Basics of HTTP

Relevant Links

- HTTP on MDN[1] good place to start
- Browser Networking[2] free online book
- HTTP specification (more a reference than a read)[3]
- HTTP the definitive guide[4] not free

Notes

HTTP

HTTP stands for HyperText Transfer Protocol. It is what is known as an "Application Layer Networking protocol". This means that it does not concern itself with how network packets will traverse the internet, but instead aims at a higher-level description of the information in those packets. It is the primary mode of communication between web browsers and web servers.

Here are its main characteristics.

stateless The protocol is "stateless". In other words, each request that the browser sends to the server has no memory of prior requests and replies.

This is an important feature of the protocol, and something that actually lent to its popularity. It makes the implementation of it on both the server and browser side easier, and makes the overall protocol simpler, as neither side needs to maintain any information from previous requests. Other protocols had been proposed around the same time, but HTTP won in the end as the standard, in large part due to its simplicity.

One of the consequences of course is that in situations where we need to maintain the history of what has occured, both browser and server need to agree on a way to do that (e.g. keeping someone "logged in").

client/server The HTTP protocol is characterized by the uneven description of the two parties involved. One party is the *client* (typically your web browser) and the other is the *server*. The client sends **requests** to the server, and the server sends back **responses**.

session The typical interaction between client and server, called a *session* would go something like this:

[1] https://developer.mozilla.org/en-US/docs/Web/HTTP
[2] http://chimera.labs.oreilly.com/books/1230000000545/index.html
[3] http://www.w3.org/Protocols/rfc2616/rfc2616.html
[4] http://shop.oreilly.com/product/9781565925090.do

1. Client establishes a network connection with server. This is typically done via TCP/IP and requires some amount of initial setup.
2. Client sends a request packet and waits for the answer.
3. Server receives the packet, then prepares and sends a response.
4. In HTTP 1.1 and later, the client may send further requests and receive responses.
5. When the client has no more requests, they close the connection.

Clients and servers specify themselves via their IP addresses (and port numbers). This is taken care of at the TCP/IP layer.

request The client sends an "HTTP request" to the server over the network. That request includes the **request method**, followed by the **resource path** as well as the protocol version, typically HTTP/1.1.

It will be followed a series of **headers**, that can identify various aspects of the request, like the content type, the host name, the content length, the accepted languages for the reply and so on.

Some request methods allow content, which can be found following the headers. Here is an example from the MDN page:

```
POST /contact_form.php HTTP/1.1
Host: developer.mozilla.org
Content-Length: 64
Content-Type: application/x-www-form-urlencoded

name=Joe%20User&request=Send%20me%20one%20of%20your%20catalogue
```

This says that the request uses the POST method and the resource path is /contact_form.php. There are 3 headers, one specifying the host, and two more specifying the details about the content. After that and a following empty line we find the content (name...).

Here are the main request methods. A browser can typically only use the first two.

GET A GET request asks for some resource and is not meant to cause any changes to the server.

POST A POST request is used when submitting forms for example. Is it typically expected to be used for updating some server information.

PUT Used for changing information of some server resource. Often performed via a POST instead.

HEAD The reply will contain just the header information, without any content body.

DELETE Used for deleting server resources (if allowed).

response Server responses have a special first line containing the protocol followed by the **response status code**. This is followed by response headers, and finally the content body of the response. Here is an example:

```
HTTP/1.1 200 OK
Date: Sat, 09 Oct 2010 14:28:02 GMT
Server: Apache
Last-Modified: Tue, 01 Dec 2009 20:18:22 GMT
ETag: "51142bc1-7449-479b075b2891b"
Accept-Ranges: bytes
Content-Length: 29769
Content-Type: text/html

<!DOCTYPE html... (here comes the 29769 bytes of the requested web page)
```

There are many different request and response headers, look at the documentation for more information.

Here are some typical response status codes, there are many more:

200 *OK*. The resource was available and is returned.
301 *Moved Permanently*. The resource has been moved to a different location. A location header gives the new location.
304 *Not Modified*. The resource hasn't been modified since the last time we asked for it. Our cached version will do just fine.
404 *Not Found*. The requested resource cannot be found. Typically the result of typos in the request.
500 *Internal Server Error*. Usually indicates problems with the server's configuration/availability.

You typically don't have to directly create these requests and responses as text, there are libraries that do that for you and allow you to talk about these responses on the level of objects. We will see some examples when we discuss XMLHttpRequest next. For now, you can use your browser to look at the requests and responses for this page.

Open up the developer tools and find the Network tab. Reload the page and you should see multiple network requests present, one of them is for an html page. Notice the method, content type and status. Then click further in one of those requests to see more details.