LAB 5

PHP BASICS

PART-II

September 08, 2022

FUNCTIONS in PHP

- Functions are a way to group common tasks or calculations to be reused easily.
- A function takes one more input in the form of parameter, does some processing and returns (not always) a value.
- There are more than 1000 of built-in functions in PHP

•Defining a function in PHP:

```
function sum($x, $y)
{
        $z = $x + $y;
        return $z;
}
```

•Calling a function in PHP:

```
$result = sum($a, 5);
print('I am human, I am.');
```

Note : echo is not a function.

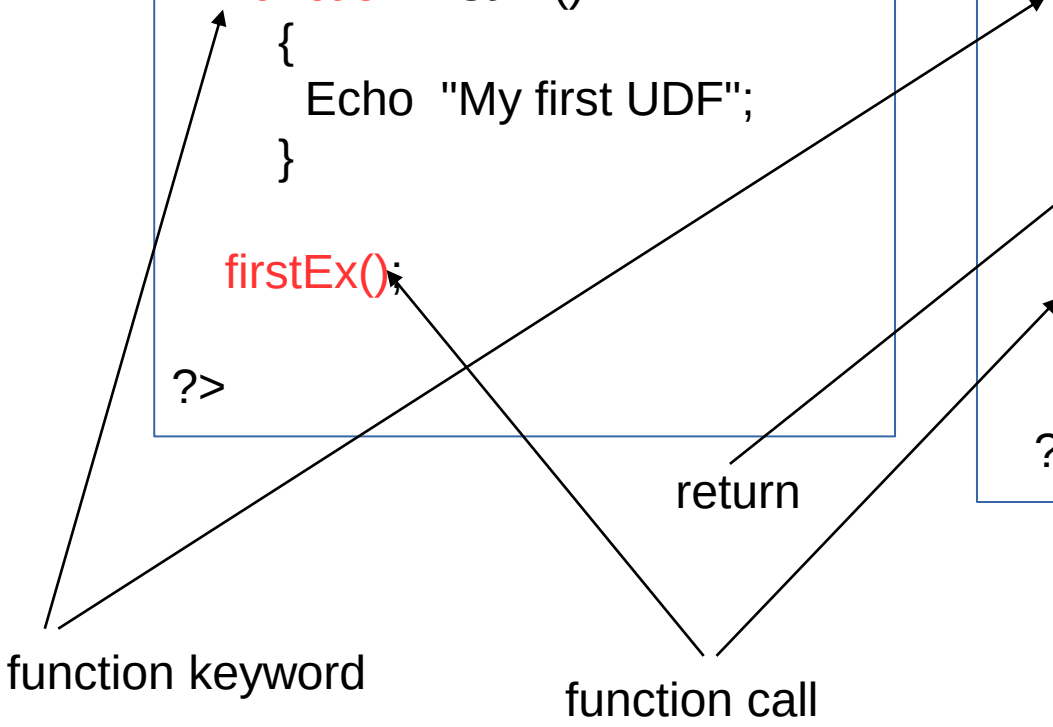
FUNCTIONS (Contd…)

Parameter less

```
<?php

    function firstEx()
      {
        Echo  "My first UDF";
      }

    firstEx();

?>
```

Parameterized

```
<?php

    function sum($val1,$val2)
      {
        $sum=$val+$val2;
        return($sum);
      }

    $result = sum(12,23);
    echo "Sum of Inputs is:",$result;
  ?>
```



FUNCTIONS (Contd…)

- PHP allows you to give parameters default values when declaring the function.

```
function print__strings ($var1 = "Hello World", $var2 = "I'm Learning PHP")
                        {
                                echo($var1);
                                echo("\n");
                                echo($var2);
                        }
```

- Function Call : `print_strings("hello");`
- Output :

```
hello
I'm Learning PHP
```

FUNCTIONS - Dynamic Parametrization

- Another way to have a dynamic number of parameters is to use built-in functions:
  - **func_num_args**
  - **func_get_args**
  - **func_get_arg**

```php
<?php
  function mean()
{
    $sum = 0;
    $param_count = func_num_args();

   for(i = 0; $i < $param_count; $i++)
            $sum += func_get_arg($i);

     $mean = $sum / $param_count;
     echo "<br> Mean: {$mean}";
}
     mean(10,20,30,40);
     mean(12.5,34,45.7,78,123,56);
   ?>
```

```php
<?php
  function mean()
{
          $sum=0;
          $all_args=func_get_args();

          foreach($all_args as $si)
              $sum=$sum+$si;

     $mean=$sum/count($all_args);

 echo "<br> Mean For each type: $mean";
}
      mean(10,20,30,40);
      mean(12.5,34,45.7,78,123,56);
    ?>
```

FUNCTIONS (Contd…)

- If you want to return multiple variables you need to return an array rather than a single variable. For example:

```
function maths ($x, $y)
{
        $z  =  $x + $y ;
        $w =  $x - $y ;
        $ret = array ("tot"=>$z, "diff"=>$w);
        return $ret;
}
```

- When calling this from your script you need to call it into an array.

```
$returnVal = maths (10, 5);
```

$returnVal['tot'] will be the total 15, while $returnVal['diff'] will be the difference 5 in above case.

Try this PHP code to get familiar with functions.

```
<?php
echo "<b>" .add_numbers(3). "</b><br/>";
echo "<b>" .add_numbers(3, 2,5). "</b><br/>";
echo "<b>" .divide(3,2.0). "</b><br/>";

function add_numbers($var1 = 0, $var2 = 0, $var3 = 0)
   {
          return $var1 + $var2 + $var3;
   }
function divide($dividee, $divider)
   {
          if($divider == 0)
           {
                 //can't divide by 0
                  return false;
           }

          $result = $dividee/$divider;
          return $result;
   }
?>
```

OUTPUT:

```
3
10
1.5
```

FUNCTIONS and VARIABLES

- Variables: declaration anywhere in the script
  - Scope of a variable :
    - Local: use prohibited outside a function.
    - Global: use in a function is prohibited.
      - global keyword
      - $GLOBALS
    - Static: memory state retained across function calls.

Ex: Scope of a variable

```
<?php
     $global_var="I am Global";
     function _funsum()
       {
         //global $global_var;                                    // (2) un,(3) com
         //$global_var="i am not global,I am LOCAL";        // (1) un,(2) com. (3) un
         echo " In the function", $global_var;
         //echo " Inside the function $GLOBALS[global_var]";    // (3)un
       }
     _funsum();
     Echo "Valued printed outside the function::".$global_var;
  ?>
```

```
<?php
    function count_mycalls()
      {
        static $countme=0;
        $countme++;
        echo "<br>COUNT TO BE PRINTED".$countme;
      }
     for($ti=1;$ti<10;$ti++)
        count_mycalls();
 ?>
```

Date and Time (1)

- Three functions:
  - time()
    - Returns the current time in the number of seconds elapsed since Unix Epoch (January, 1 , 1970 00:00:00 GMT). 32 bit integer value
  - getdate()
    - Returns an associative array with information related to the timestamp.
  - date()
    - Formats a timestamp to a more readable date and time.
    - *date (format, timestamp);*

Date and Time (2)

```
<?php
   $current=time();
    echo "The current time is: $current\n"; ?>
```

Output like :
The current time is:
1506302668

```
<?php
  $time1 = time();
  sleep(10);
  $time2 = time();
  $x = $time2 - $time1;
  echo "Difference is: “ .$x. “ seconds";
?>
```

Output:
Difference is : 10 seconds.

Output:
10/02/12
10-02-12
12:10:25

```
<?php
  echo date("d/m/y") . "<br/>";
  echo date("d-m-y") . "<br/>";
  echo date("h:m:s") . "<br/>“; ?>
```

Date and Time (4) - strtotime() Function

- strtotime(*timestring);* It parses an English textual datetime into a Unix timestamp (the number of seconds since January 1 1970 00:00:00 GMT).
- In the m/d/y or d-m-y formats;
  - If the separator is a slash (/), then the American m/d/y is assumed.
  - If the separator is a dash (-) or a dot (.), then the European d-m-y format is assumed.

```
<?php
echo(strtotime("now") . "<br>");
echo(strtotime("3 October 2005") . "<br>");
echo(strtotime("+5 hours") . "<br>");
echo(strtotime("+1 week") . "<br>");
echo(strtotime("+1 week 3 days 7 hours 5 seconds") . "<br>");
echo(strtotime("next Monday") . "<br>");
echo(strtotime("last Sunday")); ?>
```

Output in secs

```
1567754102
1128297600
1567772102
1568358902
1568643307
1567987200
1567296000
```

Date and Time (5) - Formatted

```
<?php
	$_1=strtotime('1981-12-17');
	echo date('Y-m-d,H:i:s',$_1)."<br>";
	echo date('l, F jS,Y,H:i:s',$_1)."<br>";
	echo date('l, F jS,Y,H:i:s')."<br>";
	echo date('c')."<br>";
	echo date('r')."<br>";
?>
```

Learn about formatting of date()

Output:

```
1981-12-17,00:00:00
Thursday, December 17th,1981,00:00:00
Monday, September 25th,2017,11:31:09
2017-09-25T11:31:09+05:30
Mon, 25 Sep 2017 11:31:09 +0530
```

Date and Time (6) – Another Example

```
<?php
    $my_DoB=strtotime("2013/06/19 4:23:15");
    $DoB_array = getdate($my_DoB);

    foreach($DoB_array as $key => $val)
            print "$key = $val <br>";


     echo "<br>--------------------<br>";
     echo "My DoB is :"
          .$DoB_array['mday']. "/"
          .$DoB_array['mon']. "/“
          .$DoB_array['year'];
     echo "<br>--------------------<br>";
 ?>
```

```
seconds = 15
minutes = 23
hours = 4
mday = 19
wday = 3
mon = 6
year = 2013
yday = 169
weekday = Wednesday
month = June
0 = 1371615795


--------------------
My DoB is :19/6/2013
--------------------
```

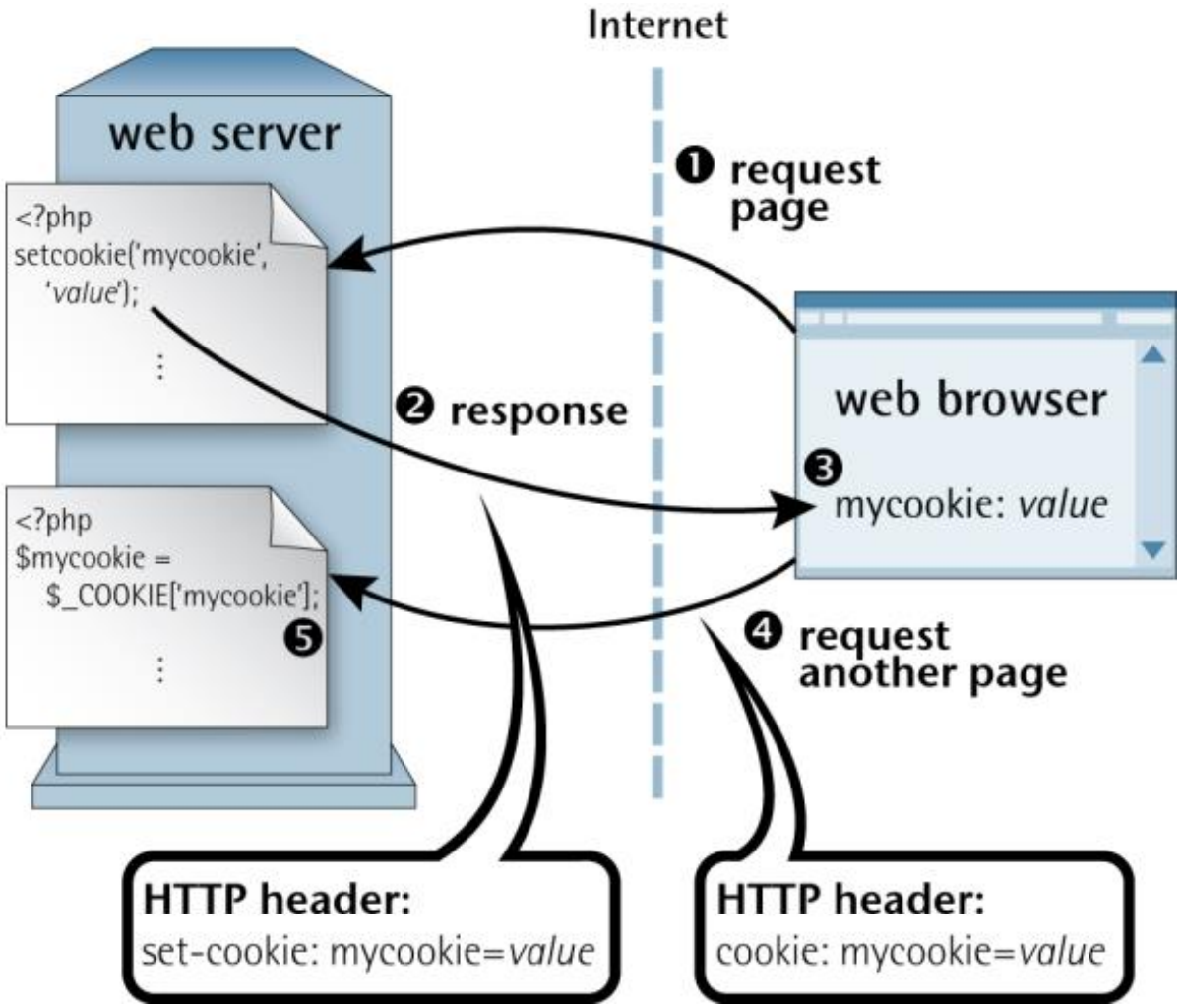
- Cookies
- Sessions
- Files
- Errors

COOKIES

- A cookie is a small piece of information given to a Web browser by a Web server. The browser stores the information in a text file. This information is then sent back to the server each time the browser requests a page from the server.
- Cookies can live on a client's computer for as long as a developer has decided it to remain. This means unlike sessions, if you close browser, cookies can remain in your computer.
- Cookies are generally used to store usernames and user preferences for specific websites which he doesn't want to retype each time he access pages from that website.
- Sensitive information like authentication information is not stored using cookies as they are modifiable on local machines.

COOKIES - Mechanism

- Used to identify a user



COOKIES - USE

- Cookies are most commonly used to track website activity. When you visit some sites, the server gives you a cookie that acts as your identification card. Upon each return visit to that site, your browser passes that cookie back to the server. In this way, a web server can gather information about which web pages are used the most, and which pages are gathering the most repeat hits.

- Cookies are also used for online shopping. Online stores often use cookies that record any personal information you enter, as well as any items in your electronic shopping cart, so that you don't need to re-enter this information each time you visit the site.

- Servers can use cookies to provide personalized web pages. When you select preferences at a site that uses this option, the server places the information in a cookie. When you return, the server uses the information in the cookie to create a customized page for you.

COOKIES

- Cookies are stored on client's machine using setcookie() function. This function should always called before the <html> tag.
- **Setcookie (name, value, expiration, path, domain, security)**
- Name - This sets the name of the cookie and is stored in an environment variable called HTTP_COOKIE_VARS. This variable is used while accessing cookies.
- Value - This sets the value of the named variable and is the content that you actually want to store.
- Expiry - This specify a future time in seconds. After this time cookie will become inaccessible. If this parameter is not set then cookie will automatically expire when the Web Browser is closed.

Setcookie (name, value, expiration, path, domain, security)

- Path - This specifies the directories for which the cookie is valid. A single forward slash character permits the cookie to be valid for all directories.
- Domain - This can be used to specify the domain name in very large domains and must contain at least two periods to be valid. All cookies are only valid for the host and domain which created them.
- Security - This can be set to 1 to specify that the cookie should only be sent by secure transmission using HTTPS otherwise set to 0 which mean cookie can be sent by regular HTTP.
- Example:

```
setcookie('name','user', time()+60);
```

- To access the cookie variable name

```
$username = $_COOKIE['user'];
```

Setting and Accessing Cookies

- setcookie() function: setcookie(cookie_name,cookie_value);
- Accessing
  - $_COOKIE superglobal variable, remains active across the pages

```
//Setcookie example
<?php
 $cookie_name='user_color';
 $cookie_value='red';
 setcookie($cookie_name,$cookie_value);
//Accessing cookie using $_COOKIE
echo "<br> user color is".$_COOKIE[$cookie_name];
echo "<br> user color is". $_COOKIE['user_color'];
?>
```

COOKIE Example 1

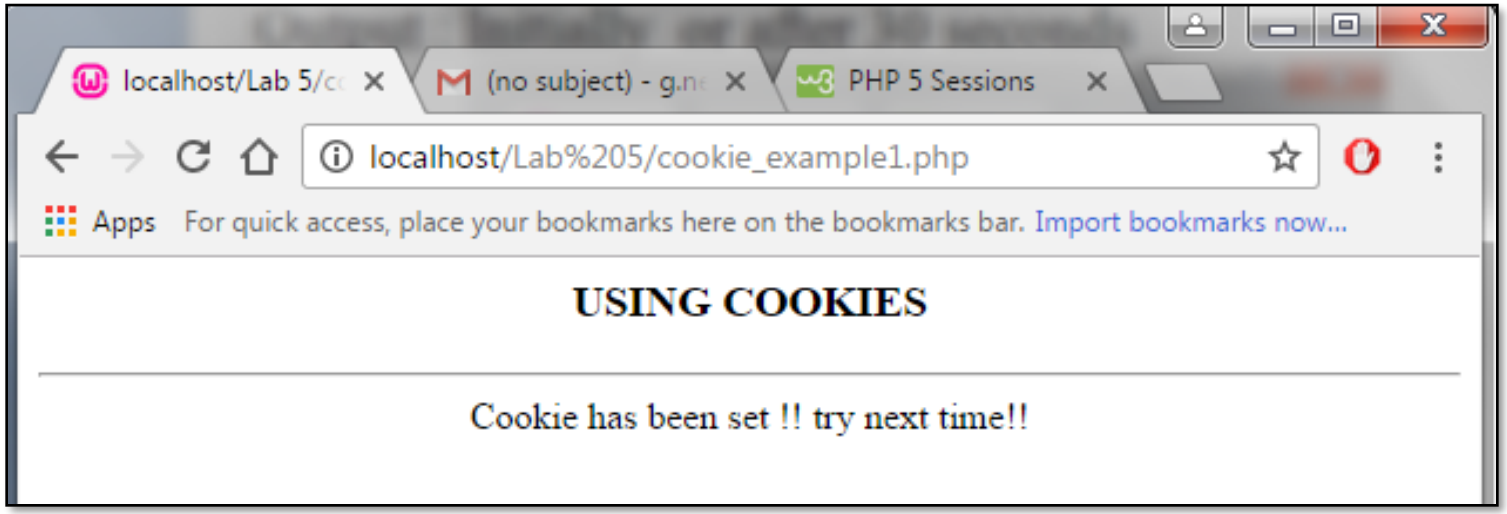
- Following PHP code checks the life time of a cookie.

```
<center><h3>USING COOKIES</h3></center>
<hr/>
<?php

$cookie_name = "user";
$cookie_value = "PHP Programmer";

if (isset($_COOKIE[$cookie_name]))
        echo "<center>Welcome
<b>".$_COOKIE[$cookie_name]."</b>!</center> <br />";
else
{
        setcookie($cookie_name,$cookie_value,time()+30);
        echo "<center>Cookie has been set !! try next
time!!</center>";
} ?>
```

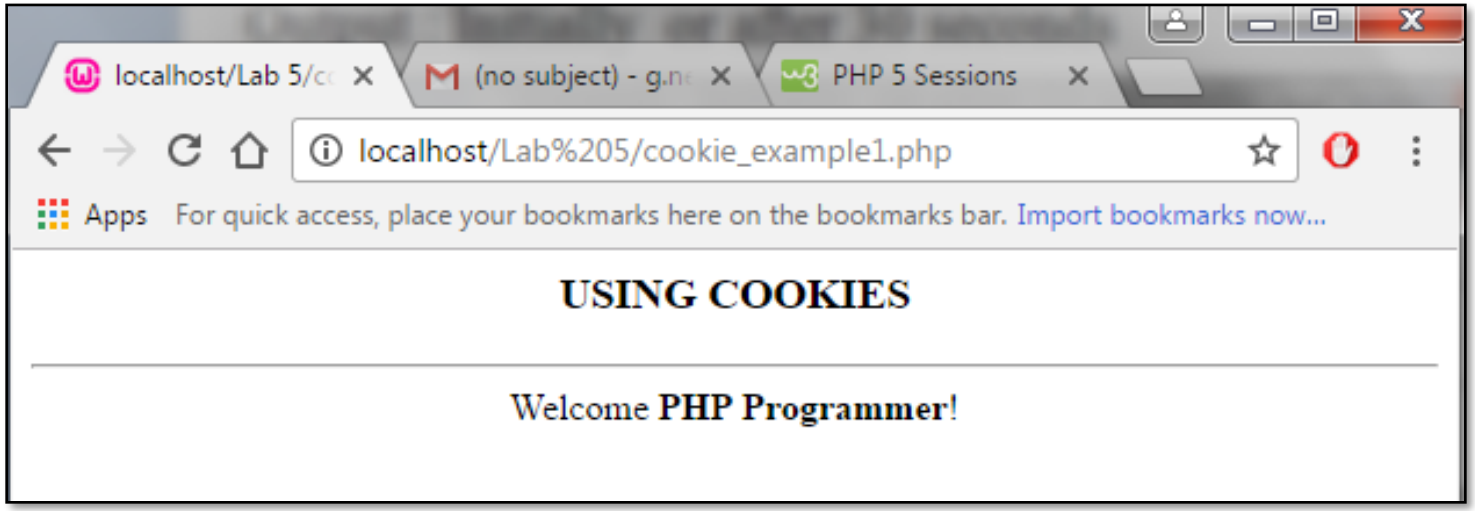
The isset() function checks whether a variable is set and is not NULL. This function also checks if a declared variable, array or array key has null value, if it does, isset() returns false, it returns true in all other possible cases.

COOKIE Example 1

- Output : Initially or after 30 seconds



Output : After first attempt and before 30 seconds elapsed

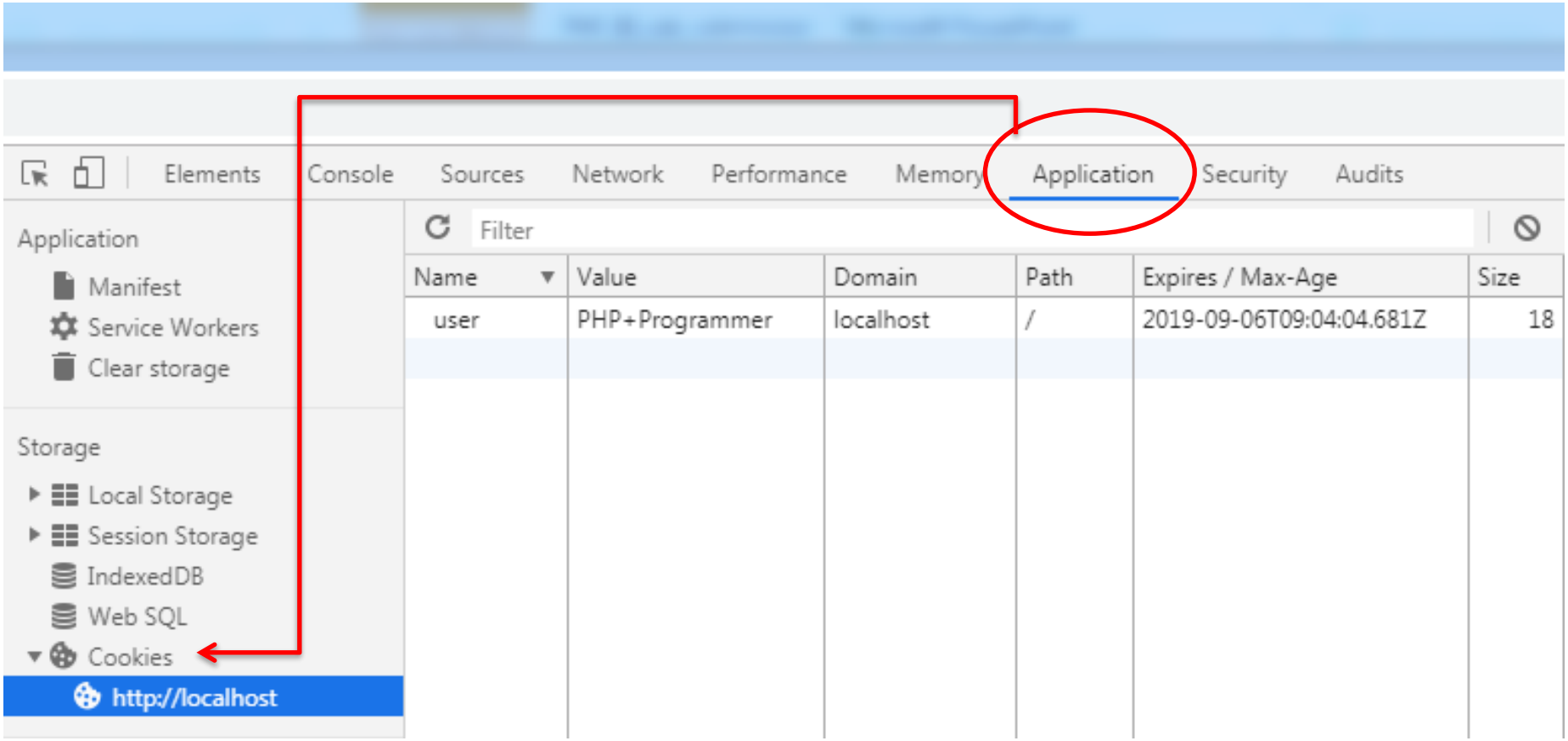


COOKIE Check for Example 1

1) Right click on your php output (webpage) on browser.

2) Select “Inspect element” option.

3) Go to “Application” tab and on left toolbar click on “Cookies” option to see the details of cookie set by you for this webpage.



COOKIE Example 2

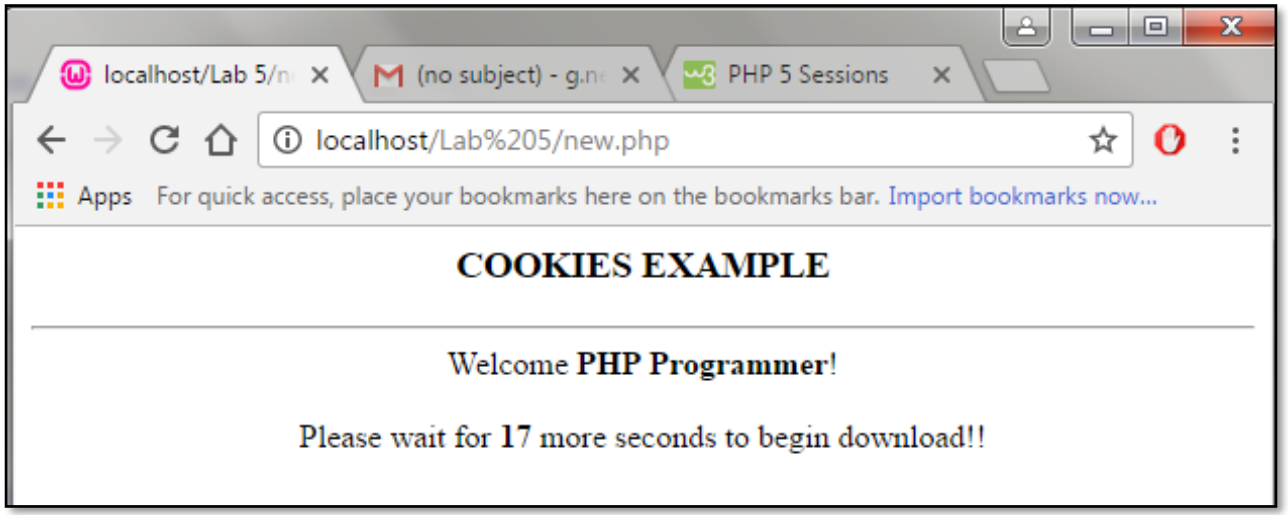
- Following PHP code checks the time in a cookie stored on clients machine and if it is found to be more than 20 seconds, allows him to download file. Otherwise it tells him to wait for remaining time.

```
<center><h3>COOKIES EXAMPLE</h3></center>
<hr/>
<?php
$cookie_name = "user";
$cookie_value = "PHP Programmer";
if (isset($_COOKIE["time"])){
        $past=$_COOKIE['time'];
        $now=time();
        $diff=$now-$past;
        echo "<center>Welcome
<b>".$_COOKIE[$cookie_name]."</b>!</center> <br />";
        if($diff > 20)
         echo "<center><a href='sample1.txt'>click
here to download file</a>!</center> <br />";
```

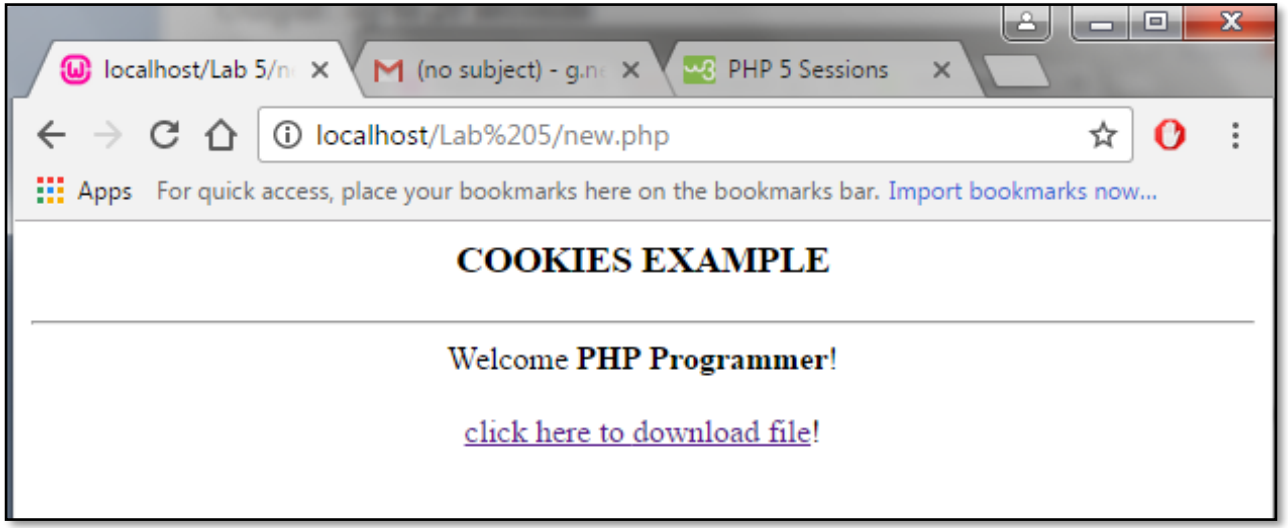
```
else{
        $left=20-$diff;
 echo "<center>Please wait for <b>".$left."</b>
 more seconds to begin download!!</center>
 <br />";
        }
}
else{
  setcookie($cookie_name,$cookie_value,time()
  +120);
  setcookie("time",time(),time()+120);
  echo "<center>Cookie has been set !! try next
  time!!</center>";
}
?>
```

Cookie Example-2

- Output: up to 20 seconds



Output: After 20 seconds elapsed



Cookies

- Delete a cookie：

  setcookie(‘user_color’,’ ’,time()-3600);

- Limitation:
  - Stored in insecure plain text format
  - Restriction on amount of data
- Donot place sensitive information in cookies
- Use it for
  - User identifier
  - User preferences
- Use cookies as tokens to manage the users.

SESSION

- It is basically a temporary set of variables whose values persist across subsequent web pages.
- How long they exist ?
  - Default : User closes the browser window
  - Session’s maximum time allotment is exceeded
  - A call to session_destroy() function.
- Session variables hold information about one single user, and are available to all pages in one application.

SESSION

- Every session is assigned a unique session ID, which keeps all the current information together.
- Session ID can be passed through the URL or through the use of cookies.
- session_start( )
  - Position : top of every script for which we want our session to be active
  - Initialization
    - Session cookies assignment
    - Initializes the session storage
- Setting session variables:
  - PHP super global array variable $_SESSION
  - contains currently registered variables to a script's session

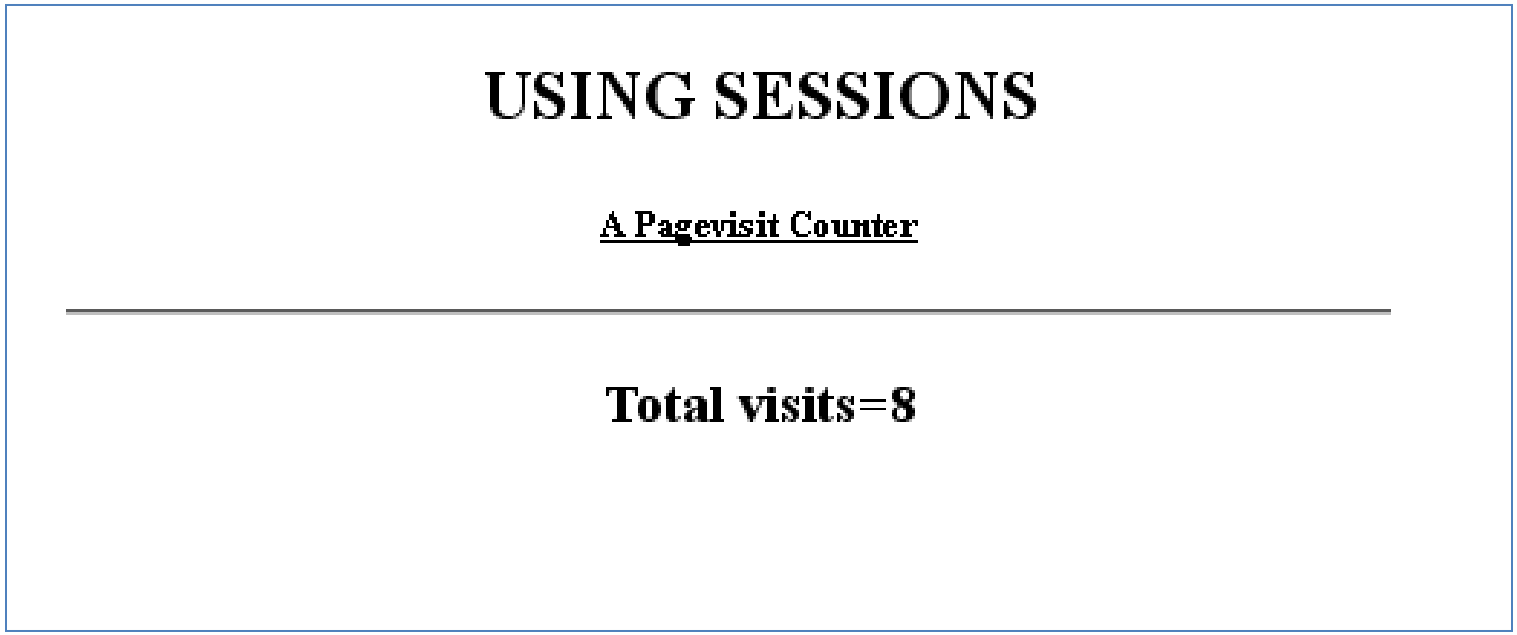
SESSION EXAMPLE 1

This PHP script displays the number of times a user accessed this page

```
<center><h2>USING SESSIONS</h2></center>
<center><h5><u>A Pagevisit Counter</u></h5></center>
<hr/>
<?php
session_start();
if(isset($_SESSION['visits']))
{
        $_SESSION['visits']=$_SESSION['visits']+1;
        if($_SESSION['visits']>20)
        {
                session_destroy();
                exit();
        }
}
else
        $_SESSION['visits']=1;
echo "<center><h3>Total visits=". $_SESSION['visits']."</h3></center>";
?>
```

SESSION EXAMPLE 1

… and here is the output of the previous script:



SESSION EXAMPLE 2
AUTHENTICATION THROUGH SESSION

Save this file as friend_request.php and execute

```
<?php
        session_start();
        $_SESSION['username'] = 'amit jain';
        $_SESSION['authuser'] = 1;

?>

<body>
        <a href = "friends.php"> Click here to view my friends</a>
</body>
```

SESSION EXAMPLE 2 (Contd..)

```
<?php
        session_start();
        //find out whether user has logged in with valid password
        if(isset($_SESSION['authuser']))
        {
         if($_SESSION['authuser']!=1)    {
                echo 'Sorry!! you do not have permission to view this page!';
                exit();                         }
        }
        else{
                echo 'You need to login to view this content!<br/>';
                echo 'Go to login page!';
                exit();
        }

?>
        <b>Hello PHP Programmer!!</b><br/>
        <b>Your friends are:</b>
        <table border=2>
        <tr><td>Suresh</td></tr>
        <tr><td>Anil</td></tr>
        <tr><td>Aditya</td></tr>
        <tr><td>Sanjay</td></tr>
        <tr><td>Shaan</td></tr>
        </table><br/><br/>
   <b>Thanks to visit your friends!!</b>
</body>
```

Save this file as friends.php

AUTHENTICATION THROUGH SESSION (Contd…)

You will find following output if session is valid:

FILES

- Files are useful for :
  - Retrieving remote web pages for local processing
  - Storing data without a database
  - Saving information that other programs need access to
- Use fopen(filename,mode) to open files

```
<?php
$file1=fopen(“myfile.txt”, “r”) or exit(“unable to open file”);
?>
```

- Use fclose( ) function to close an open file.

```
<?php
   $file1 = fopen(“myfile.txt","r");
   //some code to be executed
   fclose($file1);
      ?>
```

FILES

- Listed below are various file opening modes.

Mode	Purpose
r	Opens the file for reading only. Places the file pointer at the beginning of the file.
r+	Opens the file for reading and writing. Places the file pointer at the beginning of the file.
w	Opens the file for writing only. Places the file pointer at the beginning of the file. and truncates the file to zero length. If files does not exist then it attempts to create a file.
w+	Opens the file for reading and writing only. Places the file pointer at the beginning of the file. and truncates the file to zero length. If files does not exist then it attempts to create a file.
a	Opens the file for writing only. Places the file pointer at the end of the file. If files does not exist then it attempts to create a file.
a+	Opens the file for reading and writing only. Places the file pointer at the end of the file. If files does not exist then it attempts to create a file

FILES

- Summary:

Mode	Readable?	Writeable?	File pointer	Truncate?	Create?
r	Yes	No	Beginning	No	No
r+	Yes	Yes	Beginning	No	No
w	No	Yes	Beginning	Yes	Yes
w+	Yes	Yes	Beginning	Yes	Yes
a	No	Yes	End	No	Yes
a+	Yes	Yes	End	No	Yes

Files : Important functions

bool **feof** (file)
checks if the "end-of-file" (EOF) has been reached.

string **fgets** (file [,length])
Gets a line from file pointer

string **fgetc** (file)
Gets a character from the given file pointer.

bool **file_exists** (filename)
Checks whether a file or directory exists.

bool **unlink** (filename [, context])
Deletes a file

int **fwrite** (file , string [, length])
Writes the contents of string to the file stream pointed to by handle .

int **fputs** (file , string [, length])
The fputs() function is an alias of the fwrite() function.

File example 1

Read and display the data from comma separated file.

```
<html>
<?php
 $fpointer=fopen("name.txt","r") or exit("Unable to open read file!");

 echo "<table align= \"center\" border=10 >";
 echo  "<tr> <td> Name </td>
        <td> Surname </td>
        <td> Branch </td>
        <td> Year   </td></tr>";

 while(!feof($fpointer))
  {
  $temp=fgets($fpointer);
  $temp=explode(",",$temp);

  if($temp[0]!='')
  { echo "<tr> <td>".$temp[0]. "</td>
        <td>".$temp[1]. "</td>
        <td>".$temp[2]. "</td>
        <td>".$temp[3]. "</td></tr>";}
 }
echo "</table>";
 ?>
</html>
```

fileprocess.php

```
sarita,patel,CSE,4
suman,kaushal,DESIGN,3
amit,pal,CSE,4
shubham,darr,ME,2
ramnagar,choubey,CSE,1
atif,khan,ME,3
john,martin,DESIGN,2
```

Data.txt

File example : read and write

```
<html>
 <head>   <title> FILE EXAMPLE </title>  </head>

 <body>
  <?php
   $file1 = fopen("hello.txt", "r") or exit("Unable to open read file!");
   $file2 = fopen("hello_out.txt", "w+") or exit("Unable to open write file!");

  //Output a line of the file until the end is reached

  while(!feof($file1))
   {
     $line=fgets($file1);
     fputs($file2,$line);
   }
  fclose($file2);
  fclose($file1);

  echo "<center>File written successfully. Now opening hello_out.txt for reading:</center><br/>";

  $file2=fopen("hello.txt","r") or exit("Unable to open file!");
  while (!feof($file2))
   {
     echo fgets($file2);   //use fgetc
     echo "<br>";
   }
  fclose($file2);
  ?>
 </body>
</head>
```

ERRORS

- Three main types of PHP script errors:
  - Fatal Errors
    - Program cannot recover from them. Execution is halted.
  - Warnings
    - Non fatal errors. Execution is continued
  - Notices
    - This could indicate an error or it could also happen in normal course of execution of a script.

- User level error handling
  - trigger_error( ) function
  - Using set_error_handler()

ERRORS : Non Fatal Errors

You can see in example below that execution continues even if a we try to divide a number by 0.

```
<center><h2>Error Display</h2></center>
<center><h5><u>Nonfatal Error</u></h5></center>
<hr/>
<center>
<?php
$x=$_GET['x'];
$y=3;

$z=$y/$x;
echo "division = $z <br/>";
$z=$y*$x;
echo "Multiplication = $z";
?>
</center>
```

Calling script by:

http://localhost/error.php?x=0

Error Display

Nonfatal Error

(!) Warning: Division by zero in C:\wamp\www\error2.php on line 9

Call Stack

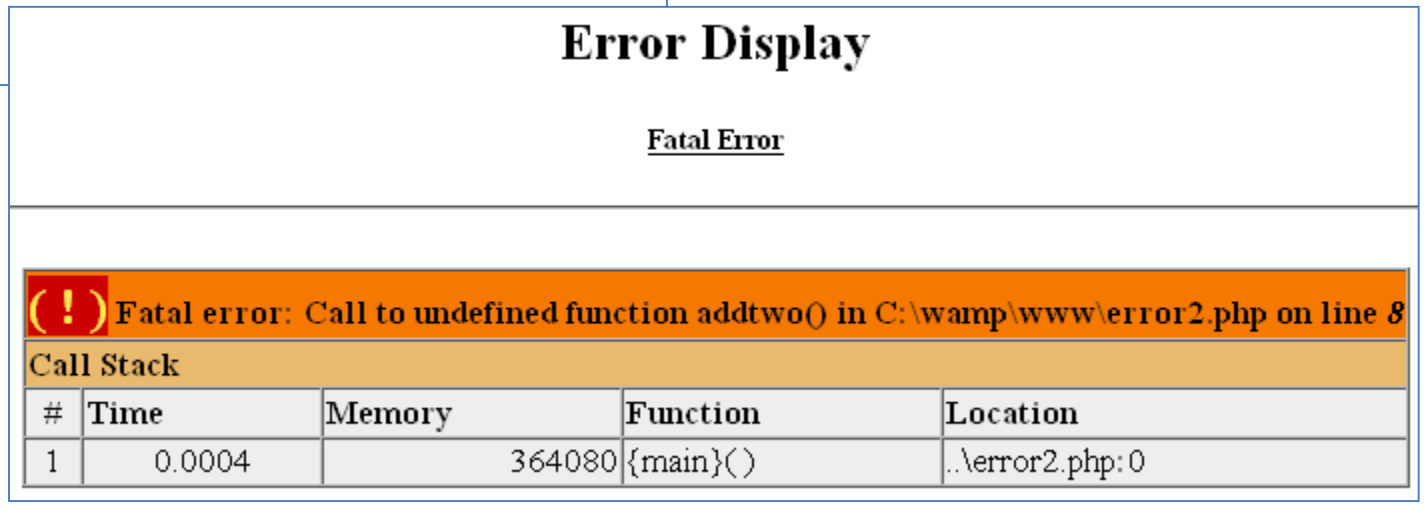
#	Time	Memory	Function	Location
1	0.0094	364960	{main}( )	..\error2.php: 0

division =
Multiplication = 0

ERRORS : Fatal Errors

Non existent function addtwo() is called which generates a fatal error and line below the function call does not execute.

```
<center><h2>Error Display</h2></center>
<center><h5><u>Fatal Error</u></h5></center>
<hr/>
<center>
<?php
$x=3;
$y=4;
$z=addtwo($x,$y);
echo "addition = $z";
?>
</center>
```



Error Display

Fatal Error

(!) Fatal error: Call to undefined function addtwo() in C:\wamp\www\error2.php on line *8*

Call Stack				
#	Time	Memory	Function	Location
1	0.0004	364080	{main}()	..\error2.php:0

ERRORS : Notices

You can see in example below that execution continues even if a we try to access a non existent key in $_GET[] array.

```
<center><h2>Error Display</h2></center>
<center><h5><u>Non-fatal Error</u></h5></center>
<hr/>
<center>
<?php
if ($_GET['name']=='amit')
 echo "Hello amit!! welcome to website";
else
 echo "Hello Guest!! welcome to website.<br/>";

 echo "guests dont need authentication.";
?>
</center>
```

Error Display

Non-fatal Error

(!) Notice: Undefined index: name in C:\wamp\www\error1.php on line *6*

Call Stack

#	Time	Memory	Function	Location
1	0.0009	363144	{main}()	..\error1.php:0

Hello Guest!! welcome to website.
guests dont need authentication.

Error handling using trigger_error( )

- Syntax:

trigger_error (errormsg , errortype);

Parameter	Description
errormsg	Required. Specifies the error message for this error. Max 1024 bytes in length
errortype	Specifies the error type for this error. Possible values: •E_USER_ERROR •E_USER_WARNING •E_USER_NOTICE (default)

Example Code:

```
<?php
$divisor=0;
if ($divisor == 0) {
    trigger_error("Cannot divide by zero",
     E_USER_ERROR);
}
?>
```

- The trigger_error() function creates a user-level error message.
- It can be used with
  - the built-in error handler, or
  - with a user-defined function set by the set_error_handler() function.

ERRORS : Custom Error Handlers

- Syntax of custom error handler is given below:

  error_function_name (error_level, error_message [, error_file, error_line, error_context])

Parameter	Description
error_level	Required. Specifies the error report level for the user-defined error. Must be a value number.
error_message	Required. Specifies the error message for the user-defined error
error_file	Optional. Specifies the filename in which the error occurred
error_line	Optional. Specifies the line number in which the error occurred
error_context	Optional. Specifies an array containing every variable, and their values

- The default error handler for PHP is the built in error handler.
- Use set_error_handler to make a function the default error handler for the duration of the script.

  set_error_handler (“error_function_name");

ERRORS : Custom Error Handler

Following code depicts an error handler customized by a programmer.

```
<?php
//error handler function
function customError($errno, $errstr, $errfile, $errline)
{
 echo "<h2>OOPS!!</h2>";
 echo "<p>Errors have occured while executing this page. Contact the <a
          href='mailto:admin@iiitdmj.ac.in'>Administrator</a>
          to report it.</p><hr/>";
 echo "<b>Error Type</b>: $errno<br/>";
 echo "<b>Error Message</b>: $errstr<br/>";
 echo "<b>File Name</b>: $errfile<br/>";
 echo "<b>Line Number</b>: $errline<br/>";  }

//set error handler
set_error_handler("customError");

//trigger error
echo($test);
?>
```

ERRORS : Custom Error Handler

… and here is the output of the customized error handling script:

OOPS!!

Errors have occured while executing this page. Contact the Administrator to report it.

Error Type: 8
Error Message: Undefined variable: test
File Name: /users/student/phd/2011/cse/1110162/public_html/error4.php
Line Number: 18

ERRORS : Error levels

Value	Constant	Description
2	E_WARNING	Non-fatal run-time errors. Execution of the script is not halted
8	E_NOTICE	Run-time notices. The script found something that might be an error, but could also happen when running a script normally
256	E_USER_ERROR	Fatal user-generated error. This is like an E_ERROR set by the programmer using the PHP function trigger_error()
512	E_USER_WARNING	Non-fatal user-generated warning. This is like an E_WARNING set by the programmer using the PHP function trigger_error()
1024	E_USER_NOTICE	User-generated notice. This is like an E_NOTICE set by the programmer using the PHP function trigger_error()
4096	E_RECOVERABLE_ERROR	Catchable fatal error. This is like an E_ERROR but can be caught by a user defined handle (see also set_error_handler())
8191	E_ALL	All errors and warnings, except level E_STRICT (E_STRICT will be part of E_ALL as of PHP 6.0)

Assignment-7

Logo image

Bank Name (Big Font-use header tags)

Dashboard | Open A/C | Loan | Deposit

(Use table structure to make menu option)

Dashboard

List of Depositors

Name	A/C No.	Balance	Details
		Click to View	

List of Borrowers

Name	A/C No.	Balance	Details
		Click to View	