## UNIT VI: Cookies and Sessions

## 6.1 Using Cookies

A cookie is often used to identify a user. A cookie is a small file that the server embeds on the user's computer. Each time the same computer requests a page with a browser, it will send the cookie too. With PHP, you can both create and retrieve cookie values.

Create Cookies With PHP

A cookie is created with the setcookie() function.

Syntax

setcookie(name, value, expire, path, domain, secure, httponly);

Only the name parameter is required. All other parameters are optional.

PHP Create/Retrieve a Cookie

```
$cookie_name = "user";
$cookie_value = "Gopal Shinde";
setcookie($cookie_name, $cookie_value, time() + (86400 * 30), "/");
```

The "/" means that the cookie is available in entire website.

We then retrieve the value of the cookie "user" (using the global variable $_COOKIE). We also use the isset() function to find out if the cookie is set

```
<?php

if(!isset($_COOKIE[$cookie_name])) {

  echo "Cookie named '" . $cookie_name . "' is not set!";

}
else
{
  echo "Cookie '" . $cookie_name . "' is set!<br>";

  echo "Value is: " . $_COOKIE[$cookie_name];
}

?>
```

## 6.2 Using Sessions

Session is the time span between login & logout on a particular website by user.

A PHP session solves this problem by allowing you to store user information on the server for later use

If you need a permanent storage you may want to store the data in a database. Sessions work by creating a unique id (UID) for each visitor and store variables based on this UID.

The UID is either stored in a cookie.

A PHP session variable is used to store information about, or change settings for a user session. Session variables hold information about one single user, and are available to all pages in one application.

Starting a PHP Session

Before you can store user information in your PHP session, you must first start up the session.

The session_start() function must appear BEFORE the <html> tag:

```
<?php session_start(); ?>
<html><body></body></html>
```

Storing a Session Variable

The correct way to store and retrieve session variables is to use the PHP $_SESSION variable:

```
<?php session_start();
// store session data
$_SESSION['views']=1;
?>
<?php
//retrieve session data
echo "Pageviews=". $_SESSION['views'];
?>
```

Output:

Pageviews=1

Destroying a Session

If you wish to delete some session data, you can use the unset() or the session_destroy() function.

The unset() function is used to free the specified session variable:

```
<?php unset($_SESSION['views']);
?>
```

You can also completely destroy the session by calling the session_destroy() function:

```
<?php session_destroy();
?>
```

## 6.3 Sessions and Cookies

We can either use cookies or session for logging in and logging out or identifying user in your site.

Both are easy to use in PHP, but when to use Cookie & when to use Session is depending upon the following benefits of cookie & session.

Sessions have the following advantages over cookies:

1. They are generally more secure (because the data is being retained on the server).

2. They allow for more data to be stored.

3. They can be used without cookies.

Whereas cookies have the following advantages over sessions:

1. They are easier to program.

2. They require less of the server.

To store and retrieve just a couple of small pieces of information use cookies

For most of your Web applications, though, you'll use sessions.

## 6.4 Improving Session Security

Important information (like username, password) is generally stored in a session (you should never store important information in a cookie), for that security is more important.

An unauthorized person tries to hack or access a user session through the session ID than the data on the server.

The session ID is the key to the session data. By default, PHP will store this in a cookie, which is better for security.

It is possible in PHP to use sessions without cookies, it is harmful: If you identify another user's session ID, then you can easily access to their data. So storing the session ID in a cookie makes it somewhat harder to steal.

One method of preventing hijacking is to store some sort of user identifier in the session.

Instead of storing this value in the session as it is if we use md5() function for encrypting data, then we can add security.

This function returns a 32-character hexadecimal string based upon a value. In general, no two strings will have the same md5() result.

**Syntax:**

```
$_SESSION['agent'] = md5($data);
```

**The End**