Internet Engineering HW1 (HTTP)

Shahid Beheshti University, Class of Fall 2020.

Homework Description:

You have studied throughout this course what is http protocol, but one question may come in mind, how the http protocol works in the real world? In this homework you will achieve this knowledge by writing a simple **http version 1.0 client** in java via **socket programming**.

Design Specification:

Your program must be written in **java**, and you **cannot** import any third party library for this specific homework.
By running the program, either the .jar file or the main class of the source code, a command-line application will get up and running and will wait for receiving an http address. (you do not need to handle https, only http will suffice) Or a `command` (which will be discussed later what are the commands)

> The http address schema in regex is as follows:
>
> ```
> (?i)(http://)?([-a-zA-Z0-9@:%._\+~#=]{1,256}\.[a-zA-Z0-9()]{1,6}\b)([-a-zA-Z0-9()@:%_\+.~#?&//=]*)
> ```
>
> A bigger version for better readability:
>
> ```
> (?i)(http://)?([-a-zA-Z0-9@:%._\+~#=]{1,256}\.[a-zA-Z0-9()]
> {1,6}\b)([-a-zA-Z0-9()@:%_\+.~#?&//=]*)
> ```

Some examples will match against this regex are:
http://google.com
http://google.com/mamad.html
http://www.google.com/mamad/abbas/reza.html
google.com/reza.html
http://jsonplaceholder.typicode.com/todos
jsonplaceholder.typicode.com/todos
192.168.1.1
http://192.168.1.1
http://192.168.1.1/mamad.html

You can use This link to check your result against the given regex, or try it in the Java language itself or use IDE tools to check it.

(It is worth mentioning that the important point in this part of the homework is that your program should be able to separate between the server name and requesting resources on the http address in the http protocol. You are free to handle the http url however you want, but if you would like to use regex, this can be a very good regex because not only it will match against all http addresses, but also it will group the server name and resource quite nicely.)

(It is also worth mentioning that the input will try to be matched to your program, so don't worry too much about this part :), but try to do it correctly and don't deviate from it too much or you may lose some scores).

After the program has received the address,
If the address provided was valid the program will get input for http method:
`GET` for get method,
`POST` for post method,
`PUT` for put method,
`PATCH` for patch method,
`DELETE` for delete method,

the program will make an http request to the corresponding server, and will ask the server for the correct resource under the http protocol.

- If the response of the server suggested that the requested resource was an **HTML document,** the program will save the content (i.e. response body) as a file on disk with a proper name and .html extension.
- If the server responds that the requested resource is **plain text**, the CLI will simply show the response body.
- If the server responds that the requested resource is in **JSON format**, the content (i.e. response body) will also be saved as a file on disk with a proper name and .json extension.
- Also if the server responds via different http **status codes rather than 200**, like **3XX, 4XX, 5XX, 204, 201**, etc., it is expected that such cases will have feedback to the user. (something like, for example if the status code was 404, the client prints “404, not found” or something like that, the important part is handling the status codes and showing that the program can in fact understand http status codes, and can understand the difference between error codes (4XX, 5XX) vs success codes (200))

- You are free to do whatever you want with other cases.
- Handling the download of other resources which are linked to the HTML document page (CSS stylesheets, JS scripts, Images, fonts, NOT other html hrefs (which if this gets tried to be done, it will probably enter an infinite loop), etc) will result into 20% additional points.
- Handling the download of BLOBs (Binary Large OBjects, e.g. mp4 files, mp3 files, etc.) and downloading it via multiple http streams (like how a download manager works), will result in 10% additional points.

The program should be able to receive multiple addresses during a run, and should act accordingly based on the http address provided by the user,and the server response.

Commands:

- `exit`: After the user is finished, typing `exit` in the command line should terminate the program.

- `set-student-id-header`: after running this command, the CLI waits for an input from the user, which is the student id, and then sets this value as **`x-student-id`** header of every subsequent request.
- `remove-student-id-header`: if student id was not set, this will do nothing, otherwise it removes the header from every subsequent request.

Implementation Specification:

Here's are some tips that may help you during implementation of this homework:
1- The HTTP protocol is an application protocol which rests on the foundation of TCP protocol. You can communicate to any computer (official term is host), via TCP protocol if you know the IP and port of such host. The standard http port for any TCP communication is port 80.
2- Java exposes a comprehensive and easy to use api for programmes who intend to do network TCP programming, which is called Socket programming in Java, which you have worked on, in your AP class.
3-As you may already know, the HTTP request and response, both have 2 sections; Header and the Body. The header of the request is where you would specify what exactly you would want from the server. And in the response header the server would tell you meta information that you need to know.
3.1- You can send or get any kind of Key-value pairs which are comma separated in the header, as Request or response Header, but there are some de-facto standards which everyone uses to inform each other about some important meta data. One of which **`Content-Type`.** Which usually determines the type of the content which server is sending to the client. Usually the value of this field is a standard MIME type You can see the full list of standard values here, and also the list of more commonly used mime types here. But for this assignment you need only these three MIME types: (If you are planning to implement additional features for additional scores, It would be a good idea to check common mime types)

1. For HTML the standard mime type (The value which will be send on Content-Type header) is: `text/html; charset=<CHARSET>`
2. For plain text the standard mime type is: `text/plain; charset=<CHARSET>` (or not sending the Content-Type header)
3. For JSON the standard mime type is: `application/json; charset=<CHARSET>`
   Where the `<CHARSET>` is the character set of the response body (utf-8 for UTF-8, or etc.) (it's okay not to care about the charset part in this homework, if you are using java and socket. Because if you feed the input stream of java socket object into a Scanner, the .nextLine() method automagically translates the inputStream bytes into utf-8.)
   3.2- The other header which can help you to understand when the data transmission is ended from the server is to look at the **`Content-Length`** header**.** Which usually holds the amount of character (or bytes if the request body is a blob) that you should expect to receive in the request body.
   (it's okay not to care about the charset part in this homework, if you are using HTTP/1.0. Because the server will push an EOF at the end of the input stream). (but if you are planning to implement higher versions of http protocol, keep this header in mind.)

4. If nothing comes after the hostname (the first part of the url) the http client must request "/" resource.
5. It is worth mentioning that the line separators should be CRLF ("\r\n") **NOT** a single Line Feed ("\n")

Testing:

You can test your code, by running http request to the following server ip: 51.89.222.172
There are some routes you can test your client on. (The following server will only be available for testing only for a short period of time (probably until the deadlines))
You must set x-student-id header with your student id value in order to be authorized otherwise you will get a response with 401 status code.

- **`METHOD GET on "/"`** : it will return a text to show if your client works successfully or not
- **`METHOD GET on "/test.html"`**: it will return a sample html page which your client should save.
- **`METHOD GET on "/sample-json"`**: it will return a sample json format text which your client should save.
- **`METHOD POST on "/sample-json"`**: it will return another sample json formatted text.
- **`METHOD GET on "/sample-error1"`**: it will return a response with 404 status.
- **`METHOD GET on "/sample-error1-with-body"`**: it will return a 404 status with some text to show.
- **`METHOD GET on "/sample-error2"`**: it will return a 403 status
- **`METHOD GET on "/sample-error3"`**: it will return a 500 status.

- (Information for the students): If you call "/status" resource via GET method, you would receive a text which tells the statuses of requests that you have done to the server. The data which the server sends is a **plain text** and not json. But some part it is formatted to look like a json, you could review the response which can help you to see which requests you should fulfil to pass all the tests, and also understanding the structure of the json (in this response, the curly braces "{...}" and the content inside of it is considered to be a valid JSON format) can help boost your knowledge for future lessons of this course.

Best of luck.
Teaching Assistant team.