Extending the simple web server

Davide Lissoni Mat.179878
23/09/2016

1.Introduction:

This assignment consist in an implementation of a web server used to get http request and response The server in particular has to be an extension of the “simple http server” (latemar.science.unitn.it/segue_userFiles/2015WebArchitectures/src.zip) which has the possibility to handle dynamic web pages as well as static web pages.

Static web pages are pages where the user can only view the content provided by the author and also, the content of the page does not change.
Dynamic pages instead, are web page whose content, at least in part, is generated by the server at the moment. Therefore the content could change each time the page is invoked.

In our particular case the server, once taken the get request, has to parse it and then decide if the request resource consist in a static or dynamic page and response according to this.
In the case where the request ask for a static page, then the response will consist in the simple static page requested, if any.
Whether the page requested is dynamic, the server will executes the external process indicated in the request,if any, catching and printing its output, which has to be in html format. Furthermore parameters can be passed to the process and in this case they have to be present in the request.

2.Implementation:

Since the “simple http server” already provide methods allow to listen on a specified port and service simple HTTP "get file" requests trying (once got the request) to open the file requested, the extension implemented has to manage the case where the request resource is a dynamic page.
In order to doing that the http server has been modified by adding a parse on the http request in order to understand what the server has to do and, a method used to run an external process and visualize its output.

-check the GET http request:

For the parse method it has been managed the request as a StringTokenizer in order to break the string in different token simplify in this way the controls on the string. The system at the beginning control if the GET request is well formed and, only in that case, check if the string of the request starts with the string “process/”.
If yes, the requested page is dynamic, than the system will take the name of the external process to run and its parameters from the request by using the Java String split() method in this way:

```
String splitted[]=request.split("\\?");
```

Where the name of the process will be in splitted[0] while the parameters will be present in splitted[1]. The splitted[1] substring it will be in turn splitted in the
`“public String[] valueparams(String query,OutputStream out)” function` in order to get their values . For example if the request is :

GET /process/add?op1=22&op2=54 HTTP/1.1

the name of the process will be “add” while the parameters values will be “1” and “2”.

If no, the system will open the file present in the request if and only if the file is present in the project folder.
In the case where the request is empty (`GET / HTTP/1.1`) the system will open the file index.html already present in the project folder.

-run the external process

In order to run the external process and get its output has been follow the online guide http://www.rgagnon.com/javadetails/java-0014.html. For this project I chose to support exe (extension) processes and the parameters are passed as arguments of the exe process. The operating system used to test the program is ubuntu 15.10 then the command line used to run the process is like:

```
./processname.exe param1 param2.
```

3.Deployment:

In order to run the server using an integrated development environment, just import the project and run it, specifying, optionally the port number as argument(I chose as default port number 8000).

The server can also be started from command line by moving in the project folder and type the command:

java -jar tinyhttpd.jar argument.

In order to visualize a static page just open the browser and digit the follow url:
localhost:<port-number-choosen>/file-name
Note that the file must be present on the project folder. In order to put a file in the project folder just copy it inside the folder.

In order to run a process and than visualize its result in a dynamic page, open the browser and digit the url below:

l*ocalhost:<port-number-choosen>/process/<process-name>?<param1>=value&<param2>=value*

Note that the project can run every exe external process present in the folder “process” located in the project-folder, but, for request-parsing issues, two parameters must be always present in the http request even if the external process don't ask any parameter as input. The process will run anyway without problem and the parameters, when not needed, will not take into consideration.

4.Comments and notes:

The method DataInputStream readLine() present in the “simple http server” is deprecated giving some socket connection problems. In any case the slides reccomended by the professor are correct and the solution is to change the type from DataInputStream to BufferReader.