

Project #3 : Web Server Programming

이름 : 여주안

학번 : 2017027265

과제 설명 – 내용 및 목적 서술

The project implements multi-thread Web Server. While the server is running, the client(browser) can access to server resources by IP address and port number. The goal of this project is to learn how to build Web Server Application which can process multiple requests at the same time and test it on the browser by using port forwarding.

source Files : 본인이 작성한 소스 파일을 캡처 및 중요하고 필요하다고 생각하는 부분을 서술

<WebServer.java>

The server makes a new thread when it gets a new request. It can process multiple requests at the same time.

```
package project1;

import java.io.IOException;

public class WebServer {

    public static void main(String[] args) {
        try {
            ServerSocket serverSocket = new ServerSocket(7010);

            while (true) {
                Socket connectionSocket = serverSocket.accept();

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

                // Create a new thread to process the request
                Thread thread = new Thread(request);
                thread.start();
            }
        } catch (IOException e) {
            System.out.println(e.getMessage());
        } catch (Exception e) {
            System.out.println(e.getMessage());
        }
    }
}
```

<HttpRequest.java>

HttpRequest.java implements Runnable interface. The run() method is called by web server thread.

processRequest method reads request line and send response message. It calls parseRequestLine method to parse the request message and sendResponseMessage method to send response message. parseRequestLine method uses StringTokenizer class to parse request message. sendResponseMessage method sets status message and writes entityBody.

```
package project1;

import java.io.BufferedReader;

enum StatusCode {
    OK, BAD_REQUEST, FORBIDDEN, NOT_FOUND, INTERNAL_SERVER_ERROR, HTTP_VERSION_NOT_SUPPORTED
}

public class HttpRequest implements Runnable {
    final static String CRLF = "\r\n";
    final static String HTTP_VERSION = "1.1";
    final static String DEFAULT_CONTENT_TYPE = "application/octet-stream";

    final static int BUFFER_IN_SIZE = 2048;
    final static int BUFFER_OUT_SIZE = 2048;
    final static Properties CONTENT_TYPES = new Properties();
    final static EnumMap<StatusCode, String> SCODES = new EnumMap<StatusCode, String>(StatusCode.class);

    static {
        CONTENT_TYPES.setProperty("html", "text/html");
        CONTENT_TYPES.setProperty("jpg", "image/jpeg");

        SCODES.put(StatusCode.OK, "200");
        SCODES.put(StatusCode.BAD_REQUEST, "400");
        SCODES.put(StatusCode.FORBIDDEN, "403");
        SCODES.put(StatusCode.NOT_FOUND, "404");
        SCODES.put(StatusCode.INTERNAL_SERVER_ERROR, "500");
        SCODES.put(StatusCode.HTTP_VERSION_NOT_SUPPORTED, "505");
    }

    StatusCode code;
    Socket socket;
    File requestedFile;

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

    @Override
    public void run() {
        // process request
        try {
            processRequest();
        } catch (Exception e) {
            System.out.println("Exception occurred while processing request : ");
            e.printStackTrace();
        }
    }

    private void processRequest() throws Exception {
        InputStream is = null;
        DataOutputStream os = null;
        FileInputStream fis = null;
        BufferedReader br = null;

        try {
            is = socket.getInputStream();
            os = new DataOutputStream(socket.getOutputStream());
            br = new BufferedReader(new InputStreamReader(is), BUFFER_IN_SIZE);

            String requestLine = br.readLine();
            String errorMsg = parseRequestLine(requestLine);

            String headerLine = null;
            while ((headerLine = br.readLine()).length() != 0) {
                System.out.println(headerLine);
            }
        }
    }
}
```

```

        if (errorMsg == null) {
            try {
                fis = new FileInputStream(requestedFile);
            } catch (FileNotFoundException e) {
                System.out.println("!! FileNotFoundException");
                e.printStackTrace();
                code = StatusCode.NOT_FOUND;
            }
        } else {
            System.out.println();
            System.out.println(errorMsg);
        }
        sendResponseMessage(fis, os);
    } finally {
        // close streams and socket (HTTP/1.0)
        if (os != null) {
            os.close();
        }
        if (br != null) {
            br.close();
        }
        if (fis != null) {
            fis.close();
        }
        socket.close();
    }
}

private String parseRequestLine(String requestLine) {
    System.out.println();
    System.out.println("Received HTTP request: ");
    System.out.println(requestLine);

    StringTokenizer tokens = new StringTokenizer(requestLine);
    if (tokens.countTokens() != 3) {
        code = StatusCode.NOT_FOUND;
        return "Request line is malformed, Returning BAD NOT FOUND.";
    }

    String method = tokens.nextToken().toUpperCase();
    String fileName = tokens.nextToken();
    fileName = "." + fileName;
    File file = new File(fileName);

    if (!file.exists()) {
        code = StatusCode.NOT_FOUND;
        return "Request file " + fileName + " does not exist. Returning NOT FOUND.";
    }

    if (!file.canRead()) {
        code = StatusCode.FORBIDDEN;
        return "Request file " + fileName + " is not readable. Returning FORBIDDEN.";
    }

    if (file.isDirectory()) {
        File[] list = file.listFiles(new FilenameFilter() {
            @Override
            public boolean accept(File dir, String name) {
                if (name.equalsIgnoreCase("index.html"))
                    return true;
                return false;
            }
        });

        if (list == null || list.length == 0) {
            code = StatusCode.NOT_FOUND;
            return "No index file found at requested location " + fileName + ". Returning NOT FOUND.";
        } else if (list.length != 1) {
            code = StatusCode.INTERNAL_SERVER_ERROR;
            return "Found more than one index file at requested location " + fileName
                + ". Returning INTERNAL SERVER ERROR.";
        }

        // index file
        file = list[0];
    }
}

```

```

        requestedFile = file;
        // extract HTTP version from the request line
        String version = tokens.nextToken().toUpperCase();
        if(version.equals("HTTP/1.0")) {
            code = StatusCode.BAD_REQUEST;
            return "HTTP version string is malformed. Returning BAD REQUEST.";
        }
        if(!version.matches("HTTP/([1-9][0-9.]*)")) {
            code = StatusCode.BAD_REQUEST;
            return "HTTP version string is malformed. Returning BAD REQUEST.";
        }
        if(!version.equals("HTTP/1.0") && !version.equals("HTTP/1.1")) {
            code = StatusCode.HTTP_VERSION_NOT_SUPPORTED;
            return version + " not supported. Returning HTTP VERSION NOT SUPPORTED.";
        }

        code = StatusCode.OK;
        return null;
    }

    private void sendResponseMessage(FileInputStream fis, DataOutputStream os) throws Exception {
        String statusLine = "HTTP/" + HTTP_VERSION + " " + SCODES.get(code) + " ";
        String entityBody = "<HTML>" + CRLF + " <HEAD><TITLE>?</TITLE></HEAD>" + CRLF + " <BODY>?</BODY>" + CRLF;

        // construct message string
        String message;
        switch (code) {
            case OK:
                message = "OK";
                break;
            case BAD_REQUEST:
                message = "Bad Request";
                break;
            case FORBIDDEN:
                message = "Forbidden";
                break;
            case NOT_FOUND:
                message = "Not Found";
                break;
            case HTTP_VERSION_NOT_SUPPORTED:
                message = "HTTP Version Not Supported";
                break;
            default:
                message = "Undefined";
        }

        statusLine = statusLine + message;
        if (code != StatusCode.OK) {
            entityBody = entityBody.replaceAll("\\?", message + " - sent by Juan's WebServer");
        }

        System.out.println("statusLine: " + statusLine);
        //System.out.println("entityBody: " + CRLF + entityBody);
        // send the status line
        os.writeBytes(statusLine + CRLF);
        // construct and send the header lines
        sendHeaderLines(os);
        os.writeBytes(CRLF);

        if (code == StatusCode.OK) {
            System.out.println("Sending requested file to client...");
            sendBytes(fis, os);
        } else {
            System.out.println("Sending error message to client...");
            os.writeBytes(entityBody);
        }
    }

    private void sendHeaderLines(DataOutputStream os) throws Exception {
        StringBuffer headerLines = new StringBuffer();
        String contentTypeLine = "Content-type: ";
        System.out.println("code: " + code);

        switch (code) {
            case OK:
                contentTypeLine += contentType(requestedFile.getName()) + CRLF;
                //contentTypeLine += "" + CRLF;
                contentTypeLine += "Content-Length: 1024" + CRLF;
                //contentTypeLine += "Content-length: " + Long.toString(requestedFile.length()) + CRLF;
                break;
        }
    }

```

```

        default:
            contentTypeLine += "text/html" + CRLF;
    }

    headerLines.append(contentTypeLine + CRLF);
    os.writeBytes(headerLines.toString());
}

private void sendBytes(FileInputStream fis, OutputStream os) throws Exception {
    // construct 1k buffer to hold bytes on their way to the socket
    byte[] buffer = new byte[BUFFER_OUT_SIZE];
    int bytes = 0;
    // copy requested file into the socket's output stream
    while ((bytes = fis.read(buffer)) != -1) {
        os.write(buffer, 0, bytes);
    }
}

private String contentType(String fileName) {
    String fname = fileName.toLowerCase();
    int lastdot = fname.lastIndexOf(".");
    if((lastdot != -1) && (lastdot != fname.length() - 1)) {
        return CONTENT_TYPES.getProperty(fname.substring(lastdot+1), DEFAULT_CONTENT_TYPE);
    }
    return DEFAULT_CONTENT_TYPE;
}
}

```

sendHeaderLines method writes header line of the response(Content-Type and Content-Length). It calls contentType method to add content type property. When the server successfully found the resource, it calls sendBytes method to send file data.

<index.html>

The file length should be 1024. The browser checks if the actual file length matches content-length property.

```

<html>
<body>
    KYH SERVER TEST PAGE 1234567890ABC<br>
    Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nam bibendum mi suscipit, mattis metus sit amet, e
    Nunc velit elit, egestas non posuere a, hendrerit non enim. Maecenas sed tortor erat. Sed non nulla sapien.
</body>
</html>

```

Instructions: 본인의 소스 실행 방법 (본인이 실행 시킨 방법)

I used Eclipse IDE to write and execute source code.

First, (compile and) run the webServer.java file. Check if it prints 200 OK response when you contact the server on localhost.

Next, set port forwarding and use the same port number when you create server socket. Test the web server by using IP address and port number.

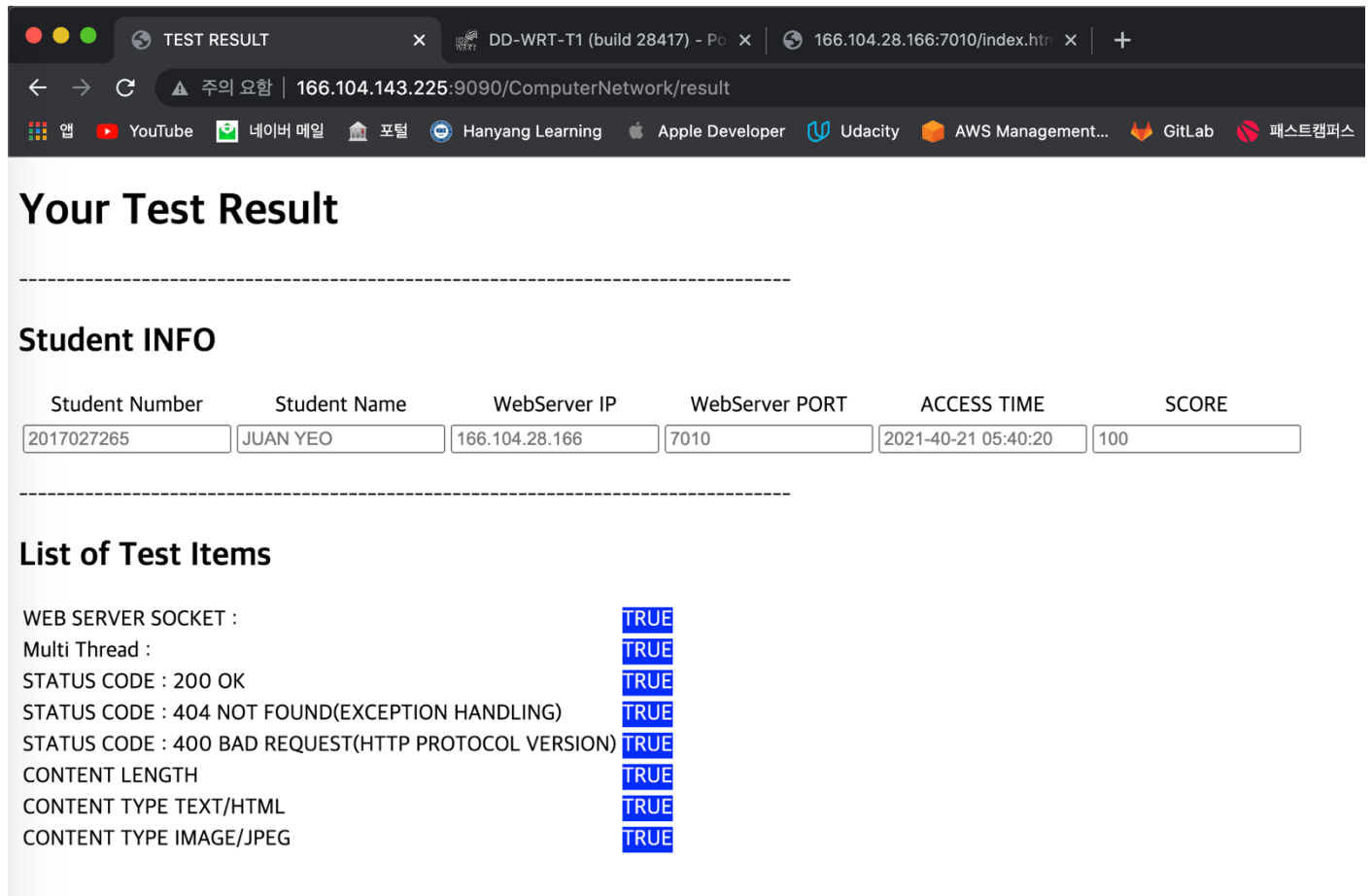
Finally, start the wireshark program and check the packet transport history. (query HTTP)

How the program works: 프로그램 구동 방식

이 프로그램은 웹 서버에서 각각의 요청마다 쓰레드를 생성해서 여러개의 요청을 한번에 처리할 수 있도록 한다. 요청을 읽고 응답을 보내는 프로그램은 Runnable 클래스로 구현한다. 우선 요청의 URL 주소를 추출한 뒤, 해당 위치에 파일이 있는지 확인한다. 요청 양식과 파일 존재 여부에 따라 상태 메시지를 작성한다. 파일이 있다면 파일 형식에 알맞은 Content-Type 값을 찾고 Content-Length를 1024로 설정하여 헤더를 작성한다. 마지막으로 헤더와 파일 데이터를 합친 응답 메시지를 작성하여 브라우저로 응답을 보내는 방식이다.

프로그램을 실행할 때는 WebServer.java를 실행한 뒤, 브라우저에서 IP 주소와 포트 번호를 입력하여 연결한다. 포트포워딩이 완료되면 localhost 대신 IP 주소로 연결할 수 있다.

Results: 결과 화면 캡처 및 설명.



Your Test Result

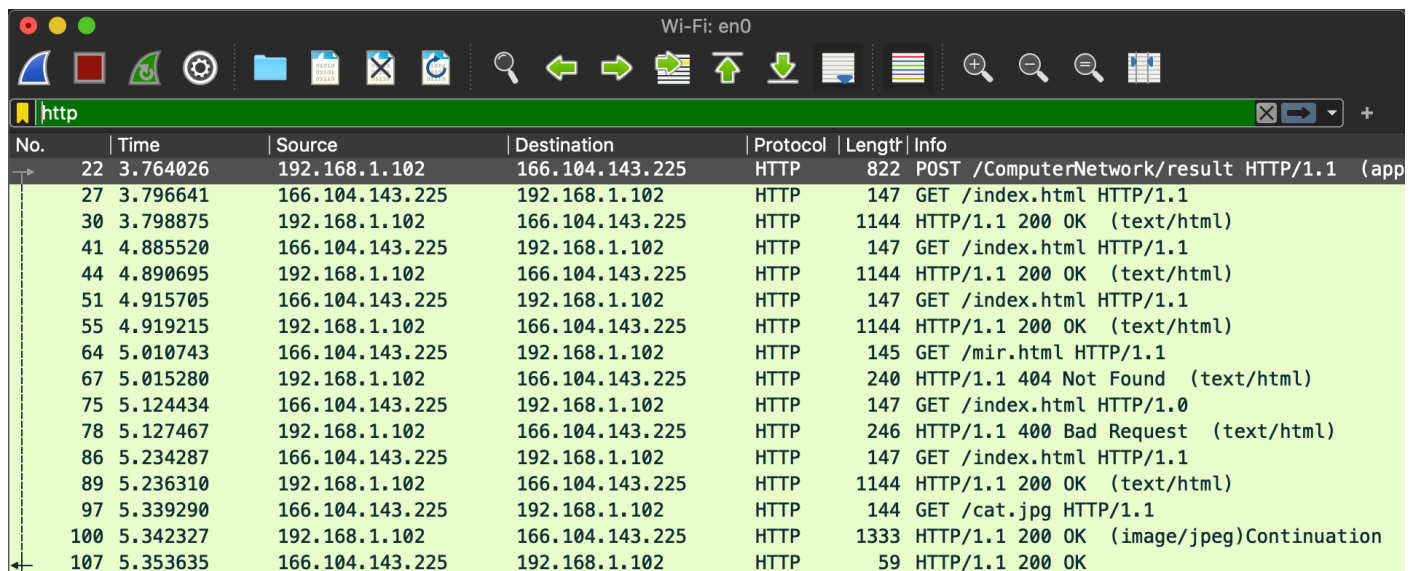
Student INFO

Student Number	Student Name	WebServer IP	WebServer PORT	ACCESS TIME	SCORE
2017027265	JUAN YEO	166.104.28.166	7010	2021-40-21 05:40:20	100

List of Test Items

WEB SERVER SOCKET :	TRUE
Multi Thread :	TRUE
STATUS CODE : 200 OK	TRUE
STATUS CODE : 404 NOT FOUND(EXCEPTION HANDLING)	TRUE
STATUS CODE : 400 BAD REQUEST(HTTP PROTOCOL VERSION)	TRUE
CONTENT LENGTH	TRUE
CONTENT TYPE TEXT/HTML	TRUE
CONTENT TYPE IMAGE/JPEG	TRUE

모든 테스트를 정상적으로 통과하였다. IP 주소는 166.104.28.166, 포트포워딩 된 포트 번호는 7010임을 확인할 수 있다. 응답 상태는 200, 404, 400 경우를 테스트했고 Content Length가 1024로 설정되어 있는지, 파일에 따라 알맞은 Content Type이 설정되어 있는지 테스트했다.



No.	Time	Source	Destination	Protocol	Length	Info
22	3.764026	192.168.1.102	166.104.143.225	HTTP	822	POST /ComputerNetwork/result HTTP/1.1 (app
27	3.796641	166.104.143.225	192.168.1.102	HTTP	147	GET /index.html HTTP/1.1
30	3.798875	192.168.1.102	166.104.143.225	HTTP	1144	HTTP/1.1 200 OK (text/html)
41	4.885520	166.104.143.225	192.168.1.102	HTTP	147	GET /index.html HTTP/1.1
44	4.890695	192.168.1.102	166.104.143.225	HTTP	1144	HTTP/1.1 200 OK (text/html)
51	4.915705	166.104.143.225	192.168.1.102	HTTP	147	GET /index.html HTTP/1.1
55	4.919215	192.168.1.102	166.104.143.225	HTTP	1144	HTTP/1.1 200 OK (text/html)
64	5.010743	166.104.143.225	192.168.1.102	HTTP	145	GET /mir.html HTTP/1.1
67	5.015280	192.168.1.102	166.104.143.225	HTTP	240	HTTP/1.1 404 Not Found (text/html)
75	5.124434	166.104.143.225	192.168.1.102	HTTP	147	GET /index.html HTTP/1.0
78	5.127467	192.168.1.102	166.104.143.225	HTTP	246	HTTP/1.1 400 Bad Request (text/html)
86	5.234287	166.104.143.225	192.168.1.102	HTTP	147	GET /index.html HTTP/1.1
89	5.236310	192.168.1.102	166.104.143.225	HTTP	1144	HTTP/1.1 200 OK (text/html)
97	5.339290	166.104.143.225	192.168.1.102	HTTP	144	GET /cat.jpg HTTP/1.1
100	5.342327	192.168.1.102	166.104.143.225	HTTP	1333	HTTP/1.1 200 OK (image/jpeg)Continuation
107	5.353635	166.104.143.225	192.168.1.102	HTTP	59	HTTP/1.1 200 OK

Wireshark를 통해 확인한 HTTP 패킷 목록이다. 테스트를 위해 요청과 응답이 여러 번 오고 간 것을 확인할 수 있었다.

something Else: 과제에 대해서 건의점, 조교에게 전달되는 점, 질문, 앞으로의 과제에서의 희망사항 등

최대한 영어로 쓰려고 노력해봤는데 프로그램 구동 방식과 결과에 대해서는 정확히 전달하고 싶어서 한국어로 작성하였습니다. 웹 서버를 직접 만들어보니 과제지만 재미있었습니다. 연구실에 찾아갔을 때, 포트포워딩과 테스트를 도와주셔서 감사합니다 :)