Assignment 2: HTTP Client and HTTP Server using Java Sockets

The goal of this assignment is to learn Socket programming and get familiarized with the basics of distributed programming. In addition, this assignment also helps to understand the Hypertext Transfer Protocol (HTTP), which is one of the widely used protocols on the Internet.

Assignment Description

This assignment consists of two parts. In the first part, you will implement a HTTP client to retrieve web pages from a HTTP server. And, in the second part of the assignment, you will build a HTTP server to host a web page.

HTTP Client

In the first part, you will build a HTTP client that communicates with a HTTP server to serve web pages for end users. The HTTP client program should support HTTP version 1.1. For your assignment, you should implement a HTTP client, which accepts the following arguments in the command line as follows:

CommandLine$ HTTPClient HTTPCommand URI Port

In the above line, the HTTPClient denotes the name of your java executable. The HTTPCommand refers to HEAD, GET, PUT or POST. You can use any URI's such as http://www.example.com or http://www.google.com to test your client implementation. And, the default port number of 80 should be used to connect with the HTTPserver.

After each request, your client should display the response received from the HTTP server on the terminal and also store the response in an html file locally. When you retrieve a webpage from a HTTP server, you should scan the HTML file and check for embedded objects such as images. If you find an embedded object in the HTML file, you should use the GET command to retrieve those objects as well. In order to reduce the complexity, we do not expect you to retrieve all types of embedded objects. During the demonstration, you should at least retrieve image files from a HTTP server. The retrieved images should be stored locally.

For PUT and POST commands, your client should read a string from an interactive command prompt. These two commands will be tested with your HTTP server program.

HTTP Server

The HTTP server program should host a simple web page on your local machine. This server program should be multi-threaded to support multiple clients at the same time. The server should support the following operations: HEAD, GET, PUT and POST.

During the demonstration, a HTTP Client will retrieve the web page hosted on your

server. The HTTP Client can either be your HTTP Client program or any 3rd party clients such as Firefox or Chrome. If you have followed the protocol standards correctly, any HTTP Client will be able to interact with your server.

In addition, we expect your server to host the web page that is retrieved using your HTTP Client program. If your HTTP Client program retrieved the web page and the associated embedded objects correctly, then any 3rd party clients such as Firefox or Chrome should be able to render it.

In the case of PUT and POST commands, your server should store the data received from clients in a text file.

Your server should use persistent connections and support if-modified-since HTTP header. In addition, note that the host header field is mandatory for HTTP version 1.1.

You should at least support the following status codes on your server:

- 200 OK
- 404 Not Found
- 500 Server Error
- 304 Not Modified

Along with the status codes, the server should send the date, content type and content length headers to the client.

General Guidelines

- You are not allowed to use HTTPURLConnection package for this assignment. You should only use the sockets package to complete the assignment.
- You should document your code. During the evaluation, we will go through your code and may ask you to explain how a certain method works in your code.
- You should know the difference between HTTP 1.0 and HTTP 1.1.
- Telnet can be used to test and understand the HTTP commands. You can use telnet as a debugging tool during your implementation.

Practical Information

- You should either work alone or in groups of two. You should email your group details i.e. names and student numbers to gowrisankar.ramachandran@cs.kuleuven.be on or before 12-03-2015. The subject of this email should be **CN:HTTP**.
- You are strongly encouraged to use the computers in the lab.
- You have 2 weeks to complete the assignment. In the 3rd week, you should demonstrate your assignment to the teaching assistants. You will be marked based on your performance in the demonstration.
- For students working in groups, both the students should be prepared to demonstrate the assignment. If you cannot explain your code, we will assume that you did not write it.

- The third session is only meant for marking your assignment. Therefore, you should be ready to demonstrate your code at the start of the 3rd practical session. You will only be marked in your assigned session.

Marking Specifications

The following specifications will be used during the marking session.

HTTP Client Marking (worth 3 out of 8 marks)

Mark	Expected Functionality
(D) Below 1	Client is not functional or sufficiently demonstrated.
(C) 1 to 1.5	Client only works with a limited set of web pages.
(B) 1.5 to 2	Client works with a wide variety of web pages.
(A) 2.5	Client works correctly with all web pages
(A+) 3	As above with documented code and elegant design.

HTTP Server Marking (worth 5 out of 8 marks)

Mark	Expected Functionality
(D) Below 1.5	Server is not functional or sufficiently demonstrated.
(C) 1.5 to 3	Server handles only one client.
(B) 3 to 4	Server supports HTTP 1.1, but has threading problems or poor support for status codes.
(A) 4.5	Server successfully serves the web page retrieved using your HTTP client program.
(A+) 5	As above with documented code and elegant design.

References:

1. HTTP made really easy [Strongly recommended].
   Link: http://www.jmarshall.com/easy/http/
2. The HTTP Specification: HTTP 1.0 (RFC 1945).
3. The HTTP Specification: HTTP 1.1 (RFC 2616).
4. The definition of embedded objects or MIME types (RFC 1521).