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

1. Name : Kumaresh Vijayakumar /Date (Month/Date/Year) 10-09-2022

2. Java version used for the execution of addition program :

java version "18.0.2.1" 2022-08-18

Java(TM) SE Runtime Environment (build 18.0.2.1+1-1)

Java HotSpot(TM) 64-Bit Server VM (build 18.0.2.1+1-1, mixed mode, sharing)

3. Steps followed to run this code:

- Place WebAdd.html file and MiniWebserver.java code in your device with in the folder.

- Proceed further by compiling and executing the MiniWebserver.java.

- Now open the HTML file WebAdd.html and it will navigate to the localhost at port 2540

file:///C:/Users/Kumaresh/Documents/Test/WebAdd.html

- Now, enter the Name of Person, First Number and Second Number values

- Hit Submit button.

- A new HTML page with the localhost along with the required output displayed in the same HTML page.

-You can repeatedly submit the data’s without reloading the html web browser

4.Answers for checklist Questions:

a. Question\_1 in the Checklist: Explained how MIME-types are used to tell the browser what data is coming.

Answer: MIME types help us to disturb the data which is sent from the server. This shows the format used to disturb the data from the server. They can be also mentioned as Media Types. Here is a example, if the server is sending a plain text message as a input data which is given by the user client browser understands it and it displays the text into the web browser.

It need not be disturbed. For a change consider the text is coming from the server it should show up all the elements found on the html in the respective user interface before showing the output result to the client side user. So, considering the both of the cases mentioned previously here there are completely different type of details is sent here MIME types

shows us exactly the body of message with its form. Hence MIME type is a type of identifier. The client and server use this type of technique how the body is being disturbed. Multiple MIME types are available : image/jpeg, video/3gpp etc. Here in MiniWebserver we have used "text/html" MIME type which is also mentioned in the program. This MIME type is written in the content type header in the HTTP. This content type shows the body of the response given but the client user. This is the technique that shows MIME types are used to talk about the incoming data to the browser.

b. Question\_2 in the checklist: Explained how you would return the contents of requested files (web pages) of type HTML (text/html)

Answer: For the contents of the webpages to be returned of type HTML

I should:

--> Establish the connection with the server at port 2540 and client.

--> Then Open the HTML file at the mentioned URL/link and have to assure the request is sent from the user side.

--> Next, we have to parse the file name and read in all the contents of HTML to variables if this is needed to perform.

--> After parsing the file name start calculating with the inputs given by the user.

--> Next to calculations html response is built with header and sent.

--> At last, after sending response close the connection between client and server.

c. Question\_3 in the checklist: Explained how you would return the contents of requested files (web pages) of type TEXT (text/plain)

Answer: For the contents of the webpages to be returned of type HTML

I should:

--> Establish the connection with the server at port 2540 and client.

--> Then start the parsing of file name in the beginning itself.

--> Now open the given file and read in all the contents of the text file which is given.

--> Here we start to build the HTTP response and should have a correct charset which is mentioned for the text in HTTP header. (This helps in encoding)

--> Now have to send the details of the header with the response attached.

--> At last after sending response close the connection between client and server.

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

//Coding Starts Here:

//getting all I/O directories

import java.io.\*;

//getting all the java socket and networking directories

import java.net. \*;

// initial class creation MiniWebserver for execution

public class MiniWebserver

{

// this is the Main Class used in the execution of the java program

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

int length = 6; //this is the length of the que which operating system can req.

int portNumber = 2540; // using specific port number 2540

Socket socket; // local variable of socket

// Declaring Server socket variable that takes in request at port 2540 with 6 incoming request for connections

ServerSocket serverSocket = new ServerSocket(portNumber, length);

// Displaying message on the terminal of launching up server

System.out.println("Kumaresh Vijayakumar's MiniServer is running at port:2540."); /// display message in console that takes input req from port 2540

try { // here using try method to block

while(true)

{

socket = serverSocket.accept(); ///this accepts the connection and creates object to server which helps starting NewWorker and introducing start()

new NewWorker(socket).start(); /// here launch the object of NewWorker and introducing new method start()

}

}

catch (IOException ioException) {

ioException.printStackTrace(); //here handles I/O and throw exceptions of it

}

}

}

class NewWorker extends Thread // here mini web server class extends the thread for the class

{

Socket socket; //socket variable is declared for the worker

String BrowserResponse;

NewWorker (Socket s) // here we can find NewWorker constructor is defined

{

socket = s;

}

public void run() //here running the new method as previous mentioned start()

{

try { //// here using try method to block

String BrowserData; //here we find that variable is declared .this all the input data from the browser..

BufferedReader Data1 = new BufferedReader(new InputStreamReader(socket.getInputStream())); ///here we declare variable for input data for reading the inputs from the browser

PrintStream Data2 = new PrintStream(socket.getOutputStream()); // here we declare variable for the data to write....

BrowserData = Data1.readLine(); //here data line is read to our variable of BrowserData

//System.out.print("Data From Browser: " + BrowserData);

if(BrowserData == null) //here we are checking the BrowserData is null or not null if not proceeding forward towards the program

{

System.out.println("No Response");// after checking the BrowserData as null or not if it is null this will be the display message in console

}

String subStringData = BrowserData.substring(4); //here we are taking the new substring from the BrowserData which starts from the index on 4

//System.out.print("\nSubStringData : " + subStringData);

if (subStringData.contains("fake-cgi")) //here we check the string which is pointed to has fake-cgi

{

BrowserResponse = MyWebADD(subStringData); //here introducing the new method again as MyWebADD and need to parse the substringData which is taken as parameter

//System.out.print("Result addition" + BrowserResponse);

}

StringBuilder htmlWebResponse = new StringBuilder(); //here we build a response of web in HTML

htmlWebResponse.append(sendHTMLHeader(BrowserResponse)); //here we build the header for the response of web in HTML

htmlWebResponse.append(Submit()); // here we build submit button for the response of web in html

PrintHTML(htmlWebResponse.toString(), Data2); /// here we display the web response form with the output which is the addition of two numbers as result

System.out.flush(); // here we use flush to clear the buffer completely for the next attempt

socket.close(); // here we close the socket completely without the server

}

catch (IOException ioException) ///here we handle I/O errors and exception returned

{

ioException.printStackTrace();

}

}

public String MyWebADD(String subStringData) // this is the new method to perform the addition process which is the required output in html and the input will be given by the user with the name.here we use the name and numbers of the user so that we can display all with the output

// here we use parse technique to split the inputs which are given by the user and modify our output with the values entered.

{

String trimName = subStringData.substring(21); // here we start to parse the inputs which are from the substring of subStringData

String [] data = trimName.split("[=&\s]"); // here is the main parse tech. is used to split the NameOfPerson keyword, name value,FirstNumber keyword and value =,second number keyword and value...to array of string ....

String NameOfPerson= data[1]; // here name of person is defined with variable and gives it to the array's initial place

String FirstNumber = data[3]; // here FirstNumber is defined with variable and gives it to array's 3rd place

String SecondNumber = data[5]; // here SecondNumber is defined with variable and gives it to array's 5th place

int addition = Integer.parseInt(FirstNumber) + Integer.parseInt(SecondNumber); // here we add first number and second number and then parse it in the means of integer

BrowserResponse = "Hi " + NameOfPerson+ "! Have the rest of day with smile and happiness. Here is the sum of " + FirstNumber

+ " and " + SecondNumber + " is " + addition; // here is the display and shares the result of addition of the 2 numbers which is given as input by the user.This output page shows with the text and input numbers so that they can check those as\_well...

return BrowserResponse; // here we return back the value of BrowserResponse

}

static String sendHTMLHeader(String BrowserResponse) // this is the method which is used to build the header of HTML which takes the data from BrowserResponse and display the result which is the

// addition of two numbers which is given by the user.The output is displayed on the localhost WebAdd.HTML.And the response goes to the same port

// and this performing addition which displays as output in the form of html

{

StringBuilder dataStringofhtml = new StringBuilder(); // here we declare the variable to dataStringofhtml which is used in the method in creating header

dataStringofhtml.append("<html><head><meta charset=\"UTF-8\"><title>WebAdd</title><link rel=\"icon\" href=\"data:,\"></head><body>\n"); // here the header is appended

dataStringofhtml.append("<H1> WebAdd </H1>");

dataStringofhtml.append("<FORM method=\"GET\" action=\"http://localhost:2540/WebAdd.fake-cgi\">"); // here the required form is appended

dataStringofhtml.append("Enter your name and two numbers. My program will return the sum:<p>"); // here the text is appended so that it displays on the webBrowser

dataStringofhtml.append("<INPUT TYPE=\"text\" NAME=\"NameOfPerson\" size=20 value=\"Enter Your Name\"><P>"); // here the input type is appended for NameOfPerson,FirstNumber,SecondNumber and declares all with its defaults

dataStringofhtml.append("<INPUT TYPE=\"text\" NAME=\"FirstNumber\" size=5 value=\"0\" id=\"a\"> <br>");

dataStringofhtml.append("<INPUT TYPE=\"text\" NAME=\"SecondNumber\" size=5 value=\"0\" id=\"b\"> <p>");

if(BrowserResponse.length() > 0) // here we check the output in calculation of add,it shows the BrowserResponse message in top of the submit option so that our required output is correct...

{

dataStringofhtml.append("<p><strong>Result of Addition: </strong>" + BrowserResponse + "</p>");

}

return dataStringofhtml.toString(); // returns the dataStringhtml back as the calculation is completed

}

static String Submit() // this method is used to revert back the submit button on the html

{

return "<input type=\"submit\" value=\"Submit Numbers\"" + "</p>\n</form></body></html>\n";

}

static void PrintHTML(String html, PrintStream outStream) // this method is used to show the given HTML as the output this has length of content and type of content

{

outStream.println("HTTP/1.1 200 OK"); // sends the ok reply to browser and says the webBrowser that the given req. was processed

outStream.println("Content-Length: " + html.length()); // sends the length of content to the mentioned browser and shows the output

outStream.println("Content-Type: text/html" + "\r\n\r\n"); // sends the type of content to the mentioned browser and shows the output

outStream.println(html); // prints the html to the output mentioned browser

}

}