Code for proxy server

import java.io.\* ;

import java.net.\* ;

import java.util.\*;

class HttpRequestRunnable implements Runnable{

protected Socket clientSocket;

public HttpRequestRunnable(Socket clientSocket) {

this.clientSocket = clientSocket;

final String CRLF = "\r\n";

}//each HTTP request must end with a carriage return and linefeed,

// this is for convenience

public void run() {

try {

processRequest();

} catch (Exception e) {

System.*out*.println(e);

}

} // Implement the run() method of the Runnable interface.

//sets up server socket, calls the processRequest method

private void processRequest() throws Exception {

// Get a reference to the socket's input and output streams.

//get the input, pass to the buffered reader

clientSocket.getInputStream();

//new InputStreamReader();

BufferedReader input = new BufferedReader(new InputStreamReader(clientSocket.getInputStream()));

PrintWriter output = new PrintWriter(clientSocket.getOutputStream(), true);

/\*//test code - I did get a response from the client...- but it wasn't the expected

//kept getting: Client says: GET / HTTP/1.1

String arg1;

arg1 = input.readLine();

System.out.println("Client says: " + arg1); //Note from the client to the server (received)

output.println("The message was received. Thank you!"); //note from the server to the client (reply)

\*/

//Get the request line of the HTTP request message.

//Wasn't sure how to do this, looked up different ways to make HTTP requests in Java, this was a sync.

// request which fits TCP (Java 1.1 HttpURLConnection)

URL url = new URL("http://host.someschool.edu:8888/");

HttpURLConnection connection = (HttpURLConnection) url.openConnection();

connection.setRequestProperty("accept", "application/json");

InputStream responseStream = connection.getInputStream();

ObjectMapper mapper = new ObjectMapper();

APOD apod = mapper.readValue(responseStream, APOD.class);

String requestLine = (String apod);

//prints out the URL that was created after being established

//however this was not working

System.*out*.println("apod.title");

// Extract the filename from the request line.

StringTokenizer tokens = new StringTokenizer(requestLine) ; tokens.nextToken() ;

// skip over the method, which should be "GET"

String fileName = tokens.nextToken() ;

// Prepend a "." so that file request is within the current directory.

fileName = "." + fileName ;

//Open the requested file.

FileinputStream fis = null;

boolean fileExists = true;

try {

fis = new FileinputStream(fileName) ; } catch (FileNotFoundException e) {

fileExists = false ;

}

//Construct the response message.

String statusLine = null;

String contentTypeLine = null;

String entityBody = null;

if ( fileExists) {

statusLine = ?;

contentTypeLine = "Content-type: " + contentType(fileName) + "\r\n";

}

else{

statusLine = ? ;

contentTypeLine = ? ;

entityBody = "<HTML>" +

"<HEAD><TITLE>Not Found</TITLE></HEAD>" + "<BODY>Not Found</BODY></HTML>";

}

// Send the status line.

output.writeBytes(statusLine) ;

//Send the content type line

os.writeBytes(1024) ;

//Send a blank line to indicate the end of the header lines.

os.writeBytes("\r\n") ;

//Send the entity body.

if ( fileExists) {

*sendBytes*(fis, output) ;

fis.close() ;

} else {

output.writeBytes( 1024 ) ;

}

// Display the request line.

System.*out*.println() ;

System.*out*.println(requestLine.body().get().title);//print out the HTTP result

//get and display the header lines

String headerLines = null;

while( (headerLines = input.readLine()).length() != 0) {

System.*out*.println(headerLines) ;

}

// Close streams and socket.

output.close() ;

input.close() ;

clientSocket.close() ;

}

private static void sendBytes(FileinputStream fis, OutputStream os) throws Exception{

// Construct a lK buffer to hold bytes on their way to the socket.

byte[] buffer = new byte[1024] ;

int bytes = O;

//Copy requested file into the socket's output stream.

while((bytes = fis.read(buffer) ) != -1 ) {

os.write(buffer, O, bytes) ;

}

private class APOD {

//imported for the HTML request, from

//https://github.com/mjg123/java-http-clients/blob/master/src/main/java/com/twilio/APOD.java

//linked from the how to on this page:

//https://www.twilio.com/blog/5-ways-to-make-http-requests-in-java

public final String copyright;

public final String date;

public final String explanation;

public final String hdUrl;

public final String mediaType;

public final String serviceVersion;

public final String title;

public final String url;

public APOD(@JsonProperty("copyright") String copyright,

@JsonProperty("date") String date,

@JsonProperty("explanation") String explanation,

@JsonProperty("hdurl") String hdUrl,

@JsonProperty("media\_type") String mediaType,

@JsonProperty("service\_version") String serviceVersion,

@JsonProperty("title") String title,

@JsonProperty("url") String url) {

this.copyright = copyright;

this.date = date;

this.explanation = explanation;

this.hdUrl = hdUrl;

this.mediaType = mediaType;

this.serviceVersion = serviceVersion;

this.title = title;

this.url = url;

}

}

}

*/\*\**

*\*HttpRequestRunnable.java*

*\*****@author:*** *Breeda Gonzalez*

*\*This program is a web server created using the programing instructions from the assignment 1*

*\* This checks for TCP connections, creates the TCP the connection, services the request,*

*\* and then closes the TCP connection.*

*\*/*

import java.io.\* ;

import java.net.\* ;

import java.util.\*;

public cclass Web\_Server {

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

int portNumber = 8888; //set the port number

//Create socket for listening

ServerSocket serverSocket = new ServerSocket(portNumber);

//start listening for the TCP connections using while Loop

while (true) { //if there is no connection, listen infinitely

try {

Socket clientSocket = serverSocket.accept();

//new thread grabs the client server request

new Thread(new HttpRequestRunnable(clientSocket)).start();

} catch (IOException exception) {

System.out.println(exception.getMessage());

}

}

}

}