HTTP is a stateless (or non-persistent) protocol. Each request is treated by its own. A request will not know what was done in the previous requests. The protocol is designed to be stateless for simplicity. However, some Internet applications, such as e-commerce shopping cart, require the state information to be passed one request to the next. Since the protocol is stateless, it is the responsibility of the application to maintain state information within their application.

Information about individual visit to a Web site is called state information. HTTP was originally designed to be stateless, which means that Web browser stored no data about pages viewed on previous visits to a Web site.

A few techniques can be used to maintain state information across multiple HTTP requests, namely,

1. Sessions
2. Cookie
3. Hidden fields of the HTML form.
4. URL rewriting.

Session Variable.

Session variables are variables, which remain common for the whole application of other pages but for one particular user. They also can be used across the whole application But they finish when a particular user session ends.

A session creates a file in a temporary directory on the server where registered session variables and their values are stored. This data will be available to all pages on the site during that visit. The location of the temporary file is determined by a setting in the ***php.ini*** file called session. save_path. Before using any session variable make sure you have setup this path.

When a session is started, following things happen:-

- PHP first creates a unique identifier for that particular session which is a random string of 32 hexadecimal numbers such as 3c7foj34c3jj973hjkop2fc937e3443.
- A cookie called PHPSESSID is automatically sent to the user's computer to store unique session identification string.
- A file is automatically created on the server in the designated temporary directory and bears the name of the unique identifier prefixed by sess_ iess_3c7foj34c3jj973hjkop2fc937e3443.

When a PHP script wants to retrieve the value from a session variable, PHP automatically gets the unique session identifier string from the PHPSESSID cookie and then looks in its temporary directory for the file bearing that name and a validation can be done by comparing both values.

Starting a PHP Session

A PHP session is easily started by making a call to the session start() function. These function first checks if a session is already started and if none is started then it starts one. It is recommended to put the call to session_start() at the beginning of the page.

Session variables are stored in associative array called $_SESSION[]. These variables can be accessed during lifetime of a session.

The following example starts a session then register a variable called counter that is incremented each time the page is visited during the session.

Make use of isset() function to check if session variable is already set or not.

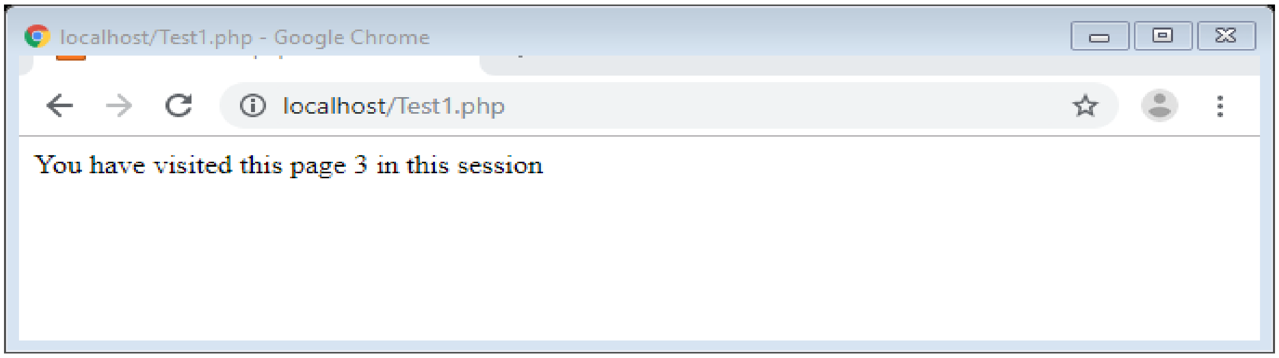
Type this code in a “SessionTest.php” file and load this file many times to see the result:

Example

```
<?php
  session_start();
   if( isset( $_SESSION['counter'] ) )
  {
        $_SESSION['counter'] += 1;
   }else
 {
          $_SESSION['counter'] = 1;
  }

  $msg = "You have visited this page ".  $_SESSION['counter']  . " in this session" ;
 echo $msg;
?>
```

It will produce the following result –



```
You have visited this page 3 in this session
```

Destroying a PHP Session

A PHP session can be destroyed by session_destroy() function. This function does not need any argument and a single call can destroy all the session variables. If you want to destroy a single session variable then you can use unset () function to unset a session variable.

Here is the example to unset a single variable –

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

Here is the call which will destroy all the session variables –

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

Turning on Auto Session

You don't need to call ***start_session()*** function to start a session when a user visits your site if you can set session.auto_start variable to 1 in ***php.ini*** file.

Cookie Variable.

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. The name of the **cookie** is automatically assigned to a variable of the same name.

Http is a stateless protocol, cookies allow us to track the state of the application using small files stored on the user's computer. The page requested that follow are personalized based on the set preferences in the cookies. Tracking the pages visited by a user.

Cookies are used to store the information of a web page in a remote browser, so that when the same user comes back to that page, that information can be retrieved from the browser itself.

PHP has a ***setcookie()*** function to send a cookie. ***setcookie()*** has several parameters. Following table discusses those.

Syntax: ***setcookie(name, value, expire, path, domain, secure, httponly)***

Parameter	Description	Data Type
name	Name of the cookie.	String
value	Value of the cookie, stored in client's computer.	String
expire	Unix timestamp, i.e. number of seconds since January 1st, 1970 (called as Unix Epoch).	Integer
path	Server path in which the cookie will be available.	String
domain	To which domain the cookie is available.	String
secure	If set true, the cookie is available over a secure connection only.	Boolean
httponly	If set true, the cookie is available over HTTP protocol only. Scripting languages like JavaScript won't be able to access the cookie.	Boolean

Last three parameters are optional.

Set Cookies:

```
<?php
 setcookie("user_name", "M.Masood", time()+ 180,'/',"localhost",1,1 );
 setcookie("user_password", "abcd1234", time()+ 180,'/',"localhost",1,1 );
 echo 'the cookie has been set for 180 seconds';
?>
```

Read or Accsess Cookie values.

```
<?php
   print_r($_COOKIE);
// or
   echo “<br> User name is “.  $_COOKIE[‘user_name’];
   echo “<br> User Password is “.  $_COOKIE[‘user_password’];
?>
```

Delete the Cookies:

There is no special dedicated function provided in PHP to delete a cookie. All we have to do is to update the expire-time value of the cookie by setting it to a past time using the ***setcookie()*** function. A very simple way of doing this is to deduct a few seconds from the current time.

Syntax: *setcookie(name, time() - 3600);*

```
<?php
  // Set the expiration date to one second ago

   setcookie("user_name", "Mr.Masood", time()-1,"/");
   setcookie("user_password", "xyzbcd1234", time()-1,"/");
    echo “ Cookies has been delete….”;
?>
```

Exercise

Theory Questions

1. Define the term "state information".
2. What are advantages and disadvantages of a stateless design?
3. What are the four tools for maintaining state information?
4. Describe the cookies.
5. How you can differentiate between the Cookies variables and session variables.
6. Query strings permanently maintain state information True or False with justify.

Practical Questions

1. Write a program to save your user ID, Name and Password in the cookies.
2. Write a program to save user ID and Password in the session variables.
3. Write a program to display total visitor of this web site. Increase visitor number when visit this web site. By using session technique.

Objective MCQ's

1. Stored information about a previous visit to a Web site is called _____ information.
   a) HTTP
   b) State
   c) Client-side
   d) Prior

2. What is the correct syntax for creating a temporary cookie that contains a value of "blue"?
   a) $color = setcookie("blue");
   b) setcookie("color","blue",time()+3000);
   c) setcookie("blue","color");
   d) setcookie("blue");

3. We use the _____ to read cookies in PHP.
   a) $_COOKIE[] autoglobal
   b) $_COOKIES[] autoglobal
   c) setcookie()
   d) getcookie()

4. Which of the following example specified that a cookie should expire in three days.
   a) Time()+48h
   b) Time()+24h*3
   c) Time()+60*60*24*7
   d) Time() +60*60*24*3

5. We use the _____ to access session variables in PHP.
   e) $_SESSION[ ] autoglobal
   f) $_SESSIONS[ ] autoglobal
   g) session_start()
   h) getsession()