Electrcial and computer Engineering

ENCS5311, Distributed Systems and Algorithms

**Project #1 Due: 28/2/2018**

**Web server and homepage**

In this project a front-end and back-end parts of a multi-threaded webserver should be implemented.

In the front-end part you have to use HTML (and may other tools such as CSS, java script) to implement a personal web page for your group. The web page should contain pictures, links and should look nice.

The server should be able to deliver your homepage to the browser. It should able to deal with html, txt, jpg, jpeg, gif, andpngfiles. Version 1.0 of HTTP, as defined in RFC 1945 should be implemented.To implement the sever you have to complete the attached code.Every time, you see a **“?”**, a missing detail must be added. You have to explain what you add.

When an http request arrives, the server prints on the screen this request, port number at browser side and source IP. You can use the getInetAddress() and getPort() methods to get the source address and port.

<http://www.ietf.org/rfc/rfc1945.txt>

**Submit a single file on ritaj that contains:**

1- The code for the webpage and for the server

2-Screenshots of accessing the webpage from a computer (localhost:8899)

3-Screenshots of accessing the webpage from a mobile phone (ip\_address \_of\_the\_computer:8899)

4-Screenshot of an http request

5-Source address and port of a packet.

**Bonus:**

1. The sever should provide a different webpage for the same content if it is called from a smartphone.
2. Add security

**The general structure of the program:**

import java.io.\* ;

import java.net.\* ;

import java.util.\* ;

public final class WebServer

{

public static void main(String argv[]) throws Exception

{

...

}

}

final class HttpRequest implements Runnable

{

...

}

}

**Normally, http protocol uses port 80, but we want to use port 8899**

public static void main(String argv[]) throws Exception

{

// set the port number:

int port = 8899;

...

}

**Then open a socket and wait for a connection**

// establish the listen socket:

?

// process HTTP service requests in an infinite loop:

while (true)

{

// listen for a TCP connection request:

?

...

}

**Then for multi-threaded**

// construct an object to process the HTTP request message:

HttpRequest request = new HttpRequest(?);

// create a new thread to process the request:

Thread thread = new Thread(request);

// start the thread:

thread.start();

**Then you have to implement *HttpRequest***

*final class HttpRequest implements Runnable*

*{*

*final String CRLF = "\r\n";*

*Socket socket;*

*// constructor:*

*public HttpRequest(Socket socket) throws Exception*

*{*

*this.socket = socket;*

*}*

*// implement the run() method of the Runnable interface:*

*public void run()*

*{*

*...*

*}*

*private void processRequest() throws Exception*

*{*

*...*

*}*

*}*

***Run() can be implemented as***

*// implement the run() method of the Runnable interface:*

*public void run()*

*{*

*try*

*{*

*processRequest();*

*}*

*catch (Exception e)*

*{*

*System.out.println(e);*

*}*

*}*

Then complete processRequest()

*private void processRequest() throws Exception*

*{*

*// get a reference to the socket’s input and output streams:*

*InputStream is = ?;*

*DataOutputStreamos= ?;*

*// Set up input stream filters.*

*?*

*BufferedReaderbr= ?;*

*...*

*}*

To read the request from the browser:

*// get the request line of the HTTP request message:*

*String requestLine= ?;*

*// display the request line:*

*System.out.println();*

*System.out.println(requestLine);*

*// get and display the header lines:*

*String headerLine = null;*

*while ((headerLine = br.readLine()).length() != 0)*

*{*

*System.out.println(headerLine);*

*}*

To close the sockets you can use

*// Close streams and socket.*

*os.close();*

*br.close();*

*socket.close();*

The file name should be extracted using StringTokenizer class.

// 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;

Then open the requested file

// open the requested file:

FileInputStreamfis = null;

booleanfileExists = true;

try

{

fis = new FileInputStream(fileName);

}

catch (FileNotFoundException e)

{

fileExists = false;

}

Create a response such that if the file doesn’t exist it sends back 404 Not Found

Otherwise it send back the file:

// construct the response message:

String statusLine = null;

String contentTypeLine = null;

String entityBody = null;

if (fileExists)

{

statusLine= ?;

contentTypeLine = "Content-Type: " +

contentType( fileName ) + CRLF;

}

else

{

statusLine= ?;

contentTypeLine= ?;

entityBody = "<html>" +

"<head><title>Not Found</title></head>" +

"<body>Not Found</body></html>";

}

**Then send the header lines**

// send the status line:

os.writeBytes(statusLine);

// send the content type line:

os.writeBytes(?);

// send a blank line to indicate the end of the header lines:

os.writeBytes(CRLF);

now we send the body

// send the entity body:

if (fileExists)

{

sendBytes(fis, os);

fis.close();

}

else

{

os.writeBytes(?);

}

To send a file use

private static void sendBytes(FileInputStreamfis, OutputStreamos) throws Exception

{

// construct a 1K buffer to hold bytes on their way to the socket:

byte[] buffer = new byte[1024];

int bytes = 0;

// copy requested file into the socket’s output stream:

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

{

os.write(buffer, 0, bytes);

}

}

**You have to specify the content type**

private static String contentType(String fileName)

{

if(fileName.endsWith(".htm") || fileName.endsWith(".html"))

{

return "text/html";

}

if(?)

{

?;

}

if(?)

{

?;

}

return "application/octet-stream";

}