PHP

Mrs.V.Mareeswari

Assistant Professsor

School of Information Technology and Engineering

VIT University

Cabin No:SJT 210-A30

Introduction

- PHP is a server scripting language, and is a powerful tool for making dynamic and interactive Web pages quickly.
- PHP is a widely-used, free, and efficient alternative to competitors such as Microsoft's ASP.
- PHP is an acronym for "PHP Hypertext Preprocessor"
- PHP is a widely-used, open source scripting language.
- PHP scripts are executed on the server.
- PHP costs nothing, it is free to download and use.

What is a PHP File?

- PHP files can contain text, HTML, CSS, JavaScript, and PHP code.
- PHP code are executed on the server, and the result is returned to the browser as plain HTML.
- PHP files have extension ".php“.
- With PHP you are not limited to output HTML. You can output images, PDF files, and even Flash movies. You can also output any text, such as XHTML and XML.
- It is powerful enough to be at the core of the biggest blogging system on the web (WordPress)!
- It is deep enough to run the largest social network (Facebook)!

Why PHP?

- PHP runs on various platforms (Windows, Linux, Unix, Mac OS X, etc.)
- PHP is compatible with almost all servers used today (Apache, IIS, etc.)
- PHP supports a wide range of databases.
- PHP is free. Download it from the official PHP resource: www.php.net
- PHP is easy to learn and runs efficiently on the server side.

Basic Syntax

- A PHP script can be placed anywhere in the document.
- A PHP script starts with **<?php** and ends with **?>**

```
<html> <body>
<h1>My first PHP page</h1>
<?php                                   ➡  Don't give space
echo "Hello World!<br>";;
echo "Introduction", " to". " PHP"; →comma & dot for concatenation
?>                                      ➡  Don't give space
</body> </html>
```

Type program → Notepad
Save program →c:\wamp\www\basics.php
In Lab:c or d:\Appserv\www\PROGRAM\basics.php
Run in web browser → http://localhost/basics.php

Comments

```
<?php
   // This is a single line comment

   # This is also a single line comment

   /*
   This is a multiple lines comment block
   that spans over more than
   one line
   */
?>
```

Case Sensitivity

- In PHP, all user-defined functions, classes, and keywords (e.g. if, else, while, echo, etc.) are NOT case-sensitive.
- <?php
  ECHO "Hello World!
";
  echo "Hello World!
";
  EcHo "Hello World!
";
  ?>

- However; in PHP, all variables are case-sensitive.
- <?php
  $color="red";
  echo "My car is " . $color . "
";
  echo "My pen is " , $COLOR;
  echo "My boat is " . $coLOR;
  ?>

Output:

- My car is red
- Error : Undefined variable

Rules for PHP variables

- A variable starts with the $ sign, followed by the name of the variable.
- A variable name must start with a letter or the underscore character.
- A variable name cannot start with a number.
- A variable name can only contain alpha-numeric characters and underscores (A-z, 0-9, and _).
- Variable names are case sensitive ($y and $Y are two different variables).
- **Note:** When you assign a text value to a variable, put quotes around the value. Eg: $color="red"

PHP is a Loosely Typed Language

- PHP has no command for declaring a variable.
- A variable is created the moment you first assign a value to it:

```
<?php
$txt="Hello world!";
$x=5;
$y=10.5;
?>
```

- In the example above, notice that we did not have to tell PHP which data type the variable is.
- PHP automatically converts the variable to the correct data type, depending on its value.
- In other languages such as C, C++, and Java, the programmer must declare the name and type of the variable before using it.

Variables Scope

- In PHP, variables can be declared anywhere in the script.
- The scope of a variable is the part of the script where the variable can be referenced/used.
- PHP has three different variable scopes:
  - Local
  - global
  - static

```
<?php
$x=5; // global scope
function myTest() {
  $y=10; // local scope
  echo "<p>Test variables inside the function:</p>";
  echo "Variable x is: $x";
  echo "<br>";
  echo "Variable y is: $y";
}
myTest();
echo "<p>Test variables outside the function:</p>";
echo "Variable x is: $x";
echo "<br>";
echo "Variable y is: $y";
```

- A variable declared **outside** a function has a GLOBAL SCOPE and can only be accessed outside a function.
- A variable declared **within** a function has a LOCAL SCOPE and can only be accessed within that function.

Output

Notice: Undefined variable: x in **C:\wamp\www\basics.php** on line **28**

Variable x is:

Variable y is: 10

Test variables outside the function:

Variable x is: 5

Notice: Undefined variable: y in **C:\wamp\www\basics.php** on line **38**

Variable y is:

The global Keyword

- The global keyword is used to access a global variable from within a function.

```
<?php
  $x=5;
  $y=10;

  function myTest() {
    global $x,$y;
    $y=$x+$y;
  }

  myTest();
  echo $y; // outputs 15
  ?>
```

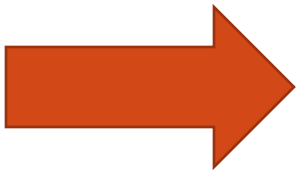
- PHP also stores all global variables in an array called $GLOBALS[*index*]. The *index* holds the name of the variable. This array is also accessible from within functions and can be used to update global variables directly.

```
<?php
$x=5;
$y=10;
function myTest() {
  $GLOBALS['y']=
$GLOBALS['x']+$GLOBALS['y'];
}
myTest();
echo $y; // outputs 15
?>
```

The static Keyword

```
<?php
function myTest() {
  static $x=0;
  echo $x;
  $x++;
}
myTest();
myTest();
myTest();

?>
```



- Normally, when a function is completed/executed, all of its variables are deleted. However, sometimes we want a local variable NOT to be deleted. We need it for a further job.
- Then, each time the function is called, that variable will still have the information it contained from the last time the function was called.
- **Note:** The variable is still local to the function.

echo and print Statements

- There are some differences between echo and print:
  - echo - can output one or more strings
  - print - can only output one string, and returns always 1

Tip: echo is marginally faster compared to print as echo does not return any value.

- echo and print is a language construct, and can be used with or without parentheses: echo or echo() and print or print().

Example:

- echo "This", " string", " was". " made", " with multiple parameters.";
- echo "Study PHP at $txt2";
- print "Study PHP at $txt2";

Data types

String, Integer, Floating point numbers, Boolean, Array, Object, NULL.

Examples:

- $x = "Hello world!";
- $x = 'Hello world!';
- $x = 5985; var_dump($x); → int(5985)
- $x = -345; // negative number
- $x = 0x8C; // hexadecimal number
- $x = 047; // octal number
- $x = 10.365; $x = 2.4e3;
- $x=true; $y=false;
- $x=null;

The PHP var_dump() function returns the data type and value of variables.

String Functions

- strlen("Hello world!") → 12
- strpos("Hello world!","world") → 6
- str_shuffle("Hello World")→ loerll WdoH
- str_replace("world","Peter","Hello world!")→Hello Peter!
- str_split("Hello")→a string into an array
- str_split("Hello",3)→Array ([0] => Hel [1] => lo)
- str_word_count("Hello world!")→Count the number of words
- strcmp("Hello world!","Hello world!")→case-sensitive comparison
- strcasecmp("Hello","HELLO")→ case-insensitive comparison
- strncmp("Hello world!","Hello earth!",6)→ first n chars (case-sensitive)
- strpos("I love php, I love php too!","php")→7 → pos of first occurrence
- substr("Hello world",6)→ world
- substr("Hello world",1,8)→ ello wor
- strtolower(), strtoupper()
- substr_count("Hello world. The world is nice","world")→2

Constant

- Constants are automatically global across the entire script.
- The first parameter defines the name of the constant,
- the second parameter defines the value of the constant,
- and the optional third parameter specifies whether the constant name should be case-insensitive. Default is false.

```
<?php
  define("GREETING", "Welcome", true);
  echo GREETING; //Welcome
  echo "<br>";
  echo greeting; //Welcome   → greeting is printed when sets false
?>
```

Operators

- $txt1 = "Hello“ ;
  $txt2 = $txt1 . " world!“; →"Hello world!“ → Concatenation
- $txt1 = "Hello“;
  $txt1 .= " world!“; →"Hello world!“→ Concatenation assignment
- $x=100; $y="100";
  ($x == $y); // returns true because values are equal
  ($x === $y); // returns false because types are not equal
  ($x != $y); // returns false because values are equal // $x <> $y
  ($x !== $y); // returns true because types are not equal

Array Operators

$x = array("a" => "red", "b" => "green");
$y = array("c" => "blue", "d" => "yellow");
$z = $x + $y; // union of $x and $y
$z → red,green,blue,yellow
$x == $y → false
$x === $y → false
$x != $y → true
$x <> $y → true
$x !== $y → true

- Control Statements: if,if-else,if-elseif-else, switch
- Loop Statements: while, do-while,for, foreach

```
<html><body>
<?php
$t=date("H");
if ($t<"10")
{   echo "Have a good
morning!";}
 elseif ($t<"20")
{    echo "Have a good day!"; }
else
{    echo "Have a good night!";}
?>
 </body></html>
```

```
echo date("d/m/y"),"<br>";
echo date('H')."<br>";
echo date('l')."<br>";
```

foreach

- The foreach loop works only on arrays, and is used to loop through each key/value pair in an array.
-
```
<?php
$colors = array("red","green","blue","yellow");

foreach ($colors as $value) {
  echo "$value <br>";
}
?>
```

red
green
blue
yellow

Date Functions

- d - The day of the month (from 01 to 31)
- l (lowercase 'L') - A full textual representation of a day
- w - A numeric representation of the day (0 for Sunday, 6 for Saturday)
- m - A numeric representation of a month (from 01 to 12)
- M - A short textual representation of a month (three letters)
- F - A full textual representation of a month (January through December)
- Y - A four digit representation of a year (2014)
- y - A two digit representation of a year (14)
- a - Lowercase am or pm A - Uppercase AM or PM
- h - 12-hour format of an hour (01 to 12) H - 24-hour format of an hour (00 to 23)

Arrays

- **Numeric array** - An array with a numeric index. Values are stored and accessed in linear fashion
- **Associative array** - An array with strings as index. This stores element values in association with key values rather than in a strict linear index order.
- **Multidimensional array** - An array containing one or more arrays and values are accessed using multiple indices

Numeric Array

```
<html> <body>
<?php
/* First method to create array. */
$numbers = array( 1, 2, 3, 4, 5);
foreach( $numbers as $value )
{
echo "Value is $value <br />";
}
```

```
/* Second method to create array.
*/
$num[0] = "one";
$num[1] = "two";
$num[2] = "three";
foreach( $num as $value )
{ echo "Value is $value <br />";
}
?>
</body> </html>
```

Associative array

```
/* First method to create array. */
$salaries = array( "Mohammad" => 2000, "John" => 1000, "Kumar" => 500 );
echo "Salary of Mohammad is ". $salaries['Mohammad'] . "<br>";
echo "Salary of John is ". $salaries['John']. "<br>";
echo "Salary of Kumar is ". $salaries['Kumar']. "<br>";
```

```
/* Second method to create array. */
$salaries['Mohammad'] = "high";
$salaries['John'] = "medium";
$salaries['Kumar'] = "low";
echo "Salary of Mohammad is ". $salaries['Mohammad'] . "<br >";
echo "Salary of John is ". $salaries['Kumar']. "<br>";
```

Multidimensional Array

```
$marks = array( "mohammad" => array ( "physics" => 35, "maths" => 30, "chemistry" => 39 ), "john" => array ( 10,20,30 ), "kumar" => array ( "physics" => 31, "maths" => 22, "chemistry" => 39 ) );

 /* Accessing multi-dimensional array values */
echo "Marks for mohammad in physics : " ;
echo $marks['mohammad']['physics'] . "<br />";
echo "Marks for john in maths : ";
echo $marks['john'][1] . "<br />";  //numerical array
echo "Marks for kumar in chemistry : " ;
echo $marks['kumar']['chemistry'] ; //associative array
```

Sorting

- sort() - sort arrays in ascending order
- rsort() - sort arrays in descending order
- asort() - sort associative arrays in ascending order, according to the value
- ksort() - sort associative arrays in ascending order, according to the key
- arsort() - sort associative arrays in descending order, according to the value
- krsort() - sort associative arrays in descending order, according to the key

Superglobals -regardless of scope - and you can access them from any function, class or file without having to do anything special.

- $GLOBALS
- $_SERVER
- $_REQUEST
- $_POST
- $_GET
- $_FILES
- $_ENV
- $_COOKIE
- $_SESSION

FormHandling.html

```
<html>
<body>
<form action="FormHandling.php" method="post">
Name: <input type="text" name="name"><br>
E-mail: <input type="text" name="email"><br>
<input type="submit" value="Click Here">
</form>
</body>
</html>
```

FormHandling.php

```
<html>
<body>
Welcome <?php echo $_POST["name"]; ?><br>
Your email address is: <?php echo $_POST["email"]; ?>
</body>
</html>
```

$_GET["name"]

http://localhost/FormHandling.html

Welcome Mareeswari
Your email address is: vmareeswari@vit.ac.in

Validation

```
<?php
   if (empty($_POST["name"]))
      echo "Name is Missing";
   else
      echo "Welcome", $_POST["name"];

   if(isset($_POST["gender"]))
      echo "Your gender is:",$_POST["gender"];
?>
```

File Inclusion

- You can include the content of a PHP file into another PHP file before the server executes it.
  - The include() Function
  - The require() Function
- This is a strong point of PHP which helps in creating functions, headers, footers, or elements that can be reused on multiple pages.
- This will help developers to make it easy to change the layout of complete website with minimal effort. If there is any change required then instead of changing thousand of files just change included file.

File Inclusion

- You can include the content of a PHP file into another PHP file before the server executes it.
  - The include() Function
  - The require() Function
- This is a strong point of PHP which helps in creating functions, headers, footers, or elements that can be reused on multiple pages.
- This will help developers to make it easy to change the layout of complete website with minimal effort. If there is any change required then instead of changing thousand of files just change included file.

Using include function

Example.php

```
<html><body>
<h1>Welcome to my home page!</h1>
<p>Some text.</p>
<p>Some more text.</p>
<?php include 'footer.php';?>
</body></html>
```

footer.php

```
<?php
echo "<p>Copyright &copy; 1999-" . date("Y") . "
Tutorials.com</p>";
?>
```

Output

Welcome to my home page!

Some text.

Some more text.

Copyright © 1999-2014 Tutorials.com

include

- Use **include** when the file is not required and application should continue when file is not found.
- include will only produce a warning (E_WARNING) and the script will continue.

```
<?php include 'noFileExists.php';
echo "I have a $color $car.";
?>
```

require

- Use **require** when the file is required by the application.
- require will produce a fatal error (E_COMPILE_ERROR) and stop the script.

```
<?php require
'noFileExists.php';
echo "I have a $color $car.";
?>
```

Exercise

- Assume we have a standard menu file called "menu.php“. All pages in the Web site should use this menu file. How it can be implemented?
- Assume we have a file called "vars.php", with some variables defined. How it can be used by calling file?

File Handling

- The readfile() function reads a file and writes it to the output buffer.

```
<?php
echo readfile("dictionary.txt");
?>
```

```
<?php
$myfile = fopen("webdictionary.txt", "r") or die("Unable to open file!");
echo fread($myfile,filesize("webdictionary.txt"));
fclose($myfile);
?>
```

The first parameter of fread() contains the name of the file to read from and the second parameter specifies the maximum number of bytes to read.

Modes	Description
r	Open a file for read only. File pointer starts at the beginning of the file
w	Open a file for write only. Erases the contents of the file or creates a new file if it doesn't exist. File pointer starts at the beginning of the file
a	Open a file for write only. The existing data in file is preserved. File pointer starts at the end of the file. Creates a new file if the file doesn't exist
x	Creates a new file for write only. Returns FALSE and an error if file already exists
r+	Open a file for read/write. File pointer starts at the beginning of the file
w+	Open a file for read/write. Erases the contents of the file or creates a new file if it doesn't exist. File pointer starts at the beginning of the file
a+	Open a file for read/write. The existing data in file is preserved. File pointer starts at the end of the file. Creates a new file if the file doesn't exist
x+	Creates a new file for read/write. Returns FALSE and an error if file already exists

fgets() & fgetc

```
// Output one line until end-of-file
while(!feof($myfile)) {
  echo fgets($myfile) . "<br>";
}

// Output one character until end-of-file
while(!feof($myfile)) {
  echo fgetc($myfile);
}
```

File Writing

```
<?php
$myfile = fopen("newfile.txt", "w") or die("Unable to
open file!");
$txt = "John Doe\n";
fwrite($myfile, $txt);
$txt = "Jane Doe\n";
fwrite($myfile, $txt);
fclose($myfile);
?>
```

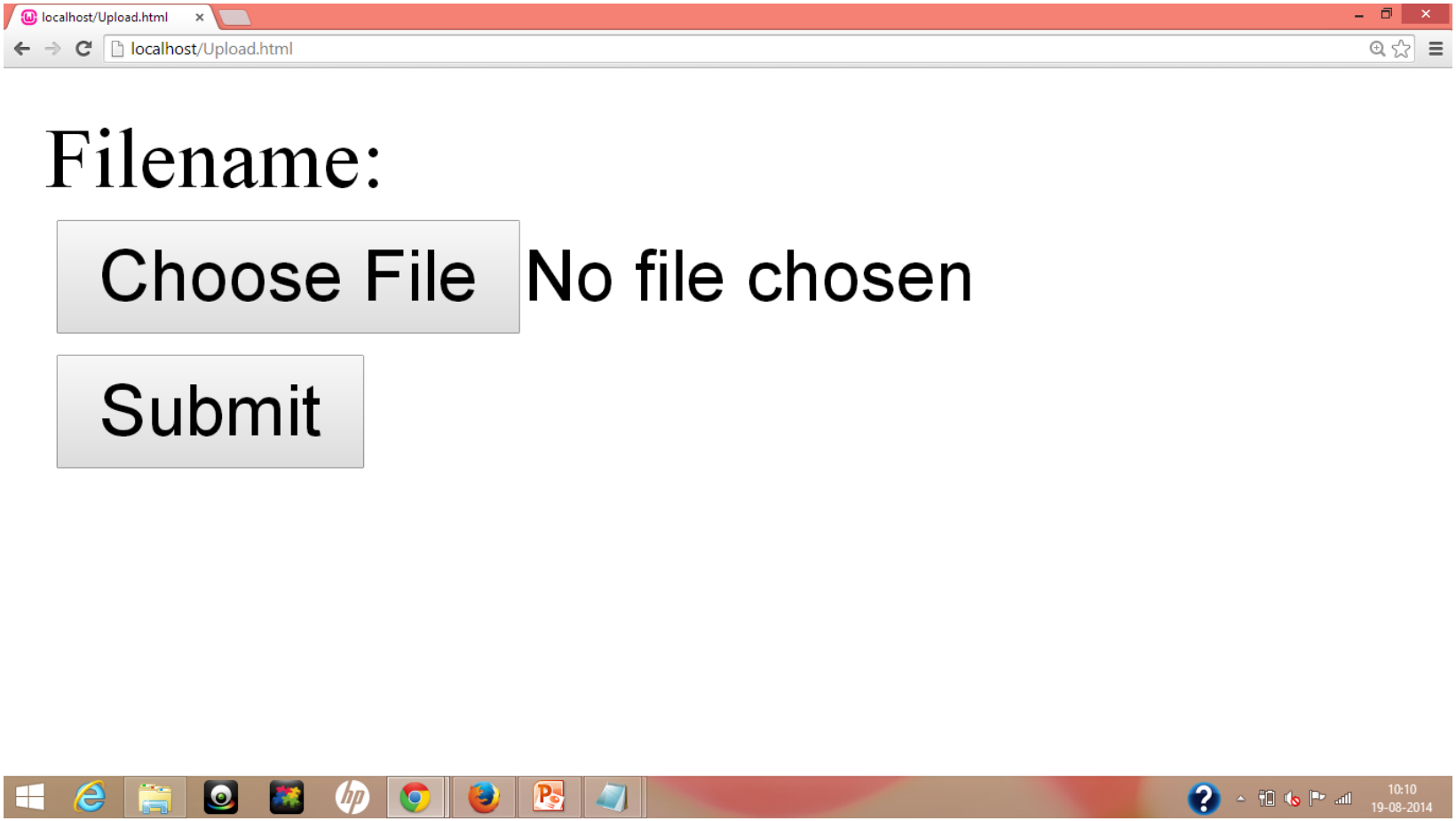
- copy(file,to_file)
- dirname() → Returns the directory name component of a path
- fflush()→Flushes buffered output to an open file
- file_exists()
- file_get_contents()→ reads a file into a string
- file_put_contents()→writes a string into a file
- fileatime()→Returns the last access time of a file
- filemtime()→Returns the last modification time of a file
- fileowner()→Returns the user ID (owner) of a file
- fputs(),fwrite()
- fseek()→Seeks in an open file
- is_dir(),is_file()
- nename()
- unlink()→delete a file

Upload files to the server

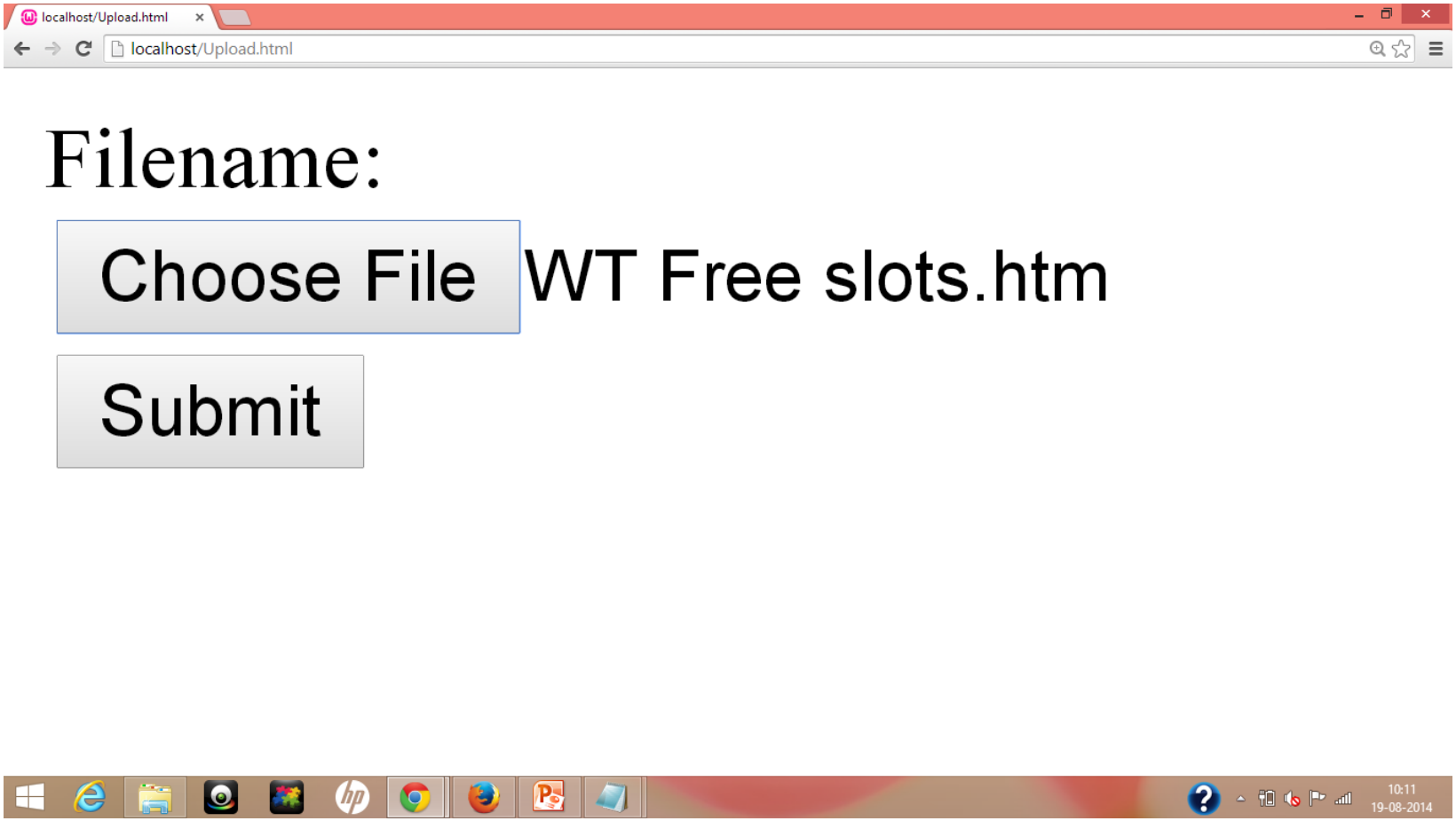
```
<html> <body>
<form action="upload_file.php" method="post"
enctype="multipart/form-data">
Filename:
<input type="file" name="file" ><br>
<input type="submit" name="submit" value="Submit">
</form></body></html>
```

The enctype attribute of the <form> tag specifies which content-type to use when submitting the form. "multipart/form-data" is used when a form requires binary data, like the contents of a file, to be uploaded

http://localhost/Upload.html



File selection



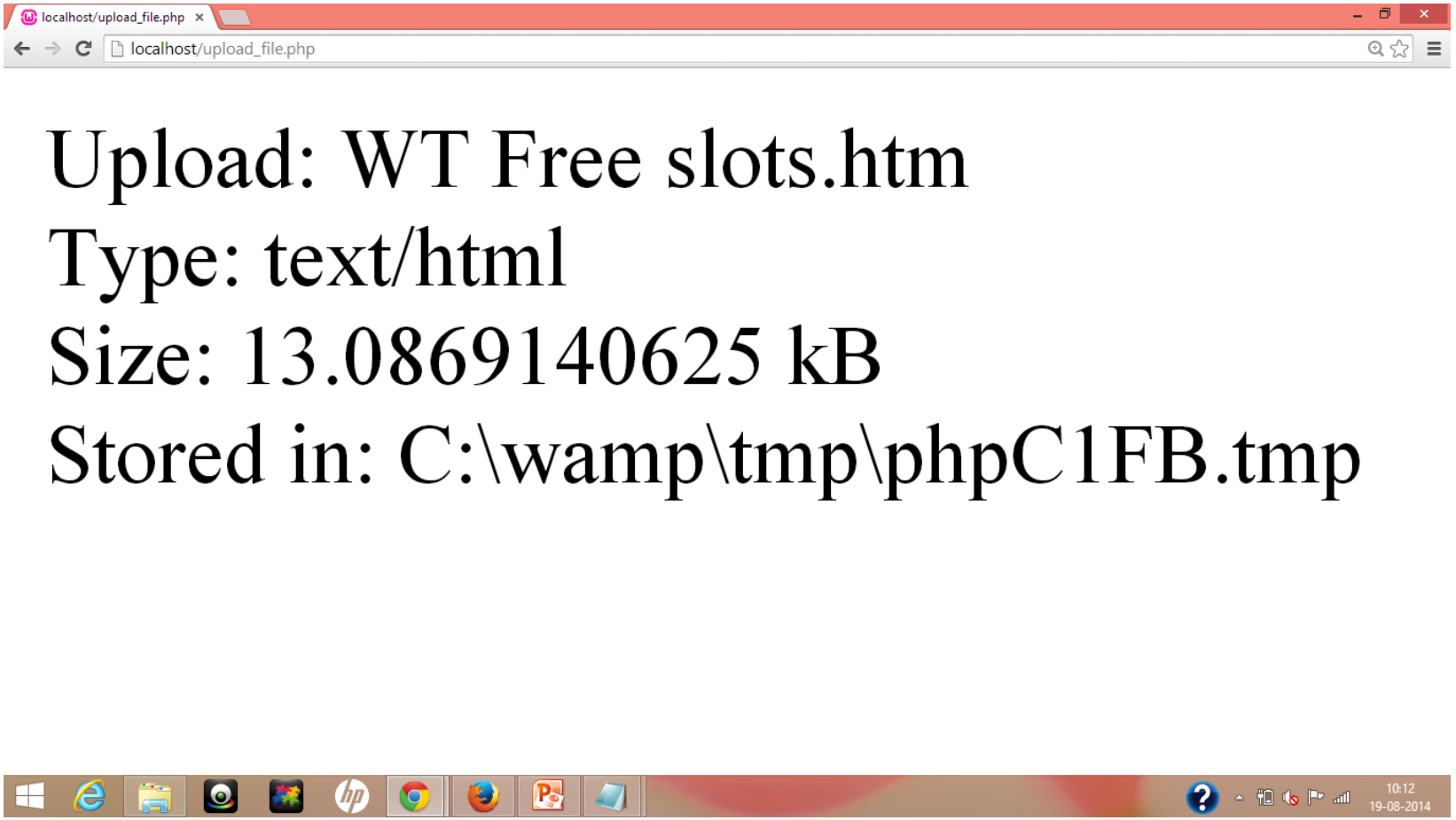
Create the upload script

```
<?php
if ($_FILES["file"]["error"] > 0) {
  echo "Error: " . $_FILES["file"]["error"] . "<br>";
} else {
  echo "Upload: " . $_FILES["file"]["name"] . "<br>";
  echo "Type: " . $_FILES["file"]["type"] . "<br>";
  echo "Size: " . ($_FILES["file"]["size"] / 1024) . " kB<br>";
  echo "Stored in: " . $_FILES["file"]["tmp_name"];
}
?>
```

By using the global PHP $_FILES array you can upload files from a client computer to the remote server.

The temporary copied files disappears when the script ends. To store the uploaded file we need to copy it to a different location.

After submitting



Cookies

- With PHP, you can both create and retrieve cookie values.
- The setcookie() function must appear BEFORE the <html> tag.
- setcookie(name, value, expire, path, domain); //syntax
- setcookie("user", "Alex Porter", time()+3600);//expire after one hour
- echo $_COOKIE["user"];// Print a cookie
- print_r($_COOKIE);// A way to view all cookies
- setcookie("user", "", time()-3600);// When deleting a cookie you should assure that the expiration date is in the past.

Sessions

- When you are working 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 and what you do because the HTTP address doesn't maintain state.
- 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 or is propagated in the URL.

A simple page-views counter

```
<?php
session_start(); // must appear BEFORE the <html> tag
if(isset($_SESSION['views'])) // checks if the "views" variable has
already been set.
$_SESSION['views']=$_SESSION['views']+1;
else
$_SESSION['views']=1;
echo "Views=". $_SESSION['views'];
?>
```

Destroying a Session

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

```
unset($_SESSION['views']);
```

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

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

Class & Object

```
<?php
class Simple
{
        public $var='Hello';
        function display()
        {
                echo $this->var;
        }
}
$obj=new Simple();
$obj->display();
?>
```

Inheritance

```
<?php
class Complex extends Simple
{
        function add($a,$b)
        {
                return ($a+$b);
        }
        function display()
        {
                parent::display();
        }
}
?>
```

```
<h1> It is possible to access the overridden methods or static properties by referencing them with parent::. </h1>
<?php
$object=new Complex();
echo $object->add(10,20);
$object->display();
?>
<p>
It is not possible to extend multiple classes; a class can only inherit from one base class.</p>
```