https://clevertechie.com/php/19/php-user-registration-form-mysql

Setting up Form CSS and HTML

First, go ahead and copy the HTML source from below codepen and place the code in a file called form.php. Also create another file named form.css in the same directory and copy and paste all of the CSS code from the codepen below into it:

Once you've saved form.php and form.css, you may go ahead and run form.php to see what the form looks like. It should look exactly the same as the one showing in the "Result" tab from the codepen above.

Creating the Database and Table

Before we start adding PHP code to our form, let's go ahead and create the database with a table which will store our registered users information in it. Below in the SQL script to create the database "accounts" and table "users":

CREATE DATABASE accounts;

CREATE TABLE `accounts`.`users`

(

`id` INT NOT NULL AUTO\_INCREMENT,

`username` VARCHAR(100) NOT NULL,

`email` VARCHAR(100) NOT NULL,

`password` VARCHAR(100) NOT NULL,

`avatar` VARCHAR(100) NOT NULL,

PRIMARY KEY (`id`)

);

Below is a complete code with error checking for connecting to MySQL database and running above SQL statements to create the database and users table:

*//connection variables*

$host = 'localhost';

$user = 'root';

$password = 'mypass123';

*//create mysql connection*

$mysqli = new mysqli($host,$user,$password);

if ($mysqli->connect\_errno) {

printf("Connection failed: %s\n", $mysqli->connect\_error);

die();

}

*//create the database*

if ( !$mysqli->query('CREATE DATABASE accounts2') ) {

printf("Errormessage: %s\n", $mysqli->error);

}

*//create users table with all the fields*

$mysqli->query('

CREATE TABLE `accounts2`.`users`

(

`id` INT NOT NULL AUTO\_INCREMENT,

`username` VARCHAR(100) NOT NULL,

`email` VARCHAR(100) NOT NULL,

`password` VARCHAR(100) NOT NULL,

`avatar` VARCHAR(100) NOT NULL,

PRIMARY KEY (`id`)

);') or die($mysqli->error);

With our HTML, CSS and the database table in place, we're now reading to start working on our form. The first step is to create a place for error messages to show up and then we'll start writing some form validation.

Starting New Session for Error Messages

Open up the form.php and add the following lines to it at the very top, make sure to use the php opening and closing tags ( I have not included the html part of form.php to keep things clean ).

<?php

*/\* form.php \*/*

session\_start();

$\_SESSION['message'] = '';

$mysqli = new mysqli("localhost", "root", "mypass123", "accounts\_complete");

*//form HTML code here.....*

?>

We have created new session because we're going to need to access $\_SESSION['message'] on the 'welcome.php' page after user successfully registers. MySQL connection has also been created right away, so we can work with the database later on.

We also need to print out $\_SESSION['message'] on the current page. From the beginning the message is set to '' (empty string) which is what we want, so nothing will be printed at this point. Let's go ahead and add the message inside the proper DIV tag:

**<div** class="alert alert-error"**>**<?= $\_SESSION['message'] ?>**</div>**

Creating Validation Rules

This form already comes with some validation rules, the keyword "required" inside the HTML input tags, is checking to make sure the field is not empty, so we don't have to worry about empty fields. Also, by setting input type to "email and "password", HTML5 validates the form for proper email and password formatting, so we don't need to create any rules for those fields either.

However, we still need to write some validation rules, to make sure the passwords are matching, the avatar file is in fact an image and make sure the user has been added to our database.

Let's create another file and call it validate.php to keep things well organized. We'll also include this file from our form.php.

<?php

*/\* form.php \*/*

session\_start();

$\_SESSION['message'] = '';

$mysqli = new mysqli("localhost", "root", "mypass123", "accounts\_complete");

require 'validate.php'; *//include validate.php file*

*//form HTML code here.....*

?>

The first thing we're going to do inside validate.php is to make sure the form is being submitted.

*/\* validate.php \*/*

*//the form has been submitted with post method*

if ($\_SERVER["REQUEST\_METHOD"] == "POST") {

}

Then we'll check if the password and confirm password are equal to each other

if ($\_SERVER["REQUEST\_METHOD"] == "POST") {

*//check if two passwords are equal to each other*

if ($\_POST['password'] == $\_POST['confirmpassword'])

{

}

}

Working with Super Global Variables

Note how we used super global variables $\_SERVER and $\_POST to get the information we needed. The keys names inside the $\_POST variable is available because we used method="post" to submit our form.

**<form** class="form" action="form.php" method="post" enctype="multipart/form-data" autocomplete="off"**>**

The key names are all the named HTML input fields with attribute name (eg: name="password", name="confirmpassword"):

**<input** type="password" placeholder="Password" name="password"**>** />

**<input** type="password" placeholder="Confirm Password" name="confirmpassword" **/>**

To clarify a bit more, here is what the $\_POST would look like (assuming all the fields in the form have been filled out) if we used a print\_r( $\_POST ) function on it, followed by die(); to terminate the script right after printing it. This is a good way of debugging your script and seeing what's going on:

if ($\_SERVER["REQUEST\_METHOD"] == "POST") {

print\_r( $\_POST ); die();

*/\*output:*

*Array ( [username] => clevertechie [email] => vladi@clevertechie.com [password] => mypass123 [confirmpassword] => mypass123 [register] => Register )*

*\*/*

Now we're going to get the rest of our submitted values from $\_POST and get them properly formatted so they can be inserted to our MySQL database table

*//the form has been submitted with post*

if ($\_SERVER["REQUEST\_METHOD"] == "POST") {

if ($\_POST['password'] == $\_POST['confirmpassword']) {

*//define other variables with submitted values from $\_POST*

$username = $mysqli->real\_escape\_string($\_POST['username']);

$email = $mysqli->real\_escape\_string($\_POST['email']);

*//md5 hash password for security*

$password = md5($\_POST['password']);

*//path were our avatar image will be stored*

$avatar\_path = $mysqli->real\_escape\_string('images/'.$\_FILES['avatar']['name']);

}

}

In the above code, we used real\_escape\_string() method to make sure our username, email and avatar\_path are formatted properly to be inserted as a valid SQL string into the database. We also used md5() hash function to create a hash string out of password for security.

How File Uploading Works

Also, notice the new super global variable $\_FILES, which holds the information about our image, which is the avatar being uploaded from the user's computer. The $\_FILES variable is available because we used enctype="multipart/form-data" in our form:

**<form** class="form" action="form.php" method="post" enctype="multipart/form-data" autocomplete="off"**>**

Here is the output if we use the print\_r( $\_FILES) followed by die(); just like we did for the $\_POST variable:

if ($\_SERVER["REQUEST\_METHOD"] == "POST") {

print\_r( $\_FILES ); die();

*/\*output:*

*Array ( [avatar] => Array ( [name] => guldan.png [type] => image/png [tmp\_name] => C:\Windows\Temp\php18D8.tmp [error] => 0 [size] => 98823 ) )*

*\*/*

*//this is how we're able to access the image name:*

$\_FILES['avatar']['name']; *//guldan.png*

When the file is first uploaded, using the post method, it will be stored in a temporary directory. That directory can be accessed with $\_FILES[ 'avatar '][ 'tmp\_name' ] which is "C:\Windows\Temp\php18D8.tmp" from the output above.

We can then copy that file from the temporary directory, to the directory that we want which is $avatar\_path. But before we copy the file, we should check if the file is in fact image, for that we'll check another key called [type] from our $\_FILES variable.

*//path were our avatar image will be stored*

$avatar\_path = $mysqli->real\_escape\_string('images/'.$\_FILES['avatar']['name']);

*//make sure the file type is image*

if (preg\_match("!image!",$\_FILES['avatar']['type']))

{

*//copy image to images/ folder*

if (copy($\_FILES['avatar']['tmp\_name'], $avatar\_path))

{

}

}

The preg\_match function matches the image from the [ 'type' ] key of $\_FILES array, we then use copy( ) function to copy our image file which takes in two parameters. The first one is the source file path which is our ['tmp\_name'] directory and the second one is the destination path which is our 'images/guldan.png' file path.

Saving User Data in a MySQL Database

We can now set some session variables which we'll need on the next page, which are username and avatar\_path, and we'll also create the SQL query which will insert all the submitted data into MySQL database:

if (copy($\_FILES['avatar']['tmp\_name'], $avatar\_path))

{

*//set session variables to display on welcome page*

$\_SESSION['username'] = $username;

$\_SESSION['avatar'] = $avatar\_path;

*//create SQL query string for inserting data into the database*

$sql = "INSERT INTO users (username, email, password, avatar) "

. "VALUES ('$username', '$email', '$password', '$avatar\_path')";

}

The final step is turn our query, using the query() method and check if it's successful. If it is, that means the user data has been saved in the "users" table successfully! We then set the final session variable $\_SESSION[ 'message' ] and redirect the user to the welcome.php page using the header( ) function:

*//check if mysql query is successful*

if ($mysqli->query($sql) === true)

{

$\_SESSION[ 'message' ] = "Registration succesful! Added $username to the database!";

*//redirect the user to welcome.php*

header( "location: welcome.php" );

}

That's pretty much all we need for the validation, we just need to add all the "else" keywords in case things don't go as planned from all the if statements we have created. Here is what the full code for validate.php looks so far:

*/\* validate.php \*/*

*//the form has been submitted with post*

if ($\_SERVER["REQUEST\_METHOD"] == "POST") {

*//two passwords are equal to each other*

if ($\_POST['password'] == $\_POST['confirmpassword']) {

*//define other variables with submitted values from $\_POST*

$username = $mysqli->real\_escape\_string($\_POST['username']);

$email = $mysqli->real\_escape\_string($\_POST['email']);

*//md5 hash password for security*

$password = md5($\_POST['password']);

*//path were our avatar image will be stored*

$avatar\_path = $mysqli->real\_escape\_string('images/'.$\_FILES['avatar']['name']);

*//make sure the file type is image*

if (preg\_match("!image!",$\_FILES['avatar']['type'])) {

*//copy image to images/ folder*

if (copy($\_FILES['avatar']['tmp\_name'], $avatar\_path)){

*//set session variables to display on welcome page*

$\_SESSION['username'] = $username;

$\_SESSION['avatar'] = $avatar\_path;

*//insert user data into database*

$sql =

"INSERT INTO users (username, email, password, avatar) "

. "VALUES ('$username', '$email', '$password', '$avatar\_path')";

*//check if mysql query is successful*

if ($mysqli->query($sql) === true){

$\_SESSION['message'] = "Registration successful!"

. "Added $username to the database!";

*//redirect the user to welcome.php*

header("location: welcome.php");

}

}

}

}

}

Setting Session Error Messages When Things Go Wrong

Let's go ahead and add all the else statements at once where we simply set the $\_SESSION[ 'message' ] error messages which will be printed out when any of our if statements fail. Add the following code right after the last if statement where we checked for successful mysqli query and within the last curly bracket like this:

if ($mysqli->query($sql) === true){

$\_SESSION['message'] = "Registration succesful!"

. "Added $username to the database!";

header("location: welcome.php");

}

else {

$\_SESSION['message'] = 'User could not be added to the database!';

}

$mysqli->close();

}

else {

$\_SESSION['message'] = 'File upload failed!';

}

}

else {

$\_SESSION['message'] = 'Please only upload GIF, JPG or PNG images!';

}

}

else {

$\_SESSION['message'] = 'Two passwords do not match!';

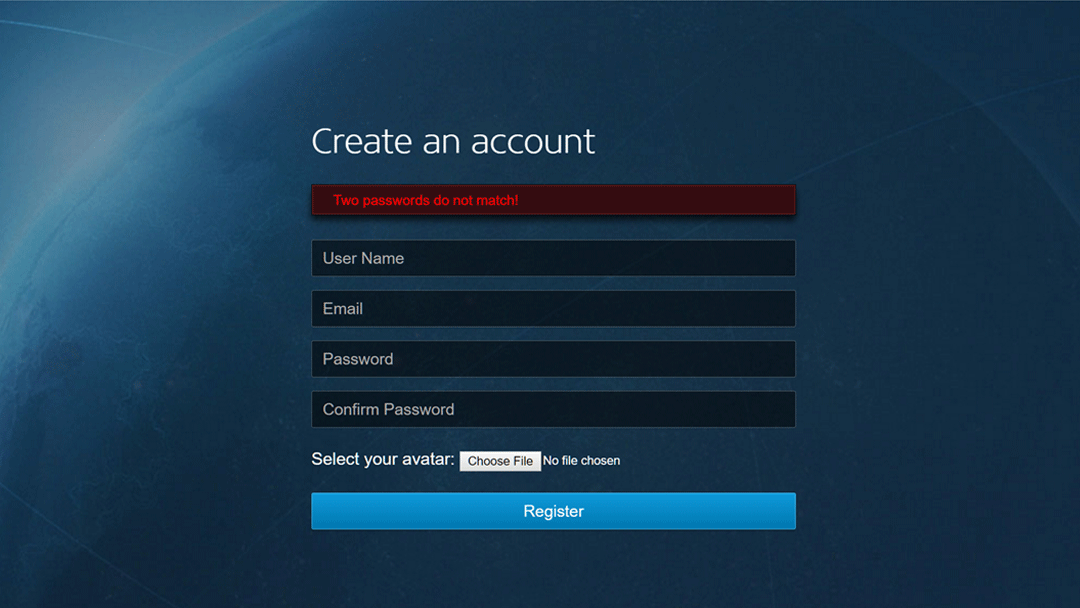
}

} *//if ($\_SERVER["REQUEST\_METHOD"] == "POST")*

The session message will then display the error message in the div tag where we put our $\_SESSION['message'] if you recall:

**<div** class="alert alert-error"**>**<?= $\_SESSION['message'] ?>**</div>**

Below is an example of what the error message is going to look like when two passwords don't match. Feel free to play around with it to trigger other error messages:



Creating User Profile Welcome Page

We're now done with the validate.php. The final step is to create welcome.php page which will display the username, avatar image and some users that have already been registered previously along with their own user names and mini avatar thumbnails. Here is what the complete welcome.php should look like, I will explain parts of it that may be confusing:

**<link** rel="stylesheet" href="form.css"**>**

<?php

*/\* welcome.php \*/*

*//$\_SESSION variables become available on this page*

session\_start();

?>

**<div** class="body content"**>**

**<div** class="welcome"**>**

**<div** class="alert alert-success"**>**<?= $\_SESSION['message'] ?>**</div>**

**<img** src="<?= $\_SESSION['avatar'] ?>"**><br** **/>**

Welcome **<span** class="user"**>**<?= $\_SESSION['username'] ?>**</span>**

<?php

$mysqli = new mysqli("localhost", "root", "mypass123", "accounts\_complete");

$sql = "SELECT username, avatar FROM users";

*//$result = mysqli\_result object*

$result = $mysqli->query( $sql );

?>

**<div** id='registered'**>**

**<span>**All registered users:**</span>**

<?php

*//returns associative array of fetched row*

while( $row = $result->fetch\_assoc() ){

echo "<div class='userlist'><span>".$row['username']."</span><br />";

echo "<img src='".$row['avatar']."'></div>";

}

?>

**</div>**

**</div>**

**</div>**

The $\_SESSION variable part from above should be easy to understand, we simply transfer over the variables from our validate.php page to this welcome.php page, if you're still confused by that, please check out [PHP Session](https://clevertechie.com/php/10/php-session) page for complete break down.

Working with MySQL Result Object

Whenever we use "SELECT" statement in our SQL query and then run that SQL with $mysqli->query( $sql ) command, the returned value is a MySQL result object. Once we have the result object, there are a few methods that become available so we can further start working with the data.

$sql = "SELECT username, avatar FROM users";

$result = $mysqli->query( $sql ); *//$result = mysqli\_result object*

One of those methods is $result->fetch\_assoc() which fetches the current row and returns an array with all the row data. So we're putting that in a [while loop](https://clevertechie.com/php/13/php-while-loop) conditional expression, which will become false when it reaches the last row in the result set, and storing the returned value from $result->fetch\_assoc() inside the $row variable.

*//returns associative array of fetched row*

while( $row = $result->fetch\_assoc() ){

echo "<div class='userlist'><span>".$row['username']."</span><br />";

echo "<img src='".$row['avatar']."'></div>";

}

Conclusion

And that's how we're able to access $row['username'] and $row['avatar'] from the associative array that is being returned, of the users that have already been registered previously and live in our users database table!

The profile welcome page should now look very similar to the one shown in the very beginning of this lesson, and the form is now complete, good job! Please post any questions you may have in the comments below.