

Satya's

PHP – MySQL -NetBeans



Satyajohnny

Greembuds Software Technologies

Satya's

- **PHP Tutorials**

- [Introduction](#)
- [What is PHP](#)
- [PHP Installation](#)
- [PHP Syntax](#)
- [Comments in PHP](#)
- [PHP Variables](#)
- [PHP String Variables](#)
- [PHP Constants](#)
- [PHP Operators](#)
- [PHP If ... Else Conditional Statements](#)
- [PHP Switch Statement](#)
- [PHP Arrays](#)
- [PHP Sorting Arrays](#)
- [PHP While Loops](#)
- [PHP for Loop](#)
- [PHP Functions](#)
- [PHP Forms Tutorial](#)
- [PHP \\$_GET Variable](#)
- [PHP \\$_POST Variable](#)

- **PHP Advanced Tutorials**

- [PHP Multidimensional Arrays](#)
- [PHP Date\(\) and Time\(\) Function](#)
- [PHP Include Files](#)
- [PHP File Handling](#)
- [PHP File Upload](#)
- [PHP Cookies](#)
- [PHP Sessions](#)
- [PHP Sending Emails](#)
- [PHP Error Handling](#)
- [PHP Bugs Debugging](#)
- [PHP Exception Handling](#)

- **PHP Database**

- [PHP MySQL Introduction](#)
- [PHP MySQL Connect](#)
- [PHP Create Db / Tb](#)
- [PHP Insert data into MySQL](#)
- [PHP MySQL Select](#)
- [PHP MySQL Where](#)
- [PHP MySQL Order By](#)
- [PHP MySQL Update Data](#)
- [PHP MySQL Delete Data](#)
- [PHP MySQL Creating Paging](#)
- [PHP MySQL Backup](#)
- [PHP ODBC Database](#)

- **PHP AND XML**

- [PHP and XML](#)

- **PHP and AJAX**

- [PHP And AJAX](#)
- [PHP And AJAX Example](#)
- [PHP AJAX And MySQL](#)
- [PHP AJAX And XML](#)
- [PHP AJAX Live Search](#)
- [PHP AJAX RSS Reader](#)
- [PHP AJAX PollP](#)

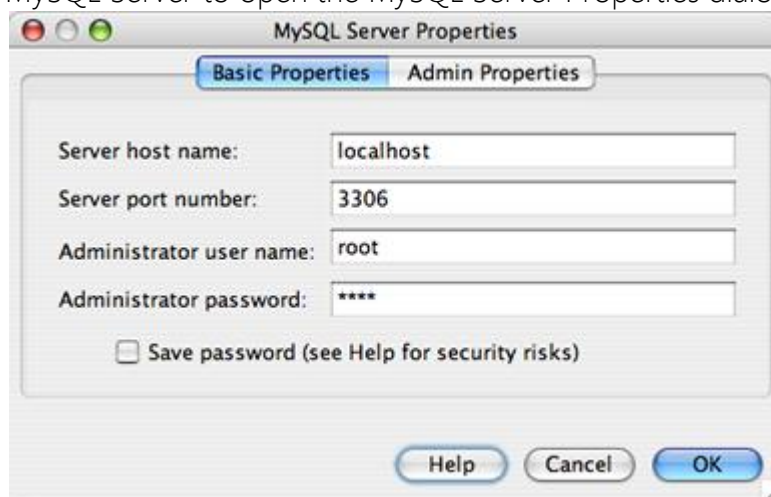
NetBeans IDE PHP

Software or Resource	Version Required
NetBeans IDE	PHP download bundle
A PHP engine	Version 5
A web server	Apache HTTP Server 2.2 is recommended.
A PHP debugger	XDebug 2.0 or later

Configuring MySQL Server Properties

NetBeans IDE comes bundled with support for the MySQL RDBMS. Before you can access the MySQL Database Server in NetBeans IDE, you must configure the MySQL Server properties.

1. Right-click the Databases node in the Services window and choose Register MySQL Server to open the MySQL Server Properties dialog box.



2. Confirm that the server host name and port are correct.

Notice that the IDE enters localhost as the default server host name and 3306 as the default server port number.

3. Enter the Administrator user name (if not displayed).

Note: You need administrative access to be able to create and remove databases.

4. Enter the Administrator password. The default is set to blank.

Note: A blank password can also be a password.

5. Click the Admin Properties tab at the top of the dialog box.
The Admin Properties tab is then displayed, allowing you to enter information for controlling the MySQL Server.
6. In the Path/URL to admin tool field, type or browse to the location of your MySQL Administration application such as the MySQL Admin Tool, PhpMyAdmin, or other web-based administration tools.

Note: mysqladmin is the MySQL admin tool found in the bin folder of the MySQL installation directory. It is a command-line tool and not ideal for use with the IDE.

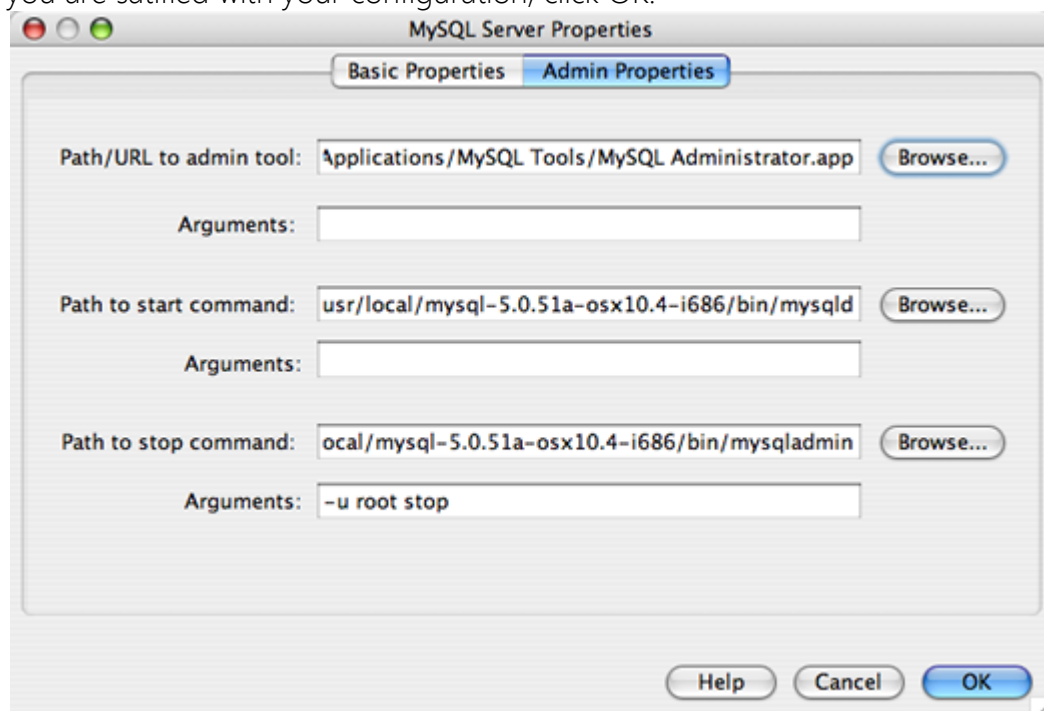
Type any arguments for the admin tool in the Arguments field.

7. In the Path to start command, type or browse to the location of the MySQL start command. To find the start command, look for mysqld in the bin folder of the MySQL installation directory.

Note: The recommended binary for Unix and NetWare is mysql_safe. The start command may also vary if MySQL was installed as part of an AMP installation.

Type any arguments for the start command in the Arguments field.

8. In the Path to stop command field, type or browse to the location of the MySQL stop command. This is usually the path to mysqladmin in the bin folder of the MySQL installation directory. If the command is mysqladmin, in the Arguments field, type -u root stop to grant root permissions for stopping the server.
9. When finished, the Admin Properties tab should resemble the following figure. If you are satisfied with your configuration, click OK.



1.C:\Program Files\MySQL\MySQL Tools for 5.0\MySQLAdministrator.exe

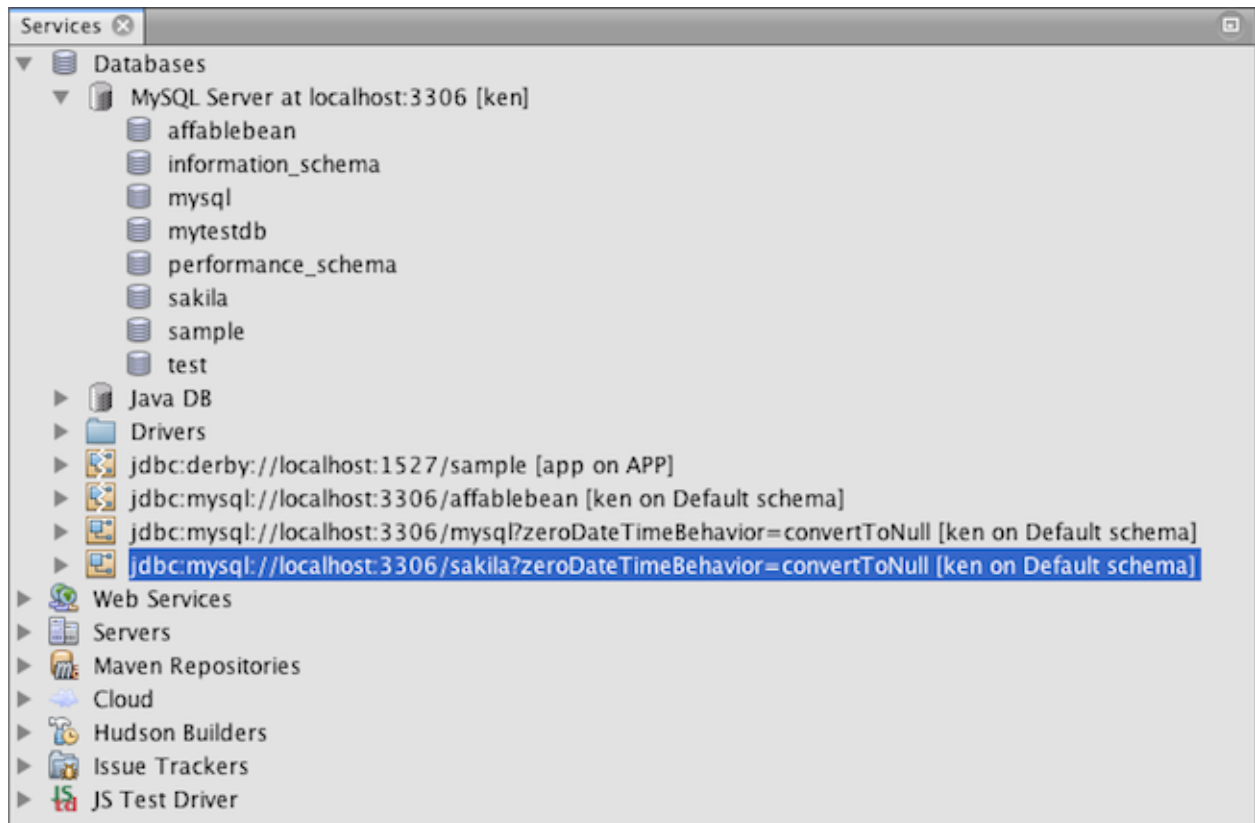
2. C:\Program Files\MySQL\MySQL Server 5.1\bin\mysqld.exe

3. C:\Program Files\MySQL\MySQL Server 5.1\bin\mysqladmin.exe

Starting the MySQL Server

Riglick → Start

right-click the Databases > MySQL Server node in the Services window and choose Connect. You might be prompted to supply a password to connect to the server.



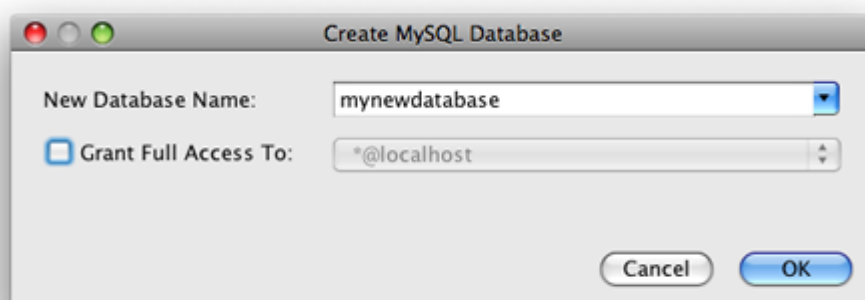
Creating and Connecting to the Database Instance

A common way of interacting with databases is through an SQL editor. NetBeans IDE has a built-in SQL Editor for this purpose. The SQL Editor is generally accessible via the Execute Command option from the right-click menu of the connection node (or of the connection node's child nodes). Now that you are connected to the MySQL server, you can create a new database instance using the SQL Editor. For purposes of this tutorial, create an instance called MyNewDatabase:

1. In the IDE's Services window, right-click the MySQL Server node and choose Create Database.

The Create MySQL Database dialog box opens.

2. In the Create MySQL Database dialog box, type the name of the new database. We will use MyNewDatabase for this tutorial. Leave the checkbox unselected at this time.



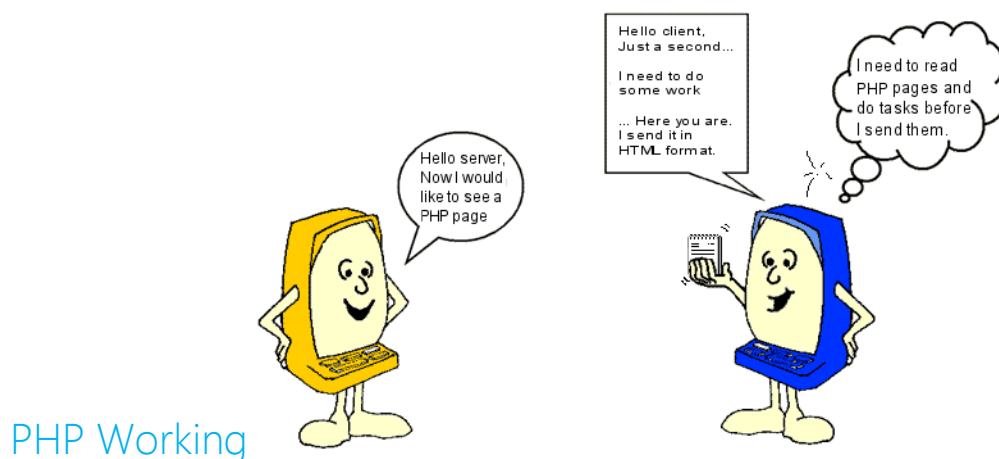
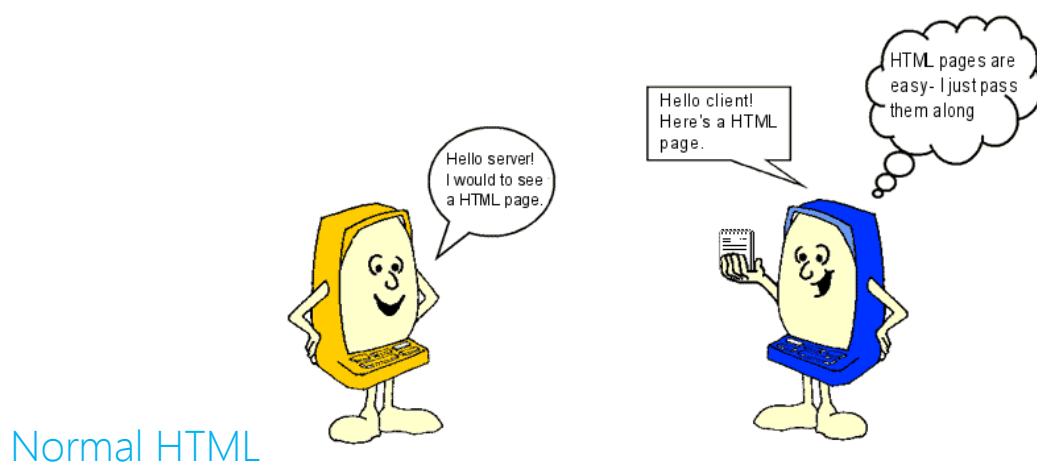
From <http://www.mytoptutorials.com/php/introduction/>

- ✓ PHP is a server-side technology.
- ✓ PHP files can contain text, HTML, JavaScript code, and PHP code
- ✓ PHP code are executed on the server, and the result is returned to the browser as plain HTML
- ✓ PHP files have a default file extension of ".php"
- ✓ PHP can generate dynamic page content
- ✓ PHP can create, open, read, write, and close files on the server
- ✓ PHP can collect form data
- ✓ PHP can send and receive cookies
- ✓ PHP can add, delete, modify data in your database
- ✓ PHP can restrict users to access some pages on your website
- ✓ PHP can encrypt data
- ✓ PHP runs on different platforms (Windows, Linux, Unix, Mac OS X, etc.)
- ✓ PHP is compatible with almost all servers used today (Apache, IIS, etc.)
- ✓ PHP has support for a wide range of databases

- ✓ PHP is free. Download it from the official PHP resource:
www.php.net
- ✓ PHP is easy to learn and runs efficiently on the server side

How does PHP work?

The best way to explain how PHP works is by comparing it with standard HTML. Imagine you type the address of an HTML document (e.g. <http://www.mysite.com/page.htm>) in the address line of the browser. This way you request an HTML page. It could be illustrated like this:



Basic PHP Syntax

```
<?php
// PHP code goes here
?>
```

Comments in PHP

```
<?
# This is a comment, and
# This is the second line of the comment
// This is a comment too. Each style comments only
print "An example with single line comments";
?>
```

PHP Variables

All variables in PHP are denoted with a leading dollar sign (\$). PHP does a good job of automatically converting types from one to another when necessary.

PHP has no command for declaring a variable.

A variable is created the moment you first assign a value to it:

```
$txt="Hello world!";
$x=5;
```

The escape-sequence replacements are:

\n is replaced by the newline character

\r is replaced by the carriage-return character

\t is replaced by the tab character

\\$ is replaced by the dollar sign itself (\$)

\\" is replaced by a single double-quote (")

\\ is replaced by a single backslash (\)

String Variables in PHP

```
<?php
$txt="Hello world!";
echo $txt;
?>
```

Output Result:

Hello world!

The PHP strlen() function

Sometimes it is useful to know the length of a string value.

The PHP strpos() function

The strpos() function is used to search for a character or a specific text within a string.

constant() function:

As indicated by the name, this function will return the value of the constant.

?php

```
define("MINSIZE", 50);
```

```
echo MINSIZE;
```

```
echo constant("MINSIZE"); // same thing as the previous line
```

?>

PHP Magic constants:

Name	Description
__LINE__	The current line number of the file.
__FILE__	The full path and filename of the file. If used inside an include, the name of the included file is returned. Since PHP 4.0.2, __FILE__ always contains an absolute path whereas in older versions it contained relative path under some circumstances.
__FUNCTION__	The function name. (Added in PHP 4.3.0) As of PHP 5 this constant returns the function name as it was declared (case-sensitive). In PHP 4 its value is always lowercased.
__CLASS__	The class name. (Added in PHP 4.3.0) As of PHP 5 this constant returns the class name as it was declared (case-sensitive). In PHP 4 its value is always lowercased.
__METHOD__	The class method name. (Added in PHP 5.0.0) The method name is returned as it was declared (case-sensitive).

PHP Conditional Statements

1.if

```
<?php
```

```
$t=date("H");
```

```
if ($t<"20")
```

```
{
```

```
    echo "Have a good day!";
```

```
}
```

```
?>
```

If-else

```
html>  
  
<body>  
  
<?php  
$d=date("D");  
if ($d=="Fri")  
    echo "Have a nice weekend!";  
else  
    echo "Have a nice day!";  
?>  
  
</body>  
  
</html>
```

elseif

```
<html>  
  
<body>  
  
<?php  
$d=date("D");  
if ($d=="Fri")
```

```
    echo "Have a nice weekend!";  
elseif ($d=="Sun")  
    echo "Have a nice Sunday!";  
else  
    echo "Have a nice day!";  
?>  
</body> </html>
```

Switich

```
html>  
<body>  
<?php  
$d=date("D");  
switch ($d)  
{  
    case "Mon":  
        echo "Today is Monday";  
        break;  
    case "Tue":  
        echo "Today is Tuesday";  
        break;  
    case "Wed":  
        echo "Today is Wednesday";  
        break;  
    case "Thu":
```

```
    echo "Today is Thursday";

    break;

case "Fri":

    echo "Today is Friday";

    break;

case "Sat":

    echo "Today is Saturday";

    break;

case "Sun":

    echo "Today is Sunday";

    break;

default:

    echo "Wonder which day is this ?";

}

?>

</body>

</html>
```

Create an Array in PHP

In PHP, the `array()` function is used to create an array:

```
array();
```

Create an Array in PHP

In PHP, the `array()` function is used to create an array:

```
array();
```

PHP Indexed Arrays

These arrays can store numbers, strings and any object but their index will be prepresented by numbers. By default array index starts from zero.

Example

Following is the example showing how to create and access numeric arrays.

Here we have used **array()** function to create array. This function is explained in function reference.

```
<html>
<body>
<?php
/* First method to create array. */
$numbers = array( 1, 2, 3, 4, 5);
foreach( $numbers as $value )
{
    echo "Value is $value <br />";
}
/* Second method to create array. */
$numbers[0] = "one";
$numbers[1] = "two";
$numbers[2] = "three";
$numbers[3] = "four";
$numbers[4] = "five";
foreach( $numbers as $value )
{
    echo "Value is $value <br />";
}
?>
</body>
</html>
```

This will produce following result:

```
Value is 1
Value is 2
Value is 3
Value is 4
Value is 5
Value is one
Value is two
Value is three
Value is four
```


Value is five

Get The Length of an Array – The count() Function

The count() function is used to return the length (the number of elements) of an array: **Example:**

```
<?php
$cars=array("Volvo","BMW","Toyota");
echo count($cars);
?>
```

Output Result:

3

Loop Through an Indexed Array

To loop through and print all the values of an indexed array, you could use a for loop, like this:**Example:**

```
<?php
$cars=array("Volvo","BMW","Toyota");
$arrlength=count($cars);
for($x=0;$x<$arrlength;$x++)
{
    echo $cars[$x];
    echo "<br>";
}
?>
```

Output Result:

Volvo
BMW
Toyota

Associative Arrays

The associative arrays are very similar to numeric arrays in term of functionality but they are different in terms of their index. Associative array will have their index as string so that you can establish a strong association between key and values.

To store the salaries of employees in an array, a numerically indexed array would not be the best choice. Instead, we could use the employees names as the keys in our associative array, and the value would be their respective salary.

NOTE: Don't keep associative array inside double quote while printing otherwise it would not return any value.

Example

```
<html>
<body>
<?php
/* First method to associate create array. */
$salaries = array(
    "mohammad" => 2000,
    "qadir" => 1000,
    "zara" => 500
);
echo "Salary of mohammad is ". $salaries['mohammad'] . "<br />";
echo "Salary of qadir is ". $salaries['qadir']. "<br />";
echo "Salary of zara is ". $salaries['zara']. "<br />";

/* Second method to create array. */
$salaries['mohammad'] = "high";
$salaries['qadir'] = "medium";
$salaries['zara'] = "low";
echo "Salary of mohammad is ". $salaries['mohammad'] . "<br />";
echo "Salary of qadir is ". $salaries['qadir']. "<br />";
echo "Salary of zara is ". $salaries['zara']. "<br />";
?>
</body>
</html>
```

This will produce following result:

```
Salary of mohammad is 2000
Salary of qadir is 1000
Salary of zara is 500
Salary of mohammad is high
Salary of qadir is medium
Salary of zara is low
```

Loop Through an Associative Array

To loop through and print all the values of an associative array, you could use a foreach loop, like this: **Example:**

```
<?php
$age=array("Peter"=>"35","Ben"=>"37","Joe"=>"43");
foreach($age as $x=>$x_value)
{
    echo "Key=" . $x . ", Value=" . $x_value;
    echo "<br>";
}
?>
```

Output Result:

```
Key=Peter, Value=35
Key=Ben, Value=37
Key=Joe, Value=43
```

Multidimensional Arrays

A multi-dimensional array each element in the main array can also be an array. And each element in the sub-array can be an array, and so on. Values in the multi-dimensional array are accessed using multiple index.

Example

In this example we create a two dimensional array to store marks of three students in three subjects:

This example is an associative array, you can create numeric array in the same fashion.

```
<html>
<body>
<?php
    $marks = array(
        "mohammad" => array
        (
            "physics" => 35,
            "maths" => 30,
            "chemistry" => 39
        ),
        "qadir" => array
        (
            "physics" => 30,
```

```

        "maths" => 32,
        "chemistry" => 29
    ),
    "zara" => array
    (
        "physics" => 31,
        "maths" => 22,
        "chemistry" => 39
    )
);
/* Accessing multi-dimensional array values */
echo "Marks for mohammad in physics : " ;
echo $marks['mohammad']['physics'] . "<br />";
echo "Marks for qadir in maths : ";
echo $marks['qadir']['maths'] . "<br />";
echo "Marks for zara in chemistry : " ;
echo $marks['zara']['chemistry'] . "<br />";
?>
</body>
</html>

```

This will produce following result:

Marks for mohammad in physics : 35

Marks for qadir in maths : 32

Marks for zara in chemistry : 39

PHP Sorting Arrays

 May 9th, 2013  admin

The elements in an array can be sorted in alphabetical or numerical order, descending or ascending.

PHP – Sort Functions For Arrays

In this chapter, we will go through the following PHP array sort functions:

- `sort()` – sort arrays in ascending order
- `rsort()` – sort arrays in descending order
- `asort()` – sort associative arrays in ascending order, according to the value
- `ksort()` – sort associative arrays in ascending order, according to the key
- `arsort()` – sort associative arrays in descending order, according to the value
- `krsort()` – sort associative arrays in descending order, according to the key

Sort Array in Ascending Order – sort()

The following example sorts the elements of the \$cars array in ascending alphabetical order:**Example:**

```
<?php
$cars=array("Volvo","BMW","Toyota");
sort($cars);
?>
```

Output Result:

```
BMW
Toyota
Volvo
```

The following example sorts the elements of the \$numbers array in ascending numerical order:**Example:**

```
<?php
$numbers=array(4,6,2,22,11);
sort($numbers);
?>
```

Output Result:

```
2
4
6
11
22
```

Sort Array in Descending Order – rsort()

The following example sorts the elements of the \$cars array in descending alphabetical order:**Example:**

```
<?php
$cars=array("Volvo","BMW","Toyota");
rsort($cars);
?>
```

Output Result:

```
Volvo
Toyota
BMW
```

The following example sorts the elements of the \$numbers array in descending numerical order: **Example:**

```
<?php
$numbers=array(4,6,2,22,11);
rsort($numbers);
?>
```

Output Result:

```
22
11
6
4
2
```

Sort Array in Ascending Order, According to Value – asort()

The following example sorts an associative array in ascending order, according to the value: **Example:**

```
<?php
$age=array("Peter"=>"35","Ben"=>"37","Joe"=>"43");
asort($age);
?>
```

Output Result:

```
Key=Peter, Value=35
Key=Ben, Value=37
Key=Joe, Value=43
```

Sort Array in Ascending Order, According to Key – ksort()

The following example sorts an associative array in ascending order, according to the key: **Example:**

```
<?php
$age=array("Peter"=>"35","Ben"=>"37","Joe"=>"43");
ksort($age);
?>
```

Output Result:

```
Key=Ben, Value=37  
Key=Joe, Value=43  
Key=Peter, Value=35
```

Sort Array in Descending Order, According to Value – arsort()

The following example sorts an associative array in descending order, according to the value:**Example:**

```
<?php  
$age=array("Peter"=>"35","Ben"=>"37","Joe"=>"43");  
arsort($age);  
?>
```

Output Result:

```
Key=Joe, Value=43  
Key=Ben, Value=37  
Key=Peter, Value=35
```

Sort Array in Descending Order, According to Key – krsort()

The following example sorts an associative array in descending order, according to the key:**Example:**

```
<?php  
$age=array("Peter"=>"35","Ben"=>"37","Joe"=>"43");  
krsort($age);  
?>
```

Output Result:

```
Key=Peter, Value=35  
Key=Joe, Value=43  
Key=Ben, Value=37
```

PHP While Loops

 May 10th, 2013  admin

Loops execute a block of code a specified number of times, or while a specified condition is true.

PHP Loops

Often when you write code, you want the same block of code to run over and over again in a row. Instead of adding several almost equal lines in a script we can use loops to perform a task like this.

In PHP, we have the following looping statements:

- **while** - loops through a block of code while a specified condition is true
- **do...while** – loops through a block of code once, and then repeats the loop as long as a specified condition is true
- **for** - loops through a block of code a specified number of times
- **foreach** - loops through a block of code for each element in an array

The while Loop

The while loop executes a block of code while a condition is true. If the test expression is true then the code block will be executed. After the code has executed the test expression will again be evaluated and the loop will continue until the test expression is found to be false.

Syntax

```
while (condition)  
{  
    code to be executed;  
}
```


Example

The example below first sets a variable *i* to 1 (`$i=1;`).

Then, the while loop will continue to run as long as *i* is less than, or equal to 5. *i* will increase by 1 each time the loop runs:

```
<html>
<body>
<?php
$i=1;
while($i<=5)
{
    echo "The number is " . $i . "<br>";
    $i++;
}
?>
</body>
</html>
```

Output:

```
The number is 1
The number is 2
The number is 3
The number is 4
The number is 5
```

The do...while Statement

The do...while statement will always execute the block of code once, it will then check the condition, and repeat the loop while the condition is true.

Syntax

```
do
{
    code to be executed;
}
while (condition);
```

Example

The example below first sets a variable *i* to 1 (*\$i=1;*).

Then, it starts the do...while loop. The loop will increment the variable *i* with 1, and then write some output. Then the condition is checked (is *i* less than, or equal to 5), and the loop will continue to run as long as *i* is less than, or equal to 5:


```
<html>
<body>
<?php
$i=1;
do
{
    $i++;
    echo "The number is " . $i . "<br>";
}
while ($i<=5);
?>
</body>
</html>
```

Output:

```
The number is 2
The number is 3
```

```
The number is 4
The number is 5
The number is 6
```

PHP for Loop

 May 10th, 2013  admin

Loops in PHP are used to execute the same block of code a specified number of times.

for Loop

The for loop is used when you know in advance how many times the script should run.

Syntax

```
for (init; condition; increment)
{
    code to be executed;
}
```

Parameters:

- *init*: Mostly used to set a counter (but can be any code to be executed once at the beginning of the loop)
- *condition*: Evaluated for each loop iteration. If it evaluates to TRUE, the loop continues. If it evaluates to FALSE, the loop ends.
- *increment*: Mostly used to increment a counter (but can be any code to be executed at the end of the iteration)

Note: The *init* and *increment* parameters above can be empty or have multiple expressions (separated by commas).

Example

The example below defines a loop that starts with *i*=1. The loop will continue to run as long as the variable *i* is less than, or equal to 5. The variable *i* will increase by 1 each time the loop runs:

```
<html>
<body>

<?php
for ($i=1; $i<=5; $i++)
{
    echo "The number is " . $i . "<br>";
}
?>

</body>
</html>
```

Output:

```
The number is 1
The number is 2
The number is 3
The number is 4
The number is 5
```

The foreach Loop

The foreach loop is used to loop through arrays.

Syntax

```
foreach ($array as $value)
{
    code to be executed;
}
```

For every loop iteration, the value of the current array element is assigned to \$value (and the array pointer is moved by one) – so on the next loop iteration, you'll be looking at the next array value.

Example


The following example demonstrates a loop that will print the values of the given array:

```
<html>
<body>
<?php
$x=array("one","two","three");
foreach ($x as $value)
{
    echo $value . "<br>";
}
?>
</body>
</html>
```

Output:

```
one
two
three
```

PHP Functions

 May 13th, 2013  admin

The real power of PHP comes from its functions. In PHP, there are more than 700 built-in functions.

PHP functions are similar to other programming languages. A function is a piece of code which takes one more input in the form of parameter and does some processing and returns a value.

You already have seen many functions like **fopen()** and **fread()** etc. They are built-in functions but PHP gives you option to create your own functions as well.

There are two parts which should be clear to you:

- Creating a PHP Function
- Calling a PHP Function

In fact you hardly need to create your own PHP function because there are already more than 1000 of built-in library functions created for different area and you just need to call them according to your requirement.

Please refer to [PHP Function Reference](#) for a complete set of useful functions.

Create a PHP Function

A function will be executed by a call to the function.

Syntax

```
function functionName()  
{  
    code to be executed;  
}
```

PHP function guidelines:

- Give the function a name that reflects what the function does
- The function name can start with a letter or underscore (not a number)

Example

A simple function that writes my name when it is called:

```
<html>  
<body>  
<?php  
function writeName()  
{  
    echo "Kai Jim Refsnes";  
}  
  
echo "My name is ";  
writeName();  
?>  
</body>
```

```
</html>
```

Output:

```
My name is Kai Jim Refsnes
```

PHP Functions – Adding parameters

To add more functionality to a function, we can add parameters. A parameter is just like a variable.

Parameters are specified after the function name, inside the parentheses.

Example 1

The following example will write different first names, but equal last name:

```
<html>
<body>
<?php
function writeName($fname)
{
    echo $fname . " Refsnes.<br>";
}

echo "My name is ";
writeName("Kai Jim");
echo "My sister's name is ";
writeName("Hege");
echo "My brother's name is ";
writeName("Stale");
?>

</body>
</html>
```

Output:

```
My name is Kai Jim Refsnes.
My sister's name is Hege Refsnes.
My brother's name is Stale Refsnes.
```

Example 2

```
<html>

<head>

<title>Writing PHP Function with Parameters</title>

</head>

<body>

<?php

function addFunction($num1, $num2)

{

    $sum = $num1 + $num2;

    echo "Sum of the two numbers is : $sum";

}

addFunction(10, 20);

?>

</body>

</html>
```

This will display following result:

```
Sum of the two numbers is : 30
```

PHP Functions – Return values

To let a function return a value, use the return statement.

Example

```
<html>
<body>
<?php
function add($x,$y)
{
$total=$x+$y;
return $total;
}

echo "1 + 16 = " . add(1,16);
?>

</body>
</html>
```

Output:

```
1 + 16 = 17
```

Passing Arguments by Reference:

It is possible to pass arguments to functions by reference. This means that a reference to the variable is manipulated by the function rather than a copy of the variable's value.

Any changes made to an argument in these cases will change the value of the original variable. You can pass an argument by reference by adding an ampersand to the variable name in either the function call or the function definition.

Following example depicts both the cases.

```
<html>

<head>

<title>Passing Argument by Reference</title>

</head>

<body>

<?php
```



```
function addFive($num)

{

    $num += 5;

}


function addSix(&$num)

{

    $num += 6;

}

$orignum = 10;

addFive( &$orignum );

echo "Original Value is $orignum<br />";

addSix( $orignum );

echo "Original Value is $orignum<br />";

?>

</body>

</html>
```

This will display following result:

Original Value is 15

Original Value is 21

Setting Default Values for Function Parameters:

You can set a parameter to have a default value if the function's caller doesn't pass it.

Following function prints NULL in case use does not pass any value to this function.

```
<html>

<head>

<title>Writing PHP Function which returns value</title>

</head>

<body>

<?php

function printMe($param = NULL)

{

    print $param;

}

printMe("This is test");

printMe();

?>

</body>

</html>
```

This will produce following result:

This is test

Dynamic Function Calls:

It is possible to assign function names as strings to variables and then treat these variables exactly as you would the function name itself. Following example depicts this behaviour.

```
<html>

<head>

<title>Dynamic Function Calls</title>

</head>

<body>

<?php

function sayHello()

{

    echo "Hello<br />";

}

$function_holder = "sayHello";

$function_holder();

?>



</body>

</html>
```

This will display following result:

Hello

PHP Forms Tutorial

 May 15th, 2013  admin

The PHP \$_GET and \$_POST variables are used to retrieve information from forms, like user input.

PHP Form Handling

The most important thing to notice when dealing with HTML forms and PHP is that any form element in an HTML page will **automatically** be available to your PHP scripts.

Example

The example below contains an HTML form with two input fields and a submit button:

```
<html>
<body>

<form action="welcome.php" method="post">
Name: <input type="text" name="fname">
Age: <input type="text" name="age">
<input type="submit">
</form>

</body>
</html>
```

When a user fills out the form above and clicks on the submit button, the form data is sent to a PHP file, called "welcome.php":

"welcome.php" looks like this:

```
<html>
<body>

Welcome <?php echo $_POST["fname"]; ?>!<br>
You are <?php echo $_POST["age"]; ?> years old.

</body>
</html>
```

Output could be something like this:

```
Welcome John!  
You are 28 years old.
```

When to use method="get"?

When using method="get" in HTML forms, all variable names and values are displayed in the URL.

Note: This method should not be used when sending passwords or other sensitive information!

However, because the variables are displayed in the URL, it is possible to bookmark the page. This can be useful in some cases.

Note: The get method is not suitable for very large variable values. It should not be used with values exceeding 2000 characters.

When to use method="post"?

Information sent from a form with the POST method is invisible to others and has no limits on the amount of information to send.

However, because the variables are not displayed in the URL, it is not possible to bookmark the page.

The PHP \$_REQUEST Variable

The predefined \$_REQUEST variable contains the contents of both \$_GET, \$_POST, and \$_COOKIE.

The PHP **\$_REQUEST variable** can be used to get the result from form data sent with both the GET and POST methods.

Example

```
Welcome <?php echo $_REQUEST["fname"]; ?>!  
You are <?php echo $_REQUEST["age"]; ?> years old.
```

PHP Date() and Time() Function

 May 16th, 2013  admin

Dates are so much part of everyday life that it becomes easy to work with them without thinking. PHP also provides powerful tools for date arithmetic that make manipulating dates easy.

Getting the Time Stamp with time():

PHP's **time()** function gives you all the information that you need about the current date and time. It requires no arguments but returns an integer.

The integer returned by **time()** represents the number of seconds elapsed since midnight GMT on January 1, 1970. This moment is known as the UNIX epoch, and the number of seconds that have elapsed since then is referred to as a time stamp.

```
<?php  
  
print time();  
  
?>
```

It will produce following result:

```
948316201
```

This is something difficult to understand. But PHP offers excellent tools to convert a time stamp into a form that humans are comfortable with.

Converting a Time Stamp with getdate():

The function **getdate()** optionally accepts a time stamp and returns an associative array containing information about the date. If you omit the time stamp, it works with the current time stamp as returned by **time()**.

Following table lists the elements contained in the array returned by **getdate()**.

Key	Description	Example
seconds	Seconds past the minutes (0-59)	20
minutes	Minutes past the hour (0 – 59)	29
hours	Hours of the day (0 – 23)	22
mday	Day of the month (1 – 31)	11

Key	Description	Example
wday	Day of the week (0 – 6)	4
mon	Month of the year (1 – 12)	7
year	Year (4 digits)	1997
yday	Day of year (0 – 365)	19
weekday	Day of the week	Thursday
month	Month of the year	January
0	Timestamp	948370048

Now you have complete control over date and time. You can format this date and time in whatever format you wan.

Example:

Try out following example

```
<?php

$date_array = getdate();

foreach ( $date_array as $key => $val )

{

    print "$key = $val<br />";

}

$formated_date = "Today's date: ";

$formated_date .= $date_array[mday] . "/";

$formated_date .= $date_array[mon] . "/";

$formated_date .= $date_array[year];
```

```
print $formatted_date;
```

```
?>
```

It will produce following result:

```
seconds = 27
```

```
minutes = 25
```

```
hours = 11
```

```
mday = 12
```

```
wday = 6
```

```
mon = 5
```

```
year = 2007
```

```
yday = 131
```

```
weekday = Saturday
```

```
month = May
```

```
0 = 1178994327
```

```
Today's date: 12/5/2007
```

The PHP `date()` function is used to format a time and/or date.

The PHP `date()` Function

The PHP `date()` function formats a timestamp to a more readable date and time.

A timestamp is a sequence of characters, denoting the date and/or time at which a certain event occurred.

Syntax


```
date(format,timestamp)
```

Parameter	Description
format:	Required. Specifies the format of the timestamp
timestamp:	Optional. Specifies a timestamp. Default is the current date and time

The date() optionally accepts a time stamp if omitted then current date and time will be used. Any other data you include in the format string passed to date() will be included in the return value.

Following table lists the codes that a format string can contain:

Format	Description	Example
a	'am' or 'pm' lowercase	pm
A	'AM' or 'PM' uppercase	PM
d	Day of month, a number with leading zeroes	20
D	Day of week (three letters)	Thu
F	Month name	January
h	Hour (12-hour format – leading zeroes)	12
H	Hour (24-hour format – leading zeroes)	22
g	Hour (12-hour format – no leading zeroes)	12
G	Hour (24-hour format – no leading zeroes)	22
i	Minutes (0 – 59)	23
j	Day of the month (no leading zeroes)	20
l (Lower 'L')	Day of the week	Thursday
L	Leap year ('1' for yes, '0' for no)	1

m	Month of year (number – leading zeroes)	1
M	Month of year (three letters)	Jan
r	The RFC 2822 formatted date	Thu, 21 Dec 2000 16:01:07 +0200
n	Month of year (number – no leading zeroes)	2
s	Seconds of hour	20
U	Time stamp	948372444
y	Year (two digits)	06
Y	Year (four digits)	2006
z	Day of year (0 – 365)	206
Z	Offset in seconds from GMT	+5

Try out following **example**:

```
<?php
print date("m/d/y G.i:s<br>", time());

print "Today is ";

print date("j of F Y, \a\\t g.i a", time());

?>
```

It will produce following result:

```
01/20/00 13.27:55

Today is 20 of January 2000, at 1.27 pm
```

PHP Date() – Adding a Timestamp

The optional *timestamp* parameter in the `date()` function specifies a timestamp. If you do not specify a timestamp, the current date and time will be used.

The `mktime()` function returns the Unix timestamp for a date.

The Unix timestamp contains the number of seconds between the Unix Epoch (January 1 1970 00:00:00 GMT) and the time specified.

Syntax for `mktime()`

```
mktime(hour,minute,second,month,day,year,is_dst)
```

To go one day in the future we simply add one to the day argument of `mktime()`:

```
<?php
$tomorrow =
mktime(0,0,0,date("m"),date("d")+1,date("Y"));
echo "Tomorrow is ".date("Y/m/d", $tomorrow);
?>
```

The output of the code above could be something like this:

```
Tomorrow is 2009/05/12
```

PHP Include Files

 May 18th, 2013  admin

You can include the content of a PHP file into another PHP file before the server executes it. The `include` and `require` statements are used to insert useful codes written in other files, in the flow of execution. There are two PHP functions which can be used to include one **PHP file** into another PHP file.

- The `include()` Function
- The `require()` Function

This is a strong point of PHP which helps in creating functions, headers, footers, or elements that can be reused on multiple pages. This will help developers to make it easy to change the layout of complete website with minimal effort. If there is any change required then instead of changing thousand of files just change included file.

Include and require are identical, except upon failure:

- `require` will produce a fatal error (`E_COMPILE_ERROR`) and stop the script
- `include` will only produce a warning (`E_WARNING`) and the script will continue

So, if you want the execution to go on and show users the output, even if the include file is missing, use include. Otherwise, in case of Framework, CMS or a complex PHP application coding, always use require to include a key file to the flow of execution. This will help avoid compromising your application's security and integrity, just in-case one key file is accidentally missing.

The include() Function

Syntax:

```
include 'filename';
```

Example:

Assume that you have a standard header file, called "header.php". To include the header file in a page, use include/require:

```
<html>
<body>
<?php include 'header.php'; ?>
<h1>Welcome to my home page!</h1>
<p>Some text.</p>
</body>
</html>
```

example 2:

Assume we have an include file with some variables defined ("vars.php"):

```
<?php
$color='red';
$car='BMW';
?>
```

Then the variables can be used in the calling file:

```
<html>
<body>
<h1>Welcome to my home page.</h1>
<?php include 'vars.php';
echo "I have a $color $car"; // I have a red BMW
?>
</body>
</html>
```

example 3:

Assume you want to create a common menu for your website. Then create a file menu.php with the following content.

```
<a href="http://www.mytoptutorials.com/">Home</a> -  
<a  
href="http://www.mytoptutorials.com/html/">HTML</a>  
-  
<a  
href="http://www.mytoptutorials.com/php/">PHP</a> -  
<br />
```

Now create as many pages as you like and include this file to create header. For example now your test.php file can have following content.

```
<html>  
<body>  
<?php include("menu.php"); ?>  
<p>This is an example to show how to include PHP  
file! </p>  
</body>  
</html>
```

This will produce following result:

```
Home – HTML – PHP  
This is an example to show how to include PHP file. You  
can include menu.php file in as many as files you like!
```

The require() Function

The require() function takes all the text in a specified file and copies it into the file that uses the include function. If there is any problem in loading a file then the **require()** function generates a fatal error and halt the execution of the script.

So there is no difference in require() and include() except they handle error conditions. It is recommended to use the require() function instead of include(), because scripts should not continue executing if files are missing or misnamed.

Syntax:

```
require 'filename';
```

You can try using above example with require() function and it will generate same result. But if you will try following two examples where file does not exist then you will get different results.

```
<html>  
<body>
```

```
<?php include("xxmenu.php"); ?>
<p>This is an example to show how to include wrong
PHP file!</p>
</body>
</html>
```

This will produce following result

```
This is an example to show how to include wrong PHP
file!
```

Now lets try same example with require() function.

```
<html>
<body>
<?php require("xxmenu.php"); ?>
<p>This is an example to show how to include wrong
PHP file!</p>
</body>
</html>
```

PHP File Handling



May 18th, 2013



admin

This chapter will explain following functions related to files:

- Opening a file
- Reading a file
- Writing a file
- Closing a file

Opening a File

The PHP **fopen()** function is used to open a file. It requires two arguments stating first the file name and then mode in which to operate.

The first parameter of this function contains the name of the file to be opened and the second parameter specifies in which mode the file should be opened:

```
<html>
<body>
<?php
$file=fopen("welcome.txt","r");
?>
</body>
</html>
```

Files modes can be specified as one of the six options in this table.

Mode	Description
r	Read only. Opens the file for reading only. Places the file pointer at the beginning of the file.

Mode	Description
r+	Read/Write.Opens the file for reading and writing. Places the file pointer at the beginning of the file.
w	Write only.Opens the file for writing only. Places the file pointer at the beginning of the file. and truncates the file to zero length. If files does not exist then it attempts to create a file.
w+	Read/Write.Opens the file for reading and writing only. Places the file pointer at the beginning of the file. and truncates the file to zero length. If files does not exist then it attempts to create a file.
a	Append.Opens the file for writing only. Places the file pointer at the end of the file. If files does not exist then it attempts to create a file.
a+	Read/Append.Opens the file for reading and writing only. Places the file pointer at the end of the file. If files does not exist then it attempts to create a file.
x	Write only.Creates a new file. Returns FALSE and an error if file already exists
x+	Read/Write.Creates a new file. Returns FALSE and an error if file already exists

If an attempt to open a file fails then **fopen** returns a value of **false** otherwise it returns a **file pointer** which is used for further reading or writing to that file.

After making a changes to the opened file it is important to close it with the **fclose()** function. The **fclose()** function requires a file pointer as its argument and then returns **true** when the closure succeeds or **false** if it fails.

Note: If the **fopen()** function is unable to open the specified file, it returns 0 (false).

Example

The following example generates a message if the **fopen()** function is unable to open the specified file:

```
<html>
<body>

<?php
$file=fopen("welcome.txt","r") or exit("Unable to open file!");
?>

</body>
</html>
```

Closing a File

The **fclose()** function is used to close an open file:

```
<?php
$file = fopen("test.txt","r");

//some code to be executed

fclose($file);
?>
```

Check End of file

The `feof()` function checks if the "end-of-file" (EOF) has been reached.

The `feof()` function is useful for looping through data of unknown length.

Note: You cannot read from files opened in `w`, `a`, and `x` mode!

```
if (feof($file)) echo "End of file";
```

Reading a file

Once a file is opened using `fopen()` function it can be read with a function called `fread()`. This function requires two arguments. These must be the file pointer and the length of the file expressed in bytes.

The file's length can be found using the `filesize()` function which takes the file name as its argument and returns the size of the file expressed in bytes.

So here are the steps required to read a file with PHP.

- Open a file using `fopen()` function.
- Get the file's length using `filesize()` function.
- Read the file's content using `fread()` function.
- Close the file with `fclose()` function.

The following example assigns the content of a text file to a variable then displays those contents on the web page.

```
<html>

<head>

<title>Reading a file using PHP</title>

</head>

<body>


<?php

$filename = "/home/user/guest/tmp.txt";

$file = fopen( $filename, "r" );

if( $file == false )

{
```



```
echo ( "Error in opening file" );

exit();

}

$filesize = filesize( $filename );

$filetext = fread( $file, $filesize );

fclose( $file );

echo ( "File size : $filesize bytes" );

echo ( "<pre>$filetext</pre>" );

?>

</body>

</html>
```

Reading a File Line by Line

The `fgets()` function is used to read a single line from a file.

Note: After a call to this function the file pointer has moved to the next line.

Example

The example below reads a file line by line, until the end of file is reached:

```
<?php
$file = fopen("welcome.txt", "r") or exit("Unable to open file!");
//Output a line of the file until the end is reached
while(!feof($file))
{
    echo fgets($file). "<br>";
}
fclose($file);
?>
```

Reading a File Character by Character

The `fgetc()` function is used to read a single character from a file.

Note: After a call to this function the file pointer moves to the next character.

Example

The example below reads a file character by character, until the end of file is reached:

```
<?php
$file=fopen("welcome.txt","r") or exit("Unable to open file!");
while (!feof($file))
{
    echo fgetc($file);
}
fclose($file);
?>
```

Writing a file

A new file can be written or text can be appended to an existing file using the PHP **`fwrite()`** function. This function requires two arguments specifying a **file pointer** and the string of data that is to be written. Optionally a third integer argument can be included to specify the length of the data to write. If the third argument is included, writing would stop after the specified length has been reached.

The following example creates a new text file then writes a short text heading inside it. After closing this file its existence is confirmed using **`file_exists()`** function which takes file name as an argument

```
<?php

$filename = "/home/user/guest/newfile.txt";

$file = fopen( $filename, "w" );

if( $file == false )

{

    echo ( "Error in opening new file" );

    exit();

}

fwrite( $file, "This is a simple test\n" );

fclose( $file );
```

```
?>

<html>

<head>

<title>Writing a file using PHP</title>

</head>

<body>

<?php

if( file_exist( $filename ) )

{

    $filesize = filesize( $filename );

    $msg = "File created with name $filename ";

    $msg .= "containing $filesize bytes";

    echo ($msg );

}

else

{

    echo ("File $filename does not exist" );


}

?>

</body>

</html>
```

PHP File Upload Script

 May 19th, 2013  admin

With PHP, it is possible to upload files to the server. A PHP script can be used with a HTML form to allow users to upload files to the server. Initially files are uploaded into a temporary directory and then relocated to a target destination by a PHP script.

Information in the [phpinfo.php](#) page describes the temporary directory that is used for file uploads as `upload_tmp_dir` and the maximum permitted size of files that can be uploaded is stated as `upload_max_filesize`. These parameters are set into PHP configuration file `php.ini`

The process of uploading a file follows these steps

- The user opens the page containing a HTML form featuring a text files, a browse button and a submit button.
- The user clicks the browse button and selects a file to upload from the local PC.
- The full path to the selected file appears in the text filed then the user clicks the submit button.
- The selected file is sent to the temporary directory on the server.
- The PHP script that was specified as the form handler in the form's action attribute checks that the file has arrived and then copies the file into an intended directory.
- The PHP script confirms the success to the user.

As usual when writing files it is necessary for both temporary and final locations to have permissions set that enable file writing. If either is set to be read-only then process will fail.

An uploaded file could be a text file or image file or any document.

Create an Upload-File Form

To allow users to upload files from a form can be very useful.

Look at the following HTML form for uploading files:

```
<html>
<body>

<form action="upload_file.php" method="post"
enctype="multipart/form-data">
<label for="file">Filename:</label>
<input type="file" name="file" id="file"> <br>
<input type="submit" name="submit" value="Submit">
</form>

</body>
</html>
```

This will display following result:

Filename:

NOTE: This is just dummy form and would not work.

Notice the following about the HTML form above:

- The `enctype` attribute of the `<form>` tag specifies which content-type to use when submitting the form. "multipart/form-data" is used when a form requires binary data, like the contents of a file, to be uploaded
- The `type="file"` attribute of the `<input>` tag specifies that the input should be processed as a file. For example, when viewed in a browser, there will be a browse-button next to the input field

Note: Allowing users to upload files is a big security risk. Only permit trusted users to perform file uploads.

Create an upload script

The "upload_file.php" file contains the code for uploading a file:

```
<?php
if ($_FILES["file"]["error"] > 0)
{
    echo "Error: " . $_FILES["file"]["error"] . "<br>";
}
else
{
    echo "Upload: " . $_FILES["file"]["name"] . "<br>";
    echo "Type: " . $_FILES["file"]["type"] . "<br>";
    echo "Size: " . ($_FILES["file"]["size"] / 1024) . " kB<br>";
    echo "Stored in: " . $_FILES["file"]["tmp_name"];
}
?>
```

By using the global PHP \$_FILES array you can upload files from a client computer to the remote server.

The first parameter is the form's input name and the second index can be either "name", "type", "size", "tmp_name" or "error". Like this:

- \$_FILES["file"]["name"] – the name of the uploaded file
- \$_FILES["file"]["type"] – the type of the uploaded file
- \$_FILES["file"]["size"] – the size in kilobytes of the uploaded file
- \$_FILES["file"]["tmp_name"] – the name of the temporary copy of the file stored on the server
- \$_FILES["file"]["error"] – the error code resulting from the file upload

This is a very simple way of uploading files. For security reasons, you should add restrictions on what the user is allowed to upload.

Upload Restrictions

In this script we add some restrictions to the file upload. The user may upload .gif, .jpeg, and .png files; and the file size must be under 20 kB:

```
<?php
$allowedExts = array("gif", "jpeg", "jpg", "png");
$extension = end(explode(".", $_FILES["file"]["name"]));
if ((($_FILES["file"]["type"] == "image/gif")
|| ($_FILES["file"]["type"] == "image/jpeg")
|| ($_FILES["file"]["type"] == "image/jpg")
|| ($_FILES["file"]["type"] == "image/pjpeg")
|| ($_FILES["file"]["type"] == "image/x-png")
|| ($_FILES["file"]["type"] == "image/png"))
&& ($_FILES["file"]["size"] < 20000)
&& in_array($extension, $allowedExts))
{
    if ($_FILES["file"]["error"] > 0)
    {
        echo "Error: " . $_FILES["file"]["error"] . "<br>";
    }
    else
    {
        echo "Upload: " . $_FILES["file"]["name"] . "<br>";
    }
}
```

```

    echo "Type: " . $_FILES["file"]["type"] . "<br>";
    echo "Size: " . ($_FILES["file"]["size"] / 1024) . " kB<br>";
    echo "Stored in: " . $_FILES["file"]["tmp_name"];
  }
}
else
{
    echo "Invalid file";
}
?>

```

Store the Uploaded File

The examples above create a temporary copy of the uploaded files in the PHP temp folder on the server.

The temporary copied files disappears when the script ends. To Saving the uploaded file we need to copy it to a different location:

```

<?php
$allowedExts = array("gif", "jpeg", "jpg", "png");
$extension = end(explode(".", $_FILES["file"]["name"]));
if ((($_FILES["file"]["type"] == "image/gif")
|| ($_FILES["file"]["type"] == "image/jpeg")
|| ($_FILES["file"]["type"] == "image/jpg")
|| ($_FILES["file"]["type"] == "image/pjpeg")
|| ($_FILES["file"]["type"] == "image/x-png")
|| ($_FILES["file"]["type"] == "image/png")))
&& ($_FILES["file"]["size"] < 20000)
&& in_array($extension, $allowedExts))
{
    if ($_FILES["file"]["error"] > 0)
    {
        echo "Return Code: " . $_FILES["file"]["error"] . "<br>";
    }
}
else
{
    echo "Upload: " . $_FILES["file"]["name"] . "<br>";
    echo "Type: " . $_FILES["file"]["type"] . "<br>";
    echo "Size: " . ($_FILES["file"]["size"] / 1024) . " kB<br>";
    echo "Temp file: " . $_FILES["file"]["tmp_name"] . "<br>";

    if (file_exists("upload/" . $_FILES["file"]["name"]))
    {
        echo $_FILES["file"]["name"] . " already exists. ";
    }
    else
    {
        move_uploaded_file($_FILES["file"]["tmp_name"],
        "upload/" . $_FILES["file"]["name"]);
        echo "Stored in: " . "upload/" . $_FILES["file"]["name"];
    }
}
}
else
{
    echo "Invalid file";
}

```

```
?>
```

PHP Cookies

 May 21st, 2013  admin

A cookie is often used to identify a user. Cookies are text files stored on the client computer and they are kept of use tracking purpose. Each time the same computer requests a page with a browser, it will send the cookie too. PHP transparently supports HTTP cookies.

There are three steps involved in identifying returning users:

- Server script sends a set of cookies to the browser. For example name, age, or identification number etc.
- Browser stores this information on local machine for future use.
- When next time browser sends any request to web server then it sends those cookies information to the server and server uses that information to identify the user.

This chapter will teach you how to set cookies, how to access them and how to delete them.

The Anatomy of a PHP Cookie :

Cookies are usually set in an HTTP header (although JavaScript can also set a cookie directly on a browser). A PHP script that sets a cookie might send headers that look something like this:

```
HTTP/1.1 200 OK
Date: Fri, 04 Feb 2000 21:03:38 GMT
Server: Apache/1.3.9 (UNIX) PHP/4.0b3
Set-Cookie: name=xyz; expires=Friday, 04-Feb-07 22:03:38 GMT;
           path=/; domain=tutorialspoint.com
Connection: close
Content-Type: text/html
```

As you can see, the Set-Cookie header contains a name value pair, a GMT date, a path and a domain. The name and value will be URL encoded. The expires field is an instruction to the browser to "forget" the cookie after the given time and date.

If the browser is configured to store cookies, it will then keep this information until the expiry date. If the user points the browser at any page that matches the path and domain of the cookie, it will resend the cookie to the server. The browser's headers might look something like this:

```
GET / HTTP/1.0
Connection: Keep-Alive
User-Agent: Mozilla/4.6 (X11; I; Linux 2.2.6-15apmac ppc)
Host: zink.demon.co.uk:1126
Accept: image/gif, */*
Accept-Encoding: gzip
Accept-Language: en
Accept-Charset: iso-8859-1,*,utf-8
Cookie: name=xyz
```

A PHP script will then have access to the cookie in the environmental variables `$_COOKIE` or `$HTTP_COOKIE_VARS[]` which holds all cookie names and values. Above cookie can be accessed using `$HTTP_COOKIE_VARS["name"]`.

How to Create a Cookie?

The `setcookie()` function is used to set a cookie in PHP .

Note: The `setcookie()` function must appear BEFORE the `<html>` tag. This function requires upto six arguments and should be called before `<html>` tag. For each cookie this function has to be called separately.

Syntax

```
setcookie(name, value, expire, path, domain, security);
```

Here is the detail of all the arguments:

- **Name** - This sets the name of the cookie and is stored in an environment variable called `HTTP_COOKIE_VARS`. This variable is used while accessing cookies.
- **Value** - This sets the value of the named variable and is the content that you actually want to store.
- **Expiry** - This specifies a future time in seconds since 00:00:00 GMT on 1st Jan 1970. After this time cookie will become inaccessible. If this parameter is not set then cookie will automatically expire when the Web Browser is closed.
- **Path** - This specifies the directories for which the cookie is valid. A single forward slash character permits the cookie to be valid for all directories.
- **Domain** - This can be used to specify the domain name in very large domains and must contain at least two periods to be valid. All cookies are only valid for the host and domain which created them.
- **Security** - This can be set to 1 to specify that the cookie should only be sent by secure transmission using HTTPS otherwise set to 0 which means cookie can be sent by regular HTTP.

Following example will create two cookies **name** and **age** these cookies will be expired after one hour.

```
<?php
    setcookie("name", "John Watkin", time()+3600, "/", "", 0);
    setcookie("age", "36", time()+3600, "/", "", 0);
?>
<html>
<head>
<title>Setting Cookies with PHP</title>
</head>
<body>
<?php echo "Set Cookies"?>
</body>
</html>
```

Example 2:

In the example below, we will create a cookie named "user" and assign the value "Alex Porter" to it. We also specify that the cookie should expire after one hour:

```
<?php
setcookie("user", "Alex Porter", time()+3600);
?>

<html>
.....
```

Note: The value of the cookie is automatically URLencoded when sending the cookie, and automatically decoded when received (to prevent URLencoding, use `setrawcookie()` instead).

Example 3:

You can also set the expiration time of the cookie in another way. It may be easier than using seconds.

```
<?php
$expire=time()+60*60*24*30;
```



```
setcookie("user", "Alex Porter", $expire);
?>

<html>

.....
```

In the example above the expiration time is set to a month (*60 sec * 60 min * 24 hours * 30 days*).

Accessing Cookies with PHP

PHP provides many ways to access cookies. Simplest way is to use either `$_COOKIE` or `$HTTP_COOKIE_VARS` variables. Following example will access all the cookies set in above example.

```
<html>
<head>
<title>Accessing Cookies with PHP</title>
</head>
<body>
<?php
echo $_COOKIE["name"]. "<br />";
/* is equivalent to */
echo $HTTP_COOKIE_VARS["name"]. "<br />";

echo $_COOKIE["age"] . "<br />";
/* is equivalent to */
echo $HTTP_COOKIE_VARS["name"] . "<br />";
?>
</body>
</html>
```

You can use **isset()** function to check if a cookie is set or not.

```
<html>
<head>
<title>Accessing Cookies with PHP</title>
</head>
<body>
<?php
if( isset($_COOKIE["name"]))
    echo "Welcome " . $_COOKIE["name"] . "<br />";
else
    echo "Sorry... Not recognized" . "<br />";
?>
</body>
</html>
```

example : In the example below, we retrieve the value of the cookie named "user" and display it on a page:

```
<?php
// Print a cookie
echo $_COOKIE["user"];

// A way to view all cookies
print_r($_COOKIE);
?>
```

In the following example we use the `isset()` function to find out if a cookie has been set:

```
<html>
```

```
<body>
<?php
if (isset($_COOKIE["user"]))
    echo "Welcome " . $_COOKIE["user"] . "!<br>";
else
    echo "Welcome guest!<br>";
?>

</body>
</html>
```

How to Delete Cookie with PHP

Officially, to delete a cookie you should call `setcookie()` with the name argument only but this does not always work well, however, and should not be relied on.

It is safest to set the cookie with a date that has already expired:

```
<?php
setcookie( "name", "", time() - 60, "/", "", 0);
setcookie( "age", "", time() - 60, "/", "", 0);
?>
<html>
<head>
<title>Deleting Cookies with PHP</title>
</head>
<body>
<?php echo "Deleted Cookies" ?>
</body>
</html>
```

Another Delete example:

```
<?php
// set the expiration date to one hour ago
setcookie("user", "", time()-3600);
?>
```

What if a Browser Does NOT Support Cookies?

If your application deals with browsers that do not support cookies, you will have to use other methods to pass information from one page to another in your application. One method is to pass the data through forms (forms and user input are described earlier in this tutorial).

The form below passes the user input to "welcome.php" when the user clicks on the "Submit" button:

```
<html>
<body>

<form action="welcome.php" method="post">
Name: <input type="text" name="name">
Age: <input type="text" name="age">
<input type="submit">
</form>

</body>
```

```
</html>
```

Retrieve the values in the "welcome.php" file like this:

```
<html>
<body>

Welcome <?php echo $_POST["name"]; ?>.<br>
You are <?php echo $_POST["age"]; ?> years old.

</body>
</html>
```

PHP Sessions

 May 21st, 2013  admin

An alternative way to make data accessible across the various pages of an entire website is to use a **PHP Session**. A PHP session variable is used to store information about, or change settings for a user session. Session variables hold information about one single user, and are available to all pages in one application.

A session creates a file in a temporary directory on the server where registered session variables and their values are stored. This data will be available to all pages on the site during that visit.

The location of the temporary file is determined by a setting in the **php.ini** file called **session.save_path**. Before using any session variable make sure you have setup this path.

When a session is started following things happen:

- PHP first creates a unique identifier for that particular session which is a random string of 32 hexadecimal numbers such as 3c7foj34c3jj973hjkop2fc937e3443.
- A cookie called **PHPSESSID** is automatically sent to the user's computer to store unique session identification string.
- A file is automatically created on the server in the designated temporary directory and bears the name of the unique identifier prefixed by sess_ ie sess_3c7foj34c3jj973hjkop2fc937e3443.

When a PHP script wants to retrieve the value from a session variable, PHP automatically gets the unique session identifier string from the PHPSESSID cookie and then looks in its temporary directory for the file bearing that name and a validation can be done by comparing both values.

A session ends when the user loses the browser or after leaving the site, the server will terminate the session after a predetermined period of time, commonly 30 minutes duration.

PHP Session Variables

When you are working with an application, you open it, do some changes and then you close it. This is much like a Session. The computer knows who you are. It knows when you start the application and when you end. But on the internet there is one problem: the web server does not know who you are and what you do because the HTTP address doesn't maintain state.

A PHP session solves this problem by allowing you to store user information on the server for later use (i.e. username, shopping items, etc). However, session information is temporary and will be deleted after the user has left the website. If you need a permanent storage you may want to store the data in a database.

Sessions work by creating a unique id (UID) for each visitor and store variables based on this UID. The UID is either stored in a cookie or is propagated in the URL.

Starting a PHP Session

Before you can store user information in your PHP session, you must first start up the session.

A PHP session is easily started by making a call to the `session_start()` function. This function first checks if a session is already started and if none is started then it starts one. It is recommended to put the call to `session_start()` at the beginning of the page.

Session variables are stored in associative array called `$_SESSION[]`. These variables can be accessed during lifetime of a session.

Note: The `session_start()` function must appear BEFORE the `<html>` tag:

```
<?php session_start(); ?>
<html>
<body>

</body>
</html>
```

The code above will register the user's session with the server, allow you to start saving user information, and assign a UID for that user's session.

The following example starts a session then register a variable called **counter** that is incremented each time the page is visited during the session.

Make use of `isset()` function to check if session variable is already set or not.

Put this code in a `test.php` file and load this file many times to see the result:

```
<?php
session_start();
if( isset( $_SESSION['counter'] ) )
{
    $_SESSION['counter'] += 1;
}
else
{
    $_SESSION['counter'] = 1;
}
$msg = "You have visited this page ". $_SESSION['counter'];
$msg .= "in this session.";
?>
<html>
<head>
<title>Setting up a PHP session</title>
</head>
<body>
<?php echo ( $msg ); ?>
</body>
</html>
```

Storing a Session Variable

The correct way to store and retrieve session variables is to use the PHP `$_SESSION` variable:

```
<?php
session_start();
// store session data
$_SESSION['views']=1;
?>
<html>
```

```
<body>
<?php
//retrieve session data
echo "Pageviews=". $_SESSION['views'];
?>

</body>
</html>
```

Output result:

```
Pageviews=1
```

In the example below, we create a simple page-views counter. The `isset()` function checks if the "views" variable has already been set. If "views" has been set, we can increment our counter. If "views" doesn't exist, we create a "views" variable, and set it to 1:

```
<?php
session_start();

if(isset($_SESSION['views']))
$_SESSION['views']=$_SESSION['views']+1;
else
$_SESSION['views']=1;
echo "Views=". $_SESSION['views'];
?>
```

Destroying a PHP Session:

A PHP session can be destroyed by `session_destroy()` function. This function does not need any argument and a single call can destroy all the session variables. If you want to destroy a single session variable then you can use `unset()` function to unset a session variable.

The `unset()` function is used to free the specified session variable:

```
<?php
session_start();
if(isset($_SESSION['views']))
    unset($_SESSION['views']);
?>
```

You can also completely destroy the session by calling the `session_destroy()` function:

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

Note: `session_destroy()` will reset your session and you will lose all your stored session data.

Turning on Auto Session:

You don't need to call `start_session()` function to start a session when a user visits your site if you can set `session.auto_start` variable to 1 in `php.ini` file.

Sessions without cookies:

There may be a case when a user does not allow to store cookies on their machine. So there is another method to send session ID to the browser.

Alternatively, you can use the constant `SID` which is defined if the session started. If the client did not send an appropriate session cookie, it has the form `session_name=session_id`. Otherwise, it expands to an empty string. Thus, you can embed it unconditionally into URLs.

The following example demonstrates how to register a variable, and how to link correctly to another page using `SID`.

```
<?php
session_start();

if (isset($_SESSION['counter'])) {
    $_SESSION['counter'] = 1;
} else {
    $_SESSION['counter']++;
}
?>

$msg = "You have visited this page ". $_SESSION['counter'];
$msg .= "in this session.";
echo ( $msg );
<p>
To continue click following link <br />
<a href="nextpage.php?<?php echo htmlspecialchars(SID); >">
</p>
```

PHP Sending Emails

 May 22nd, 2013  admin

PHP allows you to send e-mails directly from a script. PHP must be configured correctly in the `php.ini` file with the details of how your system sends email. Open `php.ini` file available in `/etc/` directory and find the section headed **[mail function]**.

Windows users should ensure that two directives are supplied. The first is called `SMTP` that defines your email server address. The second is called `sendmail_from` which defines your own email address.

The configuration for Windows should look something like this:

```
[mail function]
; For Win32 only.
SMTP = smtp.secureserver.net

; For win32 only
sendmail_from = webmaster@mytoptutorials.com
```

Linux users simply need to let PHP know the location of their **sendmail** application. The path and any desired switches should be specified to the `sendmail_path` directive.

The configuration for Linux should look something like this:

```
[mail function]
; For Win32 only.
SMTP =

; For win32 only
sendmail_from =

; For Unix only
sendmail_path = /usr/sbin/sendmail -t -i
```

The PHP mail() Function

The PHP mail() function is used to send emails from inside a script. This function requires three mandatory arguments that specify the recipient's email address, the subject of the message and the actual message additionally there are other two optional parameters.

Syntax

```
mail(to,subject,message,headers,parameters)
```

Parameter	Description
to	Required. Specifies the receiver / receivers of the email
subject	Required. Specifies the subject of the email. Note: This parameter cannot contain any newline characters
message	Required. Defines the message to be sent. Each line should be separated with a LF (\n). Lines should not exceed 70 characters
headers	Optional. Specifies additional headers, like From, Cc, and Bcc. The additional headers should be separated with a CRLF (\r\n)
parameters	Optional. Specifies an additional parameter to the sendmail program

Note: For the mail functions to be available, PHP requires an installed and working email system. The program to be used is defined by the configuration settings in the php.ini file. Read more in our [PHP Mail reference](#).

As soon as the mail function is called PHP will attempt to send the email then it will return true if successful or false if it is failed.

Multiple recipients can be specified as the first argument to the mail() function in a comma separated list.

Example:

Following example will send an HTML email message to xyz@somedomain.com. You can code this program in such a way that it should receive all content from the user and then it should send an email.

```
<html>
<head>
<title>Sending email using PHP</title>
</head>
<body>
<?php
    $to = "xyz@somedomain.com";
    $subject = "This is subject";
    $message = "This is simple text message.";
    $header = "From:abc@somedomain.com \r\n";
    $retval = mail ($to,$subject,$message,$header);
    if( $retval == true )
    {
        echo "Message sent successfully...";
    }
    else
    {
        echo "Message could not be sent...";
    }
}
```

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

PHP Simple E-Mail

The simplest way to send an email with PHP is to send a text email.

In the example below we first declare the variables (\$to, \$subject, \$message, \$from, \$headers), then we use the variables in the mail() function to send an e-mail:

```
<?php
$to = "someone@example.com";
$subject = "Test mail";
$message = "Hello! This is a simple email message.";
$from = "someoneelse@example.com";
$headers = "From:" . $from;
mail($to,$subject,$message,$headers);
echo "Mail Sent.";
?>
```

PHP Mail Form

With PHP, you can create a feedback-form on your website. The example below sends a text message to a specified e-mail address:

```
<html>
<body>

<?php
if (isset($_REQUEST['email']))
//if "email" is filled out, send email
{
    //send email
    $email = $_REQUEST['email'] ;
    $subject = $_REQUEST['subject'] ;
    $message = $_REQUEST['message'] ;
    mail("someone@example.com", $subject,
    $message, "From:" . $email);
    echo "Thank you for using our mail form";
}
else
//if "email" is not filled out, display the form
{
    echo "<form method='post' action='mailform.php'>
    Email: <input name='email' type='text'> <br>
    Subject: <input name='subject' type='text'> <br>
    Message:<br>
    <textarea name='message' rows='15' cols='40'>
    </textarea> <br>
    <input type='submit'>
    </form>";
}
?>
```



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

This is how the example above works:

- First, check if the email input field is filled out
- If it is not set (like when the page is first visited); output the HTML form
- If it is set (after the form is filled out); send the email from the form
- When submit is pressed after the form is filled out, the page reloads, sees that the email input is set, and sends the email

Sending HTML email:

When you send a text message using PHP then all the content will be treated as simple text. Even if you will include HTML tags in a text message, it will be displayed as simple text and HTML tags will not be formatted according to HTML syntax. But PHP provides option to send an HTML message as actual HTML message.

While sending an email message you can specify a Mime version, content type and character set to send an HTML email.

Example:

Following example will send an HTML email message to xyz@somedomain.com copying it to afgh@somedomain.com. You can code this program in such a way that it should receive all content from the user and then it should send an email.

```
<html>
<head>
<title>Sending HTML email using PHP</title>
</head>
<body>
<?php
    $to = "xyz@somedomain.com";
    $subject = "This is subject";
    $message = "<b>This is HTML message.</b>";
    $message .= "<h1>This is headline.</h1>";
    $header = "From:abc@somedomain.com \r\n";
    $header = "Cc:afgh@somedomain.com \r\n";
    $header .= "MIME-Version: 1.0\r\n";
    $header .= "Content-type: text/html\r\n";
    $retval = mail ($to,$subject,$message,$header);
    if( $retval == true )
    {
        echo "Message sent successfully...";
    }
    else
    {
        echo "Message could not be sent...";
    }
?>
</body>
</html>
```

PHP Sending attachments with email:

To send an email with mixed content requires to set **Content-type** header to **multipart/mixed**. Then text and attachment sections can be specified within **boundaries**.

A boundary is started with two hyphens followed by a unique number which can not appear in the message part of the email. A PHP function **md5()** is used to create a 32 digit hexadecimal number to create unique number. A final boundary denoting the email's final section must also end with two hyphens.

Attached files should be encoded with the **base64_encode()** function for safer transmission and are best split into chunks with the **chunk_split()** function. This adds **\r\n** inside the file at regular intervals, normally every 76 characters.

Following is the example which will send a file **/tmp/test.txt** as an attachment. you can code your program to receive an uploaded file and send it.

```
<html>
<head>
<title>Sending attachment using PHP</title>
</head>
<body>
<?php
    $to = "xyz@somedomain.com";
    $subject = "This is subject";
    $message = "This is test message.";
    # Open a file
    $file = fopen( "/tmp/test.txt", "r" );
    if( $file == false )
    {
        echo "Error in opening file";
        exit();
    }
    # Read the file into a variable
    $size = filesize("/tmp/test.txt");
    $content = fread( $file, $size);

    # encode the data for safe transit
    # and insert \r\n after every 76 chars.
    $encoded_content = chunk_split( base64_encode($content));

    # Get a random 32 bit number using time() as seed.
    $num = md5( time() );

    # Define the main headers.
    $header = "From:xyz@somedomain.com\r\n";
    $header .= "MIME-Version: 1.0\r\n";
    $header .= "Content-Type: multipart/mixed; ";
    $header .= "boundary=$num\r\n";
    $header .= "--$num\r\n";

    # Define the message section
    $header .= "Content-Type: text/plain\r\n";
    $header .= "Content-Transfer-Encoding:8bit\r\n\r\n";
    $header .= "$message\r\n";
    $header .= "--$num\r\n";

    # Define the attachment section
    $header .= "Content-Type: multipart/mixed; ";
    $header .= "name=\"test.txt\"\r\n";
    $header .= "Content-Transfer-Encoding:base64\r\n";
    $header .= "Content-Disposition:attachment; ";
    $header .= "filename=\"test.txt\"\r\n\r\n";
    $header .= "$encoded_content\r\n";
    $header .= "--$num--";


    # Send email now
```

```

$retval = mail ( $to, $subject, "", $header );
if( $retval == true )
{
    echo "Message sent successfully...";
}
else
{
    echo "Message could not be sent...";
}
?>
</body>
</html>

```

Error Handling In PHP

 May 23rd, 2013  admin

The default error handling in PHP is very simple. An error message with filename, line number and a message describing the error is sent to the browser.

When creating scripts and web applications, error handling is an important part. If your code lacks error checking code, your program may look very unprofessional and you may be open to security risks.

This tutorial contains some of the most common error checking methods in PHP.

We will show different error handling methods:

- Simple "die()" statements
- Custom errors and error triggers
- Error reporting

PHP Basic Error Handling: Using the die() function

The first example shows a simple script that opens a text file:

```

<?php
$file=fopen("welcome.txt","r");
?>

```

If the file does not exist you might get an error like this:

Warning: fopen(welcome.txt) [function.fopen]: failed to open stream:
No such file or directory in C:\webfolder\test.php on line 2

To prevent the user from getting an error message like the one above, we test whether the file exist before we try to access it:

```

<?php
if(!file_exists("welcome.txt"))
{
    die("File not found");
}
else
{
    $file=fopen("welcome.txt","r");
}
?>

```

Now if the file does not exist you get an error like this:

File not found

The code above is more efficient than the earlier code, because it uses a simple error handling mechanism to stop the script after the error.

However, simply stopping the script is not always the right way to go. Let's take a look at alternative PHP functions for handling errors.

Creating a PHP Custom Error Handler

Creating a custom error handler is quite simple. We simply create a special function that can be called when an error occurs in PHP.

This function must be able to handle a minimum of two parameters (error level and error message) but can accept up to five parameters (optionally: file, line-number, and the error context):

Syntax

```
error_function(error_level,error_message,  
error_file,error_line,error_context)
```

Parameter	Description
error_level	Required. Specifies the error report level for the user-defined error. Must be a value number. See table below for possible error report levels
error_message	Required. Specifies the error message for the user-defined error
error_file	Optional. Specifies the filename in which the error occurred
error_line	Optional. Specifies the line number in which the error occurred
error_context	Optional. Specifies an array containing every variable, and their values, in use when the error occurred

PHP Error Report levels

These error report levels are the different types of error the user-defined error handler can be used for:

Value	Constant	Description
2	E_WARNING	Non-fatal run-time errors. Execution of the script is not halted
8	E_NOTICE	Run-time notices. The script found something that might be an error, but could also happen when running a script normally

Value	Constant	Description
256	E_USER_ERROR	Fatal user-generated error. This is like an E_ERROR set by the programmer using the PHP function trigger_error()
512	E_USER_WARNING	Non-fatal user-generated warning. This is like an E_WARNING set by the programmer using the PHP function trigger_error()
1024	E_USER_NOTICE	User-generated notice. This is like an E_NOTICE set by the programmer using the PHP function trigger_error()
4096	E_RECOVERABLE_ERROR	Catchable fatal error. This is like an E_ERROR but can be caught by a user defined handle (see also set_error_handler())
8191	E_ALL	All errors and warnings (E_STRICT became a part of E_ALL in PHP 5.4)

Now lets create a function to handle errors:

```
function customError($errno, $errstr)
{
    echo "<b>Error:</b> [$errno] $errstr<br>";
    echo "Ending Script";
    die();
}
```

The code above is a simple error handling function. When it is triggered, it gets the error level and an error message. It then outputs the error level and message and terminates the script.

Now that we have created an error handling function we need to decide when it should be triggered.

Set PHP Error Handler

The default error handler for PHP is the built in error handler. We are going to make the function above the default error handler for the duration of the script.

It is possible to change the error handler to apply for only some errors, that way the script can handle different errors in different ways. However, in this example we are going to use our custom error handler for all errors:

```
set_error_handler("customError");
```

Since we want our custom function to handle all errors, the set_error_handler() only needed one parameter, a second parameter could be added to specify an error level.

Example

Testing the error handler by trying to output variable that does not exist:

```
<?php
//error handler function
function customError($errno, $errstr)
{
    echo "<b>Error:</b> [$errno] $errstr";
}

//set error handler
set_error_handler("customError");

//trigger error
echo($test);
?>
```

The output of the code above should be something like this:

```
Error: [8] Undefined variable: test
```

PHP Trigger an Error

In a script where users can input data it is useful to trigger errors when an illegal input occurs. In PHP, this is done by the `trigger_error()` function.

Example

In this example an error occurs if the "test" variable is bigger than "1":

```
<?php
$test=2;
if ($test>1)
{
    trigger_error("Value must be 1 or below");
}
?>
```

The output of the code above should be something like this:

```
Notice: Value must be 1 or below
in C:\webfolder\test.php on line 6
```

An error can be triggered anywhere you wish in a script, and by adding a second parameter, you can specify what error level is triggered.

Possible error types:

- `E_USER_ERROR` – Fatal user-generated run-time error. Errors that can not be recovered from. Execution of the script is halted
- `E_USER_WARNING` – Non-fatal user-generated run-time warning. Execution of the script is not halted
- `E_USER_NOTICE` – Default. User-generated run-time notice. The script found something that might be an error, but could also happen when running a script normally

Example

In this example an `E_USER_WARNING` occurs if the "test" variable is bigger than "1". If an `E_USER_WARNING` occurs we will use our custom error handler and end the script:

```
<?php
//error handler function
function customError($errno, $errstr)
```

```

{
    echo "<b>Error:</b> [$errno] $errstr<br>";
    echo "Ending Script";
    die();
}

//set error handler
set_error_handler("customError",E_USER_WARNING);

//trigger error
$test=2;
if ($test>1)
{
    trigger_error("Value must be 1 or below",E_USER_WARNING);
}
?>

```

The output of the code above should be something like this:

```

Error: [512] Value must be 1 or below
Ending Script

```

Now that we have learned to create our own errors and how to trigger them, lets take a look at error logging.

PHP Error Logging

By default, PHP sends an error log to the server's logging system or a file, depending on how the error_log configuration is set in the php.ini file. By using the error_log() function you can send error logs to a specified file or a remote destination.

Sending error messages to yourself by e-mail can be a good way of getting notified of specific errors.

Send an Error Message by EMail

In the example below we will send an e-mail with an error message and end the script, if a specific error occurs:

```

<?php
//error handler function
function customError($errno, $errstr)
{
    echo "<b>Error:</b> [$errno] $errstr<br>";
    echo "Webmaster has been notified";
    error_log("Error: [$errno] $errstr",1,
    "someone@example.com","From: webmaster@example.com");
}

//set error handler
set_error_handler("customError",E_USER_WARNING);

//trigger error
$test=2;
if ($test>1)
{
    trigger_error("Value must be 1 or below",E_USER_WARNING);
}
?>

```


The output of the code above should be something like this:

Error: [512] Value must be 1 or below
Webmaster has been notified

And the mail received from the code above looks like this:

Error: [512] Value must be 1 or below

PHP Exception Handling

 May 24th, 2013  admin

PHP Exception Handling are used to change the normal flow of a script if a specified error occurs.

What is PHP Exception

With PHP 5 came a new object oriented way of dealing with errors.

Exception handling is used to change the normal flow of the code execution if a specified error (exceptional) condition occurs. This condition is called an exception.

This is what normally happens when an exception is triggered:

- The current code state is saved
- The code execution will switch to a predefined (custom) exception handler function
- Depending on the situation, the handler may then resume the execution from the saved code state, terminate the script execution or continue the script from a different location in the code

We will show different error handling methods:

- Basic use of Exceptions
- Creating a custom exception handler
- Multiple exceptions
- Re-throwing an exception
- Setting a top level exception handler

Note: Exceptions should only be used with error conditions, and should not be used to jump to another place in the code at a specified point.

Basic Use of PHP Exceptions

When an exception is thrown, the code following it will not be executed, and PHP will try to find the matching "catch" block.

If an exception is not caught, a fatal error will be issued with an "Uncaught Exception" message.

Lets try to throw an exception without catching it:

```
<?php
//create function with an exception
function checkNum($number)
{
    if($number>1)
    {
        throw new Exception("Value must be 1 or below");
    }
    return true;
}

//trigger exception
checkNum(2);
?>
```


The code above will get an error like this:

```
Fatal error: Uncaught exception 'Exception'  
with message 'Value must be 1 or below' in C:\webfolder\test.php:6  
Stack trace: #0 C:\webfolder\test.php(12):  
checkNum(28) #1 {main} thrown in C:\webfolder\test.php on line 6
```

Try, throw and catch

To avoid the error from the example above, we need to create the proper code to handle an exception.

Proper exception code should include:

1. Try – A function using an exception should be in a "try" block. If the exception does not trigger, the code will continue as normal. However if the exception triggers, an exception is "thrown"
 2. Throw – This is how you trigger an exception. Each "throw" must have at least one "catch"
 3. Catch – A "catch" block retrieves an exception and creates an object containing the exception information
- Lets try to trigger an exception with valid code:

```
<?php  
//create function with an exception  
function checkNum($number)  
{  
    if($number>1)  
    {  
        throw new Exception("Value must be 1 or below");  
    }  
    return true;  
}  
  
//trigger exception in a "try" block  
try  
{  
    checkNum(2);  
    //If the exception is thrown, this text will not be shown  
    echo 'If you see this, the number is 1 or below';  
}  
  
//catch exception  
catch(Exception $e)  
{  
    echo 'Message: ' . $e->getMessage();  
}  
?>
```

The code above will get an error like this:

```
Message: Value must be 1 or below
```

Example explained:

The code above throws an exception and catches it:

1. The checkNum() function is created. It checks if a number is greater than 1. If it is, an exception is thrown
2. The checkNum() function is called in a "try" block
3. The exception within the checkNum() function is thrown
4. The "catch" block retrives the exception and creates an object (\$e) containing the exception information
5. The error message from the exception is echoed by calling \$e->getMessage() from the exception object

However, one way to get around the "every throw must have a catch" rule is to set a top level exception handler to handle errors that slip through.

Creating a Custom PHP Exception Class

Creating a custom exception handler is quite simple. We simply create a special class with functions that can be called when an exception occurs in PHP. The class must be an extension of the exception class.

The custom exception class inherits the properties from PHP's exception class and you can add custom functions to it.

Lets create an exception class:

```
<?php
class customException extends Exception
{
    public function errorMessage()
    {
        //error message
        $errorMsg = 'Error on line '.$this->getLine().' in '.$this->getFile()
        .': <b>'.$this->getMessage().'</b> is not a valid E-Mail address';
        return $errorMsg;
    }
}

$email = "someone@example...com";

try
{
    //check if
    if(filter_var($email, FILTER_VALIDATE_EMAIL) === FALSE)
    {
        //throw exception if email is not valid
        throw new customException($email);
    }
}

catch (customException $e)
{
    //display custom message
    echo $e->errorMessage();
}

?>
```

The new class is a copy of the old exception class with an addition of the `errorMessage()` function. Since it is a copy of the old class, and it inherits the properties and methods from the old class, we can use the exception class methods like `getLine()` and `getFile()` and `getMessage()`.

Example explained:

The code above throws an exception and catches it with a custom exception class:

1. The `customException()` class is created as an extension of the old exception class. This way it inherits all methods and properties from the old exception class
2. The `errorMessage()` function is created. This function returns an error message if an e-mail address is invalid
3. The `$email` variable is set to a string that is not a valid e-mail address
4. The "try" block is executed and an exception is thrown since the e-mail address is invalid
5. The "catch" block catches the exception and displays the error message

Multiple PHP Exceptions

It is possible for a script to use multiple exceptions to check for multiple conditions.

It is possible to use several if..else blocks, a switch, or nest multiple exceptions. These exceptions can use different exception classes and return different error messages:

```
<?php
class customException extends Exception
{
    public function errorMessage()
    {
        //error message
        $errorMsg = 'Error on line '.$this->getLine().' in '.$this->getFile()
        .': <b>'.$this->getMessage().</b> is not a valid E-Mail address';
        return $errorMsg;
    }
}

$email = "someone@example.com";

try
{
    //check if
    if(filter_var($email, FILTER_VALIDATE_EMAIL) === FALSE)
    {
        //throw exception if email is not valid
        throw new customException($email);
    }
    //check for "example" in mail address
    if(strpos($email, "example") !== FALSE)
    {
        throw new Exception("$email is an example e-mail");
    }
}

catch (customException $e)
{
    echo $e->errorMessage();
}

catch(Exception $e)
{
    echo $e->getMessage();
}

?>
```

Example explained:

The code above tests two conditions and throws an exception if any of the conditions are not met:

1. The customException() class is created as an extension of the old exception class. This way it inherits all methods and properties from the old exception class
2. The errorMessage() function is created. This function returns an error message if an e-mail address is invalid
3. The \$email variable is set to a string that is a valid e-mail address, but contains the string "example"
4. The "try" block is executed and an exception is not thrown on the first condition
5. The second condition triggers an exception since the e-mail contains the string "example"
6. The "catch" block catches the exception and displays the correct error message

If the exception thrown were of the class customException and there were no customException catch, only the base exception catch, the exception would be handled there.

Re-throwing PHP Exceptions

Sometimes, when an exception is thrown, you may wish to handle it differently than the standard way. It is possible to throw an exception a second time within a "catch" block.

A script should hide system errors from users. System errors may be important for the coder, but is of no interest to the user. To make things easier for the user you can re-throw the exception with a user friendly message:

```
<?php
class customException extends Exception
{
    public function errorMessage()
    {
        //error message
        $errorMsg = $this->getMessage().' is not a valid E-Mail address.';
        return $errorMsg;
    }
}

$email = "someone@example.com";

try
{
    try
    {
        //check for "example" in mail address
        if(strpos($email, "example") !== FALSE)
        {
            //throw exception if email is not valid
            throw new Exception($email);
        }
    }
    catch(Exception $e)
    {
        //re-throw exception
        throw new customException($email);
    }
}

catch (customException $e)
{
    //display custom message
    echo $e->errorMessage();
}

?>
```

Example explained:

The code above tests if the email-address contains the string "example" in it, if it does, the exception is re-thrown:

1. The customException() class is created as an extension of the old exception class. This way it inherits all methods and properties from the old exception class
2. The errorMessage() function is created. This function returns an error message if an e-mail address is invalid
3. The \$email variable is set to a string that is a valid e-mail address, but contains the string "example"
4. The "try" block contains another "try" block to make it possible to re-throw the exception
5. The exception is triggered since the e-mail contains the string "example"
6. The "catch" block catches the exception and re-throws a "customException"
7. The "customException" is caught and displays an error message

If the exception is not caught in its current "try" block, it will search for a catch block on "higher levels".

Set a Top Level Exception Handler

The `set_exception_handler()` function sets a user-defined function to handle all uncaught exceptions.

```
<?php
function myException($exception)
{
    echo "<b>Exception:</b> ", $exception->getMessage();
}

set_exception_handler('myException');
throw new Exception('Uncaught Exception occurred');
?>
```

The output of the code above should be something like this:

Exception: Uncaught Exception occurred

In the code above there was no "catch" block. Instead, the top level exception handler triggered. This function should be used to catch uncaught exceptions.

Rules for php exceptions

- Code may be surrounded in a try block, to help catch potential exceptions
- Each try block or "throw" must have at least one corresponding catch block
- Multiple catch blocks can be used to catch different classes of exceptions
- Exceptions can be thrown (or re-thrown) in a catch block within a try block

PHP MySQL Introduction

What is MySQL?

- MySQL is a database system used on the web
- MySQL is a database system that runs on a server
- MySQL is ideal for both small and large applications
- MySQL is very fast, reliable, and easy to use
- MySQL supports standard SQL
- MySQL compiles on a number of platforms
- MySQL is free to download and use
- MySQL is developed, distributed, and supported by Oracle Corporation
- MySQL is named after co-founder Monty Widenius's daughter: My

PHP + MySQL

- PHP combined with MySQL are cross-platform (you can develop in Windows and serve on a Unix platform)

Queries

A query is a question or a request.

We can query a database for specific information and have a recordset returned.

Look at the following query (using standard SQL):

```
SELECT LastName FROM Employees
```

PHP 5 MySQLi Functions

Function	Description
<u>mysqli_affected_rows()</u>	Returns the number of affected rows in the previous MySQL operation
<u>mysqli_autocommit()</u>	Turns on or off auto-committing database modifications
<u>mysqli_change_user()</u>	Changes the user of the specified database connection
<u>mysqli_character_set_name()</u>	Returns the default character set for the database connection
<u>mysqli_close()</u>	Closes a previously opened database connection
<u>mysqli_commit()</u>	Commits the current transaction
<u>mysqli_connect_errno()</u>	Returns the error code from the last connection error
<u>mysqli_connect_error()</u>	Returns the error description from the last connection error
<u>mysqli_connect()</u>	Opens a new connection to the MySQL server
<u>mysqli_data_seek()</u>	Adjusts the result pointer to an arbitrary row in the result-set
<u>mysqli_debug()</u>	Performs debugging operations
<u>mysqli_dump_debug_info()</u>	Dumps debugging info into the log
<u>mysqli_errno()</u>	Returns the last error code for the most recent function call
<u>mysqli_error_list()</u>	Returns a list of errors for the most recent function call
<u>mysqli_error()</u>	Returns the last error description for the most recent function call
<u>mysqli_fetch_all()</u>	Fetches all result rows as an associative array, a numeric array, or both

<u>mysqli_fetch_array()</u>	Fetches a result row as an associative, a numeric array, or both
<u>mysqli_fetch_assoc()</u>	Fetches a result row as an associative array
<u>mysqli_fetch_field_direct()</u>	Returns meta-data for a single field in the result set, as an object
<u>mysqli_fetch_field()</u>	Returns the next field in the result set, as an object
<u>mysqli_fetch_fields()</u>	Returns an array of objects that represent the fields in a result set
<u>mysqli_fetch_lengths()</u>	Returns the lengths of the columns of the current row in the result set
<u>mysqli_fetch_object()</u>	Returns the current row of a result set, as an object
<u>mysqli_fetch_row()</u>	Fetches one row from a result-set and returns it as an enumerated array
<u>mysqli_field_count()</u>	Returns the number of columns for the most recent query
<u>mysqli_field_seek()</u>	Sets the field cursor to the given field offset
<u>mysqli_field_tell()</u>	Returns the position of the field cursor
<u>mysqli_free_result()</u>	Frees the memory associated with a result
<u>mysqli_get_charset()</u>	Returns a character set object
<u>mysqli_get_client_info()</u>	Returns the MySQL client library version
<u>mysqli_get_client_stats()</u>	Returns statistics about client per-process
<u>mysqli_get_client_version()</u>	Returns the MySQL client library version as an integer
<u>mysqli_get_connection_stats()</u>	Returns statistics about the client connection
<u>mysqli_get_host_info()</u>	Returns the MySQL server hostname and the connection type
<u>mysqli_get_proto_info()</u>	Returns the MySQL protocol version
<u>mysqli_get_server_info()</u>	Returns the MySQL server version
<u>mysqli_get_server_version()</u>	Returns the MySQL server version as an integer
<u>mysqli_info()</u>	Returns information about the most recently executed query
<u>mysqli_init()</u>	Initializes MySQLi and returns a resource for use with <code>mysqli_real_connect()</code>
<u>mysqli_insert_id()</u>	Returns the auto-generated id used in the last query

<u>mysql_kill()</u>	Asks the server to kill a MySQL thread
<u>mysqli_more_results()</u>	Checks if there are more results from a multi query
<u>mysqli_multi_query()</u>	Performs one or more queries on the database
<u>mysqli_next_result()</u>	Prepares the next result set from mysqli_multi_query()
<u>mysqli_num_fields()</u>	Returns the number of fields in a result set
<u>mysqli_num_rows()</u>	Returns the number of rows in a result set
<u>mysqli_options()</u>	Sets extra connect options and affect behavior for a connection
<u>mysqli_ping()</u>	Pings a server connection, or tries to reconnect if the connection has gone down
mysqli_prepare()	Prepares an SQL statement for execution
<u>mysqli_query()</u>	Performs a query against the database
<u>mysqli_real_connect()</u>	Opens a new connection to the MySQL server
<u>mysqli_real_escape_string()</u>	Escapes special characters in a string for use in an SQL statement
mysqli_real_query()	Executes an SQL query
mysqli_reap_async_query()	Returns the result from async query
<u>mysqli_refresh()</u>	Refreshes tables or caches, or resets the replication server information
<u>mysqli_rollback()</u>	Rolls back the current transaction for the database
<u>mysqli_select_db()</u>	Changes the default database for the connection
<u>mysqli_set_charset()</u>	Sets the default client character set
mysqli_set_local_infile_default()	Unsets user defined handler for load local infile command
mysqli_set_local_infile_handler()	Set callback function for LOAD DATA LOCAL INFILE command
<u>mysqli_sqlstate()</u>	Returns the SQLSTATE error code for the last MySQL operation
<u>mysqli_ssl_set()</u>	Used to establish secure connections using SSL
<u>mysqli_stat()</u>	Returns the current system status

<u>mysqli_stmt_init()</u>	Initializes a statement and returns an object for use with mysqli_stmt_prepare()
mysqli_store_result()	Transfers a result set from the last query
<u>mysqli_thread_id()</u>	Returns the thread ID for the current connection
<u>mysqli_thread_safe()</u>	Returns whether the client library is compiled as thread-safe
mysqli_use_result()	Initiates the retrieval of a result set from the last query executed using the mysqli_real_query()
mysqli_warning_count()	Returns the number of warnings from the last query in the connection

PHP How Connect to the MySQL

 June 2nd, 2013  admin

Use the PHP mysqli_connect() function to open a new connection to the MySQL server.

Open a Connection to the MySQL Server :

Before we can access data in a database, we must open a connection to the MySQL server.

In PHP, this is done with the mysqli_connect() function.

Syntax

```
mysqli_connect(host,username,password,dbname);
```

Parameter	Description
host	Optional. Either a host name or an IP address
username	Optional. The MySQL user name
password	Optional. The password to log in with
dbname	Optional. The default database to be used when performing queries

Note: There are more available parameters, but the ones listed above are the most important.

In the following example we store the connection in a variable (\$con) for later use in the script:

```
<?php
// Create connection
$con=mysqli_connect("example.com","peter","abc123","my_db");
// Check connection
if (mysqli_connect_errno($con))
{
    echo "Failed to connect to MySQL: " . mysqli_connect_error();
}
?>
```

How Close a Connection

The connection will be closed automatically when the script ends. To close the connection before, use the `mysqli_close()` function:

```
<?php
$con=mysqli_connect("example.com","peter","abc123","my_db");
// Check connection
if (mysqli_connect_errno($con))
{
    echo "Failed to connect to MySQL: " . mysqli_connect_error();
}

mysqli_close($con);
?>
```

PHP Mysql Create Database and Tables

 June 3rd, 2013  admin

A database holds one or more tables. To create and delete a database you should have admin privilege. It's very easy to create a new MySQL database. PHP uses `mysql_query` function to create a MySQL database. This function takes two parameters and returns TRUE on success or FALSE on failure.

Syntax :

```
mysql_query( sql, connection );
```

Create a Database

We must add the CREATE DATABASE statement to the `mysql_query()` function to execute the command.

The following example creates a database named "my_db":

```
<?php
$con=mysqli_connect("example.com","peter","abc123");
// Check connection
if (mysqli_connect_errno())
{
    echo "Failed to connect to MySQL: " . mysqli_connect_error();
}

// Create database
$sql="CREATE DATABASE my_db";
if (mysql_query($con,$sql))
{
    echo "Database my_db created successfully";
}
else
{

```

```

echo "Error creating database: " . mysqli_error($con);
}
?>

```

Creating Database 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 `mysqli_query()` function.

The CREATE TABLE statement is used to create a table in MySQL.

We must add the CREATE TABLE statement to the `mysqli_query()` function to execute the command.

The following example creates a table named "Persons", with three columns. The column names will be "FirstName", "LastName" and "Age":

```

<?php
$con=mysqli_connect("example.com","peter","abc123","my_db");
// Check connection
if (mysqli_connect_errno())
{
    echo "Failed to connect to MySQL: " . mysqli_connect_error();
}

// Create table
$sql="CREATE TABLE persons(FirstName CHAR(30),LastName
CHAR(30),Age INT)";

// Execute query
if (mysqli_query($con,$sql))
{
    echo "Table persons created successfully";
}
else
{
    echo "Error creating table: " . mysqli_error($con);
}
?>

```

Note: When you create a database field of type CHAR, you must specify the maximum length of the field, e.g. CHAR(50).The data type specifies what type of data the column can hold. For a complete reference of all the data types available in MySQL, go to our complete [Data Types reference](#).

Example 2:

Try out following example to create a table:

```

<?php
$dbhost = 'localhost:3036';
$dbuser = 'root';
$dbpass = 'rootpassword';
$conn = mysql_connect($dbhost, $dbuser, $dbpass);
if(! $conn )
{
    die('Could not connect: ' . mysql_error());
}
echo 'Connected successfully';
$sql = 'CREATE TABLE employee( '

```

```

'emp_id INT NOT NULL AUTO_INCREMENT, '
'emp_name VARCHAR(20) NOT NULL, '
'emp_address VARCHAR(20) NOT NULL, '
'emp_salary INT NOT NULL, '
'join_date timestamp(14) NOT NULL, '
'primary key ( emp_id ));

mysql_select_db('test_db');
$retval = mysql_query( $sql, $conn );
if( ! $retval )
{
    die('Could not create table: ' . mysql_error());
}
echo "Table employee created successfully\n";
mysql_close($conn);
?>

```

In case you need to create many tables then its better to create a text file first and put all the SQL commands in that text file and then load that file into \$sql variable and excute those commands.

Consider the following content in sql_query.txt file

```

CREATE TABLE employee(
    emp_id INT NOT NULL AUTO_INCREMENT,
    emp_name VARCHAR(20) NOT NULL,
    emp_address VARCHAR(20) NOT NULL,
    emp_salary INT NOT NULL,
    join_date timestamp(14) NOT NULL,
    primary key ( emp_id ));

<?php
$dbhost = 'localhost:3036';
$dbuser = 'root';
$dbpass = 'rootpassword';
$conn = mysql_connect($dbhost, $dbuser, $dbpass);
if( ! $conn )
{
    die('Could not connect: ' . mysql_error());
}
$query_file = 'sql_query.txt';
$fp = fopen($query_file, 'r');
$sql = fread($fp, filesize($query_file));
fclose($fp);

mysql_select_db('test_db');
$retval = mysql_query( $sql, $conn );
if( ! $retval )
{
    die('Could not create table: ' . mysql_error());
}
echo "Table employee created successfully\n";
mysql_close($conn);
?>

```

Primary Keys and Auto Increment Fields

Each table in a database should have a primary key field.

A primary key is used to uniquely identify the rows in a table. Each primary key value must be unique within the table. Furthermore, the primary key field cannot be null because the database engine requires a value to locate the record.

The following example sets the PID field as the primary key field. The primary key field is often an ID number, and is often used with the AUTO_INCREMENT setting. AUTO_INCREMENT automatically increases the value of the field by 1 each time a new record is added. To ensure that the primary key field cannot be null, we must add the NOT NULL setting to the field:

```
$sql = "CREATE TABLE Persons
(
PID INT NOT NULL AUTO_INCREMENT,
PRIMARY KEY(PID),
FirstName CHAR(15),
LastName CHAR(15),
Age INT
)";
```

PHP MySQL Insert Data

 June 4th, 2013  admin

Data can be entered into MySQL tables by executing SQL INSERT statement through PHP function `mysql_query`. The INSERT INTO statement is used to insert new records in a table.

Insert Data Into a Database Table

The INSERT INTO statement is used to add new records to a database table.

Syntax

It is possible to write the INSERT INTO statement in two forms.

The first form doesn't specify the column names where the data will be inserted, only their values:

```
INSERT INTO table_name
VALUES (value1, value2, value3,...)
```

The second form specifies both the column names and the values to be inserted:

```
INSERT INTO table_name (column1, column2, column3,...)
VALUES (value1, value2, value3,...)
```

To learn more about SQL, please visit our [SQL tutorial](#).

To get PHP to execute the statements above we must use the `mysql_query()` function. This function is used to send a query or command to a MySQL connection.

Example

In the previous chapter we created a table named "Persons", with three columns; "FirstName", "LastName" and "Age". We will use the same table in this example. The following example adds two new records to the "Persons" table:

```
<?php
$con=mysqli_connect("example.com","peter","abc123","my_db");
// Check connection
if (mysqli_connect_errno())
{
echo "Failed to connect to MySQL: " . mysqli_connect_error();
}
```

```
mysqli_query($con,"INSERT INTO Persons (FirstName, LastName,
Age)
VALUES ('Peter', 'Griffin',35)");

mysqli_query($con,"INSERT INTO Persons (FirstName, LastName,
Age)
VALUES ('Glenn', 'Quagmire',33)");

mysqli_close($con);
?>
```

PHP Insert Data From a Form Into a Database

Now we will create an HTML form that can be used to add new records to the "Persons" table.

Here is the HTML form:

```
<html>
<body>

<form action="insert.php" method="post">
Firstname: <input type="text" name="firstname">
Lastname: <input type="text" name="lastname">
Age: <input type="text" name="age">
<input type="submit">
</form>

</body>
</html>
```

When a user clicks the submit button in the HTML form in the example above, the form data is sent to "insert.php".

The "insert.php" file connects to a database, and retrieves the values from the form with the PHP \$_POST variables.

Then, the mysqli_query() function executes the INSERT INTO statement, and a new record will be added to the "Persons" table.

Here is the "insert.php" page:

```
<?php
$con=mysqli_connect("example.com","peter","abc123","my_db");
// Check connection
if (mysqli_connect_errno())
{
    echo "Failed to connect to MySQL: " . mysqli_connect_error();
}

$sql="INSERT INTO Persons (FirstName, LastName, Age)
VALUES
('$$_POST[firstname]','$$_POST[lastname]','$$_POST[age]')";

if (!mysqli_query($con,$sql))
{
    die('Error: ' . mysqli_error($con));
}
echo "1 record added";

mysqli_close($con);
?>
```

Example:

Try out following example to insert record into employee table.

```
<?php
$dbhost = 'localhost:3036';
$dbuser = 'root';
$dbpass = 'rootpassword';
$conn = mysql_connect($dbhost, $dbuser, $dbpass);
if(! $conn )
{
    die('Could not connect: ' . mysql_error());
}
$sql = 'INSERT INTO employee ' .
    '(emp_name,emp_address, emp_salary, join_date) ' .
    'VALUES ( "guest", "XYZ", 2000, NOW() )';
mysql_select_db('test_db');
$retval = mysql_query( $sql, $conn );
if(! $retval )
{
    die('Could not enter data: ' . mysql_error());
}
echo "Entered data successfully\n";
mysql_close($conn);
?>
```

In real application, all the values will be taken using HTML form and then those values will be captured using PHP script and finally they will be inserted into MySQL tables.

While doing data insert its best practice to use function `get_magic_quotes_gpc()` to check if current configuration for magic quote is set or not. If this function returns false then use function `addslashes()` to add slashes before quotes.

Example:

Try out this example by putting this code into `add_employee.php`, this will take input using HTML Form and then it will create records into database.

```
<html>
<head>
<title>Add New Record in MySQL Database</title>
</head>
<body>
<?php
if(isset($_POST['add']))
{
    $dbhost = 'localhost:3036';
    $dbuser = 'root';
    $dbpass = 'rootpassword';
    $conn = mysql_connect($dbhost, $dbuser, $dbpass);
    if(! $conn )
    {
        die('Could not connect: ' . mysql_error());
    }

    if(! get_magic_quotes_gpc() )
    {
        $emp_name = addslashes ($_POST['emp_name']);
        $emp_address = addslashes ($_POST['emp_address']);
    }
    else
```

```

{
    $emp_name = $_POST['emp_name'];
    $emp_address = $_POST['emp_address'];
}
$emp_salary = $_POST['emp_salary'];
$sql = "INSERT INTO employee ".
        "(emp_name,emp_address, emp_salary, join_date) ".
        "VALUES('$emp_name','$emp_address',$emp_salary, NOW())";
mysql_select_db('test_db');
$retval = mysql_query( $sql, $conn );
if( ! $retval )
{
    die('Could not enter data: ' . mysql_error());
}
echo "Entered data successfully\n";
mysql_close($conn);
}
else
{
    ?>
<form method="post" action="<?php $_PHP_SELF ?>">
<table width="400" border="0" cellspacing="1" cellpadding="2">
<tr>
<td width="100">Employee Name</td>
<td><input name="emp_name" type="text" id="emp_name"> </td>
</tr>
<tr>
<td width="100">Employee Address</td>
<td><input name="emp_address" type="text"
id="emp_address"> </td>
</tr>
<tr>
<td width="100">Employee Salary</td>
<td><input name="emp_salary" type="text" id="emp_salary"> </td>
</tr>
<tr>
<td width="100"> </td>
<td> </td>
</tr>
<tr>
<td width="100"> </td>
<td>
<input name="add" type="submit" id="add" value="Add
Employee">
</td>
</tr>
</table>
</form>
<?php
}
?>
</body>
</html>

```

PHP MySQL Select



June 12th, 2013 admin

The SELECT statement is used to select data from a database.

Select Data From a Database Table

The SELECT statement is used to select data from a database.

Syntax

```
SELECT column_name(s)
FROM table_name
```

To get PHP to execute the statement above we must use the `mysqli_query()` function. This function is used to send a query or command to a MySQL connection.

Data can be fetched from MySQL tables by executing SQL SELECT statement through PHP function `mysqli_query`. You have several options to fetch data from MySQL.

The most frequently used option is to use function `mysqli_fetch_array()`. This function returns row as an associative array, a numeric array, or both. This function returns FALSE if there are no more rows.

The following example selects all the data stored in the "Persons" table (The * character selects all the data in the table):

```
<?php
$con=mysqli_connect("example.com","peter","abc123","my_db");
// Check connection
if (mysqli_connect_errno())
{
    echo "Failed to connect to MySQL: " . mysqli_connect_error();
}

$result = mysqli_query($con,"SELECT * FROM Persons");

while($row = mysqli_fetch_array($result))
{
    echo $row['FirstName'] . " " . $row['LastName'];
    echo "<br>";
}

mysqli_close($con);
?>
```

The example above stores the data returned by the `mysqli_query()` function in the `$result` variable.

Next, we use the `mysqli_fetch_array()` function to return the first row from the recordset as an array. Each call to `mysqli_fetch_array()` returns the next row in the recordset. The while loop loops through all the records in the recordset. To print the value of each row, we use the PHP `$row` variable (`$row['FirstName']` and `$row['LastName']`).

The output of the code above will be:

```
Peter Griffin
Glenn Quagmire
```

Below is a simple example to fetch records from **employee** table.

Example :

Try out following example to display all the records from employee table.

```
<?php
$dbhost = 'localhost:3036';
$dbuser = 'root';
$dbpass = 'rootpassword';
```

```

$conn = mysql_connect($dbhost, $dbuser, $dbpass);
if(! $conn )
{
    die('Could not connect: ' . mysql_error());
}
$sql = 'SELECT emp_id, emp_name, emp_salary FROM employee';
mysql_select_db('test_db');
$retval = mysql_query( $sql, $conn );
if(! $retval )
{
    die('Could not get data: ' . mysql_error());
}
while($row = mysql_fetch_array($retval, MYSQL_ASSOC))
{
    echo "EMP ID :{$row['emp_id']} <br> ".
        "EMP NAME : {$row['emp_name']} <br> ".
        "EMP SALARY : {$row['emp_salary']} <br> ".
        "_____<br>";
}
echo "Fetched data successfully\n";
mysql_close($conn);
?>

```

The content of the rows are assigned to the variable \$row and the values in row are then printed.

NOTE: Always remember to put curly brackets when you want to insert an array value directly into a string.

In above example the constant **MYSQL_ASSOC** is used 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.

PHP provides another function called `mysql_fetch_assoc()` which also returns the row as an associative array.

Example :

Try out following example to display all the records from employee table using `mysql_fetch_assoc()` function.

```

<?php
$dbhost = 'localhost:3036';
$dbuser = 'root';
$dbpass = 'rootpassword';
$conn = mysql_connect($dbhost, $dbuser, $dbpass);
if(! $conn )
{
    die('Could not connect: ' . mysql_error());
}
$sql = 'SELECT emp_id, emp_name, emp_salary FROM employee';
mysql_select_db('test_db');
$retval = mysql_query( $sql, $conn );
if(! $retval )
{
    die('Could not get data: ' . mysql_error());
}
while($row = mysql_fetch_assoc($retval))
{
    echo "EMP ID :{$row['emp_id']} <br> ".
        "EMP NAME : {$row['emp_name']} <br> ".
        "EMP SALARY : {$row['emp_salary']} <br> ".
        "_____<br>";
}

```

```

}
echo "Fetched data successfully\n";
mysql_close($conn);
?>

```

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.

Example:

Try out following example to display all the records from employee table using **MYSQL_NUM** argument.

```

<?php
$dbhost = 'localhost:3036';
$dbuser = 'root';
$dbpass = 'rootpassword';
$conn = mysql_connect($dbhost, $dbuser, $dbpass);
if(! $conn )
{
    die('Could not connect: ' . mysql_error());
}
$sql = 'SELECT emp_id, emp_name, emp_salary FROM employee';
mysql_select_db('test_db');
$retval = mysql_query( $sql, $conn );
if(! $retval )
{
    die('Could not get data: ' . mysql_error());
}
while($row = mysql_fetch_array($retval, MYSQL_NUM))
{
    echo "EMP ID :{$row[0]} <br> ".
        "EMP NAME : {$row[1]} <br> ".
        "EMP SALARY : {$row[2]} <br> ".
        "_____<br>";
}
echo "Fetched data successfully\n";
mysql_close($conn);
?>

```

All the above three examples will produce same result.

Releasing Memory:

It's a good practice to release cursor memory at the end of each **SELECT** statement. This can be done by using PHP function **mysql_free_result()**. Below is the example to show how it has to be used.

Example:

Try out following example

```

<?php
$dbhost = 'localhost:3036';
$dbuser = 'root';
$dbpass = 'rootpassword';
$conn = mysql_connect($dbhost, $dbuser, $dbpass);
if(! $conn )
{
    die('Could not connect: ' . mysql_error());
}

```

```

$sql = 'SELECT emp_id, emp_name, emp_salary FROM employee';
mysql_select_db('test_db');
$retval = mysql_query( $sql, $conn );
if( ! $retval )
{
    die('Could not get data: ' . mysql_error());
}
while($row = mysql_fetch_array($retval, MYSQL_NUM))
{
    echo "EMP ID :{$row[0]} <br> ".
        "EMP NAME : {$row[1]} <br> ".
        "EMP SALARY : {$row[2]} <br> ".
        "_____<br>";
}
mysql_free_result($retval);
echo "Fetched data successfully\n";
mysql_close($conn);
?>

```

While fetching data you can write as complex SQL as you like. Procedure will remain same as mentioned above.

Display the Result in an HTML Table

The following example selects the same data as the example above, but will display the data in an HTML table:

```

<?php
$con=mysqli_connect("example.com","peter","abc123","my_db");
// Check connection
if (mysqli_connect_errno())
{
    echo "Failed to connect to MySQL: " . mysqli_connect_error();
}

$result = mysqli_query($con,"SELECT * FROM Persons");

echo "<table border='1'>
<tr>
<th>Firstname</th>
<th>Lastname</th>
</tr>";

while($row = mysqli_fetch_array($result))
{
    echo "<tr>";
    echo "<td>" . $row['FirstName'] . "</td>";
    echo "<td>" . $row['LastName'] . "</td>";
    echo "</tr>";
}
echo "</table>";

mysqli_close($con);
?>


```

The output of the code above will be:

Firstname	Lastname
Glenn	Quagmire

Peter	Griffin
-------	---------

PHP MySQL Where

 June 12th, 2013  admin

The WHERE clause is used to filter records.

The WHERE clause is used to extract only those records that fulfill a specified criterion.

Syntax

```
SELECT column_name(s)
FROM table_name
WHERE column_name operator value
```

To get PHP to execute the statement above we must use the `mysqli_query()` function. This function is used to send a query or command to a MySQL connection.

Example

The following example selects all rows from the "Persons" table where "FirstName='Peter'":

```
<?php
$con=mysqli_connect("example.com","peter","abc123","my_db");
// Check connection
if (mysqli_connect_errno())
{
    echo "Failed to connect to MySQL: " . mysqli_connect_error();
}

$result = mysqli_query($con,"SELECT * FROM Persons
WHERE FirstName='Peter'");

while($row = mysqli_fetch_array($result))
{
    echo $row['FirstName'] . " " . $row['LastName'];
    echo "<br>";
}
?>
```

The output of the code above will be:

Peter Griffin

PHP MySQL Order By

 June 13th, 2013  admin

The ORDER BY keyword is used to sort the data in a recordset.

The ORDER BY keyword sort the records in ascending order by default.

If you want to sort the records in a descending order, you can use the DESC keyword.

Syntax

```
SELECT column_name(s)
FROM table_name
ORDER BY column_name(s) ASC|DESC
```

Example

The following example selects all the data stored in the "Persons" table, and sorts the result by the "Age" column:

```
<?php
$con=mysqli_connect("example.com","peter","abc123","my_db");
// Check connection
if (mysqli_connect_errno())
{
    echo "Failed to connect to MySQL: " . mysqli_connect_error();
}

$result = mysqli_query($con,"SELECT * FROM Persons ORDER BY
age");

while($row = mysqli_fetch_array($result))
{
    echo $row['FirstName'];
    echo " " . $row['LastName'];
    echo " " . $row['Age'];
    echo "<br>";
}

mysqli_close($con);
?>
```

The output of the code above will be:

```
Glenn Quagmire 33
Peter Griffin 35
```

Order by Two Columns

It is also possible to order by more than one column. When ordering by more than one column, the second column is only used if the values in the first column are equal:

```
SELECT column_name(s)
FROM table_name
ORDER BY column1, column2
```

PHP MySQL Update Data

 June 13th, 2013  admin

The UPDATE statement is used to update existing records in a table.

Update Data In a Database syntax:

```
UPDATE table_name
SET column1=value, column2=value2,...
```

```
WHERE some_column=some_value
```

Note: Notice the WHERE clause in the UPDATE syntax. The WHERE clause specifies which record or records that should be updated. If you omit the WHERE clause, all records will be updated!

To get PHP to execute the statement above we must use the `mysqli_query()` function. This function is used to send a query or command to a MySQL connection.

Example

Earlier in the tutorial we created a table named "Persons". Here is how it looks:

FirstName	LastName	Age
Peter	Griffin	35
Glen	Quagmire	33

The following example updates some data in the "Persons" table:

```
<?php
$con=mysqli_connect("example.com","peter","abc123","my_db");
// Check connection
if (mysqli_connect_errno())
{
    echo "Failed to connect to MySQL: " . mysqli_connect_error();
}

mysqli_query($con,"UPDATE Persons SET Age=36
WHERE FirstName='Peter' AND LastName='Griffin'");

mysqli_close($con);
?>
```

After the update, the "Persons" table will look like this:

FirstName	LastName	Age
Peter	Griffin	36
Glen	Quagmire	33

PHP MySQL Delete Data

 June 16th, 2013  admin

The DELETE FROM statement is used to delete records from a database table.

Syntax

```
DELETE FROM table_name
WHERE some_column = some_value
```

Note: Notice the WHERE clause in the DELETE syntax. The WHERE clause specifies which record or records that should be deleted. If you omit the WHERE clause, all records will be deleted!

To get PHP to execute the statement above we must use the `mysqli_query()` function. This function is used to send a query or command to a MySQL connection.

Example

Look at the following "Persons" table:

FirstName	LastName	Age
Peter	Griffin	35
Glenn	Quagmire	33

The following example deletes all the records in the "Persons" table where LastName='Griffin':

```
<?php
$con=mysqli_connect("example.com","peter","abc123","my_db");
// Check connection
if (mysqli_connect_errno())
{
    echo "Failed to connect to MySQL: " . mysqli_connect_error();
}

mysqli_query($con,"DELETE FROM Persons WHERE
LastName='Griffin'");

mysqli_close($con);
?>
```

After the deletion, the table will look like this:

FirstName	LastName	Age
Glenn	Quagmire	33

PHP MySQL Creating Paging

 June 17th, 2013  admin

Its always possible that your **SQL SELECT statement query** may result into thousand of records. But its is not good idea to display all the results on one page. So we can divide this result into many pages as per requirement.

Paging means showing your query result in multiple pages instead of just put them all in one long page.

MySQL helps to generate paging by using **LIMIT** clause which will take two arguments. First argument as OFFSET and second argument how many records should be returned from the database.

Below is a simple example to fetch records using **LIMIT** clause to generate paging.

Example:

Try out following example to display 10 records per page.

```
<html>
<head>
<title>Paging Using PHP</title>
</head>
<body>
<?php
$dbhost = 'localhost:3036';
$dbuser = 'root';
$dbpass = 'rootpassword';
$rec_limit = 10;

$conn = mysql_connect($dbhost, $dbuser, $dbpass);
if(! $conn )
```



```

{
    die('Could not connect: ' . mysql_error());
}
mysql_select_db('test_db');
/* Get total number of records */
$sql = "SELECT count(emp_id) FROM employee ";
$retval = mysql_query( $sql, $conn );
if( ! $retval )
{
    die('Could not get data: ' . mysql_error());
}
$row = mysql_fetch_array($retval, MYSQL_NUM );
$rec_count = $row[0];
if( isset($_GET{'page'}) )
{
    $page = $_GET{'page'} + 1;
    $offset = $rec_limit * $page ;
}
else
{
    $page = 0;
    $offset = 0;
}
$left_rec = $rec_count - ($page * $rec_limit);
$sql = "SELECT emp_id, emp_name, emp_salary ".
        "FROM employee ".
        "LIMIT $offset, $rec_limit";
$retval = mysql_query( $sql, $conn );
if( ! $retval )
{
    die('Could not get data: ' . mysql_error());
}
while($row = mysql_fetch_array($retval, MYSQL_ASSOC))
{
    echo "EMP ID :{$row['emp_id']} <br> ".
        "EMP NAME : {$row['emp_name']} <br> ".
        "EMP SALARY : {$row['emp_salary']} <br> ".
        "_____<br>";
}
if( $page > 0 )
{
    $last = $page - 2;
    echo "<a href=\"$_PHP_SELF?page=$last\">Last 10 Records</a>";
    echo "<a href=\"$_PHP_SELF?page=$page\">Next 10 Records</a>";
}
else if( $page == 0 )
{
    echo "<a href=\"$_PHP_SELF?page=$page\">Next 10 Records</a>";
}
else if( $left_rec < $rec_limit )
{
    $last = $page - 2;


```

```

    echo "<a href=\"$_PHP_SELF?page=$last\">Last 10 Records</a>";
}
mysql_close($conn);
?>

```

PHP mysql backup

 June 17th, 2013  admin

Perform MySQL backup using PHP

It is always good practice to take a regular backup of your database. There are three ways you can use to take backup of your MySQL database.

- Using SQL Command through PHP.
- Using MySQL binary mysqldump through PHP.
- Using phpMyAdmin user interface.

Using SQL Command through PHP

You can execute SQL SELECT command to take a backup of any table. To take a complete database dump you will need to write separate query for separate table. Each table will be stored into separate text file.

Example:

Try out following example of using SELECT INTO OUTFILE query for creating table backup :

```

<?php
$dbhost = 'localhost:3036';
$dbuser = 'root';
$dbpass = 'rootpassword';
$conn = mysql_connect($dbhost, $dbuser, $dbpass);
if(! $conn )
{
    die('Could not connect: ' . mysql_error());
}
$table_name = "employee";
$backup_file = "/tmp/employee.sql";
$sql = "SELECT * INTO OUTFILE '$backup_file' FROM $table_name";

mysql_select_db('test_db');
$retval = mysql_query( $sql, $conn );
if(! $retval )
{
    die('Could not take data backup: ' . mysql_error());
}
echo "Backedup data successfully\n";
mysql_close($conn);
?>

```

There may be instances when you would need to restore data which you have backed up some time ago. To restore the backup you just need to run LOAD DATA INFILE query like this :

```

<?php
$dbhost = 'localhost:3036';
$dbuser = 'root';
$dbpass = 'rootpassword';

```

```

$conn = mysql_connect($dbhost, $dbuser, $dbpass);
if(! $conn )
{
    die('Could not connect: ' . mysql_error());
}
$table_name = "employee";
$backup_file = "/tmp/employee.sql";
$sql = "LOAD DATA INFILE '$backup_file' INTO TABLE
$table_name";

mysql_select_db('test_db');
$retval = mysql_query( $sql, $conn );
if(! $retval )
{
    die('Could not load data : ' . mysql_error());
}
echo "Loaded data successfully\n";
mysql_close($conn);
?>

```

Using MySQL binary mysqldump through PHP:

MySQL provides one utility **mysqldump** to perform database backup. Using this binary you can take complete database dump in a single command.

Example:

Try out following example to take your complete database dump:

```

<?php
$dbhost = 'localhost:3036';
$dbuser = 'root';
$dbpass = 'rootpassword';

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

    "test_db | gzip > $backup_file";
system($command);
?>

```

Using phpMyAdmin user interface:

If you have **phpMyAdmin** user interface available then its very easy for your to take backup of 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.

PHP ODBC Database

 June 19th, 2013  admin

ODBC is an Application Programming Interface (API) that allows you to connect to a data source (e.g. an MS Access database).

Create an ODBC Connection

With an ODBC connection, you can connect to any database, on any computer in your network, as long as an ODBC connection is available.

Here is **how to create an ODBC connection to a MS Access Database**:

1. Open the **Administrative Tools** icon in your Control Panel.
2. Double-click on the **Data Sources (ODBC)** icon inside.
3. Choose the **System DSN** tab.
4. Click on **Add** in the System DSN tab.
5. Select the **Microsoft Access Driver**. Click **Finish**.
6. In the next screen, click **Select** to locate the database.
7. Give the database a **Data Source Name (DSN)**.
8. Click **OK**.

Note that this configuration has to be done on the computer where your web site is located. If you are running Internet Information Server (IIS) on your own computer, the instructions above will work, but if your web site is located on a remote server, you have to have physical access to that server, or ask your web host to set up a DSN for you to use.

Connecting to an ODBC

The `odbc_connect()` function is used to connect to an ODBC data source. The function takes four parameters: the data source name, username, password, and an optional cursor type.

The `odbc_exec()` function is used to execute an SQL statement.

Example

The following example creates a connection to a DSN called northwind, with no username and no password. It then creates an SQL and executes it:

```
$conn=odbc_connect('northwind','');  
$sql="SELECT * FROM customers";  
$rs=odbc_exec($conn,$sql);
```

Retrieving Records

The `odbc_fetch_row()` function is used to return records from the result-set. This function returns true if it is able to return rows, otherwise false.

The function takes two parameters: the ODBC result identifier and an optional row number:

```
odbc_fetch_row($rs)
```

Retrieving Fields from a Record

The `odbc_result()` function is used to read fields from a record. This function takes two parameters: the ODBC result identifier and a field number or name.

The code line below returns the value of the first field from the record:

```
$compname=odbc_result($rs,1);
```

The code line below returns the value of a field called "CompanyName":

```
$compname=odbc_result($rs,"CompanyName");
```

Closing an ODBC Connection

The `odbc_close()` function is used to close an ODBC connection.

```
odbc_close($conn);
```

An ODBC Example

The following example shows how to first create a database connection, then a result-set, and then display the data in an HTML table.

```
<html>
<body>

<?php
$conn=odbc_connect('northwind','');
if (!$conn)
    {exit("Connection Failed: " . $conn);}
$sql="SELECT * FROM customers";
$rs=odbc_exec($conn,$sql);
if (!$rs)
    {exit("Error in SQL");}
echo "<table><tr>";
echo "<th>Companyname</th>";
echo "<th>Contactname</th></tr>";
while (odbc_fetch_row($rs))
    {
        $compname=odbc_result($rs,"CompanyName");
        $conname=odbc_result($rs,"ContactName");
        echo "<tr><td>$compname</td>";
        echo "<td>$conname</td></tr>";
    }
odbc_close($conn);
echo "</table>";
?>

</body>
</html>
```

PHP and XML

 June 21st, 2013  admin

What is XML?

XML is used to describe data and to focus on what data is. An XML file describes the structure of the data.

In XML, no tags are predefined. You must define your own tags.

XML is a markup language that looks a lot like HTML. An XML document is plain text and contains tags delimited by `<` and `>`. There are two big differences between XML and HTML:

- XML doesn't define a specific set of tags you must use.
- XML is extremely picky about document structure.

XML gives you a lot more freedom than HTML. HTML has a certain set of tags: the `<a>` `` tags surround a link, the `<p>` starts a paragraph and so on. An XML document, however, can use any tags you want. Put `<rating>` `</rating>` tags around a movie rating, `>height<` `</height>` tags around someone's height. Thus XML gives you option to device your own tags.

XML is very strict when it comes to document structure. HTML lets you play fast and loose with some opening and closing tags. BUT this is not the case with XML.

HTML list that's not valid XML:

```
<ul>
<li>Braised Sea Cucumber
<li>Baked GIBLETS with Salt
<li>Abalone with Marrow and Duck Feet
</ul>
```

This is not a valid XML document because there are no closing `` tags to match up with the three opening `` tags. Every opened tag in an XML document must be closed.

HTML list that is valid XML:

```
<ul>
<li>Braised Sea Cucumber</li>
<li>Baked GIBLETS with Salt</li>
<li>Abalone with Marrow and Duck Feet</li>
</ul>
```

Parsing an XML Document:

PHP 5's new **SimpleXML** module makes parsing an XML document, well, simple. It turns an XML document into an object that provides structured access to the XML.

To create a SimpleXML object from an XML document stored in a string, pass the string to `simplexml_load_string()`. It returns a SimpleXML object.

Try out following **example** :

```
<?php
$channel = <<<_XML_
<channel>
<title>What's For Dinner</title>
<link>http://menu.example.com/</link>
<description>Choose what to eat tonight.</description>
</channel>
_XML_

$xml = simplexml_load_string($channel);
print "The $xml->title channel is available at $xml->link. ";
print "The description is \"$xml->description\"";
?>
```

It will produce following result:

The What's For Dinner channel is available at <http://menu.example.com/>. The description is "Choose what to eat tonight."

NOTE: You can use function `simplexml_load_file(filename)` if you have XML content in a file.

For a complete detail of XML parsing function check [PHP Function Reference](#).

Generating an XML Document:

SimpleXML is good for parsing existing XML documents, but you can't use it to create a new one from scratch.

The easiest way to generate an XML document is to build a PHP array whose structure mirrors that of the XML document and then to iterate through the array, printing each element with appropriate formatting.

Example:

```
<?php
$channel = array('title' => "What's For Dinner",
                'link' => 'http://menu.example.com/',
                'description' => 'Choose what to eat tonight. ');
print "<channel>\n";
foreach ($channel as $element => $content) {
    print "<$element>";
    print htmlentities($content);
    print "</$element>\n";
}
print "</channel>";
?>
```

It will produce following result:

```
<channel>
<title>What's For Dinner</title>
<link>http://menu.example.com/</link>
<description>Choose what to eat tonight.</description>
</channel></html>
```

AJAX and PHP

 June 22nd, 2013  admin

AJAX is used to create more interactive applications.

AJAX PHP Example

The following example will demonstrate how a web page can communicate with a web server while a user type characters in an input field:

Start typing a name in the input field below:

First name:
Suggestions:

The HTML Page – Example Explained

When a user types a character in the input field above, the function "showHint()" is executed. The function is triggered by the "onkeyup" event:

```
<html>
<head>
<script>
function showHint(str)
{
if (str.length==0)
{
document.getElementById("txtHint").innerHTML="";
return;
}
if (window.XMLHttpRequest)
{// code for IE7+, Firefox, Chrome, Opera, Safari
xmlhttp=new XMLHttpRequest();
}
else
{// code for IE6, IE5
xmlhttp=new ActiveXObject("Microsoft.XMLHTTP");
}
xmlhttp.onreadystatechange=function()
{
if (xmlhttp.readyState==4 && xmlhttp.status==200)
{
document.getElementById("txtHint").innerHTML=xmlhttp.responseText;
}
}
xmlhttp.open("GET","gethint.php?q="+str,true);
xmlhttp.send();
}
</script>
</head>
<body>

<p><b>Start typing a name in the input field below:</b></p>
<form>
First name: <input type="text" onkeyup="showHint(this.value)">
</form>
<p>Suggestions: <span id="txtHint"></span></p>

</body>
</html>
```

Source code explanation:

If the input field is empty (str.length==0), the function clears the content of the txtHint placeholder and exits the function.

If the input field is not empty, the showHint() function executes the following:

- Create an XMLHttpRequest object
- Create the function to be executed when the server response is ready
- Send the request off to a file on the server
- Notice that a parameter (q) is added to the URL (with the content of the input field)

The PHP File

The page on the server called by the JavaScript above is a PHP file called "gethint.php".

The source code in "gethint.php" checks an array of names, and returns the corresponding name(s) to the browser:

```
<?php
// Fill up array with names
$a[]="Anna";
$a[]="Brittany";
$a[]="Cinderella";
$a[]="Diana";
$a[]="Eva";
$a[]="Fiona";
$a[]="Gunda";
$a[]="Hege";
$a[]="Inga";
$a[]="Johanna";
$a[]="Kitty";
$a[]="Linda";
$a[]="Nina";
$a[]="Ophelia";
$a[]="Petunia";
$a[]="Amanda";
$a[]="Raquel";
$a[]="Cindy";
$a[]="Doris";
$a[]="Eve";
$a[]="Evita";
$a[]="Sunniva";
$a[]="Tove";
$a[]="Unni";
$a[]="Violet";
$a[]="Liza";
$a[]="Elizabeth";
$a[]="Ellen";
$a[]="Wenche";
$a[]="Vicky";

//get the q parameter from URL
$q=$_GET["q"];

//lookup all hints from array if length of q>0
if (strlen($q) > 0)
{
    $hint="";
    for($i=0; $i<count($a); $i++)
    {
        if (strtolower($q)==strtolower(substr($a[$i],0,strlen($q))))
        {
            if ($hint=="")
            {
                $hint=$a[$i];
            }
        }
    }
    else
```

```

    {
        $hint=$hint." , ".$a[$i];
    }
}
}

// Set output to "no suggestion" if no hint were found
// or to the correct values
if ($hint == "")
{
    $response="no suggestion";
}
else
{
    $response=$hint;
}

//output the response
echo $response;
?>

```

Explanation: If there is any text sent from the JavaScript (`strlen($q) > 0`), the following happens:

1. Find a name matching the characters sent from the JavaScript
2. If no match were found, set the response string to "no suggestion"
3. If one or more matching names were found, set the response string to all these names
4. The response is sent to the "txtHint" placeholder

PHP and AJAX Example

 June 22nd, 2013  admin

To clearly illustrate how easy it is to access information from a database using Ajax and PHP, we are going to build MySQL queries on the fly and display the results on "ajax.html". But before we proceed, let's do ground work. Create a table using the following command.

NOTE: We are assuming you have sufficient privilege to perform following MySQL operations

```

CREATE TABLE `ajax_example` (
  `name` varchar(50) NOT NULL,
  `age` int(11) NOT NULL,
  `sex` varchar(1) NOT NULL,
  `wpm` int(11) NOT NULL,
  PRIMARY KEY (`name`)
)

```

Now dump the following data into this table using the following SQL statements

```

INSERT INTO `ajax_example` VALUES ('Jerry', 120, 'm', 20);
INSERT INTO `ajax_example` VALUES ('Regis', 75, 'm', 44);
INSERT INTO `ajax_example` VALUES ('Frank', 45, 'm', 87);
INSERT INTO `ajax_example` VALUES ('Jill', 22, 'f', 72);
INSERT INTO `ajax_example` VALUES ('Tracy', 27, 'f', 0);
INSERT INTO `ajax_example` VALUES ('Julie', 35, 'f', 90);

```

Client Side HTML file

Now let's have our client side HTML file which is ajax.html and it will have following code

```
<html>
<body>
<script language="javascript" type="text/javascript">
<!--
//Browser Support Code
function ajaxFunction(){
var ajaxRequest; // The variable that makes Ajax possible!

try{
    // Opera 8.0+, Firefox, Safari
    ajaxRequest = new XMLHttpRequest();
}catch (e){
    // Internet Explorer Browsers
    try{
        ajaxRequest = new ActiveXObject("Msxml2.XMLHTTP");
    }catch (e) {
        try{
            ajaxRequest = new ActiveXObject("Microsoft.XMLHTTP");
        }catch (e){
            // Something went wrong
            alert("Your browser broke!");
            return false;
        }
    }
}

// Create a function that will receive data
// sent from the server and will update
// div section in the same page.
ajaxRequest.onreadystatechange = function(){
    if(ajaxRequest.readyState == 4){
        var ajaxDisplay = document.getElementById('ajaxDiv');
        ajaxDisplay.innerHTML = ajaxRequest.responseText;
    }
}

// Now get the value from user and pass it to
// server script.
var age = document.getElementById('age').value;
var wpm = document.getElementById('wpm').value;
var sex = document.getElementById('sex').value;
var queryString = "?age=" + age ;
queryString += "&wpm=" + wpm + "&sex=" + sex;
ajaxRequest.open("GET", "ajax-example.php" +
    queryString, true);
ajaxRequest.send(null);
}
//-->
</script>
<form name='myForm'>
Max Age: <input type='text' id='age' /> <br />
Max WPM: <input type='text' id='wpm' />
<br />
Sex: <select id='sex'>
<option value="m">m</option>
<option value="f">f</option>
</select>
<input type='button' onclick='ajaxFunction()'
```

```

        value='Query MySQL'/>
</form>
<div id='ajaxDiv'>Your result will display here</div>
</body>
</html>

```

NOTE: The way of passing variables in the Query is according to HTTP standard and the have formA

```
URL?variable1=value1;&variable2=value2;
```

Now the above code will give you a screen as given below

NOTE: This is dummy screen and would not work

Max Age:

Max WPM:

Sex:

Your result will display here

Server Side PHP file

So now your client side script is ready. Now we have to write our server side script which will fetch age, wpm and sex from the database and will send it back to the client. Put the following code into "ajax-example.php" file

```

<?php
$dbhost = "localhost";
$dbuser = "dbusername";
$dbpass = "dbpassword";
$dbname = "dbname";
//Connect to MySQL Server
mysql_connect($dbhost, $dbuser, $dbpass);
//Select Database
mysql_select_db($dbname) or die(mysql_error());
// Retrieve data from Query String
$age = $_GET['age'];
$sex = $_GET['sex'];
$wpm = $_GET['wpm'];
// Escape User Input to help prevent SQL Injection
$age = mysql_real_escape_string($age);
$sex = mysql_real_escape_string($sex);
$wpm = mysql_real_escape_string($wpm);
//build query
$query = "SELECT * FROM ajax_example WHERE sex = '$sex'";
if(is_numeric($age))
$query .= " AND age <= $age";
if(is_numeric($wpm))
$query .= " AND wpm <= $wpm";
//Execute query
$qry_result = mysql_query($query) or die(mysql_error());

//Build Result String
$display_string = "<table>";

```

```

$display_string .= "<tr>";
$display_string .= "<th>Name</th>";
$display_string .= "<th>Age</th>";
$display_string .= "<th>Sex</th>";
$display_string .= "<th>WPM</th>";
$display_string .= "</tr>";

// Insert a new row in the table for each person returned
while($row = mysql_fetch_array($qry_result)){
$display_string .= "<tr>";
$display_string .= "<td>$row[name]</td>";
$display_string .= "<td>$row[age]</td>";
$display_string .= "<td>$row[sex]</td>";
$display_string .= "<td>$row[wpm]</td>";
$display_string .= "</tr>";
}
echo "Query: " . $query . "<br />";
$display_string .= "</table>";
echo $display_string;
?>

```

Now try by entering a valid value in "Max Age" or any other box and then click Query MySQL button.


Max Age:

Max WPM:

Sex:

Your result will display here

PHP AJAX and MySQL

 June 24th, 2013  admin

AJAX can be used for interactive communication with a **database**.

AJAX Database Example

The following example will demonstrate how a web page can **fetch information from a database with AJAX**:

Person info will be listed here...

Example Explained – The MySQL Database

The database table we use in the example above looks like this:

id	FirstName	LastName	Age	Hometown	Job
1	Peter	Griffin	41	Quahog	Brewery
2	Lois	Griffin	40	Newport	Piano Teacher
3	Joseph	Swanson	39	Quahog	Police Officer

id	FirstName	LastName	Age	Hometown	Job
4	Glenn	Quagmire	41	Quahog	Pilot

Example Explained – The HTML Page

When a user selects a user in the dropdown list above, a function called "showUser()" is executed. The function is triggered by the "onchange" event:

```
<html>
<head>
<script>
function showUser(str)
{
  if (str=="")
  {
    document.getElementById("txtHint").innerHTML="";
    return;
  }
  if (window.XMLHttpRequest)
    // code for IE7+, Firefox, Chrome, Opera, Safari
    xmlhttp=new XMLHttpRequest();
  }
  else
    // code for IE6, IE5
    xmlhttp=new ActiveXObject("Microsoft.XMLHTTP");
  }
  xmlhttp.onreadystatechange=function()
  {
    if (xmlhttp.readyState==4 && xmlhttp.status==200)
    {
      document.getElementById("txtHint").innerHTML=xmlhttp.responseText;
    }
  }
  xmlhttp.open("GET","getuser.php?q="+str,true);
  xmlhttp.send();
}
</script>
</head>
<body>

<form>
<select name="users" onchange="showUser(this.value)">
<option value="">Select a person:</option>
<option value="1">Peter Griffin</option>
<option value="2">Lois Griffin</option>
<option value="3">Glenn Quagmire</option>
<option value="4">Joseph Swanson</option>
</select>
</form>
<br>
<div id="txtHint"> <b>Person info will be listed here.</b> </div>

</body>
</html>
```

The showUser() function does the following:

- Check if a person is selected

- Create an XMLHttpRequest object
- Create the function to be executed when the server response is ready
- Send the request off to a file on the server
- Notice that a parameter (q) is added to the URL (with the content of the dropdown list)

AJAX – The PHP File

The page on the server called by the JavaScript above is a PHP file called "getuser.php".

The source code in "getuser.php" runs a query against a MySQL database, and returns the result in an HTML table:

```
<?php
$q=$_GET["q"];
$con = mysql_connect('localhost', 'peter', 'abc123');
if (!$con)
{
    die('Could not connect: ' . mysql_error());
}
mysql_select_db("ajax_demo", $con);
$sql="SELECT * FROM user WHERE id = '".$q."'";
$result = mysql_query($sql);
echo "<table border='1'>
<tr>
<th>Firstname</th>
<th>Lastname</th>
<th>Age</th>
<th>Hometown</th>
<th>Job</th>
</tr>";
while($row = mysql_fetch_array($result))
{
    echo "<tr>";
    echo "<td>" . $row['FirstName'] . "</td>";
    echo "<td>" . $row['LastName'] . "</td>";
    echo "<td>" . $row['Age'] . "</td>";
    echo "<td>" . $row['Hometown'] . "</td>";
    echo "<td>" . $row['Job'] . "</td>";
    echo "</tr>";
}
echo "</table>";
mysql_close($con);
?>
```

Explanation: When the query is sent from the JavaScript to the PHP file, the following happens:

1. PHP opens a connection to a MySQL server
2. The correct person is found
3. An HTML table is created, filled with data, and sent back to the "txtHint" placeholder

PHP – AJAX and XML

AJAX can be used for interactive communication with an XML file.

AJAX XML Example

The following example will demonstrate how a web page can fetch information from an XML file with AJAX:

CD info will be listed here...

Example Explained – The HTML Page

When a user selects a CD in the dropdown list above, a function called "showCD()" is executed. The function is triggered by the "onchange" event:

```
<html>
<head>
<script>
function showCD(str)
{
if (str=="")
{
document.getElementById("txtHint").innerHTML="";
return;
}
if (window.XMLHttpRequest)
{// code for IE7+, Firefox, Chrome, Opera, Safari
xmlhttp=new XMLHttpRequest();
}
else
{// code for IE6, IE5
xmlhttp=new ActiveXObject("Microsoft.XMLHTTP");
}
xmlhttp.onreadystatechange=function()
{
if (xmlhttp.readyState==4 && xmlhttp.status==200)
{
document.getElementById("txtHint").innerHTML=xmlhttp.responseText;
}
}
xmlhttp.open("GET","getcd.php?q="+str,true);
xmlhttp.send();
}
</script>
</head>
<body>
<form>
Select a CD:
<select name="cds" onchange="showCD(this.value)">
<option value="">Select a CD:</option>
<option value="Bob Dylan">Bob Dylan</option>
<option value="Bonnie Tyler">Bonnie Tyler</option>
<option value="Dolly Parton">Dolly Parton</option>
</select>
</form>
```



```
<div id="txtHint"> <b>CD info will be listed here...</b> </div>
</body>
</html>
```

The showCD() function does the following:

- Check if a CD is selected
- Create an XMLHttpRequest object
- Create the function to be executed when the server response is ready
- Send the request off to a file on the server
- Notice that a parameter (q) is added to the URL (with the content of the dropdown list)

The PHP File

The page on the server called by the JavaScript above is a PHP file called "getcd.php".



The PHP script loads an XML document, "[cd_catalog.xml](#)", runs a query against the XML file, and returns the result as HTML:

```
<?php
$q=$_GET["q"];
$xmlDoc = new DOMDocument();
$xmlDoc->load("cd_catalog.xml");
$x=$xmlDoc->getElementsByTagName('ARTIST');
for ($i=0; $i<=$x->length-1; $i++)
{
    //Process only element nodes
    if ($x->item($i)->nodeType==1)
    {
        if ($x->item($i)->childNodes->item(0)->nodeValue == $q)
        {
            $y=($x->item($i)->parentNode);
        }
    }
}
$cd=($y->childNodes);
for ($i=0;$i<$cd->length;$i++)
{
    //Process only element nodes
    if ($cd->item($i)->nodeType==1)
    {
        echo("<b>" . $cd->item($i)->nodeName . "</b> ");
        echo($cd->item($i)->childNodes->item(0)->nodeValue);
        echo("<br>");
    }
}
?>
```

When the CD query is sent from the JavaScript to the **PHP page**, the following happens:

1. PHP creates an XML DOM object
2. Find all <artist> elements that matches the name sent from the JavaScript
3. Output the album information (send to the "txtHint" placeholder)

PHP AJAX Live Search

 June 27th, 2013  admin

AJAX can be used to create more user-friendly and interactive searches.

AJAX Live Search

The following example will demonstrate a live search, where you get search results while you type.

Live search has many benefits compared to traditional searching:

- Results are shown as you type
- Results narrow as you continue typing
- If results become too narrow, remove characters to see a broader result

Search for a Mytoptutorials page in the input field below:

The results in the example above are found in an XML file ([links.xml](#)). To make this example small and simple, only six results are available.

AJAX Live Search Example – The HTML Page

When a user types a character in the input field above, the function "showResult()" is executed. The function is triggered by the "onkeyup" event:

```
<html>
<head>
<script>
function showResult(str)
{
if (str.length==0)
{
document.getElementById("livesearch").innerHTML="";
document.getElementById("livesearch").style.border="0px";
return;
}
if (window.XMLHttpRequest)
{// code for IE7+, Firefox, Chrome, Opera, Safari
xmlhttp=new XMLHttpRequest();
}
else
{// code for IE6, IE5
xmlhttp=new ActiveXObject("Microsoft.XMLHTTP");
}
xmlhttp.onreadystatechange=function()
{
if (xmlhttp.readyState==4 && xmlhttp.status==200)
{
document.getElementById("livesearch").innerHTML=xmlhttp.responseText;
document.getElementById("livesearch").style.border="1px solid
#A5ACB2";
}
}
}
xmlhttp.open("GET","livesearch.php?q="+str,true);
```

```
xmlHttp.send();
}
</script>
</head>
<body>
<form>
<input type="text" size="30" onkeyup="showResult(this.value)">
<div id="livesearch"> </div>
</form>
</body>
</html>
```

Source code explanation:

If the input field is empty (str.length==0), the function clears the content of the livesearch placeholder and exits the function.

If the input field is not empty, the showResult() function executes the following:

- Create an XMLHttpRequest object
- Create the function to be executed when the server response is ready
- Send the request off to a file on the server
- Notice that a parameter (q) is added to the URL (with the content of the input field)

AJAX Live Search The PHP File

The page on the server called by the **JavaScript** above is a PHP file called "livesearch.php".

The source code in "livesearch.php" searches an XML file for titles matching the search string and returns the result:

```
<?php
$xmlDoc=new DOMDocument();
$xmlDoc->load("links.xml");

$x=$xmlDoc->getElementsByTagName('link');

//get the q parameter from URL
$q=$_GET["q"];

//lookup all links from the xml file if length of q>0
if (strlen($q)>0)
{
$hint="";
for($i=0; $i<($x->length); $i++)
{
$y=$x->item($i)->getElementsByTagName('title');
$z=$x->item($i)->getElementsByTagName('url');
if ($y->item(0)->nodeType==1)
{
//find a link matching the search text
if (strstr($y->item(0)->childNodes->item(0)->nodeValue,$q))
{
if ($hint=="")
{
$hint="<a href=" .
$z->item(0)->childNodes->item(0)->nodeValue .
" target='_blank'>" .
$y->item(0)->childNodes->item(0)->nodeValue . "</a>";
}
}
}
}
```

```

else
{
$hint=$hint . "<br /><a href=" .
$z->item(0)->childNodes->item(0)->nodeValue .
"" target='_blank'>" .
$y->item(0)->childNodes->item(0)->nodeValue . "</a>";
}
}
}
}
}

// Set output to "no suggestion" if no hint were found
// or to the correct values
if ($hint=="")
{
$response="no suggestion";
}
else
{
$response=$hint;
}


//output the response
echo $response;
?>

```

If there is any text sent from the JavaScript (`strlen($q) > 0`), the following happens:

- Load an XML file into a new XML DOM object
- Loop through all `<title>` elements to find matches from the text sent from the JavaScript
- Sets the correct url and title in the `"$response"` variable. If more than one match is found, all matches are added to the variable
- If no matches are found, the `$response` variable is set to "no suggestion"

PHP example – AJAX RSS Reader

 July 4th, 2013  admin

AJAX RSS Reader

An RSS Reader is used to read RSS Feeds.

The following example will demonstrate an RSS reader, where the RSS-feed is loaded into a webpage without reloading:

RSS-feed will be listed here...

Example explain – AJAX RSS Reader HTML Page

When a user selects an RSS-feed in the dropdown list above, a function called "showRSS()" is executed. The function is triggered by the "onchange" event:

```

<html>
<head>
<script>
function showRSS(str)
{
if (str.length==0)
{

```

```

document.getElementById("rssOutput").innerHTML="";
return;
}
if (window.XMLHttpRequest)
    { // code for IE7+, Firefox, Chrome, Opera, Safari
    xmlhttp=new XMLHttpRequest();
    }
else
    { // code for IE6, IE5
    xmlhttp=new ActiveXObject("Microsoft.XMLHTTP");
    }
xmlhttp.onreadystatechange=function()
{
    if (xmlhttp.readyState==4 && xmlhttp.status==200)
    {
        document.getElementById("rssOutput").innerHTML=xmlhttp.responseText;
    }
}
xmlhttp.open("GET","getrss.php?q="+str,true);
xmlhttp.send();
}
</script>
</head>
<body>
<form>
<select onchange="showRSS(this.value)">
<option value="">Select an RSS-feed:</option>
<option value="Google">Google News</option>
<option value="MSNBC">MSNBC News</option>
</select>
</form>
<br>
<div id="rssOutput">RSS-feed will be listed here...</div>
</body>
</html>

```

The showRSS() function does the following:

- Check if an RSS-feed is selected
- Create an XMLHttpRequest object
- Create the function to be executed when the server response is ready
- Send the request off to a file on the server
- Notice that a parameter (q) is added to the URL (with the content of the dropdown list)

The AJAX RSS Reader PHP File

The page on the server called by the JavaScript above is a PHP file called "getrss.php":

```

<?php
//get the q parameter from URL
$q=$_GET["q"];

//find out which feed was selected
if($q=="Google")
{
    $xml=("http://news.google.com/news?ned=us&topic=h&output=rss");
}
elseif($q=="MSNBC")

```

```

{
$xml=("http://rss.msnbc.msn.com/id/3032091/device/rss/rss.xml");
}

$xmlDoc = new DOMDocument();
$xmlDoc->load($xml);

//get elements from "<channel>"
$channel=$xmlDoc->getElementsByTagName('channel')->item(0);
$channel_title = $channel->getElementsByTagName('title')
->item(0)->childNodes->item(0)->nodeValue;
$channel_link = $channel->getElementsByTagName('link')
->item(0)->childNodes->item(0)->nodeValue;
$channel_desc = $channel->getElementsByTagName('description')
->item(0)->childNodes->item(0)->nodeValue;

//output elements from "<channel>"
echo("<p><a href='\" . $channel_link
. '\">\" . $channel_title . \"</a>\"");
echo("<br>");
echo($channel_desc . "\"</p>");

//get and output "<item>" elements
$x=$xmlDoc->getElementsByTagName('item');
for ($i=0; $i<=2; $i++)
{
$item_title=$x->item($i)->getElementsByTagName('title')
->item(0)->childNodes->item(0)->nodeValue;
$item_link=$x->item($i)->getElementsByTagName('link')
->item(0)->childNodes->item(0)->nodeValue;
$item_desc=$x->item($i)->getElementsByTagName('description')
->item(0)->childNodes->item(0)->nodeValue;

echo ("<p><a href='\" . $item_link
. '\">\" . $item_title . "\"</a>\"");
echo ("<br>");
echo ($item_desc . "\"</p>");
}
?>

```

When a request for an **RSS feed** is sent from the **JavaScript**, the following happens:

- Check which feed was selected
- Create a new XML DOM object
- Load the RSS document in the xml variable
- Extract and output elements from the channel element
- Extract and output elements from the item elements

PHP Example – AJAX Poll

 July 5th, 2013  admin

The following example will demonstrate a poll where the result is shown without reloading.

Do you like PHP and AJAX so far?

Yes:

No:



Example Explained – AJAX Poll HTML Page

When a user choose an option above, a function called "getVote()" is executed. The function is triggered by the "onclick" event:

```
<html>
<head>
<script>
function getVote(int)
{
if (window.XMLHttpRequest)
  { // code for IE7+, Firefox, Chrome, Opera, Safari
    xmlhttp=new XMLHttpRequest();
  }
else
  { // code for IE6, IE5
    xmlhttp=new ActiveXObject("Microsoft.XMLHTTP");
  }
xmlhttp.onreadystatechange=function()
{
  if (xmlhttp.readyState==4 && xmlhttp.status==200)
  {
    document.getElementById("poll").innerHTML=xmlhttp.responseText;
  }
}
xmlhttp.open("GET","poll_vote.php?vote="+int,true);
xmlhttp.send();
}
</script>
</head>
<body>

<div id="poll">
<h3>Do you like PHP and AJAX so far?</h3>
<form>
Yes:
<input type="radio" name="vote" value="0"
onclick="getVote(this.value)">
<br>No:
<input type="radio" name="vote" value="1"
onclick="getVote(this.value)">
</form>
</div>

</body>
</html>
```

The getVote() function does the following:

- Create an XMLHttpRequest object
- Create the function to be executed when the server response is ready
- Send the request off to a file on the server
- Notice that a parameter (vote) is added to the URL (with the value of the yes or no option)

AJAX Poll PHP File

The page on the server called by the JavaScript above is a PHP file called "poll_vote.php":

```
<?php
$vote = $_REQUEST['vote'];

//get content of textfile
$filename = "poll_result.txt";
$content = file($filename);

//put content in array
$array = explode("||", $content[0]);
$yes = $array[0];
$no = $array[1];

if ($vote == 0)
{
    $yes = $yes + 1;
}
if ($vote == 1)
{
    $no = $no + 1;
}

//insert votes to txt file
$insertvote = $yes."||".$no;
$fp = fopen($filename,"w");
fputs($fp,$insertvote);
fclose($fp);
?>

<h2>Result:</h2>
<table>
<tr>
<td>Yes:</td>
<td>
'
height='20'>
<?php echo(100*round($yes/($no+$yes),2)); ?>%
</td>
</tr>
<tr>
<td>No:</td>
<td>
'
height='20'>
<?php echo(100*round($no/($no+$yes),2)); ?>%
</td>
</tr>
</table>
```

The value is sent from the JavaScript, and the following happens:

1. Get the content of the "poll_result.txt" file
2. Put the content of the file in variables and add one to the selected variable
3. Write the result to the "poll_result.txt" file
4. Output a graphical representation of the poll result

AJAX Poll Text File

The text file (poll_result.txt) is where we store the data from the poll.

It is stored like this:

```
0||0
```

The first number represents the "Yes" votes, the second number represents the "No" votes.

Note: Remember to allow your web server to edit the text file. Do **NOT** give everyone access, just the web server (PHP).

Problems & Solutions

Security settings could not be applied 1045 access denied for user mysql

Xamp

"mysql not starting in xampp error on port 3306"

The problem is that two mysql services are installed....

One service is installed by a manager for databases and the other service is installed by xampp.

Another solution might be to stop the service provided by the program manager and

start xampps service .. from Windows -> Services.. And then you wont have to changes ports.

i change the port number to 3308 from 3306 in \xampp\mysql\bin\my.ini file in a couple of places and MySQL port settings in the \xampp\php\php.ini file.but i can't start the MySQL service, it still conflict port 3306

Got to httpd.conf search 3306 and replace with 3307 then restart xampp service..

in's working...

MySQL

any thanks fellas.

Thanks to you things are swimming again and everything is rosy.

And Longneck I've left a pint of Guinness for you at the bar!

For any other fellow newbies here is the whole fix:-

```
mysql>UPDATE mysql.user
```

```
->SET Password=PASSWORD("*****")
```

```
->WHERE User="root";
```

```
mysql>FLUSH PRIVILEGES;
```

This is from Sitepoint's "Build your own Database Driven Website using PHP and MYSQL"

And then as Longneck said specify the password in the config.inc.php file for phpmyadmin.

Kind regards,

Dermot.

Hello Taralee02, please can you repeat the steps above, when you enter the following text :

```
UPDATE mysql.user
```

```
->SET Password=PASSWORD("*****")
```

```
->WHERE User="root";
```

make sure to write your password (your password should replace the stars) example:

if your password is (apple) than you have to write the command like that :

```
UPDATE mysql.user
```

```
->SET Password=PASSWORD("apple")
```

```
->WHERE User="root";
```

and after that don't forget (FLUSH PRIVILEGES;) command (step number 4)

now number 7 is very important, please repeat the steps and try again, and don't forget number 7, I'm waiting you.

I'm just to confirm you that it's the solution.

yesterday I tried to fix the problem, but the only think that was missing is the first line in MySQL command:

UPDATE mysql.user

I was writing (UPDATE mysql) : so please follow the next steps if you want to resolve your problem with (#1045 - Access denied for user 'root'@'localhost' (using password: NO):

1 : go to your WAMP icon on your PC desktop screen and LEFT CLICK to open the menu, you will see MYSQL folder, CLICK to see MYSQL CONSOLE, open it.

2: now you have DOS screen (a black screen) :

A: if you already set a password, type it

B: if you did not do this step yet, write the following red text

B1: **use mysql;** and click ENTER on your keyboard

3: now write the following red text and click ENTER :

UPDATE mysql.user

->SET Password=PASSWORD("***")**

->WHERE User="root";

don't worry about this sign (->), because for example when you write (UPDATE mysql.user) and you click ENTER on your keyboard, a new line appear with this sign (->), your command will execute when you write this sign (;) at the end of your text and click ENTER.

NOTE: replace the password ***** by your password.

4: now write the following red text and click ENTER :

FLUSH PRIVILEGES;

5: and to exit the black DOS screen now, write **exit** and click enter.

----- we are finished from MySQL now -----

6: go to WAMP folder (open your My Computer, click on C driver, and you will see WAMP folder), click on APPS folder, and than click on your PHPMYADMIN folder (e.g my folder called phpmyadmin2.11.6) and find the config.inc.php

7: open config.inc.php and find the following orange text:

\$cfg['Servers'][\$i]['password'] = ''; // MySQL password (only needed

and add your password now that you used in step number 3 like that :

[COLOR="rgb(255, 140, 0)">\$cfg['Servers'][\$i]['password'] = 'yourpasswordhere'; // MySQL password (only needed[/COLOR]

8: now save this modification, and close config.inc.php

9: go to your web browser and type the following link :

<http://localhost/phpmyadmin/>

Error Number 1045

In some case the follow error occur: The security settings could not be applied.

Error Number 1045.

Access denied for user 'root'@'localhost'

(using password: YES)

That means MySQL could not change the password. The configuration has been executed nevertheless!
You do not have to specify a password when you login to the server.

Resources

and enjoy, this is what happened with me.

NB: I wrote all this explications, because I'm also learning PHP and I found a lot on this website helped me, so I'm trying also to help you.

THE BEST

This is a solution, Error 1045 MySQL Instalation.

1. Stop MySQL from running(control panel | Administrative Tools | Services)
2. Make sure to uninstall the previous MySQL from control panel
3. remove all mysql directories and names from window explorer even those are hidden, In th case that there are some applications which are using mysql and are always on run and you can not remove the mysql dependencies from, you rename them. For example you have Ruby1.9, go head just rename it to RubyBack1.9 and rename them back once you finish with mysql installation. Or other alternative is you kill the process once you reach to step 9
4. go to firewall from control panel, click on advanced setting in the left panel. Now, click on inbound rules. Add two mysql's under name one with Domain Profile the other with Private Profile.
5. click on the one with Domain Profile you just have added.
6. click on the Properties down in the right panel
7. under scope tab on the top, click on the radio button and type 127.0.0.1
8. under Protocol and Ports tab pick for protocol tcp and type for port 3306. Make sure to click OK to save the changes.
9. restart your computer
10. Now, go to the task manager and kill any mysql or its product jobs if any is running.
11. Finally, re-install mysql-5.5.28-win32.msi or mysql-5.5.28-win64.msi accordingly.

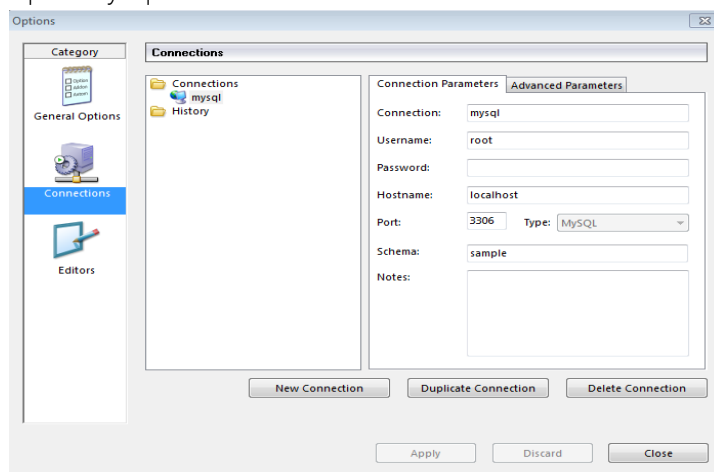
Good Luck

My Solutions

If my Sql Error

- Re – Install MySQL
- Open Command Line Mysql Admin

- Enter Password Empty (Just Press Enter)
- `mysql>UPDATE mysql.user`
`->SET Password=PASSWORD("root")`
`->WHERE User="root";`
`mysql>FLUSH PRIVILEGES;`
- Excute this
- Close Command line
- Test with New password at command line
- Open MySql Admin Window



-
- Create Connection
- Login with root
- Njoy....say Thanks to Satya