```
} else {
```

```
?>
```

```
<table width="100%" border="0" align="center" cellpadding="2" cellspacing="1">
```

```
<tr>
```

```
<td width="30" align="center"><strong>Error : The Page You're Looking
```

```
For Doesn't Exist</strong></td>
```

```
</tr>
```

```
</table>
```

```
<?php
```

```
}
```

```
?>
```

To decide which page to include index.php will look up the `$_GET` variable to see if `$_GET['page']` is available. For example the url for the [image list page](#) is "index.php?page=list-image". So in this case the value of `$_GET['page']` is "list-image" so the file we need to include is "list-image.php". If `$_GET['page']` is empty or doesn't exist then we'll just use "list-album" as the default page.

Before including any file we first check if the intended file is in our list of allowed pages. You see, the bad thing about including other pages is that it can cause **serious security problem** if you don't

take some precautions.

Here's an example. Suppose an evil (or nosy) guy decided to enter this url "index.php?page=../../super-secret-file", then without checking the requested page against the list of allowed pages you could expose some secret code to him or even worse . In the code above you can see that if a user is requesting a page which is not in the list we just display an error message. We don't check if the file actually exist or not. If it's not in the allowed list then we will never include it.

Okay, now let's start making the image gallery from the first part, [adding a new album](#)

Admin : Add New Album

This is a very simple form where you can enter the album name, description and image. After you click the "Add Album" button the script will do the followings :

1. Save the album image, resize it if necessary
2. Save the album information to database

Below is the screenshot of the form:

Album Name	<input type="text"/>
Description	<input type="text"/>
Image	<input type="text"/> <input type="button" value="Browse..."/>
	<input type="button" value="Add Album"/> <input type="button" value="Cancel"/>

And here is the code snippet :

Example : [admin/add-album.php](#)

Source code : [admin/add-album.phps](#)

```
require_once '../library/config.php';
require_once '../library/functions.php';
```

```
if(isset($_POST['txtName']))
{
```

```

$albumName = $_POST['txtName'];
$albumDesc = $_POST['mtxDesc'];

$imgName = $_FILES['fileImage']['name'];
$tmpName = $_FILES['fileImage']['tmp_name'];

// we need to rename the image name just to avoid
// duplicate file names
// first get the file extension
$ext = strrchr($imgName, ".");

// then create a new random name
$newName = md5(rand() * time()) . $ext;

// the album image will be saved here
$imgPath = ALBUM_IMG_DIR . $newName;

// resize all album image
$result = createThumbnail($tmpName, $imgPath, THUMBNAIL_WIDTH);

if (!$result) {
    echo "Error uploading file";
    exit;
}

if (!get_magic_quotes_gpc()) {
    $albumName = addslashes($albumName);
    $albumDesc = addslashes($albumDesc);
}

$query = "INSERT INTO tbl_album (al_name, al_description, al_image, al_date)
VALUES ('$albumName', '$albumDesc', '$newName', NOW())";

mysql_query($query)
or die('Error, add album failed : ' . mysql_error());

// the album is saved, go to the album list
echo "<script>window.location.href='index.php?page=list -album';</script>";
exit;
}

```

Since we save the images as files instead inserting them to the database we need to make sure there won't be any name duplication problem. To prevent this we just generate some random name for

every images that we upload. Take a look at code below :



```
$ext = strrchr($imgName, ".");  
$newName = md5(rand() * time()) . $ext;
```

The first line is to extract the file extension from the file name. As an example let say we upload an image named "hyperalbum.jpg". Then `strrchr("hyperalbum.jpg", ".")` will return ".jpg". On the second line we generate a random number using `rand()` multiply it with current time and generate the hash code using `md5()`. It is a very common practice to use the combination of `md5()`, `rand()` and `time()` functions to generate a random name. After we append the file extension to the new name we can then use it to save the uploaded image.

But before we save the image we need to resize the image if it's too large. As you can see in the [album list](#) we only need small images for the album icons. To make the thumbnail we use `createThumbnail()` function defined in [functions.php](#). Once everything is saved we print a little javascript code to go to the album list page. Note that we **cannot** simply use `header("Location: index.php?page=list-album")` to redirect to the album list page since a call to `header()` will only have an effect when no other output is sent before the call.

Admin : Album List

When your first login to the admin area and after adding a new album you can see this page. There's nothing really interesting on this one. It's just a plain list of albums where we can see the albums we have and how many images on each album. Here is the snapshot :

#	Album Name	Images		
1	 Cars	5	Modify	Delete
2	 One Piece	3	Modify	Delete
Add Album				

In the "Images" column you can see how many images contained in an album. If you click on the number you will go to the image list so can see all the images in a particular album. And I'm sure i

don't need to explain what that button with "Add Album" written on it does.

If you re-read the sql containing the [table definitions](#) of this gallery you can see that tbl_album doesn't contain any column storing the number of images in it. That number is the result of the left join in the sql query. You can see the sql code below.

```
$sql = "SELECT al_id,
        al_name,
        al_image,
        COUNT(im_album_id) AS al_numimage
FROM tbl_album al
    LEFT JOIN tbl_image im ON al.al_id = im.im_album_id
GROUP BY al_id
ORDER BY al_name ";
```

In this query we must use LEFT JOIN instead of INNER JOIN because an album can have zero image in it. If we use INNER JOIN then the empty albums will not be shown in the list.

Now, if you right click on an album icon and view it's properties you can see that the url for the icon is pointing to a PHP script instead of an image. The url looks like this :

viewImage.php?type=album&name=3b6a267a967d7535ff3b1ebc3d9e3c1e.jpg

In this image gallery whenever we would want to display an album or image icon or the full -size image we always use the viewImage.php file instead of linking to the actual image. There are several reasons to do this. The first is so you could move the images directory outside of your web root to prevent leechers from taking all the images.

The image gallery in our example doesn't do this. You could go to the [images directory](#) and list all the images in the gallery. If you set the value of ALBUM_IMG_DIR and GALLERY_IMG_DIR to a directory outside your webroot then you can prevent this. For example if your web root is /home/myname/public_html you can set ALBUM_IMG_DIR to /home/myname/images/album and GALLERY_IMG_DIR to /home/myname/images/gallery/.


The second reason is that you may want to restrict the access to your gallery. For example the visitors must login before they can see the images. In viewImage.php you could check for the session variable to determine that. So if the visitor hasn't login yet you just display some blank or warning images

You can checkout the code [here](#). It's really a simple script which requires two inputs. The type of image you wish to display (album icon, image icon or full size image) and the image file name. Then we only need to set the appropriate headers, read the image file and send it to the browser.

Next we'll see how to [modify & delete an album](#).

Modify & Delete Album

Here is the page where you can modify an album's name, description and image icon

Album Name	<input type="text" value="Cool Images"/>
Description	<input type="text" value="Some cool images that i found while surfing on the net"/>
Image	<div><input type="text" value=""/><input type="button" value="Browse..."/></div>
<div><input type="button" value="Modify"/> <input type="button" value="Cancel"/></div>	

We display the album thumbnail here just to make sure we're updating the right album and because i think the form is way too dull without any images.

After you hit the "Modify" button we just need to save the new name & description to the database. However, before we changing the thumbnail we must first check if a new thumbnail image is provided or not.

Here is the code snippet :

Source : [modify-album.phps](#)

// ... some code here

```
if(isset($_POST['txtName'])) {
    $albumId = $_POST['hidAlbumId'];
    $albumName = $_POST['txtName'];
    $albumDesc = $_POST['mtxDesc'];

    if (!get_magic_quotes_gpc()) {
        $albumName = addslashes($albumName);
        $albumDesc = addslashes($albumDesc);
    }

    if ($_FILES['fleImage']['tmp_name'] != "") {
        $imgName = $_FILES['fleImage']['name'];
        $tmpName = $_FILES['fleImage']['tmp_name'];
```

```

// just like when we add this album
// we will need to rename the image name to avoid
// duplicate file name problem
$newName = md5(rand() * time()) . strrchr($imgName, ".");

// resize the new album image
$result = createThumbnail($tmpName, ALBUM_IMG_DIR . $newName,
                        THUMBNAIL_WIDTH);

if (!$result) {
    echo "Error uploading file";
    exit;
}

// since a new image for this album is specified
// we'll need to delete the old one
$sql = "SELECT al_image
        FROM tbl_album
        WHERE al_id = $albumId";

$result = mysql_query($sql)
        or die('Error, get album info failed. ' .
            mysql_error());
$row = mysql_fetch_assoc($result);
unlink(ALBUM_IMG_DIR . $row['al_image' ]);

$newName = ""$newName"";
} else {
    // don't change the image
    $newName = "al_image";
}

$query = "UPDATE tbl_album
        SET al_name      = '$albumName',
            al_description = '$albumDesc',
            al_image      = $newName
        WHERE al_id = $albumId";

mysql_query($query)
or die('Error, modify album failed : ' . mysql_error());

// after saving the modification go to the detail page
echo "<script>>window.location.href='index.php?page=album-detail&alId=$albumId';</script>";
}

```

```
// ... more code down here
```

As you can see on the code above we update the album thumbnail only if a new image is provided. The process is no different than when we add a new album. After the new image is uploaded and the thumbnail is created successfully we then delete the old one using `unlink()`. That's the php function you'll need when you want to delete a file.

Now, you need to take a look at the above code again but this time focus on this little piece

```
// ... some code here
```

```
if ($_FILES['fileImage']['tmp_name'] != "") {  
    // ... upload the new image & delete the old one  
  
    $newName = ""$newName"";  
} else {  
    $newName = "al_image";  
}  
  
$query = "UPDATE tbl_album  
    SET al_name      = '$albumName',  
        al_description = '$albumDesc',  
        al_image      = $newName  
    WHERE al_id = $albumId";  
  
// ... and more down here
```

If you're wondering why in the if block `$newName` is surrounded by single quotes but in the else block `$newName` is given the value `"al_image"` without the single quotes, here's the explanation. Note that in the sql query string `$albumName` and `$albumDesc` is surrounded by quotes but `$newName` isn't. We need to do this so that when the image is not changed we can set the query to leave the image alone.

Here's an example so you can understand it better. For the first scenario imagine that the admin update an album (id = 123) and modify it's name to "my new album", description is changed to "my new description" and he also give a new image and it is saved under the new name "3b6a267a967d7535ff3b1ebc3d9e3c1e.jpg". The script will go through the if block because the value of `$_FILES['fileImage']['tmp_name']` won't be empty and then the value of `$query` will be ...

```
UPDATE tbl_album SET al_name = 'my new album', al_description = 'my new description',  
al_image = '3b6a267a967d7535ff3b1ebc3d9e3c1e.jpg' WHERE al_id = 123
```

... which will update the image information including the image name.

For the second scenario the admin only change the name and description but leave the image

alone. Now the script will go to the else block and the value of \$query will be ...

```
UPDATE tbl_album SET al_name = 'my new album', al_description = 'my new description',  
al_image = al_image WHERE al_id = 123
```

... which will update the album name and description but leave the old image name as it is.

This is certainly **not** the only way to do it. You can easily add an if else statement and make a new query for each scenario. I'm just showing an alternative.

Delete Album

You can delete an album by clicking the "Delete" link on the album list. You'll see a confirmation box to make sure you really want to delete an album because when you delete album all images are deleted also.

The code for deleting an album is located in the index.php file. The code flow is like this :

1. Get the album id
2. Make a query to get the name of that album and the thumbnail filename. Print an error message if the album doesn't exist
3. If the album exist get images file name and delete the images plus the album icon
4. Delete the album data and the images from the database

Here is the code

Source code : <admin/index.phps>

```
// ... some code on top
```

```
if (isset($_GET['deleteAlbum']) && isset($_GET['album'])) {  
    $albumId = $_GET['album'];
```

```
    // get the album name since we need to display
```

```
    // a message that album 'foo' is deleted
```

```
    $result = mysql_query("SELECT al_name, al_image
```

```
                           FROM tbl_album
```

```
                           WHERE al_id = $albumId")
```

```
    or die('Delete image failed. ' . mysql_error());
```

```
    if (mysql_num_rows($result) == 1) {
```

```
        $row = mysql_fetch_assoc($result);
```

```
        $albumName = $row['al_name'];
```

```
        $albumImage = $row['al_image'];
```

```

// get the image filenames first so we can delete them
// from the server
$result = mysql_query("SELECT im_image, im_thumbnail
                      FROM tbl_image
                      WHERE im_album_id = $albumId")
    or die(mysql_error());
while ($row = mysql_fetch_assoc($result)) {
    unlink(GALLERY_IMG_DIR . $row['im_image']);
    unlink(GALLERY_IMG_DIR . 'thumbnail/' .
        $row['im_thumbnail']);
}

unlink(ALBUM_IMG_DIR . $albumImage);

$result = mysql_query("DELETE FROM tbl_image
                      WHERE im_album_id = $albumId")
    or die('Delete image failed. ' . mysql_error());
$result = mysql_query("DELETE FROM tbl_album
                      WHERE al_id = $albumId")
    or die('Delete album failed. ' . mysql_error());

// album deleted successfully, let the user know about it
echo "<p align=center>Album '$albumName' deleted.</p>";
} else {
    echo "<p align=center>Cannot delete a non-existent album.</p>";
}
}

// ... and some more on the bottom

```

Now we start [adding some images into the albums](#)

Admin : Add Image

Adding a new image is easy. Just select in which album you want the image stored, supply the image name, description, add the image and click the "Add Image" button.

Here's the form screenshot :

Album	One Piece ▼
Image Title	<input type="text"/>
Description	<div></div>
Image	<input type="text"/> <input type="button" value="Browse..."/>
	<input type="button" value="Add Image"/> <input type="button" value="Cancel"/>





The image title is limited to 64 characters however there is no restriction on the length of the image description. And, you are free to use **any html tags** in the description.

The code for this part is very similar to the one when adding new album so i won't explain it in detail here. You could read the source code and i'm sure you'll understand. The difference is that the image is stored in GALLERY_IMG_DIR and the image icon and store it in GALLERY_IMG_DIR/thumbnail.

Image List

After you add a new image or when you click on "List Image" on the left navigation link you can see the list of all images. By default this page will show all images **from all albums**. To show images just from one album you can select the album name from the combo box on the top right corner. When you select an album it will trigger the onChange event of the combo box and call a javascript function called viewImage() along with the id of the selected album. The function will then redirect you to the appropriate list.

Album : - All Album - 

#	Image	Date		
1	 All strawhats in the snow	2006-05-07	Modify	Delete
2	 CLK Coupe	2006-05-06	Modify	Delete
3	 CLS	2006-05-06	Modify	Delete
4		2006-05-06	Modify	Delete

The javascript code is show below.

Source code : [admin/index.phps](#)


```
function viewImage(albumId) {
  if (albumId != "") {
    window.location.href = 'index.php?page=list -image&album=' +
      albumId;
  } else {
    window.location.href = 'index.php?page=list -image';
  }
}
```

When you select a specific album from the combo box t he album id is sent to this function. The if block is executed and you go to the page for that specific album. But if you select the " --- All Album--" option then the value of albumId will be an empty string. The else block is executed and you'll go the the default image list.

Next, [modify and delete image](#)

Modify & Delete Image

Here is the snapshot of the modify image form. It's very similar to the form to add the image except for the image thumbnail. Clicking on the thumbnail will call the javascript function `viewLargeImage()` which will open a popup displaying the full-sized image. If you checkout the [form code](#) you can see that we use `htmlspecialchars()` function when printing the image description in the textarea. Since html tags are allowed in the description we need that function so the html code in the description won't screw up the page.

Album	One Piece ▼
Image Title	Pigeon Guy (a.k.a Lucci)
Description	<p><p>This scene happen in the first fight between the strawhats and cp9. After the cp9 beat up luffy and zoro, lucci decided to show "something interesting" which is his transformation to half-beast mode. He has the power of neko neko no mi, model leopard. The original image is in black and white. This colored image is the winner of the coloring competition held by the-site-that-doesnt-exist. I can't remember who the winner was so i can't give him proper credit. I wonder if i'm allowed to put this picture online...</p></p>
Image	 <input type="button" value="Browse..."/>
	<input type="button" value="Modify Image"/> <input type="button" value="Cancel"/>

After you click the "Modify Image" button we continue saving the new information and also save the new image (if a new one is provided). The process is just a repetition of the code to modify album data and image. The difference is that now we modify two images, the full -sized image and the thumbnail.

Example : <admin/index.php?page=modify-image&imgId=13>

Source code : <admin/modify-image.phps>

```
// ... some code here
```

```
if (isset($_POST['txtTitle'])) {
    $albumId = $_POST['cboAlbum'];
    $imgTitle = $_POST['txtTitle'];
    $imgDesc = $_POST['mtxDesc'];

    if ($_FILES['fleImage']['tmp_name'] != "") {
```

```

$images = uploadImage('fileImage', GALLERY_IMG_DIR);

if ($images['image'] == " && $images['thumbnail'] == ") {
    echo "Error uploading file";
    exit;
}

$image    = "" . $images['image'] . "";
$thumbnail = "" . $images['thumbnail'] . "";

$sql = "SELECT im_image, im_thumbnail
        FROM tbl_image
        WHERE im_id = $imgId";

$result = mysql_query($sql)
        or die('Error, get image info failed. ' .
            mysql_error());
$row = mysql_fetch_assoc($result);
unlink(GALLERY_IMG_DIR . $row['im_image']);
unlink(GALLERY_IMG_DIR . 'thumbnail/' . $row['im_thumbnail']);
} else {
    // the old image is not replaced
    $image    = "im_image";
    $thumbnail = "im_thumbnail";
}

if (!get_magic_quotes_gpc()) {
    $albumName = addslashes($albumName);
    $albumDesc = addslashes($albumDesc);
}

$sql = "UPDATE tbl_image
        SET im_album_id  = $albumId,
            im_title     = '$imgTitle',
            im_description = '$imgDesc',
            im_image      = $image,
            im_thumbnail  = $thumbnail,
            im_date       = NOW()
        WHERE im_id = $imgId";

mysql_query($sql) or die('Error, update image failed : ' .
    mysql_error());

echo "<script>window.location.href = 'index.php?page=image-detail&imgId=$imgId';</script>";
}

```

```
// ... more code here
```

Delete Image

To delete an image we just need the image id and the album id. Then we get the image and icon name so we can delete them from the server. Once the files are deleted we continue removing the image information from the database.

Source code : [admin/list-image.phps](#)

```
// ... a little code here
```

```
if (isset($_GET['delete']) && isset($_GET['album']) &&
    isset($_GET['imgId'])) {
    // get the image file name so we
    // can delete it from the server
    $sql = "SELECT im_image, im_thumbnail
            FROM tbl_image
            WHERE im_id = {$_GET['imgId']}
              AND im_album_id = {$_GET['album']}";
    $result = mysql_query($sql)
        or die('Delete album failed. ' . mysql_error());
    if (mysql_num_rows($result) == 1) {
        $row = mysql_fetch_assoc($result);

        // remove the image and the thumbnail from the server
        unlink(GALLERY_IMG_DIR . $row['im_image']);
        unlink(GALLERY_IMG_DIR . 'thumbnail/' . $row['im_thumbnail']);

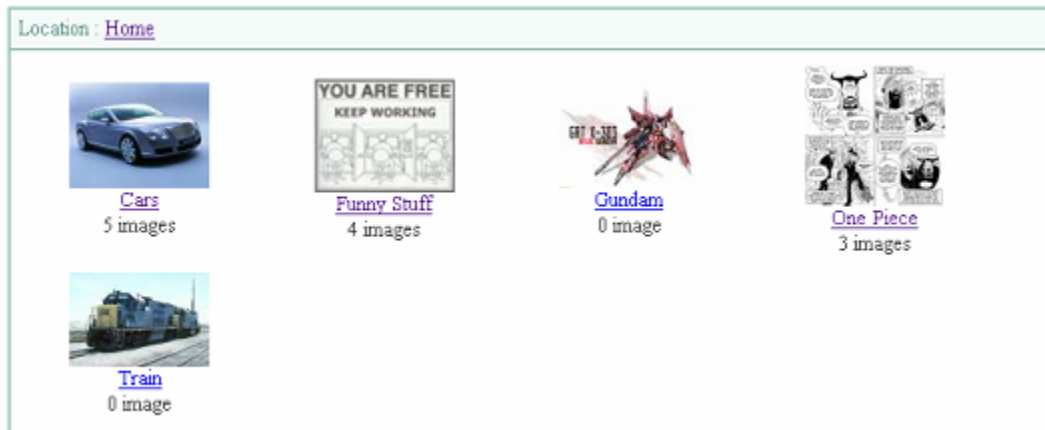
        // and then remove the database entry
        $sql = "DELETE FROM tbl_image
                WHERE im_id = {$_GET['imgId']}
                AND im_album_id = {$_GET['album']}";
        mysql_query($sql)
        or die('Delete album failed. ' . mysql_error());
    }
}
```

```
// ... a little code there
```

Next, we continue to the "ordinary user" part. Starting from the [album list](#).

Album List

Here is the page that the gallery visitors see. The code behind it is mostly the same as the one in the admin pages. For instance the album list, image list and image detail page are pretty much the same code. The main page (index.php) also act like a frame just like the ad min main page. The big difference is that we need to make these pages more pretty than the admin pages. I added some [CSS](#) to make [these pages](#) look better but since my web design skill really sucks all i can add is just some greenish look for it.



The other difference is the way the thumbnails are arranged. On the admin pages we list the albums and images in a simple numbered list in a table. But now the thumbnails in the album list and image list are arranged in rows and columns. It look better that way.

Coding the script to display the thumbnails in rows and columns is a little bit tricky (just a little bit). Here is the code that does it.

```
// ... some code here
```

```
echo '<table width="700" border="0" cellspacing="1" cellpadding="2" align="center">';
```

```
$colsPerRow = 4;
```

```
// width of each column in percent
```

```
$colWidth = (int)(100/$colsPerRow);
```

```
// ... more code here
```

First we specify how many columns we want in one table row. For this gallery we set the value to **four images in one row**. Change this value and the code will automatically rearranged the way the thumbnails are displayed. From the value of \$colsPerRow we then calculate the value of each **column width** using this formula : $\$colWidth = (int)(100/\$colsPerRow)$. So if \$colsPerRow is four

then the width is 25 (in percent) and if we set \$colsPerRow value to three then the column width will be 33 (percent). The (int) part in the formula works just like the mathematical function floor(). It round down a fraction to it's closest integer. You can rewrite the formula into this : \$colWidth = floor(100/\$colsPerRow)but i simply prefer using (int).

```
// ... previous code
while ($row = mysql_fetch_assoc($result)) {
    if ($i % $colsPerRow == 0) {
        // start a new row
        echo '<tr>';
    }

    $numImages = $row['al_numimage'] > 1 ?
        $row['al_numimage'] . ' images'
        : $row['al_numimage'] . ' image';

    echo '<td width="' . $colWidth . '%">' .
        '<a href="index.php?page=list-image&album=' . $row['al_id'] . '">' .
        '' .
        '<br>' . $row['al_name'] . '</a><br />' . $numImages . '</td>';

    if ($i % $colsPerRow == $colsPerRow - 1) {
        // end this row
        echo '</tr>';
    }

    $i += 1;
}

// ... more to come
```

In the while loop we start arranging the thumbnails. Since we build the table rows dynamically we must know when to start and end a row. Here we use the **modulo** of a counter (\$i) with the number of columns on each row (\$colsPerRow). If the result is zero then we start a new row by printing <tr> and if the value is \$colsPerRow - 1 we end that row by printing </tr>.

Making the columns is the easy part. Just print the opening tag (<td>) fill it with the thumbnail, album name and how many images in that album then print the closing tag (</td>).

After the while loop complete there is one more work to be done. That is to check if the last row is full or not. From the snapshot above you can see that we have five albums so the last row only contain one thumbnail. If we just finish the code here and do nothing the layout **could** be broken (not always, depending on how smart your browser is). To mend it we print blank columns to fill in the empty spaces.

To find out how many blank columns needed we use the last value of the counter (\$i) and keep

increasing it's value by one until it's modulo with \$colsPerRow reach zero, like this ...

```
// ... previous code
```

```
// print blank columns
if ($i % $colsPerRow != 0) {
  while ($i++ % $colsPerRow != 0) {
    echo '<td width="" . $colWidth . "%">&nbsp;</td>';
  }
  echo '</tr>';
}

echo '</table>';
```

To make things clearer take a look at the snapshot below. It's the same snapshot of the album list above but this time the table border is visible so you can see that on the second row we have printed three empty columns.

 <u>Cars</u> 5 images	 <u>Funny Stuff</u> 4 images	 <u>Gundam</u> 0 image	 <u>One Piece</u> 3 images
 <u>Train</u> 0 image			

We can now proceed to the [image list page](#)

Image List & Detail

Clicking on one of the album icon from the album list page will bring you to this page. The layout is similar to the album list which means the code is also similar so no need to repeat the explanation here.



Click any of the image icons and you will go to the image detail page.

Image Detail

This is the main stage of the gallery which displays the full -sized image and the description along with other trinkets. Take a look at [this one page](#) . You may feel that this is the wrong way to design a web page because the image is too large to fit in the screen. But personally i prefer preserving the original art instead of resizing it to fit into the page design

Now it's time to bring your attention to that navigation link that you see on top of every page. I'm talking about this one :

Location : [Home](#) > [One Piece](#) > [All strawhats in the snow](#)

This kind of navigation link is widely known as breadcrumbs. Everytime you browse around the albums and images the breadcrumb always display your current location in the gallery. I'm sure you've seen this many times when surfing the web. This simple piece of navigation is very convenient for the visitors to move around the gallery, moving back from the image detail page to the image list or the album list.

The function needed to display the breadcrumb is called `showBreadcrumb()`. You can find this function in `library/functions.php`. Here's the function code :

Source code : [library/functions.phps](#)

```
function showBreadcrumb()
{
    if (isset($_GET['album'])) {
        $album = $_GET['album'];
        $sql = "SELECT al_name
                FROM tbl_album
                WHERE al_id = $album";
```



```

$result = mysql_query($sql)
    or die('Error, get album name failed. ' .
        mysql_error());
$row = mysql_fetch_assoc($result);
echo ' &gt; <a href="index.php?page=list-image&album=' .
    $album . '">' . $row['al_name'] . '</a>';

if (isset($_GET['image'])) {
    $image = $_GET['image'];
    $sql = "SELECT im_title
        FROM tbl_image
        WHERE im_id = $image";

    $result = mysql_query($sql)
        or die('Error, get image name failed. ' .
            mysql_error());
    $row = mysql_fetch_assoc($result);

    echo ' &gt; <a href="index.php?page=image-detail&album=' . $album . '&image=' .
        $image . '">' . $row['im_title'] . '</a>';

}
}
}

```

The first if statement check if there's an album id in the page url. If an album id is found we create a link to the album. The next if block check if an image id is available in the url and if we found one we create a link to the image (which is the currently displayed page)

There's another navigation means in that page. If you scroll down the page you can see the next and previous link. Using this link you can navigate from one image to another that are still in one album.

To find the previous image we search the database for **one** image in the same album as current image where the id is less than the current image id.

// ... above here is the sql query to fetch the image detail.

```

// $image contain the image's information
$image = mysql_fetch_assoc($result);

```

```

// set the initial value for previous and next image id
$prev = $next = 0;

```

```

// get the previous image
$sql = "SELECT im_id
    FROM tbl_image

```

```

WHERE im_id < $imageId
AND im_album_id = {$image['im_album_id']}
ORDER BY im_id DESC
LIMIT 0, 1";

```

```
$result = mysql_query($sql) or die('Error, get image info failed. ' . mysql_error());
```

```

if (mysql_num_rows($result) > 0) {
    $row = mysql_fetch_assoc($result);
    $prev = $row['im_id'];
}

```

```
// ... the code to get the next image
```

I highlighted the ORDER BY clause in the above code because it is really important. We really need it since there is **no guarantee** that the database will store the records in orderly fashion. For example, after you update an image the internal order of the records in the database would change so without the ORDER BY clause we get an image which id is less than the current image id **BUT** it may not be the closest one. So if the current image id is 20 and there are three other images whose id are 15, 17 and 18 then without the ORDER BY clause the sql query may return any one of those ids while the one we actually want is image id 18.

Using the ORDER BY clause the query will return the images in descending order (18, 17, 15). And because we add the LIMIT clause the query will return only one record. The one containing image id 18.

The following code is the one to fetch the next image. For this one we just need to find an image in the same album as current image where the id is greater than the current image id. Note that in this code the order clause use ASC instead of DESC so that the returned record will be sorted in ascending order.

```
// ... the code to get previous image
```

```

// get the next image
$sql = "SELECT im_id
FROM tbl_image
WHERE im_id > $imageId
AND im_album_id = {$image['im_album_id']}
ORDER BY im_id ASC
LIMIT 0, 1";

```

```
$result = mysql_query($sql) or die('Error, get image info failed. ' . mysql_error());
```

```

if (mysql_num_rows($result) > 0) {
    $row = mysql_fetch_assoc($result);
    $next = $row['im_id'];
}

```

}

This is the end of the image gallery tutorial. To download the code for this image gallery [click here](#) . Be sure to modify the configuration file before running the script.

Finding Web Hosting For PHP And MySQL

There are a lot of things to consider when choosing a web hosting company. But one thing for sure is that **price is no longer important**. Web hosting is a very competitive field so the price just keeps getting lower and lower. It is so easy to find cheap web hosting for PHP and MySQL

The important things to consider when choosing PHP and MySQL web hosting are :

1. **PHP and MySQL versions**

If a web hosting company say that they support PHP 4 make sure it's the latest version not the 4.0.1 version. Same thing for MySQL but with an extra precaution. Some webhosting company only support MyISAM tables, so if you need InnoDB make sure you ask if it's available.

2. **Specific setting / feature**

For example, you want to create a PHP script which change a file's permission using chmod(). Guess what, if PHP is run as the server your code won't work and there is nothing you can do about it. This exact thing happen to me with this website. I didn't foresee that i would make such application. Anyway just try imagining what you want to do with your web site and if you are uncertain whether a web host will support a feature, just ask.

3. **Connection speed**

All web hosting company claim that they have fast connection speed, but you have to test it to believe it. Use [NetMechanic's free service](#) to test the web host company's homepage. If the result is good then the claim is most likely true.

4. **Data transfer**

When you just started a website it doesn't matter much. But as your website grows you have to make sure you have enough bandwidth. One or two gigabytes (Gb) per month is more than enough for most web sites

5. **Access to raw log file and online statistics (log file analyzer)**

If you are serious in building a website, for commercial purposes for example, this is very critical. You can discover a lot of information from log files like the keywords used to find your website, most visited pages, peak times etc.

6. **Storage space**

Measure your own website, start small but leave room for expansion. For a small website 15

megabytes is plenty.

7. **Customer support**

They have to be there when you need them, period. Try asking some questions before you decide to go with a hosting company. If it takes more than 24 hours for them to reply then find another host.

For those of you who just want to experiment you can use MySQL PHP free hosting. There are probably hundreds of them out there, you just need to pick one. They usually place banners or other kinds of advertising on your web page and most are slow. Anyway you can't expect much from free services.

Actually even if you just experimenting with PHP and MySQL it's better to have your own web site. Free hosting usually have lots of restriction so you really can't do much experiment like opening a socket connection and stuff like that.

There's this one website that rank ten [PHP MySQL website hosting](#) company based on price, quality, performance and features. If you need a second opinion you can go there.

A bit off topic here. If you intend to build an e-business (for yourself or for a client) instead of just a website you should consider about [Site Build It](#). It's an all-in-one site-building, site-hosting, and site-marketing service. It even outperformed Microsoft's bCentral and Yahoo Web Hosting Pro. The only drawback is that the HTML templates provided look a bit lame. But since now you can upload your own template i guess it's not a big deal anymore.

Freelance PHP MySQL Jobs

As far as i know [Scriptlance](#) is one of the best place to find freelance programming jobs. Each day you can find about 50 new job postings. Most of the jobs will require advance knowledge on PHP and MySQL like creating a dating website, car rental website, or a shopping cart but some are quite easy like creating a site counter and signup form.

When you are new to Scriptlance it's better if you stick to find these easy jobs first and try to get good reviews. As you go learn how to do the more difficult ones. Check out the job (project) descriptions just to see what kind of jobs in demand and keep it in a list. Because there may be similar jobs in the future you should create a draft on the design (algorithm, database, process, etc). That way when you get enough experience and you see a similar job posted you can complete it faster

If you want to do some freelance jobs here's a little checklist that might help :

- Get a clear description of the project. Ask whenever you are unsure about something. Create

a demo / mockup whenever possible so you can be sure that you and your client are talking about the same thing.

- Can you do it? Seriously, can you complete each and every features requested? Your client won't be happy if you say you can complete the job but fail to do so. One quick way to be sure is to make a prototype before even bidding the project then propose it to your (future) client to see if it is what she expected.
- Don't be too optimistic. If you think you can complete a job in one day make sure you can do it in one day. This include all the testing and bug fixing.
- Can you accept the payment? Some will want to pay using PayPal if you can accept PayPal it's no problem but if you can't maybe you should consider finding another project.
- Ask for the PHP and MySQL version used by your client and develop the project using the same version. This will reduce the risk of creating buggy scripts just because your version and your client's version is different
- Test and retest.
- Be ready to fix bugs. All software have bugs, but make sure you are ready to fix them when it's found. Your clients will certainly expect a prompt response so give it to them. Even if your script is buggy but if you are prompt in responding and fixing it you can get good reviews.
- **KYOB**
That's short for **Kick Your Own Butt** ! Freelancing requires hard discipline and because there's no boss or supervisor breathing on your neck it can be hard to feel that you **are** working. Sure, you still have deadlines, but the client is probably on the other side of the planet and most likely you will never see her face to face. It just doesn't feel the same as an ordinary job. KYOB is probably the most important ability you must master if you really want to be a successful freelancer.

By the way, if you're interested i suggest you read about [Site Build It](#).

From my personal experience you can really earn a substantial income by making a website about something you know and enjoy. Your website doesn't always need to be related to computers, programming, web development or internet. You can make a site related your hobby if you want to. SBI's step by step process really works.

I do feel you really should [take a peek](#) on SBI and when you still have doubts just go [ask a real humang being about it](#).

PHP MySQL Resources

- [PHP Homepage](#)
Of course, I have to list this site here. This is where you can to download the PHP bundle and

documentations too

It also has a lot of other important resources such as links to various PHP projects. You can also [subscribe to PHP mailing list](#) from this website. There mailing list are divided to cover specific issues, so you may want to choose the most appropriate to your needs.

- [MySQL Homepage](#)
Here you can download the latest MySQL release, get the MySQL news update. The mailing list is also a great resources for anyone who want to build dynamic websites using MySQL, so don't to join.
- [Zend](#)
PHP is powered by the Zend engine. This website is the official homepage of the company who built the engine. Here you can get the Zend Optimizer. It's a very useful tool which can give your PHP scripts a 40-100% increase in speed. It also has [PHP programming contest](#), a directory of [free PHP scripts and applications](#), [articles and tutorials](#).

Zend also offer a [PHP Certification](#). They said it can "Differentiate yourself from competitors when looking for a new job or at your annual salary review". Maybe I'll try to get that certification and see if I can get a better salary : -). By the way, you can get a free chapter of the Zend PHP Certification Guide in Zend homepage. It's really an interesting reading.
- [PHPBuilder](#)
Great articles and tutorials in PHP programming. Anyone from a beginner to an expert can really learn from this site.
- [Scriptlance](#)
One of the best place to find freelance programming jobs. Last time i check there are over 300 open project related to PHP. It' really a good place to find freelance projects
- [DevShed](#)
Lots of resources for open source stuff. Not all articles here is related to PHP or MySQL. Actually you can get a whole breed of web development issues here.
- [Eclipse.org](#)
My favourite PHP editor. It's heavy (use lots of memory) but still great. Built in support for CVS (Concurrent Versioning System) makes it really easy to manage projects with multiple developers. By the way if you want to use Eclipse **don't forget** to get the PHP plugin from [sourceforge.net](#) or from [phpeclipse.de](#)
- [PHPPatterns.com](#)
Learn how to apply Design Pattern in PHP. If you already learn Object Oriented Programming (OOP) in PHP this site can help you applying the concept of OOP better. There's a funny article here about how far you can go in applying patterns. It's about writing a Hello World script in the [super hard way](#).

- [Sitesell Value Exchange](#)
A real Link Exchanger that works. You can find high -value, similarly themed sites, and exchange links to truly increase your link popularity, in a way that the search engines love.
- [Vision.To Design](#)
Web Applications, PHP/MySQL Advanced Solutions, Web Design
- [dbQwikSite](#)
It's a software that generates PHP code against your MySQL database so you can speed up your coding.
- [W3C Link Checker](#)
Use this tool to make sure that your website has no dead links. A very very useful tool. If you're building a website you shouldn't go live before checking the link using this.
- [ScriptsToProfit.com PHP/MySQL Website Scripts](#)
Offering quality free and affordable PHP/MySQL website scripts to make your online business more interactive.
- [Big Webmaster Directory](#)
Webmaster tools, scripts and tutorials.
- [ScriptSearch.com](#)
You can find free scripts, source code, books and code examples in here.
- [PHPFreaks.com](#)
PHP and MySQL Resources. They have a linux forum too
- [URL.biz - Programming](#)
- [Search engine optimisation at NetRegistry](#)
Helping more businesses in Australia with search engine submission & optimisation