Assignment 1 - Dynamic Web site, without using Web Languages

Web Architectures - 02/10/2022

Sabin Andone

Section 1

1.1 Introduction

For the first part of this assignment, it was necessary to modify the TinyHttpd web server provided in the class and extend it in such a way that all the URLs that start with the token "process" launch an external process. The format of an URL with such token would assume a format like the following:

http://localhost:8000/process/...

In particular, http://localhost:8000/process/reverse?par1=string activates an external process that takes in input the parameter string, reverses it and then returns the result. For example, given in input the string "example", it returns the string "elpmaxe". For this work, the realization of the solution was done using IntelliJ Idea Ultimate on Windows

1.2 Solution

The solution of this part of the assignment consists of the extension of the TinyHttpd web server provided in class. The principal two additions are:

1. modification of the parse request step in order to consider the process token case and to reverse the parameter string (note: the figures which the description of this modification is referring are all located in section "1.3 Running time and screenshots" since this is part of the steps of the running time of the program)

2. addition of a function that can launch the external process named startProcess(String token, String result)

Regarding the parse request step, in order to take the process token and subsequently the reverse token, a new StringTokenizer variable was created and this new variable takes the request as input. Immediately after its creation, it checks if there are more tokens in the request, as in Figure 1.2.

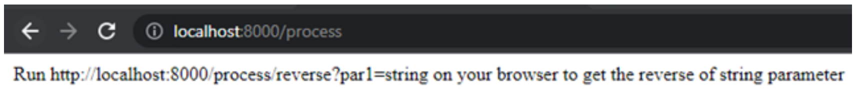
After that, it is necessary to check if there is the process token in the request and then check if also the reverse token is present in the request (Figure 1.3).

If the reverse token is present, then we execute the reverse process, otherwise we execute the other process, which will tell the user to put an URL in the format http://localhost:8000/process/reverse?par1=string in order to launch the reverse process. Note that if reverse token is in the request, we also take the parameter string and reverse it (Figure 1.4).

startProcess(String token, String result) is the function that launches the external process. It is a function that takes in input the token and the result of the reverse operation (note that if reverse is not in the URL, we just give in input an empty string and it will not be considered). By calling this function, the first thing it does is checking the operating system on which the JVM is running at the moment. After that we create the ProcessBuilder and then we run the command. It is important to consider first the operating system since Windows uses batch files whereas other operating systems use shell files.

1.3 Running time and screenshots

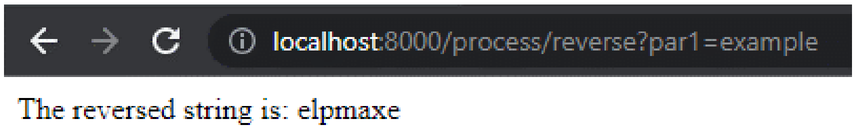
Once we run the TinyHttpd program in IntelliJ, it is necessary to go to the browser and insert an URL that starts with process token. In the first example we have the URL http://localhost:8000/process which tells the user to put an URL in http://localhost:8000/process/reverse?par1=string format in order to launch the reverse process.



Run http://localhost:8000/process/reverse?par1=string on your browser to get the reverse of string parameter

If, instead, we launch the reverse process, we obtain the reversed string on the browser:

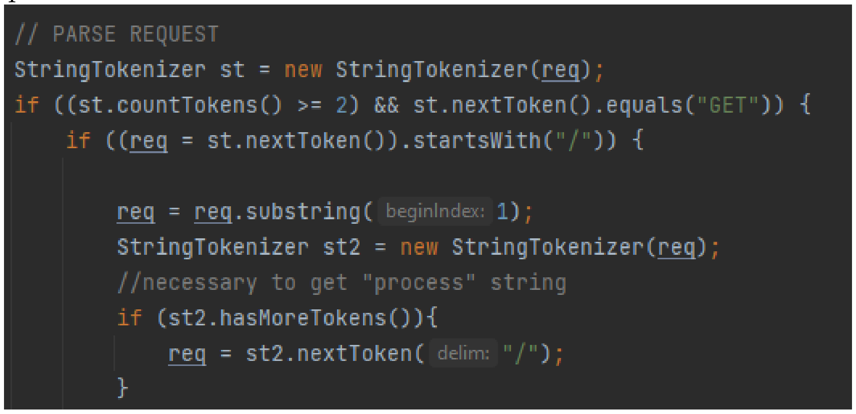
Figure 1.1: In this case we have the string "example" as parameter and we get "elpmaxe" as the output of the process.



The reversed string is: elpmaxe

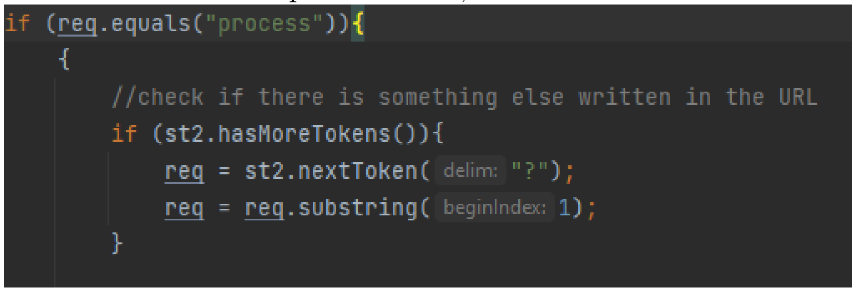
When the program starts, it creates the TinyHttpd server which will wait for any TinyHttpdConnection that may arrive at the server in the future. For this assignment, the initial port has been changed to port 8000.
When any URL is launched on the browser, a TinyHttpdConnection is set and then the start() function is called which triggers the run() function. When run() is called the first thing it does is opening the sockets for reading and writing, reading the request and request headers. These steps are the same as in the original TinyHttpd project so the code was not changed up to this point (except for the port, which was requested to be port 8000).

Figure 1.2: After that it is necessary to do the parse request step, in which we take the request and create two string tokenizers that can retrieve tokens from the request. In this case we first create the first string tokenizer which takes the entire request as input and gives the token which contains our URL. The second string tokenizer is used to retrieve the process and retrieve tokens from the URL.



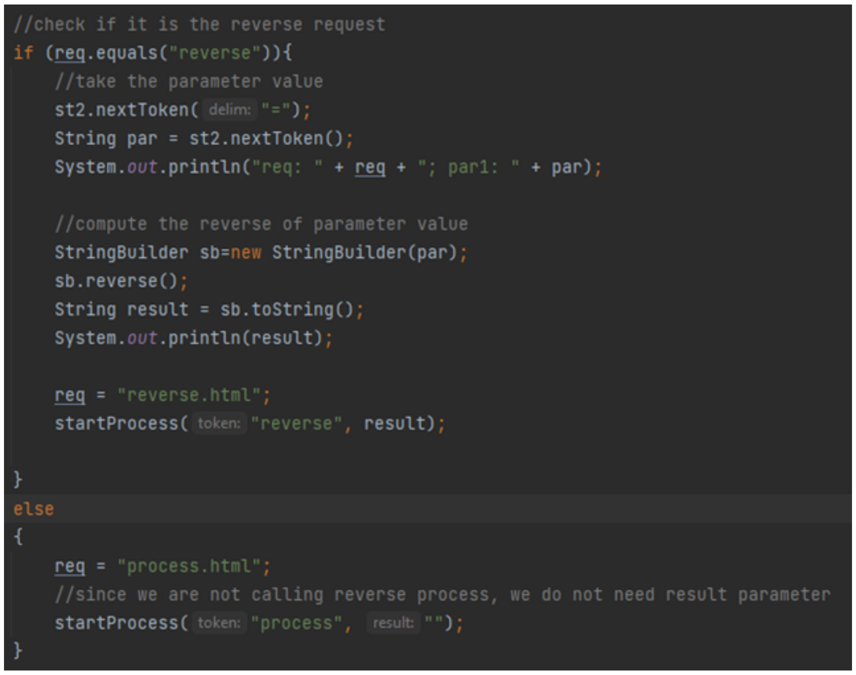
```
// PARSE REQUEST
StringTokenizer st = new StringTokenizer(req);
if ((st.countTokens() >= 2) && st.nextToken().equals("GET")) {
    if ((req = st.nextToken()).startsWith("/")) {

        req = req.substring( beginIndex: 1);
        StringTokenizer st2 = new StringTokenizer(req);
        //necessary to get "process" string
        if (st2.hasMoreTokens()){
            req = st2.nextToken( delim: "/");
        }
```

Figure 1.3: Checks if the URL starts with process token, and in case checks also if it has the reverse token.



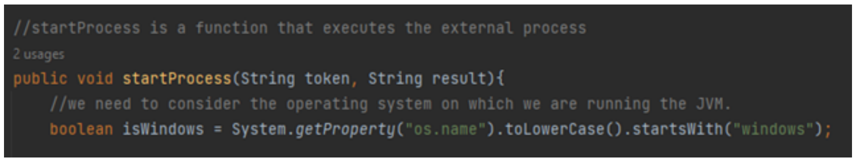
```
if (req.equals("process")){
    {
        //check if there is something else written in the URL
        if (st2.hasMoreTokens()){
            req = st2.nextToken( delim: "?");
            req = req.substring( beginIndex: 1);
        }
```

Figure 1.4: If also reverse token is present in the URL it takes the parameter string and reverses it. It is necessary to note that req variable is set to the name of the html page which will be created by the external process.



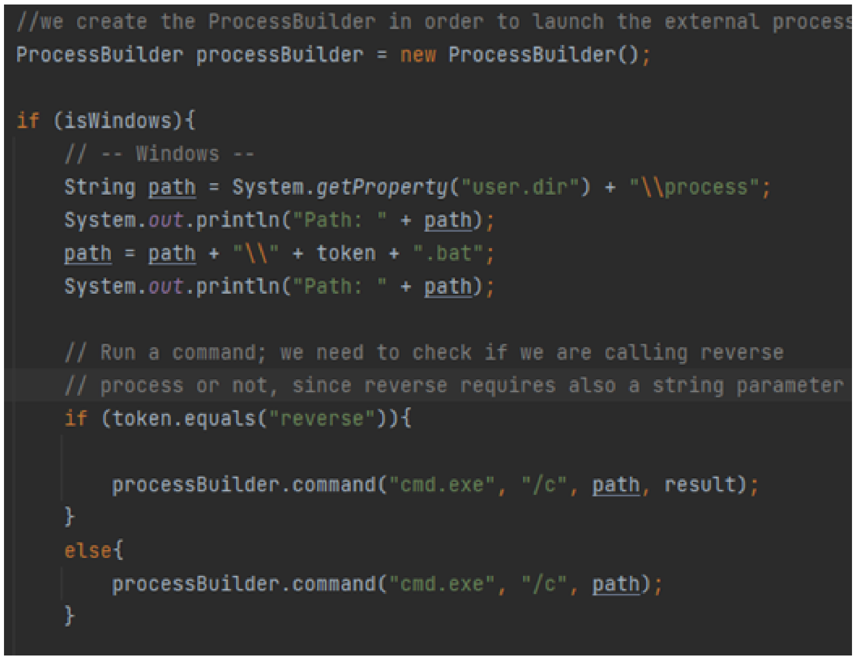
```
//check if it is the reverse request
if (req.equals("reverse")){
    //take the parameter value
    st2.nextToken( delim: "=");
    String par = st2.nextToken();
    System.out.println("req: " + req + "; par1: " + par);

    //compute the reverse of parameter value
    StringBuilder sb=new StringBuilder(par);
    sb.reverse();
    String result = sb.toString();
    System.out.println(result);

    req = "reverse.html";
    startProcess( token: "reverse", result);

}
else
{
    req = "process.html";
    //since we are not calling reverse process, we do not need result parameter
    startProcess( token: "process",  result: "");
}
```

Figure 1.5: As said in the previous section, startProcess(String token, String result) is the function that launches the external process. It is called by giving in input the name of the process to trigger and the value of the parameter string (or an empty string if it is not the reverse process). It considers the operating system of the JVM.



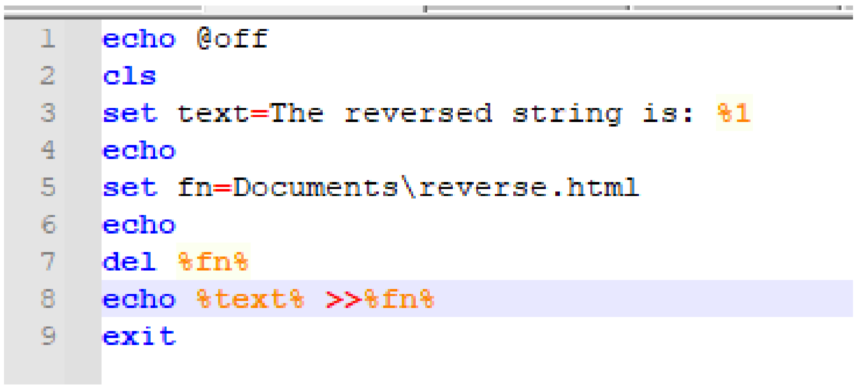
```
//startProcess is a function that executes the external process
2 usages
public void startProcess(String token, String result){
    //we need to consider the operating system on which we are running the JVM.
    boolean isWindows = System.getProperty("os.name").toLowerCase().startsWith("windows");
```

Figure 1.6: Create the ProcessBuilder in order to execute the command that launches the external process.



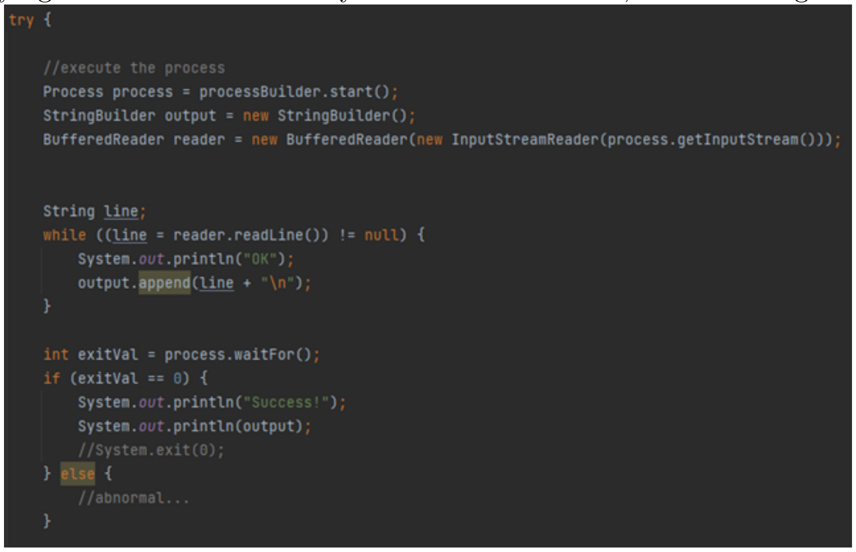
```java
//we create the ProcessBuilder in order to launch the external process
ProcessBuilder processBuilder = new ProcessBuilder();

if (isWindows){
    // -- Windows --
    String path = System.getProperty("user.dir") + "\\process";
    System.out.println("Path: " + path);
    path = path + "\\" + token + ".bat";
    System.out.println("Path: " + path);

    // Run a command; we need to check if we are calling reverse
    // process or not, since reverse requires also a string parameter
    if (token.equals("reverse")){

        processBuilder.command("cmd.exe", "/c", path, result);
    }
    else{
        processBuilder.command("cmd.exe", "/c", path);
    }
```

Figure 1.7: The process which is called in the case of reverse token is a batch file which takes the string parameter and puts it in a text variable which then is written into an html file which is created in that moment. If we don't call the reverse command, then another process will be triggered which will say to the user to use the correct URL for the reverse process.



```bat
echo @off
cls
set text=The reversed string is: %1
echo
set fn=Documents\reverse.html
echo
del %fn%
echo %text% >>%fn%
exit
```

Figure 1.8: The external process is being executed. After its execution, the program continues by opening the requested file and copying it to the client. Finally it closes all channels, as in the original TinyHttpd project.



```
try {

    //execute the process
    Process process = processBuilder.start();
    StringBuilder output = new StringBuilder();
    BufferedReader reader = new BufferedReader(new InputStreamReader(process.getInputStream()));


    String line;
    while ((line = reader.readLine()) != null) {
        System.out.println("OK");
        output.append(line + "\n");
    }

    int exitVal = process.waitFor();
    if (exitVal == 0) {
        System.out.println("Success!");
        System.out.println(output);
        //System.exit(0);
    } else {
        //abnormal...
    }
```

Section 2

2.1 Introduction

For the second part of this assignment, it was necessary to run the java program developed for the first part of the assignment using an Apache WebServer. In order to be able to do so, it was suggested to create a batch (or a shell file if not on Windows) that:

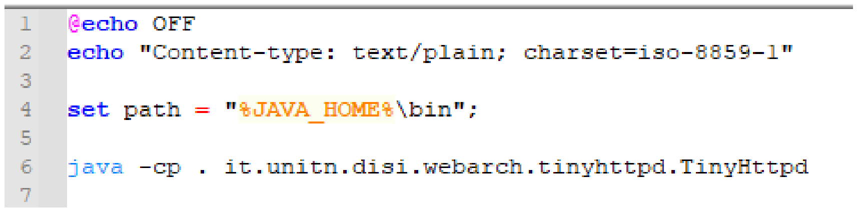
1. retrieves the parameters
2. launches the java program "prog" as java prog params

2.2 Idea for solution and running time

Note: this section only proposes ideas for the solution since the actual solution was not possible to be executed correctly due to problems with xampp program (see section 2.3 Comments)

The idea for the solution of the second part of this assignment is to develop a batch file that ables to user to execute the java program by calling such batch file directly from the browser. Such file must be located in cgi-bin directory. The code of the batch file is shown in Figure 2.1.

Figure 2.1: Batch code source: first we need to specify the content-type of the output which will be printed on the client's screen. After that it is necessary to set the path to the bin folder of the jdk installed on the user's machine. Finally we execute the java command which runs the java program. Note that in this case we directly execute the .class file which was compiled previously. -cp is necessary to be able to execute the package since the java program is organized in the it.unitn.disi.webarch.tinyhttpd package. For this work, a copy of the package with the .class files was created inside cgi-bin in order to be able to execute the batch file and run correctly the java program



```
@echo OFF
echo "Content-type: text/plain; charset=iso-8859-1"

set path = "%JAVA_HOME%\bin";

java -cp . it.unitn.disi.webarch.tinyhttpd.TinyHttpd

```

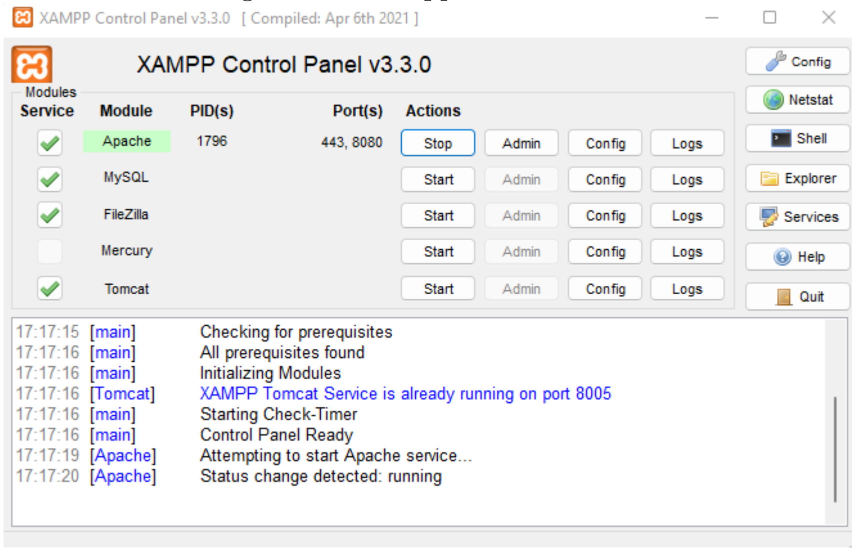
Once the batch file is done, it is necessary to use the Apache Web Server by opening Xampp Control Panel and start Apache module as in Figure 2.2.

After we start Apache web server, the user must put into the browser the URL:

http://localhost:8080/cgi-bin/launcher.bat

where launcher.bat is the name of the batch file created to execute the java file. Now that the batch file is executed the TinyHttpd program is executed the TinyHttpd server is ready to receive requests from the client's browser (as in the first part of the assignment).

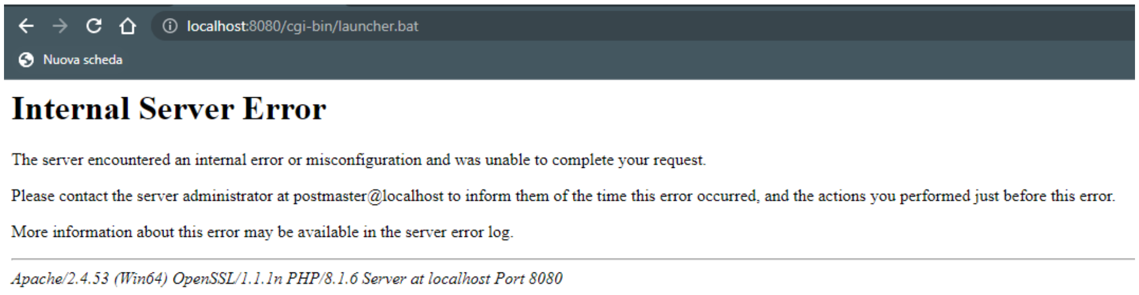
Figure 2.2: Xampp Control Panel



```
17:17:15 [main]     Checking for prerequisites
17:17:16 [main]     All prerequisites found
17:17:16 [main]     Initializing Modules
17:17:16 [Tomcat]   XAMPP Tomcat Service is already running on port 8005
17:17:16 [main]     Starting Check-Timer
17:17:16 [main]     Control Panel Ready
17:17:19 [Apache]   Attempting to start Apache service...
17:17:20 [Apache]   Status change detected: running
```

2.3 Comments

Problem: due to unknown reasons (perhaps due to configuration issues of xampp), by using the Apache web server with xampp it was not possible to launch the batch file directly from the browser (error 500 - Internal server error always appears), thus it was not possible to execute the java program by using Apache with xampp.



Internal Server Error

The server encountered an internal error or misconfiguration and was unable to complete your request.

Please contact the server administrator at postmaster@localhost to inform them of the time this error occurred, and the actions you performed just before this error.

More information about this error may be available in the server error log.

Apache/2.4.53 (Win64) OpenSSL/1.1.1n PHP/8.1.6 Server at localhost Port 8080

Inside Apache (error.log) the following error is shown everytime the batch file has been attempted to be executed: ”Premature end of script headers: launcher.bat”.