



# UNIVERSITY OF TRENTO

## ASSIGNMENT 1

Vinci Nicolò 220229

[nicolo.vinci@studenti.unitn.it](mailto:nicolo.vinci@studenti.unitn.it)

Web Architectures 2021/2022

03 October 2021

# Contents

<b>1</b>	<b>First Part</b>	<b>2</b>
1.1	Introduction . . . . .	2
1.2	Development . . . . .	2
1.3	App running . . . . .	2
<b>2</b>	<b>Second Part</b>	<b>6</b>
2.1	Introduction . . . . .	6
2.2	Development . . . . .	6
2.3	App running . . . . .	6

# 1 First Part

## 1.1 Introduction

The first part of the assignment consists of developing a web server exploiting the Java socket. The web server must be able to serve all the incoming *GET* request. If a *GET* request contains a token *process*, the web server should launch an external program if it exists. The next token after *process* is the name of the external program. Moreover, there may be a parameter to pass to the external program. It can be found after the delimiter *?* in the URL. Then, the web server returns the external program output to the browser. The web server has been developed starting by the *MiniHTTPD* project provided by the professor. Hence, the web server can be launched passing the port as parameter or it will start on the default port 8888.

## 1.2 Development

First of all, a *Reverse.Java* program has been developed in order to get a string as parameter and return its reverse. So, a client should perform a *GET* request with the following URL *http://localhost:8888/process/reverse?roma*. So, the web server receives that request, it will call *Reverse.Java* as an external process passing to it the parameter *roma*. Then, the web server retrieves the output of the program and forwards it to the browser. Considering the previous parameter, the output will be *amor*. The web server also returns an upper html fragment called *fragment1.html* and a lower html fragment called *fragment2.html*. The output of the external program will be in the middle of them. Firstly, the request is splitted in different tokens exploiting the *StringTokenizer*. The delimiters to split request are the blank space, the */* and the *?*. Two checks are performed by the web server on the request:

- first check: the request must be a *GET* request and contain more than 1 token. The first token after *http://localhost:8888/* must be *process*.
- second check: the next token after *process* must be *reverse*, which is the name of the external program.

If one of the two checks does not pass, the web server returns *400 bad request*. After that, the web server also retrieves the parameter and calls the *Reverse.Java* program. If no parameter is passed in the request, the external program will return *You must insert a string!*. Otherwise, it will return the reverse.

## 1.3 App running

Firstly, the web server should be launched in order to be ready to listen on possible incoming request. In the figure 1, it has been started on the default port 8888.

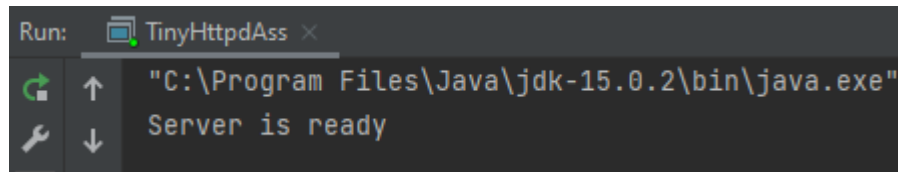


Figure 1: Web server in ready state

Then, if a user performs a request such as *http://localhost:8888/process/reverse?roma*. The result is shown in figure 2.

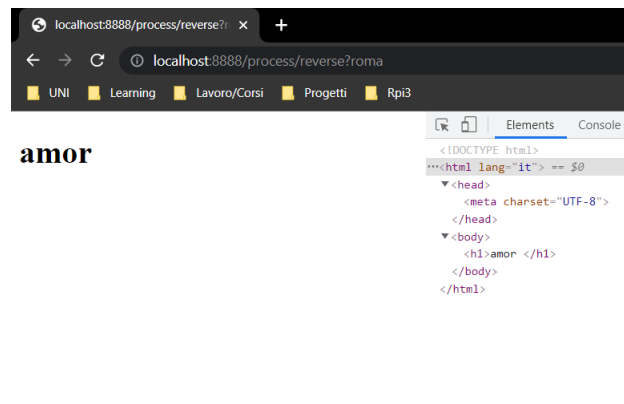


Figure 2: Request result

Instead, if the user performs a request without passing a parameter such as *http://localhost:8888/process/reverse* or *http://localhost:8888/process/reverse?*, the response will be *You must insert a string!* as shown in the figure 3.

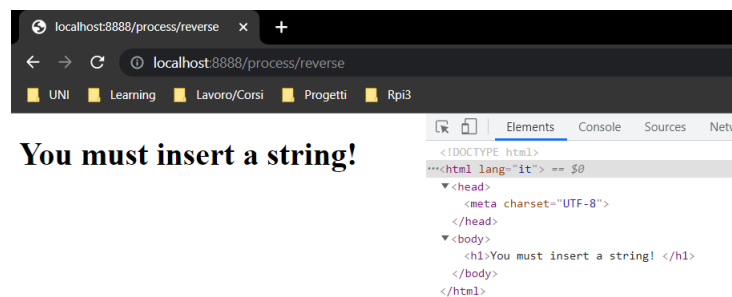


Figure 3: Request result without parameter

Otherwise, if the user performs a request with a wrong external program name such as *http://localhost:8888/process/revers* or omitting the external program name such as *http://localhost:8888/process*, the web server returns a *400 bad request* response.

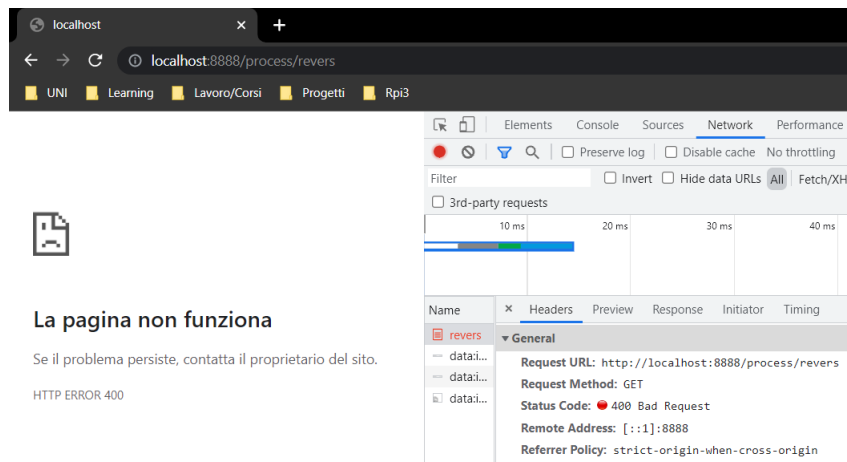


Figure 4: Request result with wrong external program name

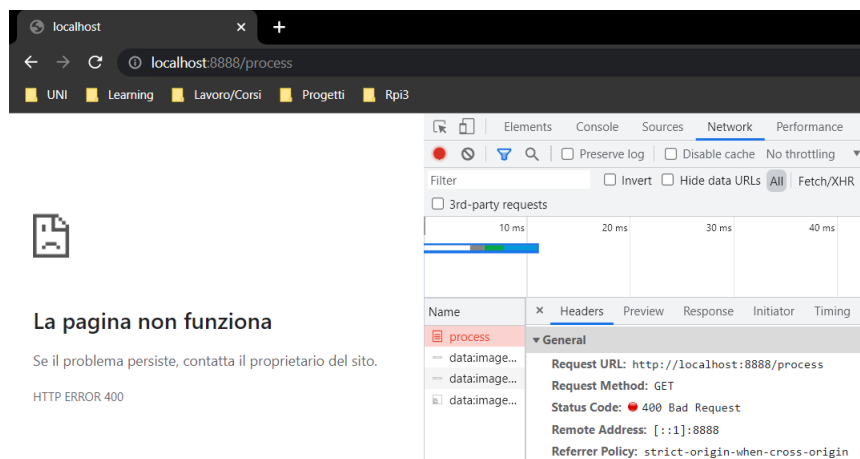


Figure 5: Request result omitting external program name

Also if the user performs a request without *process* in the URL, the web server will return *400 bad request* response.

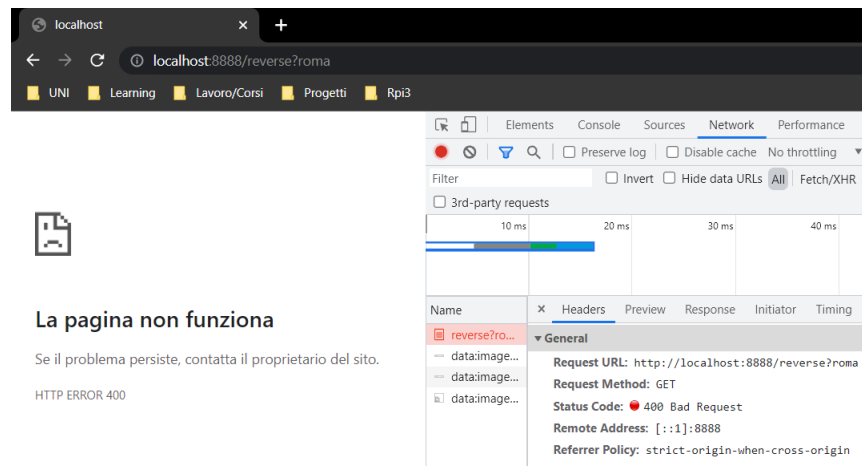


Figure 6: Request result without *process*

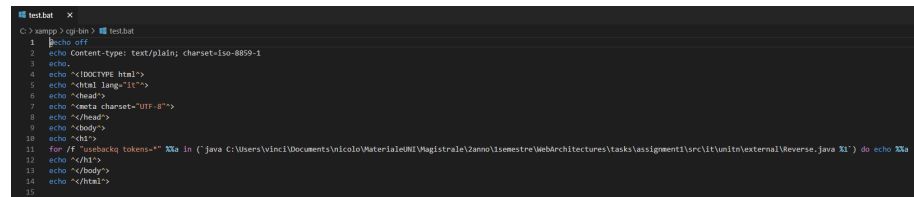
## 2 Second Part

### 2.1 Introduction

The second part of the assignment consists of installing and running an Apache WebServer with XAMPP. A batch script has been developed and placed in the *cgi-bin* folder in order to return the reverse of a string. The latter will be passed as parameter. So, a user can call the batch script and the web server will execute it and return dynamically the output.

### 2.2 Development

First of all XAMPP has been installed in order to be able to run an Apache WebServer on port 80 or 443. Then, the script in the figure 7 has been developed.



```
test.bat
@echo off
set Content-Type: text/plain; charset=iso-8859-1
echo
echo <!DOCTYPE html>
echo <html lang="it">
echo <head>
echo <meta charset="UTF-8">
echo </head>
echo <body>
echo <div>
for /f "usebackq tokens=*" %1 in ('java C:\Users\vinci\Documents\nicolo\Materiali\Omi\Magistrale\2anno\1semestre\WebArchitectures\tasks\assignment1\src\it\unit5\external\Reverse.java %1') do echo %1
echo </div>
echo </body>
echo </html>
```

Figure 7: Batch script

Firstly, the script defines the header *Content-type* to specify the MIME type of the body. After that, the first html fragment is specified and the external Java program called *Reverse.java* is called. The input parameter is retrieved with *%1* and it is passed to the external program. The *for* cycle at line 11 of the script is able to retrieve the Java program output. At the end, the second html fragment is defined in order to close the upper html tags. Hence, the batch script has been stored in the *cgi-bin* folder.

### 2.3 App running

First of all, the Apache WebServer needs to be started via XAMPP GUI as show in the figure 8. The user can perform the request *http://localhost:80* or *http://localhost:443* in order to check if the WebServer has correctly been started.

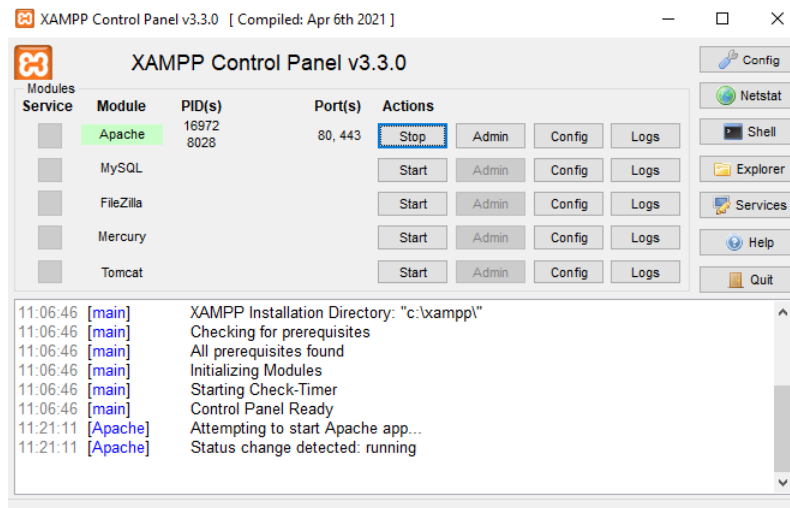


Figure 8: Start Apache WebServer

Then, the user can call the batch script performing a request such as *http://localhost:80/cgi-bin/test.bat?roma*. The html response will be printed out as shown in figure 9. However, there is the reverse of the input string between the `<h1>` tag.

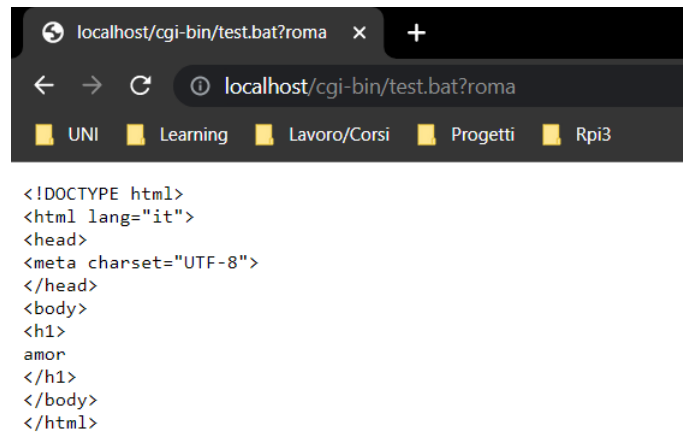


Figure 9: Run test.bat

Instead, the user can make a request without passing a parameter such as *http://localhost:80/cgi-bin/test.bat*.

The response will be *You must insert a string!* as show in the figure 10.



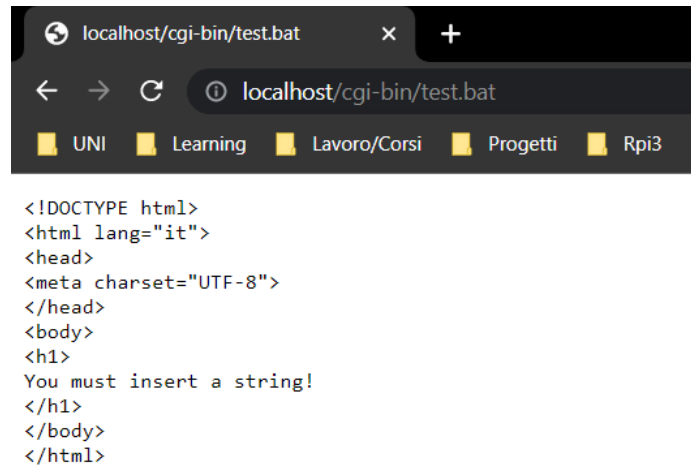


Figure 10: Call test.bat without parameter

Otherwise, the user can perform a wrong request calling a wrong batch script such as

*http://localhost:80/cgi-bin/wrong.bat.*

The response will be a *404 Not Found* error as shown in the figure 11.

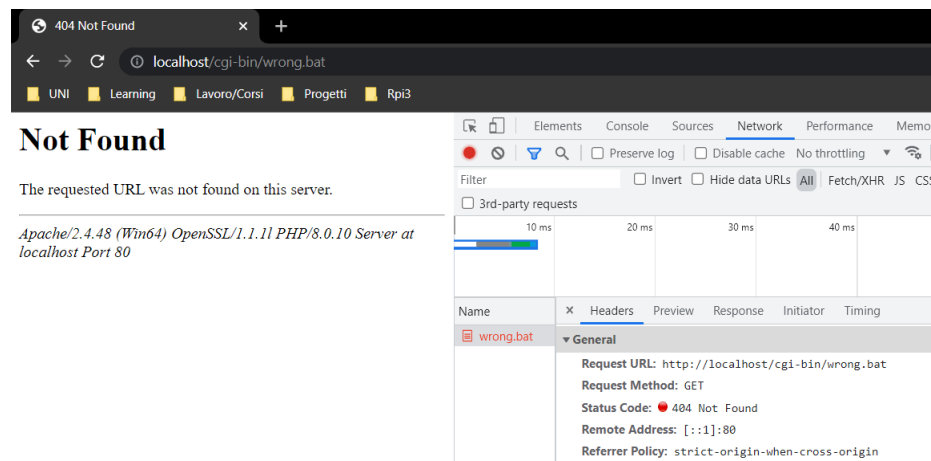


Figure 11: Wrong call