HTTP basics

Programming Web applications

Feliz Gouveia

fribeiro@ufp.edu.pt

Timeline

- 1975: ARPANET
- 1983: TCP/IP is published
- 1990: Internet replaces ARPANET
- 1991: HTTP (Hypertext Transfer Protocol) is invented
  - At CERN (Geneva), from the need to share tons of information between several groups

HTTP: Hyper Text Transfer Protocol

- First defined in 1991 (version 1.1 in 1999)
  - www.ietf.org/rfc/rfc1945.txt
- Client – server protocol
- Uses TCP/IP
  - Port 80
- Basic sequence:
  - Client sends message
  - Server answers
  - Server does not keep state between requests

HTTP

- Basic idea of HTTP is to apply a set of methods to resources
- Resources are identified by a URI (Uniform Resource Identifier)
- A URI can be:
  - URL (Uniform Resource Locator)
  - URN (Uniform Resource Name)
- Most of the time a URI is what we call a web address

URI (Universal Resource Identifier)

- Is of the form:
  - schema : (absoluteURI | relativeURI) ["#" fragment]
- It specifies an address where to find a resource (URL) or a name of a resource (URN)

URN

- Defined in RFC 2141
- A URN does not define how to find a resource
  - We may need a plugin to do that
- A URN specifies a name, for example:
  - urn:isbn:123456789
  - swift:+351-223456789
- List of registered URNs
  - http://www.iana.org/assignments/urn-namespaces/urn-namespaces.xhtml

URL

- Defined in RFC 1738
- Can be absoute or relative (to some base path)
- Example:
  - http://www.ufp.pt
  - http://www.ufp.pt:80
  - http://uniformjs.com/#intro
  - /js/stylesheet.js (relative URL)
  - http://example.pt/pub?**q**=2&**a**=23
    - Parameters **q** and **a**, separated by **&**
    - The part after **?** is the query string

URL

- Some characters are not allowed in the URL, so they must be “URL encoded” (also known as percent-encoded): %hex hex
- Try http://meyerweb.com/eric/tools/dencoder/
- Common cases are spaces (%20), slashes (%2F), question marks (%3F)

HTTP Control parameters

- Version: the protocol version (current 1.1)
- Content coding: coding applied to the resource
  - deflate/zlib
  - gzip
  - compress
- Compression reduces the size of traffic, and some HTTP servers do it automatically for certain types of files
- More next

Media types

- Allow the server to express how the resource should be processed
  - For example a PDF file should be opened in a PDF viewer
- Media types are registered at:
  - http://www.iana.org/assignments/media-types/media-types.xhtml
- An unknown media type gets a fairly general treatment

Character encoding

- Specifies how characters are encoded for transmission
- Initially ISO-8859-1 was most used
  - Single byte encoding, limited set of chars (191)
- We should now (always) specify UTF-8
  - One to four byte encoding
- Source of frequent headaches
  - Some software seems to stick to ISO encoding
  - As streams of chars go through several applications, they can suffer unexpected transformations

HTTP requests

- Request are sent by the client (user agent)
- A method:
  - GET, POST, HEAD
- A resource identified by a URI
- A head
- A body

- A GET is the result of clicking a hyperlink or writting the address on a browser
- A POST is the result of submitting a form

The methods

- GET is expected to be read-only, has no effect on the state of the server
  - Simply: GET is a READ
- POST is expected to change the state of the server
  - Simply: POST is a WRITE
  - Try to resubmit a form; most often the browser will warn that you are resubmiting
- HEAD is a GET but the server only returns headers (no body)
  - Useful to test the modification date of a resource and avoid getting it if not newer than cached version

HTTP responses

- Sent by the server, they specify:
  - A code (see next)
  - A header
  - A body

Response code families (RFC 1945)

- 1xx: Informational - Not used, but reserved for future use
- 2xx: Success - The action was successfully received, understood, and accepted
- 3xx: Redirection - Further action must be taken in order to complete the request
- 4xx: Client Error - The request contains bad syntax or cannot be fulfilled
- 5xx: Server Error - The server failed to fulfill an apparently valid request

Example

- request:
  - GET http://www.w3.org/pub/WWW/info.html
  - Accept: text/html
  - If-Modified-Since: Saturday, 10-July-2010 10:00:00 GMT
  - User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.1; en-GB; rv:1.9.2.10) Gecko/20100914 Firefox/3.6.10 (.NET CLR 3.5.30729)
- response:
  - "200" ; OK

User agent caching

- The user agent (typically a browser) keeps a local copy of resources in cache and avoids downloading them from the server
- Static content (like image files, scripts, stylesheets) is cached
- Dynamic content is fetched from the server each time it is needed
- The server can also specify if content is to be cached or not but the browsers typically ignore

Other optimization

- Manage the number of requests and the size of requests
- Besides caching, optimization includes resource size and rendering, and images
- Good reading at:
  - https://developers.google.com/web/fundamentals/performance/optimizing-content-efficiency/

Recommended tools

- For Firefox
  - *Firebug* (http://getfirebug.com/)
  - *HTML Validator* / Tidy
  - Optionaly *Live HTTP Headers*
- In Chrome and IE9+ use their Developper Tools
- A text editor (Notepad++)
- An IDE (Eclipse, Netbeans,…)

For more information

- W3C: the web standards organization
  - http://www.w3.org
- IETF: The Internet Engineering Task Force
  - http://www.ietf.org/
  - To search for RFC (Request for Comments) use http://www.ietf.org/download/rfc-index.txt
- IANA: for DNS
  - http://www.iana.org/