/\*--------------------------------------------------------

1. Name / Date: Vatsal Parikh, May 1, 2022

2. Java version used (java -version), if not the official version for the class:

java 11.0.12

3. Precise command-line compilation examples / instructions:

> javac MiniWebServer.java

4. Precise examples / instructions to run this program:

In separate shell windows:

> java MiniWebServer

Open WebAdd.html

All acceptable commands are displayed below:

> java MiniServer

5. List of files needed for running the program.

a. MiniServer.java

6. Notes:

You don't need WebAdd.html to run this program. As you can open it from

the link provided in the command line provided after running MiniServer.java

This is possible because I am also controlling the trailing part of the URL.

The link provided in the command line contains only "add" in the trailing part but in the action of form

it is "webadd". I am checking for that part and am providing the output accordingly.

This program takes name and two numbers from the html response provided by WebAdd page.

It can input check and provides different output for each case.

It can handle below cases

1) if the user provide neither name nor numbers

2) if the user does not provide name

3) if the user does not provide numbers by default it takes 0 as input for both numbers

4) if the user only enters name but not the numbers

5) if the user enters string in the numbers field

You can also return the contents of HTML file by providing these lines in the program

-- import java.nio.file.Files;

-- import java.nio.file.Path;

-- import java.nio.file.Paths;

-- Path path = Paths.get("D:/Study/DS/Mini Web Server/WebAdd.html");

-- String html = Files.readString(path);

-- out.println(HTMLResponse + html);

NOTE: You need to change the program. You cannot just copy/paste above lines directly in the program.

(As of now this program can either return from HTML file or directly from a stringvariable)

----------------------------------------------------------\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

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

-> By using the connection type, connection status, content length and reponse we are

able to provide browser with what kind of data, of what kind through which connection is coming.

2. How you would return the contents of requested files of type HTML (text/html)

-> By changing Content-Type to text/html. Connection type and connection status remains

the same and content length depends upon the size of content.

3. How you would return the contents of requested files of type TEXT (text/plain)

-> By changing Content-Type to text/plain.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

// Importing input/output and networking libraries

import java.io.\*;

import java.net.\*;

// MiniWebListenWorker class

class MiniWebListenWorker extends Thread {

// Socket used for making the connection

Socket sock;

// Class constructor which intializes the socket

MiniWebListenWorker (Socket s) {sock = s;}

// Thread which will be used to handle requests

public void run(){

// Input and output which will be used to read from socket and write to socket

PrintStream out = null;

BufferedReader in = null;

// Trying to assign in/out to socket's(client) Input/Output stream and returning error if not possible

try {

out = new PrintStream(sock.getOutputStream());

in = new BufferedReader(new InputStreamReader(sock.getInputStream()));

// sockdata will be used to get the response provided by WebAdd.html

String sockdata;

// name, num1, and num2 will store the user's name and two numbers which is used for sum

String name =""; // Default value

int num1 = 0; // Default value

int num2 = 0; // Default value

// This will be the start of html reponse which this program will give

String response = "<HTML> <BODY> <h1> WebAdd </h1> <p>";

sockdata = in.readLine ();

// This checks the trailing part of the URL string and do the following only if sockdata contains webadd

// It is useful as we can run this program without the WebAdd.html file.

if (sockdata.contains("webadd")){

// Splitting the data

String[] input = sockdata.split("&");

// Capturing name and if it is empty then asking the user for a name

name = input[0].substring(input[0].indexOf("person") + 7, input[0].length());

if(name.length()>0){

name = "Welcome " + name.replace("+", " ") + "!! </p>";

}

else{

name = "Welcome User!! Please enter your name. </p>";

}

// Capturing both numbers and if it is string asking the user for proper input

try{

num1 = Integer.parseInt(input[1].substring(input[1].indexOf("num1") + 5, input[1].length()));

String[] temp = input[2].split(" ");

num2 = Integer.parseInt(temp[0].substring(temp[0].indexOf("num2") + 5, temp[0].length()));

response += name + "<p>The sum of " + Integer.toString(num1)

+ " and " + Integer.toString(num2) + " is "

+ Integer.toString(num1 + num2) + "</p>";

}

catch(NumberFormatException e){

response += name + "<p>Please enter numeric values. </p>";

}

}

System.out.println("Sending the HTML Reponse now: ");

// Variable form contains the HTML reponse which is the replica of WebAdd.html but instead of this we can

// provide the reponse from the content of WebAdd.html. You need to change below code if that's what you want.

String form = "<FORM method=\"GET\" action=\"http://localhost:2540/webadd\">"

+ "Enter your name and two numbers. Server will return the sum.<p>"

+ "<INPUT TYPE=\"text\" NAME=\"person\" size=20 placeholder=\"Enter your name..\"><P>"

+ "<INPUT TYPE=\"text\" NAME=\"num1\" size=12 placeholder=\"Enter number..\"> <br>"

+ "<INPUT TYPE=\"text\" NAME=\"num2\" size=12 placeholder=\"Enter number..\"> <p>"

+ "<INPUT TYPE=\"submit\" VALUE=\"Submit\">"

+ "</FORM> </BODY>"

+ "</HTML>";

// MIME header which is used to specify the connection type, connection status, content length and reponse

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

out.println("Connection: close");

//int Len = response.length() + form.length();

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

// The length was aroud 520 for worst case but just to be on the safe side I provided 550 as content length.

out.println("Content-Length: 550");

out.println("Content-Type: text/html \r\n\r\n");

// The reponse for the previous query and new form is returned back to the browser

out.println(response + form);

// Closing the connection

sock.close();

} catch (Exception x) {

System.out.println("Exception: "+x);

}

}

}

// MiniWebserver class

public class MiniWebServer {

// Main method

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

// Number of request at a time is stored in a queue of length 6

int q\_len = 6;

// Port number used for communication

int port = 2540;

Socket sock;

// Creating a server socket

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

// Letting the user know that the server is running

System.out.println("Vatsal Parikh's MiniWebServer running at 2540.");

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

// Waiting for a client connection

while (true) {

// Accepting requests from clients

sock = servsock.accept();

new MiniWebListenWorker (sock).start();

}

}

}