Practical PHP

COMP-3077 Web-Based Data Management

Saja Al-Mamoori

String Functions

Function	Description
strlen()	Returns the length of a string
strnatcasecmp()	Compares two strings using a "natural order" algorithm (case-insensitive) . It returns -1 for mismatch, 1 for match.
strnatcmp()	Compares two strings using a "natural order" algorithm (case-sensitive). It returns -1 for mismatch, 1 for match.
strrchr()	Finds the last occurrence of a string inside another string
strrev()	Reverses a string
strspn()	Returns the number of characters found in a string that contains only characters from a specified charlist
strtolower()	Converts a string to lowercase letters
strtoupper()	Converts a string to uppercase letters
str_repeat()	Repeats a string a specified number of times

Example

```
<?php
echo strrev(" .dlrow olleH"); // Reverse string
echo str_repeat("Hip ", 2); // Repeat string
echo strtoupper("hooray!"); // String to uppercase
?>
```

- This example uses three string functions to output the following text:
- **Hello world. Hip Hip HOORAY!**

Date and Time Functions

- PHP uses standard Unix timestamps
- To determine the current timestamp, you can use the time function:

```
echo time();
```

- Timestamp is stored as seconds, to get the next week’s Timestamp:

```
echo time() + 7 * 24 * 60 * 60;
```

- to create a timestamp for a given date, you can use the mktime function:

```
echo mktime(0, 0, 0, 1, 1, 2000);
```

Date and Time Functions

- For mktime , The parameters to pass are, in order from left to right:
- The number of the hour (0–23)
- The number of the minute (0–59)
- The number of seconds (0–59)
- The number of the month (1–12)
- The number of the day (1–31)
- The year (1970–2038, or 1901–2038 with PHP 5.1.0+ on 32-bit signed systems)

Date and Time Functions

- To display the date, use the date function, which supports a plethora of formatting options:
- date($format, $timestamp);
- Example:

echo date("l F jS, Y - g:ia", time());

Will display:

Tuesday June 6th, 2017 7:38pm

- Format symbols meanings: http://php.net/manual/en/function.date.php

Using checkdate

- But how can you check whether a user has submitted a valid date to your program?
- The answer is to pass the month, day, and year to the checkdate function.
- It returns a value of TRUE if the date is valid, or FALSE if it is not.

```
<?php
 $month = 9;     // September (only has 30 days)
 $day   = 31;    // 31st
 $year  = 2018;  // 2018
  if (checkdate($month, $day, $year))
           echo "Date is valid";
  else   echo "Date is invalid"; ?>
```

File Handling

- Sometimes it can be quicker and more convenient to directly access files on the hard disk (rather than MySQL).
- e.g. modifying images such as uploaded user avatars, or log files that you wish to process.
- file naming convention: Windows and Mac OS X filenames are not case-sensitive, but Linux and Unix ones are.
- Therefore, you should always assume that the system is case-sensitive and stick to a convention such as all lowercase filenames.

Checking Whether a File Exists

- file_exists function determines whether a file already exists.
- It returns either TRUE or FALSE, and is used like this:

```
if (file_exists("testfile.txt"))
echo "File exists";
```

<<<_END

- <<<_END let you write any thing between <<<_END and _END.
- It works for multiple line text
- It works very well while displaying HTML code using php

```
<? php echo <<<_END
Hello,
I can write multiple line here,
Sincerely,
Instructor
 _END
?>
```

Creating a File

```
<?php // testfile.php
$fh = fopen("testfile.txt", 'w') or die("Failed to create file");
  $text = <<<_END
 Line 1
Line 2
 Line 3
_END;
  fwrite($fh, $text) or die("Could not write to file");
 fclose($fh);
  echo "File 'testfile.txt' written successfully"; ?>
?>
```

At this point, testfile.txt doesn’t exist

Creating a File – fopen mode

MODE	Action	Description
'w'	Write from file start and truncate file.	Open for writing only; place the file pointer at the beginning of the file and truncate the file to zero length. If the file doesn't exist, attempt to create it.
'w+'	Write from file start, truncate file, and allow reading.	Open for reading and writing; place the file pointer at the beginning of the file and truncate the file to zero length. If the file doesn't exist, attempt to create it.
'a'	Append to file end.	Open for writing only; place the file pointer at the end of the file. If the file doesn't exist, attempt to create it.
'a+'	Append to file end and allow reading.	Open for reading and writing; place the file pointer at the end of the file. If the file doesn't exist, attempt to create it.
'r'	Read from file start.	Open for reading only; place the file pointer at the beginning of the file. Return FALSE if the file doesn't already exist.
'r+'	Read from file start and allow writing.	Open for reading and writing; place the file pointer at the beginning of the file. Return FALSE if the file doesn't already exist.

Reading from Files

- The easiest way to read from a text file is to grab a whole line through fgets (think of the final s as standing for string)

```
<?php
  $fh = fopen("testfile.txt", 'r') or    die("File does not exist or you lack
permission to open it");
  $line = fgets($fh);
  fclose($fh);
  echo $line;
?>
```

```
Output:
Line 1
```

Reading from Files

- you can retrieve multiple lines or portions of lines through the fread function, as

```
<?php
$fh = fopen("testfile.txt", 'r') or    die("File does not exist or you lack
permission to open it");
  $text = fread($fh, 3);
  fclose($fh);
 echo $text;
 ?>
```

```
Output:
Lin
```

I've requested three characters in the fread call

Copying Files

- copy function to create a clone of testfile.txt, and save it as copyfile.php, and then call up the program in your browser.

```
<?php // copyfile.php
copy('testfile.txt', 'testfile2.txt') or die("Could not copy file");
echo "File successfully copied to 'testfile2.txt'";
 ?>
```

Alternative Syntax

```
<?php // copyfile2.php
 if (!copy('testfile.txt', 'testfile2.txt'))
 echo "Could not copy file";  else echo "File successfully copied to 'testfile2.txt'";
 ?>
```

Moving a File

- To move a file, rename it with the rename function

```
<?php // movefile.php
if (!rename('testfile2.txt', 'testfile2.new'))
   echo "Could not rename file";
else echo "File successfully renamed to 'testfile2.new'";
?>
```

- You can use the rename function on directories too.
- if the original file doesn’t exist, you can call the file_exists function first to check.

Deleting a File

- Deleting a file is just a matter of using the unlink function to remove it from the filesystem.

```
<?php // deletefile.php
 if (!unlink('testfile2.new'))
echo "Could not delete file";
 else echo "File 'testfile2.new' successfully deleted";
?>
```

Updating Files

- add more data to a saved , which you can do in many ways:
- append write modes
- Or open file with (r+) mode, seek the end, then add data.
- <?php // update.php

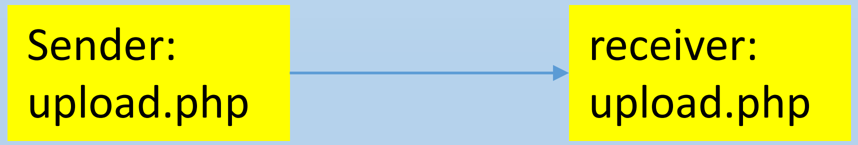
```
    $fh   = fopen("testfile.txt", 'r+') or die("Failed to open file");  $text = fgets($fh);
    fseek($fh, 0, SEEK_END);
fwrite($fh, "$text") or die("Could not write to file");  fclose($fh);
  echo "File 'testfile.txt' successfully updated";
?>
```

$_FILES function

- Uploading items to the current script via HTTP post method
- It also requires a special type of encoding called **multipart/form-data**
- Then
- $_FILES function receive the file information in the receiver page
- your browser will handle the rest of uploading the file.

Uploading Files

```
<?php
// upload.php
 echo <<<_END    <html><head><title>PHP Form Upload</title></head>
<body>
  <form method='post' action='upload.php' enctype='multipart/form-data'>
  Select File: <input type='file' name='filename' size='10'>
   <input type='submit' value='Upload'>
   </form>
_END;
echo "</body></html>";
?>
```



Uploading Files

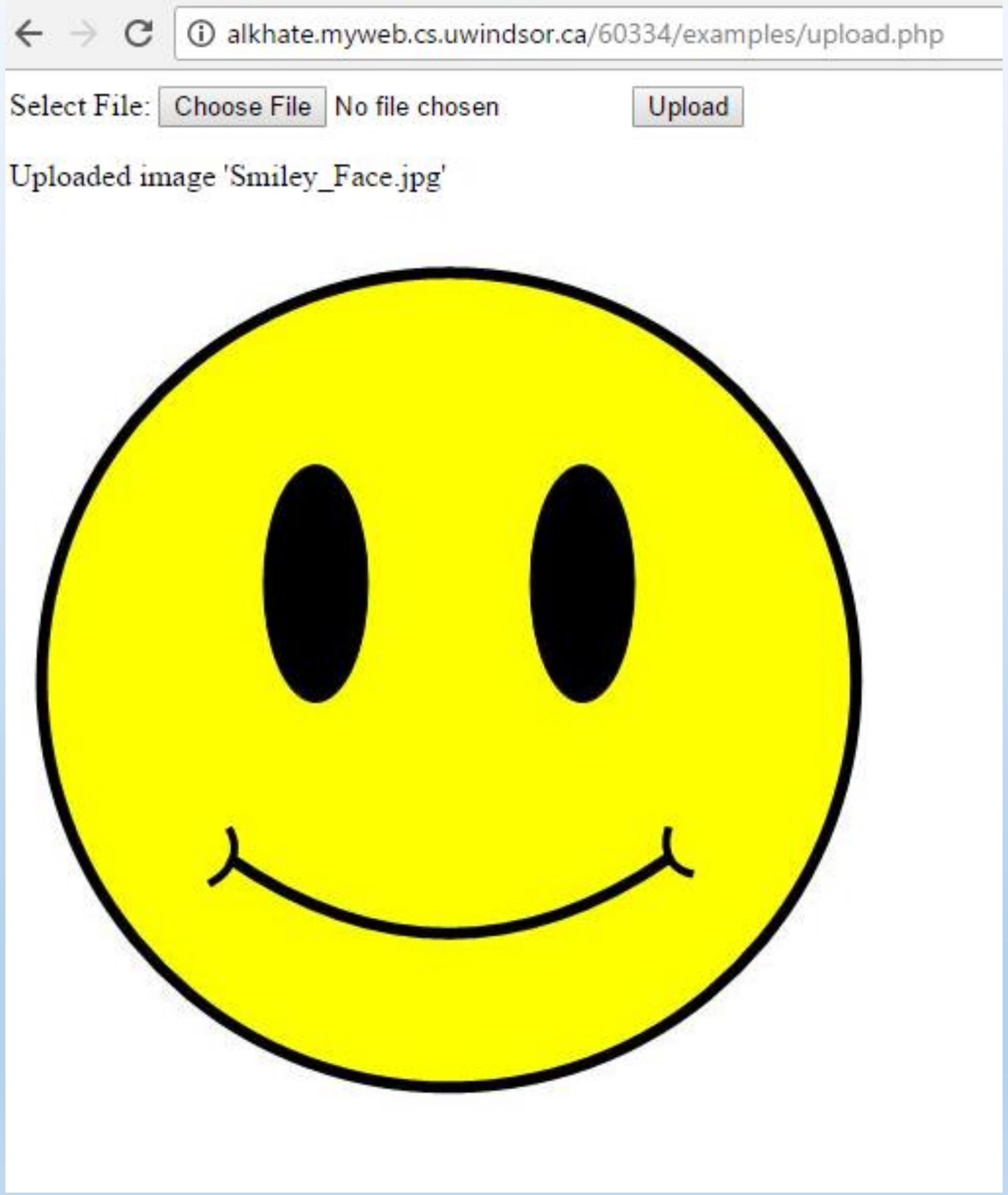
- <?php // upload.php echo <<<_END ……_END;

```
if ($_FILES)  {
 $name = $_FILES['filename']['name'];
move_uploaded_file($_FILES['filename']['tmp_name'], $name);
echo "Uploaded image '$name'<br><img src='$name'>";
 }
echo "</body></html>"; ?>
```

- Try it:
- http://alkhate.myweb.cs.uwindsor.ca/60334/examples/upload.php

- The code: chapter 7 – 15 Examples in

http://lpmj.net/4thedition/


alkhate.myweb.cs.uwindsor.ca/60334/examples/upload.php
Select File: Choose File No file chosen Upload
Uploaded image 'Smiley_Face.jpg'

What is happening?

- The first line of the multiline echo statement starts an HTML document, displays the title, and then starts the document’s body.
- the POST method of form submission, sets the target for posted data to the program **upload.php (the program itself).**
- Form tells the browser that data is encoded via the content type of **multipart/ form-data**.
- Creating the form contents.
- A common technique in web programming in which a single program is **called twice**: once when the user **first visits a page**, and again when the user presses the Submit button.

What is happening?

- all uploaded files are placed into the associative system array $_FILES.
- Therefore, a quick check to see whether $_FILES contains anything is sufficient to determine whether the user has uploaded a file. This is done with the statement if ($_FILES).
- The first time the user visits the page, before uploading a file, $_FILES is empty, so the program skips if ($_FILES) block of code.
- Once the program realizes that a file was uploaded, the actual name, as read from the uploading computer, is retrieved and placed into the variable **$name**.
- move_uploaded_file to move the file from tmp into permanent file.

Using $_FILES

- Five things are stored in the $_FILES array when a file is uploaded, as shown in Table (where file is the file upload field name supplied by the submitting form).

Array element	Contents
$_FILES['file']['name']	The name of the uploaded file (e.g., smiley.jpg)
$_FILES['file']['type']	The content type of the file (e.g., image/jpeg)
$_FILES['file']['size']	The file's size in bytes
$_FILES['file']['tmp_name']	The name of the temporary file stored on the server
$_FILES['file']['error']	The error code resulting from the file upload

Content type

- Content types used to be known as MIME (Multipurpose Internet Mail Extension) types, but because their use later expanded to the whole Internet, now they are often called Internet media types.

Content Types			
application/pdf	image/gif	multipart/form-data	text/xml
application/zip	image/jpeg	text/css	video/mpeg
audio/mpeg	image/png	text/html	video/mp4
audio/x-wav	image/tiff	text/plain	video/quicktime

Validation

- due to the possibility of users attempting to hack into your server.
- Its very important to validate the type of the uploaded files.
- Also, to check whether the user sent the right type of file.
- Taking all this into account, the upload.php example is modified to be as the following.

```
<?php // upload2.php
 echo <<<_END
.... _END;
```

Validation – validate the extension

```
if ($_FILES)  {
$name = $_FILES['filename']['name'];
switch($_FILES['filename']['type'])    {
case 'image/jpeg': $ext = 'jpg'; break;
case 'image/gif':  $ext = 'gif'; break;
case 'image/png':  $ext = 'png'; break;
case 'image/tiff': $ext = 'tif'; break;
default:           $ext = '';    break;
} //checking the acceptable type
```

Validation

```
 if ($ext)    {
 $n = "image.$ext";
move_uploaded_file($_FILES['filename']['tmp_name'], $n);
echo "Uploaded image '$name' as '$n':<br>";
 echo "<img src='$n'>";
  }
 else echo "'$name' is not an accepted image file";
 }  else echo "No image has been uploaded";
  echo "</body></html>"; ?>
```

Validation

- Due to different OS in different servers, use:

```
$name = preg_replace("/[^A-Za-z0-9.]/", "", $name);
```

- This leaves only the characters A–Z, a–z, 0–9, and periods in the string $name, and strips out everything else.
- Even better, to ensure that your program will work on all systems, regardless of whether they are case-sensitive or case-insensitive, you should probably use the following command instead, which changes all uppercase characters to lowercase at the same time:

```
$name = strtolower(ereg_replace("[^A-Za-z0-9.]", "", $name));
```

References

- Learning PHP, MySQL & JavaScript, 4th Edition. With jQuery, CSS & HTML5, by Robin Nixon, O'Reilly Media, 2014, 978-1-4919-1866-1, chapter 7