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

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

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

```
<?php
/* Database credentials. Assuming you are running MySQL
```

```
server with default setting (user 'root' with no password) */
define('DB_SERVER', 'localhost');
define('DB_USERNAME', 'root');
define('DB_PASSWORD', '');
define('DB_NAME', 'demo');

/* Attempt to connect to MySQL database */
```

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

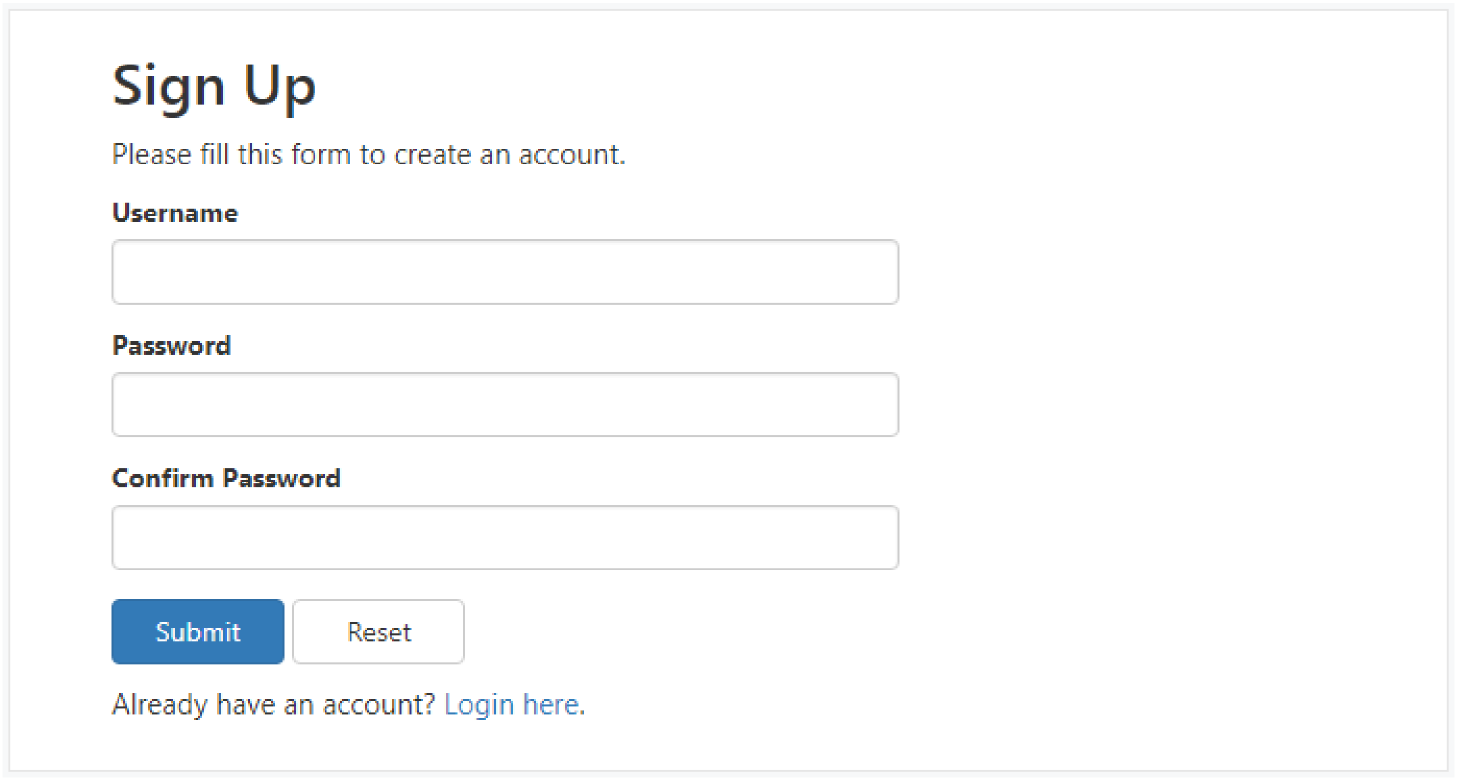
```
<?php
// Include config file
require_once "config.php";

// Define variables and initialize with empty values
$username = $password = $confirm_password = "";
$username_err = $password_err = $confirm_password_err = "";

// Processing form data when form is submitted
if($_SERVER["REQUEST_METHOD"] == "POST"){

    // Validate username
    if(empty(trim($_POST["username"]))){
        $username_err = "Please enter a username.";
```

— 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

Submit Reset

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.

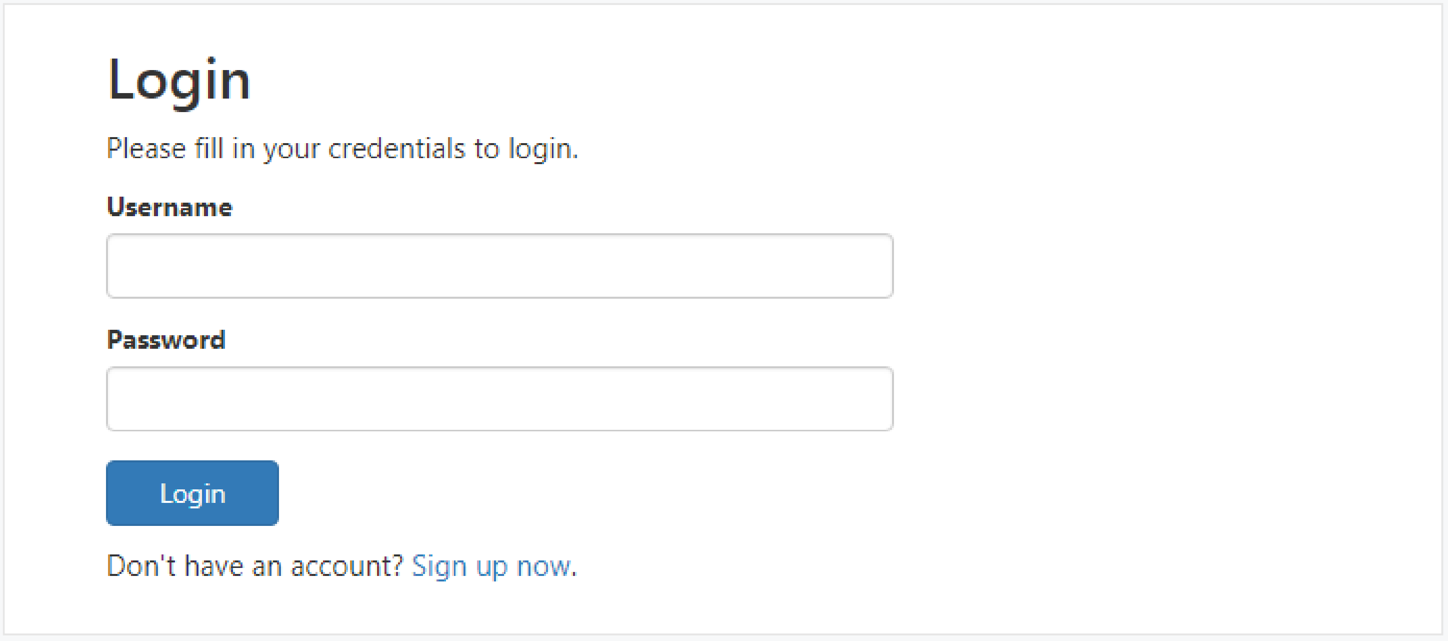
Example Procedural Object Oriented PDO Download

```
<?php
// Initialize the session
session_start();

// Check if the user is already logged in, if yes then redirect him
to welcome page
if(isset($_SESSION["loggedin"]) && $_SESSION["loggedin"] === true){
    header("location: welcome.php");
    exit;
}

// Include config file
require_once "config.php";
```

— 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

Login

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

```
<?php
// Initialize the session
session_start();

```

```
// Check if the user is logged in, if not then redirect him to login
page
if(!isset($_SESSION["loggedin"]) || $_SESSION["loggedin"] !== true){
    header("location: login.php");
    exit;
```

If data comes from external sources like form filled in by anonymous users, there is a risk that it may contain malicious script indented 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 `&lt;script&gt;alert("XSS")&lt;/script&gt;` 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**

```php
<?php
// Initialize the session
session_start();

// Unset all of the session variables
$_SESSION = array();

// Destroy the session.
session_destroy();

// Redirect to login page
header("location: login.php");
exit;
?>
```

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

```php
<?php
// Initialize the session
session_start();

// Check if the user is logged in, if not then redirect to login page
if(!isset($_SESSION["loggedin"]) || $_SESSION["loggedin"] !== true){
    header("location: login.php");
    exit;
}

// Include config file
require_once "config.php";

// Define variables and initialize with empty values
```