Unit 4

Web Techniques

1. Introduction

PHP gives a very strong support to WEB designing.

A WEB-SITE is a collection of different web pages has to be maintained by server, and as a part of it, the PHP developer has to understand number of details about server.

In this chapter, we get access to form parameters and uploaded files, sending and retrieving the values of cookies, redirecting the browser to use PHP sessions.

2. Variables

Server configuration and request information – including form parameters and cookies – are accessible in three different ways from PHP scripts.

This information is collectively referred to as:

EGPCS (Environment, GET, POST, Cookies, Server)

For this the register_globals option in php.ini is enabled.

PHP creates six global arrays that contain the EGPCS information.

Example: http://localhost/test.php/

POST Method

i. The POST method encodes the form data in the body of the HTTP request.

ii. The data is not shown in the URL.

iii. The data cannot be bookmarked.

GET requests are idempotent-i.e. one GET request for a particular URL, including form parameters, is the same as two or more requests for that URL.

POST requests are not idempotent i.e. they cannot be cached and the server is reconnected every time the page is displayed.

The type of method that was used to request a PHP page is available through $_SERVER['REQUEST_METHOD'].

For example

```
if($_SERVER['REQUEST_METHOD'] = = GET)
{
  //you are handling a GET request
}
else
{
  die("you may only GET this page");
}
```

4.2 Parameters

The form parameters i.e. user data generated can be accessed via the PHP page using the arrays $_POST, $_GET and $_FILES.

The keys are parameter names and the values are the values of those parameters. In PHP variable names periods in field names are converted to underscores (_) in the array.

Consider the simple *example* of form using the POST method to pass name and age of a person.

```
<html>
<head><title>display information</title></head>
<body>
<form action="action.php" method="post">
```

```
    <p>Your name: <input type="text" name="name"/></p>
    <p>Your age: <input type="text" name="age"/></p>
    <p><input type="submit" /></p>
</form>
</body>
</html>
action.php
<?php
     $name=$_POST['name'];
     $age= $_POST['age'];
     Echo "Hi $name You are $age years old.";
?>
```

Note

The $_POST['name'] and $_POST['age'] variables are automatically set for us by PHP. Since the *method* of our form is POST. If we used the method *GET* then our form information would live in the $_GET auto global instead.

Consider the following *example* which splits a string into smaller substrings supplied by the user.

```
<html>
<head>
<title>Splitting up a string into smaller substrings
</title>
</head>
<body>
<form action = "string.php" method="POST">
<p>Enter the string: <input type="text" name="string"/></p>
<p>Enter the length of substrings:<input type="text"
name = "length"/></p>
<p><input type = "submit" /></p>
</form>
</body>
</html>
<html><head><title>substrings of the string</title></head>
<body>
<?php
$string = $_POST['string'];
$length = $_POST['length'];
$substring = ceil(strlen($string)/$length);
echo"the $length - letter substrings of $string are :<br/>\n";
for($i = 0 ;$i<$substring;$i++)
{
```

```
    $split=substr($string,$i*3,3);
    print("%d:%s<br/>\n",$i+1,$split);
}
?>
</body>
</html>
```

4.3 Automatic Quoting of Parameters

Magic quotes is the name of a PHP feature that automatically quotes input data, by using the addslashes() function magic_quotes_gpc ini option. If this option is enabled, PHP adds slashes (like the addslashes() function) on all external data. If we write our code on a system with this option disabled, and then move it to a server with magic_quotes_gpc enabled, user input will suffer from "backslash pollution."

What are Magic Quotes?

When on, all ' (single-quote), " (double quote), \ (backslash) and NULL characters are escaped with a backslash automatically. This is identical to what addslashes() does.

For instance if we enter the word "O' My Dear" in the form of above *example* and submit this value the word actually submitted is "O\' My Dear". That's magic_quotes_gpc at work.

```
<?php
echo get_magic_quotes_gpc(); // returns the current configuration
setting / i.e. 0 for off, 1 for on. So over here it echoes 1
echo $_POST['lastname'];              // O\'My Dear
echo addslashes($_POST['lastname']); // O\\\'My Dear
if(!get_magic_quotes_gpc())
{
   $lastname = addslashes($_POST['lastname']);
}
else
{
   $lastname = $_POST['lastname'];
}
echo $lastname; // O\'My Dear
?>
```

4.4 Self Processing Pages

The same PHP page can be used for both generating a form and process it.

Consider the following *example*: in which the form accepts Fahrenheit temperature and converts into Celsius in the same page.

```
<html><head><title>Temperature conversion program</title></head>
<body>
<?php
if($_SERVER['REQUEST_METHOD']=='GET')
{
?>
<form action="<?php echo $_SERVER['PHP_SELF']?>" method="POST">
Fahrenheit temperature is : <input type="text" name="Fahrenheit"/><br/>
<input type="submit" name="Conversion Temp"/><br/>
</form>
<?php
}
elseif($_SERVER['REQUEST_METHOD']=='POST')
{
  $fahr=$_POST['Fahrenheit'];
  $cesius=($fahr-32)*5/9;
  printf("%.2fF is %.2fC",$fahr,$celsius);
}
else
{
  die("The program works with GET and POST requests Only !!");
}
?>
</body>
</html>
```

4.5 Sticky Forms

A technique in which the result of a query is accompanied by a search forms whose default values are those of the previous queries.

1

Apr. 2017 – *4M*
What is sticky form? Explain with example.

Example,

If we give any "search string" to a search engine, the top of the result page contains another search box, which already contains the same "search string". To refine our search further we can add extra keywords to the "search string".

This is the sticky behaviour of form.

The basic technique to make the form sticky is to use the submitted form value as the default value when creating the HTML field.

Example,

```
<html>
<head>
<title>Temperature conversion program</title>
</head>
<body>
<?php
   $fahr=$_GET['Fahrenheit'];
?>
<form action="<?php echo $_SERVER['PHP_SELF']?>" method="GET">
   Fahrenheit temperature is: <input type="text" name="Fahrenheit"
   value="<?php echo $fahr?>"/>
<br/>
<input type="submit" name="Conversion temp"/><br/>
</form>

<?php
if(!is_null($fahr))
{
   $celsius=($fahr-32)*5/9;
  printf("%.2fF is %.2fC",$fahr,$celsius);
}
?>
</body>
</html>
```

4.6 Multivalued Parameters

HTML selection lists, created with the select tag, can allow multiple selections. PHP recognizes the multiple values that the browser passes to a form processing script, we need to make the name field in the HTML form end with [ ].

1

Apr. 2018 – 4M
Write a short note on Multivalued parameter

For example,

```
<html>
<head>
<title>Languages Information</title>
</head>
<body>
<?php
<form action="<?php echo $_SERVER['PHP_SELF']; ?>" method="GET">
Select the computer languages you know:<br/>
   <select multiple name="languages[]">
       <option value="oracle">Oracle</option>
       <option value="postgres">Postgres</option>
       <option value="mysql">Mysql</option>
 <option value="msaccess">Ms-Access</option>
   </select><br/>
<input type="submit" name="s" value="Record my knowledge!"/>
</form>
<?php
if(array_key_exists('s', $_GET))
{
$desc=join(" ", $_GET['languages']);
echo "you have the knowledge of following languages $desc.";
}
?>
</body>
</html>
```

4.7 Sticky Multivalued Parameters

Multiple selection form elements sticky is possible, all we need is to check to see whether each possible value in the form was one of the submitted values.

For example,

```
oracle: <input type="checkbox" name="lang[]"value="oracle"
<?= if(is_array($_GET['lang'])and
in_array('oracle', $_GET['lang']))
{
"checked";
}
?>/>
<br/>
```

This technique for each checkbox, but that's repetitive and error-prone. At this point, it's easier to write a function to generate the HTML for the possible values and work from a copy of the submitted parameters.

Example, below shows a new version of the multiple selection checkboxes, with the form made sticky.

```
<html>
<head><title>Personal Languages known</title></head>
<body>
<?php
// fetch form values, if any
$attrs = $_GET['attributes'];
if(! is_array($attrs)){$attrs = array();
}
// create HTML for identically-named checkboxes
function make_check($name, $query, $options)
{
foreach($options as $value => $label)
{
printf('%s<input type="checkbox" name="%s[]"value="%s"',
$label, $name, $value);
if(in_array($value, $query)) {echo "checked";
}
echo"/><br/>\n";
}
}
// the list of values and labels for the checkboxes
$personal_knowledge = array(
'oracle'=> 'Oracle',
'postgres'=> 'Postgres',
'Mysql' =>'MySQL',
```

```
'MsAccess'=>'msaccess'
);
?>
<form action="<?php $_SERVER['PHP_SELF']?>"method="GET">
Select your personal knowledge:<br/>
<?php
make_check('attributes', $attrs, $personal_knowledge);?>
<br/>
<input type="submit" name="s" value="Record my knowledge!"/>
</form>
<?php
if(array_key_exists('s', $_GET))
{
$description = join(" ", $_GET['attributes']);
echo "You have a $description knowledge of languages.";
}
?>
</body>
</html>
```

The heart of this code is the `make_check()` subroutine.

4.8 File Uploads

1

Apr. 2019 – 4M
How to handle file upload in php?

File upload feature of PHP can be used to upload images or related materials.

Because the browser needs to do a little bit more than just send a POST with the relevant data, you need to use a specially crafted form for file uploads. Here is an *example* of such a special form:

```
<form enctype="multipart/form-data"
  action="handle_img.php"method="post">
<input type="hidden" name="MAX_FILE_SIZE" value="16000" />
  Send this file: <input name="my_image" type="file" /><br />
<input type="submit" value="Upload" />
</form>
```

The differences between file upload forms and normal forms are bold in the code listing.

i. enctype attribute: instructs the browser to send a different type of POST request. i.e., request body containing the encoded files (and other form fields).

ii. The file upload field itself is the type file, which displays an input field and a browse button that allows a user to browse through the file system to find a file.

iii. The hidden input field sends a MAX_FILE_SIZE to the browser, setting the maximum allowable size of the file being uploaded.

Handling the Incoming Uploaded File

The $_FILES array contains an array of information about each file that is uploaded. The handler script can access the information using the name of the uploaded file as the key.

In the above *example* $_FILES['my_image'] variable contains the following information for the uploaded file.

Key	Value	Description
Name	String(8)	Name of the file on the file system of the user who uploaded it.
Type	String(10)	The MIME type of the file.
Tmp_name	String(14)	The temporary file name on the server's file system.
Error	Int(0)	The error code.
Size	Int(2045)	The size in bytes of the upload file.

A few possible errors can occur during a file upload. The following table shows the error conditions.

Constant	Description
UPLOAD_ERR_OK	File uploaded successfully.
UPLOAD_ERR_INI_SIZE	Size of uploaded file exceeded the value of the upload_max_file.
UPLOAD_ERR_FORM_SIZE	Size of the uploaded files exceeded the value MAX_FILE_SIZE.
UPLOAD_ERR_PARTIAL	Due to certain problem file didn't uploaded.
UPLOAD_ERR_NO_FILE	No file uploaded since the user didn't upload form.

The correct way to test whether a file was successfully uploaded is to use the function `is_uploaded_file()`, as follows:

```
if(is_uploaded_file($_FILES['toProcess']['tmp_name'])
{
  // successfully uploaded
}
```

Files are stored in the server's default temporary files directory, which is specified in *php.ini* with the `upload_tmp_dir` option.

4.9 Form Validation

When we allow users to input data, we typically need to validate that data before using it or storing it for later use. There are several strategies available for validating data.

Consider a simple *example* which shows a self-processing page with a form. The page allows the user to input a first name, last name, age, address, salary and area; if the form elements-the name, lname, age, address and salary is empty, the page is presented anew with a message detailing what's wrong. Any form fields the user already filled out are set to the values already entered.

Example,

```
<html>
<head>
<title>Simple Form validation</title>
</head>
<body>
<?php
if(isset($_POST['submit']))
{
$age=$_POST['age'];
$fname=$_POST['fname'];
$lname=$_POST['lname'];
$add=$_POST['add'];
$s=$_POST['a'];
if($age=="" or $fname=="" or $lname=="" or $add=="" or $s=="")
{
echo "All Fields are compulsory !!!";
}
}
?>
<form action="formvalid.php"<?php echo $_server[php_self]?>
method="POST">
First name : <input name="fname" type="text">
Age: <input name="age" type="text"><br/>
Last name : <input name="lname" type="text"><br/></br>
Address: <textarea name="add" rows="4" cols="40"></textarea></br>
<br> What is your current salary?
<select name="sal">
<option value=0> under 1000</option>
<option value=1> 1000-2000</option>
```

```
<option value=2> 2000-10000</option>
</select><br/><br/>
Your Sex:
Male<input name="a" type="radio">
Female<input name="a" type="radio">
<br>
<br>
<input type="submit" name="submit" value="submit">
</body>
</html>
```

Example: Form validation

Consider another *example* in which the page allows the user to input a media item the name, media type, and filename are required fields. If the user neglects to give a value to any of them, the page is presented anew with a message detailing what's wrong. Any form fields the user already filled out are set to the values she entered. The text of the submit button changes from "Create" to "Continue" when the user is correcting the form.

```
<?php
$name = $_POST['name'];
$mtype = $_POST['mtype'];
$filename = $_POST['filename'];
$caption = $_POST['caption'];
$tried = ($_POST['tried']=='yes');
if($tried)
{
  $valid = (!empty($name) && !empty($mtype) && !empty($filename));
  if(!$valid)
  {
    ?>
    <p>
      The name, media type, and filename are required fields. Please
      fill out to continue.
    </p>
    <?php
  }
}
if($tried && $valid)
{
  echo '<p>The item has been created.</p>';
```

```
}
// was this type of media selected? print "selected" if so
function media_select($type)
{
  global $mtype;
  if ($mtype == $type) { echo "selected";
}
}
?>
<form action="<?= $PHP_SELF ?>" method="POST">
  Name: <input type=text name="name" value="<?= $name ?>"/><br/>
  Status: <input type="checkbox" name="status" value="active"
<?php if($status == 'active') { echo 'checked'; }?>/>Active<br/>
  Media: <select name="mtype">
<option value="">Choose one</option>
<option value="picture" <?php media_select('picture')?> />
  Picture</option>
<option value="audio" <?php media_select('audio')?> />Audio</option>
<option value="movie" <?php media_select('movie')?> />Movie</option>
</select><br/>
  File: <input type="text" name="filename" value="<?= $filename ?>"/>
<br/>
  Caption: <textarea name="caption"><?= $caption ?></textarea><br/>
<input type="hidden" name="tried" value="yes"/>
<input type="submit"
 value="<?php echo $tried ? 'Continue':'Create'; ?>"/></form>
```

5. Setting Response Headers

Apr. 2018 – 4M
Write a short note on Setting response header
Apr. 2017 – 2M
List any two PHP HTTP function.

The HTTP response that a server sends back to a client contains

i. Headers that identify the type of content in the body of the response,

ii. The server that sent the **response.**

iii. How many bytes are in the body?

iv. When the response was sent, etc.

PHP and Apache normally take care of the headers. If we want to send back something that's not HTML, set the expiration time for a page, redirect the client's browser, or generate a specific HTTP error, we'll need to use the `header()` function.

The only thing that we must remember is that before any of the body is generated the `header()` function should be called (or `setcookie()`, if you're setting cookies) i.e. `header()` must happen at the very top of our file, even before the `<html>` tag.

For example,

```
<?php
header('Content-Type: text/plain');
?>
Date: today
From: Vision
To: Vision Publications
Subject: Hello!
Hello welcome to Visions Publications!!
```

Attempting to set headers after the document has started results in this warning:

Warning: Cannot add header information - headers already sent

5.1 Different Content Types

The Content-Type header identifies the type of document being returned. Ordinarily this is `"text/html"`, indicating an HTML document; `"text/plain"` forces the browser to treat the page as plain text.

5.2 Redirections

Redirection is used to send the browser to a new URL

```
<?php
header('Location: http://www.visions.com/hello.html');
exit();
?>
```

If we provide a partial URL (*example*, "/hello.html"), the redirection is handled internally by the web server. But, if there are relative URLs in the new document, the browser will interpret them as being relative to the document it requested, not the document it was sent.

5.3 Expiration

A server can explicitly inform the browser, and any proxy caches that might be between the server and browser, of a specific date and time for the document to expire.

To set the expiration time of a document, use the Expires header:

header ('Expires: Fri, 8 April 2010 05:30:00 GMT');

To expire a document three hours from the time the page was generated, we use `time()` and `gmstrftime()` to generate the expiration date string:

```
$now = time();
$then = gmstrftime("%a, %d %b %Y %H:%M:%S GMT", $now + 60*60*3);
header("Expires: $then");
```

To indicate that a document "never" expires, use the time a year from now:

```
$now = time();
$then = gmstrftime("%a, %d %b %Y %H:%M:%S GMT", $now + 365*86440);
header("Expires: $then");
```

To mark a document as already expired, use the current time or a time in the past:

```
$then = gmstrftime("%a, %d %b %Y %H:%M:%S GMT");
header("Expires: $then");
```

To prevent a browser or proxy cache from storing your document:

```
header("Expires: Mon, 26 Jul 1997 05:00:00 GMT");
header("Last-Modified:". gmdate("D, d M Y H:i:s"). " GMT");
header("Cache-Control: no-store, no-cache, must-revalidate");
header("Cache-Control: post-check=0, pre-check=0", false);
header("Pragma: no-cache");
```

5.4 Authentication

HTTP authentication mechanism uses request headers and response status. A browser can send a username and password in the request headers. If the Username and Password aren't sent or aren't satisfactory, the server sends a "401 Unauthorized" response.

To handle authentication in PHP, check the username and password (the `PHP_AUTH_USER` and `PHP_AUTH_PW` elements of `$_SERVER`) and call `header()` to set the realm and send a "401 Unauthorized" response:

```
header('WWW-Authenticate: Basic realm="Top Secret Files"');
header("HTTP/1.0 401 Unauthorized");
```

Following example checks to make sure that the password is the username, reversed:

```
<?php
$authentication = 0;
$username = $_SERVER['PHP_AUTH_USER'];
$password = $_SERVER['PHP_AUTH_PW'];
if(isset($username) && isset($password) && $user === strrev($password))
{
   $authentication = 1;
}
if(!$authentication)
{
   header('WWW-Authenticate: Basic realm="Top Secret Files"');
   header('HTTP/1.0 401 Unauthorized');
}
?>
```

If we have to protect more than one page put the above code into a separate file and `include` it at the top of every protected page.

6. Maintaining State

HTTP is a stateless protocol, which means that once a web server completes a client's request for a web page, the connection between the two goes away.

State is useful, though *for example*, if we can't keep track of a sequence of requests from a single user. We need to know when a user puts an item in his cart, when he adds items, when he removes them, and what's in the cart when he decides to check out.

Session tracking is used to keep track of state information between requests .hidden form fields can also be used to achieve this. PHP treats hidden form fields just like normal form fields, so the values are available in the `$_GET` and `$_POST` arrays.

With hidden form fields, we can pass around the entire contents of a shopping cart. While hidden form fields work in all browsers, they work only for a sequence of dynamically generated forms, so they aren't as generally useful as some other techniques.

Another technique is URL rewriting, where every local URL on which the user might click is dynamically modified to include extra information. *For example*, if we assign every user a unique ID, you might include that ID in all URLs, as follows:

```
http://www.vision.com/test.php?userid=123
```

Another technique for maintaining state is to use cookies.

A *cookie* is a bit of information that the server can give to a client. Cookies are useful for retaining information through repeated visits by a browser, but they're not without their own problems.

A problem with cookies is that some browsers don't support, and even with browsers that do, the user can disable cookies.

The best way to maintain state with PHP is to use the built-in session-tracking system. This system lets us create persistent variables that are accessible from different pages of our application, as well as in different visits to the site by the same user.

6.1 Cookies

3

Apr. 2019 – 2M
What is SetCookie() function?
Oct. 17, Apr. 17 – 4M
Explain cookie with example.

A cookie is basically a string that contains several fields. A server can send one or more cookies to a browser in the headers of a response.

The `value` field of the cookie is the payload—servers can store any data they like there (within limits), such as a unique code identifying the user, preferences, etc.

Syntax

```
setcookie(name [,value [,expire [,path [,domain [,secure ]]]]]);
```

Note Cookies are sent as headers in the response, setcookie() must be called before any of the body of the document is sent.

i. **Name:** A unique name for a particular cookie. There can be multiple cookies with different names and attributes. The name must not contain whitespace or semicolons.

ii. **Value:** The arbitrary string value attached to this cookie. There's no specific limit on the size of a cookie value, it probably can't be much larger than 3.5 KB.

iii. **Expire:** The expiration date for this cookie. If no expiration date is specified, the browser exits, the cookie disappears. The expiration date is specified as the number of seconds since midnight, January 1, 1970, GMT. *For example*, pass `time()+60*60*2` to expire the cookie in two hours' time.

iv. **Path:** The browser will return the cookie only for URLs below this path. The default is the directory in which the current page resides.

v. **Domain:** The browser will return the cookie only for URLs within this domain. The default is the server hostname.

vi. **Secure:** The browser will transmit the cookie only over *https* connections. The default is `false`, meaning that it's okay to send the cookie over insecure connections.

When a browser sends a cookie back to the server, we can access that cookie through the $_COOKIE array. The key is the cookie name, and the value is the cookie's value field.

```
<?php
$page_acc = $_COOKIE['page'];
setcookie('page', ++$page_acc);
?>
```

When decoding cookies, any periods (.) in a cookie's name are turned into underscores.

Example,

```
<?php
$value = "Vision Publications";
setcookie("Test", $value);
setcookie("Test", $value, time()+3600); /*expire in 1 hour*/
?>
```

Example,

```
<?php
// set the cookies
SETCOOKIE("cookie[three]", "three");
SETCOOKIE("cookie[two]", "two");
SETCOOKIE("cookie[one]", "one");
// after the page reloads, print them out
if(isset($_COOKIE['cookie']))
{
   foreach($_COOKIE['cookie'] as $name => $value)
{
    echo "$name : $value<br/>\n";
   }
}
?>
```

Output
three: three
two: two
one: one

Consider the following example in which the cookie is set, we may read the cookie by accessing cookie2.php for up to 24 hours. After that, the cookie will expire.

cookie1.php

```
<?php
    $username = $_POST['username'];
    $fcolor = $_POST['fcolor'];
    $self =  $_SERVER['PHP_SELF'];
    if(($username != null) and ($fcolor != null))
    {
```

```
    SETCOOKIE("firstname", $username , time() + 86400 ); // 24 hours
    SETCOOKIE( "fontcolor", $fcolor, time() + 86400);
     exit();
   }
?>
<html>
<head>
<title>Setting color and username</title>
</head>
<body>
    <h1>setting username and font color using cookies</h1>
    <hr>
    <form action ="<?php echo $_SERVER['PHP_SELF']?>" method = post">
       Please enter your first name: <input type="text" name =
            "username"><br><br>
       Please choose your favorite font color:<br>
    <input type = "radio" name = "fcolor" value = "Red">Red
    <input type = "radio" name = "fcolor" value = "Green">Green
    <input type = "radio" name = "fcolor" value = "Blue">Blue
    <br><br><input type = "submit" value = "submit">
    </form>
```

cookie2.php

```
<?php
    if(isset($_COOKIE['firstname']))
    {
$user = $_COOKIE['firstname'];
       $color= $_COOKIE['fontcolor'];
    }
  else
    { $user =  $_POST['username'];
      $color = $_POST['fcolor'];
    }
?>
<html>
<head>
<title> setting username and font color using cookies </title>
</head>
<body>
    <h1> cookie stuff</h1>
    <hr>
  <h2>Hello: <?php echo $user; ?></h2>
    <h2>Your color: <?php echo $color ; ?></h2>
```

```
    <hr>
    <br/>
     <?php
    echo("<br/>");
    echo("<pre>");
    echo("Cookie info:\n");
    print_r($_COOKIE);
    echo("</pre>");
?>
```

4

Oct. 2017 – *4M*
Write a short note on Session
Oct.17, Apr. 17 – *2M*
How to check whether a variable is set with the session?
Oct. 2016 – *4M*
What is session? Explain with the help of suitable program.

6.2 Sessions

Sessions allow us to create multipage forms (such as shopping carts), save user authentication information from page to page, and store persistent user preferences on a site.

First-time each visitor is issued a unique session ID.

By default, the session ID is stored in a cookie called `PHPSESSID`. If the user's browser does not support cookies or has cookies turned off, the session ID is propagated in URLs within the web site.

Every session has a data store associated with it.

Register variables to be loaded from the data store when each page starts and saved back to the data store when the page ends.

Registered variables persist between pages, and changes to variables made on one page are visible from others.

Session Basics

To enable sessions for a page, we need to call `session_start()` before any of the document has been generated:

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

This assigns a new session ID if it has to, possibly creating a cookie to be sent to the browser, and loads any persistent variables from the store.

We can register a variable with the session by passing the name of the variable to `session_register()`.

For example, here is a basic hit counter:

```
<?php
session_start();
session_register('hits');
++$hits;
?>
```

This page has been viewed <?= $hits ?> times.

The `session_start()` function loads registered variables into the associative array `$HTTP_SESSION_VARS`.

The keys are the variables' names (*example*, `$HTTP_SESSION_VARS['hits']`).

If `register_globals` is enabled in the *php.ini* file, the variables are also set directly.

`session_unregister()`: unregister a variable from a session, which removes it from the data store.

`session_is_registered(`: function returns `true` if the given variable is registered.

`session_id(`: function returns the current session ID.

`Session_destroy(`: To end a session, it removes the data store for the current session, but it doesn't remove the cookie from the browser cache.

Consider the following example of registering the session variable,

```
<?php
session_start();
if (!isset($_SESSION['cnt']))
{
   $_SESSION['cnt'] = 0;
}
else
{
  $_SESSION['cnt']++;
}
?>
```

Unregister a session variable;

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

Example,

Counting the number of hits a user has made to a page

```
<?php
if (!session_is_registered('cnt'))
{
   session_register('cnt');
   $cnt = 1;
}
else
{
   $cnt++;
}
?>
<p>Hello visitor, you have seen this page <?php echo $cnt;?>
   times.</p>
<p>To continue, <a href="next.php?<?php echo strip_tags(SID); ?>">
  click here</a>.
// The strip_tags() is used when printing the SID in order to
prevent XSS related attacks.</p>
```

next.php

```
<?php
session_start(); ?>
<html>
<head>
      <title>Sessions</title>
</head>
<body>
    <hr>
        <h3>PHPSESSID = <?php echo session_id();?></h3>
    <hr>
    <h2>Click the next page and see number of visits during this
        visit.</h2>
<a href="next2.php?<?php echo(SID);?>">Next page</a>
</body>
</html>
```

next2.php

```
<?php
session_start();
($_SESSION['count'])? $_SESSION['count']++ : $_SESSION['count'] = 1;
?>

<html>
<head>
      <title>Sessions</title>
</head>
<body>
    <h1>session(page 2)</h1>
    <hr>
        <h3>PHPSESSID = <?php echo session_id();?></h3>
    <hr>
    <h2>You have been here <?php echo($_SESSION['count']);?>
        times in this session.</h2>
    <a href="next1.php?<?php echo(SID);?>">Previous page</a>
</body>
</html>
</html>
```

Alternatives to Cookies

The session ID is passed from page to page in the `PHPSESSID` cookie by default.

PHP's session system supports two alternatives:

i. Form fields and

ii. URL - PHP can rewrite the HTML files, adding the session ID to every relative link. To make this in work, PHP must be configured with the `-enable-trans-id` option when compiled.

Custom Storage

By default, PHP stores session information in files in your server's temporary directory. Session's variables are stored in a separate file. Every variable is serialized into the file in a proprietary format. And the changes to these can be made in the *php.ini* file.

The location of the session files can be changed by setting the `session.save_path` value in *php.ini*.

PHP can store session information in one of two formats in the current session store—either PHP's built-in format or WDDX. The format can be changed by setting the

`session.serialize_handler` value in your *php.ini* file.

We can write our own functions for reading and writing the registered variables. To install the custom session store.

i. Set `session.save_handler` to `user` in your *php.ini* file.

ii. Write functions for opening a new session, closing a session, reading session information, writing session information, destroying a session, and cleaning up after a session.

iii. And register them with the `session_set_save_handler()` function.

Syntax

```
session_set_save_handler(open_fn, close_fn, read_fn, write_fn,
destroy_fn, gc_fn);
```

To make all the PHP files within a directory to use our custom session store, set the following options in your *httpd.conf* file:

```
<Directory "/var/html/test">
php_value session.save_handler user
php_value session.save_path mydatabase
php_value session.name session_store
</Directory>
```

The following sample code uses a MySQL database for a session store

```
CREATE TABLE my_session_store
(
  mysession_id char(32) not null PRIMARY KEY,
  expire timestamp,
  value text not null
);
```

After creating all the handler functions for open,close,read,write and destroy, install them by calling `session_set_save_handler()` with the appropriate function names. With the preceding examples, call:

```
session_set_save_handler('open', 'close', 'read', 'write',destroy',
'gc');
```

You must call `session_set_save_handler()` before starting a session with `session_start()`. This is normally accomplished by putting the store functions and call to `session_set_save_handler()` in a file that's included in every page that needs the custom session handler.

6.3 Combining Cookies and Sessions

Combination of cookies and session handler can preserve state across visits. Any state that should be forgotten when a user leaves the site, any state that should persist between user visits, can be left up to PHP's built-in sessions.

Following *example* allows the user to select text and background colours and stores those values in a cookie. Any visits to the page within the next week send the colour values in the cookie.

```
<?php
if($_POST['bgcolor'])
{
setcookie('bgcolor', $_POST['bgcolor'], time() + (60 * 60 * 24 * 7));
}
$bgcolor = empty($bgcolor) ? 'gray' : $bgcolor;
?>
<body bgcolor="<?= $bgcolor ?>">
<form action="<?= $PHP_SELF ?>" method="POST">
<select name="bgcolor">
<option value="violet">Violet</option>
<option value="indigo">Indigo</option>
<option value="blue">Blue</option>
<option value="green">Green</option>
<option value="yellow">Yellow</option>
<option value="orange">Orange</option>
<option value="red">Red</option>
</select>
<input type="submit"/>
</form>
</body>
```

7. SSL

SSL, or Secure Sockets Layer, is an encryption-based Internet security protocol. It was developed by Netscape in 1995 for the purpose of ensuring privacy, authentication, and data integrity in Internet communications.

How does SSL work?

- In order to provide a high degree of **privacy**, SSL encrypts data that is transmitted across the web. This means that anyone who tries to intercept this data will only see a garbled mix of characters that is nearly impossible to decrypt.
- SSL initiates an **authentication** process called a handshake between two communicating devices to ensure that both devices are really who they claim to be.
- SSL also digitally signs data in order to provide **data integrity**, verifying that the data is not tampered with before reaching its intended recipient.

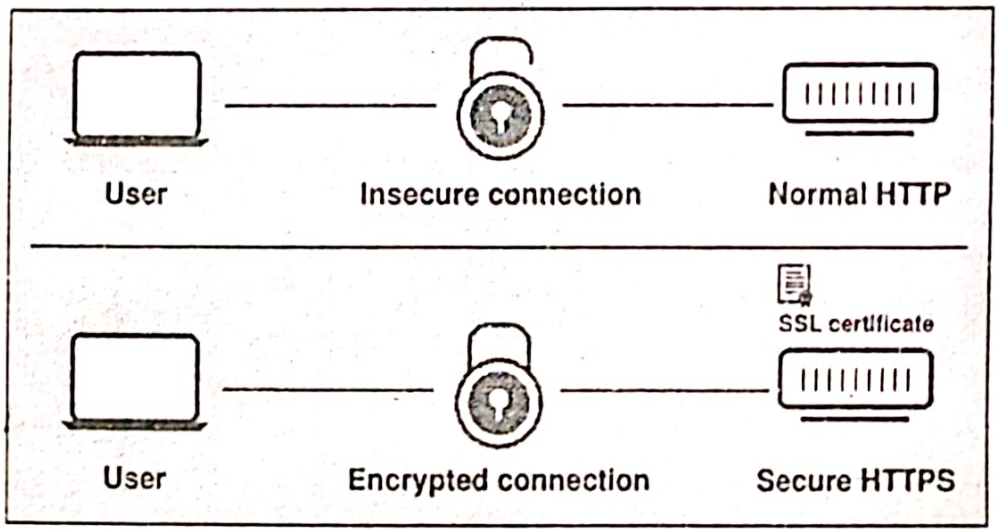

Figure 4.1: HTTP vs HTTPS

An https:// URL indicates a secure connection for that document, unlike an http:// URL.

The HTTPS entry in the $_SERVER array is set to 'on' if the PHP page was generated in response to a request over an SSL connection.

To prevent a page from being generated over a nonencrypted connection, simply use:

```
if ($_SERVER['HTTPS'] !== 'on')
{
  die("Secure Connection !!!!");
}
```

Solved Programs

1. Write php program to accept student rno, name on page1.php and marks of 3 subjects on page2.php and display students all information on page3.php.

Solution

Accepting Rno and Name Details

(page1.php)

```
<html>
<body>
<form action="page2.php" method="post">
Enter rno:<input type =text name="t1">
Enter name:<input type =text name="t2">
<input type="submit" value="Send">
</form>
</body>
</html>
```

Setting rno and name with cookies, Accepting marks of 3 subjects

(page2.php)

```
<html>
<body>
<?php
$rno=$_POST['t1'];
$name=$_POST['t2'];
setcookie('rno', $rno);
setcookie('name', $name);
echo"cookies are set";
?>
<form action="page3.php" method="post">
Enter Subject1 marks<input type="text" name="t1">
Enter Subject2 marks<input type="text" name="t2">
```

```
Enter Subject3 marks<input type="text" name="t3">
<input type="submit" value="send">
</form>
</body>
</html>
```

Printing rno and name from cookies, Displaying marks of 3 subjects

(page3.php)

```
<html>
<body>
<?php
$rno=$_COOKIE['rno'];
$name=$_COOKIE['name'];
$sub1=$_POST['t1'];
$sub2=$_POST['t2'];
$sub3=$_POST['t3'];
$total=$sub1+$sub2+$sub3;
$average=$total/3;
echo"Student details <br>";
echo "Rollno : $rno<br>";
echo "Name : $name<br>";
echo "Marks of Subect1 : $sub1<br>";
echo "Marks of Subject2 : $sub2<br>";
echo "Marks of Subect3 : $sub3<br>";
echo"Total : $total<br>";
echo"Average : $average<br>";
?>
</body>
</html>
```

SUMMARY

- PHP is designed as web scripting language, develops dynamic websites which have forms, sessions, and self processing pages.
- The http controls web browser requests files from web servers and how the servers send the files back. The method: GET, POST, HEAD tells the server how to handle requests.
- Server configuration and request information can be obtained via the variables.
- The $_SERVER array gives useful information from the web server.
- Processing the form is possible via the global array $_POST and $_GET.
- Automatic quoting of parameters is done by **Magic quotes** a PHP feature that automatically quotes input data.
- Self processing pages can be used for both generating a form and process it.
- Sticky form is a technique in which the result of a query is accompanied by a search form whose default values are those of the previous queries.
- Multiple values can be passed for a single field using the [ ].
- File upload feature of PHP can be used to upload images or related materials.
- Form validation allows users to input data to validate that data before using it or storing it for later use.
- Setting response headers via: headers.
- Session management is used to help web applications maintain their state across several HTTP requests when needed.
- 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.

Exercises

A. Answer the following questions: **[1 Mark]**

1. What is the use of phpinfo()?
2. Which function would you use to start a session?
3. How can you associate a variable with a session?
4. List any two Php HTTP functions..
5. How to check whether a variable is set with a session?
6. What is HTTP?
7. Which arrays are used to access form parameters from php code.
8. Write short note on cookies.
9. How can we destroy the session, how can we unset the variable of a session?
10. How we can get the Cookie Values and destroy the cookie?
11. How can we know that a session is started or not?
12. Describe ways in which I can register the variables into session.
13. What function would you use to redirect the browser to a new page?
14. How do you get the user's IP address in PHP?

B. Answer the following questions: **[5 Marks]**

1. Compare between GET method and POST method.
2. Write short note on sticky form.
3. What is sticky form? Explain sticky form with suitable example.

4. Write php program to accept student rno, name on page1.php and marks of 3 subjects on page2.php and display students all information on page3.php.

5. Explain how to combine session and cookies.

6. Write a short note on Cookies.

7. Which arrays are used to access form parameters from php code.

8. What is the difference between GET and POST methods in php? Explain it with proper example.

9. Create a calculator script that allows the user to submit two numbers and choose an operation to perform on them (addition, multiplication, division, subtraction).

10. Explain how Cookie can be set.

Questions Asked in Previous Year Exams

2 Marks

1. What is $http_response_header? [Apr. 2019]
2. What is SetCookie() function? [Apr. 2019]
3. Enlist the HTTP Request methods. [Apr. 2018]
4. What is setcookie() function? [Apr. 18]
5. Name any two variables of global array. [Oct. 2017]
6. How to check whether a variable is set with the session? [Oct. 17, Apr. 17]
7. Name any two variables of $_SERVER array. [Apr. 2017]
8. List any two PHP HTTP function. [Apr. 2017]
9. What is $_SERVER? [Oct. 2016]

4 Marks

1. How to handle file upload in php? [Apr. 2019]
2. To create form that accept the user details. Write php program to capitalize of first letter of each name and check user e-mail address contain @ symbol. [Apr. 2019]
3. Create student registration form and display details in the next page. (Use sticky form concept). [Apr. 2019]
4. Write a PHP script to upload the file and display its information (Use $_FILES). [Apr. 2019]
5. Write short notes on:
   i. Redirection [Apr. 19, 18, Oct. 16]
   ii. Multivalued parameter [Apr. 2018]
   iii. Setting response header [Oct. 17, Apr. 17]
   iv. Session [Oct. 2017]

6. What is self-processing form? Explain with the help of program. [Apr. 2018]

7. Write a PHP program to accept two strings from user and check whether entered strings are matching or not? (Use sticky form concept). [Apr. 2018]

8. What is self-processing form? Explain with the help of program. [Oct. 2017]

9. Write a PHP script to upload the file and display its information (Use $_FILES). [Oct. 2017]

10. Write a php program to accept two strings from user and check whether entered strings are matching or not (use sticky form concept). [Oct. 2017]

11. Explain cookie with example. [Oct. 17, Apr. 17]

12. What is sticky form? Explain with example. [Apr. 2017]

13. Write a php program to upload the file and display its information (use $_FILES). [Apr. 2017]

14. Write a php program to accept two strings from user and check whether entered strings are matching or not (use sticky form concept). [Apr. 2017]

15. Write a php program to capitalize of first letter of each name and check user email address contain @ symbol. [Apr. 2017]

16. What are the response header? Give various ways of setting response header. [Oct. 2016]

17. Create a login form with a username and password. Once the user logs in, the second form should be displayed to accept user details enter information within a specified time limit, expire his session and give a warning otherwise display details ($-SESSION). [Oct. 2016]

18. What is session? Explain with the help of suitable program. [Oct. 2016]

19. Write a PHP program to upload the file and display its information (use $-FILES). [Oct. 2016]

20. Write a PHP program to accept two strings from user and check whether entered strings are matching or not. (use sticky form concept). [Oct. 2016]