HTTP Headers

How browsers talk to servers

This is more of an outline than a tutorial. I wanted to give our web team a quick overview of what headers are and what they mean for client-server communication. Mainly, what HTTP headers mean for cookies and for other information that they see when they're programming on web pages.

It's unlikely I'll ever do this tutorial again, so this document is probably more useful as a starting point for giving a tutorial, rather than as a standalone tutorial.

What does this do?

```
<?
	setcookie('greeting', 'yes');
?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/
xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
	<head>
		<title>Mimsy Were the Borogoves</title>
		<meta http-equiv="Content-Type" content="text/html;charset=utf-8" />
		<meta name="description" content="Hello, World." />
	</head>
	<body>
		<h1>World</h1>

		<?IF ($_COOKIE['greeting'] == 'yes'):?>
			<p>World, say hello.</p>
		<?ELSE:?>
			<p>Please go away.</p>
		<?ENDIF;?>

	</body>
</html>
```

What will this page display? The answer has to do with HTTP headers and the way that browsers and servers communicate with each other.

Examining headers

curl

curl --head URL

HTTP Client

For looking at what the browser sends as well as what the server returns.

- http://ditchnet.org/httpclient/

Common headers

Headers always look like “Some-Name: some value”.

- Content-Type
- Date
- User-Agent
- Last-Modified
- Cache-Control

Cookies

Both the client and the server send headers. Among the headers that clients send are cookies, in the “Cookie:” header.

Setting cookies doesn’t make them available in the script you set them in, because cookies are headers. The headers have already been sent.

Trusting headers

Headers from clients can't be trusted; you can't trust User-Agent, Date, HTTP-Referer, or anything else that the client sends (just look at the options in curl and HTTP Client: they can be faked easily).

Manipulating headers

The "header()" function in PHP lets you manipulate headers, and this is most commonly how we do it when we need to set headers inside of a PHP script. Browsers give us a way of manipulating the headers they're storing for the page, in the HEAD of a web document. We use it to set the character set of our newer web pages:

```
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
```

This tells the browser to replace whatever it has for the Content-Type header and replace it with text/html; charset=UTF-8.

Finally, you can also manipulate headers in the .htaccess file of Apache servers. You can add or change them directly with the "header" directive. There are also directives for commonly-manipulated headers such as Content-Type (AddType) and Location (Redirect, RedirectPermanent, RedirectMatch).

```
AddType 'text/html; charset=UTF-8' .html
RedirectPermanent /as/ http://www.example.com/antstorm/
Header set Content-Type "text/html; charset=UTF-8"
Header add Timing "Hello Joe. It took %D microseconds for Apache to serve this
request."
Header unset Location
```

For more information, see these pages:

```
http://php.net/manual/function.header
http://www.w3schools.com/TAGS/att_meta_http_equiv.asp
http://httpd.apache.org/docs/2.2/mod/mod_headers.html
http://httpd.apache.org/docs/2.2/mod/mod_alias.html
http://httpd.apache.org/docs/2.2/mod/mod_mime.html
```

Redirects

One of the most common manipulations is to redirect the visitor to another page. You might want to force them to always use the secure version of a page, or you might want to create a single, simple redirect page that redirects different types of visitors to different pages.

```
header("Location: http://my.example.com/really/long/oracle.db/style.aspx/url/
AS5SK689HG");
```

Downloads

We've already looked at one header manipulation: the downloads class sends headers that cause the browser to download the file instead of display it.

```
header('Content-Type: audio/mpeg');
header('Content-Disposition: attachment; filename="Complicated Man.mp3"');
```

What is a content type?

The Internet uses MIME types to identify a type of file, so that the visitor's operating system can know what applications are able to open it. You can see more about MIME types at:

- http://www.w3schools.com/media/media_mimeref.asp
- http://en.wikipedia.org/wiki/Internet_media_type

MIME types are used by e-mail (just as most headers are) as well as web servers.

Host

On the client end:

```
Host: www.hoboes.com
Host: hoboes.com
Host: www.godsmonsters.com
```

Asking for http://www.hoboes.com/ is just like asking for http://216.92.252.156/; the client adds Host: www.hoboes.com (or www.godsmonsters.com) to let the server know which hostname is being requested. Before HTTP/1.1, every hostname had to have its own IP address.

In HTTP Client, you can set "Host" to each of the above examples and get a different page back each time. The IP address is the same in each case; the server looks at the Host header to see which hostname the browser wants to see.

Filling out forms, getting a cookie

In this part of the tutorial, we will log in to a remote server by sending the headers and body that get sent when a visitor fills out a form and submits it. We will then take the cookie that the remote server sends to the client on a successful login and use that cookie to maintain the session.

```
curl --form user=username --form password="password" https://www.example.com/
login/ --trace-ascii -
```

Look for the Set-Cookie: header from the server. Copy it out and use it in HTTP Client.

```
Cookie: cookie=value
Cookie: ExampleKey=79202038fea9608456ad6aa363173dea
```

In HTTP Client, send this body to http://www.example.com/login/:

```
user=username&password=password
```

with

```
Content-Type: application/x-www-form-urlencoded
```

to see the cookie and cookie value.

Then, look for the Set-Cookie header and use that cookie to enter the page.

If the password contains special characters, they may need to be encoded. For example, an ampersand will need to be %26, and an apostrophe %27.

Response codes

Code	Meaning
200	Success
301	Moved permanently
303	See other
307	Temporary redirect
403	Forbidden
404	Page not found
418	I'm a teapot
503	Service unavailable

See http://en.wikipedia.org/wiki/List_of_HTTP_status_codes for more information. Response codes are sent as part of the very first header, usually "HTTP/1.1 XXX Status Name", for example, "HTTP/1.1 503 Service Unavailable".

Common PHP Header Functions

header()

```
http://php.net/manual/function.header
```

The basic “header” function lets you send any header you want to the client. You can send any “Header: value”, and you can also set the return code:

```
header("HTTP/1.1 404 Not Found");
header($_SERVER["SERVER_PROTOCOL"]." 404 Not Found");
```

However, it makes it easier to set the response code. This is especially useful for redirects:

```
header("Location: http://www.example.com/new/location/", true, 301);
```

setcookie()

This lets you send cookies to the client without knowing the format for the Set-Cookie header.

```
setcookie("TestCookie", $value);
setcookie("TestCookie", $value, time()+3600);  /* expire in 1 hour */
setcookie("TestCookie", $value, time()+3600, "/~rasmus/", ".example.com", 1);
```

apache_request_headers()

This provides all of the headers that the client sent, as an associative array.

headers_sent()

This returns true or false, depending on whether or not headers have already been sent to the client.

Necessary tutorial files

- HTTP Client
- curl