



PHP BASIC

PHP Introduction
PHP Getting Started
PHP Syntax
PHP Variables
PHP Constants
PHP Echo and Print
PHP Data Types
PHP Strings
PHP Operators
PHP If...Else
PHP Switch...Case
PHP Arrays
PHP Sorting Arrays
PHP Loops
PHP Functions
PHP Math Operations
PHP GET and POST

PHP ADVANCED

PHP Date and Time
PHP Include Files
PHP File system
PHP Parsing Directories
PHP File Upload
PHP File Download
PHP Cookies
PHP Sessions
PHP Send Email
PHP Form Handling
PHP Form Validation
PHP Filters
PHP Error Handling
PHP Classes and Objects
PHP Magic Constants
PHP JSON Parsing
PHP Regular Expressions
PHP Exception Handling

PHP & MySQL DATABASE

PHP MySQL Introduction
PHP MySQL Connect
PHP MySQL Create Database
PHP MySQL Create Table
PHP MySQL Insert
PHP MySQL Prepared
PHP MySQL Last Inserted ID
PHP MySQL Select
PHP MySQL Where
PHP MySQL Limit
PHP MySQL Order By
PHP MySQL Update

ADVERTISEMENTS



PHP MySQL Login System

< >

In this tutorial you will learn how to build a login system with PHP and MySQL.

Implementing User Authentication Mechanism

User authentication is very common in modern web application. It is a security mechanism that is used to restrict unauthorized access to member-only areas and tools on a site.

In this tutorial we'll create a simple registration and login system using the PHP and MySQL. This tutorial is comprised of two parts: in the first part we'll create a user registration form, and in the second part we'll create a login form, as well as a welcome page and a logout script.

Building the Registration System

In this section we'll build a registration system that allows users to create a new account by filling out a web form. But, first we need to create a table that will hold all the user data.

Step 1: Creating the Database Table

Execute the following SQL query to create the *users* table inside your MySQL database.

Example

[Download](#)

```
1 CREATE TABLE users (  
2     id INT NOT NULL PRIMARY KEY AUTO_INCREMENT,  
3     username VARCHAR(50) NOT NULL UNIQUE,  
4     password VARCHAR(255) NOT NULL,  
5     created_at DATETIME DEFAULT CURRENT_TIMESTAMP  
6 );
```

Please check out the tutorial on [SQL CREATE TABLE statement](#) for the detailed information about syntax for creating tables in MySQL database system.

Step 2: Creating the Config File

After creating the table, we need create a PHP script in order to connect to the MySQL database server. Let's create a file named "config.php" and put the following code inside it.

Example

Procedural

Object Oriented

PDO

[Download](#)

```
1 <?php  
2 /* Database credentials. Assuming you are running MySQL  
3 server with default setting (user 'root' with no password) */  
4 define('DB_SERVER', 'localhost');  
5 define('DB_USERNAME', 'root');  
6 define('DB_PASSWORD', '');  
7 define('DB_NAME', 'demo');  
8  
9 /* Attempt to connect to MySQL database */  
10 $link = mysqli_connect(DB_SERVER, DB_USERNAME, DB_PASSWORD, DB_NAME);
```

PHP MySQL Delete
PHP MySQL CRUD Application
PHP MySQL Ajax Search
PHP MySQL Login System


PHP EXAMPLES

PHP Practice Examples
PHP FAQ's Answers

PHP REFERENCE

PHP String Functions
PHP Array Functions
PHP File System Functions
PHP Date/Time Functions
PHP Calendar Functions
PHP MySQLi Functions
PHP Filters
PHP Error Levels
[More references](#)


If you've downloaded the Object Oriented or PDO code examples using the download button, please remove the text "-oo-format" or "-pdo-format" from file names before testing the code.

 **Note:** Replace the credentials according to your MySQL server setting before testing this code, for example, replace the database name 'demo' with your own database name, replace username 'root' with your own database username, specify database password if there's any.

Step 3: Creating the Registration Form

Let's create another PHP file "register.php" and put the following example code in it. This example code will create a web form that allows user to register themselves.

This script will also generate errors if a user tries to submit the form without entering any value, or if username entered by the user is already taken by another user.

Example	Procedural	Object Oriented	PDO		Download
<pre>1 <?php 2 // Include config file 3 require_once "config.php"; 4 5 // Define variables and initialize with empty values 6 \$username = \$password = \$confirm_password = ""; 7 \$username_err = \$password_err = \$confirm_password_err = ""; 8 9 // Processing form data when form is submitted 10 if(\$_SERVER["REQUEST_METHOD"] == "POST"){ 11 12 // Validate username 13 if(empty(trim(\$_POST["username"]))){ 14 \$username_err = "Please enter a username.";</pre>					

— The output of the above example (i.e. signup form) will look something like this:

Sign Up

Please fill this form to create an account.

Username

Password


Confirm Password

Already have an account? [Login here.](#)

In the above example, we've used the PHP `password_hash()` function to create password hash from the password string entered by the user (*line no-75*). This function creates a password hash using a strong one-way hashing algorithm. It also generates and applies a random salt automatically when hashing the password; this means that even if two users have the same passwords, their password hashes will be different.

At the time of login we'll verify the given password with the password hash stored in the database using the PHP `password_verify()` function, as demonstrated in the next example.

We've used the Bootstrap framework to make the [form layouts](#) quickly and beautifully. Please, checkout the [Bootstrap tutorial](#) section to learn more about this framework.

 **Tip:** Password salting is a technique which is widely used to secure passwords by randomizing password hashes, so that if two users have the same password, they will not have the same password hashes. This is done by appending or prepending a random string, called a salt, to the password before hashing.

Building the Login System

In this section we'll create a login form where user can enter their username and password. When user submit the form these inputs will be verified against the credentials stored in the database, if the username and password match, the user is authorized and granted access to the site, otherwise the login attempt will be rejected.

Step 1: Creating the Login Form

Let's create a file named "login.php" and place the following code inside it.

Example	Procedural	Object Oriented	PDO	Download
<pre>1 <?php 2 // Initialize the session 3 session_start(); 4 5 // Check if the user is already logged in, if yes then redirect him to welcome page 6 if(isset(\$_SESSION["loggedin"]) && \$_SESSION["loggedin"] === true){ 7 header("location: welcome.php"); 8 exit; 9 } 10 11 // Include config file 12 require_once "config.php"; 13</pre>				

— The output of the above example (i.e. login form) will look something like this:

Login

Please fill in your credentials to login.

Username

Password

Don't have an account? [Sign up now.](#)

Step 2: Creating the Welcome Page

Here's the code of our "welcome.php" file, where user is redirected after successful login.

Example	Download
<pre>1 <?php 2 // Initialize the session 3 session_start(); 4 5 // Check if the user is logged in, if not then redirect him to login page 6 if(!isset(\$_SESSION["loggedin"]) \$_SESSION["loggedin"] !== true){</pre>	

```

7     header("location: login.php");
8     exit;
9 }
10

```

If data comes from external sources like form filled in by anonymous users, there is a risk that it may contain malicious script intended to launch cross-site scripting (XSS) attacks. Therefore, you must escape this data using the PHP `htmlspecialchars()` function before displaying it in the browser, so that any HTML tag it contains becomes harmless.

For example, after escaping special characters the string `<script>alert("XSS")</script>` becomes `<script>alert("XSS")</script>` which is not executed by the browser.

Step 3: Creating the Logout Script

Now, let's create a "logout.php" file. When the user clicks on the log out or sign out link, the script inside this file destroys the session and redirect the user back to the login page.

Example

[Download](#)

```

1  <?php
2  // Initialize the session
3  session_start();
4
5  // Unset all of the session variables
6  $_SESSION = array();
7
8  // Destroy the session.
9  session_destroy();
10
11 // Redirect to login page
12 header("location: login.php");
13 exit;
14 ?>

```

Adding the Password Reset Feature

Finally, in this section we will add the password reset utility to our login system. Using this feature *logged in users* can instantly reset their own password for their accounts.

Example

[Procedural](#)
[Object Oriented](#)
[PDO](#)

[Download](#)

```

1  <?php
2  // Initialize the session
3  session_start();
4
5  // Check if the user is logged in, if not then redirect to login page
6  if(!isset($_SESSION["loggedin"]) || $_SESSION["loggedin"] !== true){
7      header("location: login.php");
8      exit;
9  }
10
11 // Include config file
12 require_once "config.php";
13
14 // Define variables and initialize with empty values

```

[« PREVIOUS PAGE](#)
[NEXT PAGE »](#)

ADVERTISEMENTS

Free Online HTML Editor

Create, Edit W3C HTML Code

Create and edit W3C valid HTML code without writing a single line of code.
froala.com

OPEN

Free Online HTML Generator

Build HTML w/ WYSIWYG Editor

Create and edit W3C valid HTML code without writing a single line of code.
froala.com

OPEN

Bootstrap Code Snippets

Free ready to use UI design elements, templates and code snippets

Browse Snippets

Is this website helpful to you? Please give us a like, or share your feedback to help us improve. Connect with us on Facebook and Twitter for the latest updates.

ABOUT US

- Our Story
- Terms of Use
- Privacy Policy

CONTACT

- Contact Us
- Report Error
- Advertise

INTERACTIVE TOOLS

- Font Awesome Icon Search Utility
- Title & Meta Length Calculator
- Bootstrap Button Generator
- SQL Playground
- HTML Formatter
- HTML Color Picker
- HTML Editor

