Introduction to HTTP

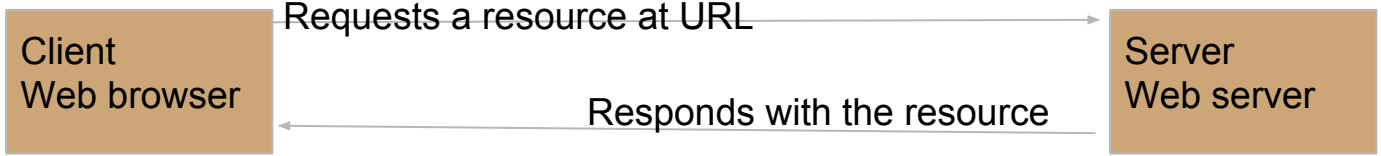
HyperText Transfer Protocol

Contents

- What is it?
- Example session (with screen shots and sh*t)
- Request methods (some of them at least)
- Anatomy of a URL
- A look at a request being sent to a web server
- Example using telnet
- HTTP Status Codes
- Header fields and examples
- What can you use as a web server?

What is it

- HyperText Transfer Protocol
- Application protocol (high up in abstraction stack)
- Used on the world wide web
- Request-Response protocol (for client-server)
- Typical client: A web browser
- Typical server: a web server
- A client typically requests a resource specified in a URL
- The server typically responds with the resource



Example session

Client requests http://localhost:8080/some_file.html

Server responds with a status line: "HTTP/1.1 200 OK" and a message.

The body of the message is the contents of some_file.html

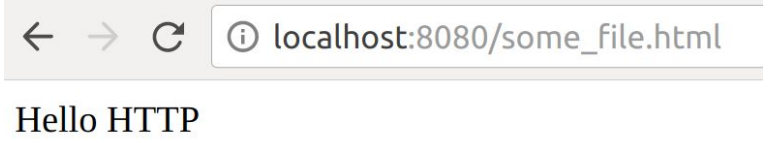
```
<html>
<head><title>Some web page title</title></head>
<body>
<p>Hello HTTP</p>
</body>
</html>
```

Example session - screenshot

Terminal showing communication

```
rikard@dellasoul:~
rikard@dellasoul:~$ GET -s http://localhost:8080/some_file.html
200 OK
<html>
<head><title>Some web page title</title></head>
<body>
<p>Hello HTTP</p>
</body>
</html>
rikard@dellasoul:~$
```

Screenshot of browser



Request methods

A client can send a request using various methods. The three most important are shown here:

- GET Standard request for a resource *side-effect-free*
- HEAD Request only the headers for a request for a resource *side-effect-free*
- POST Send data may *have side-effects*

Anatomy of a URL

scheme:[//[user[:password]@]host[:port]][/path][?query][#fragment]

Example:

http://localhost:8080/get_data?max_price=200&type=vodka

http://	←scheme
localhost	←host
:8080	←port
/get_data	←path
?max_price=200&type=vodka	← query (parameters with key=value pairs)

Testing GET and HEAD in the command line

Install libwww-perl for your unix-like environment (GNU/Linux, Macos, Cygwin/Ubuntu for windows).

Then you should have the commands:

- GET
- HEAD
- POST
- etc...

On Cygwin and MacOS, you'll have to use `lwp-request -m GET` instead

Example run against a web server

```
HEAD http://www.sunet.se          # or: lwp-request -m HEAD http://www.sunet.se
200 OK
Connection: close
Date: Mon, 13 Nov 2017 14:57:50 GMT
Age: 0
Vary: Accept-Encoding
Content-Type: text/html; charset=UTF-8
Client-Date: Mon, 13 Nov 2017 14:57:51 GMT
Client-Peer: 2001:6b0:8:2::233:443
Client-Response-Num: 1
Client-SSL-Cert-Issuer: /C=NL/ST=Noord-Holland/L=Amsterdam/O=TERENA/CN=TERENA SSL CA 3
Client-SSL-Cert-Subject: /C=SE/ST=Stockholm/L=Stockholm/O=SUNET/CN=*.sunet.se
Client-SSL-Cipher: ECDHE-RSA-AES128-SHA256
Client-SSL-Socket-Class: IO::Socket::SSL
X-Cache: MISS
X-Cache-Hits: 0
```

You can also run telnet and connect on e.g. port 80

```
telnet www.sunet.se 80
Trying 2001:6b0:8:2::233...
Connected to webc.sunet.se.
Escape character is '^]'.
HEAD / HTTP/1.0            <--------- This line entered interactively

HTTP/1.1 301 Moved Permanently
Date: Mon, 13 Nov 2017 15:44:12 GMT
Server: Apache/2.4.7 (Ubuntu)
Location: https:///
Connection: close
Content-Type: text/html; charset=iso-8859-1

Connection closed by foreign host.
```

HTTP status codes

The status line (as well as part of the header of a response) contains a status code. Some examples would be:

● 200 OK	standard ok status
● 301 Moved Permanently	the resource has moved
● 400 Bad Request	server can’t understand the request
● 404 Not Found	server can’t find the resource
● 500 Internal Server Error	server ran into some problems

Header fields of the response

After the status line of a header response, these are some example headers with example values (format: `Headerfield: Value`):

- `Date: Tue, 15 Nov 1994 08:12:31 GMT`
  - *obviously, the date of the response!*
- `Server: Apache/2.4.1 (Unix)`
  - *server used for the response*
- `Location: http://www.w3.org/pub/WWW/People.html`
  - *new location when a resource has been moved*
- `Connection: close`
  - *close or keep-alive (http 1.0 uses close, http 1.1 has support for keep-alive)*
- `Content-Type: text/html; charset=iso-8859-1`
  - *type of content of the resource sent back (text, image, json, xml, data, etc...)*

What can you use to play with being a server?

- php -S localhost 8080
- a Java Servlet (running inside a servlet container)
- install a small web server like tinyhttpd
- install a larger web server like apache
- use netcat (see separate lecture for examples)

That's it!

End of short introduction to HTTP.

Links:

- https://en.wikipedia.org/wiki/Hypertext_Transfer_Protocol
- https://www.httpwatch.com/httpgallery/introduction/
- https://www.ntu.edu.sg/home/ehchua/programming/webprogramming/HTTP_Basics.html
- https://code.tutsplus.com/tutorials/http-the-protocol-every-web-developer-must-know-part-1--net-31177