PHP Filter, Form, Post Get

PHP Web Concepts

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.

- Example

```
<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>
```

```
You are using Mozilla! on Windows!
```

Display Images Randomly

- The PHP **rand()** function is used to generate a random number.
- The 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 specifiies the seed number as its argument.

- Example

```
<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>
```

Browser Redirection

- The PHP **header()** function supplies raw HTTP headers to the browser and can be used to redirect it to another location.

- 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.

- Example

```
<?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 Wise Web Consortium
      </option>
      <option value="http://www.google.com">
            Google Search Page
      </option>
   </select>
   <input type="submit" />
   </form>
</body>
</html>
```

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

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 upto 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.

- Example

```
<?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>
```

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 GET method.

- Example

```
<?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>
```

The $_REQUEST variable

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

```
<?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>
```

File Operation, Function

Table of Contents

PHP Files & I/O

- Explain 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 files does not exist then it attemts 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 attemts 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 attemts 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 attemts 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 files'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.

- 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.

- Example assigns the content of a text file to a variable 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>
```

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 will stop after the specified length has been reached.

- Example creates a new text file then writes a short text heading insite it. After closing this file its existence is confirmed using **file_exist()** 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>
```

PHP Functions

- A function is a piece of code which takes one more input in the form of parameter and does some processing and returns a value.

- You already have seen many functions like **fopen()** and **fread()** etc. They are built-in functions but PHP gives you option to create your own functions as well.
  - Creating a PHP Function
  - Calling a PHP Function

Creating PHP Function

- Suppose you want to create a PHP function which will simply write a simple message on your browser when you will call it.

- While creating a function its name should start with keyword **function** and all the PHP code should be put inside { and } braces.

- Following example creates a function called writeMessage() and then calls it just after creating it.

```
<html>
<head>
<title>Writing PHP Function</title>
</head>
<body>

<?php
/* Defining a PHP Function */
function writeMessage()
{
  echo "You are really a nice person, Have a nice time!";
}
/* Calling a PHP Function */
writeMessage();
?>
</body>
</html>
```

PHP Functions with Parameters

- PHP gives you option to pass your parameters inside a function.
- These parameters work like variables inside your function.

- Example takes two integer parameters and add them together and then print them.

```
<html>
<head>
<title>Writing PHP Function with Parameters</title>
</head>
<body>

<?php
function addFunction($num1, $num2)
{
  $sum = $num1 + $num2;
  echo "Sum of the two numbers is : $sum";
}
addFunction(10, 20);
?>
</body>
</html>
```

Passing Arguments by Reference

- It is possible to pass arguments to functions by reference.
- This means that a reference to the variable is manipulated by the function rather than a copy of the variable's value.
- Any changes made to an argument in these cases will change the value of the original variable.

```
<html>
<head>
<title>Passing Argument by Reference</title>
</head>
<body>
<?php
function addFive($num)
{
   $num += 5;
}

function addSix(&$num)
{
   $num += 6;
}
$orignum = 10;
addFive( &$orignum );
echo "Original Value is $orignum<br />";
addSix( $orignum );
echo "Original Value is $orignum<br />";
?>
</body>
</html>
```