



# PHP

## Web development



Raj Parihar

✉ [rajp7jowa@gmail.com](mailto:rajp7jowa@gmail.com)

☎ 0 77 479 77 503

🌐 [bit.do/p7mca](http://bit.do/p7mca)

🌐 [bit.do/krashi](http://bit.do/krashi)

# PHP Tutorial From beginner to master

---



## Hypertext Preprocessor

PHP is a powerful tool for making dynamic and interactive Web pages.

PHP is the widely-used, free, and efficient alternative to competitors such as Microsoft's ASP.

In our PHP tutorial you will learn about PHP, and how to execute scripts on your server

## What is PHP?

- PHP stands for **PHP: Hypertext Preprocessor**
- PHP is a server-side scripting language, like ASP
- PHP scripts are executed on the server
- PHP supports many databases (MySQL, Informix, Oracle, Sybase, Solid, PostgreSQL, Generic ODBC, etc.)
- PHP is an open source software
- PHP is free to download and use

## What is a PHP File?

- PHP files can contain text, HTML tags and scripts
- PHP files are returned to the browser as plain HTML
- PHP files have a file extension of ".php"

## What is MySQL?

- MySQL is a database server
- MySQL is ideal for both small and large applications
- MySQL supports standard SQL
- MySQL compiles on a number of platforms
- MySQL is free to download and use

## PHP + MySQL

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

## Why PHP?

- PHP runs on different platforms (Windows, Linux, Unix, etc.)
- PHP is compatible with almost all servers used today (Apache, IIS, etc.)
- PHP is easy to learn and runs efficiently on the server side

## Where to Start?

To get access to a web server with PHP support, you can:

- Install Apache (or IIS) on your own server, install PHP, and MySQL
- Or find a web hosting plan with PHP and MySQL support

## Already have a web server?

If your server supports PHP you don't need to do anything.

Just create some .php files in your web directory, and the server will parse them for you. Because it is free, most web hosts offer PHP support. However, if your server does not support PHP, you must install PHP.

PHP code is executed on the server, and the plain HTML result is sent to the browser.

### Basic PHP Syntax

A PHP scripting block always starts with **<?php** and ends with **?>**. A PHP scripting block can be placed anywhere in the document.

On servers with shorthand support enabled you can start a scripting block with **<?** and end with **?>**.

For maximum compatibility, we recommend that you use the standard form (**<?php**) rather than the shorthand form.

```
<?php
?>
```

A PHP file normally contains HTML tags, just like an HTML file, and some PHP scripting code.

Below, we have an example of a simple PHP script which sends the text "Hello World" to the browser:

```
<html>
<body>

<?php
echo "Hello World";
?>

</body>
</html>
```

Each code line in PHP must end with a semicolon. The semicolon is a separator and is used to distinguish one set of instructions from another.

There are two basic statements to output text with PHP: **echo** and **print**. In the example above we have used the echo statement to output the text "Hello World".

**Note:** The file must have a .php extension. If the file has a .html extension, the PHP code will not be executed.

## Comments in PHP

In PHP, we use `//` to make a single-line comment or `/*` and `*/` to make a large comment block.

```
<html>
<body>

<?php
//This is a comment

/*
This is
a comment
block
*/
?>

</body>
</html>
```

# PHP Variables

---

A variable is used to store information.

---

## Variables in PHP

Variables are used for storing a values, like text strings, numbers or arrays.

When a variable is declared, it can be used over and over again in your script.

All variables in PHP start with a \$ sign symbol.

The correct way of declaring a variable in PHP:

```
$var_name = value;
```

New PHP programmers often forget the \$ sign at the beginning of the variable. In that case it will not work.

Let's try creating a variable containing a string, and a variable containing a number:

```
<?php  
$txt="Hello World!";  
$x=16;  
?>
```

---

## PHP is a Loosely Typed Language

In PHP, a variable does not need to be declared before adding a value to it.

In the example above, you see that you do not have to tell PHP which data type the variable is.

PHP automatically converts the variable to the correct data type, depending on its value.

In a strongly typed programming language, you have to declare (define) the type and name of the variable before using it.

In PHP, the variable is declared automatically when you use it.

---

## Naming Rules for Variables

- A variable name must start with a letter or an underscore "\_"
- A variable name can only contain alpha-numeric characters and underscores (a-z, A-Z, 0-9, and \_)

- A variable name should not contain spaces. If a variable name is more than one word, it should be separated with an underscore (\$my\_string), or with capitalization (\$myString)

## PHP String Variables

---

A string variable is used to store and manipulate text.

---

### String Variables in PHP

String variables are used for values that contains characters.

In this chapter we are going to look at the most common functions and operators used to manipulate strings in PHP.

After we create a string we can manipulate it. A string can be used directly in a function or it can be stored in a variable.

Below, the PHP script assigns the text "Hello World" to a string variable called \$txt:

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

The output of the code above will be:

```
Hello World
```

Now, lets try to use some different functions and operators to manipulate the string.

---

### The Concatenation Operator

There is only one string operator in PHP.

The concatenation operator (.) is used to put two string values together.

To concatenate two string variables together, use the concatenation operator:

```
<?php
$txt1="Hello World!";
$txt2="What a nice day!";
echo $txt1 . " " . $txt2;
?>
```

The output of the code above will be:

```
Hello World! What a nice day!
```

If we look at the code above you see that we used the concatenation operator two times. This is because we had to insert a third string (a space character), to separate the two strings.

---

## The strlen() function

The strlen() function is used to return the length of a string.

Let's find the length of a string:

```
<?php
echo strlen("Hello world!");
?>
```

The output of the code above will be:

```
12
```

The length of a string is often used in loops or other functions, when it is important to know when the string ends. (i.e. in a loop, we would want to stop the loop after the last character in the string).

---

## The strpos() function

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

If a match is found, this function will return the position of the first match. If no match is found, it will return FALSE.

Let's see if we can find the string "world" in our string:

```
<?php
echo strpos("Hello world!", "world");
?>
```

The output of the code above will be:

```
6
```

The position of the string "world" in our string is position 6. The reason that it is 6 (and not 7), is that the first position in the string is 0, and not 1.



---

# Complete PHP String Reference

For a complete reference of all string functions, go to our [complete PHP String Reference](#).

The reference contains a brief description, and examples of use, for each function!

---

## PHP Operators

---

Operators are used to operate on values.

---

### PHP Operators

This section lists the different operators used in PHP.

#### Arithmetic Operators

Operator	Description	Example	Result
+	Addition	x=2 x+2	4
-	Subtraction	x=2 5-x	3
*	Multiplication	x=4 x*5	20
/	Division	15/5 5/2	3 2.5
%	Modulus (division remainder)	5%2 10%8 10%2	1 2 0
++	Increment	x=5 x++	x=6
--	Decrement	x=5 x--	x=4

#### Assignment Operators

Operator	Example	Is The Same As
=	x=y	x=y

+=	x+=y	x=x+y
-=	x-=y	x=x-y
*=	x*=y	x=x*y
/=	x/=y	x=x/y
.=	x.=y	x=x.y
%=	x%=y	x=x%y

### Comparison Operators

Operator	Description	Example
==	is equal to	5==8 returns false
!=	is not equal	5!=8 returns true
>	is greater than	5>8 returns false
<	is less than	5<8 returns true
>=	is greater than or equal to	5>=8 returns false
<=	is less than or equal to	5<=8 returns true

### Logical Operators

Operator	Description	Example
&&	and	x=6 y=3  (x < 10 && y > 1) returns true
	or	x=6 y=3  (x==5    y==5) returns false
!	not	x=6 y=3  !(x==y) returns true

# PHP If...Else Statements

---

Conditional statements are used to perform different actions based on different conditions.

---

## Conditional Statements

Very often when you write code, you want to perform different actions for different decisions.

You can use conditional statements in your code to do this.

In PHP we have the following conditional statements:

- **if statement** - use this statement to execute some code only if a specified condition is true
  - **if...else statement** - use this statement to execute some code if a condition is true and another code if the condition is false
  - **if...elseif...else statement** - use this statement to select one of several blocks of code to be executed
  - **switch statement** - use this statement to select one of many blocks of code to be executed
- 

## The if Statement

Use the if statement to execute some code only if a specified condition is true.

### Syntax

```
if ( condition ) code to be executed if condition is true;
```

The following example will output "Have a nice weekend!" if the current day is Friday:

```
<html>
<body>

<?php
$d=date("D");
if ($d=="Fri") echo "Have a nice weekend!";
?>

</body>
</html>
```

Notice that there is no ..else.. in this syntax. You tell the browser to execute some code **only if the specified condition is true**.

---

## The if...else Statement

Use the if....else statement to execute some code if a condition is true and another code if a condition is false.

### Syntax

```
if ( condition )  
    code to be executed if condition is true;  
else  
    code to be executed if condition is false;
```

### Example

The following example will output "Have a nice weekend!" if the current day is Friday, otherwise it will output "Have a nice day!":

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

If more than one line should be executed if a condition is true/false, the lines should be enclosed within curly braces:

```
<html>  
<body>  
  
<?php  
$d=date("D");  
if ($d=="Fri")  
{  
    echo "Hello!<br />";  
    echo "Have a nice weekend!";  
    echo "See you on Monday!";  
}  
?>  
  
</body>  
</html>
```

## The if...elseif...else Statement

Use the if....elseif...else statement to select one of several blocks of code to be executed.

### Syntax

```
if ( condition )  
    code to be executed if condition is true;  
elseif ( condition )  
    code to be executed if condition is true;  
else  
    code to be executed if condition is false;
```

### Example

The following example will output "Have a nice weekend!" if the current day is Friday, and "Have a nice Sunday!" if the current day is Sunday. Otherwise it will output "Have a nice day!":

```
<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>
```

## PHP Switch Statement

---

Conditional statements are used to perform different actions based on different conditions.

---

### The PHP Switch Statement

Use the switch statement to select one of many blocks of code to be executed.

### Syntax

```
switch (n)  
{  
case label1:
```

```
    code to be executed if n=label1;
    break;
case label2:
    code to be executed if n=label2;
    break;
default:
    code to be executed if n is different from both label1 and label2;
}
```

This is how it works: First we have a single expression  $n$  (most often a variable), that is evaluated once. The value of the expression is then compared with the values for each case in the structure. If there is a match, the block of code associated with that case is executed. Use **break** to prevent the code from running into the next case automatically. The default statement is used if no match is found.

### Example

```
<html>
<body>

<?php
switch ($x)
{
case 1:
    echo "Number 1";
    break;
case 2:
    echo "Number 2";
    break;
case 3:
    echo "Number 3";
    break;
default:
    echo "No number between 1 and 3";
}
?>

</body>
</html>
```

## PHP Arrays

---

An array stores multiple values in one single variable.

---

### What is an Array?

A variable is a storage area holding a number or text. The problem is, a variable will hold only one value.

An array is a special variable, which can store multiple values in one single variable.

If you have a list of items (a list of car names, for example), storing the cars in single variables could look like this:

```
$cars1="Saab";  
$cars2="Volvo";  
$cars3="BMW";
```

However, what if you want to loop through the cars and find a specific one? And what if you had not 3 cars, but 300?

The best solution here is to use an array!

An array can hold all your variable values under a single name. And you can access the values by referring to the array name.

Each element in the array has its own index so that it can be easily accessed.

In PHP, there are three kind of arrays:

- **Numeric array** - An array with a numeric index
- **Associative array** - An array where each ID key is associated with a value
- **Multidimensional array** - An array containing one or more arrays

---

## Numeric Arrays

A numeric array stores each array element with a numeric index.

There are two methods to create a numeric array.

1. In the following example the index are automatically assigned (the index starts at 0):

```
$cars=array("Saab", "Volvo", "BMW", "Toyota");
```

2. In the following example we assign the index manually:

```
$cars[0]="Saab";  
$cars[1]="Volvo";  
$cars[2]="BMW";  
$cars[3]="Toyota";
```

### Example

In the following example you access the variable values by referring to the array name and index:

```
<?php  
$cars[0]="Saab";  
$cars[1]="Volvo";  
$cars[2]="BMW";  
$cars[3]="Toyota";
```

```
echo $cars[0] . " and " . $cars[1] . " are Swedish cars.";
?>
```

The code above will output:

```
Saab and Volvo are Swedish cars.
```

---

## Associative Arrays

An associative array, each ID key is associated with a value.

When storing data about specific named values, a numerical array is not always the best way to do it.

With associative arrays we can use the values as keys and assign values to them.

### Example 1

In this example we use an array to assign ages to the different persons:

```
$ages = array("Peter"=>32, "Quagmire"=>30, "Joe"=>34);
```

### Example 2

This example is the same as example 1, but shows a different way of creating the array:

```
$ages['Peter'] = "32";
$ages['Quagmire'] = "30";
$ages['Joe'] = "34";
```

The ID keys can be used in a script:

```
<?php
$ages['Peter'] = "32";
$ages['Quagmire'] = "30";
$ages['Joe'] = "34";

echo "Peter is " . $ages['Peter'] . " years old.";
?>
```

The code above will output:

```
Peter is 32 years old.
```



---

## Multidimensional Arrays

In a multidimensional 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.

### Example

In this example we create a multidimensional array, with automatically assigned ID keys:

```
$families = array
(
    "Griffin"=>array
    (
        "Peter",
        "Lois",
        "Megan"
    ),
    "Quagmire"=>array
    (
        "Glenn"
    ),
    "Brown"=>array
    (
        "Cleveland",
        "Loretta",
        "Junior"
    )
);
```

The array above would look like this if written to the output:

```
Array
(
    [Griffin] => Array
        (
            [0] => Peter
            [1] => Lois
            [2] => Megan
        )
    [Quagmire] => Array
        (
            [0] => Glenn
        )
    [Brown] => Array
        (
            [0] => Cleveland
            [1] => Loretta
            [2] => Junior
        )
)
```

## Example 2

Lets try displaying a single value from the array above:

```
echo "Is " . $families['Griffin'][2] .  
" a part of the Griffin family?";
```

The code above will output:

```
Is Megan a part of the Griffin family?
```

---

## Complete PHP Array Reference

For a complete reference of all array functions, go to our [complete PHP Array Reference](#).

The reference contains a brief description, and examples of use, for each function!

## PHP Looping - While Loops

---

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.

#### Syntax

```
while (condition )  
{
```

```
code to be executed    ;  
}
```

### Example

The example below defines a loop that starts with  $i=1$ . The 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 defines a loop that starts with `i=1`. It will then increment `i` with 1, and write some output. Then the condition is checked, 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
```

The for loop and the foreach loop will be explained in the next section

## PHP Looping - For Loops

---

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

---

### The 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 loop)

**Note:** Each of the 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  $i$  is less than, or equal to 5.  $i$  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

---

The real power of PHP comes from its functions.

In PHP, there are more than 700 built-in functions.

---

### PHP Built-in Functions

For a complete reference and examples of the built-in functions, please visit our [PHP Reference](#).

---

## PHP Functions

In this chapter we will show you how to create your own functions.

To keep the browser from executing a script when the page loads, you can put your script into a function.

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

You may call a function from anywhere within a page.

---

## 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

The following function has two parameters:

```
<html>
<body>

<?php
function writeName($fname,$punctuation)
{
echo $fname . " Refsnes" . $punctuation . "<br />";
}

echo "My name is ";
writeName("Kai Jim",".");
echo "My sister's name is ";
writeName("Hege","!");
echo "My brother's name is ";
writeName("Ståle","?");
?>
```



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

Output:

```
My name is Kai Jim Refsnes.
My sister's name is Hege Refsnes!
My brother's name is Ståle Refsnes?
```

---

## 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
```

---

## PHP Forms and User Input

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

</body>
</html>
```

Output could be something like this:

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

The PHP \$\_GET and \$\_POST functions will be explained in the next chapters.

---

## Form Validation

User input should be validated on the browser whenever possible (by client scripts). Browser validation is faster and reduces the server load.

You should consider server validation if the user input will be inserted into a database. A good way to validate a form on the server is to post the form to itself, instead of jumping to a different page. The

user will then get the error messages on the same page as the form. This makes it easier to discover the error.

---

The built-in `$_GET` function is used to collect values in a form with `method="get"`.

---

## The `$_GET` Function

The built-in `$_GET` function is used to collect values from a form sent with `method="get"`.

Information sent from a form with the GET method is visible to everyone (it will be displayed in the browser's address bar) and has limits on the amount of information to send (max. 100 characters).

### Example

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

When the user clicks the "Submit" button, the URL sent to the server could look something like this:

```
http://www.w3schools.com/welcome.php?fname=Peter&age=37
```

The "welcome.php" file can now use the `$_GET` function to collect form data (the names of the form fields will automatically be the keys in the `$_GET` array):

```
Welcome <?php echo $_GET["fname"]; ?>.<br />
You are <?php echo $_GET["age"]; ?> 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 large variable values; the value cannot exceed 100 characters.

---

The built-in `$_POST` function is used to collect values in a form with `method="post"`.

---

## The `$_POST` Function

The built-in `$_POST` function is used to collect values from a form sent with `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.

**Note:** However, there is an 8 Mb max size for the POST method, by default (can be changed by setting the `post_max_size` in the `php.ini` file).

### Example

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

When the user clicks the "Submit" button, the URL will look like this:

```
http://www.w3schools.com/welcome.php
```

The "welcome.php" file can now use the `$_POST` function to collect form data (the names of the form fields will automatically be the keys in the `$_POST` array):

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

---

## 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` Function

The PHP built-in `$_REQUEST` function contains the contents of both `$_GET`, `$_POST`, and `$_COOKIE`.

The `$_REQUEST` function can be used to collect form data sent with both the GET and POST methods.

### Example

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

## HTML Form with PHP

Use form controls and Post method

Front End filename : **index.html**

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <title>Registration Form</title>
  </head>
  <body>
    <form action="reg.php" method="POST">
      <h1>Registration Form </h1>
      <label for="sname">Student Name :</label>
      <input type="text" name="sname" id="sname">
      <br><br>
      <label for="smob">Mobile :</label>
      <input type="number" name="smob" id="smob">
      <br><br>
      <label for="grad">Graduate :</label>
      <input type="checkbox" name="grad" id="grad"
value="Graduate">
      <br><br>
      <label for="gender">Gender : </label>
      <input type="radio" name="gen" value="Male"> Male
      <input type="radio" name="gen" value="FeMale">FeMale
      <br><br>
      <input type="Submit" value="Register">
    </form>
  </body>
</html>
```

Back End filename :

### **reg.php**

```
<h1>Student Form Submitted Value</h1>
<?php
  //for input type text >> student name
  if($_POST["sname"] != ""){
    echo "<br> Student Name :".$_POST["sname"];
  }
  //for input type number >> mobile
  if($_POST["smob"] != ""){
    echo "<br> Mobile :".$_POST["smob"];
  }
  //for checkbox
  if(isset($_POST["grad"])){
    echo "<br> You are Graduate.";
  }else{
    echo "<br> You are not Graduate.";
  }
  if(isset($_POST["gen"])){
    echo "<br> Gender :". $_POST["gen"];
  }
  ?>
```

# MySQL

## MySQL - Introduction for MySQL

PHP के साथ MySQL ये बहुत ही मशहूर database का प्रकार है | MySQL का इस्तेमाल PHP के साथ Web server पर application बनाने के लिए किया जाता है |

MySQL ये एक open-source RDBMS(Relational DataBase Management System) है |

MySQL का abbreviation 'My Structured Query Language' है |

MySQL का इस्तेमाल XAMPP और LAMP इन मशहूर Web Servers पर भी किया जाता है |

MySQL को मशहूर websites जैसे कि, Google, Facebook, Twitter जैसी और भी कई websites इस्तेमाल करती है

MySQL ये free होता है |

MySQL का development Oracle Corp. द्वारा किया गया है |

MySQL को Windows, Linux, Mac OS जैसे Operating System पर किया जाता है |

MySQL Database में कुछ tables create किये जा सकते हैं जिसमें rows में data होता है |

यहाँ पर एक table पर 5 rows दिए गए हैं | जिसमें 5 students की 'student\_name', 'percentages' और 'grade' की information दी गयी है |

DataBase Name: college

TableName: student

id	student_name	percentages	grade
1	rakesh	48.45	C
2	raju	68.85	B
3	akshay	80	A
4	raman	75.25	A
5	deep	90.96	A

## MySQL - Connecting Database

PHP में अगर MySQL के database के table में store किये data को manipulate करना हो तो पहले database को connect करना पड़ता है | जब तक database connection successfully established नहीं होता तब तक table का data access नहीं हो पाता |

ज्यादातर database connect करने के लिए MySQLi(MySQL improved) का इस्तेमाल किया जाता है | ये PHP 5 और उसके अगले versions को support करता है |

Database connect करने के लिए mysqli\_connect() इस function का इस्तेमाल किया जाता है |

## Syntax for mysqli\_connect()

```
mysqli_connect(host_name, user_name, password, database)
```

### Example for Connecting Database

```
<?php
$server = "localhost";
$user = "root";
$password = "";
$db = "tutorial";
$conn = mysqli_connect($server, $user, $password, $db);
if($conn){
    echo "Connected successfully.";
}
else{
    echo mysqli_connect_error();
}
?>
```

## MySQL - Creating Table

MySQL के एक database में कई सारे tables create किये जाते हैं।

Database में table को create करने के लिए **CREATE TABLE table\_name** इस statement का इस्तेमाल किया जाता है।

यहाँ पर 'Student' नाम table लिया गया है और उसमें id, student\_name और grade नाम के तीन fields लिए गए हैं।

यहाँ पर INT(2), VARCHAR(50) और VARCHAR(1) ये MySQL के data types लिए गए हैं।

निचे MySQL के सभी data types लिए गए हैं।

## Data Types for MySQL

MySQL में data types तीन प्रकार में विभाजित किये गए हैं।

- Numeric Data Types
- Date and Time Data Types
- String Data Types

MySQL में कुछ ही Data Types इस्तेमाल किये जाते हैं।

**INT**: यहाँ पर integer की value signed(negative) या unsigned(positive) होती है। यहाँ पर INT की length 1 से 11 digits तक होती है। जब INT के लिए कोई length दी नहीं जाती तो '11' ये default length उसे मिल जाती है।

For eg. INT(8)

**VARCHAR**: varchar में characters की length 1 से 255 तक होती है।

For eg. VARCHAR(210)

**TINYTEXT**: tinytext में characters की length 255 तक होती है | अगर length दी नहीं जाती तो वो default वहा पर होती है |

**TEXT**: text में characters की length 65535 तक होती है | अगर length दी नहीं जाती तो वो default वहा पर होती है |

**MEDIUMTEXT**: text में characters की length 16777215 तक होती है | अगर length दी नहीं जाती तो वो default वहा पर होती है |

**LONGTEXT**: text में characters की length 4294967295 तक होती है | अगर length दी नहीं जाती तो वो default वहा पर होती है |

**DATE**: यहाँ पर date 'YYYY-MM-DD' इस format में दिए जाते हैं | यहाँ पर 1000-01-01 से लेकर 9999-12-30 तक date दी जा सकती है | For eg. 2090-12-15

**DATETIME**: यहाँ पर date और time 'YYYY-MM-DD HH:MM:SS' इस format में होती है | यहाँ पर 1000-01-01 00:00:00 से लेकर 9999-12-30 23:59:59 तक date और time दिया जा सकता है | For eg. 2090-12-15 22:58:45

## Mysql - attributes

Columns पर जब data type और उनकी length दी जाती है तब उन्हें **attributes** भी दिए जाते हैं |

**AUTO INCREMENT**: यहाँ पर हर एक बार 1-1 से value increase की जाती है |

**NOT NULL**: यहाँ पर null value allowed नहीं होती | हर record पर कुछ ना कुछ value होनी चाहिए |

**UNSIGNED**: यहा पर सिर्फ positive value और zero allowed होता है |

**DEFAULT** default\_value : यहाँ पर Default value set की जा सकती है | For eg. DEFAULT 0

**PRIMARY KEY**: यहाँ पर एक value दोबारा नहीं दी जा सकती | यहाँ पर AUTO\_INCREMENT इस attribute से duplicate value create नहीं होती

### Example for Creating Table using MySQLi

```
<?php
$server = "localhost";
$user = "root";
$password = "";
$db = "info";
$conn = mysqli_connect($server, $user, $password, $db);
if($conn){
    echo "Connected successfully.";
}
else{
    echo "Connection failed : ".mysqli_connect_error();
}
$tb_create = "CREATE TABLE student(
    id INT(2) AUTO_INCREMENT PRIMARY KEY,
    student_name VARCHAR(50) NOT NULL,
    percentages FLOAT NOT NULL,
    grade VARCHAR(1) NOT NULL
)";
if (mysqli_query($conn, $tb_create)) {
    echo "Table Created successfully.";
} else {
    echo "Table Creation failed : " . mysqli_error($conn);
}
mysqli_close($conn);
?>
```



## MySQL - Deleting Table

Table को delete करने के लिए '**DROP TABLE table\_name**' statement का इस्तेमाल किया जाता है |

Example for Deleting Table using MySQLi

```
<?php
$server = "localhost";
$user = "root";
$password = "";
$db = "info";
$conn = mysqli_connect($server, $user, $password, $db);
if($conn){
    echo "Connected successfully.";
}
else{
    echo "Connection failed : ".mysqli_connect_error();
}
$tb_delete = "DROP TABLE student";
if (mysqli_query($conn, $tb_delete)) {
    echo "Table Deleted successfully.";
} else {
    echo "Could not delete table : " . mysqli_error($conn);
}
mysqli_close($conn);
?>
```

## MySQL - Insert Records in Table

जब Table create जाता है तब उसमें records insert करने के लिए 'INSERT INTO table\_name' इस statement का इस्तेमाल किया जाता है |

records को insert इस प्रकार से किया जाता है |

```
INSERT INTO table_name (column 1, column 2, ..., column n)
VALUES('value 1', 'value 2', ..., 'value n')
```

Table में records insert करने के लिए,

columns को double("") या single(") quotes के अन्दर लिखा नहीं जाता |

जब Numeric data type values लिखी जाती है तब उन्हें double या single quotes में नहीं लिखा जाता |

string values को double या single quotes दिया जाता है |

NULL value को double या single quotes के अन्दर नहीं लिखा जाता |

## Example for Inserting Record in Table using MySQLi

```
<?php
$server = "localhost";
$user = "root";
$password = "";
$db = "info";
$conn = mysqli_connect($server, $user, $password, $db);
if($conn){
    echo "Connected successfully.";
}
else{
    echo mysqli_connect_error();
}
$tb_insert = "INSERT INTO student(student_name, percentages, grade)
VALUES ('Rakesh', '86.45', 'A')";
if (mysqli_query($conn, $tb_insert)) {
    echo "Record inserted successfully.";
} else {
    echo "Error inserting record : " . mysqli_error($conn);
}
mysqli_close($conn);
?>
```

### MySQL - Select Record

जब Table में records insert किये जाते हैं तब उस records को retrieve करने के लिए 'SELECT' statement का इस्तेमाल किया जाता है |

यहाँ पर table के single column से records retrieve करने के लिए इस syntax का इस्तेमाल किया जाता है |

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

यहाँ पर table के सभी columns से records retrieve करने के लिए इस syntax का इस्तेमाल किया जाता है |

```
SELECT * FROM table_name
```

यहाँ पर इस table को 'SELECT' statement द्वारा records retrieve किया जाएगा |

Database Name : info Table Name : student			
id	student_name	percentages	grade
1	Rakesh	48.45	C
2	Raju	68.85	B
3	Akshay	80	A
4	Raman	75.25	A
5	Deep	90.96	A

## Example for SELECT all COLUMNS using MySQLi

निचे दिए हुए script पर `mysqli_query()` function पर सामान्यतः दो parameters दिए जाते हैं | एक parameter पर connection दिया गया है और दूसरे पर MySQL का statement का string दिया गया है | इस function को `$result` पर store किया गया है |

`mysqli_fetch_row()` ये function number of rows; return करता है | script पर अगर number of rows 0 से अधिक होंगे तभी if का statement execute होगा |

जब number of rows 0 से अधिक होते हैं तब दिए हुए while condition पर `mysqli_fetch_assoc()` इस function में सभी results; associative array में loop के साथ iterate किये जाते हैं और जब rows खत्म हो जाते हैं | तब ये function NULL return करता है |

```
<?php
$conn = mysqli_connect("localhost", "root", "", "info");
if($conn){
    echo "Connected successfully.<br />";
}
else{
    echo "Connection failed : ".mysqli_connect_error();
}
$select = "SELECT * FROM student";
$result = mysqli_query($conn, $select);
echo "<table>
<tr>
<th>id</th>
<th>Student Name</th>
<th>Percentages</th>
<th>Grade</th>
</tr>";
if(mysqli_num_rows($result) > 0){
    while($rows = mysqli_fetch_assoc($result)){
        echo "<tr>
<td>".$rows['id']. "</td>
<td>".$rows['student_name']. "</td>
<td>".$rows['percentages']. "</td>
<td>".$rows['grade']. "</td>
</tr>";
    }
    echo "</table>";
}
else{
    echo "rows not found in table";
}
mysqli_close($conn);
?>
```

## MySQL - Update Record

MySQL के table के records को update करने के लिए UPDATE Statement का इस्तेमाल किया जाता है | Update statement से दिए हुए rows में कुछ बदल किये जाते हैं |

```
UPDATE table_name SET column 1 = newvalue 1, column 2 = newvalue 2,
...,column n = newvalue n
WHERE column = value
```

यहाँ पर पहले row का student\_name और percentages इस columns के records update किये जायेंगे |

## MySQL - Records Order By

ORDER BY clause का इस्तेमाल SELECT statement के साथ किया जाता है | यहाँ पर Table के records को column के जरिये ascending(ASC) और descending(DESC) order से sort किया जाता है |

Note : अगर ASC या DESC keyword का इस्तेमाल नहीं किया जाता तो ASC default set हो जाता है |

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

### Example for Sorting Records with ORDER BY clause using MySQLi

Program में 'grade' column से Ascending Order से Records को sort किया गया है |

```
<?php
$conn = mysqli_connect("localhost", "root", "", "info");
if($conn){
    echo "Connected successfully.<br />";
}
else{
    echo "Connection failed : ".mysqli_connect_error();
}
$select = "SELECT * FROM student ORDER BY grade ASC";
$result = mysqli_query($conn, $select);
echo "<table>
<tr>
<th>id</th>
<th>Student Name</th>
<th>Percentages</th>
<th>Grade</th>
</tr>";
if(mysqli_num_rows($result) > 0){
    while($rows = mysqli_fetch_assoc($result)){
        echo "<tr>
        <td>".$rows['id']. "</td>
        <td>".$rows['student_name']. "</td>
        <td>".$rows['percentages']. "</td>
        <td>".$rows['grade']. "</td>
        </tr>";
    }
    echo "</table>";
}
else{
    echo "rows not found in table";
}
mysqli_close($conn);
?>
```

## MySQL - Delete Record

जब row या record को delete करना हो तो DELETE table\_name statement के साथ WHERE clause का इस्तेमाल किया जाता है।

```
DELETE FROM table_name  
WHERE column_name=some_value
```

Note : जब DELETE statement WHERE clause के साथ लिखा नहीं जाता तो table के सारे rows delete हो जाते हैं।

Example for DELETE statement using MySQLi

```
<?php  
$server = "localhost";  
$user = "root";  
$password = "";  
$db = "info";  
$conn = mysqli_connect($server, $user, $password, $db);  
if($conn){  
    echo "Connected successfully."  
}  
else{  
    echo "Connection failed : ".mysqli_connect_error();  
}  
$tb_update = "DELETE FROM student WHERE id=2";  
if (mysqli_query($conn, $tb_update)) {  
    echo "Some row deleted successfully."  
} else {  
    echo "Row deleting failed : " . mysqli_error($conn);  
}  
mysqli_close($conn);  
?>
```

