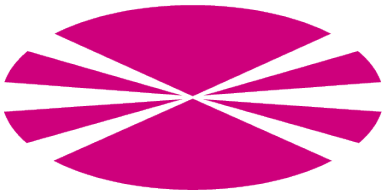

UNIVERSIDADE DA CORUÑA

COMPUTER NETWORKS LAB

Assignment 1: Java Web Server

Java Web Server

In this assignment you will develop a basic Java web server capable of interacting with a web browser to navigate through a web site.

Requirements

The requirements for the web server are:

- To use HTTP protocol version 1.0, which implies that independent HTTP requests will be generated for each web object (i.e. html files, images, etc.).
- The server must be multithread to be able to process multiple requests in parallel.
- To process the GET and HEAD methods of an HTTP request.
- To correctly process the If-Modified-Since in a request, if received.

The web server will receive and process an HTTP request and prepare and send an HTTP response. Regarding the status line of the HTTP response, the web server must implement the following codes:

- 200 OK
- 304 Not Modified: as a reply to an If-Modified-Since request, when required.
- 400 Bad Request: the HTTP request received follows a format or uses a method that is not understood by the server.
- 404 Not Found: the resource requested does not exist in the server.

Regarding the header lines of the HTTP response, the web server must, at least, implement the following:

- **Date**: date and time when the response was created and sent.
- **Server**: specifies the web server name that has attended the request. The student will select the server name according to his preferences.
- **Content-Length**: indicates the number of bytes of the resource sent.
- **Content-Type**: specifies the type of resource contained in the entity body. The following types must be implemented:
  - text/html: indicates that the response is in HTML format.
  - text/plain: indicates that the answer is in plain text.
  - image/gif: indicates that it is an image in gif format.

- image/png: indicates that it is an image in png format.
- application/octet-stream: used when the file format is not identified.

- **Last-Modified**: indicates the date and time when the resource was modified for the last time.

In order to correctly process a GET request with the If-Modified-Since header option, if the resource requested was last modified after the date and time specified by the option, then the resource must be sent normally (status code 200 OK). Otherwise, the server must reply a status code 304 Not modified in the HTTP header and no resource is sent.

Development

Students will use the project **https://github.com/GEI-Red-614G010172223/java-labs-<user-login>** as base code. In particular, files contained in the **es.udc.redes.webserver** package must be modified. If a student has not created the project, the instructions in https://github.com/GEI-Red-614G010172223/java-labs/blob/master/README.md should be followed.

The project includes the directory **p1-files** which contains utility files for the development of the web server, such as sample files for the formats required: index.html, LICENSE.txt, fic.png or udc.gif. We strongly recommend to use this directory as base directory to retrieve all resources requested, that is, the paths in the request will be relative to this folder. As for the *working directory* configured for the web server in IntelliJ, it must maintain its default value (java-labs-<user-login> directory).

The web server should be executed indicating the listening port number:

java es.udc.redes.webserver.WebServer <port>

We recommend following a two-step approach to implement the web server. In a first stage, using the TCP echo multithread server, the server will just display the HTTP request received. This can be tested simply by starting your server in a non-reserved port (e.g. port 5000) and use your browser to request the URL http://localhost:5000/index.html.

In the second step, the server should be able to generate the appropriate HTTP response to the request along with the file requested. If any error occurs (e.g. the file cannot be found), the server should reply with the corresponding HTTP code and a HTML page for this type of error (sample HTML error pages are provided in the p1-files directory).

Testing

While developing your web server, in the initial steps, we recommend using the **"nc" command** to test the execution of your server.

Through the following steps you can test different aspects of your web server:

1. Use the 'nc' command to connect to the server and leave it open (this is to prove that the server is multithread).
2. Open a browser and load a HTML page containing plain text, gif and png images.
3. If the page is displayed => Multithread OK, GET OK, basic formats OK
4. Return to the terminal running the 'nc' command, and send a HEAD request (e.g. HEAD index.html HTTP/1.0).
5. Verify that it returns the correct status line and the header lines => HEAD OK.
6. Verify that the following headers are sent correctly: Date, Server, Content-Type, Content-Length and Last-Modified => HTTP header parameters OK.
7. Start another "nc" command to connect to the server and send an incorrect request (e.g. GETO index.html HTTP/1.0): check that the server returns a 400 Bad Request error + an error page (with the correct header lines) => Bad Request OK
8. Request a resource that does not exist (e.g. GET indeeex.html HTTP/1.0): check the server returns a 404 Not Found error + an error page (with the correct header lines) => Not Found OK.
9. Using your browser, reload a HTML page (for example, http://localhost:5000/index.html) and check that the response code is 304 Not Modified, using the Developer tools in the browser.

We provide a Java application (p1-files/httptester.jar) to verify that your web server works correctly:

java -jar httptester.jar <host> <port> [<0-9>]

During the evaluation, httptester.jar tests will be used to verify that your web server works properly. Therefore, we strongly recommend testing your code with this tool before its submission.

Evaluation

The implemented classes will be uploaded to the student github repository: **https://github.com/GEI-Red-614G010172223/java-labs-<user-login>**.

Once the last changes have been done, the student must tag the last commit in order to identify the version to be evaluated and then the tag must be published. The git commands to use are:

1. **git tag -a p1 -m "p1"**
2. **git push origin p1**

The submission of this assignment will be done using Github. The deadline to upload the code is March 17, at 23:59. Commits after this date will not be considered. **It is mandatory to complete the "Repositorio de Github / Github repository" quiz in Moodle before deadline in order for the submission to be evaluated.**

If the code is not delivered before the deadline, the assignment will be considered as not presented. It will be also considered as not presented if a tag named as p1 does not exist in the repository or if the Moodle quiz is not filled properly.

The sockets programming in Java has a weight of up to 1.25 points in the final grade of this subject. The qualification will be computed summing the following aspects:

1. Grade for httptester.jar script (0.5 points maximum). The teacher will evaluate p1 using this script. The grade obtained will be weighted up to 0,5, if, and only if, the student reaches 5 points (out of 10) in the first practical exam (0, otherwise).
2. First practical exam (0.75 points maximum). There will be a written examination where p0 and p1 content will be evaluated. This will take place in the corresponding theory group. Any group change must be requested and justified prior to the exam date. In order to take this exam, the student must have delivered the p1 assignment. The grade (out of 10) will be weighted up to a maximum of 0.75 (whether or not the threshold of 5 points is reached). **The exam date will be March 23.**