PHP Sessions

In this tutorial you will learn how to store certain data on the server on a temporary basis using PHP session.

What is a Session

Although you can store data using cookies but it has some security issues. Since cookies are stored on user's computer it is possible for an attacker to easily modify a cookie content to insert potentially harmful data in your application that might break your application.

Also every time the browser requests a URL to the server, all the cookie data for a website is automatically sent to the server within the request. It means if you have stored 5 cookies on user's system, each having 4KB in size, the browser needs to upload 20KB of data each time the user views a page, which can affect your site's performance.

You can solve both of these issues by using the PHP session. A PHP session stores data on the server rather than user's computer. In a session based environment, every user is identified through a unique number called session identifier or SID. This unique session ID is used to link each user with their own information on the server like emails, posts, etc.

Tip: The session IDs are randomly generated by the PHP engine which is almost impossible to guess. Furthermore, because the session data is stored on the server, it doesn't have to be sent with every browser request.

Starting a PHP Session

Before you can store any information in session variables, you must first start up the session. To begin a new session, simply call the PHP `session_start()` function. It will create a new session and generate a unique session ID for the user.

The PHP code in the example below simply starts a new session.

Example Download

```
<?php
// Starting session
session_start();
?>
```

The `session_start()` function first checks to see if a session already exists by looking for the presence of a session ID. If it finds one, i.e. if the session is already started, it sets up the session variables and if doesn't, it starts a new session by creating a new session ID.

Note: You must call the `session_start()` function at the beginning of the page i.e. before any output generated by your script in the browser, much like you do while setting the cookies with `setcookie()` function.

Storing and Accessing Session Data

You can store all your session data as key-value pairs in the `$_SESSION[]` superglobal array. The stored data can be accessed during lifetime of a session. Consider the following script, which creates a new session and registers two session variables.

Example Download

```
<?php
// Starting session
session_start();

// Storing session data
$_SESSION["firstname"] = "Peter";
$_SESSION["lastname"] = "Parker";
?>
```

To access the session data we set on our previous example from any other page on the same web domain — simply recreate the session by calling `session_start()` and then pass the corresponding key to the `$_SESSION` associative array.

Example Download

```
<?php
// Starting session
session_start();

// Accessing session data
echo 'Hi, ' . $_SESSION["firstname"] . ' ' . $_SESSION["lastname"];
?>
```

The PHP code in the example above produce the following output.

```
Hi, Peter Parker
```

Note: To access the session data in the same page there is no need to recreate the session since it has been already started on the top of the page.

Destroying a Session

If you want to remove certain session data, simply unset the corresponding key of the `$_SESSION` associative array, as shown in the following example:

Example **Download**

```php
<?php
// Starting session
session_start();

// Removing session data
if(isset($_SESSION["lastname"])){
    unset($_SESSION["lastname"]);
}
?>
```

However, to destroy a session completely, simply call the `session_destroy()` function. This function does not need any argument and a single call destroys all the session data.

Example **Download**

```php
<?php
// Starting session
session_start();

// Destroying session
session_destroy();
?>
```

Note: Before destroying a session with the `session_destroy()` function, you need to first recreate the session environment if it is not already there using the `session_start()`function, so that there is something to destroy.

Every PHP session has a timeout value — a duration, measured in seconds — which determines how long a session should remain alive in the absence of any user activity. You can adjust this timeout duration by changing the value of `session.gc_maxlifetime` variable in the PHP configuration file (`php.ini`).