PHP Web Services

CSE 190 M (Web Programming) Spring 2008
University of Washington

Lecture outline

- What is a web service?
- Arrays
- Query parameters
- File I/O

What is a web service?

web service: a software system exposing useful functionality that can be invoked through the internet using common protocols

- a web service is sort of like a remote function (or set of functions) that you can call by contacting a program on a web server
- many web services accept parameters and produce results
- web services can be written in PHP and contacted by the browser in XHTML and/or Ajax code

Arrays

Creating an array

```
$name = array();                                # create
$name = array(value0, value1, ..., valueN);

$name[index]                                    # get element value
$name[index] = value;                           # set element value
$name[] = value;                                # append
```
PHP

```
$a = array();     # empty array (length 0)
$a[0] = 23;       # stores 23 at index 0 (length 1)
$a2 = array("some", "strings", "in", "an", "array");
$a2[] = "Ooh!";   # add string to end (at index 5)
```
PHP

- to append, use bracket notation without specifying an index
- element type is not specified; can mix types

Array functions

- `count` : number of elements in the array
- `print_r` : print array's contents
- using as a list: `array_pop`, `array_push`, `array_shift`, `array_unshift`
- searching and reordering: `in_array`, `array_search`, `array_reverse`, `sort`, `rsort`, `shuffle`
- creating, filling, filtering: `array_fill`, `array_merge`, `array_intersect`, `array_diff`, `array_slice`, `range`
- processing elements: `array_sum`, `array_product`, `array_unique`, `array_filter`, `array_reduce`

Array function example

```
$tas = array("MD", "BH", "KK", "HM", "JP");
for ($i = 0; $i < count($tas); $i++) {
  $tas[$i] = strtolower($tas[$i]);
}                                  # ("md", "bh", "kk", "hm", "jp")
$morgan = array_shift($tas);       # ("bh", "kk", "hm", "jp")
array_pop($tas);                   # ("bh", "kk", "hm")
array_push($tas, "ms");            # ("bh", "kk", "hm", "ms")
array_reverse($tas);               # ("ms", "hm", "kk", "bh")
sort($tas);                        # ("bh", "hm", "kk", "ms")
$best = array_slice($tas, 1, 2);   # ("hm", "kk")
```
PHP

foreach loop

```
foreach ($array as $variableName) {
  ...
}
```
PHP

```
$stooges = array("Larry", "Moe", "Curly", "Shemp");
~~for ($i = 0; $i < count($stooges); $i++) {~~
~~  print "Moe slaps {$stooges[$i]}\n";~~
~~}~~
foreach ($stooges as $stooge) {
  print "Moe slaps $stooge\n";  # even himself!
}
```
PHP

- a convenient way to loop over each element of an array without indexes

Splitting/joining strings

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

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

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

Unpacking an array: `list`

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

```
$line = "stepp:17:m:94";
list($username, $age, $gender, $iq) = explode(":", $line);
```
PHP

- the `list` function accepts a comma-separated list of variable names as parameters
- assign an array (or the result of a function that returns an array) to store that array's contents into the variables

Non-consecutive arrays

```
$autobots = array("Optimus", "Bumblebee", "Grimlock");
$autobots[100] = "Hotrod";
```
PHP

- the indexes in an array do not need to be consecutive
- the above array has a `count` of 4, with 97 blank elements between `"Grimlock"` and `"Hotrod"`

Associative arrays

```
$blackbook = array();
$blackbook["marty"] = "206-685-2181";
$blackbook["stuart"] = "206-685-9138";
...
print "Marty's number is " . $blackbook["marty"] . ".\n";
```
PHP

- **associative array** (a.k.a. **map**, **dictionary**, **hash table**) : an array that uses non-integer indexes
- associates a particular index "key" with a value
  - key `"marty"` maps to value `"206-685-2181"`
- syntax for embedding an associative array element in interpreted string:

  ```
  print "Marty's number is {$blackbook['marty']}.\n";
  ```
  PHP

Creating an associative array

```php
$name = array();
$name["key"] = value;
...
$name["key"] = value;
```
PHP

```php
$name = array(key => value, ..., key => value);
```
PHP

```php
$blackbook = array("marty" => "206-685-2181",
                   "stuart" => "206-685-9138",
                   "jenny" => "206-867-5309");
```
PHP

- an associative array can be declared either initially empty, or with a set of predeclared key/value pairs

Printing an associative array

```php
print_r($blackbook);
```
PHP

```
Array
(
    [jenny] => 206-867-5309
    [stuart] => 206-685-9138
    [marty] => 206-685-2181
)
```

- `print_r` function displays all keys/values in the array
- `var_dump` function is much like `print_r` but prints more info
- unlike `print`, these functions require parentheses

foreach loop and associative arrays

```php
foreach ($blackbook as $key => $value) {
  print "$key's phone number is $value\n";
}
```
PHP

```
jenny's phone number is 206-867-5309
stuart's phone number is 206-685-9138
marty's phone number is 206-685-2181
```

- both the key and the value are given a variable name
- the elements will be processed in the order they were added to the array

Associative array functions

```php
# if (!isset($blackbook["marty"]))
if (!array_key_exists("marty", $blackbook)) {
  print "No phone number found for Marty Stepp.\n";
}
```

- `array_key_exists` : whether the array contains a value for the given key
- `array_keys`, `array_values` : all keys or all values in the array
- `asort`, `arsort` : sorts by value, in normal or reverse order
- `ksort`, `krsort` : sorts by key, in normal or reverse order

Query parameters

Scripts that accept query string parameters from their web requests

The main idea

```
https://example.com/student_login.php?username=stepp&sid=1234567
```

- almost every interesting web service requires parameters to guide its behavior
- parameters are passed as a query string in the HTTP GET or POST request
  - above, parameter `username` has value `stepp`, and `sid` has value `1234567`
- PHP code can examine and utilize the value of these parameters
- in PHP, query parameters are exposed as elements of global associative arrays

Query parameters: $_REQUEST

```php
$user_name = $_REQUEST["username"];
$student_id = (int) $_REQUEST["sid"];
```

PHP

- `$_REQUEST["`*parameter name*`"]`
  returns (as a string) the value of the query parameter with that name
- if no such parameter was passed, you'll get a warning when trying to access it; test for this with `isset`
- other special global arrays: `$_GET`, `$_POST`, `$_COOKIE`, `$_SERVER`, `$_FILES`, `$_ENV`, `$_SESSION`

Example: Exponent web service

```
header("Content-type: text/plain");

$base = $_REQUEST["base"];
$exp = $_REQUEST["exponent"];
$result = pow($base, $exp);

print "$base ^ $exp = $result\n";
```
PHP

```
http://example.com/exponent.php?base=3&exponent=4
```

```
3 ^ 4 = 81
```

Example: Print-all-parameters web service

```
header("Content-type: text/plain");

foreach ($_REQUEST as $param => $value) {
  print "Parameter $param has value $value\n";
}
```
PHP

```
http://example.com/print_params.php?name=Marty+Stepp&sid=1234567
```

```
Parameter name has value Marty Stepp
Parameter sid has value 1234567
```

Checking for a parameter's existence

```
if (isset($_REQUEST["creditcard"])) {
  $cc = $_REQUEST["creditcard"];
  ...
} else {
  print "You did not submit a credit card number.\n";
  ...
  return;
}
```
PHP

- `isset` function returns TRUE if a given variable/element has a value
  - you can also use the `array_key_exists` function for this
- if a required parameter is missing, you can abort the rest of script using `return;` or the `die` function

GET or POST?

```
if ($_SERVER["REQUEST_METHOD"] == "GET") {
  # process a GET request
  ...
} elseif ($_SERVER["REQUEST_METHOD"] == "POST") {
  # process a POST request
  ...
}
```
PHP

- some PHP web services process both GET and POST requests
- can find out which kind of request we are currently processing by looking at the `"REQUEST_METHOD"` key of the global `$_SERVER` array

URL-encoding

- certain characters are not allowed in URL query parameters:
  - examples: `" "`, `"/"`, `"="`, `"&"`
- whenever you want to pass a query parameter that does contain one of these characters, it must be **URL-encoded**
  - `"Marty's cool!?"` → `"Marty%27s+cool%3F%21"`
- you don't usually need to worry about this:
  - JavaScript Ajax requests automatically URL-encode their parameters
  - PHP scripts that accept query parameters automatically URL-decode them
  - ... but occasionally the weird encoded version does pop up
    (e.g. when debugging queries in Firebug)

File I/O

Interacting with files and directories in PHP

PHP file I/O functions

- reading/writing entire files: `file_get_contents`, `file_put_contents`
- asking for information: `file_exists`, `filesize`, `fileperms`, `filemtime`, `is_dir`, `is_readable`, `is_writable`, `disk_free_space`
- manipulating files and directories: `copy`, `rename`, `unlink`, `chmod`, `chgrp`, `chown`, `mkdir`, `rmdir`
- reading directories: `scandir`, `glob`

Reading/writing files

```
$text = file_get_contents("schedule.txt");
$lines = explode("\n", $text);
$lines = array_reverse($lines);
$text = implode("\n", $lines);
file_put_contents("schedule.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

Reading files example

```
# Returns how many lines in this file are empty or just spaces.
function count_blank_lines($file_name) {
  $text = file_get_contents($file_name);
  $lines = explode("\n", $text);
  $count = 0;
  foreach ($lines as $line) {
    if (strlen(trim($line)) == 0) {
      $count++;
    }
  }
  return $count;
}
...
print count_blank_lines("lecture18-php_web_services.html");
```
PHP

Reading directories

```php
$folder = "images";
$files = scandir($folder);
foreach ($files as $file) {
  if ($file != "." && $file != "..") {
    print "I found an image: $folder/$file\n";
  }
}
```
PHP

- `scandir` returns an array of all files in a given directory
- annoyingly, the current directory (`"."`) and parent directory (`".."`) are included in the array; you probably want to skip them

Headers

```php
header("Content-type: text/plain");
```
PHP

- by default, a PHP script's output is assumed to be HTML
- use the `header` function to specify non-HTML output
  - must appear before any other output generated by the script
- `header` can also be used to send back HTTP error codes:
  - `header("Content-type: text/plain");`
  - `header("Content-type: application/xml");`
  - `header("HTTP/1.1 400 Invalid Request");`
  - `header("HTTP/1.1 404 File Not Found");`
  - `header("HTTP/1.1 500 Server Error");`