Secure Password Storage with Password Hashing

In the previous tutorial, passwords were stored as plain text in the database. In this tutorial, you will update your project to store passwords securely using PHP password hashing.

Why Hash Passwords?

When passwords are stored as plain text, anyone with access to the database can read them.

Example:

username	password
admin	password123
teacher	secret123

A hashed password looks like this:

```
$2y$10$F8xJYQjN8s0T8f9mK6n7Iu8j9L2mWm9J4hJj7Qz0vB5lK3sHn8QyW
```

The original password cannot easily be recovered from the hash.

Create a Password Hash

Create a new PHP file called:

```
hash_password.php
```

Add the following code:

```
<?php

$password = "password123";

$hashedPassword = password_hash($password, PASSWORD_DEFAULT);

echo $hashedPassword;

?>
```

Save the file in your web server folder and run it in your browser.

Example:

```
http://localhost/hash_password.php
```

You should see a long string similar to:

```
$2y$10$F8xJYQjN8s0T8f9mK6n7Iu8j9L2mWm9J4hJj7Qz0vB5lK3sHn8QyW
```

Copy the Hash

Select and copy the generated hash.

You will use this value instead of the plain text password.

Update the Admin Account

Open phpMyAdmin.

Select the `users` table.

Locate the `admin` account and click **Edit**.

Replace:

```
password123
```

with your generated password hash.

Click **Go** to save the changes.

Verify the Password Was Updated

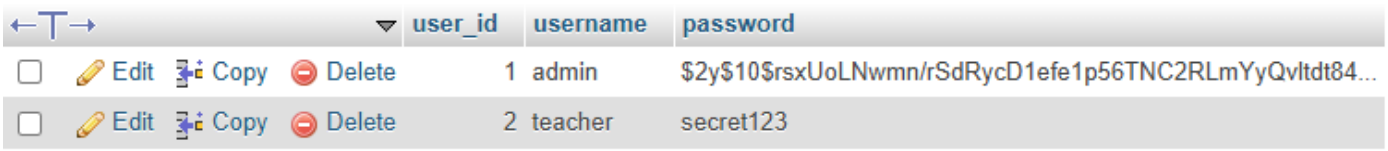
Run:

```
SELECT * FROM users;
```

The password column should now contain a long hash instead of a readable password.

Example:

user_id	username	password
1	admin	$2y$10$...



Verify a Password

Create a new file called:

```
verify_password.php
```

Add the following code:

```
<?php

$password = "password123";

$hash = '$2y$10$F8xJYQjN8s0T8f9mK6n7Iu8j9L2mWm9J4hJj7Qz0vB5lK3sHn8QyW';

if (password_verify($password, $hash)) {
    echo "Password is correct";
} else {
```

```
    echo "Password is incorrect";
}


?>
```

Replace the example hash with your own generated hash.

Open the file in your browser.

Example:

```
http://localhost/verify_password.php
```

You should see:

```
Password is correct
```

Test an Incorrect Password

Change:

```
$password = "password123";
```

to:

```
$password = "wrongpassword";
```

Refresh the page.

You should now see:

```
Password is incorrect
```

Common Mistake

Do not use:

```
md5()
```

or

```
sha1()
```

for password storage.

Modern PHP applications should use:

```
password_hash()
password_verify()
```

These functions automatically use secure hashing algorithms and are updated as PHP improves.

Complete Example

Generate a Hash

```
<?php

$password = "password123";

$hashedPassword = password_hash($password, PASSWORD_DEFAULT);

echo $hashedPassword;

?>
```

Verify a Password

```
<?php

$password = "password123";

$hash = '$2y$10$YOUR_HASH_HERE';

if (password_verify($password, $hash)) {
```

```
    echo "Password is correct";
} else {
    echo "Password is incorrect";
}

?>
```

You are now storing passwords securely and are ready to create a user registration form.

Revision #1
Created 2026-06-08 04:29:59 UTC by Mr Napper
Updated 2026-06-09 03:30:37 UTC by Mr Napper