Cookies and Sessions in PHP

This lab will cover:

1. Setting and retrieving cookies.
2. Using cookies with PHP's $_COOKIE superglobal.
3. Setting and retrieving session variables using $_SESSION.
4. Discussion on use cases for cookies and sessions.

Part 1: Working with Cookies

Step 1: Create a PHP page to Set a Cookie

1. **Purpose**: This example will create a cookie to store a username.
2. **Code**: Create a file named setcookie.php with the following code.

```
<?php
// Set a cookie named "username" that expires in 1 day
setcookie("username", "JohnDoe", time() + 86400, "/"); // 86400 seconds = 1 day

echo "The cookie 'username' has been set!";
?>
```

- **Explanation**:
  - setcookie() function sets a cookie on the client's browser.
  - The parameters are:
    - "username": the name of the cookie.
    - "JohnDoe": the value to store in the cookie.
    - time() + 86400: sets the expiration time (1 day from now).
    - "/": the path (here, root / to make it available across the site).

Step 2: Create a PHP Page to Retrieve the Cookie

1. **Purpose**: This will check if the "username" cookie is set and retrieve its value.
2. **Code**: Create a file named getcookie.php with the following code.

```
<?php
if(isset($_COOKIE["username"])) {
   echo "Hello, " . $_COOKIE["username"] . "! Welcome back.";
```

```
} else {
   echo "No 'username' cookie found. Please set it first.";
}
?>
```

- **Explanation**:
    - $_COOKIE["username"] retrieves the value of the "username" cookie if it exists.
    - isset() checks if the "username" cookie has been set.

Step 3: Delete a Cookie

1. **Purpose**: Show how to delete a cookie by setting its expiration time in the past.
2. **Code**: Create a file named deletecookie.php.

```
<?php
// Delete the "username" cookie by setting its expiration time to the past
setcookie("username", "", time() - 3600, "/");

echo "The 'username' cookie has been deleted.";
?>
```

- **Explanation**:
    - Setting the cookie with a negative expiration time effectively deletes it from the client's browser.

Use Cases for Cookies:

- **Cookies** are suited for **storing small, non-sensitive data** on the client side, such as preferences or identifying return visitors.
- They are limited in size and can be accessed by the client, so they should not be used for storing sensitive data.

Part 2: Working with Sessions

Step 1: Start a Session and Set Session Variables

1. **Purpose**: Demonstrate server-side session management by setting session variables.
2. **Code**: Create a file named setsession.php with the following code.

```
<?php
session_start(); // Start the session

// Set session variables
$_SESSION["username"] = "JohnDoe";
$_SESSION["role"] = "Admin";

echo "Session variables are set.";
?>
```

- **Explanation**:
  - session_start() starts a new session or resumes an existing one.
  - $_SESSION superglobal array is used to store session data.
  - Session data is stored on the server, and a session ID is sent to the client in a cookie.

Step 2: Retrieve Session Variables

1. **Purpose**: Retrieve and display the session variables set in the previous step.
2. **Code**: Create a file named getsession.php with the following code.

```
<?php
session_start(); // Start the session

// Check if session variables are set and display them
if(isset($_SESSION["username"]) && isset($_SESSION["role"])) {
   echo "Hello, " . $_SESSION["username"] . "! Your role is " . $_SESSION["role"] . ".";
} else {
   echo "Session variables are not set.";
}
?>
```

- **Explanation**:
    - session_start() ensures the session data can be accessed.
    - Checking isset() on $_SESSION variables before use is a good practice.

Step 3: Destroy the Session

1. **Purpose**: End the session and clear all session data.
2. **Code**: Create a file named destroysession.php.

```
<?php
session_start(); // Start the session

// Destroy all session data
session_unset(); // Unset session variables
session_destroy(); // Destroy the session

echo "Session has been destroyed.";
?>
```

- **Explanation**:
    - session_unset() removes all session variables.
    - session_destroy() destroys the session completely.

Use Cases for Sessions:

- **Sessions** are best suited for storing **sensitive or user-specific data on the server**. For example, user authentication details, shopping cart items, etc.
- Since session data is stored on the server, it's more secure than client-side cookies for sensitive information.

Summary

Feature	Cookies	Sessions
Storage Location	Client-side	Server-side
Data Size	Limited (~4KB)	Can store larger data
Security	Less secure, visible to client	More secure, data not exposed

Expiration	Can be set manually	Expires when the browser or session ends by default
Best Use Cases	Remembering preferences, non-sensitive data	Authentication, sensitive data

Lab Checklist

1. **Set and retrieve cookies**:
    a. Visit setcookie.php to set a cookie.
    b. Visit getcookie.php to check if the cookie is present.
    c. Visit deletecookie.php to remove the cookie.
2. **Set and retrieve session variables**:
    a. Visit setsession.php to start a session and set variables.
    b. Visit getsession.php to retrieve session data.
    c. Visit destroysession.php to end the session and clear data.

This lab will give you hands-on experience with both cookies and sessions, helping you understand their distinct roles in managing user data in web applications.

Exercise: Now make a simple login-logout page with a simple cart.