CS 471/571

Java Sockets 2
Web Server

- Implement a very simple Web Server that can process GET requests
- Java
- C
HTTP request message: general format

- **Method**: The first line of the request message specifies the type of request. Possible values are GET, POST, PUT, DELETE, etc.
- **URL**: The request is directed at a specific resource identified by a URL.
- **Version**: The protocol version being used for the request.
- **Header Fields**: A list of header fields, each consisting of a field name and a value, separated by carriage return and line feed (\r\n).
- **Entity Body**: Contains the data sent to the server as part of the request. It is located at the bottom of the message and is preceded by two carriage returns and line feeds (\r\n\r\n).

The diagram illustrates the structure of an HTTP request message, showing how each component is organized and separated by carriage returns and line feeds.
HTTP response message: general format

```
version    sp  Status code    sp  phrase   cr  lf
header field name    sp  value   cr  lf
header field name    sp  value   cr  lf
cr  lf
entity body
```

status line

header lines

body
Simple Web Server in Java (from previous version of Kurose and Ross)

```java
import java.io.*;
import java.net.*;
import java.util.*;

public final class WebServer {
    public static void main(String argv[]) throws Exception {
        // Get the port number from the command line.
        int port = (new Integer(argv[0])).intValue();

        // Establish the listen socket.
        ServerSocket socket = new ServerSocket(port);

        // Process HTTP service requests in an infinite loop.
    }
}
```
Simple Web Server (Java)

```java
while (true) {
    // Listen for a TCP connection request.
    Socket connection = socket.accept();

    // Construct an object to process the HTTP request message.
    HttpRequest request = new HttpRequest(connection);

    // Create a new thread to process the request.
    Thread thread = new Thread(request);

    // Start the thread.
    thread.start();
}
```
Simple Web Server (Java)

```java
import java.io.*;
import java.net.*;
import java.util.*;

final class HttpRequest implements Runnable {
    final static String CRLF = "\r\n";
    Socket socket;

    public HttpRequest(Socket socket) throws Exception {
        this.socket = socket;
    }

    public void run() {
        try {
            processRequest();
        } catch (Exception e) {
            System.out.println(e);
        }
```
Simple Web Server (Java)

private void processRequest() throws Exception {
    // Get a reference to the socket's input and output streams.
    InputStream is = socket.getInputStream();
    DataOutputStream os = new DataOutputStream(socket.getOutputStream());

    // Set up input stream filters.
    BufferedReader br = new BufferedReader(new InputStreamReader(is));

    // Get the request line of the HTTP request message.
    String requestLine = br.readLine();

    // 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 = "HTTP/1.0 200 OK" + CRLF;
    contentTypeLine = "Content-Type: " +
        contentType(fileName) + CRLF;
} else {
    statusLine = "HTTP/1.0 404 Not Found" + CRLF;
    contentTypeLine = "Content-Type: text/html" + CRLF;
    entityBody = "<HTML>"
        "<HEAD><TITLE>Not Found</TITLE></HEAD>" +
        "<BODY>Not Found</BODY></HTML>";
}

// Send the status line.
os.writeBytes(statusLine);
// Send the content type line.
os.writeBytes(contentTypeLine);
// Send a blank line to indicate the end of the header lines.
os.writeBytes(CRLF);
Simple Web Server (Java)

// Send the entity body.
if (fileExists) {
    sendBytes(fis, os);
    fis.close();
} else {
    os.writeBytes(entityBody) ;
}

// Close streams and socket.
os.close();
br.close();
socket.close();
private static void sendBytes(FileInputStream fis, OutputStream os) 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);
    }
}
private static String contentType(String fileName) {
    if (fileName.endsWith(".htm") || fileName.endsWith(".html")) {
        return "text/html";
    }
    if (fileName.endsWith(".ram") || fileName.endsWith(".ra")) {
        return "audio/x-pn-realaudio";
    }
    return "application/octet-stream";
}