Using PHP Sessions to Keep Users Logged In

PHP sessions store a small amount of trusted server-side state between requests. A login system can use them to remember the authenticated user’s identifier, username and role.

Start the session

Call `session_start()` before HTML or other output:

```
<?php
session_start();
```

Set identity after successful login

Only set these values after a database user has been found and `password_verify()` succeeds:

```
session_regenerate_id(true);

$_SESSION["user_id"] = (int) $user["user_id"];
$_SESSION["username"] = $user["username"];
$_SESSION["role"] = $user["role"];
```

The role must come from the database. Never trust a role supplied by a login or registration form.

Read session values safely

```
<?php
session_start();
```

```
$username = $_SESSION["username"] ?? "";
$role = $_SESSION["role"] ?? "user";
?>
<p>
  Signed in as
  <?= htmlspecialchars($username, ENT_QUOTES, "UTF-8") ?>
</p>
```

Session values still need escaping when inserted into HTML.

What belongs in a session

Suitable:

- internal user identifier
- display username
- authorised role
- short status messages
- CSRF tokens

Avoid:

- plain-text passwords
- complete database records
- sensitive information not needed across requests
- values copied directly from unvalidated form fields

Important limits

A session records authentication state; it does not automatically protect a page. Every protected server-side route must check the required identity and role.

Check

- ☐ Session starts before output.
- ☐ ID is regenerated after successful login.
- ☐ Identity and role come from the database.
- ☐ Password is never stored in the session.

- [ ] Session text is escaped when displayed.
- [ ] Protected routes perform their own checks.

Revision #2
Created 2026-06-08 04:55:14 UTC by Mr Napper
Updated 2026-08-18 23:49:47 UTC by Mr Napper