Web Architectures

assignment 1

Dynamic website without using Web Languages

Sebastiano Chiari - 220527

sebastiano.chiari@studenti.unitn.it

October 3, 2021

1 Part 1

1.1 Introduction

The aim of the first part is to implement a simple web server in Java that creates dynamic pages without using web languages or interfaces. This is achieved by modify the provided simple web server in order to launch an external process whenever is submitted a request for an URL that start with the token `/process`.

More, an URL like `http://localhost:8000/process/reverse?par1=ROMA` should activate an external process that takes the given string (passed as a parameter) and should return a dynamic web page with the reversed string (e.g. AMOR displayed as output).

1.2 Implementation

The starting point is the provided simple web server Java project. This server provides a welcome page given a default path or an empty request; more, given a specific path, the server tries to retrieve the requested resource in the Document folder of the project, replying accordingly (the resource if it is present or a `404` error).

I extended the `MinyHTTPD` class, by adding some controls in order to catch if the URL contains the `/process` token. This was achieved thanks to an intense use of the `String-Tokenizer` class, in order to chop the request string into pieces according to the previous knowledge of how the URL would be look like.

Once the `/process` token is recognized, the server checks if the URL contains also the `/reverse` token and the necessary parameter. If so, an external process (the `Reverse` Java class) is executed through the `ProcessBuilder` class and the output is saved. Then, a dynamic web page is built and the user will receive a web page with the provided input string and the reversed output.

1.3 Results

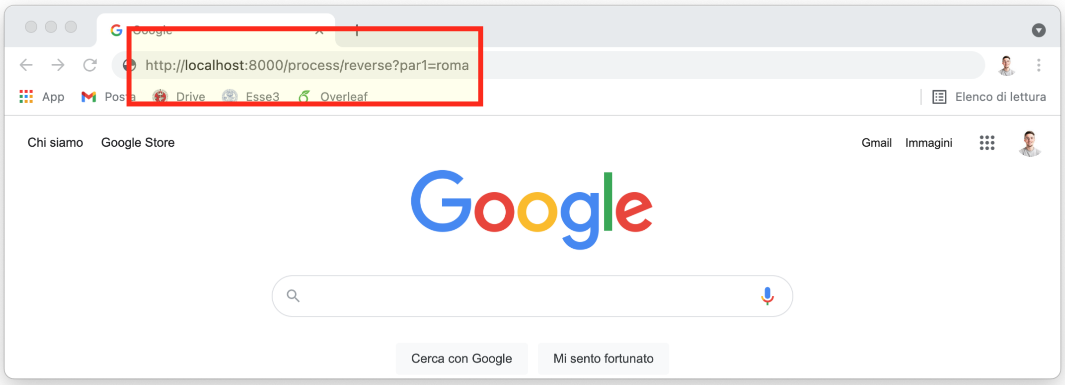
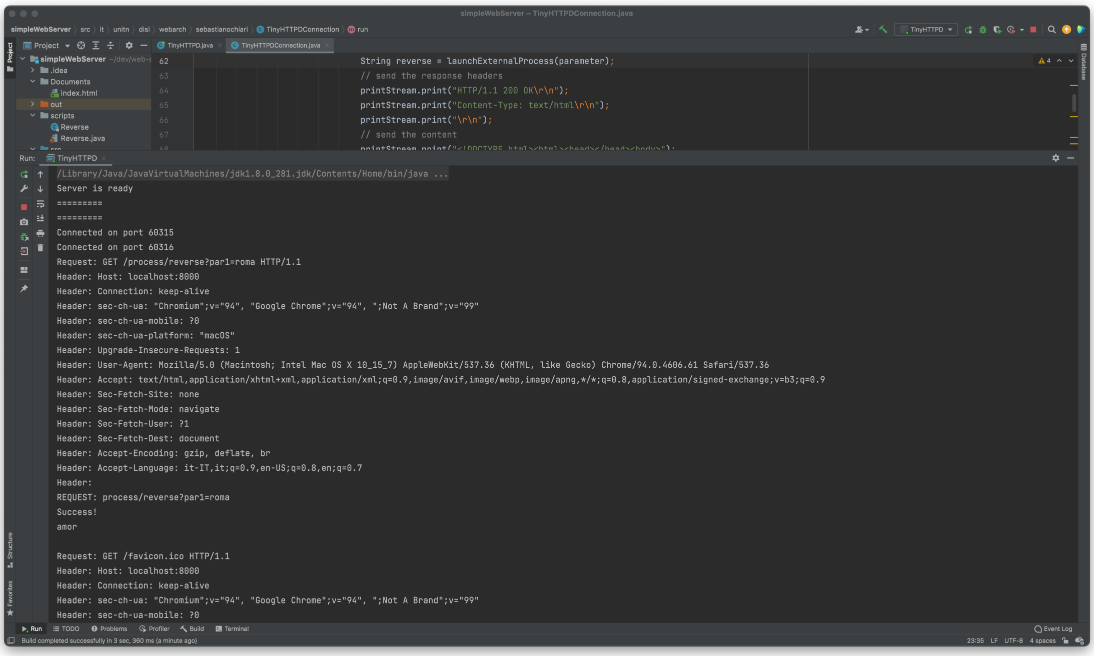


Figure 1: Requesting the external process to the simple web server



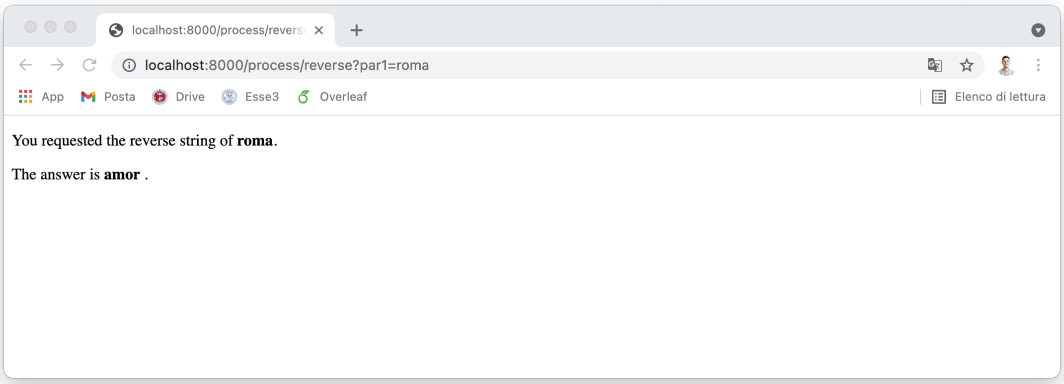
```
String reverse = launchExternalProcess(parameter);
// send the response headers
printStream.print("HTTP/1.1 200 OK\r\n");
printStream.print("Content-Type: text/html\r\n");
printStream.print("\r\n");
// send the content
```

```
/Library/Java/JavaVirtualMachines/jdk1.8.0_281.jdk/Contents/Home/bin/java ...
Server is ready
=========
=========
Connected on port 60315
Connected on port 60316
Request: GET /process/reverse?par1=roma HTTP/1.1
Header: Host: localhost:8000
Header: Connection: keep-alive
Header: sec-ch-ua: "Chromium";v="94", "Google Chrome";v="94", ";Not A Brand";v="99"
Header: sec-ch-ua-mobile: ?0
Header: sec-ch-ua-platform: "macOS"
Header: Upgrade-Insecure-Requests: 1
Header: User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/94.0.4606.61 Safari/537.36
Header: Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
Header: Sec-Fetch-Site: none
Header: Sec-Fetch-Mode: navigate
Header: Sec-Fetch-User: ?1
Header: Sec-Fetch-Dest: document
Header: Accept-Encoding: gzip, deflate, br
Header: Accept-Language: it-IT,it;q=0.9,en-US;q=0.8,en;q=0.7
Header:
REQUEST: process/reverse?par1=roma
Success!
amor

Request: GET /favicon.ico HTTP/1.1
Header: Host: localhost:8000
Header: Connection: keep-alive
Header: sec-ch-ua: "Chromium";v="94", "Google Chrome";v="94", ";Not A Brand";v="99"
Header: sec-ch-ua-mobile: ?0
```

simpleWebServer – TinyHTTPDConnection.java
simpleWebServer › src › it › unitn › disi › webarch › sebastianochiari › TinyHTTPDConnection › run
TinyHTTPD.java
TinyHTTPDConnection.java
Run: TinyHTTPD
Run
TODO
Problems
Profiler
Build
Terminal
Build completed successfully in 3 sec, 360 ms (a minute ago)

Figure 2: Server logs with the request headers



You requested the reverse string of **roma**.

The answer is **amor** .

Figure 3: The response of the server with a dynamic web page

1.4 Comments and notes

Given the large amount of `StringTokenizer` instances used, the URL parsing section could be very inefficient. More, the whole structure of the parsing is the weakest point of the web server: this is because the parsing itself is built over the previous knowledge of how the URLs would look like and no other cases or possible user's misuses are handled.

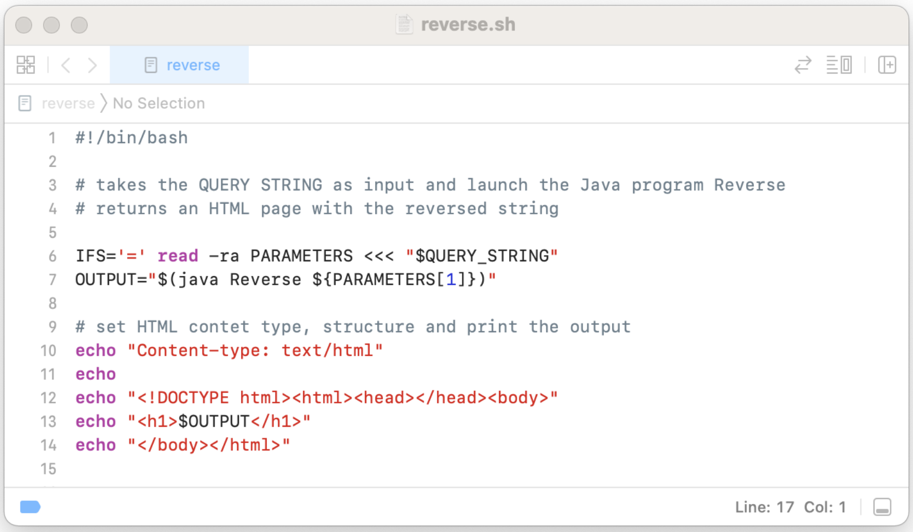
2 Part 2

2.1 Introduction

The aim of the second part is to implement a bash script for an Apache Web Server, that launches the reverse process built in part 1.

2.2 Implementation

I implemented the following `reverse.sh` script inside the `cgi-bin` directory of the web server. First, the script parses the `QUERY_STRING` object and retrieves the parameter. Then, the Java class `Reverse` (placed inside the `cgi-bin` folder too) is executed and the output is stored inside the `OUTPUT` variable. The script sets the content type and the basic structure of an HTML page and it prints the reversed string.



```
#!/bin/bash

# takes the QUERY STRING as input and launch the Java program Reverse
# returns an HTML page with the reversed string

IFS='=' read -ra PARAMETERS <<< "$QUERY_STRING"
OUTPUT="$(java Reverse ${PARAMETERS[1]})"

# set HTML contet type, structure and print the output
echo "Content-type: text/html"
echo
echo "<!DOCTYPE html><html><head></head><body>"
echo "<h1>$OUTPUT</h1>"
echo "</body></html>"

```

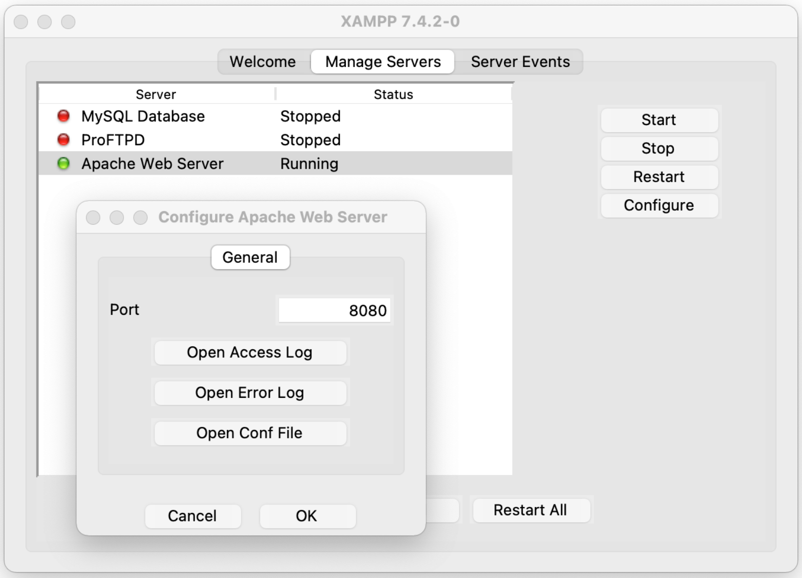
2.3 Results

I started the Apache Web server on the port 8080 (the default one) with the manager tool of XAMPP (Figure 4).

I requested the resource (`reverse.sh`) from the `cgi-bin` folder and I gave the needed parameter (Figure 5).

Previously, I had to change the permissions of the executable shell script in order to allow Apache to use it (Figure 6). Otherwise, Apache would have returned a `500`-type error to the client, with the following error message: `End of script output before headers: reverse.sh`.

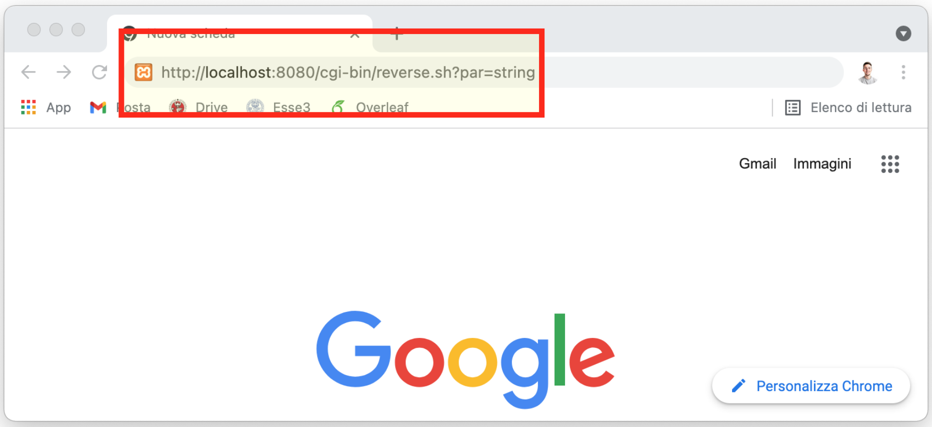
Then, the server returns the reversed string correctly (Figure 7).



Server	Status
MySQL Database	Stopped
ProFTPD	Stopped
Apache Web Server	Running

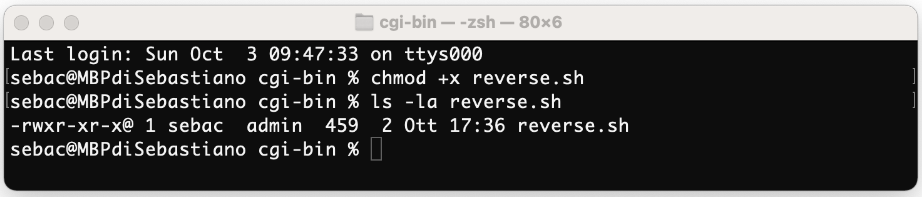
Port 8080

Figure 4: Apache web server configuration



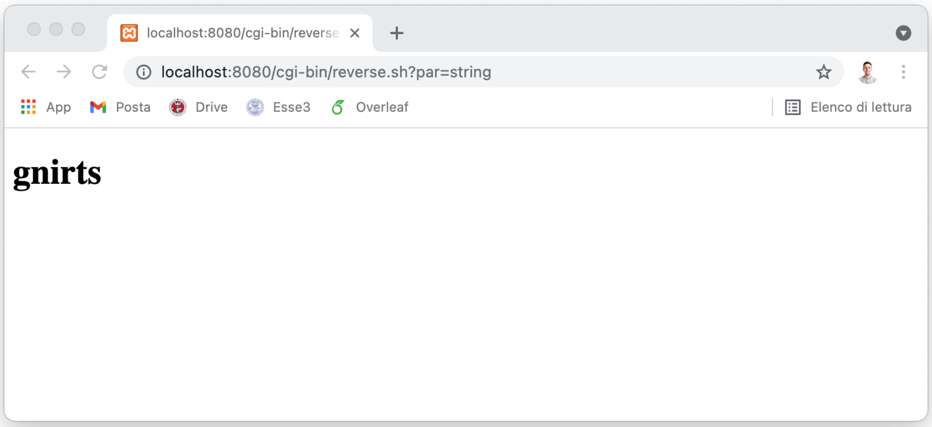
```
http://localhost:8080/cgi-bin/reverse.sh?par=string
```

Figure 5: Request the resource to the cgi-bin folder



```
Last login: Sun Oct  3 09:47:33 on ttys000
sebac@MBPdiSebastiano cgi-bin % chmod +x reverse.sh
sebac@MBPdiSebastiano cgi-bin % ls -la reverse.sh
-rwxr-xr-x@ 1 sebac  admin  459  2 Ott 17:36 reverse.sh
sebac@MBPdiSebastiano cgi-bin %
```

Figure 6: Change the permissions


localhost:8080/cgi-bin/reverse
localhost:8080/cgi-bin/reverse.sh?par=string
App
Posta
Drive
Esse3
Overleaf
Elenco di lettura

gnirts

Figure 7: Response with the correct reversed string