Pemrograman Web

11. PHP: File, Function, Regular Expression

M. Udin Harun Al Rasyid, S.Kom, Ph.D
http://lecturer.eepis-its.edu/~udinharun
udinharun@eepis-its.edu

Table of Contents

- **PHP Files & I/O**
- **PHP Functions**
- **PHP -Regular Expressions**

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>
```

PHP Functions retruning value

- A function can return a value using the **return** statement in conjunction with a value or object. return stops the execution of the function and sends the value back to the calling code.

- You can return more than one value from a function using **return array(1,2,3,4)**.

- Following example takes two integer parameters and add them together and then returns their sum to the calling program.

```
<html>
<head>
<title>Writing PHP Function which returns value</title>
</head>
<body>

<?php
function addFunction($num1, $num2)
{
  $sum = $num1 + $num2;
  return $sum;
}
$return_value = addFunction(10, 20);
echo "Returned value from the function : $return_value
?>
</body>
</html>
```

Setting Default Values for Function Parameters

- You can set a parameter to have a default value if the function's caller doesn't pass it.

- Following function prints NULL in case use does not pass any value to this function.

```
<html>
<head>
<title>Writing PHP Function which returns value</title>
</head>
<body>

<?php
function printMe($param = NULL)
{
   print $param;
}
printMe("This is test");
printMe();
?>

</body>
</html>
```

```
This is test
```

Dynamic Function Calls

- It is possible to assign function names as strings to variables and then treat these variables exactly as you would the function name itself.

```
<html>
<head>
<title>Dynamic Function Calls</title>
</head>
<body>
<?php
function sayHello()
{
   echo "Hello<br />";
}
$function_holder = "sayHello";
$function_holder();
?>
</body>
</html>
```

```
Hello
```

PHP -Regular Expressions

- Regular expressions are nothing more than a sequence or pattern of characters itself. They provide the foundation for pattern-matching functionality.

- Using regular expression you can:
  - search a particular string inside a another string
  - replace one string by another string
  - split a string into many chunks

- PHP offers functions specific to two sets of regular expression functions, each corresponding to a certain type of regular expression.
  - POSIX Regular Expressions
  - PERL Style Regular Expressions

POSIX Regular Expressions

- The structure of a POSIX regular expression is not dissimilar to that of a typical arithmetic expression: various elements (operators) are combined to form more complex expressions.

- The simplest regular expression is one that matches a single character, such as g, inside strings such as g, haggle, or bag.

Brackets

- Brackets ([]) have a special meaning when used in the context of regular expressions. They are used to find a range of characters.

Expression	Description
[0-9]	It matches any decimal digit from 0 through 9.
[a-z]	It matches any character from lowercase a through lowercase z.
[A-Z]	It matches any character from uppercase A through uppercase Z.
[a-Z]	It matches any character from lowercase a through uppercase Z.

PHP's Regexp POSIX Functions

Function	Description
ereg()	The ereg() function searches a string specified by string for a string specified by pattern, returning true if the pattern is found, and false otherwise.
ereg_replace()	The ereg_replace() function searches for string specified by pattern and replaces pattern with replacement if found.
eregi()	The eregi() function searches throughout a string specified by pattern for a string specified by string. The search is not case sensitive.
eregi_replace()	The eregi_replace() function operates exactly like ereg_replace(), except that the search for pattern in string is not case sensitive.
split()	The split() function will divide a string into various elements, the boundaries of each element based on the occurrence of pattern in string.
spliti()	The spliti() function operates exactly in the same manner as its sibling split(), except that it is not case sensitive.
sql_regcase()	The sql_regcase() function can be thought of as a utility function, converting each character in the input parameter string into a bracketed expression containing two characters.

PERL Style Regular Expressions

tion	Description
_match()	The preg_match() function searches string for pattern, returning true if pattern exists, and false otherwise.
_match_all()	The preg_match_all() function matches all occurrences of pattern in stri
_replace()	The preg_replace() function operates just like ereg_replace(), except tha regular expressions can be used in the pattern and replacement input parameters.
_split()	The preg_split() function operates exactly like split(), except that regula expressions are accepted as input parameters for pattern.
_grep()	The preg_grep() function searches all elements of input_array, returning elements matching the regexp pattern.
_quote()	Quote regular expression characters

Finish