

# 11. PHP – Web Concepts

This session demonstrates how PHP can provide dynamic content according to browser type, randomly generated numbers or User Input. It also demonstrated how the client browser can be redirected.

## Identifying Browser & Platform

PHP creates some useful **environment variables** that can be seen in the phpinfo.php page that was used to setup the PHP environment.

One of the environment variables set by PHP is **HTTP\_USER\_AGENT** which identifies the user's browser and operating system.

PHP provides a function `getenv()` to access the value of all the environment variables. The information contained in the `HTTP_USER_AGENT` environment variable can be used to create dynamic content appropriate to the browser.

The following example demonstrates how you can identify a client browser and operating system.

**NOTE:** The function `preg_match()` is discussed in PHP Regular expression session.

```
<html>
<body>
<?php
    $viewer = getenv( "HTTP_USER_AGENT" );
    $browser = "An unidentified browser";
    if( preg_match( "/MSIE/i", "$viewer" ) )
    {
        $browser = "Internet Explorer";
    }
    else if( preg_match( "/Netscape/i", "$viewer" ) )
    {
        $browser = "Netscape";
    }
    else if( preg_match( "/Mozilla/i", "$viewer" ) )
    {
        $browser = "Mozilla";
    }
    $platform = "An unidentified OS!";
    if( preg_match( "/Windows/i", "$viewer" ) )
```

```

{
    $platform = "Windows!";
}
else if ( preg_match( "/Linux/i", "$viewer" ) )
{
    $platform = "Linux!";
}
echo("You are using $browser on $platform");
?>
</body>
</html>

```

This is producing the following result on my machine. This result may be different for your computer, depending on what browser you are using.

```

Your browser: Google Chrome 51.0.2704.103 on windows reports:
Mozilla/5.0 (Windows NT 6.3; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/51.0.2704.103 Safari/537.36

```

## Display Images Randomly

The PHP **rand()** function is used to generate a random number. This function can generate numbers with-in a given range. The random number generator should be seeded to prevent a regular pattern of numbers being generated. This is achieved using the **srand()** function that specifies the seed number as its argument.

The following example demonstrates how you can display different image each time out of four images:

```

<html>
<body>
<?php
    srand( microtime() * 1000000 );
    $num = rand( 1, 4 );

    switch( $num )

```

```

{
case 1: $image_file = "/home/images/alfa.jpg";
        break;
case 2: $image_file = "/home/images/ferrari.jpg";
        break;
case 3: $image_file = "/home/images/jaguar.jpg";
        break;
case 4: $image_file = "/home/images/porsche.jpg";
        break;
}
echo "Random Image : <img src=$image_file />";
?>
</body>
</html>

```

It will produce the following result:



## Using HTML Forms

The most important thing to notice when dealing with HTML forms and PHP is that any form element in an HTML page will automatically be available to your PHP scripts.

Try out the following example by putting the source code in test.php script.

```

<?php
if( $_POST["name"] || $_POST["age"] )
{
    echo "Welcome ". $_POST['name']. "<br />";
    echo "You are ". $_POST['age']. " years old.";
    exit();
}

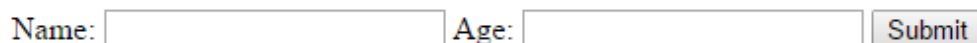
```

```

?>
<html>
<body>
  <form action="<?php $_PHP_SELF ?>" method="POST">
    Name: <input type="text" name="name" />
    Age: <input type="text" name="age" />
    <input type="submit" />
  </form>
</body>
</html>

```

It will produce the following result:



- The PHP default variable **\$\_PHP\_SELF** is used for the PHP script name and when you click "submit" button, the same PHP script will be called and will produce following result:
- The method = "POST" is used to post user data to the server script. There are two methods of posting data to the server script which are discussed in PHP GET & POST chapter.

## Browser Redirection

The PHP **header()** function supplies raw HTTP headers to the browser and can be used to redirect it to another location. The redirection script should be at the very top of the page to prevent any other part of the page from loading.

The target is specified by the **Location:** header as the argument to the **header()** function. After calling this function the **exit()** function can be used to halt parsing of rest of the code.

The following example demonstrates how you can redirect a browser request to another web page. Try out this example by putting the source code in test.php script.

```
<?php
```

```

if( $_POST["location"] )
{
    $location = $_POST["location"];
    header( "Location:$location" );
    exit();
}
?>
<html>
<body>
    <p>Choose a site to visit :</p>
    <form action="<?php $_PHP_SELF ?>" method="POST">
    <select name="location">
        <option value="http://w3c.org">
            World Wide Web Consortium
        </option>
        <option value="http://www.google.com">
            Google Search Page
        </option>
    </select>
    <input type="submit" />
    </form>
</body>
</html>

```

It will produce the following output:

Choose a site to visit :

Tutorialspoint.com ▼

Submit

## Displaying "File Download" Dialog Box

Sometime it is desired that you want to give option where a user will click a link and it will pop up a "File Download" box to the user instead of displaying actual content. This is very easy and will be achieved through HTTP header.

The HTTP header will be different from the actual header where we send **Content-Type** as **text/html\n\n**. In this case content type will be **application/octet-stream** and actual file name will be concatenated along with it.

For example, if you want make a **FileName** file downloadable from a given link, then its syntax will be as follows.

```
#!/usr/bin/perl

# HTTP Header
print "Content-Type:application/octet-stream; name=\"FileName\"\\r\\n";
print "Content-Disposition: attachment; filename=\"FileName\"\\r\\n\\n";

# Actual File Content
open( FILE, "<FileName" );
while(read(FILE, $buffer, 100) )
{
    print("$buffer");
}
```

## 12. PHP – GET and POST Methods

There are two ways the browser client can send information to the web server.

- The GET Method
- The POST Method

Before the browser sends the information, it encodes it using a scheme called URL encoding. In this scheme, name/value pairs are joined with equal signs and different pairs are separated by the ampersand.

```
name1=value1&name2=value2&name3=value3
```

Spaces are removed and replaced with the + character and any other non-alphanumeric characters are replaced with a hexadecimal values. After the information is encoded, it is sent to the server.

### The GET Method

The GET method sends the encoded user information appended to the page request. The page and the encoded information are separated by the ? character.

```
http://www.test.com/index.htm?name1=value1&name2=value2
```

- The GET method produces a long string that appears in your server logs, in the browser's Location: box.
- The GET method is restricted to send up to 1024 characters only.
- Never use GET method if you have password or other sensitive information to be sent to the server.
- GET can't be used to send binary data, like images or word documents, to the server.
- The data sent by GET method can be accessed using QUERY\_STRING environment variable.
- The PHP provides **\$\_GET** associative array to access all the sent information using GET method.

Try out the following example by putting the source code in test.php script.

```
<?php
if( $_GET["name"] || $_GET["age"] )
{
    echo "Welcome ". $_GET['name']. "<br />";
    echo "You are ". $_GET['age']. " years old.";
```

```
        exit();
    }
?>
<html>
<body>
    <form action="<?php $_PHP_SELF ?>" method="GET">
        Name: <input type="text" name="name" />
        Age: <input type="text" name="age" />
        <input type="submit" />
    </form>
</body>
</html>
```

It will produce the following result:

Name:  Age:

## The POST Method

The POST method transfers information via HTTP headers. The information is encoded as described in case of GET method and put into a header called QUERY\_STRING.

- The POST method does not have any restriction on data size to be sent.
- The POST method can be used to send ASCII as well as binary data.
- The data sent by POST method goes through HTTP header so security depends on HTTP protocol. By using Secure HTTP you can make sure that your information is secure.
- The PHP provides **\$\_POST** associative array to access all the sent information using POST method.



Try out the following example by putting the source code in test.php script.

```
<?php
if( $_POST["name"] || $_POST["age"] )
{
    echo "Welcome ". $_POST['name']. "<br />";
    echo "You are ". $_POST['age']. " years old.";
    exit();
}
?>
<html>
<body>
    <form action="<?php $_PHP_SELF ?>" method="POST">

    Name: <input type="text" name="name" />
    Age: <input type="text" name="age" />

    <input type="submit" />
    </form>
</body>
</html>
```

It will produce the following result:

Name:  Age:

## The \$\_REQUEST variable

The PHP \$\_REQUEST variable contains the contents of both \$\_GET, \$\_POST, and \$\_COOKIE. We will discuss \$\_COOKIE variable when we will explain about cookies.

The PHP \$\_REQUEST variable can be used to get the result from form data sent with both the GET and POST methods.

Try out the following example by putting the source code in test.php script.

```
<?php
if( $_REQUEST["name"] || $_REQUEST["age"] )
{
    echo "Welcome ". $_REQUEST['name']. "<br />";
    echo "You are ". $_REQUEST['age']. " years old.";
    exit();
}
?>
<html>
<body>
    <form action="<?php $_PHP_SELF ?>" method="POST">

    Name: <input type="text" name="name" />
    Age: <input type="text" name="age" />

    <input type="submit" />
    </form>
</body>
</html>
```

Here `$_PHP_SELF` variable contains the name of self script in which it is being called.

It will produce the following result:

Name:  Age:

## 13. PHp – File Inclusion

You can include the content of a PHP file into another PHP file before the server executes it. There are two PHP functions which can be used to included one PHP file into another PHP file.

- 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 thousands of files just change included file.

### The include() Function

---

The include() function takes all the text in a specified file and copies it into the file that uses the include function. If there is any problem in loading a file, then the **include()** function generates a warning but the script will continue execution.

Assume you want to create a common menu for your website. Then create a file menu.php with the following content.

```
<a href="http://www.tutorialspoint.com/index.htm">Home</a> -  
<a href="http://www.tutorialspoint.com/ebxml">ebXML</a> -  
<a href="http://www.tutorialspoint.com/ajax">AJAX</a> -  
<a href="http://www.tutorialspoint.com/perl">PERL</a> <br />
```

Now create as many pages as you like and include this file to create header. For example now your test.php file can have the following content.

```
<html>  
<body>  
<?php include("menu.php"); ?>  
<p>This is an example to show how to include PHP file!</p>  
</body>  
</html>
```

This will produce the following result:

```
Home -  
ebXML -  
AJAX -  
PERL  
  
This is an example to show how to include PHP file.
```

## The require() Function

---

The `require()` function takes all the text in a specified file and copies it into the file that uses the `include` function. If there is any problem in loading a file, then the **`require()`** function generates a fatal error and halt the execution of the script.

So there is no difference in `require()` and `include()` except they handle error conditions. It is recommended to use the `require()` function instead of `include()`, because scripts should not continue executing if files are missing or misnamed.

You can try using above example with `require()` function and it will generate same result. But if you will try the following two examples where file does not exist, then you will get different results.

```
<html>
<body>
<?php include("xxmenu.php"); ?>
<p>This is an example to show how to include wrong PHP file!</p>
</body>
</html>
```

This will produce the following result:

```
This is an example to show how to include wrong PHP file!
```

Now let us try same example with `require()` function.

```
<html>
<body>
<?php require("xxmenu.php"); ?>
<p>This is an example to show how to include wrong PHP file!</p>
</body>
</html>
```

This time file execution halts and nothing is displayed.

**NOTE:** You may get plain warning messages or fatal error messages or nothing at all. This depends on your PHP Server configuration.

## 14. PHP – Files & I/O

This chapter will explain the following functions related to files:

- Opening a file
- Reading a file
- Writing a file
- Closing a file

### Opening and Closing Files

---

The PHP **fopen()** function is used to open a file. It requires two arguments stating first the file name and then mode in which to operate.

Files modes can be specified as one of the six options in this table.

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 the file 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 the file 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 the file 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 the file does not exist, then it attempts to create a file.

If an attempt to open a file fails, then **fopen** returns a value of **false** otherwise it returns a **file pointer** which is used for further reading or writing to that file.

After making a changes to the opened file it is important to close it with the **fclose()** function. The **fclose()** function requires a file pointer as its argument and then returns **true** when the closure succeeds or **false** if it fails.

## Reading a file

---

Once a file is opened using **fopen()** function it can be read with a function called **fread()**. This function requires two arguments. These must be the file pointer and the length of the file expressed in bytes.

The file's length can be found using the **filesize()** function which takes the file name as its argument and returns the size of the file expressed in bytes.

So here are the steps required to read a file with PHP.

- Open a file using **fopen()** function.
- Get the file's length using **filesize()** function.
- Read the file's content using **fread()** function.
- Close the file with **fclose()** function.

The following example assigns the content of a text file to a variable and then displays those contents on the web page.

```
<html>
<head>
<title>Reading a file using PHP</title>
</head>
<body>

<?php
$filename = "/home/user/guest/tmp.txt";
$file = fopen( $filename, "r" );
if( $file == false )
{
    echo ( "Error in opening file" );
    exit();
}
$filesize = filesize( $filename );
$filetext = fread( $file, $filesize );

fclose( $file );

echo ( "File size : $filesize bytes" );
```

```

echo ( "<pre>$filetext</pre>" );
?>

</body>
</html>

```

It will produce the following result:

File size : 278 bytes

```

The PHP Hypertext Preprocessor (PHP) is a programming
language that allows web developers to create dynamic
content that interacts with databases.
PHP is basically used for developing web based software
applications. This tutorial helps you to build your base
with PHP.

```

## Writing a File

A new file can be written or text can be appended to an existing file using the PHP **fwrite()** function. This function requires two arguments specifying a **file pointer** and the string of data that is to be written. Optionally a third integer argument can be included to specify the length of the data to write. If the third argument is included, writing would stop after the specified length has been reached.

The following example creates a new text file and then writes a short text heading inside it. After closing this file its existence is confirmed using **file\_exists()** function which takes file name as an argument

```

<?php
$filename = "/home/user/guest/newfile.txt";
$file = fopen( $filename, "w" );
if( $file == false )
{
    echo ( "Error in opening new file" );
    exit();
}
fwrite( $file, "This is a simple test\n" );
fclose( $file );
?>

```

```
<html>
<head>
<title>Writing a file using PHP</title>
</head>
<body>

<?php
if( file_exist( $filename ) )
{
    $filesize = filesize( $filename );
    $msg = "File  created with name $filename ";
    $msg .= "containing $filesize bytes";
    echo ($msg );
}
else
{
    echo ("File $filename does not exit" );
}
?>
</body>
</html>
```

It will produce the following result:

Error in opening new file

We have covered all the function related to file input and out in the [PHP File System Function](#) chapter.