CSE 154

LECTURE 25: WEB SERVICES

What is a web service?

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
  - examples seen in CSE 154: `quote.php`, `animalgame.php`, `books_json.php`, `urban.php`, `babynames.php`

Setting content type with header

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

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

- by default, a PHP file's output is assumed to be HTML (text/html)
- use the header function to specify non-HTML output
  - must appear before any other output generated by the script

Recall: Content ("MIME") types

MIME type	related file extension
text/plain	.txt
text/html	.html, .htm, ...
text/xml	.xml
application/json	.json
text/css	.css
text/javascript	.js
image/gif	.gif

- Lists of MIME types: by type, by extension

Example: Exponent web service

Write a web service that accepts a `base` and `exponent` and outputs `base` raised to the `exponent` power. For example, the following query should output `81` :

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

solution:

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

Exercise: Baby name web service

- Write a web service that accepts a `name` and `gender` and finds and outputs the line from text file `rank.txt` with information about that name:

```
Aaron m 147 193 187 199 250 237 230 178 52 34 34 41 55
Lisa f 0 0 0 0 0 733 220 6 2 16 64 295 720
...
```

- For the following call:

```
http://example.com/babynames.php?name=Lisa&gender=f
```

- The service should output the following line:

```
Lisa f 0 0 0 0 0 733 220 6 2 16 64 295 720
```

What about errors?

- What if the user doesn't pass an important parameter?

```
http://example.com/babynames.php?gender=f          (no name passed!)
```

- What if the user passes a name that is not found in the file?

```
http://example.com/babynames.php?name=Borat&gender=m    (not found in file)
```

- What is the appropriate behavior for the web service?

Reporting errors

web service should return an HTTP "error code" to the browser, possibly followed by output

- error messages (`print`) are not ideal, because they could be confused for normal output
- these are the codes you see in Firebug's console and in your Ajax request's `status` property

HTTP code	Meaning
200	OK
301-303	page has moved (permanently or temporarily)
400	illegal request
401	authentication required
403	you are forbidden to access this page
404	page not found
410	gone; missing data or resource
500	internal server error
complete list	

Using headers for HTTP error codes

```
header("HTTP/1.1  code  description");
```
PHP

```
if ($_GET["foo"] != "bar") {
  # I am not happy with the value of foo; this is an error
  header("HTTP/1.1 400 Invalid Request");
  die("An HTTP error 400 (invalid request) occurred.");
}
```
PHP

```
if (!file_exists($input_file_path)) {
  header("HTTP/1.1 404 File Not Found");
  die("HTTP error 404 occurred: File not found ($input_file_path)");
}
```
PHP

- `header` can also be used to send back HTTP error codes
  - `header("HTTP/1.1 403 Forbidden");`
  - `header("HTTP/1.1 404 File Not Found");`
  - `header("HTTP/1.1 500 Server Error");`

Checking for a mandatory query parameter

```php
function get_query_param($name) {
  if (!isset($_GET[$name])) {
    header("HTTP/1.1 400 Invalid Request");
    die("HTTP/1.1 400 Invalid Request: missing required parameter '$name'");
  }
  if ($_GET[$name] == "") {
    header("HTTP/1.1 400 Invalid Request");
    die("HTTP/1.1 400 Invalid Request: parameter '$name' must be non-empty");
  }
  return $_GET[$name];
}
```

PHP

The $_SERVER superglobal array

index	description	example
$_SERVER["SERVER_NAME"]	name of this web server	"webster.cs.washington.edu"
$_SERVER["SERVER_ADDR"]	IP address of web server	"128.208.179.154"
$_SERVER["REMOTE_HOST"]	user's domain name	"hsd1.wa.comcast.net"
$_SERVER["REMOTE_ADDR"]	user's IP address	"57.170.55.93"
$_SERVER["HTTP_USER_AGENT"]	user's web browser	"Mozilla/5.0 (Windows; ..."
$_SERVER["HTTP_REFERER"]	where user was before this page	"http://www.google.com/"
$_SERVER["REQUEST_METHOD"]	HTTP method used to contact server	"GET" or "POST"

- call `phpinfo();` to see a complete list

GET or POST?

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

PHP

- some web services process both GET and POST requests
- to find out which kind of request we are currently processing, look at the global `$_SERVER` array's `"REQUEST_METHOD"` element

Emitting partial-page HTML data

```php
# suppose my web service accepts a "type" query parameter ...
<?php if ($_GET["type"] == "html") { ?>
  <ul>
    <?php foreach ($students as $kid) { ?>
      <li> <?= $kid ?> </li>
    <?php } ?>
  </ul>
<?php } ?>
```

PHP

- some web services do output HTML, but not a complete page
- the partial-page HTML is meant to be fetched by Ajax and injected into an existing page

Exercise: Baby name web service XML

- Modify our `babynames.php` service to produce its output as XML. For the data:

```
Morgan m 375 410 392 478 579 507 636 499 446 291 278 332 518
```

- The service should output the following XML:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<baby name="Morgan" gender="m">
  <rank year="1890">375</rank>
  <rank year="1900">410</rank>
  ...
  <rank year="2010">518</rank>
</baby>
```

XML

Emitting XML data manually

```
...
header("Content-type: text/xml");
print "<?xml version=\"1.0\" encoding=\"UTF-8\"?>\n";
print "<books>\n";
foreach ($books as $book) {
  print "  <book title=\"{$book['title']}\" author=\"{$book['author']}\" />\n";
}
print "</books>\n";
```
XML

- specify a content type of `text/xml` or `application/xml`
- print an XML prologue (the `<?xml` line), then print XML data as output
  - **important:** no whitespace output can precede the prologue; must be `print`ed
- messy; bad to embed XML syntax in `print`s; write-only (hard to read existing XML data)

PHP's XML DOM: DOMDocument

The PHP DOMDocument class represents an XML document. It has these methods:

createElement(*tag*)	create a new element node to add to the document
createTextNode(*text*)	create a new text node to add to the document
getElementById(*id*), getElementsByTagName(*tag*)	search for elements in the document
load(*filename*), loadXML(*string*)	read XML data from a file on disk or from a string
save(*filename*), saveXML()	write XML data to a file on disk or returns it as a string
validate()	return whether the current document consists of valid XML data

PHP's XML DOM: DOMElement

The PHP `DOMElement` class represents each DOM element. It has these fields/methods:

tagName, nodeValue	node's name (tag) and value (text)
parentNode, childNodes, firstChild, lastChild, previousSibling, nextSibling	references to nearby nodes
appendChild(*DOMNode*), insertBefore(*newNode*, *oldNode*), removeChild(*DOMNode*)	manipulate this node's list of children
getElementsByTagName(*tag*)	search for descendent elements within this element
getAttribute(*name*), setAttribute(*name*, *value*), removeAttribute(*name*)	get/set the value of an attribute on this tag

PHP XML DOM example

```
...
$xmldoc = new DOMDocument();                                  # <?xml version="1.0"?>
$books_tag = $xmldoc->createElement("books");
$xmldoc->appendChild($books_tag);                             # <books>
foreach ($books as $book) {
  $book_tag = $xmldoc->createElement("book");                 # <book
  $book_tag->setAttribute("title", $book["title"]);           # title="Harry Potter" />
  $book_tag->setAttribute("author", $book["author"]); # author="J.K. Rowling" />
  $books_tag->appendChild($book_tag);
}                                                             # </books>
header("Content-type: text/xml");
print $xmldoc->saveXML();
```

- much easier to read/write/manipulate complex XML
- `saveXML` automatically inserts the XML prolog for us

Exercise solution: Baby name web service XML

```
# takes a line of rankings and produces XML in the specified format
# example:  Aaron m 147 193 187 199 250 237 230 178 52 34 34 41 55
function generate_xml($line, $name, $gender) {
  $xmldom = new DOMDocument();
  $baby_tag = $xmldom->createElement("baby");      # <baby>
  $baby_tag->setAttribute("name", $name);
  $baby_tag->setAttribute("gender", $gender);

  $year = 1890;
  $tokens = explode(" ", $line);
  for ($i = 2; $i < count($tokens); $i++) {
    $rank_tag = $xmldom->createElement("rank");    # <rank>
    $rank_tag->setAttribute("year", $year);
    $rank_tag->appendChild($xmldom->createTextNode($tokens[$i]));
    $baby_tag->appendChild($rank_tag);
    $year += 10;
  }

  $xmldom->appendChild($baby_tag);
  return $xmldom;
}
```

Exercise: Baby name web service JSON

- Modify our `babynames.php` service to produce its output as JSON. For the data:

```
Morgan m 375 410 392 478 579 507 636 499 446 291 278 332 518
```

- The service should output the following JSON:

```
{
  "name": "Morgan",
  "gender": "m",
  "rankings": [375, 410, 392, 478, 579, 507, 636, 499, 446, 291, 278,
332, 518]
}
```

JSON

Emitting JSON data manually

```
...
header("Content-type: application/json");
print "{\n";
print "  \"books\": [\n";
foreach ($books as $book) {
  print "  {\"author\": \"{$book['author']}\", \"title\":
\"{$book['title']}\"}\n";
}
print "\n";
```

- specify a content type of `application/json`
- messy, just like when manually printing XML (not recommended)

PHP's JSON functions

PHP includes the following global functions for interacting with JSON data:

json_decode(*string*)	parses the given JSON data string and returns an equivalent associative array object (like JSON.parse in JavaScript)
json_encode(*object*)	returns JSON equivalent for the given object or array or value (like JSON.stringifyin JavaScript)

- `json_encode` will output associative arrays as objects and normal arrays as arrays

PHP JSON example

```php
<?php
$data = array(
  "library" => "Odegaard",
  "category" => "fantasy",
  "year" => 2012,
  "books" => array(
    array("title" => "Harry Potter", "author" => "J.K. Rowling"),
    array("title" => "The Hobbit", "author" => "J.R.R. Tolkien"),
    array("title" => "Game of Thrones", "author" => "George R. R. Martin"),
    array("title" => "Dragons of Krynn", "author" => "Margaret Weis"),
  )
);

header("Content-type: application/json");
print json_encode($data);
?>
```

PHP

PHP JSON example - output

```
{
  "library": "Odegaard",
  "category": "fantasy",
  "year": 2012,
  "books": [
    {"title": "Harry Potter", "author": "J.K. Rowling"},
    {"title": "The Hobbit", "author": "J.R.R. Tolkien"},
    {"title": "Game of Thrones", "author": "George R. R. Martin"},
    {"title": "Dragons of Krynn", "author": "Margaret Weis"},
  ]
}
```

JSON

For reference: Provided web services code

- `quote.php`
- `animalgame.php`
- `books_json.php`
- `urban.php` *(caution: contains profanity)*
- `babynames.php`