

CSC435: Web Programming

Lecture 22: Function, File I/O

Bei Xiao

American University

Tuesday, April 16, 2019

Today's lecture

More on basic PHP

Query parameters

Web services

JSON/File processing

HTML Forms

Exercisess

Announcement

No Class on Friday.

Office hour this week: Thursday 2-3pm.

But in-class exercises/ tutorials: [music.php](#)

Quiz on next Tuesday, April 23 (Basic PHP).

Take home tutorial !!

PHP tutorials for beginners (really great tutorials):

<http://www.homeandlearn.co.uk/php/php.html>

PHP, Ajax, JavaScript, JSON (important for homework!)

<http://blog.teamtreehouse.com/beginners-guide-to-ajax-development-with-php>

jQuery Ajax call and JSON return

<https://jonsuh.com/blog/jquery-ajax-call-to-php-script-with-json-return/>

Query strings and parameters

URL ?name=value&name=value...

```
http://www.google.com/search?q=Romney  
http://example.com/student_login.php?username=obourn&id=1234567
```

- **query string:** a set of parameters passed from a browser to a web server
 - often passed by placing name/value pairs at the end of a URL
 - above, parameter username has value obourn, and sid has value 1234567
- PHP code on the server can examine and utilize the value of parameters
- a way for PHP code to produce different output based on values passed by the user

Query parameters: \$_GET, \$_POST

```
$user_name = $_GET["username"];  
$id_number = (int) $_GET["id"];  
$seats_meat = FALSE;  
if (isset($_GET["meat"])) {  
    $seats_meat = TRUE;  
}
```

PHP

`$_GET["parameter name"]` or `$_POST["parameter name"]` returns a GET/POST parameter's value as a string

parameters specified as `http://...?name=value&name=value` are GET parameters

test whether a given parameter was passed with `isset`

\$_GET, \$_POST used in form submission

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

        exit();
    }
?>
```

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

Web services

- **Web service:** software functionality that can be invoked through the internet using common protocols
- Like a remote function(s) you can call by contacting a program on a web server
- Many web services accept parameters and produce results
- Can be written in PHP and contacted by the browser in HTML and/or AJAX code
- Service's output might be HTML but could be text, XML, JSON, or other content

Setting Content Type with header

```
header("Content-type: type/subtype");
```

```
header("Content-type: type/plain");  
print "This output will appear as plain text now !\n"
```

- By default, a PHP file's output is assumed to be HTML (text/html)
- However, in this course we aren't using PHP to generate HTML, so we use the [header](#) function to specify non-HTML output
- Must appear before any other output generated by the script
- (doesn't have to be the first line of code, though)
- I appear as plain text now!\n";

Example: Exponent Web Service

- Write a web service that accepts a base and exponent and outputs base to the exponent power. For example, the following query should output 8:
- <http://example.com/exponent.php?base=2&exponent=3>
- Solution

```
<?php
    header("Content-type: text/plain");
    $base = (int) $_GET["base"];
    $exp = (int) $_GET["exponent"];
    $result = pow($base, $exp);
    print $result;
?>
```

Embedded PHP Syntax Template

```
HTML content  
<?php PHP code ?>  
  
HTML content  
  
<?php PHP code ?>  
  
HTML content...
```

Any contents of a .php file between <?php and ?> are executed as PHP code

All other contents are output as pure HTML

Embedded PHP Syntax Template

```
<DOCTYPE html>
<html> <head><title>Embedded PHP</title></head>
<body>
<?php for ($i = 99; $i >= 1; $i--) { ?>
    <p> <?= $i ?> bottles of beer on the wall, <br /> <?= $i ?> bottles
of beer. <br /> Take one down, pass it around, <br /> <?= $i - 1 ?>
bottles of beer on the wall. </p>
    <?php } ?>
</body>
</html>
```

Note: This is messy! There are much better ways to do this, and you should not write any embedded PHP in this class. But you should be aware of it.

Returning JSON from PHP

We use the PHP function [json_encode](#)(array) to output JSON

`json_encode` takes PHP arrays (including nested arrays) and generates JSON strings that can be printed

NOTE: we can also use `json_decode` to convert json strings into PHP arrays.

Example PHP code

```
<?php
    header("Content-Type: application/json");
    $output = array();
    $output["name"] = "Kyle";
    $output["hobbies"] = array("reading", "frisbee");
    print(json_encode($output));

?>
```

Produces:

```
{
  "name": "Kyle",
  "hobbies": ["reading","frisbee"]
}
```

PHP file I/O functions

function name(s)	category
<u>file</u> , <u>file_get_contents</u> , <u>file_put_contents</u>	reading/writing entire files
<u>basename</u> , <u>file_exists</u> , <u>filesize</u> , <u>fileperms</u> , <u>filemtime</u> , <u>is_dir</u> , <u>is_readable</u> , <u>is_writable</u> , <u>disk_free_space</u>	asking for information
<u>copy</u> , <u>rename</u> , <u>unlink</u> , <u>chmod</u> , <u>chgrp</u> , <u>chown</u> , <u>mkdir</u> , <u>rmdir</u>	manipulating files and directories
<u>glob</u> , <u>scandir</u>	reading directories

https://www.tutorialspoint.com/php/php_files.htm

The File() function

- file function returns lines of a file as an array (\n at end of each)

```
<?php  
print_r(file("test.txt"));  
?>
```

```
Array  
(  
[0] => Hello World. Testing testing!  
[1] => Another day, another line.  
[2] => If the array picks up this line,  
[3] => then is it a pickup line?  
)
```


The File() function

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

echo fread($myfile,filesize("webdictionary.txt"));
fclose($myfile);

?>
```

The File() function

```
# display lines of file as a bulleted list
$lines = file("todolist.txt");
foreach ($lines as $line) {
    print $line;
}
```

PHP

- file returns the lines of a file as an array of strings
- each ends with \n ; to strip it, use an optional second parameter:

```
$lines = file("todolist.txt", FILE_IGNORE_NEW_LINES);
```

- common idiom: foreach or for loop over lines of file

Reading/Writing files

contents of foo.txt	file("foo.txt")	file_get_contents("foo.txt")
Hello how r u? I'm fine	array("Hello\n", # 0 "how r u?\n", # 1 "\n", # 2 "I'm fine\n" # 3)	"Hello\n how r u?\n # a single \n # string I'm fine\n"

- file function returns lines of a file as an array (\n at end of each)
- file_get_contents returns entire contents of a file as a single string
- file_put_contents writes a string into a file

Splitting/Joining strings

```
$array = explode(delimiter, string);  
$string = implode(delimiter, array);
```

PHP

```
$s  = "CSC 435 S";  
$a  = explode(" ", $s);      # ("CSC", "435", "S")  
$s2 = implode("...", $a);    # "CSC...435...S"
```

PHP

- explode and implode **convert between strings and arrays**
- for more complex string splitting, you can use regular expressions (later)

<http://php.net/explode>

Reading directories

function	description
<u>glob</u>	returns an array of all file names that match a given pattern (returns a file path and name, such as "foo/bar/myfile.txt")
<u>scandir</u>	returns an array of all file names in a given directory (returns just the file names, such as "myfile.txt")

- glob can accept a general path with the * wildcard character
- (more powerful)

Example with explode

```
Martin D Stepp  
Jessica K Miller  
Victoria R Kirst  
names.txt
```

contents of input file

```
<?php foreach (file("names.txt") as $name) {  
    $tokens = explode(" ", $name);    #separated by space  
    ?>  
    <p> author: <?= $tokens[2] ?>, <?= $tokens[0] ?> </p>  
    <?php  
}
```

author: Stepp, Marty

author: Miller, Jessica

author: Kirst, Victoria

output

<http://php.net/explode>

Glob: find pathnames matching a pattern

```
# reverse all poems in the poetry directory
$poems = glob("poetry/poem*.dat");

foreach ($poems as $poemfile) {
    $text = file_get_contents($poemfile); # read entire file
    file_put_contents($poemfile, strrev($text));
    print "I just reversed " . basename($poemfile) . "\n";
}
PHP
```

- glob can match a "wildcard" path with the * character
 - glob("foo/bar/*.doc") returns all .doc files in the foo/bar subdirectory
 - glob("food*") returns all files whose names begin with "food"
- the basename function strips any leading directory from a file path
 - basename("foo/bar/baz.txt") returns "baz.txt"

scandir example

```
<ul>
  <?php foreach (scandir("taxes/old") as $filename) { ?>
    <li>I found a file: <?= $filename ?></li>
  <?php } ?>
</ul>
```

PHP

- .
- ..
- 2007_w2.pdf
- 2006_1099.doc

output

- scandir includes current directory (".") and parent ("..") in the array
- don't need basename with scandir; returns file names only without directory

Reading directories

function	description
<u>glob</u>	returns an array of all file names that match a given pattern (returns a file path and name, such as "foo/bar/myfile.txt")
<u>scandir</u>	returns an array of all file names in a given directory (returns just the file names, such as "myfile.txt")

- glob can accept a general path with the * wildcard character
- (more powerful)

Exercise 1: JSON

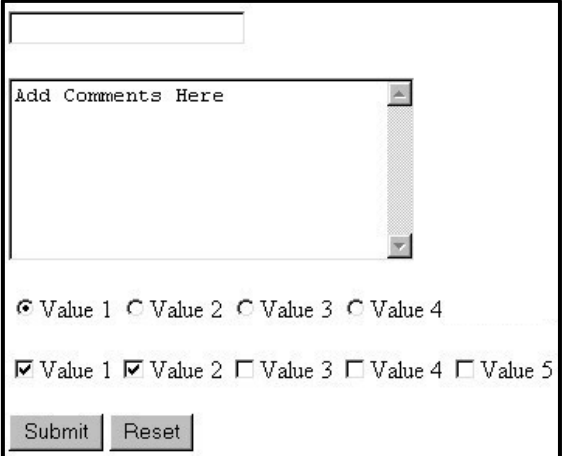
- Write a `save.php` to create a `.JSON` and save the file on your disk.
- It can contain anything. E.g.
`{'Eric':{'grades':"A"}.....}`

`File_put_contents('myfile.json',$json_data)`

- Write a `read.php` to read the JSON from your disk.
- And echo the content back to the browser.

HTML forms

- form: a group of UI controls that accepts information from the user and sends the information to a web server.
- The information is sent to the server as a query string.
- JavaScript can be used to create interactive controls



The image shows a screenshot of a web form. At the top is a single-line text input field. Below it is a larger text area with the placeholder text "Add Comments Here". Under the text area are four radio buttons labeled "Value 1", "Value 2", "Value 3", and "Value 4", with "Value 1" selected. Below the radio buttons are five checkboxes labeled "Value 1", "Value 2", "Value 3", "Value 4", and "Value 5", with "Value 1" and "Value 2" checked. At the bottom are two buttons: "Submit" and "Reset".

HTML form: <form>

```
<form action="destination URL">  
  form controls  
</form>  
HTML
```

- required action attribute gives the URL of the page that will process this form's data
- when form has been filled out and submitted, its data will be sent to the action's URL
- one page may contain many forms if so desired

HTML form: <form>

```
<form action="destination URL">  
  form controls  
</form>  
HTML
```

- required action attribute gives the URL of the page that will process this form's data
- when form has been filled out and submitted, its data will be sent to the action's URL
- one page may contain many forms if so desired

HTML form example

```
<form action="http://www.google.com/search">
  <div>
    Let's search Google:
    <input name="q" />
    <input type="submit" />
  </div>
</form>
```

HTML

Let's search Google:

output

- must wrap the form's controls in a block element such as div

Form controls: <input>

```
<!-- 'q' happens to be the name of Google's  
required parameter -->
```

```
<input type="text" name="q" value="Colbert  
Report" />
```

```
<input type="submit" value="Booyah!" />
```

HTML

Colbert Report

Booyah!

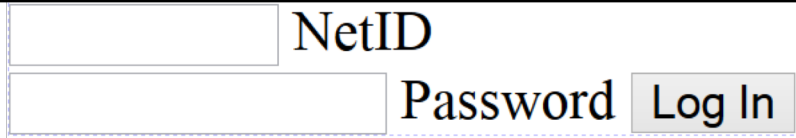
output

- input element is used to create many UI controls
 - an inline element that MUST be self-closed
- name attribute specifies name of query parameter to pass to server
- type can be button, checkbox, file, hidden, password, radio, reset, submit, text, ...
- value attribute specifies control's initial text

Text fields: <input>

```
<input type="text" size="10" maxlength="8" /> NetID <br />  
<input type="password" size="16" /> Password  
<input type="submit" value="Log In" />
```

HTML



NetID

Password Log In

output

- input attributes: disabled, maxlength, readonly, size, value
- size attribute controls onscreen width of text field
- maxlength limits how many characters user is able to type into field

Drop-down list, <select>, <option>

menus of choices that collapse and expand (inline)

```
<select name="favoritecharacter">  
  <option>Jerry</option>  
  <option>George</option>  
  <option selected="selected">Kramer</option>  
  <option>Elaine</option>  
</select>
```

HTML

Kramer ▾

Submit Query

output

- option element represents each choice
- select optional attributes: disabled, multiple, size
- optional selected attribute sets which one is initially chosen

Using <select> for lists

```
<select name="favoritecharacter[]" size="3"
multiple="multiple">
  <option>Jerry</option>
  <option>George</option>
  <option>Kramer</option>
  <option>Elaine</option>
  <option selected="selected">Newman</option>
</select>
```

HTML



Kramer
Elaine
Newman

Submit Query

output

- optional multiple attribute allows selecting multiple items with shift- or ctrl-click
 - must declare parameter's name with [] if you allow multiple selections
- option tags can be set to be initially selected

Submitting data to a web server

- Though browsers mostly retrieve data, sometimes you want to submit data to a server
 - Gmail: Send a message
 - Flickr: Upload a photo
 - Google Calendar: Create an appointment
- the data is sent in HTTP requests to the server
 - with HTML forms
 - with **Ajax**
 - the data is placed into the request as parameters

HTTP GET versus POST

GET : asks a server for a page or data

- if the request has parameters, they are sent in the URL as a query string

POST : submits data to a web server and retrieves the server's response

- if the request has parameters, they are embedded in the request's HTTP packet, not the URL

For submitting data to be saved, POST is more appropriate than GET

- GET requests embed their parameters in their URLs
- URLs are limited in length (~ 1024 characters)
- URLs cannot contain special characters without encoding
- private data in a URL can be seen or modified by users

Form POST example

```
<form action="http://food.com/app.php" method="post">
  <div>
    Name: <input type="text" name="name" /> <br />
    Food: <input type="text" name="meal" /> <br />
    <label>Meat? <input type="checkbox" name="meat"
/></label> <br />
    <input type="submit" />
  </div>
</form>
```

HTML

Name:

Food:

Meat? ☐

output

A form submitted to itself

```
<form action="" method="post">  
  ...  
</form>  
HTML
```

- A form can submit its data back to itself by setting the action to be blank (or to the page's own URL)
- benefits
 - fewer pages/files (don't need a separate file for the code to process the form data)
 - can more easily re-display the form if there are any errors

Processing a self-submitted form

```
if ($_SERVER["REQUEST_METHOD"] == "GET") {  
    # normal GET request; display self-submitting form  
    ?>  
    <form action="" method="post">...</form>  
    <?php  
} elseif ($_SERVER["REQUEST_METHOD"] == "POST") {  
    # POST request; user is submitting form back to here; process it  
    $var1 = $_POST["param1"];  
    ...  
}  
PHP
```

- a page with a self-submitting form can process both GET and POST requests
- look at the global `$_SERVER` array to see which request you're handling
- handle a GET by showing the form; handle a POST by processing the submitted form data

Uploading files (.html)

```
<form action="params.php"
      method="post" enctype="multipart/form-data">
  Upload an image as your avatar:
  <input type="file" name="avatar" />
  <input type="submit" />
</form>
```

HTML

Upload an image as your avatar: No file selected.

output

- add a file upload to your form as an input tag with type of file
- **must also set the enctype attribute of the form**
- use the method "post"

http://www.tutorialspoint.com/php/php_get_post.htm

Processing an uploaded file in PHP

- uploaded files are placed into global array `$_FILES`, not `$_POST`
- each element of `$_FILES` is itself an associative array, containing:
 - `name` : the local filename that the user uploaded
 - `type` : the MIME type of data that was uploaded, such as `image/jpeg`
 - `size` : file's size in bytes
 - `tmp_name` : a filename where PHP has temporarily saved the uploaded file
 - to permanently store the file, move it from this location into some other file

Uploading details

```
<input type="file" name="avatar" />
```

HTML

Browse... No file selected.

Submit Query

output

- example: if you upload borat.jpg as a parameter named avatar,
 - \$_FILES["avatar"]["name"] will be "borat.jpg"
 - \$_FILES["avatar"]["type"] will be "image/jpeg"
 - \$_FILES["avatar"]["tmp_name"] will be something like "/var/tmp/phpZtR4TI"

Processing uploaded file, example

```
$username = $_POST["username"];  
if (is_uploaded_file($_FILES["avatar"]["tmp_name"])) {  
    move_uploaded_file($_FILES["avatar"]["tmp_name"],  
        "$username/avatar.jpg");  
    print "Saved uploaded file as $username/avatar.jpg\n";  
} else {  
    print "Error: required file not uploaded";  
}
```

PHP

- functions for dealing with uploaded files:
 - `is_uploaded_file(filename)`
 - returns TRUE if the given filename was uploaded by the user
 - `move_uploaded_file(from, to)`
 - moves from a temporary file location to a more permanent file
- proper idiom: check `is_uploaded_file`, then do `move_uploaded_file`

Exercise 2: Fibonacci

Modify fibonacci.html to print the first 20 Fibonacci numbers in an ordered list
Using PHP.

Recall that in the fibonacci sequence, each number is the sum of the two previous numbers, with the first and second numbers being 0 and 1.

Hint: a simple for loop will do

Don't forget to use `<? $variablename ?>`
`<?= $first ?>`

Put the results in a `` element

```
<li><?= $first ?></li>
```

Lab: music list

- Brian has written a page named music.php to display his MP3s. He has already written the basic HTML and CSS code. But the HTML is redundant, and he must manually edit it every time he adds songs to his collection.
- In this lab, you will turn this page into dynamic PHP program that removes redundancy and dynamically lists the MP3 on the computer.
- Download the music.php and viewer. css

Exercise: Variable for Song Count

- Brian has 1234 songs (123 hours of music). Turn this into a PHP variable you can change.
- Download music.php and open it with your text editor.
- Add a variable for the number of songs, and use it in HTMP/PHP code. Change its value to something other than 1234, say, 5678
- Express the number of hours of music items of your variable. Assume that 10 songs can be played per hour.
- Execute Music.php in your browser and see the change.

My Music Page

I love music. I have 5678 total songs, which is over 567.8 hours of music!

Yahoo! Top Music News

Exercise: Loops for link

- The five Google music result links at the top of the page are redundant, so express them using for loop.
- You should be able to change your loop bound to get more/fewer page links. Try that Also, the links should still work when you click on them.

I love music. I have 5678 total songs, which is over 567.8 hours of music!

Google's Music Results

1. [Page 1](#)
2. [Page 2](#)
3. [Page 3](#)
4. [Page 4](#)
5. [Page 5](#)
6. [Page 6](#)
7. [Page 7](#)

Exercise: Array of Favorite Artists

- The page lists Brian's favorite artists as bulleted list:
- Create an array containing the favorite artists' names and use the array to generate the bullets in the list.
- Test that it is working by adding at least one more favorite artist (e.g. Lady Gaga)

My Favorite Artists

1. Adele
2. Rihanna
3. Taylor Swift
4. Justin Bieber

Exercise: File of Favorite Artists

- Instead of writing your own array of fav artists, read them from a file, favorite.txt.
- Download this file to your machine and upload to your localhost folder. Read the artists from it, one per line.
- For added challenge, make each artists name a link.
- Check PHP's [string functions](#).
- Example: Justin Bieber links to <https://www.vevo.com/artist/justin-bieber>

Exercise: list Mp3 files

- The page hard-codes the list of MP3 file names. Change it to read file names from the server.
- Put a few your favorite .mp3 files in a folder called “yourfolder/songs”.
- Comment out the existing list of MP3/playlist file names on the page
- Write PHP code that looks for the names of all .mp3 files in the folder using the **glob** function.
- Print each file names as an **li** list item with the class **mp3item**.

Exercise: output

Your MP3 list should look like this when you are done:

My Music and Playlists

-  154 Rap.mp3 (0 KB)
-  Be More.mp3 (5311 KB)
-  Drift Away.mp3 (5590 KB)
-  Hello.mp3 (1827 KB)
-  Just Because.mp3 (4582 KB)
-  Panda Sneeze.mp3 (0 KB)

Take home read

PHP tutorials for beginners (really great tutorials):

<http://www.homeandlearn.co.uk/php/php.html>

PHP, Ajax, JavaScript

<http://blog.teamtreehouse.com/beginners-guide-to-ajax-development-with-php>