Experiment No. : 11

Title: Write a program to manage session using PHP.

Objectives:

1. To study session handling in PHP.
2. To study various functions in handling sessions in login application in PHP.

Theory:

What is a PHP Session?

When you work with an application, you open it, do some changes, and then you close it. This is much like a Session. The computer knows who you are. It knows when you start the application and when you end. But on the internet there is one problem: the web server does not know who you are or what you do, because the HTTP address doesn't maintain state.

An overview of PHP session management is shown in Figure 1. When a user first enters the session-based application by making a request to a page that starts a session, PHP generates a session ID and creates a file that stores the session-related variables. PHP sets a cookie to hold the session ID in the response the script generates. The browser then records the cookie and includes it in subsequent requests. In the example shown in Figure 1, the script welcome.php records session variables in the session store, and a request to next.php then has access to those variables because of the session ID.

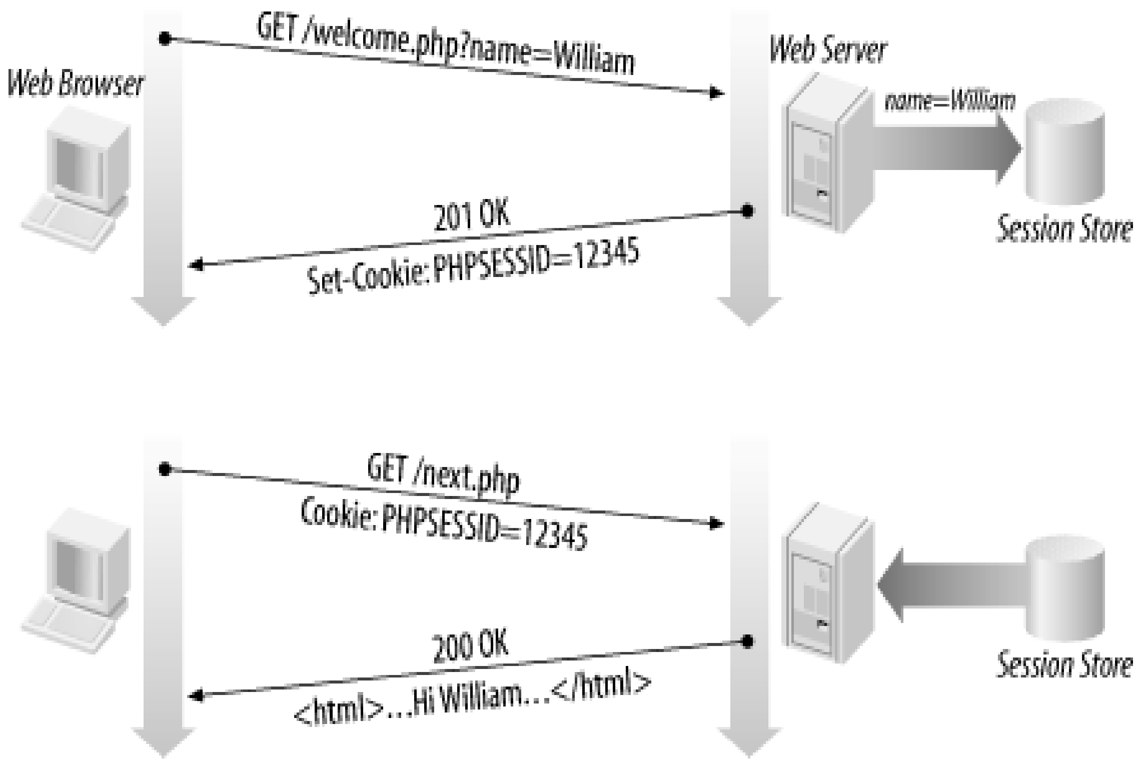

Figure 1

The out-of-the-box configuration of PHP session management uses disk-based files to store session variables. Using files as the session store is adequate for most applications in which the numbers of concurrent sessions are limited.

PHP provides a session_start ( ) function that creates a new session and subsequently identifies and establishes an existing one. Either way, a call to the session_start ( ) function initializes a session.

The first time a PHP script calls session_start ( ), a session identifier is generated, and, by default, a Set-Cookie header field is included in the response. The response sets up a session cookie in the browser with the name PHPSESSID and the value of the session identifier.

The session identifier (ID) is a random string of 32 hexadecimal digits, such as fcc17f071bca9bf7f85ca281094390b4. As with other cookies, the value of the session ID is made available to PHP scripts in the $HTTP_COOKIE_VARS associative array and in the $PHPSESSID variable.

When a new session is started, PHP creates a session file. With the default configuration, session files are written in the/tmp directory using the session identifier, prefixed with sess_, for the filename. The filename associated with our example session ID is /tmp/sess_fcc17f071bca9bf7f85ca281094390b4.

If a call is made to session_start ( ), and the request contains the PHPSESSID cookie, PHP attempts to find the session file and initialize the associated session variables as discussed in the next section. However, if the identified session file can't be found, session_start ( ) creates an empty session file.

Syntax:

```
<? php
 session_start ();                        // Start the session
$_SESSION ["favcolor"] = "green";         // Set session variables
?>

<? php
session_unset ();                         // remove all session variables
session_destroy ();                       // destroy the session
?>
```

Note: If you need a permanent storage, you may want to store the data in a database. session_destroy () will reset your session and you will lose all your stored session data.

Algorithm:

- Create a login form
- Start a session in login_process.php if username and password is same then go to secured.php otherwise go back to login.php page
- On secured.php page create logout link and jump to logout_process.php
- In logout_process.php destroy the session and jump to login.php page.

Key concepts:

Session , session_start ( ), session_destroy ( ), _SESSION, isset , session_register ( ) , session_unregister ( ), PHPSESSID