TCP-Socket-Programming
실습 과제

소프트웨어학부
20203112
이다은

1. 구현 내용

- 소켓 통신을 활용하여 Server, Client 프로그램 작성
- TCP 기반 소켓프로그래밍 작성후 Client 에서는 HTTP 프로토콜의 GET/HEAD/POST/PUT Request 를 요청하고, Server 에서는 Client 의 Request 에따라 응답 메시지를 구성하여 Response 하도록 구현
- 사용한 프로그래밍 언어: python
- localhost 로 진행
- 입출력에 이용한 파일 형식: html
- HTTP 명령어 수행시에 Server 에서 Client 가 요청하는 파일을 실제로 생성하고, update 하며 실제 환경에 맞는 Response State 를 회신함

2. HTTP 명령어 수행 결과 - client, server 출력 내용 및 메소드 설명

2-1. GET

(1) *GET test.html* 명령어 입력

: GET 명령어 뒤에 파일 이름을 입력하면, 서버에서 해당 파일을 읽어 들인 뒤 파일 내용을 body 에 담아 클라이언트에게 전송합니다. 클라이언트에서 *GET test.html* 을 요청하면 서버에서 test.html 이라는 이름의 파일이 존재하는지 확인후 파일의 내용을 response 의 body 에 담아 전송하게 됩니다.

- Server 출력 내용

```
(base) daeun@ida-eun-ui-MacBookPro-2 WebSocket % /opt/homebrew/bin/python3
 /Users/daeun/WebSocket/server.py
The server is running
Connected by ('127.0.0.1', 49733)
Perform a GET request
```

- Client 출력 내용

```
(base) daeun@ida-eun-ui-MacBookPro-2 WebSocket % python client.py
Enter the HTTP command GET test.html
Received HTTP/1.1 200 OK
Date: 2022-05-02 18:19:51.696444
Host: localhost
Content-Type: text/html
Content-Length: 286

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
    <h1> This is HTML </h1>
</body>
</html>
Enter the HTTP command
```

- Test.html 파일 내용

```
1   <!DOCTYPE html>
2   <html lang="en">
3   <head>
4       <meta charset="UTF-8">
5       <meta http-equiv="X-UA-Compatible" content="IE=edge">
6       <meta name="viewport" content="width=device-width, initial-scale=1.0">
7       <title>Document</title>
8   </head>
9   <body>
10      <h1> This is HTML </h1>
11  </body>
12  </html>
```

(2) *GET blank.html* 명령어 입력 (blank.html 이 존재하지 않는 파일인 경우)
: 클라이언트에서 *GET blank.html* 을 요청하면 서버에서 확인 후 blank.html 이라는 이름의 파일이 존재하지 않으므로, 클라이언트가 찾는 리소스가 없다는 뜻의 404 응답 코드를 response 의 헤더에 담아 클라이언트에게 전송합니다.

- Server 출력 내용

```
(base) daeun@ida-eun-ui-MacBookPro-2 WebSocket % /opt/homebrew/bin/python3
 /Users/daeun/WebSocket/server.py
The server is running
Connected by ('127.0.0.1', 49914)
Perform a GET request
[Errno 2] No such file or directory: 'blank.html'
```

- Client 출력 내용

```
(base) daeun@ida-eun-ui-MacBookPro-2 WebSocket % python client.py
Enter the HTTP command GET blank.html
Received HTTP/1.1 404 Not Found
Date: 2022-05-02 19:07:48.913914
Host: localhost
Content-Type: text/html
Content-Length: None

None
Enter the HTTP command
```

(3) 그 외 예외 처리
: 서버에서 요청을 수행하던 도중 FileNotFoundError 이 외의 에러가 발생하면 클라이언트의 요청 자체가 잘못 되었다는 뜻의 400 응답 코드를 response 의 헤더에 담아 클라이언트에게 전송합니다.

2-2. HEAD

(1) *HEAD test.html* 명령어 입력

: HEAD 요청 방식은 GET 과 유사한 방식이지만 서버에서 헤더 정보 이외에는 어떤 데이터도 보내지 않습니다. 클라이언트에서 *HEAD test.html* 을 요청하면 서버에서 test.html 이라는 이름의 파일이 존재하는지 확인 후, 파일 내용 본문을 포함하지 않고, 헤더 정보만을 response 에 담아 클라이언트로 전송합니다.

- Server 출력 내용

```
(base) daeun@ida-eun-ui-MacBookPro-2 WebSocket % /opt/homebrew/bin/python3
 /Users/daeun/WebSocket/server.py
The server is running
Connected by ('127.0.0.1', 49791)
Perform a HEAD request
```

- Client 출력 내용

```
(base) daeun@ida-eun-ui-MacBookPro-2 WebSocket % python client.py
Enter the HTTP command HEAD test.html
Received HTTP/1.1 200 OK
Date: 2022-05-02 18:38:17.044082
Host: localhost
Content-Type: text/html
Content-Length: 286

Enter the HTTP command
```

- Test.html 파일 내용

```
1   <!DOCTYPE html>
2   <html lang="en">
3   <head>
4       <meta charset="UTF-8">
5       <meta http-equiv="X-UA-Compatible" content="IE=edge">
6       <meta name="viewport" content="width=device-width, initial-scale=1.0">
7       <title>Document</title>
8   </head>
9   <body>
10      <h1> This is HTML </h1>
11  </body>
12  </html>
```

(3) 모든 예외 처리

: 서버에서 요청을 수행하던 도중 에러가 발생하면 클라이언트의 요청 자체가 잘못 되었다는 뜻의 400 응답 코드를 response 의 헤더에 담아 클라이언트에게 전송합니다.

2-3. PUT

(1) *PUT test.html This is test* 명령어 입력

: 클라이언트에서 <PUT 파일명 추가할 태그 내용>을 입력하면, 서버에서 해당 파일이 존재하는지 확인 후 FileNotFoundError 가 발생하지 않으면, 해당 파일의 body 태그 안에 클라이언트가 입력한 태그 내용을 추가합니다. 또한 서버에서 파일 내용을 response 의 body 에 담아 클라이언트에게 전송합니다. 클라이언트가 *PUT test.html This is test* 라고 요청하면 결과적으로 서버에서 파일의 내용을 수정해주고 text.html 파일 안에 <h2> This is test </h2> 라는 태그가 body 태그 안에 생깁니다.

- Server 출력 내용

```
(base) daeun@ida-eun-ui-MacBookPro-2 WebSocket % /opt/homebrew/bin/python3
 /Users/daeun/WebSocket/server.py
The server is running
Connected by ('127.0.0.1', 49863)
Perform a PUT request
```

- Client 출력 내용

```
(base) daeun@ida-eun-ui-MacBookPro-2 WebSocket % python client.py
Enter the HTTP command PUT test.html This is test
Received HTTP/1.1 200 OK
Date: 2022-05-02 18:52:55.880332
Host: localhost
Content-Type: text/html
Content-Length: 314

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
    <h2> This is test </h2>
    <h1> This is HTML </h1>
</body>
</html>
Enter the HTTP command
```

- PUT 요청 전 test.html 파일 내용

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
    <h1> This is HTML </h1>
</body>
</html>
```

- PUT 요청 후 test.html 파일 내용

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
    <h2> This is test </h2>
    <h1> This is HTML </h1>
</body>
</html>
```

(2) *PUT blank.html This is test* 명령어 입력 (blank.html 이 존재하지 않는 파일인 경우)
: 클라이언트에서 *PUT blank.html* 을 요청하면 서버에서 확인 후 blank.html 이라는 이름의 파일이 존재하지 않으므로, 클라이언트가 찾는 리소스가 없다는 뜻의 404 응답 코드를 response 의 헤더에 담아 클라이언트에게 전송합니다.

- Server 출력 내용

```
(base) daeun@ida-eun-ui-MacBookPro-2 WebSocket % /opt/homebrew/bin/python3
 /Users/daeun/WebSocket/server.py
The server is running
Connected by ('127.0.0.1', 49948)
Perform a PUT request
[Errno 2] No such file or directory: 'blank.html'
```

- Client 출력 내용

```
(base) daeun@ida-eun-ui-MacBookPro-2 WebSocket % python client.py
Enter the HTTP command PUT blank.html This is test
Received HTTP/1.1 404 Not Found
Date: 2022-05-02 19:16:13.965465
Host: localhost
Content-Type: text/html
Content-Length: None

None
Enter the HTTP command
```

(3) 그 외 예외 처리
: 서버에서 요청을 수행하던 도중 FileNotFoundError 이 외의 에러가 발생하면 클라이언트의 요청 자체가 잘못 되었다는 뜻의 400 응답 코드를 response 의 헤더에 담아 클라이언트에게 전송합니다.

2-4. POST

(1) *POST index.html <h1> This is HTML </h1>* 명령어 입력

: 클라이언트에서 <POST 생성할 파일명 파일 내용> 을 요청하면, 서버에서 해당 파일 내용으로 새로운 파일을 생성하고, 파일 내용을 response 의 body 에 담아 클라이언트에게 전송합니다. 클라이언트에서 *POST index.html <h1> This is HTML </h1>* 를 요청하면 서버에서 확인 후 exception 이 발생하지 않으면 index.html 이라는 이름으로 <h1> This is HTML </h1> 내용이 담긴 파일을 생성합니다.

- Server 출력 내용

```
(base) daeun@ida-eun-ui-MacBookPro-2 WebSocket % /opt/homebrew/bin/python3
 /Users/daeun/WebSocket/server.py
The server is running
Connected by ('127.0.0.1', 49886)
Perform a POST request
```

- Client 출력 내용

```
(base) daeun@ida-eun-ui-MacBookPro-2 WebSocket % python client.py
Enter the HTTP command POST index.html <h1> This is HTML </h1>
Received HTTP/1.1 201 Created
Date: 2022-05-02 18:59:40.541305
Host: localhost
Content-Type: text/html
Content-Length: 23

<h1> This is HTML </h1>
Enter the HTTP command
```

- POST 요청 후 생긴 파일 내용 (index.html)

```
index.html > h1
1    <h1> This is HTML </h1>
```

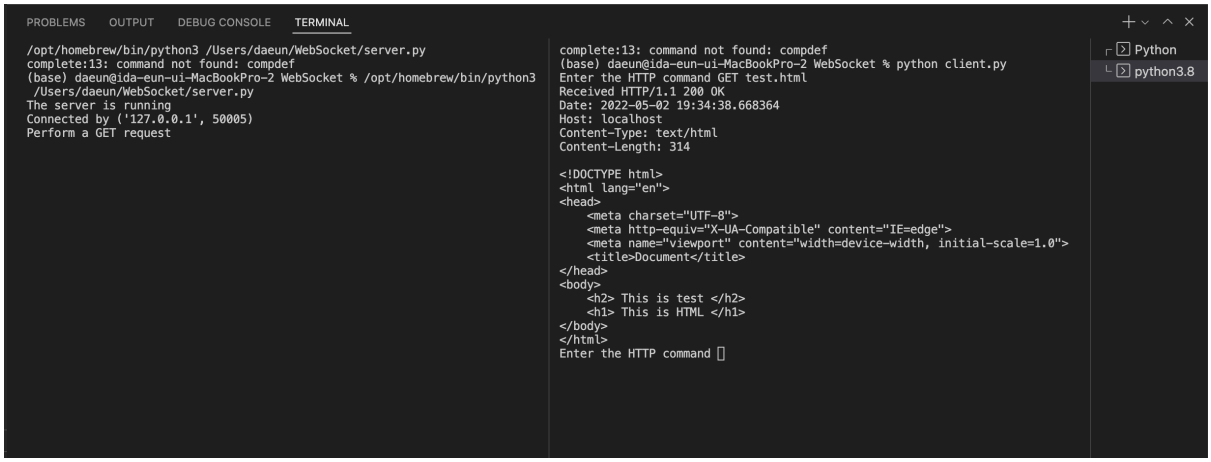
(2) 모든 예외 처리

: 서버에서 POST 요청을 수행하던 도중 에러가 발생하면 클라이언트의 요청 자체가 잘못 되었다는 뜻의 400 응답 코드를 response 의 헤더에 담아 클라이언트에게 전송합니다.

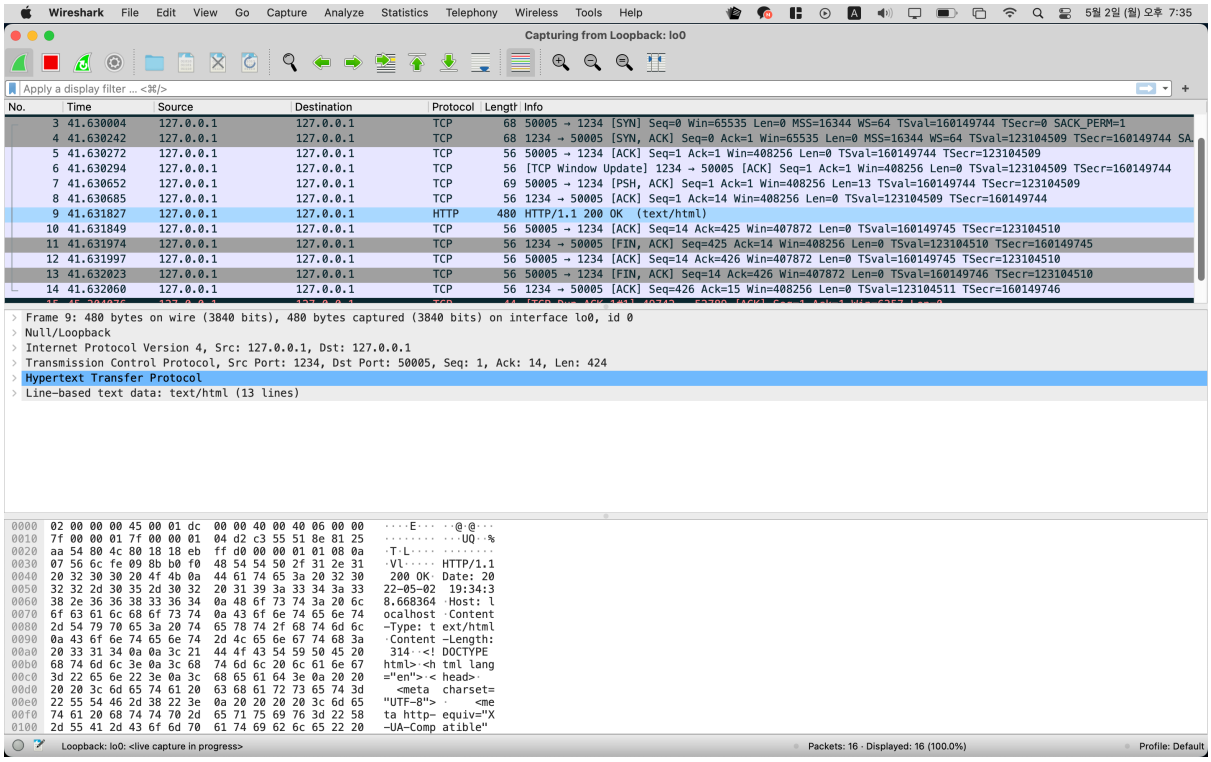
3. WireShark 캡쳐

: *GET test.html* 명령어 수행

- Server, Client 출력 내용



- 와이어샤크 캡쳐



4. 응답 코드

4-1. GET

- 200 (Success): 클라이언트의 요청이 성공적으로 처리된 경우 -> 클라이언트가 요청한 파일 내용 반환
- 400 (Bad Request): 클라이언트의 요청 자체가 잘못된 경우
- 404 (Not Found): 클라이언트에서 요청한 파일이 존재하지 않는 경우 (찾는 리소스가 없는 경우)

4-2. HEAD

- 200 (Success): 클라이언트의 요청이 성공적으로 처리된 경우 -> server 의 현재 상태 반환
- 400 (Bad Request): 클라이언트의 요청 자체가 잘못된 경우

4-3. PUT

- 200 (Success): 클라이언트의 요청이 성공적으로 처리된 경우 -> 파일의 body 태그 안에 내용 추가 후 파일 내용 반환
- 400 (Bad Request): 클라이언트의 요청 자체가 잘못된 경우
- 404 (Not Found): 클라이언트에서 요청한 파일이 존재하지 않는 경우 (찾는 리소스가 없는 경우)

4-4. POST

- 201 (Created): 클라이언트의 요청이 성공적으로 처리되어 리소스가 만들어진 경우 -> 클라이언트가 원하는 이름, 내용으로 리소스 생성 후 파일 내용 반환
- 400 (Bad Request): 클라이언트의 요청 자체가 잘못된 경우

5. 소스 코드 설명

5-1. server.py

- TCP server socket 생성

```
serverSocket = socket(AF_INET, SOCK_STREAM)
serverSocket.bind((HOST, PORT))
```

- request 별로 response format 설정

```
GetResponse = "HTTP/1.1 {header[0]} {header[1]}\nDate: {date}\nHost: {url}\nContent-Type:
text/html\nContent-Length: {l}\n\n{body}"
HeadResponse = "HTTP/1.1 {header[0]} {header[1]}\nDate: {date}\nHost: {url}\nContent-Type:
text/html\nContent-Length: {l}\n"
PutResponse = "HTTP/1.1 {header[0]} {header[1]}\nDate: {date}\nHost: {url}\nContent-Type:
text/html\nContent-Length: {l}\n\n{body}"
PostResponse = "HTTP/1.1 {header[0]} {header[1]}\nDate: {date}\nHost: {url}\nContent-Type:
text/html\nContent-Length: {l}\n\n{body}"
```

- accept 함수에서 대기하다가 클라이언트가 접속하면 새로운 소켓 리턴

```
clientSocket, addr = serverSocket.accept()
```

- 클라이언트로부터 전송받은 메시지를 split 해서 저장한 후, request 에 저장한 메소드에 따라 다른 함수 실행

```
requestMessage = message.split()
    request = requestMessage[0]

    if (request == "GET"):
        GET(clientSocket, requestMessage)
    elif (request == "HEAD"):
        HEAD(clientSocket, requestMessage)
    elif (request == "PUT"):
        PUT(clientSocket, requestMessage)
    elif (request == "POST"):
        POST(clientSocket, requestMessage)
```

5-2. client.py

- 클라이언트 소켓 만들기

```
clientSocket = socket(AF_INET, SOCK_STREAM)
clientSocket.connect((HOST, PORT))
clientSocket.send(request_message.encode())
```

- 응답 확인

```
data = clientSocket.recv(1024)
print('Received', data.decode())
```

6. 동작 환경

- localhost 로 구현
- server,client 모두 동일한 Mac OS 이용
- root and request path: 127.0.0.1