Web Architectures - Delivery 1

Marrocco Simone

October 1, 2022

Contents

1 Section 1 - Modified MiniHTTPD project 2

1.1 Introduction 2

1.2 Modify *TinyHttpdConnection.java* 2

1.2.1 Inside *run()* 2

1.2.2 Extra functions used 3

1.3 *ReverseString.java* 4

1.4 Results 5

2 Section 2 - Apache serving a bash file using the java program made before 6

2.1 Introduction 6

2.2 *reverseString.sh* 6

2.3 Results 7

1 Section 1 - Modified MiniHTTPD project

1.1 Introduction

The objective of this assignment was to modify the MiniHTTPD project given by the professor and adding some code so that the server, when the url starts with *process/*, launches an external process that takes the input from the query parameters and returns an output. In particular, we needed to create the process that reverses the input string and outputs it, made in a new java file.

1.2 Modify *TinyHttpdConnection.java*

1.2.1 Inside *run()*

In the overriden function *run()* we add the pieces of code written below.

First, we intercept only the requests that start with *process/*, since there may be files that start with *process* which we do not want to serve in this particular if-case.

Inside it, we use different combinations of *String.split()* and controls to ensure that the url is in the form
process/reverse/query_name=query_value and if not we call the function *sendErrorBadRequest*, that sends an error page (its code is explained later), with a specific error message.

When we are sure to have the correct command and at least one query, we call the function *callExternalProcess* with input the value of the first query param. Notice how, in line 22 of the code below, we split on every non letter characted: in this way, the String *command[1]* will be separated on all values = and &.

We finally return the output of the external process with a basic HTTP header.

```
1  ///// old code
2  if (req.endsWith("/") || req.equals("")) {
3    req = req + "index.html";
4  }
5  // ASSIGNMENT 1 - Marrocco Simone
6  if (req.startsWith("process/")) {
7      // req should be 'process/reverse?query_name=query_value
       ', otherwise error
8      String[] url = req.split("/");
9      if (url.length != 2){
10         sendErrorBadRequest(ps, req, "Too many page
       requested. Ask only one command");
```

```
        return;
    }
    String[] command = url[1].split("\\?");
    if (!command[0].equals("reverse")){
        sendErrorBadRequest(ps, req, "Command not yet implemented");
        return;
    }
    if (command.length != 2){
        sendErrorBadRequest(ps, req, "No query given");
        return;
    }
    String[] queries = command[1].split("[\\W]");
    if (queries.length < 2){
        sendErrorBadRequest(ps, req, "incorrect or null queries");
        return;
    }
    String query_value = queries[1];
    String output = callExternalProcess(query_value);
    ps.print("HTTP/1.1 200 OK\r\n");
    ps.print("Content-Type: text/html\r\n");
    ps.print("\r\n");
    ps.print(output);
} else
// OPEN REQUESTED FILE AND COPY IT TO CLIENT
///// continue with old code
```

Code 1: Code added inside run()

1.2.2 Extra functions used

We also use two functions: the first is simply to send a page with code 400 Bad Request and an error message explaining why the request was bad.

The second function is used to call the external process and return the output. The function does not deal with **IOException** errors but leaves the calling function, in this case *run*, to deal with it (as it already does).

The external process called is a java command with input the **ReverseString.java** file location and the user input. However, to get the first one we need to use absolute location on our machine: that means not only that the system is not secure, but the server launch would not work in another machine without changing this path. This is caused by the IntelliJ run command, that moves the compiled files to the Tomcat location which is, of course, a totally different path. For the scope of this deliverable, however, it can be ignored: to start the project we just need to change the line 14 of the code below.

The rest of the code uses the **ProcessBuilder** class to create the process and get the output.

```java
/** error page sent when bad request while processing a /
   process/ request */
private void sendErrorBadRequest(PrintStream ps, String req,
     String error_msg) {
    ps.print("HTTP/1.1 400 Bad Request\r\n\r\n");
    ps.println(error_msg);
    ps.println("The wrong URL: "+req);
    System.out.println("400 Bad Request: " + req);
}

/** call the external process that uses another Java file */
protected String callExternalProcess(String input) throws
   IOException {
    /** the bash command used. Instead of a single String
   with spaces, we need an array of Strings */
    String[] bash_command = {
            "java",
          "/home/simone/Java/Projects/MiniHTTPD-Marrocco/src
   /it/unitn/disi/webarch/tinyhttpd/ReverseString.java",
            input
    };

    ProcessBuilder processBuilder = new ProcessBuilder();
    processBuilder.command(bash_command);
    Process process = processBuilder.start();
    BufferedReader bufferedReader = new BufferedReader(new
   InputStreamReader(process.getInputStream()));
    StringBuilder stringBuilder = new StringBuilder();
    String line = null;
    while ( (line = bufferedReader.readLine()) != null) {
        stringBuilder.append(line);
        stringBuilder.append(System.getProperty("line.
   separator"));
    }
    String result = stringBuilder.toString();
    return result;
}
```

Code 2: functions added to the class TinyHttpdConnection

1.3 *ReverseString.java*

The external process called is a java command with input this file that takes a String as an argument, reverses it with the **StringBuilder** class and prints it.

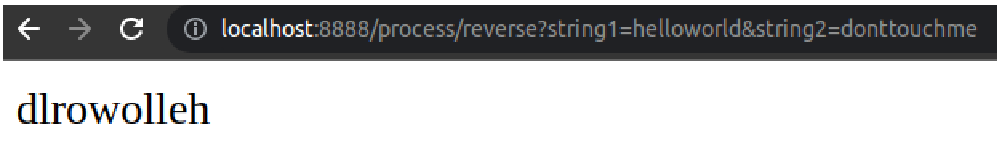
```java
package it.unitn.disi.webarch.tinyhttpd;

public class ReverseString {
    public static void main(String[] a){
        String inputString = a[0];
        if(inputString == null) System.out.println("ERROR: no params given");
        System.out.println(new StringBuilder(inputString).reverse());
    }
}
```

Code 3: Class that reverses a string

1.4 Results

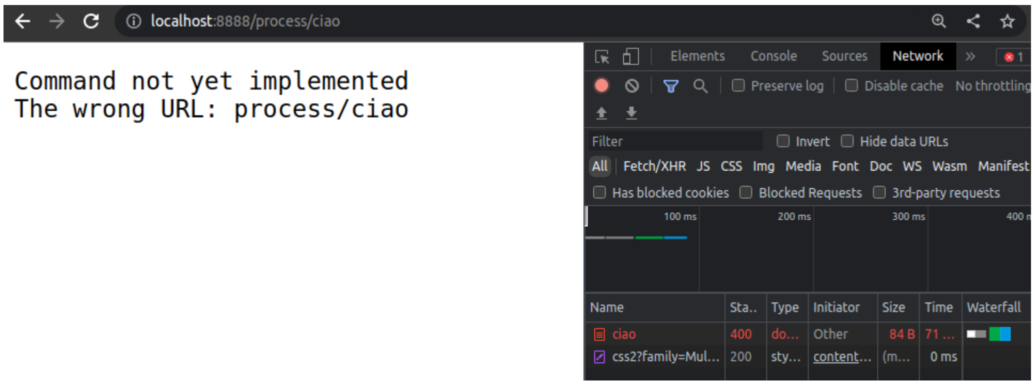
The server is correctly built. Here are some screenshots of the results.



```
localhost:8888/process/reverse?string1=helloworld&string2=donttouchme
```

dlrowolleh

Figure 1.1: The server processes correctly the first query parameter. Other ones are ignored

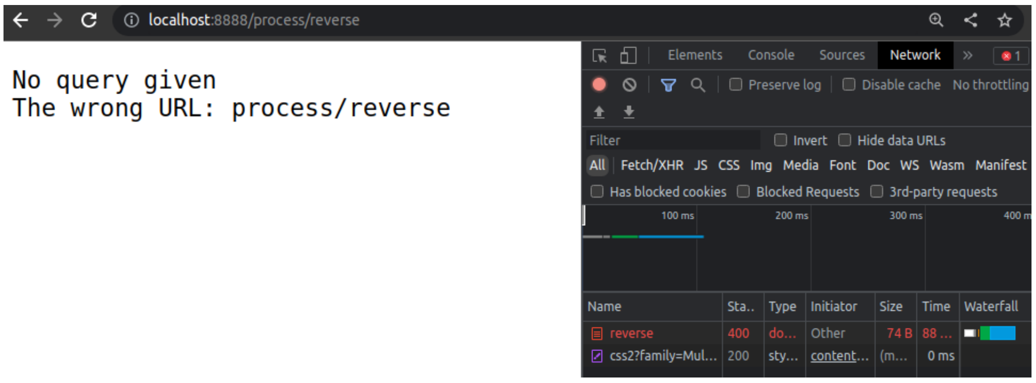


```
localhost:8888/process/ciao

Command not yet implemented
The wrong URL: process/ciao
```

Name	Sta..	Type	Initiator	Size	Time
ciao	400	do...	Other	84 B	71 ...
css2?family=Mul...	200	sty...	content...	(m...	0 ms

Figure 1.2: If a different command than *reverse* is given, the server returns 400 with an explanation


localhost:8888/process/reverse

```
No query given
The wrong URL: process/reverse
```

Name	Sta..	Type	Initiator	Size	Time	Waterfall
reverse	400	do...	Other	74 B	88 ...	
css2?family=Mul...	200	sty...	content...	(m...	0 ms	

Figure 1.3: If there are no query parameters, the server returns 400 with an explanation

2 Section 2 - Apache serving a bash file using the java program made before

2.1 Introduction

The objective of the second part was to install correctly Apache with Xampp and, inside the cgi-bin folder write a bash file that does the same as the server above: taking input from the query params, calling **ReverseString.java** to process it and returning the output. This script file would then be calleable from the browser web at the adress *localhost/cgi-bin/file.sh* when Apache is running.

2.2 *reverseString.sh*

The bash file below takes the query params from the global variable *$QUERY_STRING* given by Apache, and splits it with the command **${ QUERY_STRING##}**. In particular, this command takes the substring from the last equal sign to the end, so on the contrary of the first assignment this file would not serve only the first query parameter value, but only the last one.

We can see how here we have the same problem as before on the **ReverseString.java** file location. We also need to specify the **$JAVA_HOME** location, since Apache is not capable of reading it from our system (or, like this case, use the entire path of the command).

```
#!/bin/sh
```

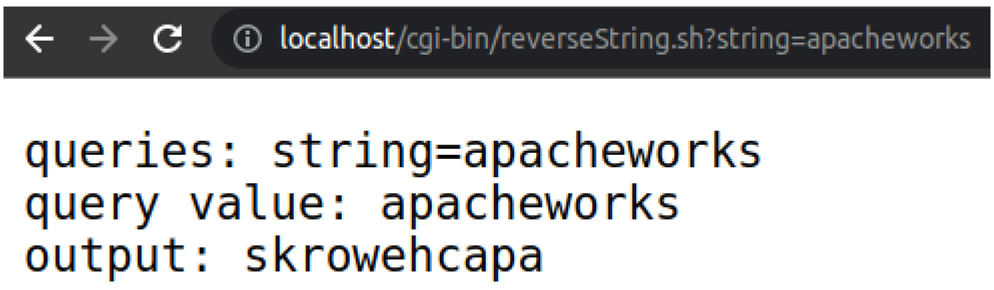
```bash
file="/home/simone/Java/Projects/MiniHTTPD-Marrocco/src/it/
    unitn/disi/webarch/tinyhttpd/ReverseString.java";
query_value=${QUERY_STRING##*=};
output=$(/home/simone/Java/openjdk-19_linux-x64_bin/jdk-19/
    bin/java $file $query_value);

echo "Content-type: text/plain; charset=iso-8859-1";
echo;
echo "queries: $QUERY_STRING";
echo "query value: $query_value";
echo "output: $output";
```

Code 4: file bash used to call ReverseString.java

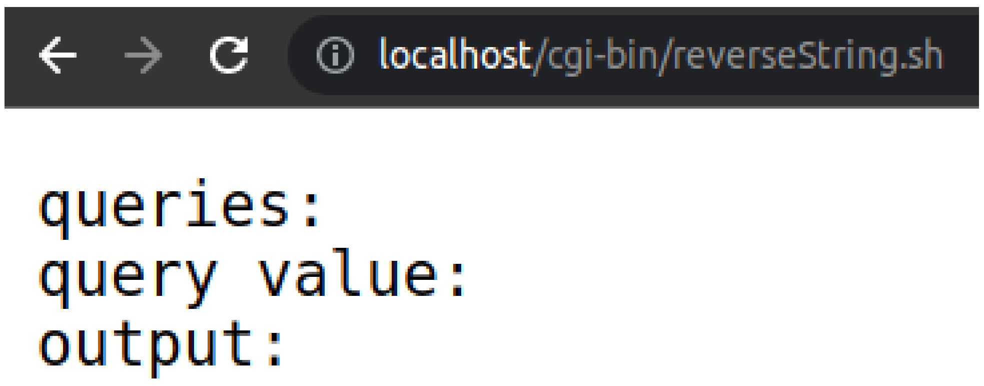
2.3 Results

The file is correctly executed. Here are some screenshots of the results.



```
queries: string=apacheworks
query value: apacheworks
output: skrowehcapa
```

Figure 2.1: The server processes correctly the query parameter



```
queries:
query value:
output:
```

Figure 2.2: If there are no query parameters, the server returns an empty string