/\* MiniWebServer.java

\*

\* Name: Rohan Dhoyda

\* Student ID: 2115018

For CSC435, when you copy this to MiniWebserver.java: Add the appropriate class header HERE.

Copyright (C) 2020 with all rights reserved. Clark Elliott

1.1

Point your browser to:

http://localhost:2540/ or...

http://localhost:2540/WebAdd.fake-cgi or...

http://localhost:2540/any/string/you/want.abc

...to get a response back. Keep in mind there may be FavIco requests.

Use the WebAdd.html form to submit a query string to WebResponse, based on

the input to the form. You can probably "click on" the file in your

directory. Locally it will have a URL of something like:

file:///C:/Users/Elliott/435/java/MiniWebserver/WebAdd.html

You should see:

Hello Browser World N

...along with some request information.

See WebAdd.html source HTML below.

To complete the MiniWebserver.java assignment: (a) Modify this file (or

start your own from scratch) so that your MiniWebserver returns (from the

WebAdd.html request) HTML containing the person's name and the sum of the

two numbers. (b) Continue modifications so that the return screen is valid

HTML that has the user's name entered by default (but modifiable), the two

numbers entered by default (but modifiable), and a working Submit button. In

other words, the user can keep pressing submit to get a response an infinite

number of times.

You can use the Firefox console (control-shift E / Network / Inspector) to

see the Internet traffic. (Note: drag the top line up to give a bigger console

window.)

You can use Wireshark to view the traffic as well.

HTML Reference site:

https://www.w3schools.com/

You may find that including the following in your HTML header helps with

facivon problems (Thanks Thomas K.!):

<head> <link rel="icon" href="data:,"> </head>

https://stackoverflow.com/questions/1321878/how-to-prevent-favicon-ico-requests

For the MiniWebserver assignment answer these questions briefly in YOUR OWN

WORDS here in your comments:

1. How MIME-types are used to tell the browser what data is coming.

Ans: MIME Type is an identifier for a particular type or format of information. The client and the server communicate to

each other by passing MIME Type back and forth expaining how to interpret the bodies of the messages. The MIME TYPE can be

image/jpg, image/png , text/plain and in our case its text/html. The server looks at this MIME Types and based on this decide how

to interpret the data. It then helps the browser to recognize the filetype or file that has been sent by the webserver i.e it will

check the extensions. The browser will then be able to choose a suitable dispalying method.

2. How you would return the contents of requested files of type HTML

(text/html)

Ans: The content type string can be manipulated if we predefine it in our case the contents of the requested file will be returned in

text/html as we have given content type as HTML.

3. How you would return the contents of requested files of type TEXT

(text/plain)

Ans: The content type string can be manipulated if we predefine it in our case the contents of the requested file will be returned in

text/html as we have given content type as HTML but we predefine our content type as text/plain it will return requested file in plain format.

\*/

**import** java.io.\*; // This will import input output libraries

**import** java.net.\*; // This will import Java networking libraries

**class** ListenWorker **extends** Thread { // This is a concept of multithreading means many of the Listenworker thread will work simultaneously

Socket sock; // Class member, socket, is defined under ListenWorker Thread.

ListenWorker (Socket u) {sock = u;} // Constructor, assign arg u to local sock

**public** **void** run(){

PrintStream out = **null**; // Input from the socket

BufferedReader in = **null**; // Output to the socket

StringBuilder HTMLResponse = **new** StringBuilder();

**try** {

out = **new** PrintStream(sock.getOutputStream()); //It will return an Output Stream object that receives data

in = **new** BufferedReader

(**new** InputStreamReader(sock.getInputStream())); //It will return and Input Stream object that receives data

System.out.println("HTML Reponse is being send now: " +

Integer.toString(MiniWebserver.i) + "\n" ); // You will have to check the datatype and parameter you need to convert into string in my case it is MiniWebserver.

HTMLResponse.append("<html> <h1> Welcome to the Future => " +

Integer.toString(MiniWebserver.i++) + "</h1> <p><p> <hr> <p>"); // The Display console will some what look like this when we click on the Submit and in side it will also count the html response

out.println("HTTP/1.1 200 OK");

out.println("Connection: Congratulations your account is now hacked and your bank account is empty as well"); // This is to fool a client but it can also give the them a shock

// int Len = HTMLResponse.length();

// out.println("Content-Length: " + Integer.toString(Len));

out.println("Content-Length: 1000"); // I am setting a content length of 1000 over here you can set high if you want. This will manually pass the content length to the server

out.println("Content-Type: text/html \r\n\r\n"); // This will manually pass the content type to the server.

String var = in.readLine();

**if**(var.contains("WebAdd"))

{//GET /WebAdd.fake-cgi?person=Rohan&num1=4&num2=5 HTTP/1.1

String[] request = var.split("\\s+");

String name = request[1].substring(request[1].indexOf("person")+7,request[1].indexOf("&num1=")).replace("+"," "); //This will retreive the name of the person of what the user enters

**int** num1 = Integer.parseInt(request[1].substring(request[1].indexOf("&num1=")+6, request[1].indexOf("&num2=")));// This will retreive the value of num 1

**int** num2 = Integer.parseInt(request[1].substring(request[1].indexOf("&num2=")+6, request[1].length())); //This will retreive the value of num2

**int** sum = num1 + num2; // This will add the value of num 1 and num 2 and will give the result

HTMLResponse.append("<h3> Hello "+name+" Sum for requested values is "+sum+"</h3>"); // This will be displayed on user's console with the result

// out.println("</html>");

out.println(HTMLResponse);

}

sock.close(); // This will close the connection with the client which means the client will be able to re establish a connection with the server, but not the server;

} **catch** (IOException x) {

System.out.println("Error: The connection needs to reset again. Listening again...");

}

}

}

**public** **class** MiniWebserver {

**static** **int** i = 0;

**public** **static** **void** main(String a[]) **throws** IOException {

**int** q\_len = 500; /\* It means that this color server can perform 500 simultaneous operation at a time or you can say 500 client at a time\*/

**int** port = 2540; // The server port assigned is port number 2540

Socket sock;

ServerSocket servsock = **new** ServerSocket(port, q\_len);

System.out.println("Rohan Dhoyda's MiniWebserver coonection established at port 2540."); // When the code will run properly this message will be displayed

System.out.println("Point Firefox browser to http://localhost:2540/abc.\n");

**while** (**true**) {

// waiting for the next client connection:

sock = servsock.accept(); //The accept() method is called by the server to validate an incoming request to the socket

**new** ListenWorker (sock).start();

}

}

}

Discussion Post

1. Do we have to write a code for ADDNUM in miniwebserver that returns the instructions back to client?
2. String manipulation is the process of changing and manipulating string data using operations like deletion, extraction, and substitution. It changes the text in order to achieve a certain result, such as extracting data, changing its format, or combining it with other text plus it is used in many programming languages and used for tasks such as data parsing, text processing, and user input validation. This is the potential threat we can face in Miniwebsever. There might be other threats also and this is the one threat I feel. What can be other security concerns that a Miniwebserver can face?