INF-2301: Computer communication and security

Web Servers and REST

Camilla Stormoen

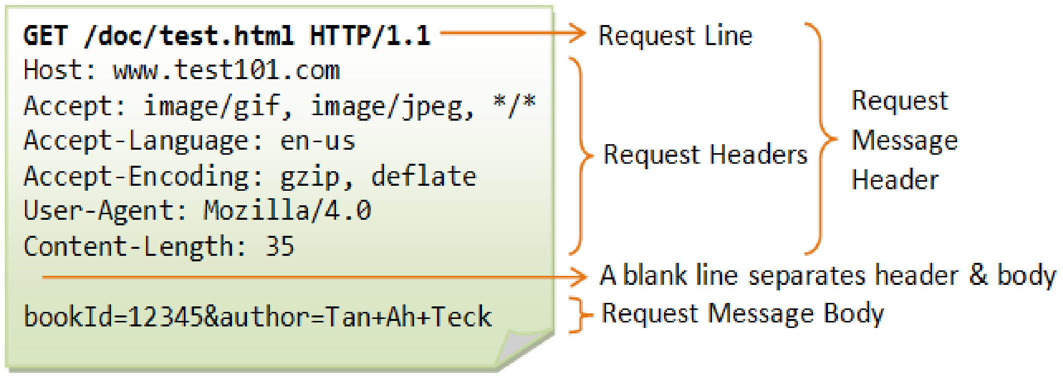
12.09.2014

Introduction

This assignment is about the Hyper Text Transport Protocol (HTTP)[1] and web servers. We shall also use HTTP to implement a RESTful[2] application. The assigment is in two parts, in the first we're going to implement a simple web server and in the second one, we're going to omplement a RESTful application message server.

Tecnical backround

HTTP request[3]

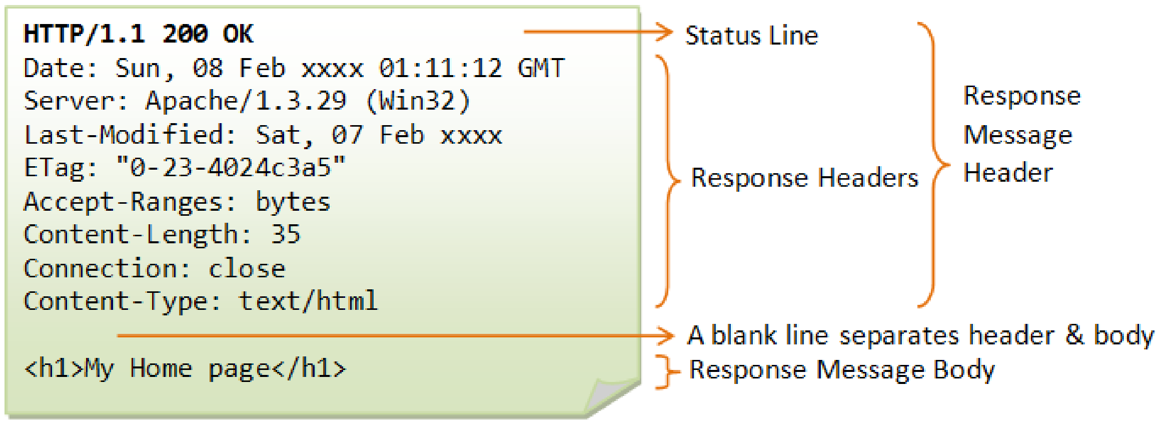


```
GET /doc/test.html HTTP/1.1
Host: www.test101.com
Accept: image/gif, image/jpeg, */*
Accept-Language: en-us
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0
Content-Length: 35

bookId=12345&author=Tan+Ah+Teck
```

Here you have an example on GET and how you can reach the requestline. This is an example on a GET request.

HTTP response



```
HTTP/1.1 200 OK
Date: Sun, 08 Feb xxxx 01:11:12 GMT
Server: Apache/1.3.29 (Win32)
Last-Modified: Sat, 07 Feb xxxx
ETag: "0-23-4024c3a5"
Accept-Ranges: bytes
Content-Length: 35
Connection: close
Content-Type: text/html

<h1>My Home page</h1>
```

The respons message is what we would send back to the client. The RFC doesn't say that the respons header must be with the respons, but the content-length must be included as a header in the response message. If the content-length is not present, the client know when to stop receiving.

1 http://en.wikipedia.org/wiki/Hypertext_Transfer_Protocol
2 http://en.wikipedia.org/wiki/Representational_state_transfer
3 http://www.ntu.edu.sg/home/ehchua/programming/webprogramming/images/HTTP_RequestMessageExample.png

Requirements

The serves should run on localhost:8080. The HTTP server should implement the following request/respond codes:
- GET
-POST
-"200 OK"
-"404 NOT FOUND"

The RESTful application should support:
-GET
-POST
-POST
-DELETE

Technical background

The HTTP protocol is a request-respons protocol. The webbrowser requesting an html page from the server. The protocol is specified in a series of publications called Request for Comments[4] (RFC) and the first thing that happens is that the server gets a request in the format of a Request Header.

Request/Respons Header

The request header defines the type of request (method, protocol and the file that is requested). The header can also consist of additional such as content-length, -type and -size etc. Every field is seperated with the EOL characters *\r\n*.
There are also a field which is the payload which is in the bottom of the header. The payload is prependent with additional *\r\n*.

Status-line

The first line defines the protocol, method and the requested URL, for the request and the protocol and status-code/message for the request. The most commons status-codes are "200 OK" and "404 NOT FOUND".

Design

The servers are made quite simple. The requirements are obtained and the the servers are ran on the simple error-checking precode which was delivered with the assigment.

When the webserver is starting, the sockets opens and listen for an incomming connection, as soon as one is made the appropiate action is taken and response is given. The action depends on the type of server. If the user types the wrong URL adress, the user gets to an error-site where the "404 NOT FOUND" will show up.

4 http://tools.ietf.org/html/rfc7230

Implementation
Both servers are made likely identical except for the supported commands, which should do different things depending on what the use is. The programming is done in python and a class is represented in the server.

HTTP server
When the HTTP server recieves a GET, there are 3 cases to consider. The first one is if the URL begins with a “/”, then the URL will automatically change the URL to the index-URL. It the user types the right URL-address, they will come to either index.html or test.xml depending on the implementations. If the client types the wrong URL-address, the users will recieve a “404 NOT FOUND” error and they have to try again.

RESTful application

The GET request in the RESTful application poens the xml file and returns all data.

The POST request itereates through all messages for an emty entry. Once done, the message is given a unique id that is higher than the previous.

The PUT is very similar to POST, the request with the given id and change the message.

The DELETE request should delete the message with the given id field.

Discussion
TCP over UDP
The connection is of type TCP-connection due to the framework around it and the safety when using it, one can abstract away the problem of losing and not receiving all the data to the client.

Why to use socket – import socket
Sockets are the network communication done by your computer. When you type in an URL-address, the socket opens and connects to the Url-address given and show it in your browser. Any network communication goes through a socket.

Improvements
Some improvments for the implementation could be more checking for error.

Conclusion
HTTP is really simple and ok when you get used to it. To read the RFC is a bit thougth, as it written in a non-friendly format. When the concept of HTTP is understood, there's not a lot of knowledge needed to operate the basic commands.