

# 컴퓨터네트워크 중간고사 대체 프로젝트

학과: 소프트웨어학과

학번: 20181608

이름: 문원기

GET한 파일이 client 위치에 생성되는 것까지 구현했습니다. 이는 3.1)에서 자세히 확인하실 수 있습니다.

또한 WireShark로 매전송마다 캡처하여 사진 첨부하였습니다.

## 1. 실행환경

운영체제 : Windows 10

사용 언어 : python

사용한 IDE : PyCharm Community Edition 2020.1.1

소켓 구현에 사용한 라이브러리 : socket

## 2. 소스 파일

server.py

```
from socket import *
import datetime

# html 만들기
html_text = """<!DOCTYPE html>
<html>
<head>
<title>Page Title</title>
</head>
<body>

<h1>This is a Heading</h1>
<p>This is a paragraph.</p>

</body>
</html>
"""

html_file = open('hello.html', 'w')
html_file.write(html_text)
html_file.close()

# socket 설정.
serverSock = socket(AF_INET, SOCK_STREAM)
serverSock.bind(('127.0.0.1', 80))
```

```

serverSock.listen(1)

connectionSock, addr = serverSock.accept()

print(str(addr), '에서 접속이 확인되었습니다.')

data = connectionSock.recv(1024)
data = data.decode('utf-8')
print('\n\n받은 데이터 : \n', data)

message = ""
dataSliced = data.split()
try:
    if dataSliced[0] == "GET":
        fileName = dataSliced[1]
        requestedFile = open('.') + fileName, 'rt', encoding='utf-8')
        sendData = requestedFile.read()
        message += "HTTP/1.1 200 OK\r\n" + "Date: " +
str(datetime.datetime.now()) + "\r\n"
        message += "Content-Type: text/html\r\n" + "Content-Length: 1" + "\r\n"
+ sendData + "\n\n"
    elif dataSliced[0] == 'POST':
        message += "HTTP/1.1 405 Method not allowed\r\n" + "Date: " +
str(datetime.datetime.now()) + "\r\n"
        message += "Content-type: text/html\r\n" + "Content-length: 0\n\n"
    elif dataSliced[0] == 'HEAD':
        fileName = dataSliced[1]
        requestedFile = open('.') + fileName, 'rt', encoding='utf-8')
        sendData = requestedFile.read()
        message += "HTTP/1.1 200 OK\r\n" + "Date: " +
str(datetime.datetime.now()) + "\r\n"
        message += "Content-type: text/html\r\n" + "Content-length: 1\n\n"
except IOError:
    message += "HTTP/1.1 404 NOT FOUND\r\n" + "Date: " +
str(datetime.datetime.now()) + "\r\n"
    message += "Content-type: text/html\r\n" + "Content-length: 0\n\n"

connectionSock.send(message.encode('utf-8'))
print('메시지를 보냈습니다.')

serverSock.close()

```

## client.py

```

from socket import *

clientSock = socket(AF_INET, SOCK_STREAM)
clientSock.connect(('127.0.0.1', 80))

print('연결 확인 되었습니다.')

print("GET 메소드- 응답 2XX 하려면 1")
print("GET 메소드- 응답 4XX 하려면 2")
print("POST 메소드- 응답 4XX 하려면 3")

```

```

print('HEAD 메소드- 응답 2XX 하려면 4')
print('HEAD 메소드- 응답 4XX 하려면 5')

inputVal = input('원하는 동작을 입력해주세요: ')
message = ""
fileMake = 0

if inputVal == '1':
    message += "GET /hello.html HTTP/1.1\r\n"+"Host: 127.0.0.1:80\r\n"+"Connection: Keep-Alive\r\n\r\n"
    fileMake = 1
elif inputVal == '2':
    message += "GET /notExisting.html HTTP/1.1\r\n"+"Host: 127.0.0.1:80\r\n"+"Connection: Keep-Alive\r\n\r\n"
    fileMake = 1
elif inputVal == '3':
    message += "POST / HTTP/1.1\r\n" + "Host: 127.0.0.1:80\r\n" + "Connection: Keep-Alive\r\n\r\n"
elif inputVal == '4':
    message += "HEAD /hello.html HTTP/1.1\r\n"+"Host: 127.0.0.1:80\r\n"+"Connection: Keep-Alive\r\n\r\n"
elif inputVal == '5':
    message += "HEAD /notExisting.html HTTP/1.1\r\n"+"Host: 127.0.0.1:80\r\n"+"Connection: Keep-Alive\r\n\r\n"

clientSock.send(message.encode('utf-8'))
print('\n\n보낸 메시지: \n', message)
print('메시지를 전송했습니다.\n\n')

data = clientSock.recv(1024)
print('받은 데이터 : ', data.decode('utf-8'))

if fileMake == 1:
    dataSliced = data.decode('utf-8').split(' ')
    if dataSliced[1] == '200':
        dataSliced = data.decode('utf-8').split('<!DOCTYPE html>')
        dataSliced[1] = "<!DOCTYPE html>\n" + dataSliced[1]
        html_file = open('client_hello.html', 'w')
        html_file.write(dataSliced[1])
        html_file.close()
        print('GET한 파일이 저장되었습니다.')

clientSock.close()

```

### 3. 프로그램 실행 결과

서버를 먼저 실행시키면 서버는 소켓을 준비하기 전에 hello.html 파일을 작성합니다.

```
client.py x hello.html x exp_client.py x exp_server.py x server.py x
1  <!DOCTYPE html>
2  <html>
3  <head>
4    <title>Page Title</title>
5  </head>
6  <body>
7
8    <h1>This is a Heading</h1>
9    <p>This is a paragraph.</p>
10
11  </body>
12  </html>
13
```

위 파일을 기준으로 GET과 POST, HEAD를 처리할 것이며,

특이사항으로는 POST 메소드가 지금 구현에서는 client의 입력을 신뢰할 수 없다는 판단 하에 Method Not Allowed를 리턴하도록 하였습니다.

POST 설명하는 부분에서 자세히 설명하겠습니다.

서버는 이후에 소켓을 설정한 후에 연결을 기다립니다.

```
Run: server x client x
C:\python38\python.exe C:/Users/wkm99/PycharmProjects/studyProject/server.py
('127.0.0.1', 1079) 에서 접속이 확인되었습니다.
```

연결이 되면 위와 같은 메시지를 출력하는데, 보시면 포트번호가 80이 아닌 것을 확인할 수 있습니다.

wireShark에서 확인해본 결과, 1079는 source port이며 80을 destination으로 삼는 것을 확인할 수 있었습니다. 즉, 정상적으로 작동하는 것이 맞습니다.

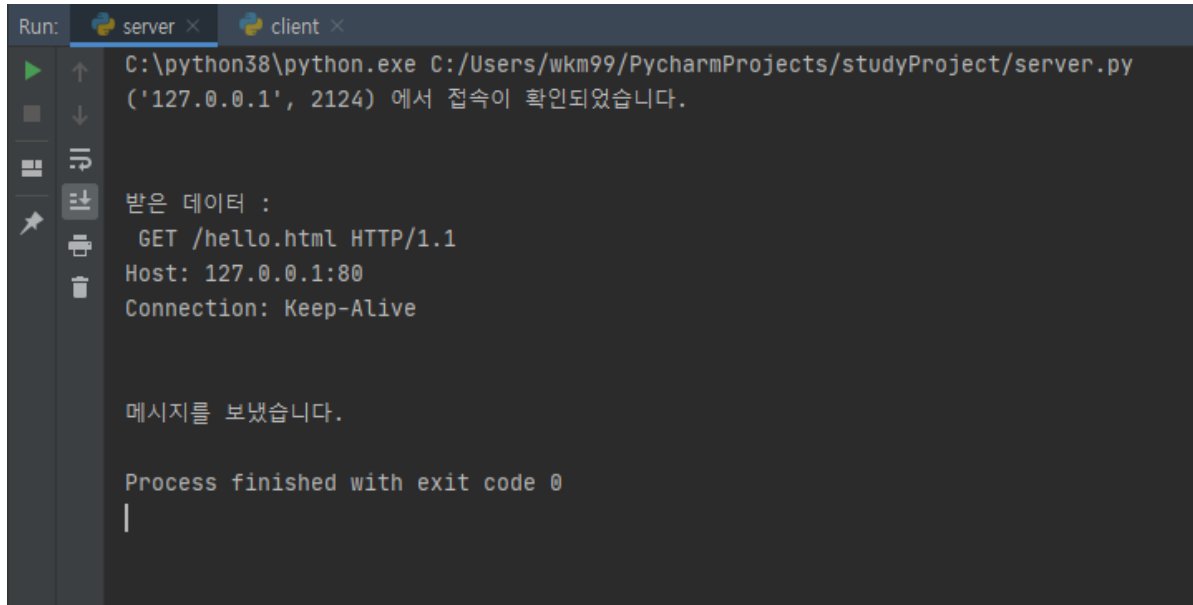
516	630.452506	127.0.0.1	127.0.0.1	HTTP	114 GET /hello.html HTTP/1.1
518	630.453274	127.0.0.1	127.0.0.1	HTTP	285 HTTP/1.1 200 OK (text/html)Co

```
name 516: 114 bytes on wire (912 bits), 114 bytes captured (912 bits) on interface \Device\NPF_{...}\Loopback
Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
Transmission Control Protocol, Src Port: 1079, Dst Port: 80, Seq: 1, Ack: 1, Len: 70
Hypertext Transfer Protocol
```

이후에는 5가지 경우로 나뉘어지는 분기가 이어지기 때문에 각각 따로 서술하겠습니다.

### 3.1) GET 메소드 - 2XX 응답.

서버 출력:

A screenshot of the PyCharm Run window. The top bar shows two tabs: 'server' and 'client'. The 'server' tab is active. The console output shows the execution of a Python script: 'C:\python38\python.exe C:/Users/wkm99/PycharmProjects/studyProject/server.py ('127.0.0.1', 2124) 에서 접속이 확인되었습니다.' followed by a separator line. Below this, it says '받은 데이터 :', followed by an indented line 'GET /hello.html HTTP/1.1'. The next two lines are 'Host: 127.0.0.1:80' and 'Connection: Keep-Alive'. Another separator line follows, then '메시지를 보냈습니다.' and another separator line. The final line is 'Process finished with exit code 0'.

클라이언트 출력:

```
Run: server x client x
C:\python38\python.exe C:/Users/wkm99/PycharmProjects/studyProject/client.py
연결 확인 되었습니다.
GET 메소드- 응답 2XX 하려면 1
GET 메소드- 응답 4XX 하려면 2
POST 메소드- 응답 4XX 하려면 3
HEAD 메소드- 응답 2XX 하려면 4
HEAD 메소드- 응답 4XX 하려면 5
원하는 동작을 입력해주세요: 1

보낸메시지:
GET /hello.html HTTP/1.1
Host: 127.0.0.1:80
Connection: Keep-Alive

메시지를 전송했습니다.

받은 데이터 : HTTP/1.1 200 OK
Date: 2022-05-03 14:56:36.247709
Content-Type: text/html
Content-Length: 1
<!DOCTYPE html>
<html>
<head>
<title>Page Title</title>
</head>
<body>

<h1>This is a Heading</h1>
<p>This is a paragraph.</p>

</body>
</html>

GET한 파일이 저장되었습니다.

Process finished with exit code 0
```

### wireShark 출력 결과물:

24	5.668889	127.0.0.1	127.0.0.1	HTTP	114 GET /hello.html HTTP/1.1
26	5.670025	127.0.0.1	127.0.0.1	HTTP	285 HTTP/1.1 200 OK (text/html)Continuation

Frame 24: 114 bytes on wire (912 bits), 114 bytes captured (912 bits) on interface \Device\NPF\_{...} id 0  
Null/Loopback

Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1  
Transmission Control Protocol, Src Port: 2124, Dst Port: 80, Seq: 1, Ack: 1, Len: 70

0000 02 00 00 00 45 00 00 6e 9c 45 40 00 80 06 00 00 ....E..n.E@.....

## 동작 설명 :

client가 GET method를 request에 작성하여 전송합니다. 이때 요청하는 파일은 hello.html이며, 이는 서버가 가지고 있는 파일입니다.

중요한 점은 GET method를 작성할 때 makeFile이라는 변수를 1로 바꿉니다.

request 메시지를 받은 server는 해당하는 hello.html 파일을 read합니다. 이때 hello.html은 엄연히 서버가 가지고 있는 파일이기에 IO Error가 발생하지 않습니다.

따라서 정상적으로 작동하여 200 OK를 response에 작성하여 전송합니다. 이때 hello.html 또한 같이 전송합니다.

클라이언트는 해당 response message를 받은 후에, 그것을 출력합니다.



그후에 client는 makeFile이 1이기에 if문에서 분기됩니다. 여기서 response message를 분석하여 200 OK인지 확인합니다.

200 OK라면 response message의 body 부분에 전송된 hello.html의 내용을 변환하여

**client\_hello.html** 파일을 만듭니다.

```
1  <!DOCTYPE html>
2
3  <html>
4  <head>
5    <title>Page Title</title>
6  </head>
7  <body>
8
9    <h1>This is a Heading</h1>
10   <p>This is a paragraph.</p>
11
12 </body>
13 </html>
14
15
```

이렇게 함으로써 GET에 대한 response 메시지가 200 OK인 경우, 제대로 파일이 같이 전송되는 것을 확인할 수 있습니다.

 client.py	2022-05-03 오후 2:30	Python 원본 파일	2KB
 client_hello.html	2022-05-03 오후 2:27	Chrome HTML D...	1KB

이 전송된 파일이 클라이언트 위치에서 생성되는 것을 확인할 수 있었습니다.

## 3.2) GET 메소드 - 4XX 응답

서버 출력 결과물:

```
Run: server x client x
C:\python38\python.exe C:/Users/wkm99/PycharmProjects/studyProject/server.py
('127.0.0.1', 1131) 에서 접속이 확인되었습니다.

받은 데이터 :
GET /notExisting.html HTTP/1.1
Host: 127.0.0.1:80
Connection: Keep-Alive

메시지를 보냈습니다.

Process finished with exit code 0
```

클라이언트 출력 결과물:

```
Run: server x client x
C:\python38\python.exe C:/Users/wkm99/PycharmProjects/studyProject/client.py
연결 확인 되었습니다.
GET 메소드- 응답 2XX 하려면 1
GET 메소드- 응답 4XX 하려면 2
POST 메소드- 응답 4XX 하려면 3
HEAD 메소드- 응답 2XX 하려면 4
HEAD 메소드- 응답 4XX 하려면 5
원하는 동작을 입력해주세요: 2

보낸메시지:
GET /notExisting.html HTTP/1.1
Host: 127.0.0.1:80
Connection: Keep-Alive

메시지를 전송했습니다.

받은 데이터 : HTTP/1.1 404 NOT FOUND
Date: 2022-05-02 10:28:38.024665
Content-type: text/html
Content-length: 0

Process finished with exit code 0
```

wireShark 캡처 결과물:



1219	1560.578681	127.0.0.1	127.0.0.1	HTTP	120	GET /notExisting.html HTTP/1.1
1221	1560.579300	127.0.0.1	127.0.0.1	HTTP	146	HTTP/1.1 404 NOT FOUND

```

> Frame 1219: 120 bytes on wire (960 bits), 120 bytes captured (960 bits) on interface \Device\NPF_{Loopback},
> Null/Loopback
> Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
> Transmission Control Protocol, Src Port: 1131, Dst Port: 80, Seq: 1, Ack: 1, Len: 76
> Hypertext Transfer Protocol

```

#### 동작 설명 :

client가 GET method를 request에 작성하여 전송합니다. 이때 요청하는 파일은 notExisting.html이며, 이는 서버에 없는 파일입니다.

위에 설명한 GET method와 마찬가지로 makeFile이라는 변수를 1로 바꿉니다.

server에서는 client가 요청한 notExisting.html 파일을 열고자하나, 해당 파일은 존재하지 않는 파일이기에 IO Error가 발생합니다.

이때 IO Error를 따로 분기시키는 try except문이 발동하여 server는 404 NOT FOUND를 response에 작성하여 전송합니다.

클라이언트는 해당 response message를 받은 후에, 그것을 출력합니다.

그후에 client는 makeFile이 1이기에 if문에서 분기됩니다.

해당 if문에서 response message를 나누어서 200 OK인지 확인합니다.

200 OK가 아닌 404 NOT FOUND이기에 별다른 동작없이 종료됩니다.

### 3.3) POST 메소드 - 4XX 응답

서버 출력 결과물:

```
Run: server x client x
C:\python38\python.exe C:/Users/wkm99/PycharmProjects/studyProject/server.py
('127.0.0.1', 1134) 에서 접속이 확인되었습니다.

받은 데이터 :
POST / HTTP/1.1
Host: 127.0.0.1:80
Connection: Keep-Alive

메시지를 보냈습니다.

Process finished with exit code 0
|
```

클라이언트 출력 결과물:

```
Run: server x client x
C:\python38\python.exe C:/Users/wkm99/PycharmProjects/studyProject/client.py
연결 확인 됐습니다.
GET 메소드- 응답 2XX 하려면 1
GET 메소드- 응답 4XX 하려면 2
POST 메소드- 응답 4XX 하려면 3
HEAD 메소드- 응답 2XX 하려면 4
HEAD 메소드- 응답 4XX 하려면 5
원하는 동작을 입력해주세요: 3

보낸메시지:
POST / HTTP/1.1
Host: 127.0.0.1:80
Connection: Keep-Alive

메시지를 전송했습니다.

받은 데이터 : HTTP/1.1 405 Method not allowed
Date: 2022-05-02 10:31:43.624692
Content-type: text/html
Content-length: 0

Process finished with exit code 0
|
```

wireShark 캡처 결과물:

1358	1746.179955	127.0.0.1	127.0.0.1	HTTP	105	POST / HTTP/1.1
1360	1746.180154	127.0.0.1	127.0.0.1	HTTP	155	HTTP/1.1 405 Method not allowed

```

<
> Frame 1358: 105 bytes on wire (840 bits), 105 bytes captured (840 bits) on interface \Device\NPF_{Loopback}, id
> Null/Loopback
> Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
> Transmission Control Protocol, Src Port: 1134, Dst Port: 80, Seq: 1, Ack: 1, Len: 61
> Hypertext Transfer Protocol

```

#### 동작 설명 :

client가 POST method를 request에 작성하여 전송합니다.

저는 client가 server에 생성하려고 하는 내용이 올바른 내용인지 의문이 들었습니다.

만약 client가 생성하는 내용이 신뢰할 수 있는 올바른 내용인지 확인할 수 있었다면 server에서도 POST 메소드를 받아들이고 올바르게 작동하도록 할 수 있었겠지만, 시간과 구현의 부족함으로 인해 거기까지 해내지는 못했습니다.

따라서 POST 메소드는 405 Method not allowed로 응답하도록 했으며, 어차피 무조건 405가 응답되기에 request 메시지에서 body 부분은 비어있도록 했습니다.

### 3.4) HEAD 메소드 - 2XX 응답

#### 서버 출력 결과물:

```

Run: server x client x
C:\python38\python.exe C:/Users/wkm99/PycharmProjects/studyProject/server.py
('127.0.0.1', 1141) 에서 접속이 확인되었습니다.

받은 데이터 :
HEAD /hello.html HTTP/1.1
Host: 127.0.0.1:80
Connection: Keep-Alive

메시지를 보냈습니다.

Process finished with exit code 0
|

```

#### 클라이언트 출력 결과물:

```
Run: server x client x
C:\python38\python.exe C:/Users/wkm99/PycharmProjects/studyProject/client.py
연결 확인 되었습니다.
GET 메소드- 응답 2XX 하려면 1
GET 메소드- 응답 4XX 하려면 2
POST 메소드- 응답 4XX 하려면 3
HEAD 메소드- 응답 2XX 하려면 4
HEAD 메소드- 응답 4XX 하려면 5
원하는 동작을 입력해주세요: 4

보낸메시지:
HEAD /hello.html HTTP/1.1
Host: 127.0.0.1:80
Connection: Keep-Alive

메시지를 전송했습니다.

받은 데이터 : HTTP/1.1 200 OK
Date: 2022-05-02 10:33:14.430738
Content-type: text/html
Content-length: 1

Process finished with exit code 0
```

#### wireShark 캡처 결과물:

1430	1836.984907	127.0.0.1	127.0.0.1	HTTP	115	HEAD /hello.html HTTP/1.1
1432	1836.985710	127.0.0.1	127.0.0.1	HTTP	139	HTTP/1.1 200 OK

<

> Frame 1430: 115 bytes on wire (920 bits), 115 bytes captured (920 bits) on interface \Device\NPF\_{Loopback

> Null/Loopback

> Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1

> Transmission Control Protocol, Src Port: 1141, Dst Port: 80, Seq: 1, Ack: 1, Len: 71

> Hypertext Transfer Protocol

#### 동작 설명 :

client가 HEAD method를 request에 작성하여 전송합니다. 이때 요청하는 파일은 hello.html이며, 이는 서버가 가지고 있는 파일입니다.

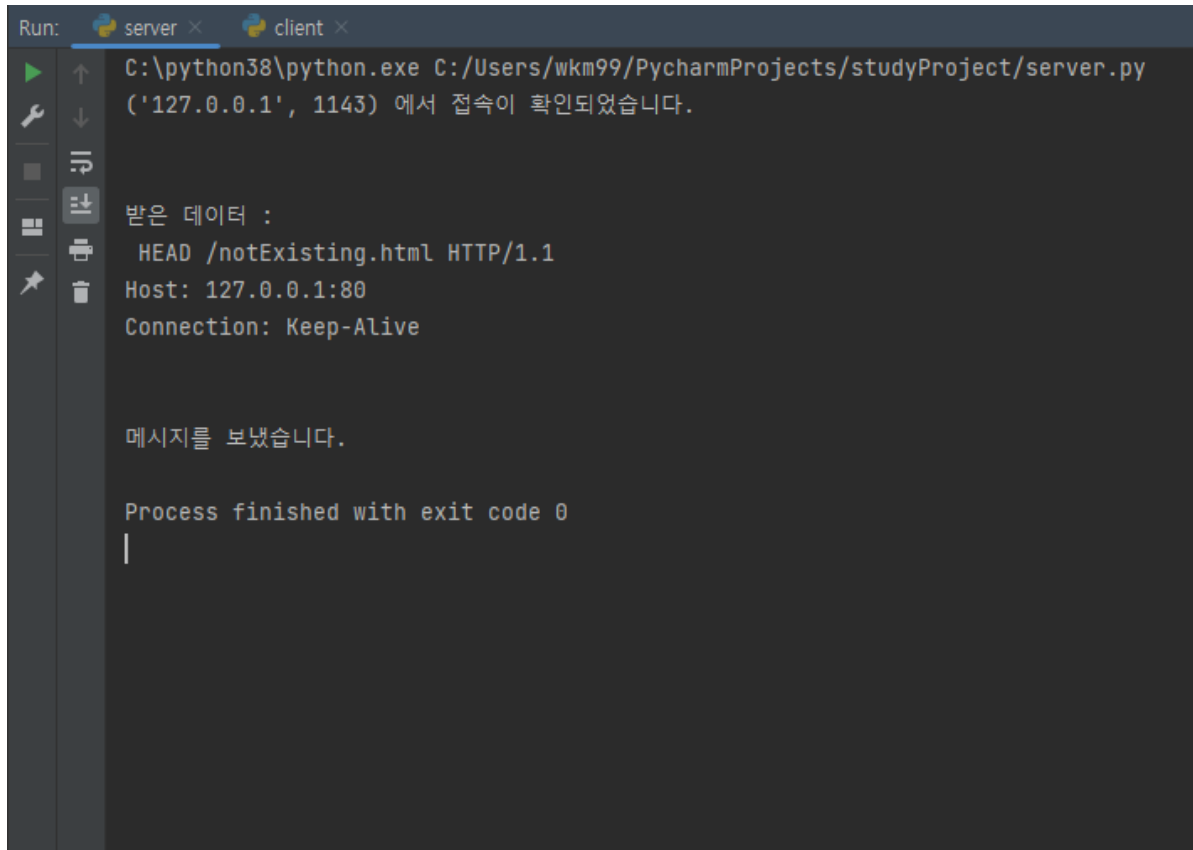
서버에서는 이 파일을 찾아 열어본 후에, GET할 때와 마찬가지로 response 메시지를 준비합니다.

이때, HEAD 메소드이기 때문에 response 헤더만 전송하며, 실질적인 파일의 내용은 이어붙이지 않습니다.

GET과는 다르게 HEAD 메소드는 response 헤더만 요구하는 메소드이기 때문입니다.

### 3.5) HEAD 메소드 - 4XX 응답

서버 출력 결과물:



The screenshot shows a PyCharm Run window with two tabs: 'server' and 'client'. The 'server' tab is active, displaying the output of a Python script. The output text is as follows:

```
Run: server x client x
C:\python38\python.exe C:/Users/wkm99/PycharmProjects/studyProject/server.py
('127.0.0.1', 1143) 에서 접속이 확인되었습니다.

받은 데이터 :
HEAD /notExisting.html HTTP/1.1
Host: 127.0.0.1:80
Connection: Keep-Alive

메시지를 보냈습니다.

Process finished with exit code 0
|
```

클라이언트 출력 결과물:

```
Run: server x client x
C:\python38\python.exe C:/Users/wkm99/PycharmProjects/studyProject/client.py
연결 확인 되었습니다.
GET 메소드- 응답 2XX 하려면 1
GET 메소드- 응답 4XX 하려면 2
POST 메소드- 응답 4XX 하려면 3
HEAD 메소드- 응답 2XX 하려면 4
HEAD 메소드- 응답 4XX 하려면 5
원하는 동작을 입력해주세요: 5

보낸메시지:
HEAD /notExisting.html HTTP/1.1
Host: 127.0.0.1:80
Connection: Keep-Alive

메시지를 전송했습니다.

받은 데이터 : HTTP/1.1 404 NOT FOUND
Date: 2022-05-02 10:34:17.964304
Content-type: text/html
Content-length: 0

Process finished with exit code 0
```

### wireShark 캡처 결과물:

1480	1900.518843	127.0.0.1	127.0.0.1	HTTP	121	HEAD /notExisting.html HTTP/1.1
1482	1900.519469	127.0.0.1	127.0.0.1	HTTP	146	HTTP/1.1 404 NOT FOUND

<

> Frame 1480: 121 bytes on wire (968 bits), 121 bytes captured (968 bits) on interface \Device\NPF\_{...} Null/Loopback

> Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1

> Transmission Control Protocol, Src Port: 1143, Dst Port: 80, Seq: 1, Ack: 1, Len: 77

> Hypertext Transfer Protocol

### 동작 설명 :

client가 HEAD method를 request에 작성하여 전송합니다. 이때 요청하는 파일은 notExisting.html이며, 이는 서버가 가지고 있지 않은 파일입니다.

서버에서는 이 파일을 찾아 열려고 하나 파일이 존재하지 않기 때문에 IO Error가 발생합니다.

이때 try except문으로 인해 분기되어 server는 404 NOT FOUND를 response에 작성하여 전송합니다.

