Web Programming Step by Step

Lecture 22
Web 2.0 and Web Services

What is "Web 2.0"?

- **Web 2.0**: A set of ideas and technologies for creating modern, interactive web applications
  - Ajax, multimedia, streaming, stateful pages, cookies, user-generated content, 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 XHTML and/or Ajax code
- service's output is often not HTML but rather text, XML, or other content types

Content ("MIME") types (1.2.3)

MIME type	related file extension
`text/plain`	`.txt`
`text/html`	`.html, .htm, ...`
`text/css`	`.css`
`text/javascript`	`.js`
`text/xml`	`.xml`
`image/gif`	`.gif`
`image/jpeg`	`.jpg, .jpeg`
`video/quicktime`	`.mov`
`application/octet-stream`	`.exe`

- Lists of MIME types: by type, by extension

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 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

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:

```
header("Content-type: text/plain");
$base = $_REQUEST["base"];
$exp = $_REQUEST["exponent"];
$result = pow($base, $exp);
print $result;
```
PHP

Recall: HTTP `GET` vs. `POST` (6.3.3)

- HTTP: the set of commands understood by a web server and sent from a browser
- **`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, a `POST` request is more appropriate than a `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

The `$_SERVER` superglobal array

index	description	example
`$_SERVER["SERVER_NAME"]`	name of this web server	`"webster.cs.washington.e`
`$_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
  ...
}
```

- 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 `$_SERVER` array
- you can also access query parameters through `$_GET` and `$_POST` rather than `$_REQUEST`

Emitting partial-page HTML data

```php
# suppose my web service accepts a "type" query parameter ...
if ($_REQUEST["type"] == "html") {
  # client wants their output to be HTML format
  ?>
  <ul>
  <?php
  foreach ($students as $kid) {
    ?>
    <li> <?= $kid ?> </li>
    <?php
  }
  ?>
  </ul>
  <?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

Emitting XML data

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

- specify a content type of `text/xml` or `application/xml`
- print an XML prologue (the `<?xml` line) first
  - **important:** no whitespace output can precede the prologue
- then print each line of XML data/tags as output
- some PHP libraries automatically generate XML for you from other data (e.g. databases)

Reporting errors

- how does a web service indicate an error to the client?
  - error messages (`print`) are not ideal, because they could be confused for normal output
- web service should return an HTTP "error code" to the browser, possibly followed by 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
403	you are forbidden to access this page
404	page not found
500	internal server error
complete list	

Using headers for HTTP error codes

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

```
if ($_REQUEST["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");`