PHP Session

PHP session is used to store and pass information from one page to another temporarily (until user close the website).

PHP session technique is widely used in shopping websites where we need to store and pass cart information e.g. username, product code, product name, product price etc from one page to another.

PHP session creates unique user id for each browser to recognize the user and avoid conflict between multiple browsers.



PHP session\_start() function

PHP session\_start() function is used to start the session. It starts a new or resumes existing session. It returns existing session if session is created already. If session is not available, it creates and returns new session.

**Syntax**

1. bool session\_start ( void )

**Example**

1. session\_start();

PHP $\_SESSION

PHP $\_SESSION is an associative array that contains all session variables. It is used to set and get session variable values.

**Example: Store information**

1. $\_SESSION["user"] = "Sachin";

**Example: Get information**

1. echo $\_SESSION["user"];

PHP Session Example

*File: session1.php*

1. <?php
2. session\_start();
3. ?>
4. <html>
5. <body>
6. <?php
7. $\_SESSION["user"] = "Sachin";
8. echo "Session information are set successfully.<br/>";
9. ?>
10. <a href="session2.php">Visit next page</a>
11. </body>
12. </html>

*File: session2.php*

1. <?php
2. session\_start();
3. ?>
4. <html>
5. <body>
6. <?php
7. echo "User is: ".$\_SESSION["user"];
8. ?>
9. </body>
10. </html>

PHP Session Counter Example

*File: sessioncounter.php*

1. <?php
2. session\_start();
4. **if** (!isset($\_SESSION['counter'])) {
5. $\_SESSION['counter'] = 1;
6. } **else** {
7. $\_SESSION['counter']++;
8. }
9. echo ("Page Views: ".$\_SESSION['counter']);
10. ?>

PHP Destroying Session

PHP session\_destroy() function is used to destroy all session variables completely.

*File: session3.php*

1. <?php
2. session\_start();
3. session\_destroy();
4. ?>

# Get and Post Methods in PHP

PHP provides two methods through which a client (browser) can send information to the server. These methods are given below, and discussed in detail:

1. GET method
2. POST method

Get and Post methods are the [HTTP](https://www.javatpoint.com/http-full-form) request methods used inside the <form> tag to send form data to the server.

[HTTP](https://www.javatpoint.com/http-tutorial) protocol enables the communication between the client and the server where a browser can be the client, and an application running on a computer system that hosts your website can be the server.

## GET method

The **GET** method is used to submit the [HTML form](https://www.javatpoint.com/html-form) data. This data is collected by the predefined **$\_GET variable** for processing.

[](https://campaign.adpushup.com/get-started/?utm_source=banner&utm_campaign=growth_hack)

The information sent from an [HTML](https://www.javatpoint.com/html-tutorial) form using the GET method is visible to everyone in the browser's address bar, which means that all the variable names and their values will be displayed in the URL. Therefore, the get method is not secured to send sensitive information.

**For Example**

1. localhost/gettest.php?username=Harry&bloodgroup=AB+

The **bold** part in the above [URL](https://www.javatpoint.com/url-full-form) is the variables name and italic part contains the values for their corresponding variable.

#### **Note that only a limited amount of information can be sent using the GET method.**

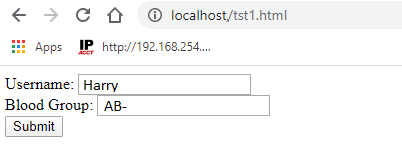
With the help of an example, let's understand how the GET method works-

### **Example**

The below code will display an HTML form containing two input fields and a submit button. In this HTML form, we used the method = "get" to submit the form data.

*file: test1.html*

1. <html>
2. <body>
4. <form action = "gettest.php" method = "GET">
5. Username: <input type = "text" name = "username" /> <br>
6. Blood Group: <input type = "text" name = "bloodgroup" /> <br>
7. <input type = "submit" />
8. </form>
10. </body>
11. </html>



Create gettest.php file, which will accept the data sent by HTML form.

*file: gettest.php*

1. <html>
2. <body>
4. Welcome <?php echo $\_GET["username"]; ?> </br>
5. Your blood group is: <?php echo $\_GET["bloodgroup"]; ?>
7. </body>
8. </html>

When the user will click on **Submit** button after filling the form, the URL sent to the server could look something like this:

**localhost/gettest.php?username=Harry&bloodgroup=AB-**

The output will look like the below output:

Welcome Harry

Your blood group is: AB-

### **Advantages of GET method (method = "get")**

* You can bookmark the page with the specific query string because the data sent by the GET method is displayed in URL.
* GET requests can be cached.
* GET requests are always remained in the browser history.

### **Disadvantages of GET Method**

* The GET method should not be used while sending any sensitive information.
* A limited amount of data can be sent using method = "get". This limit should not exceed 2048 characters.
* For security reasons, never use the GET method to send highly sensitive information like username and password, because it shows them in the URL.
* The GET method cannot be used to send binary data (such as images or word documents) to the server.

## POST method

Similar to the GET method, the **POST** method is also used to submit the HTML form data. But the data submitted by this method is collected by the predefined superglobal variable **$\_POST** instead of $\_GET.

Unlike the GET method, it does not have a limit on the amount of information to be sent. The information sent from an HTML form using the POST method is not visible to anyone.

**For Example**

1. localhost/posttest.php

#### **Note that the "post" method is more secure than the "get" method because the data sent using the POST method is not visible to user.**

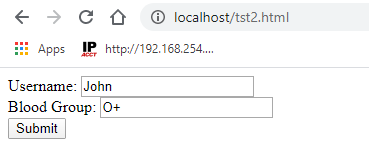
With the help of an example, let's understand how the POST method works-

### **Example**

The below code will display an HTML form containing two input fields and a submit button. In this HTML form, we used the method = "post" to submit the form data.

*file: test2.html*

1. <html>
2. <body>
4. <form action = "posttest.php" method = "post">
5. Username: <input type = "text" name = "username" /> <br>
6. Blood Group: <input type = "text" name = "bloodgroup" /> <br>
7. <input type = "submit" />
8. </form>
10. </body>
11. </html>



Now create **posttest.php** file to accept the data sent by HTML form.

*file: posttest.php*

1. <html>
2. <body>
4. Welcome <?php echo $\_POST["username"]; ?> </br>
5. Your blood group is: <?php echo $\_POST["bloodgroup"]; ?>
7. </body>
8. </html>

When the user will click on **Submit** button after filling the form, the URL sent to the server could look something like this:

**localhost/posttest.php**

The output will look like the below output:

Welcome Harry

Your blood group is: O+

### **Advantages of POST method (method = "post")**

* The POST method is useful for sending any sensitive information because the information sent using the POST method is not visible to anyone.
* There is no limitation on size of data to be sent using the POST Method. You can send a large amount of information using this method.
* Binary and ASCII data can also be sent using the POST method.
* Data security depends on the HTTP protocol because the information sent using the POST method goes through the HTTP header. By using secure HTTP, you can ensure that your data is safe.

### **Disadvantages of POST Method**

* POST requests do not cache.
* POST requests never remain in the browser history.
* It is not possible to bookmark the page because the variables are not displayed in URL.

## $\_REQUEST variable

The **$\_REQUEST** variable is a **superglobal variable**, which can hold the content of both $\_GET and $\_POST variable. In other words, the [PHP](https://www.javatpoint.com/php-tutorial) $\_REQUEST variable is used to collect the form data sent by either GET or POST methods. It can also collect the data for $\_COOKIE variable because it is not a method-specific variable.

# Difference between get and post method in PHP

The browser client can deliver data to the web server in two ways.

* The GET Method
* The POST Method

Before sending the data, the browser encrypts it using a method known as URL encoding. Name/value pairs are connected using equal signs in this system, while distinct pairings are separated by an ampersand. Spaces are substituted with the + symbol, and any other nonalphanumeric characters are replaced with hexadecimal values. After the data has been encoded, it is transferred to the server.

Now let us look at the individual methods and understand the working of these methods and understand them.

### **Get Method:**

A

The GET method appends encoded user information to the page request. The? character separates the page from the encoded content.

* The GET method generates a lengthy string that displays in your server logs and in the Location: box of your browser.
* The GET method can only send up to 1024 characters.
* If you need to send a password or other sensitive information to the server, never utilise the GET method.
* GET cannot be used to deliver binary data to the server, such as photos or word documents.
* The QUERY STRING environment variable can be used to obtain the data sent by the GET method.
* PHP offers an associative array $\_GET for accessing all information given via the GET method.

Code:

1. **<?php**
2. if( $\_GET["name"] || $\_GET["age"] ) {
3. echo "Welcome ". $\_GET['name']. "**<br** **/>**";
4. echo "You are ". $\_GET['age']. " years old.";
6. exit();
7. }
8. **?>**
9. **<html>**
10. **<body>**
12. **<form** action = "<?php $\_PHP\_SELF ?>" method = "GET"**>**
13. Name: **<input** type = "text" name = "name" **/>**
14. Age: **<input** type = "text" name = "age" **/>**
15. **<input** type = "submit" **/>**
16. **</form>**
18. **</body>**
19. **</html>**

### **Post Method:**

The POST method sends data to the server using HTTP headers. The information is encoded in the same way as specified for the GET method and included in a header named QUERY STRING.

* The POST method has no limit on the amount of data that can be delivered.
* The POST method allows you to submit both ASCII and binary data.
* Because data submitted using the POST method is routed through the HTTP header, security is dependent on the HTTP protocol. You may ensure the security of your data by utilising Secure HTTP.
* PHP offers an associative array $\_POST for accessing all information given via the POST method.

Code:

1. **<?php**
2. if( $\_POST["name"] || $\_POST["age"] ) {
3. if (preg\_match("/[^A-Za-z'-]/",$\_POST['name'] )) {
4. die ("invalid name and name should be alpha");
5. }
6. echo "Welcome ". $\_POST['name']. "**<br** **/>**";
7. echo "You are ". $\_POST['age']. " years old.";
9. exit();
10. }
11. **?>**
12. **<html>**
13. **<body>**
15. **<form** action = "<?php $\_PHP\_SELF ?>" method = "POST"**>**
16. Name: **<input** type = "text" name = "name" **/>**
17. Age: **<input** type = "text" name = "age" **/>**
18. **<input** type = "submit" **/>**
19. **</form>**
21. **</body>**
22. **</html>**

PHP Form Handling

We can create and use forms in PHP. To get form data, we need to use PHP superglobals $\_GET and $\_POST.

The form request may be get or post. To retrieve data from get request, we need to use $\_GET, for post request $\_POST.

PHP Get Form

Get request is the default form request. The data passed through get request is visible on the URL browser so it is not secured. You can send limited amount of data through get request.

Let's see a simple example to receive data from get request in PHP.

**00:17/04:28**

69.5M

1.1K

Hello Java Program for Beginners

*File: form1.html*

1. <form action="welcome.php" method="get">
2. Name: <input type="text" name="name"/>
3. <input type="submit" value="visit"/>
4. </form>

*File: welcome.php*

1. <?php
2. $name=$\_GET["name"];//receiving name field value in $name variable
3. echo "Welcome, $name";
4. ?>

PHP Post Form

Post request is widely used to submit form that have large amount of data such as file upload, image upload, login form, registration form etc.

The data passed through post request is not visible on the URL browser so it is secured. You can send large amount of data through post request.

Let's see a simple example to receive data from post request in PHP.

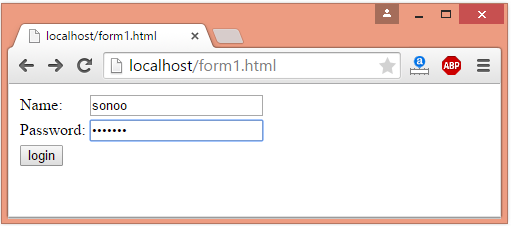
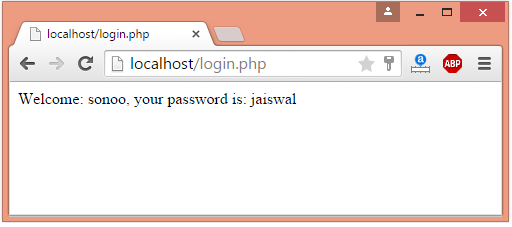
*File: form1.html*

1. <form action="login.php" method="post">
2. <table>
3. <tr><td>Name:</td><td> <input type="text" name="name"/></td></tr>
4. <tr><td>Password:</td><td> <input type="password" name="password"/></td></tr>
5. <tr><td colspan="2"><input type="submit" value="login"/>  </td></tr>
6. </table>
7. </form>

*File: login.php*

1. <?php
2. $name=$\_POST["name"];//receiving name field value in $name variable
3. $password=$\_POST["password"];//receiving password field value in $password variable
5. echo "Welcome: $name, your password is: $password";
6. ?>

Output:

# PHP Include and Require

PHP allows us to create various elements and functions, which are used several times in many pages. It takes much time to script these functions in multiple pages. Therefore, use the concept of **file inclusion** that helps to include files in various programs and saves the effort of writing code multiple times.

"PHP allows you to include file so that a page content can be reused many times. It is very helpful to include files when you want to apply the same HTML or PHP code to multiple pages of a website." There are two ways to include file in PHP.

1. include
2. require

**Both include and require are identical to each other, except failure.**

* **include** only generates a warning, i.e., E\_WARNING, and continue the execution of the script.
* **require** generates a fatal error, i.e., E\_COMPILE\_ERROR, and stop the execution of the script.

## Advantage

**Code Reusability:** By the help of include and require construct, we can reuse HTML code or PHP script in many PHP scripts.

Play Videox[](https://campaign.adpushup.com/get-started/?utm_source=banner&utm_campaign=growth_hack)

**Easy editable:** If we want to change anything in webpages, edit the source file included in all webpage rather than editing in all the files separately.

## PHP include

PHP include is used to include a file on the basis of given path. You may use a relative or absolute path of the file.

### **Syntax**

There are two syntaxes available for include:

1. **include** 'filename ';
2. Or
3. **include** ('filename');

### **Examples**

Let's see a simple PHP include example.

*File: menu.html*

1. <a href="http://www.javatpoint.com">Home</a> |
2. <a href="http://www.javatpoint.com/php-tutorial">PHP</a> |
3. <a href="http://www.javatpoint.com/java-tutorial">Java</a> |
4. <a href="http://www.javatpoint.com/html-tutorial">HTML</a>

*File: include1.php*

1. <?php **include**("menu.html"); ?>
2. <h1>This is Main Page</h1>

**Output:**

[Home](https://www.javatpoint.com/)

|

[PHP](https://www.javatpoint.com/php-tutorial)

|

[Java](https://www.javatpoint.com/java-tutorial)

|

[HTML](https://www.javatpoint.com/html-tutorial)

## This is Main Page

## PHP require

PHP require is similar to include, which is also used to include files. The only difference is that it stops the execution of script if the file is not found whereas include doesn't.

### **Syntax**

There are two syntaxes available for require:

1. **require** 'filename';
2. Or
3. **require** ('filename');

### **Examples**

Let's see a simple PHP require example.

*File: menu.html*

1. <a href="http://www.javatpoint.com">Home</a> |
2. <a href="http://www.javatpoint.com/php-tutorial">PHP</a> |
3. <a href="http://www.javatpoint.com/java-tutorial">Java</a> |
4. <a href="http://www.javatpoint.com/html-tutorial">HTML</a>

*File: require1.php*

1. <?php **require**("menu.html"); ?>
2. <h1>This is Main Page</h1>

**Output:**

[Home](https://www.javatpoint.com/)

|

[PHP](https://www.javatpoint.com/php-tutorial)

|

[Java](https://www.javatpoint.com/java-tutorial)

|

[HTML](https://www.javatpoint.com/html-tutorial)

## This is Main Page

## PHP include vs PHP require

Both include and require are same. But if the file is missing or inclusion fails, **include** allows the script to continue but **require** halts the script producing a fatal E\_COMPILE\_ERROR level error.

Let's understand the difference with the help of example:

### **Example**

*include.php*

1. <?php
2. //include welcome.php file
3. **include**("welcome.php");
4. echo "The welcome file is included.";
5. ?>

**Output:**

The **welcome.php** file is not available in the same directory, which we have included. So, it will produce a warning about that missing file but also display the output.

Warning: include(welcome.php): failed to open stream: No such file or directory in C:\xampp\htdocs\program\include.php on line 3

Warning: include(): Failed opening 'welcome.php' for inclusion (include\_path='C:\xampp\php\PEAR') in C:\xampp\htdocs\program\include.php on line 3

The welcome file is included.

*require.php*

1. <?php
2. echo "HELLO";
3. //require welcome.php file
4. **require**("welcome.php");
5. echo "The welcome file is required.";
6. ?>

**Output:**

In case of require() if the file (**welcome.php**) is not found in the same directory. The require() will generate a **fatal error** and stop the execution of the script, as you can see in the below output.

HELLO

Warning: require(Welcome.php): failed to open stream: No such file or directory in C:\xampp\htdocs\program\include.php on line 3

Fatal error: require(): Failed opening required 'Welcome.php' (include\_path='C:\xampp\php\PEAR') in C:\xampp\htdocs\program\include.php on line 3

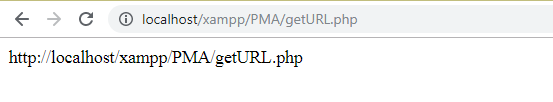
# How to get current page URL in PHP?

To get the current page URL, PHP provides a superglobal variable **$\_SERVER**. The $\_SERVER is a built-in variable of PHP, which is used to get the current page URL. It is a superglobal variable, means it is always available in all scope.

If we want the full URL of the page, then we'll need to check the protocol (or scheme name), whether it is https or http. See the example below:

1. <?php
2. **if**(isset($\_SERVER['HTTPS']) && $\_SERVER['HTTPS'] === 'on')
3. $url = "https://";
4. **else**
5. $url = "http://";
6. // Append the host(domain name, ip) to the URL.
7. $url.= $\_SERVER['HTTP\_HOST'];
9. // Append the requested resource location to the URL
10. $url.= $\_SERVER['REQUEST\_URI'];
12. echo $url;
13. ?>

**Output**



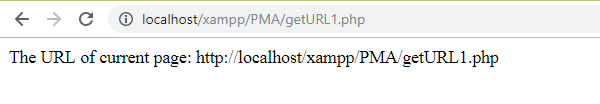
#### **Note: The isset() function is used here to check whether HTTPS is enabled or not. It checks whether a variable exists or not.**

Or, we can also get the full URL of current page using another way given in the next example.

[](https://campaign.adpushup.com/get-started/?utm_source=banner&utm_campaign=growth_hack)

1. <?php
2. $protocol = ((!empty**empty**($\_SERVER['HTTPS']) && $\_SERVER['HTTPS'] != 'off') || $\_SERVER['SERVER\_PORT'] == 443) ? "https://" : "http://";
3. $CurPageURL = $protocol . $\_SERVER['HTTP\_HOST'] . $\_SERVER['REQUEST\_URI'];
4. echo "The URL of current page: ".$CurPageURL;
5. ?>

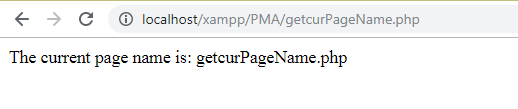
**Output**



To get only name of the current page opened at browser, see the below example:

1. <?php
2. $curPageName = substr($\_SERVER["SCRIPT\_NAME"],strrpos($\_SERVER["SCRIPT\_NAME"],"/")+1);
3. echo "The current page name is: ".$curPageName;
4. echo "</br>";
5. ?>

**Output**



PHP MySQL Connect

Since PHP 5.5, **mysql\_connect()** extension is *deprecated*. Now it is recommended to use one of the 2 alternatives.

* **mysqli\_connect()**
* **PDO::\_\_construct()**

PHP mysqli\_connect()

PHP **mysqli\_connect() function** is used to connect with MySQL database. It returns *resource* if connection is established or *null*.

**Syntax**

1. resource mysqli\_connect (server, username, password)

PHP mysqli\_close()

PHP **mysqli\_close() function** is used to disconnect with MySQL database. It returns *true* if connection is closed or *false*.

**Syntax**

1. bool mysqli\_close(resource $resource\_link)

PHP MySQL Connect Example

**Example**

1. <?php
2. $host = 'localhost:3306';
3. $user = '';
4. $pass = '';
5. $conn = mysqli\_connect($host, $user, $pass);
6. **if**(! $conn )
7. {
8. **die**('Could not connect: ' . mysqli\_error());
9. }
10. echo 'Connected successfully';
11. mysqli\_close($conn);
12. ?>

Output:

Connected successfully

# PHP MySQL Login System

In this topic, we will learn how to create a PHP MySQL Login System with the help of PHP and MySQL database. There are few steps given for creating a login system with MySQL database.

Before creating the login system first, we need to know about the pre-requisites to create the login module.

## Requirements

* We should have knowledge of HTML, CSS, PHP and MySQL for creating the login system.
* Text Editor - For writing the code. We can use any text editor such as Notepad, Notepad++, Dreamweaver, etc.
* XAMPP - XAMPP is a cross-platform software, which stands for Cross-platform(X) Apache server (A), MySQL (M), PHP (P), Perl (P). XAMPP is a complete software package, so, we don't need to install all these separately.

## Environment Setup

Now, we need to start the webserver and create the files for the login system. There are few steps that are given below to setup the environment.

* Open the XAMPP Control Panel.
* Start the Apache server by clicking on the Start button.
* Start the MySQL by clicking on the Start button.
* Create all the files needed for login.
* Create login table in the database using phpMyAdmin in XAMPP.

Now, we will create four files here for the login system.

Play Videox[](https://campaign.adpushup.com/get-started/?utm_source=banner&utm_campaign=growth_hack)

1. **index.html -** This file is created for the GUI view of the login page and empty field validation.
2. **style.css -** This file is created for the attractive view of the login form.
3. **connection.php -** Connection file contains the connection code for database connectivity.
4. **authentication.php -** This file validates the form data with the database which is submitted by the user.

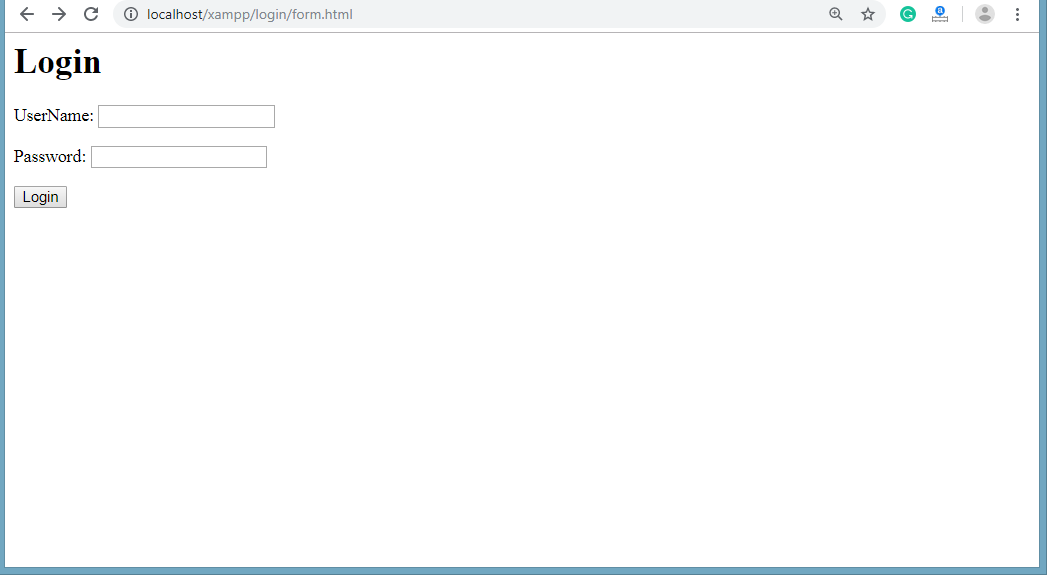
Save all these files in the htdocs folder inside the Xampp installation folder. The detailed description and code for these files are discussed below.

### **index.html**

First, we need to design the login form for the website user to interact with it. This login form is created using html and also contains the empty field validation, which is written in JavaScript. The code for the index.html file is given below:

1. **<html>**
2. **<head>**
3. **<title>**PHP login system**</title>**
4. // insert style.css file inside index.html
5. **<link** rel = "stylesheet" type = "text/css" href = "style.css"**>**
6. **</head>**
7. **<body>**
8. **<div** id = "frm"**>**
9. **<h1>**Login**</h1>**
10. **<form** name="f1" action = "authentication.php" onsubmit = "return validation()" method = "POST"**>**
11. **<p>**
12. **<label>** UserName: **</label>**
13. **<input** type = "text" id ="user" name  = "user" **/>**
14. **</p>**
15. **<p>**
16. **<label>** Password: **</label>**
17. **<input** type = "password" id ="pass" name  = "pass" **/>**
18. **</p>**
19. **<p>**
20. **<input** type =  "submit" id = "btn" value = "Login" **/>**
21. **</p>**
22. **</form>**
23. **</div>**
24. // validation for empty field
25. **<script>**
26. function validation()
27. {
28. var id=document.f1.user.value;
29. var ps=document.f1.pass.value;
30. if(id.length=="" && ps.length=="") {
31. alert("User Name and Password fields are empty");
32. return false;
33. }
34. else
35. {
36. if(id.length=="") {
37. alert("User Name is empty");
38. return false;
39. }
40. if (ps.length=="") {
41. alert("Password field is empty");
42. return false;
43. }
44. }
45. }
46. **</script>**
47. **</body>**
48. **</html>**

After executing the above code on the browser, the login page will appear as below if it does not contain style.css file.

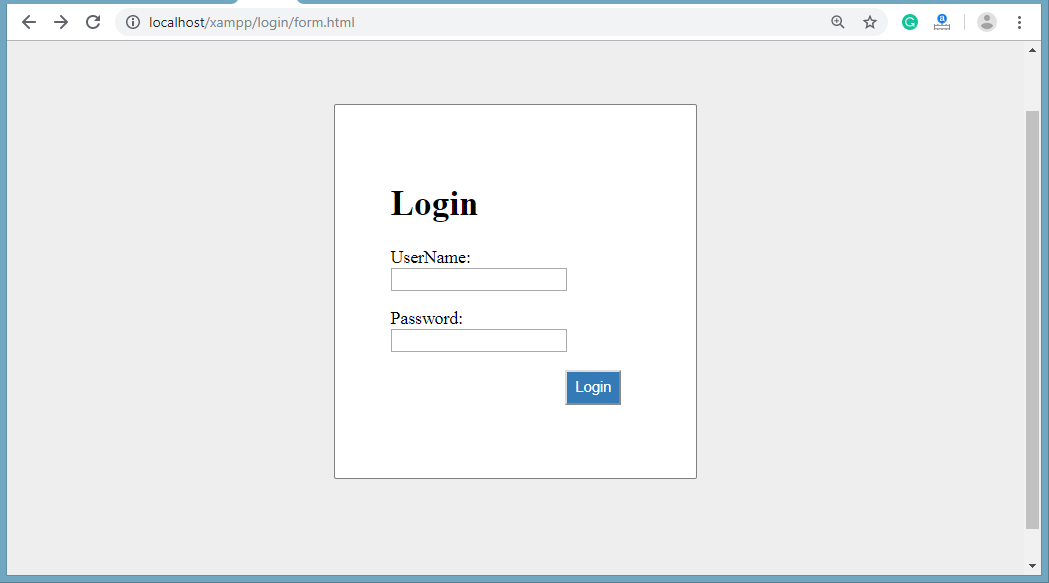


### **style.css**

Now, we will create style.css file to provide a more attractive view to the login form. The CSS code for the style.css file is given below:

1. body{
2. background: #eee;
3. }
4. #frm{
5. border: solid gray 1px;
6. width:25%;
7. border-radius: 2px;
8. margin: 120px auto;
9. background: white;
10. padding: 50px;
11. }
12. #btn{
13. color: #fff;
14. background: #337ab7;
15. padding: 7px;
16. margin-left: 70%;
17. }

After including above CSS file in index.html, the login form will be like -

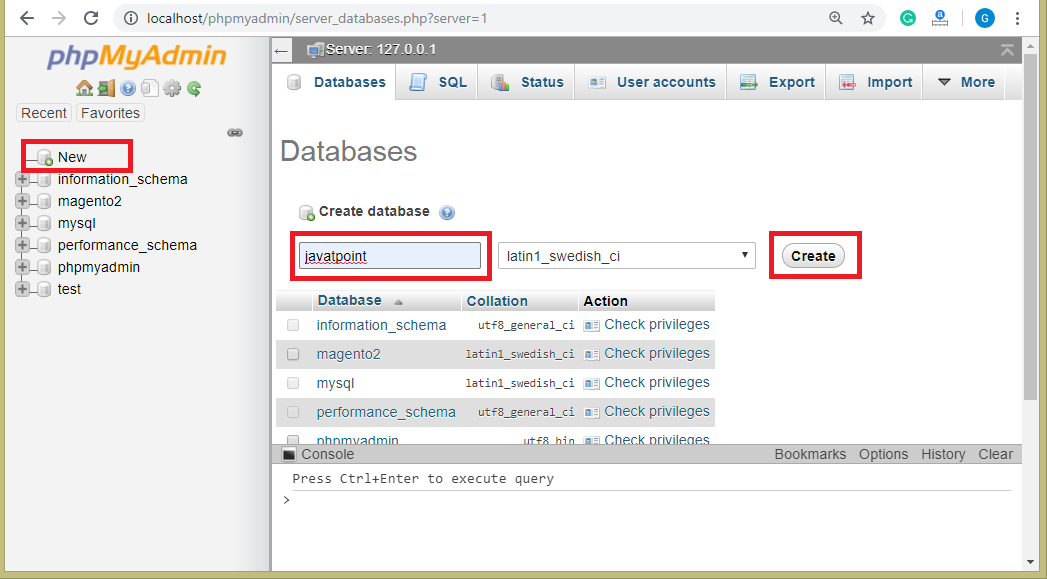


#### **Note: All these files should be saved in the same folder, so we can easily access them without any interruption.**

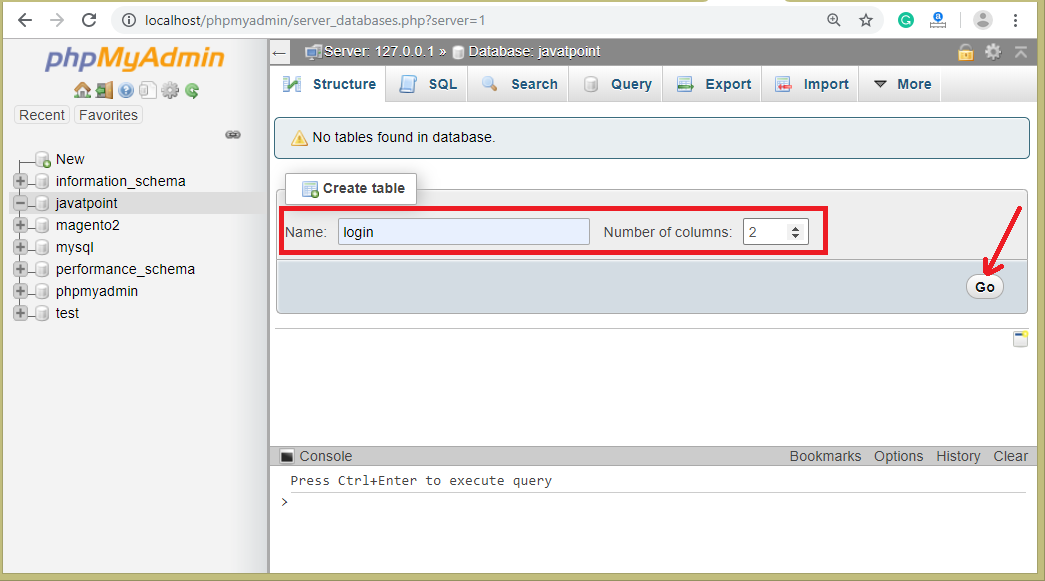
## Database and Table Creation

Now, the next step is to create the database and the login table inside the database.

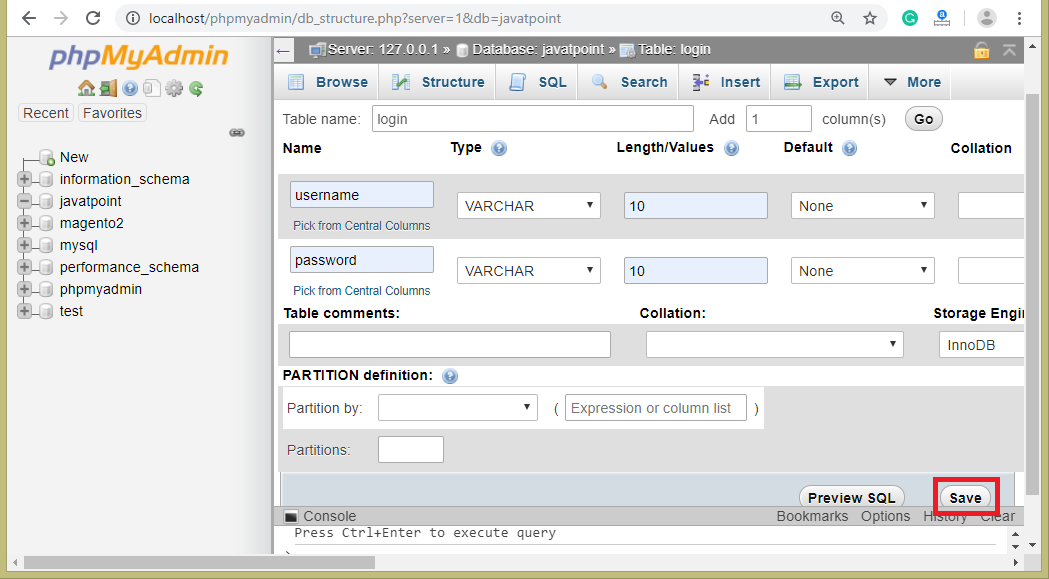
* Access the phpMyAdmin on the browser using **localhost/phpmyadmin/** and create a table in the database. Here we will create a database and table using GUI based phpMyAdmin rather than queries execution.
* Click on **New** and enter the database name and then click on **Create** button.



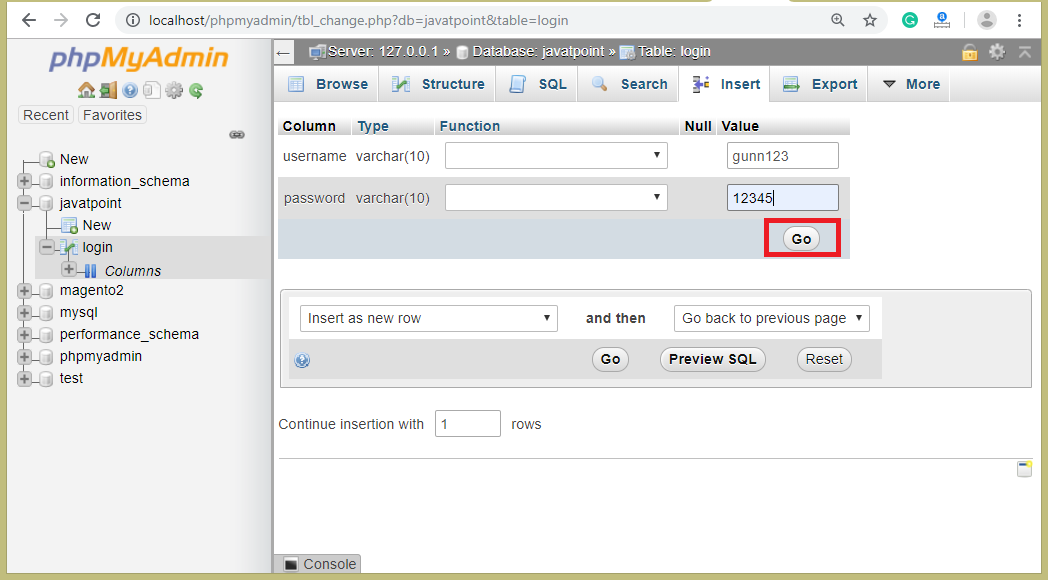
* Now we will create a login table in the database. Create a table by name **login** in the database which you have created earlier.



* Specify the column **Name** and their **Type** and **Length** in the table in which we will store the **username** and **password** for the different users and save it by clicking on the **save** button.



* Click on the insert, from where we can **insert** the records in columns. So insert the **username** and **password** here and click on **Go** button to save the record.



### **connection.php**

Next step is to do the connectivity of login form with the database which we have created in the previous steps. We will create connection.php file for which code is given below:

1. **<?php**
2. $host = "localhost";
3. $user = "root";
4. $password = '';
5. $db\_name = "javatpoint";
7. $con = mysqli\_connect($host, $user, $password, $db\_name);
8. if(mysqli\_connect\_errno()) {
9. die("Failed to connect with MySQL: ". mysqli\_connect\_error());
10. }
11. **?>**

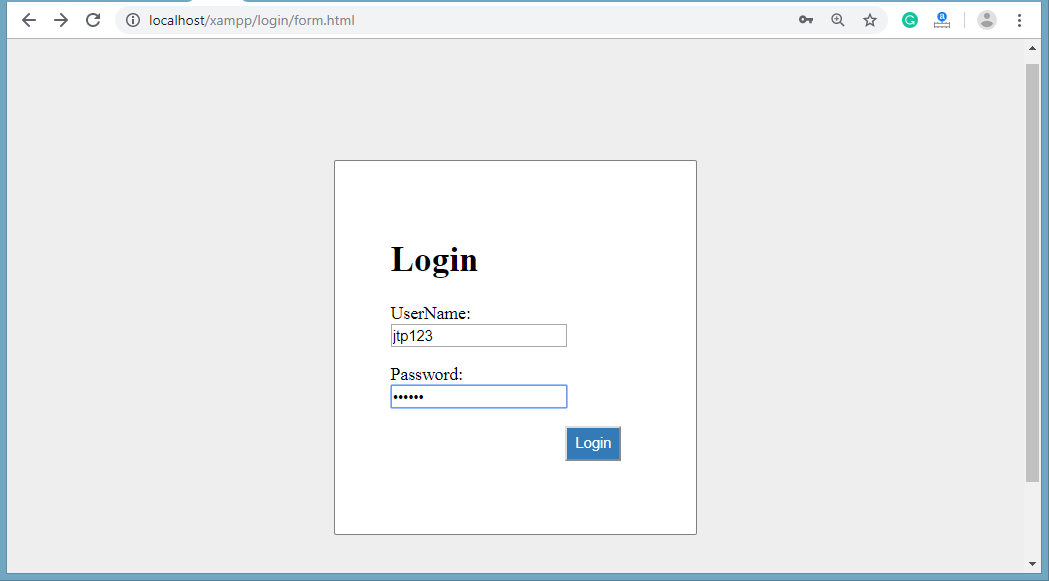
### **authentication.php**

Now, we have our database setup, so we can go with the authentication of the user. This file handles the login form data that sent through the index.html file. It validates the data sent through the login form, if the username and password match with the database, then the login will be successful otherwise login will be failed.

1. **<?php**
2. include('connection.php');
3. $username = $\_POST['user'];
4. $password = $\_POST['pass'];
6. //to prevent from mysqli injection
7. $username = stripcslashes($username);
8. $password = stripcslashes($password);
9. $username = mysqli\_real\_escape\_string($con, $username);
10. $password = mysqli\_real\_escape\_string($con, $password);
12. $sql = "select \*from login where username = '$username' and password = '$password'";
13. $result = mysqli\_query($con, $sql);
14. $row = mysqli\_fetch\_array($result, MYSQLI\_ASSOC);
15. $count = mysqli\_num\_rows($result);
17. if($count == 1){
18. echo "**<h1><center>** Login successful **</center></h1>**";
19. }
20. else{
21. echo "**<h1>** Login failed. Invalid username or password.**</h1>**";
22. }
23. **?>**

## How to run the login form?

* To run the login form, open the xampp control panel and run the apache server and PHP.
* Now, type localhost/xampp/folder name/file name in the browser and press Enter key.
* All setup is done now. Enter the username and password in the login form and click the login button.

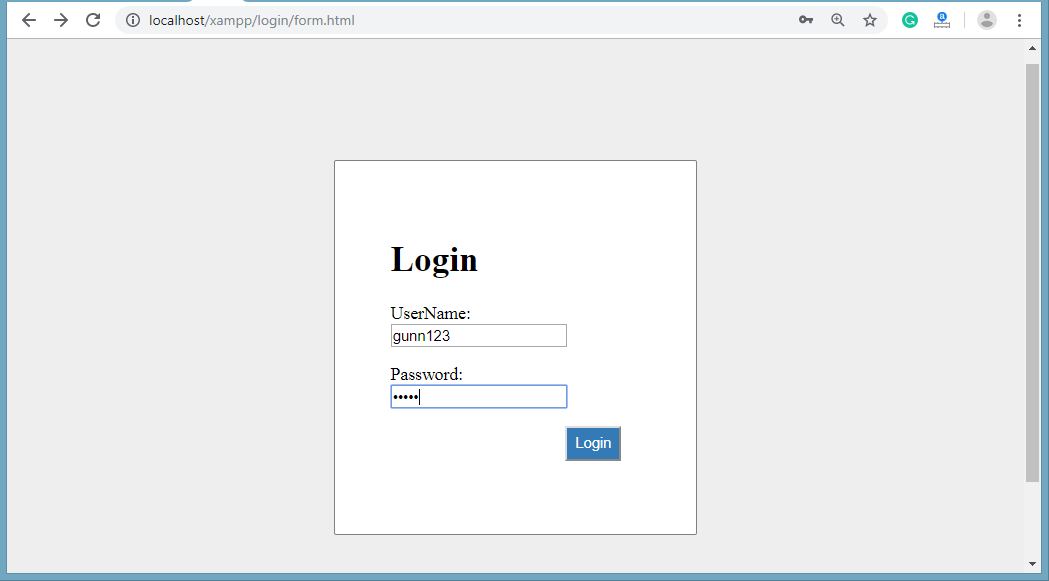


* Here, we have inserted an incorrect username, so the user is unable to log in, and it will give the login failed error.

**Output:**



* Now, we will provide correct value in the username and password. So, the user will be successfully logged in. See in the below example.



**Output**



# How to get the IP address in PHP?

Many times we need to get the IP address of the visitor for different purposes. It is very easy to collect the IP address in PHP. PHP provides PHP **$\_SERVER** variable to get the user IP address easily. We can track the activities of the visitor on the website for the security purpose, or we can know that who uses my website and many more.

The simplest way to collect the visitor IP address in PHP is the **REMOTE\_ADDR**. Pass the 'REMOTE\_ADDR' in PHP $\_SERVER variable. It will return the IP address of the visitor who is currently viewing the webpage.

#### **Note: We can display this IP address on the webpage and also even can store in database for many other purposes such as - for security, redirecting a visitor to another site, blocking/banning the visitor.**

## Get the IP address of the website

**$\_SERVER['REMOTE\_ADDR']** - It returns the IP address of the user currently visiting the webpage.

**For example**

Play Videox[](https://campaign.adpushup.com/get-started/?utm_source=banner&utm_campaign=growth_hack)

1. <?php
2. echo 'User IP Address - '.$\_SERVER['REMOTE\_ADDR'];
3. ?>

**Output**

User IP Address - ::1

But sometimes the REMOTE\_ADDR does not return the IP address of the client, and the main reason behind is to use the proxy. In such type of situation, we will try another way to get the real IP address of the user in PHP.

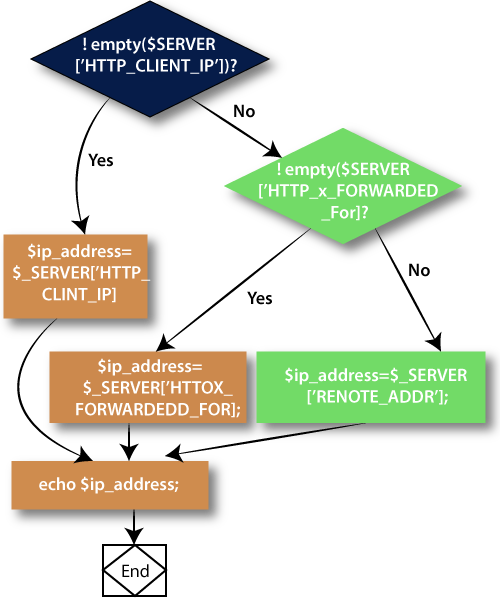
1. <?php
2. **function** getIPAddress() {
3. //whether ip is from the share internet
4. **if**(!empty**empty**($\_SERVER['HTTP\_CLIENT\_IP'])) {
5. $ip = $\_SERVER['HTTP\_CLIENT\_IP'];
6. }
7. //whether ip is from the proxy
8. **elseif** (!empty**empty**($\_SERVER['HTTP\_X\_FORWARDED\_FOR'])) {
9. $ip = $\_SERVER['HTTP\_X\_FORWARDED\_FOR'];
10. }
11. //whether ip is from the remote address
12. **else**{
13. $ip = $\_SERVER['REMOTE\_ADDR'];
14. }
15. **return** $ip;
16. }
17. $ip = getIPAddress();
18. echo 'User Real IP Address - '.$ip;
19. ?>

**Output**

User IP Address - ::1

**Flowchart:**

The flowchart for the above program will be like given below.



## Get the IP address of the website

We can also get the IP address of any website by its URL. Pass the URL of the website inside ***gethostbyname()*** function.

**For example**

1. <?php
2. $ip\_address = gethostbyname("www.google.com");
3. echo "IP Address of Google is - ".$ip\_address;
4. echo "</br>";
5. $ip\_address = gethostbyname("www.javatpoint.com");
6. echo "IP Address of javaTpoint is - ".$ip\_address;
7. ?>

**Output**

IP Address of Google is - 172.217.166.4

IP Address of javaTpoint is - 95.216.57.234

|  |  |
| --- | --- |
| **PHP** | **JavaScript** |
| PHP is a server-side scripting language. | JavaScript is a client-side scripting language. |
| PHP performs all the server-side functions like authentication, building custom web content, handling request, etc. | JavaScript is designed to create an interactive web application without interacting with the server |
| PHP can combine with HTML only. | JavaScript can combine with HTML, AJAX and also with XML. |
| PHP is used for back-end purpose only. | JavaScript is used for both front-end and back-end. |
| PHP is easy to learn. | JavaScript is complex to learn. |
| PHP is a multi-threadedlanguage, which means it blocks input/output to do multiple tasks concurrently. | JavaScriptis single-threaded, i.e.,event-driven, which means it never blocks, and everything runs in concurrent order. |
| In PHP, the code will be available and viewed after it is interpreted by the server. | A JavaScript code can be viewed Even after the output is interpreted. |
| It is synchronous by nature and waits for I/O operation to execute. | JavaScript is asynchronous by nature and does not wait for I/O operation to execute. |