24-여름학기 컴퓨터네트워크 과제 1

README

소프트웨어학부

20203080 선현승

1. **구현내용**
   - A. **소켓 모듈을 사용하여 Client-Server 구조로 HTTP 통신 구현**
   - B. **HEAD, GET, POST, DELETE , PUT에 대한 응답 처리**
     - i. **HEAD : 해당 Client에 Cookie 설정**
     - ii. **GET : "Hello! World!" 메시지 응답**
     - iii. **POST : 해당 Client의 id로 Body를 내용으로 하는 txt파일 저장**
     - iv. **DELETE : 해당 Client의 id를 이름으로 하는 txt파일 삭제**
   - C. **HTTP예외 처리 클래스 작성**
   - D. **HTTP응답을 처리하는 Parser 클래스 작성**
   - E. **WireShark로 해당 HTTP통신 캡쳐**
   - F. **Putty를 사용하여 Server와 직접 통신 캡처**

2. **기술 스택**
   - A. **사용언어 : java 21 2023-09-19 LTS**
     Java(TM) SE Runtime Environment (build 21+35-LTS-2513)
     Java HotSpot(TM) 64-Bit Server VM (build 21+35-LTS-2513, mixed mode, sharing)
   - B. **개발환경 :**
     - i. **OS : Window11**
     - ii. **IDE : VSCode**

3. **실행방법**

Github : Goldbori/ComputerNetworking (github.com)

$ git clone https://github.com/Goldbori/ComputerNetworking

으로 폴더를 내려받은 후 VSCode를 사용하여 HttpServer.java 파일을 실행시킬 수 있다.

4. 구성 파일
   A. /Server
      i. src/StatusCode.txt
      ii. BodyParser.java
      iii. ClientHandler.java
      iv. HeaderParser.java
      v. HTTPException.java
      vi. HttpServer.java
      vii. ResponseHandler.java

   B. /Client
      i. ClientHeaderParser.java
      ii. HTTPClient.java

5. 코드 설명

5.1 src/StatusCode.txt

참고자료 : https://developer.mozilla.org/ko/docs/Web/HTTP/Status

를 활용하여 HTTP 메시지의 상태 코드, 메시지를 입력해둔 텍스트 파일입니다.

HTTPException 객체가 생성될 때 파일을 읽어 각 코드별 상태를 저장합니다.

5.2 BodyParser.java

```java
Server > BodyParser.java > ...
You, 3시간 전 | 1 author (You)
package Server;

import java.io.BufferedReader;
import java.io.IOException;

You, 3시간 전 | 1 author (You)
public class BodyParser {

    public void BodyHandler(BufferedReader br, StringBuilder sb, int len) throws IOException {
        char[] bodyChars = new char[len];
                int read = br.read(bodyChars, 0, len);
                sb.append(bodyChars, 0, read);
    }
}

```

Client와 Server가 수신한 응답의 Body를 Content-Length값을 이용하여 모두 읽은 후 StringBuilder sb에 추가합니다. 이전에는 while문을 사용하여 EOL개념으로 접근하였는데 다음 요청을 처리할 때 null값을 읽는 오류가 발생하여 Content-Length를 사용하는 방식으로 변경하였습니다.

5.3 ClientHandler.java

```java
Server > ClientHandler.java > ClientHandler > sendHTTPResponse(BufferedWriter, String, String, HashMap<String, String>, StringBuilder)
You, 48분 전 | 1 author (You)
package Server;

import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.OutputStreamWriter;
import java.net.Socket;
import java.util.HashMap;

You, 48분 전 | 1 author (You)
public class ClientHandler implements Runnable {
    Socket client;
    HeaderParser hP;
    BodyParser bP;

    ClientHandler(Socket client) {
        this.client = client;
        this.hP = new HeaderParser();
        this.bP = new BodyParser();
    }

    void receiveHTTPRequest(BufferedReader br, String[] MPV, HashMap<String, String> headerMap, StringBuilder sb) throws IOException, HTTPException {
        headerMap.clear(); // 헤더 초기화
        hP.HeaderHandler(br, MPV, headerMap);
        if (headerMap.containsKey("Content-Length")) { // Body가 있는 경우
            bP.BodyHandler(br, sb, Integer.parseInt(headerMap.get("Content-Length")));  // Body 처리 완료
        }
    }
    void sendHTTPResponse(BufferedWriter bw, String method, String path, HashMap<String,String> headerMap, StringBuilder sb) throws IOException, HTTPException{
        new ResponseHandler(method, path, bw, headerMap, sb);       You, 48분 전 • 404req, 505req, 요청-응답 성공
        sb.setLength(0);
    }
    void sendExceptionResponse(BufferedWriter bw, int code) throws IOException{
        new ResponseHandler(code, bw);
    }
```

<ClientHandler.java 1~35>

Client 소켓의 다중연결을 지원하기 위한 부분입니다. Runnable 인터페이스를 구현하여 소켓이 연결될 때마다 각 Thread를 실행시켜 처리하도록 합니다.

receiveHTTPRequest, sendHTTPResponse 함수를 이용하여 HTTP메시지를 읽고 처리하여 응

답을 보낼수 있습니다.

receiveHTTPRequst 함수는 헤더를 저장하는 headerMap을 초기화하고 헤더를 읽은 뒤 "Content-Length" 헤더의 값에 따라서 Body를 읽어 냅니다.

sendHTTPResponse 는 ResponseHandler객체를 통해 메서드, 경로에 따른 응답을 처리할 수 있도록 합니다.

sendExceptionResponse 함수를 이용하여 HTTPException 발생시 해당 상태코드를 넣어서 응답을 보낼 수 있습니다.

```
    @Override
    public void run() {
        try (BufferedReader br = new BufferedReader(new InputStreamReader(client.getInputStream()));
             BufferedWriter bw = new BufferedWriter(new OutputStreamWriter(client.getOutputStream()))) {

            System.out.println("Socket connected from " + client.getInetAddress()); // 연결된 소켓의 IP address 출력
            //System.out.println("Socket closed: " + client.isClosed());
            //System.out.println("Socket connected: " + client.isConnected());

            HashMap<String, String> headerMap = new HashMap<>(); // 헤더 종류: 헤더값 저장
            String[] methodPathVersion = new String[3]; // 0: 메서드, 1: 경로, 2: HTTP 버전
            StringBuilder sb = new StringBuilder(); // Body 저장할 StringBuilder
            while (client.isConnected() && !client.isClosed()){
                try{
                    receiveHTTPRequest(br, methodPathVersion, headerMap, sb);

                }catch(HTTPException e){
                    sendExceptionResponse(bw, e.status);
                    return;
                }
                sendHTTPResponse(bw, methodPathVersion[0], methodPathVersion[1], headerMap, sb);
            }

        } catch (HTTPException e) {
            System.err.println("HTTPException: " + e.getMessage());
        } catch (IOException e) {
            System.err.println("IOException: " + e.getMessage());
        } finally {
            try {
                client.close();
                System.out.println("--------------------------------");
                System.out.println("Client disconnected from " + client.getInetAddress());
            } catch (IOException e) {
                System.err.println("Failed to close client socket: " + e.getMessage());
            }
        }
    }
}
```

<ClientHandler.java 36~73>

Runnable 인터페이스를 구현하기 위해서 run()을 오버라이딩하여 구현하였습니다.

BufferedReader, BufferedWriter를 사용하여 입출력 스트림을 연결하고 처리합니다.

변수로는 헤더정보를 저장하는 headerMap, 메서드, 경로, HTTP버전을 저장하는

methodPathVersion String배열 그리고 Body를 저장할 StringBuilder sb를 가지고 있습니다.

또한 try-catch문을 사용하여 예외처리합니다.

5.4 HeaderParser.java

```java
public class HeaderParser {

    public void HeaderHandler(BufferedReader br, String[] MPV, HashMap<String, String> headerMap)
            throws IOException, HTTPException {
        try {
            String line = br.readLine(); // request 첫줄 : 메서드' '경로' 'http버전
            System.out.println("------------------------------");
            System.out.println("Request From Client");
            System.out.println(line);

            if (line == null) {
                throw new HTTPException(400);
            }
            String[] firstData = line.split(" ");
            try {
                MPV[0] = firstData[0]; // 메서드
                MPV[1] = firstData[1]; // 경로
                MPV[2] = firstData[2]; // HTTP버전
            } catch (ArrayIndexOutOfBoundsException e) {
                throw new HTTPException(400);
            }

            if (!MPV[2].equals("HTTP/1.1"))
                throw new HTTPException(505);
            // br.readLine(); //한줄 띄고
            // System.out.println(httpMethod);

            while ((line = br.readLine()) != null) {// line에 다음줄 대입 및 null값 처리

                if (line.isEmpty())
                    break; // 앞으로 빈라인이 들어오면 EOL      You, 1초 전 • Uncommitted changes
                System.out.println(line); // 요청 출력
                String[] headerData = line.split(": ");
                headerMap.put(headerData[0], headerData[1]);

            }
```

<HeaderParser.java 8~43>

Client가 보낸 HTTP 메시지의 헤더를 처리하는 함수인 HeaderHandler를 가지고있는 클래스입니다. HTTP메시지의 첫줄은 메서드, 경로, HTTP버전을 가지고있으므로 해당줄을 먼저 처리한 뒤 Header 정보를 headerMap에 키:값으로 저장합니다. 첫번째 줄이 올바르게 대입되지 않으면 Client의 HTTP요청이 잘못된 것이므로 HTTPException(400)을 생성하여 예외처리할 수 있도록 합니다. 44번줄 뒤는 예외처리부분이므로 생략합니다.

5.5 HTTPExcpetion.java

```
9   public class HTTPException extends Exception {
10      int status;
11      String statusMsg;
12      HashMap<Integer, String> statusMap;
13
14      public HTTPException() {
15      }
16
17      public HTTPException(int status) throws IOException {
18          this.status = status;
19          statusMap = new HashMap<>();
20          this.readStatus();
21          this.statusMsg = statusMap.get(status);
22      }
23
24      @Override
25      public void printStackTrace() {
26          // TODO Auto-generated method stub
27          if (this.statusMsg == null) {
28              System.out.println("Unknown HTTP Exception");
29          } else {
30              System.out.println(this.statusMsg);
31          }
32
33      }
34
35      public void readStatus() throws IOException {
36          File status = new File("C:\\Users\\SUN\\Desktop\\ComputerNetworking\\Server\\src\\StatusCode.txt");
37
38          BufferedReader br = new BufferedReader(new FileReader(status));
39          String line;
40          while ((line = br.readLine()) != null) {
41              String[] stat = line.split(" : ");
42              this.statusMap.put(Integer.parseInt(stat[0]), stat[1]);
43          }
44          br.close();
45      }
46  }
```

<HTTPException.java 8~46>

Exception 클래스를 상속받아, 구현한 HTTP통신의 예외를 처리하는 HTTPException클래스입니다. 5.1의 정보를 사용하여 생성될 때 상태코드 정보를 읽어와서 저장합니다. 본 제출파일에서는 모든 예외를 처리하고 있지 않지만 상태코드 정보를 모두 저장하여 추후 확장가능성을 더했습니다. 생성될 때 상태코드를 입력받아 해당하는 상태정보를 printstacktrace 함수를 이용하여 출력할 수 있도록 구현하였습니다.

5.6 ResponseHandler.java

ClientHandler가 요청 헤더와 바디를 읽어들인 후 해당하는 응답전송을 구현하는 클래스입니다. 메서드와 경로에 따라서 응답을 구성해야 하므로, switch 문을 사용하여 메서드 구분을 한 뒤 해당 메서드의 경로에 따른 요청을 구분 하고 있습니다. 본 제출파일에서는 경로설정을 담당하는 Router클래스를 구현하지 못하고 정해진 경로에 따른 응답을 구현하고 있습니다.

```
15  public class ResponseHandler {
16      String method;
17      String path;
18      BufferedWriter bw;
19      HashMap<String, String> headerMap;
20
21      ResponseHandler(){}
22      ResponseHandler(int code, BufferedWriter bw) throws IOException{
23          //this.method = method;
24          //this.path = path;
25          this.bw = bw;
26          ///this.headerMap = headerMap;
27          sendTextResponse(bw, code, "");
28      }
29      ResponseHandler(String method, String path, BufferedWriter bw, HashMap<String,String> headerMap, StringBuilder body)throws IOException,HTTPException{
30
31          this.method = method;
32          this.path = path;
33          this.bw = bw;
34          this.headerMap = headerMap;
35          try{
36              methodHandler(method, body);
37          }catch(HTTPException e){
38              sendTextResponse(bw, e.status, "");
39          }
40
41      }
```

<ResponseHandler.java 15~41>

디폴트 생성자를 제외하고, 첫번째 생성자는 예외처리를 하기 위해 상태코드와 BufferedWriter를 매개변수로 받고, 두번째 생성자는 일반적인 응답 처리를 위해서 메서드, 경로, BufferedWriter, 헤더정보, Body 를 매개변수로 받고 있습니다.

```java
String methodHandler(String method, StringBuilder body) throws IOException, HTTPException{
    String responseString = "";
    switch (method) { // 메서드에 따른 처리
        case "HEAD":
            headHandler(path);
            break;
        case "GET":

            getHandler(path);
            break;
        case "POST":
            postHandler(path, body);
            break;
        case "DELETE":      You, 그저께 • add ResponseHandler
            deleteHandler(path);
            break;

        default:
            break;
    }

    return responseString;
}
```

<methodHandler 함수>

Switch – case 문을 사용하여 요청의 메서드별로 각 메서드의 Handler를 호출하는 함수입니다.

```java
67      void getHandler(String path)throws IOException, HTTPException{
68
69          if (path == null) {
70              //sendTextResponse(this.bw, 404,"");
71              throw new HTTPException(404);
72          }
73          if (path.equals("/errorpath")){
74              throw new HTTPException(404);
75          }
76          sendTextResponse(this.bw, 200, "Hello, World!");
77
78      }
79
80      void headHandler(String path) throws IOException, HTTPException{
81          if (path == null) {
82              sendTextResponse(this.bw, 404, "");
83              throw new HTTPException(404);
84          }
85          sendHeadResponse(this.bw, 200); //헤더만 보냄 + 쿠키설정
86      }
87
88      void postHandler(String path, StringBuilder body) throws IOException, HTTPException{
89          if (path == null) {
90              sendTextResponse(this.bw, 404,"");
91              throw new HTTPException(404);
92          }
93          String resbody = "";
94          try{
95              String id = headerMap.get("Cookie").split("=")[1];
96              String fileName = "file_" + id + ".txt";
97              Path filePath = Paths.get(fileName);
98              // Write body to file
99              Files.write(filePath, body.toString().getBytes(StandardCharsets.UTF_8));
100             resbody += "File saved: " + filePath;
101         }catch (Exception e){
102             sendTextResponse(this.bw, 500, "");      You, 1초 전 • Uncommitted changes
103             e.printStackTrace();
```

<get, head, post Handler>

기본적으로 Path정보가 담겨있지 않으면 올바르지 않은 경로에 대한 요청이므로 404에 대한 처리를 하고 있습니다. GET 메서드는 "Hello, World!"를 응답으로 하고있고, HEAD메서드는 해당 클라이언트의 id를 설정합니다. POST 메서드는 해당 클라이언트의 id를 제목으로 하고, 요청 Body를 내용으로 가지는 txt파일을 생성한 뒤 성공여부(실패시 505반환)를 응답합니다.

```
108     void deleteHandler(String path) throws IOException, HTTPException{
109         if (path == null) {
110             sendTextResponse(this.bw, 404,"");
111             throw new HTTPException(404);
112         }
113         String body = "";
114         try{
115             String id = headerMap.get("Cookie").split("=")[1];
116             String fileName = "file_" + id + ".txt";
117             Path filePath = Paths.get(fileName);
118             Files.delete(filePath);
119             body += "File Deleted: " + filePath;
120         }catch (Exception e){
121             e.printStackTrace();
122         }
123         sendTextResponse(this.bw, 200, body);
124     }
```

<deleteHandler>

해당 요청을 한 클라이언트의 id와 동일한 txt파일을 삭제한 뒤 해당 여부를 응답으로 반환합니다. 본 과제 테스트에서는 생성되고 삭제되는 과정을 보이기 위해 POST, DELETE 동작 사이에 3초의 딜레이를 두었습니다.

```
125     void sendHeadResponse(BufferedWriter bw, int status)throws IOException{
126         HTTPException e = new HTTPException(status);
127         String id = UUID.randomUUID().toString();
128         String responseString = String.format(
129 
130                                 "HTTP/1.1 %d %s\r\n" +
131                                 "Content-Type: text/plain\r\n" +
132                                 "Content-Length: 0\r\n" +
133                                 "Date: %s\r\n" +
134                                 "Server: HTTPServer\r\n" +
135                                 "Set-Cookie: client_id="+ id +"\r\n" +
136                                 "\r\n",
137                 status, e.statusMap.get(status), Calendar.getInstance().getTime().toString());
138 
139                 bw.write(responseString);
140                 bw.flush();
141     }
142     void sendTextResponse(BufferedWriter bw, int status, String bodyString) throws IOException{
143         HTTPException e = new HTTPException(status);
144         String responseString = String.format(
145 
146                                 "HTTP/1.1 %d %s\r\n" +
147                                 "Content-Type: text/plain\r\n" +
148                                 "Content-Length: %d\r\n" +
149                                 "Date: %s\r\n" +
150                                 "Server: HTTPServer\r\n" +
151                                 "\r\n" +
152                                 "%s",
153                 status, e.statusMap.get(status), bodyString.getBytes().length,Calendar.getInstance().getTime().toString(), bodyString);
154 
155                 bw.write(responseString);
156                 bw.flush();
157 
158     }
159 }
```

<sendHeadResponse와 sendTextResponse>

HEAD는 쿠키를 설정하기 위해서 함수를 따로 두었고 SendTextResponse는 일반적인 응답을 처리하기 위해서 작성하였습니다. 해당 메시지가 작성될 때 시각을 알려주는 Date, 해당서버정보인 Server, 를 기본헤더로 가지고있고 , sendHeadResponse는 Set-Cookie 헤더 키를 사용하여 클라이언트에 UUID로 생성한 id를 전달합니다.

5.6 HTTPServer.java

실제 Server를 구동하는 파일입니다.

```java
package Server;

import java.net.ServerSocket;
import java.net.Socket;
import java.net.SocketException;
import java.io.IOException;

You, 1시간 전 | 2 authors (You and one other)
public class HttpServer {
    Run | Debug
    public static void main(String[] args) {
        try (ServerSocket server = new ServerSocket(80)) { //http통신을 위해 80번포트에서 서버소켓 대기
            System.out.println("Listening on port 80...");
            while (true) {
                try {
                    Socket client = server.accept();
                    System.out.println("Connected to client " + client.getInetAddress());
                    client.setSoTimeout(10000);  // 10초로 타임아웃 설정

                    new Thread(new ClientHandler(client)).start();  // 각 요청을 받을 때마다 새로운 스레드로 각각 처리 가능
                } catch (SocketException e) {       You, 5시간 전 • 소켓 연결오류 해결 : 서버소켓을 열때 try-catch문으로…
                    System.err.println("SocketException: " + e.getMessage());
                } catch (IOException e) {
                    System.err.println("IOException: " + e.getMessage());
                }
            }
        } catch (IOException e) {
            System.err.println("ServerSocket initialization error: " + e.getMessage());
        }
    }
}

```

<HTTPServer.java>

5.1~5.5의 코드를 사용하여 목표기능들을 모두 구현하였습니다. HTTP통신을 할 때 사용하는 Well-Known 포트인 80번 포트를 사용하여 ServerSocket을 생성하고 대기합니다. 새로운 Client소켓이 연결될 때마다 Time-out을 10초(POST-DELETE응답사이 3초딜레이가 있어)로 설정한 뒤 새로운 Thread에서 Client에 대한 처리를 합니다. ServerSocket을 생성할 때도 try-catch 문으로 처리하지 않으면 연결오류가 발생했습니다.

5.7 ClientHeaderParser.java

```java
import java.io.BufferedReader;
import java.io.IOException;
import java.util.HashMap;

import Server.HTTPException;

You, 8시간 전 | 1 author (You)
public class ClientHeaderParser {
    public void HeaderHandler(BufferedReader br, String[] MPV, HashMap<String,String> headerMap)throws IOException, HTTPException{
        try{
            String line = br.readLine();  //request 첫줄 : 메서드' '경로' 'http버전
                System.out.println("--------------------------------");
                System.out.println("Response From Server");
                System.out.println(line);

                if (line == null){
                    //이러면 클라이언트 오류이므로 400반환
                }       You, 8시간 전 • add ClientHeaderParser, req,res HTTP 메세지 설…
                String[] firstData = line.split(" ");

                MPV[0] = firstData[0];  //HTTP 버전
                MPV[1] = firstData[1];  //Status Code
                MPV[2] = firstData[2];  //status Code 설명

                while ((line = br.readLine()) != null){//line에 다음줄 대입 및 null값 처리

                    if (line.isEmpty())break;   //앞으로 빈라인이 들어오면 EOL
                    System.out.println(line);   //응답 출력
                    String[] headerData = line.split(": ");
                    headerMap.put(headerData[0], headerData[1]);

                }
        }
        catch (Exception e){
            e.printStackTrace();
        }
```

<ClientHeaderParser>

Client가 받은 HTTP응답을 처리하는 클래스입니다. Server의 HeaderParser와 대부분 동일하지만 첫줄의 정보가 다르므로 각각 구현하였습니다. 버전, 상태코드, 상태코드 메시지를 읽어와 저장하고 응답 헤더 정보를 while문으로 읽어와 headerMap에 키:값으로 저장합니다.

5.8 HTTPClient.java

```java
package Client;

import java.net.Socket;
import java.util.HashMap;

import Server.BodyParser;
import Server.HTTPException;

import java.io.BufferedReader;
import java.io.BufferedWriter;

import java.io.IOException;
import java.io.InputStreamReader;
import java.io.OutputStreamWriter;
import java.util.*;

public class HTTPClient {

        // 와이어샤크 캡처 성공
        Run | Debug
        public static void main(String[] args) throws IOException, InterruptedException, HTTPException {
                Socket socket = new Socket("127.0.0.1", 80);
                HashMap<String, String> cookie = new HashMap<>(); // 클라이언트에 저장할 쿠키(본 예시에서는 클라이언트 id를 저장할 것)

                BufferedReader br = new BufferedReader(new InputStreamReader(socket.getInputStream())); // io 스트림 생성 ->
                                                                                                        // 연결
                BufferedWriter bw = new BufferedWriter(new OutputStreamWriter(socket.getOutputStream()));
                HashMap<String, String> headerMap = new HashMap<>(); // 헤더 종류: 헤더값 저장
                String[] methodPathVersion = new String[3]; // 0: HTTP버전, 1: Status Code, 2: Status Code 설명

                String head_request = String.format("HEAD / HTTP/1.1\r\n" + // 첫번째 request HEAD 메소드(헤더정보만 요청할 때 사용)
                                "Host: %s\r\n" +
                                "User-Agent: HTTPClient/1.0\r\n" +
                                "\r\n", socket.getInetAddress() + ":" + socket.getPort());

                // 첫번째 요청 전송 및 응답
                sendHTTPRequest(bw, head_request);
                receiveHTTPResponse(br, methodPathVersion, headerMap, cookie);
```

<HTTPClient 1~37>

HTTP통신을 진행하는 Client입니다. Socket모듈을 이용하여 TCP 소켓을 생성하고 쿠키를 저장하는 HashMap<String,String> headerMap, 헤더를 저장하는 HashMap<String,String> headerMap을 변수로 가지고 있습니다. 또한 BufferedReader, BufferedWriter를 이용하여 서버소켓과 입출력스트림을 연결하고 정보를 주고받을 수 있습니다. HEAD 요청을 통해 쿠키를 저장한 뒤 이후 요청을 진행해야 하므로 HEAD 요청을 첫번째로 하고 있습니다. 기본 헤더로는 목적지에 대한 정보를 가지고있는 Host, 브라우저 정보를 담고있는 User-Agent(본과제에서는 임의로 HTTPClient/1.0을 입력했음)를 가지고 있습니다.

```java
92      static void sendHTTPRequest(BufferedWriter bw, String request) throws IOException {
93              bw.write(request);
94              bw.flush();
95      }
96
97      static void receiveHTTPResponse(BufferedReader br, String[] MPV, HashMap<String, String> headerMap,
98                      HashMap<String, String> cookie) throws IOException, HTTPException {
99              ClientHeaderParser hP = new ClientHeaderParser();
100             BodyParser bP = new BodyParser();
101             StringBuilder sb = new StringBuilder(); // Body 저장할 StringBuilder
102             hP.HeaderHandler(br, MPV, headerMap);
103             if (headerMap.containsKey("Content-Length")) { // Body가 있는 경우
104                     bP.BodyHandler(br, sb, Integer.parseInt(headerMap.get("Content-Length"))); // Body 처리 완료
105             }
106             if (headerMap.containsKey("Set-Cookie")) { // 쿠키가 있는 경우
107                     String[] cookieData = headerMap.get("Set-Cookie").split("=");
108                     cookie.put(cookieData[0], cookieData[1]);
109                     // System.out.println(cookie.get("client_id"));
110             }
111             System.out.println(sb.toString());
112             headerMap.clear(); // headerMap 초기화
113     }
114 }
```

<HTTPClient 92~114>

sendHTTPRequest함수를 이용하여 미리 설정된 HTTP메시지를 Server로 전송할 수 있습니다.

receiveHTTPResponse 함수는 ClientHeaderParser와 BodyParser를 이용하여 서버로부터 받은 응답의 헤더와 바디를 읽고 저장합니다. 헤더에 "Content-Length"가 있을 경우 바디를 읽고 그렇지 않은 경우에는 무시합니다. 또한 "Set-Cookie"가 있을 경우에는 해당 쿠키를 읽어와 cookie변수에 저장하고 이후 요청에 사용합니다.

```java
39              String reqBody = "Hello! Server!";
40              String post_request = String.format("POST / HTTP/1.1\r\n" + // 세번째 request POST 메소드
41                              "Host: %s\r\n" +
42                              "User-Agent: HTTPClient/1.0\r\n" +
43                              "Content-Length: %d\r\n" +
44                              "Cookie: %s\r\n" + // 쿠키 전송
45                              "\r\n" +
46                              "%s", socket.getInetAddress() + ":" + socket.getPort(), reqBody.getBytes().length,
47                              "client_id=" + cookie.get("client_id"), reqBody);
48              String get_request = String.format("GET / HTTP/1.1\r\n" + // 두번째 request GET 메소드
49                              "Host: %s\r\n" +
50                              "User-Agent: HTTPClient/1.0\r\n" +
51                              "\r\n", socket.getInetAddress() + ":" + socket.getPort());
52
53              String delete_request = String.format("DELETE / HTTP/1.1\r\n" + // 네번째 request DELETE 메소드
54                              "Host: %s\r\n" +
55                              "User-Agent: HTTPClient/1.0\r\n" +
56                              "Cookie: %s\r\n" + // 쿠키 전송
57                              "\r\n", socket.getInetAddress() + ":" + socket.getPort(),
58                              "client_id=" + cookie.get("client_id"));
59
60              String request_404 = String.format("GET /errorpath HTTP/1.1\r\n" + // 404 에러 발생시키는 request
61                              "Host: %s\r\n" +
62                              "User-Agent: HTTPClient/1.0\r\n" +
63                              "\r\n", socket.getInetAddress() + ":" + socket.getPort());
```

<HTTPClient 39~63>

```
64              // 두번째 요청 전송 및 응답
65              sendHTTPRequest(bw, get_request);
66              receiveHTTPResponse(br, methodPathVersion, headerMap, cookie);
67
68              // 세번째 요청 전송 및 응답
69              sendHTTPRequest(bw, post_request);
70              receiveHTTPResponse(br, methodPathVersion, headerMap, cookie);
71              Thread.sleep(3000); // 3초 대기후 해당 파일 삭제요청 보냄
72              // 네번째 요청 전송 및 응답
73              sendHTTPRequest(bw, delete_request);
74              receiveHTTPResponse(br, methodPathVersion, headerMap, cookie);
75
76              // 404 에러 발생시키는 요청 전송 및 응답
77
78              sendHTTPRequest(bw, request_404);
79              receiveHTTPResponse(br, methodPathVersion, headerMap, cookie);
80
81              // 505 에러 발생시키는 요청 전송 및 응답
82
83              String request_505 = String.format("GET / HTTP/2\r\n" + // 505 에러 발생시키는 request
84                              "Host: %s\r\n" +
85                              "User-Agent: HTTPClient/1.0\r\n" +
86                              "\r\n", socket.getInetAddress() + ":" + socket.getPort());
87              sendHTTPRequest(bw, request_505);
88              receiveHTTPResponse(br, methodPathVersion, headerMap, cookie);
89
90          }
```

<HTTPClient 64~90>

HEAD 요청 이후 GET, POST, DELETE 요청을 진행하고 각 응답을 처리합니다. 404요청은 오류경로를 입력하여 서버가 404응답을 처리하는 응답을 하도록 하고, 505 요청은 서버에서 지원하지 않는 HTTP/2 버전을 요청하여 505응답을 하도록 구현하였습니다.

6. 클라이언트 HTTP요청에 따른 서버 응답

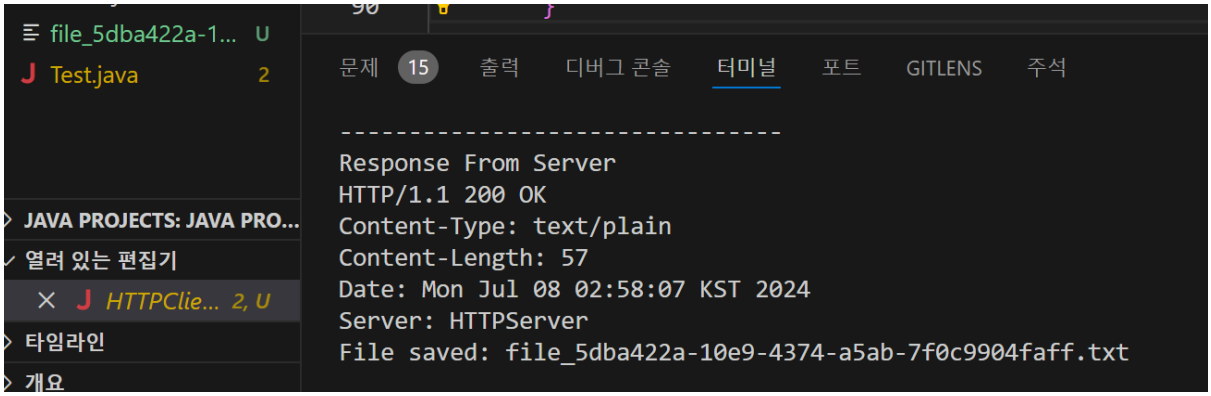
```
Listening on port 80...
Connected to client /127.0.0.1
Socket connected from /127.0.0.1
--------------------------------
Request From Client
HEAD / HTTP/1.1
Host: /127.0.0.1:80
User-Agent: HTTPClient/1.0
--------------------------------
Request From Client
GET / HTTP/1.1
Host: /127.0.0.1:80
User-Agent: HTTPClient/1.0
--------------------------------
Request From Client
POST / HTTP/1.1
Host: /127.0.0.1:80
User-Agent: HTTPClient/1.0
Content-Length: 14
Cookie: client_id=4cc6ecc9-9f97-4a05-9b47-0ac70ef49ad6
--------------------------------
Request From Client
DELETE / HTTP/1.1
Host: /127.0.0.1:80
User-Agent: HTTPClient/1.0
Cookie: client_id=4cc6ecc9-9f97-4a05-9b47-0ac70ef49ad6
--------------------------------
Request From Client
GET /errorpath HTTP/1.1
Host: /127.0.0.1:80
User-Agent: HTTPClient/1.0
--------------------------------
Request From Client
GET / HTTP/2
--------------------------------
Client disconnected from /127.0.0.1
```

<Server 측 실행결과>

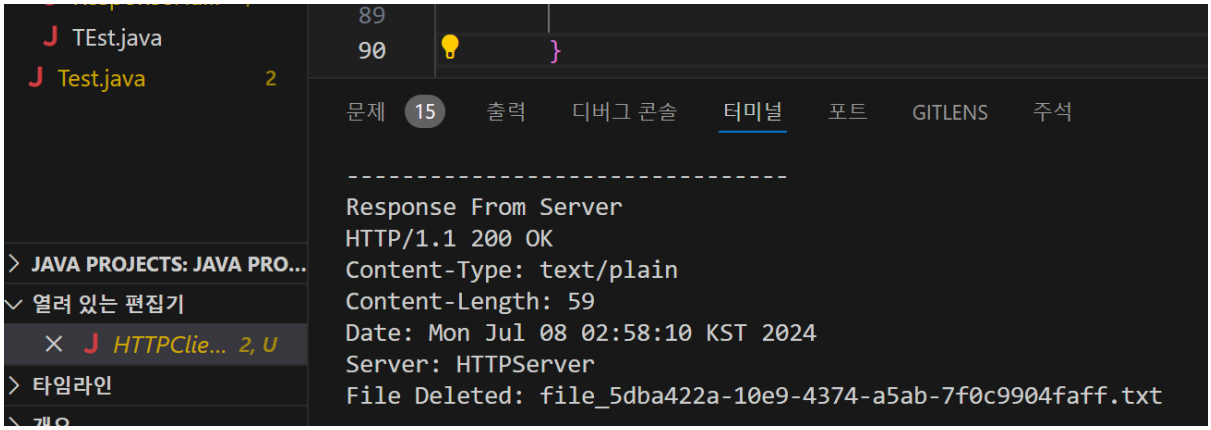
```
--------------------------------
Response From Server
HTTP/1.1 200 OK
Content-Type: text/plain
Content-Length: 0
Date: Mon Jul 08 02:54:07 KST 2024
Server: HTTPServer
Set-Cookie: client_id=4cc6ecc9-9f97-4a05-9b47-0ac70ef49ad6

--------------------------------
Response From Server
HTTP/1.1 200 OK
Content-Type: text/plain
Content-Length: 13
Date: Mon Jul 08 02:54:07 KST 2024
Server: HTTPServer
Hello, World!
--------------------------------
Response From Server
HTTP/1.1 200 OK
Content-Type: text/plain
Content-Length: 57
Date: Mon Jul 08 02:54:07 KST 2024
Server: HTTPServer
File saved: file_4cc6ecc9-9f97-4a05-9b47-0ac70ef49ad6.txt
--------------------------------
Response From Server
HTTP/1.1 200 OK
Content-Type: text/plain
Content-Length: 59
Date: Mon Jul 08 02:54:10 KST 2024
Server: HTTPServer
File Deleted: file_4cc6ecc9-9f97-4a05-9b47-0ac70ef49ad6.txt
--------------------------------
Response From Server
HTTP/1.1 404 Not Found
Content-Type: text/plain
Content-Length: 0
Date: Mon Jul 08 02:54:10 KST 2024
Server: HTTPServer

--------------------------------
Response From Server
HTTP/1.1 505 HTTP Version Not Supported
```

```
Content-Type: text/plain
Content-Length: 0
Date: Mon Jul 08 02:54:10 KST 2024
Server: HTTPServer
```

<Client 측 실행결과>



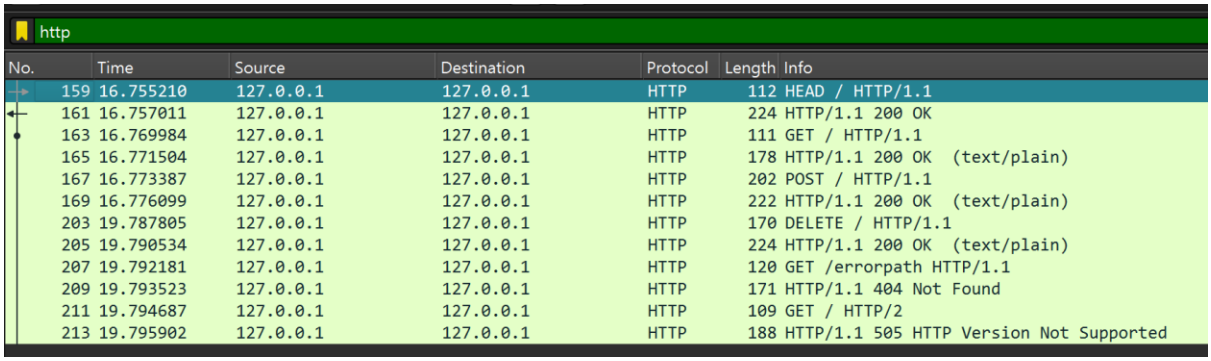
<POST요청에 따른 파일생성 성공화면>



<DELETE요청에 따른 파일삭제 성공화면>

각 요청에 따른 응답이 올바르게 전송되고 있음을 확인할 수 있습니다. 파일의 작성과 삭제는 영상에서 더 자세히 볼 수 있습니다. 해당 캡처는 PC내부에서 로컬호스트를 사용한 실행결과이고, 영상에서는 두 PC간의 통신으로 촬영하였습니다.

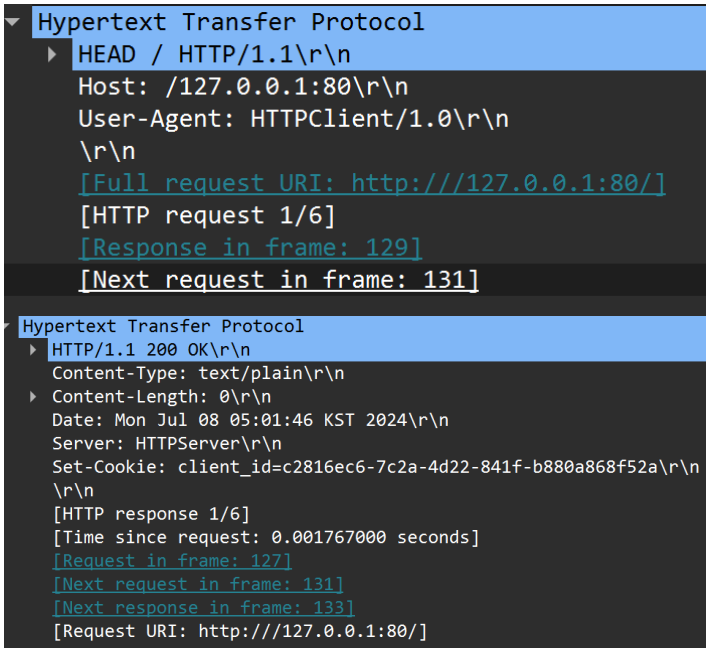
7. 와이어샤크 HTTP Format 캡처



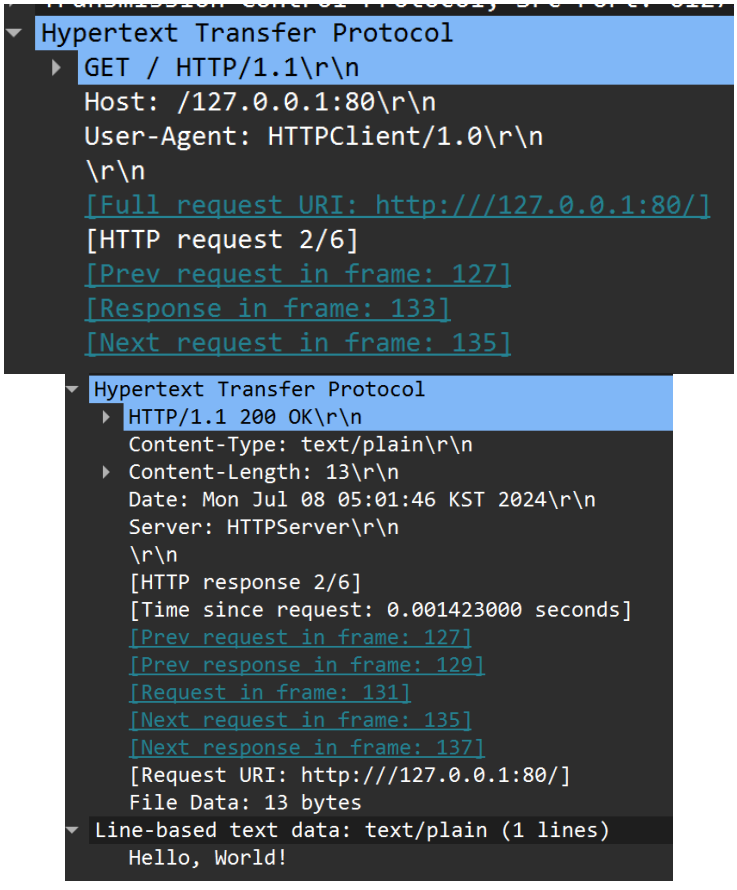
<WireShark 패킷 캡쳐화면>

클라이언트와 서버간 HTTP 패킷들이 올바르게 WireShark에서 캡처되었습니다.

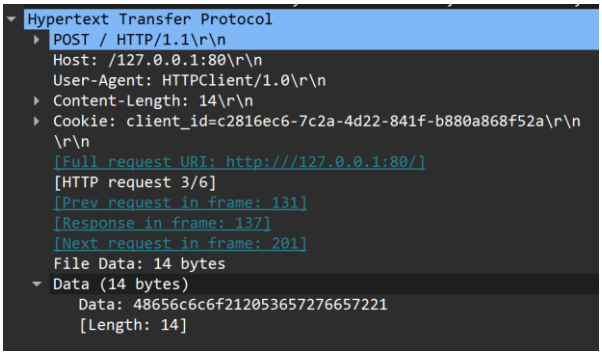
7.1 HEAD 요청과 응답 Format 캡처

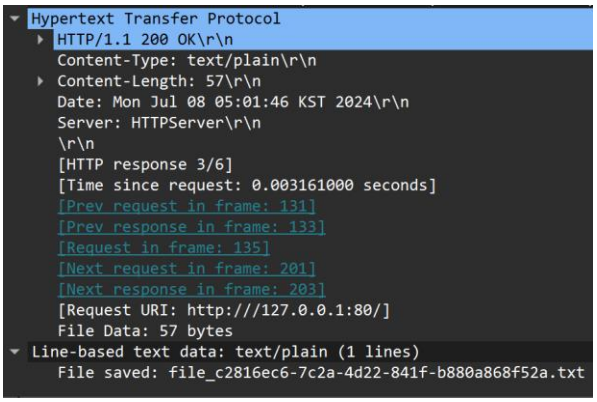


7.2 GET 요청과 응답 Format 캡처

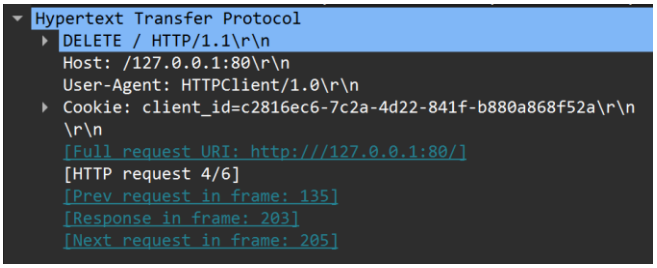


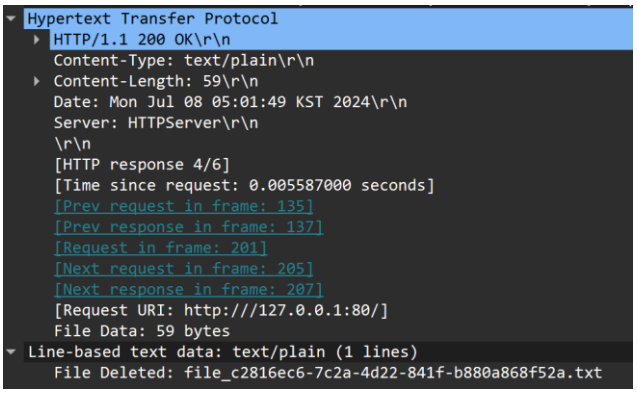
7.3 POST 요청과 응답 Format 캡처



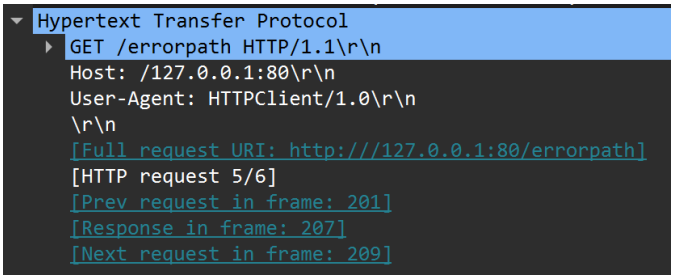


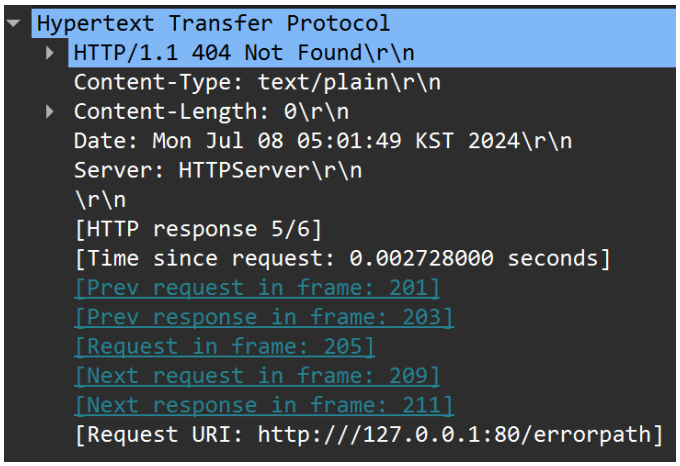
7.4 DELETE 요청과 응답 Format 캡처



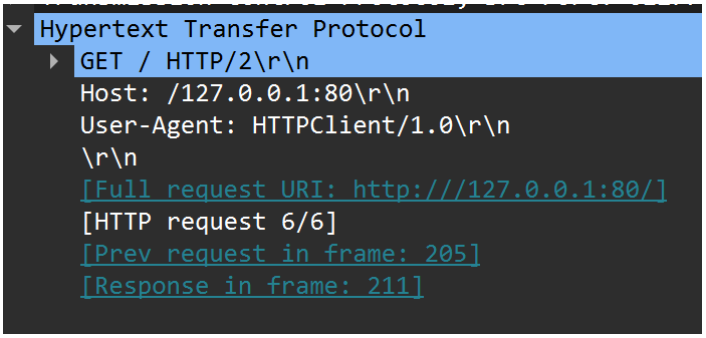


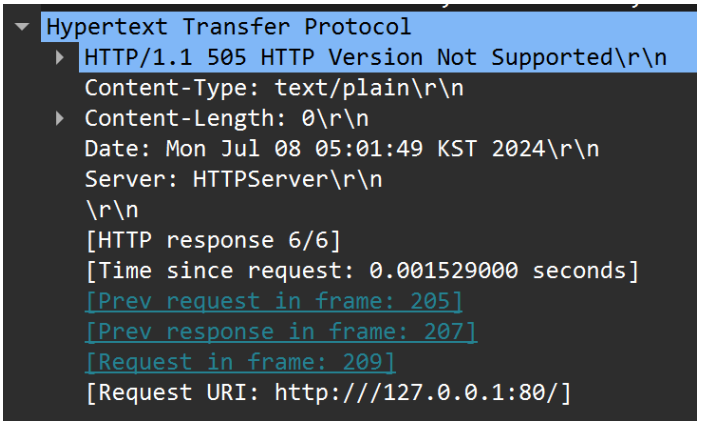
7.5 잘못된 경로의 GET 요청과 응답 Format 캡처



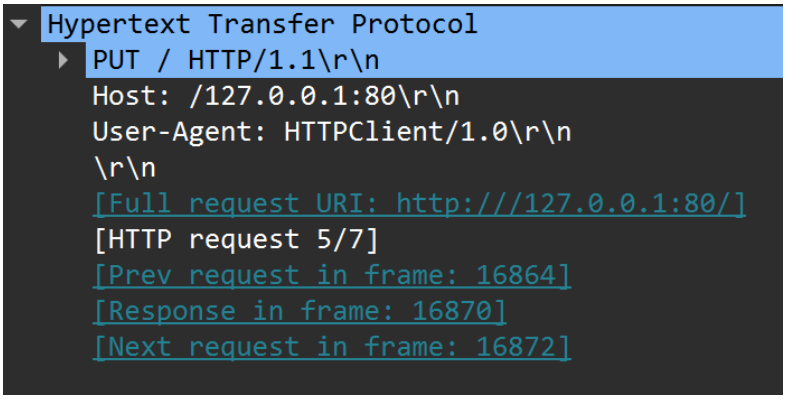


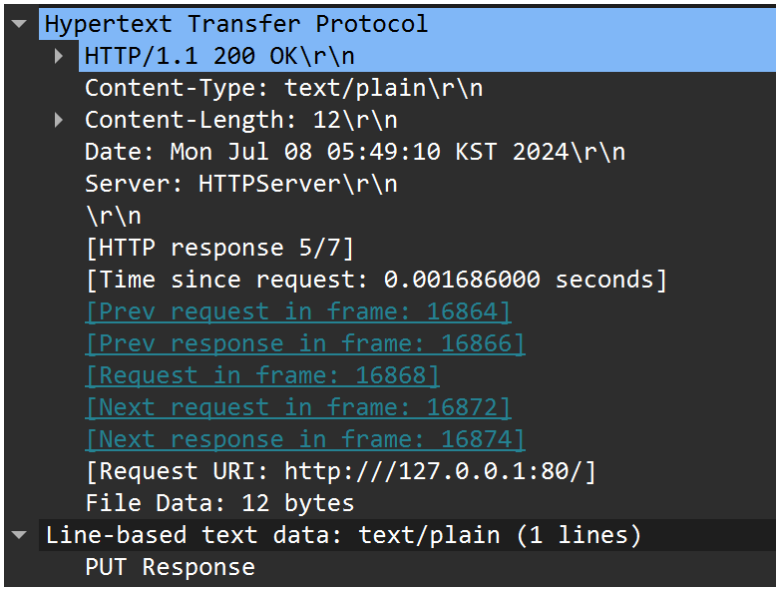
7.6 잘못된 버전의 GET 요청과 응답 Format 캡처





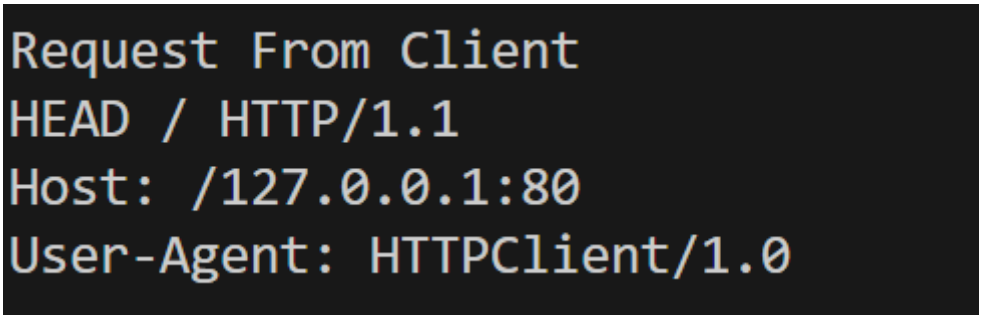
7.7 PUT 요청과 응답 Format 캡처





8. HTTP 명령어와 결과

8.1 HEAD 요청



```
Response From Server
HTTP/1.1 200 OK
Content-Type: text/plain
Content-Length: 0
Date: Mon Jul 08 03:00:38 KST 2024
Server: HTTPServer
Set-Cookie: client_id=71fc5491-c47c-4dec-8f98-c06674caa501
```

HEAD 요청은 응답본문은 요구하지 않는 요청이므로 헤더만 응답으로 반환합니다.

본 예시에서는 첫번째 요청이므로 해당 클라이언트에게 쿠키를 설정하는 "Set-Cookie"헤더를 포함하여 응답하고 있습니다.

8.2 GET 요청

```
Response From Server
HTTP/1.1 200 OK
Content-Type: text/plain
Content-Length: 13
Date: Mon Jul 08 03:00:38 KST 2024
Server: HTTPServer
Hello, World!
```

```
Request From Client
GET / HTTP/1.1
Host: /127.0.0.1:80
User-Agent: HTTPClient/1.0
```

/ 경로에 따른 자원을 서버에게 요청합니다. 본 과제에서는 기본 문자열 "Hello, World!"를 응답에 포함하여 반환 하였습니다.

8.3 POST 요청

```
Request From Client
POST / HTTP/1.1
Host: /127.0.0.1:80
User-Agent: HTTPClient/1.0
Content-Length: 14
Cookie: client_id=71fc5491-c47c-4dec-8f98-c06674caa501
```

```
Response From Server
HTTP/1.1 200 OK
Content-Type: text/plain
Content-Length: 57
Date: Mon Jul 08 03:00:38 KST 2024
Server: HTTPServer
File saved: file_71fc5491-c47c-4dec-8f98-c06674caa501.txt
```

요청의 Body인 "Hello Server!"를 해당 클라이언트의 id를 제목에 포함하여 서버에 저장하도록 하는 POST요청입니다. 올바르게 저장되었다는 상태코드인 200과 Body에 저장된 파일 명을 담아 반환합니다.

8.4 DELETE 요청

```
---------------------------------
Request From Client
DELETE / HTTP/1.1
Host: /127.0.0.1:80
User-Agent: HTTPClient/1.0
Cookie: client_id=71fc5491-c47c-4dec-8f98-c06674caa501
```

```
Response From Server
HTTP/1.1 200 OK
Content-Type: text/plain
Content-Length: 59
Date: Mon Jul 08 03:00:41 KST 2024
Server: HTTPServer
File Deleted: file_71fc5491-c47c-4dec-8f98-c06674caa501.txt
```

해당 클라이언트의 id를 제목으로하는 파일을 삭제하는 요청입니다. 해당 요청이 올바르게 처리되었다는 상태코드 200과, 삭제된 파일 명을 Body에 담아 응답으로 반환합니다.

8.5 GET 요청(404)

```
Request From Client
GET /errorpath HTTP/1.1
Host: /127.0.0.1:80
User-Agent: HTTPClient/1.0
```

```
Response From Server
HTTP/1.1 404 Not Found
Content-Type: text/plain
Content-Length: 0
Date: Mon Jul 08 03:00:41 KST 2024
Server: HTTPServer
```

해당 요청의 경로가 올바르지 않기 때문에 404 상태코드를 응답으로 반환합니다.

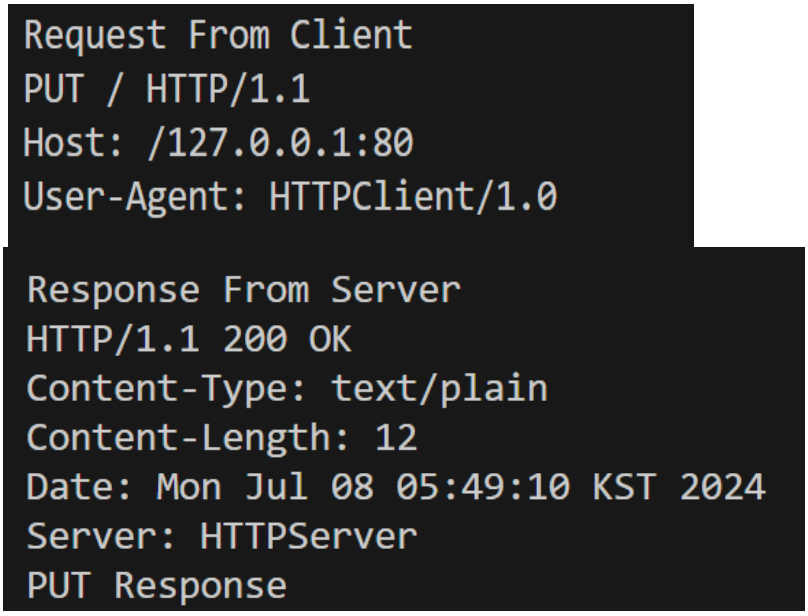
8.6 GET 요청(505)

```
Request From Client
GET / HTTP/2
```

```
Response From Server
HTTP/1.1 505 HTTP Version Not Supported
Content-Type: text/plain
Content-Length: 0
Date: Mon Jul 08 03:00:41 KST 2024
Server: HTTPServer
```

해당 HTTP요청이 요구하는 HTTP버전이 HTTP/2이므로 서버는 지원하지 않는 버전에 대해서 상태코드 505를 담아 반환합니다.

8.7 PUT 요청



9. 참고자료

https://coding-factory.tistory.com/270

https://atoz-develop.tistory.com/entry/JAVA-TCP-소켓-프로그래밍으로-간단한-HTTP-클라이언트-구현하기

[Java] HTTP 서버 만들기: 멀티스레딩 적용 — 개발자의 서랍 (tistory.com)

https://goddaehee.tistory.com/169

https://kadosholy.tistory.com/125

https://kadosholy.tistory.com/126

https://bonita-sy.tistory.com/entry/HTTP-Header-구조-및-요청응답-헤더의-주요-항목-정리