HTTP - The Hypertext Transfer Protocol

Ciro S. Costa

9 Jul, 2015

Contents

Appendix: The Sleeping Barber Problem (concurrency)

// TODO

Appendix: CGI

CGI stands for *Common Gateway Interface*, which is a standard environment for a webserver to interface with executable programs installed on the server that generates web pages dynamically. Such programs are called *CGI Scripts*.

CGI extends the idea of the HTTP server only knowing about a directy which is designated as a document collection. It allows the owner of the webserver to designate a directory within the document collection as containing executable scripts instead of pre-written page. It is not a language, but a simple protocol that can be used to communicate between Web Forms and your program.

A script can be written in any language that can read from STDI and write to STDOUT (as well as reading environment variables). It's structures as:

1. Read the user's form input
2. Do something with the Data
3. Write the HTML response to STDOUT

HTTP

A browser is basically an HTTP client because it sends requests to an HTTP server which then sends responses back to the client. HTTP server generally listen on port 80 but they can use any port.

URI (**U**niform **R**esource **I**dentifier) is a string of characters used to identify a name of a resource. Schemes specifying a concrete syntax and associated protocols define each URI.

URL (**U**niform **R**esource **L**ocator) is a reference to a *resource* that specifies the location of the resource on a computer network and a mechanism for retrieving it. It is an *URI* that, in addition to identifying a web resource (also ftp, email, database acces, etc), specifies the means of acting upon or obtaining the representation, specifying both access mechanism and network location. *scheme* defines how to connect (as well as the namespace, purpose and the syntax of the remaining part of the URL), *host* defines where to connect and the remainder specifies what to ask for. It's defined by:

`scheme://[user:password@]domain:port/path?query_string#frag_id`

Note that domain name is not case sensitive since DNS ignores case. When port is ommited the browser will connect to the default (80 for HTTP, 443 for HTTPs - remember that [0-1024] ports are restricted).

The process that the client and server goes through in the client-server model is:

1. HTTP client opens a connection and sends a request message to an HTTP server
2. The server returns a response message, usually containing the resource that was requested.
3. After delievering the response, the server closes the connection (making the HTTP a stateless protocol)

The request is well-specified as you might imagine. Even newlines have to agree on a standard: they are all CRLF (carriage return, line feed - `\r\n`).

```
<initial line, differente for request vs response>   // init line (end w/ CRLF)
Header1: value1   // headers ( each ends w/ CRLF)
...
```

```
HeaderN: valueN
                  // blank line (CRLF)
<optional message body>   // self-described
```

Initial Line

When performing a request: `method_name local_path http_version`. Example:

```
GET /path/to/file/index.html HTTP/1.0
```

When performing a response: `http_version status_code code_description`. Example:

```
HTTP/1.0 200 OK
```

Methods names and Status code are well defined (as well as those human-readable status messages). The first digit identifier the general category of the response:

1xx an information message only

2xx success of some kind

3xx redirection to another URL

4xx error on the client's side

5xx error on the server's side

Headers