Web Programming Step by Step

Lecture 8
PHP File I/O and Query Parameters
Reading: 5.4 - 5.5

5.4: Advanced PHP Syntax

Functions (5.4.1)

```
function name(parameterName, ..., parameterName) {
  statements;
}
```

PHP

```
function quadratic($a, $b, $c) {
  return -$b + sqrt($b * $b - 4 * $a * $c) / (2 * $a);
}
```

PHP

- parameter types and return types are not written
- a function with no `return` statements implicitly returns `NULL`

Calling functions

```
name(expression, ..., expression);
```

PHP

```
$x = -2;
$a = 3;
$root = quadratic(1, $x, $a - 2);
```

PHP

- if the wrong number of parameters are passed, it's an error

Default parameter values

```php
function name(parameterName, ..., parameterName) {
  statements;
}
```
PHP

```php
function print_separated($str, $separator = ", ") {
  if (strlen($str) > 0) {
    print $str[0];
    for ($i = 1; $i < strlen($str); $i++) {
      print $separator . $str[$i];
    }
  }
}
```
PHP

```php
print_separated("hello");        # h, e, l, l, o
print_separated("hello", "-");   # h-e-l-l-o
```
PHP

- if no value is passed, the default will be used (defaults must come last)

Variable scope: global and local vars

```php
$school = "UW";                  # global
...

function downgrade() {
  global $school;
  $suffix = "Tacoma";            # local

  $school = "$school $suffix";
  print "$school\n";
}
```
PHP

- variables declared in a function are **local** to that function
- variables not declared in a function are **global**
- if a function wants to use a global variable, it must have a `global` statement

6.1: Parameterized Pages

Query strings and parameters (6.1.1)

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

```
http://www.google.com/search?q=Obama
http://example.com/student_login.php?username=stepp&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 `stepp`, 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: `$_REQUEST` (6.4.2)

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

- `$_REQUEST["`*parameter name*`"]` returns a parameter's value as a string
- test whether a given parameter was passed with `isset`

5.4: PHP File Input/Output

PHP file I/O functions (5.4.5)

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

Reading/writing files

contents of foo.txt	**`file("foo.txt")`**	**`file_get_contents("foo.txt")`**

`Hello`
`how are`
`you?`
 	
`I'm fine`
	`array(`
	`  "Hello\n",      # 0`
	`  "how are\n",    # 1`
	`  "you?\n",       # 2`
	`  "\n",           # 3`
	`  "I'm fine\n"    # 4`
	`)`
		`"Hello\n`
		`how are\n`
		`you?\n`
		`\n`
		`I'm fine\n"`

- `file` returns lines of a file as an array
- `file_get_contents` returns entire contents of a file as a string

Reading/writing an entire file

```php
# reverse a file
$text = file_get_contents("poem.txt");
$text = strrev($text);
file_put_contents("poem.txt", $text);
```

PHP

- `file_get_contents` returns entire contents of a file as a string
  - if the file doesn't exist, you'll get a warning
- `file_put_contents` writes a string into a file, replacing any prior contents

Appending to a file

```php
# add a line to a file
$new_text = "P.S. ILY, GTG TTYL!~";
file_put_contents("poem.txt", $new_text, FILE_APPEND);
```

PHP

old contents	new contents
Roses are red, Violets are blue. All my base, Are belong to you.	Roses are red, Violets are blue. All my base, Are belong to you. P.S. ILY, GTG TTYL!~

- `file_put_contents` can be called with an optional third parameter
- appends (adds to the end) rather than replacing previous contents

The `file` function

```
# display lines of file as a bulleted list
$lines = file("todolist.txt");
foreach ($lines as $line) {
  ?>
  <li> <?= $line ?> </li>
  <?php
}
```
PHP

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

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

- common idiom: `foreach` loop over lines of file

Unpacking an array: `list`

```
list($var1, ..., $varN) = array;
```
PHP

```
$values = array("stepp", "17", "m", "94");
...
list($username, $age, $gender, $iq) = $values;
```
PHP

- the `list` function accepts a comma-separated list of variable names as parameters
- can be assigned from an array (or the result of a function that returns an array)
- use this to quickly "unpack" an array's contents into several variables
  - a convenience, so you can refer to `$username` instead of `$values[0]`, etc.

Fixed-length files, `file` and `list`

contents of input file `personal.txt`

```
Marty Stepp
(206) 685-2181
570-86-7326
```

```php
list($name, $phone, $ssn) = file("personal.txt");
...
```
PHP

- when you know a file's exact length/format, you can use `file` and `list` to quickly examine it
- reads the file into an array of lines and unpacks the lines into variables

Splitting/joining strings

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

```php
$s  = "CSE 190 M";
$a  = explode(" ", $s);      # ("CSE", "190", "M")
$s2 = implode("...", $a);    # "CSE...190...M"
```
PHP

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

Example with `explode`

contents of input file `names.txt`

```
Martin D Stepp
Jessica K Miller
Victoria R Kirst
```

```
foreach (file("names.txt") as $name) {
  list($first, $mid, $last) = explode(" ", $name);
  ?>

  <p> author: <?= $last ?>, <?= $first ?> </p>

  <?php
}
```
PHP

```
author: Stepp, Marty

author: Miller, Jessica

author: Kirst, Victoria
```
output

Reading directories

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

- `glob` can filter by accepting wildcard paths with the `*` character

glob example

```
# reverse all poems in the poetry directory
$poems = glob("poetry/poem*.dat");
foreach ($poems as $poemfile) {
  $text = file_get_contents($poemfile);
  file_put_contents($poemfile, strrev($text));
  print "I just reversed " . basename($poemfile);
}
```
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"
  - `glob("lecture*/slides*.ppt")` examines all directories whose names begin with `lecture` and grabs all files whose names begin with "slides" and end with ".ppt"
- the `basename` function strips any leading directory from a file path
  - `basename("foo/bar/baz.txt")` returns `"baz.txt"`

`scandir` example

```
<ul>
  <?php
  $folder = "taxes/old";
  foreach (scandir($folder) as $filename) {
    ?>
    <li> <?= $filename ?> </li>
    <?php
  }
  ?>
</ul>
```
PHP

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

output

- annoyingly, the current directory (`"."`) and parent directory (`".."`) are included in the array
- don't need `basename` with `scandir` because it returns the file's names only