SESSIONS

Sessions

- A session is a way to store information across multiple pages within a website for individual users.
- 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.
- Example: When a user logs into a website, a session can be used to remember their login details throughout their visit.

Session Lifecycle:

- Here's a step-by-step description of the PHP session lifecycle from creation to deletion:

- **Start**: Indicates the beginning of the session lifecycle.

- **Session Creation**: session_start() is called to initialize the session. This step checks if a session already exists.

- **Is a Session Already Present? (Decision Point)**:

➢ If **No**, a new session is created, and a unique session ID is assigned.

➢ If **Yes**, continue with the existing session.

Session Lifecycle:

- **Assign Session ID**: A unique session ID PHPSESSID is assigned and stored in a cookie on the client side. The session ID links to session data stored on the server.

- **Set Session Variables**:Session variables are set using the $_SESSION superglobal array.

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

- **User Navigates Pages**: Each time the user navigates through pages, the PHPSESSID cookie is sent with the HTTP request to the server. This allows the server to associate the request with the correct session.

Session Lifecycle:

- **Is the Session Expired or Manually Destroyed? (Decision Point)**: Check if the session has expired (based on session.gc_maxlifetime) or if it has been destroyed manually using session_destroy().
  - **If Expired**:The session is automatically deleted from the server, and the PHPSESSID cookie becomes invalid.
  - **If Manually Destroyed**: Using session_destroy() or unset($_SESSION), the session data is removed from the server, and the PHPSESSID cookie is deleted.
- **Session Deleted**:Session data on the server is deleted or invalidated on the client side.
- **End**: The session lifecycle ends.

Start 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

- Session variables are set with the PHP global variable: $_SESSION.

```
<?php
  session_start();  // Start a new session
  $_SESSION['username'] = 'JohnDoe';
?>
```

Setting up variables

```
<?php
  session_start();
  $_SESSION['userID'] = 1001;
  $_SESSION['cart'] = array("item1" => "Laptop", "item2" =>
"Mouse");
?>
```

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.

```
<?php
  session_start();
  echo "Username: " . $_SESSION['username'];   // Output: JohnDoe
?>
```

Removing set up variables

- To remove session variables, use unset().

- Removes a specific session variable.

```
<?php
  session_start();
  unset($_SESSION['cart']);  // Remove the 'cart' session variable
?>
```

Destroying a Session

- If you want to remove certain session data, simply unset the corresponding key of the $_SESSION associative array.

- 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.

```
<?php
  session_start();
  session_destroy();  // Destroy the entire session
?>
```

Session_unset() vs Session_destroy()

- **session_unset()**
  - Purpose: Clears all session variables.
  - Effect: Removes all variables stored in the $_session array, making them inaccessible.
  - Session State: The session itself is still active, meaning the session ID remains intact, and the session file on the server is not deleted.
  - Use Case: Use session_unset() when you want to clear session data but keep the session alive for other purposes, such as retaining the session ID for tracking purposes.

Example

```
<?php
session_start();
$_SESSION['username'] = 'JohnDoe';
$_SESSION['role'] = 'Admin';

session_unset(); // Removes all session variables

// Output
echo isset($_SESSION['username']) ? $_SESSION['username'] : "Session variable 'username' is not set.";
// Output: Session variable 'username' is not set.
?>
```

Session_unset() vs Session_destroy()

- **session_destroy()**

- Purpose: Completely terminates the session.
- Effect: Destroys the session data and removes the session ID stored on the server. Any data associated with the session is deleted, and the session becomes invalid.
- Session State: After session_destroy(), the session is no longer active, and the session file is removed from the server.
- Use Case: Use session_destroy(), when you want to log the user out completely and end the session entirely.

Example

```
<?php
session_start();
$_SESSION['username'] = 'JohnDoe';
$_SESSION['role'] = 'Admin';

session_destroy(); // Terminates the entire session

// Output
echo isset($_SESSION['username']) ? $_SESSION['username'] : "Session variable 'username' is not set.";
// Output: Session variable 'username' is not set.
?>
```