中山大學
SUN YAT-SEN UNIVERSITY

Lecture 7
More PHP for Server Side Programming

SE-805 Web 2.0 Programming

(http://my.ss.sysu.edu.cn/wiki/display/W2PSC/Home , **supported by Google;**
using some slides of & inspired by Marty Stepp's CSE 190 M courseware)

School of Software, Sun Yat-sen University

Outline

- **Embedded PHP**
- Function & Scope
- File I/O Operations

Don't print HTML tag in PHP

```
<?php
print "<!DOCTYPE html PUBLIC \"-//W3C//DTD XHTML 1.1//EN\"\n";
print " \"http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd\">\n";
print "<html xmlns=\"http://www.w3.org/1999/xhtml\">\n";
print "  <head>\n";
print "    <title>Geneva's web page</title>\n";
...
for ($i = 1; $i <= 10; $i++) {
  print "<p> I can count to $i! </p>\n";
}
?>
```

- printing HTML tags with print statements is bad style and error-prone:
  - must quote the HTML and escape special characters, e.g. \"
  - best PHP style is to minimize print/echo statements in embedded PHP code
- but without print, how do we insert dynamic content into the page?

PHP expression blocks

```
<?= expression ?>
```
PHP

```
<h2> The answer is <?= 6 * 7 ?> </h2>
```
PHP

The answer is 42

output

- PHP expression block: a small piece of PHP that evaluates and embeds an expression's value into HTML
  - <?= *expression* ?> is equivalent to:

```
<?php print expression; ?>
```
PHP

  - useful for embedding a small amount of PHP (a variable's or expression's value) in a large block of HTML without having to switch to "PHP-mode"

Expression block example

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
 "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <head><title>CSE 190 M: 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>
```

PHP

Common errors

```
...
  <body>
    <p>Watch how high I can count:
      <?php
      for ($i = 1; $i <= 10; $i++) {
        ?>
        <? $i ?>
    </p>
  </body>
</html>
```

PHP

- </body> and </html> above are inside the for loop, which is never closed
- if you forget to close your braces, you'll see an error about 'unexpected $end'
- if you forget **=** in **<?=**, the expression does not produce any output

Complex expression blocks

```
...
  <body>
    <?php
    for ($i = 1; $i <= 3; $i++) {
      ?>
      <h<?= $i ?>>This is a level <?= $i ?> heading.</h<?= $i ?>>
      <?php
    }
    ?>
  </body>
```

PHP

This is a level 1 heading.

This is a level 2 heading.

This is a level 3 heading.

output

- expression blocks can even go inside HTML tags and attributes

Including files: include , require

```php
include("filename");
```

```php
include("header.php");
```

- inserts the entire contents of the given file into the PHP script's output page
- encourages modularity
- useful for defining reused functions needed by multiple pages
- **require** is almost same as **include**, but different when the target was not found
  - the script with include will continue run with a warning message dumped to the output
  - the script with require will stop running and dump an error to the output

Outline

- Embedded PHP
- **Function & Scope**
- File I/O Operations

Functions

```
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 parameters values

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

```
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

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

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

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

- 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

Outline

- Embedded PHP
- Function & Scope
- **File I/O Operations**

PHP file I/O functions

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);
```

- 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);
```

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

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

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

- 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");
...
```

- 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

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

```
$s  = "SE 805 2010";
$a  = explode(" ", $s);      # ("SE", "805", "2010")
$s2 = implode("...", $a);    # "SE...805...2010"
```
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`

```
Tim D Lee
Alex F Kirlik
Peter R Priolli
```

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

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

  <?php
}
```

PHP

author: Lee, Tim

author: Kirlik, Alex

author: Priolli, Peter

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

```php
# 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

```php
<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

Summary

- Embedded PHP
  - expression blocks, common errors
  - include vs. require
- Function & Scope
  - calling, parameters
  - variable scope
- File I/O Operations
  - reading/writing/appending
  - file, list
  - explode/implode
  - reading directories
  - glob, scandir

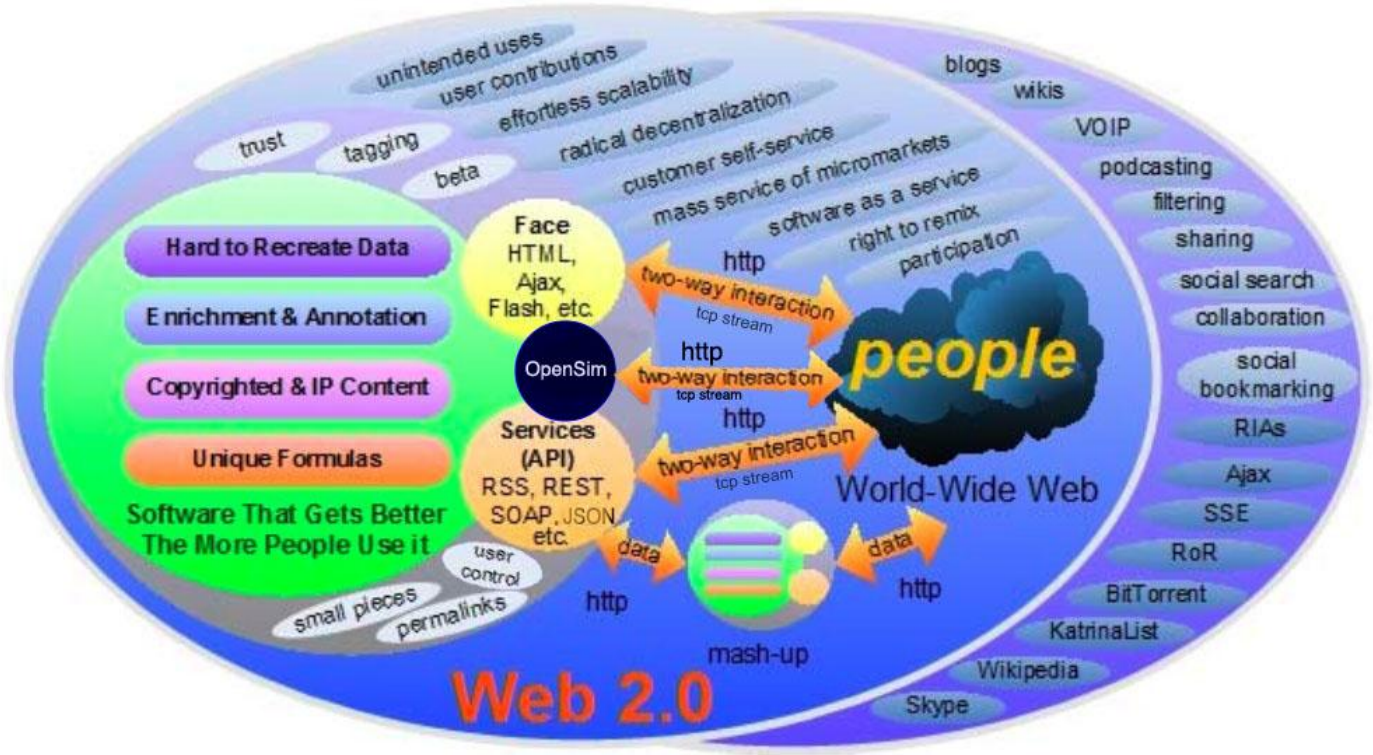
Exercises

- step 1: write a php script listing all files of a folder into a output text file
  - the folder has more than three levels sub-folders, recursively listed all files
- step 2: alter the script in step 1 by filtering the .exe, .com, .sys, and .bat files from the list
- step 3: wrap operations of step 2 into a function which can recursively list all files within a given path, and then write a php page to list all files within your web server's root

Further Readings

- PHP home page: http://www.php.net/
- W3Schools PHP tutorial: http://www.w3schools.com/PHP/
- Practical PHP Programming: http://hudzilla.org/phpwiki/
- PHP Cookbook: http://commons.oreilly.com/wiki/index.php/PHP_Cookbook

Thank you!


unintended uses
user contributions
effortless scalability
trust
tagging
beta
radical decentralization
customer self-service
mass service of micromarkets
software as a service
right to remix
participation
blogs
wikis
VOIP
podcasting
filtering
sharing
social search
collaboration
social bookmarking
RIAs
Ajax
SSE
RoR
BitTorrent
KatrinaList
Wikipedia
Skype
Hard to Recreate Data
Enrichment & Annotation
Copyrighted & IP Content
Unique Formulas
Software That Gets Better The More People Use it
Face
HTML, Ajax, Flash, etc.
OpenSim
Services (API)
RSS, REST, SOAP, JSON etc.
user control
small pieces
permalinks
http
two-way interaction
tcp stream
people
World-Wide Web
data
mash-up
Web 2.0