

# 컴퓨터네트워크 중간 대체 프로젝트

소프트웨어학과  
20181320  
이정현

# 동작환경

mac os monterey 12.2.1

LocalHost

python3

pycharm

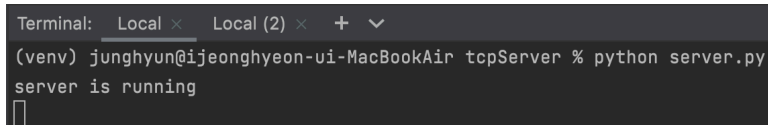
wireshark 3.6.3

1대의 laptop에서 로컬호스트로 server, client 역할을 수행하였습니다.  
소스파일의 실행은 pycharm의 터미널에서 진행하였습니다.

소스코드는 server.py , client.py 두개입니다.

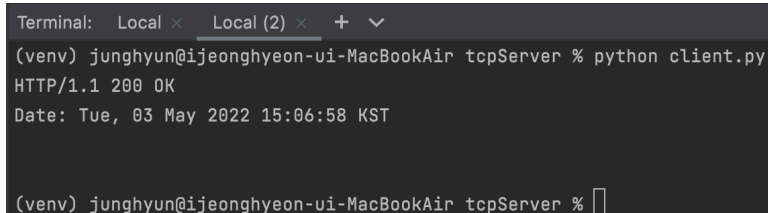
## 0.실행 방법

### 1. 서버실행

A terminal window with two tabs: 'Local' and 'Local (2)'. The active tab is 'Local (2)'. The prompt is '(venv) junghyun@ijeonghyeon-ui-MacBookAir tcpServer %'. The command 'python server.py' has been executed, and the output is 'server is running'.

```
Terminal: Local x Local (2) x + v
(venv) junghyun@ijeonghyeon-ui-MacBookAir tcpServer % python server.py
server is running
```

### 2.클라이언트 실행

A terminal window with two tabs: 'Local' and 'Local (2)'. The active tab is 'Local (2)'. The prompt is '(venv) junghyun@ijeonghyeon-ui-MacBookAir tcpServer %'. The command 'python client.py' has been executed, and the output is 'HTTP/1.1 200 OK' followed by 'Date: Tue, 03 May 2022 15:06:58 KST'.

```
Terminal: Local x Local (2) x + v
(venv) junghyun@ijeonghyeon-ui-MacBookAir tcpServer % python client.py
HTTP/1.1 200 OK
Date: Tue, 03 May 2022 15:06:58 KST

(venv) junghyun@ijeonghyeon-ui-MacBookAir tcpServer %
```

# 1. 클라이언트의 요청메세지

Get과 Head 메소드의 포맷 / Post와 Put 메소드의 포맷은 각각 동일하고  
Post, Put 메소드는 name의 입력으로 문자(숫자x)만을 가능하다고 가정하였습니다.

- 클라이언트 소켓을 생성하고, request\_message를 인자로 받아 서버로 전송.

```
def create_socket_and_send(request_message):  
    #클라이언트 소켓 생성  
    clientSock = socket(AF_INET, SOCK_STREAM)  
    clientSock.connect((serverName, serverPort))  
    #서버에 요청메세지 전송  
    clientSock.send(request_message.encode('utf-8'))  
  
    #응답 확인  
    recieve_message = clientSock.recv(65535)  
    print(recieve_message.decode())  
  
    clientSock.close()
```

요청메세지 예시

- 정상적인 요청메세지 포맷

```
#post 요청메세지  
Post_request = 'POST /index.html HTTP/1.1\r\n'  
Post_request += 'Host:127.0.0.1:12345\r\n'  
Post_request += 'Content-Type: text/plain\r\n'  
Post_request += 'Content-Length: 2\r\n'  
Post_request += 'Connection: Keep-Alive\r\n\r\n'  
Post_request += '\t{\r\n'  
Post_request += '\tname: jh\r\n'  
Post_request += '\t}\r\n\r\n\r\n'
```

Header

Body

Post와 Put 메소드의 요청메세지는 헤더와 바디로 구성되며,

중간 파란색 박스에서 줄바꿈을 두 번 해서 헤더와 바디를 구분했습니다.

- 의도적으로 에러를 유도한 요청메세지 포맷

```
#post 요청메세지 404에러 유도 잘못된 URL
Post_request_404 = 'POST /indx.html HTTP/1.1\r\n'
Post_request_404 += 'Host:127.0.0.1:12345\r\n'
Post_request_404 += 'Content-Type: text/plain\r\n'
Post_request_404 += 'Content-Length: 2\r\n'
Post_request_404 += 'Connection: Keep-Alive\r\n\r\n'
Post_request_404 += "\t{\r\n"
Post_request_404 += '\tname: jh\r\n'
Post_request_404 += '\t}\r\n\r\n'

#post 요청메세지 400에러 유도
Post_request_400 = 'POST /index.html HTTP/1.1\r\n'
Post_request_400 += 'Host:127.0.0.1:12345\r\n'
Post_request_400 += 'Content-Type: text/plain\r\n'
Post_request_400 += 'Content-Length: 2\r\n'
Post_request_400 += 'Connection: Keep-Alive\r\n\r\n'
Post_request_400 += "\t{\r\n"
Post_request_400 += '\tname: 1234\r\n'
Post_request_400 += '\t}\r\n\r\n' 숫자 입력
```

의도적으로 요청메세지를 변형해 에러를 유도했습니다.

404에러는 잘못된 URL을 입력하여 유도하였고,

숫자를 입력할 수 없다고 가정하여 숫자를 입력한 경우에는 400 에러를 발생시키도록 구현 하였습니다.

```
#Get 요청메세지
Get_request = 'GET /index.html HTTP/1.1\r\n'
Get_request += 'Host:127.0.0.1:12345\r\n'
Get_request += 'Connection: Keep-Alive\r\n\r\n'

#Get 요청메세지 404 잘못된 URL
Get_request_404 = 'GET /inex.html HTTP/1.1\r\n'
Get_request_404 += 'Host:127.0.0.1:12345\r\n'
Get_request_404 += 'Connection: Keep-Alive\r\n\r\n'
```

Get과 Head 메소드는 정상적인 경우와 URL을 의도적으로 다르게 입력하여 404 에러를 유도한 경우로 상황을 가정하였습니다.

## 2. 서버에서 클라이언트의 요청메세지 처리 방법

```
Terminal: Local x Local (2) x + v
(index) junghyun@ijeonghyeon-ui-MacBookAir tcpServer % python server1.py
server is running
['PUT', '/index.html', 'HTTP/1.1', 'Host:127.0.0.1:12345', 'Content-Type:', 'text/plain', 'Content-Length:', '3', 'Connection:', 'Keep-Alive', '{', 'name:', 'kia', '}']
```

클라이언트의 요청 메시지를 split메소드로 쪼개서 배열에 담아 분석합니다.  
제가 구현한 서버에서는 index 0번 , 1번 , 2번 , 11번, 12번 구문을 주요하게 분석했습니다.

```
def POST_PUT(request_message):
    now = datetime.datetime.now().strftime("Date: %a, %d %b %Y %H:%M:%S KST")
    # 주소가 잘못된 경우
    if (request_headers[2] != 'HTTP/1.1'):
        message = 'HTTP/1.1 505 HTTP Version Not Supported\r\n'
        message += now + '\r\n\r\n'
        return message
    if (request_headers[1] != '/index.html'):
        message = 'HTTP/1.1 404 Not Found\r\n'
        message += now + '\r\n\r\n'
        return message
    # post의 name은 숫자가 올 수 없다고 가정.
    # name의 인자로 숫자가 들어온 경우
    try:
        int(request_headers[12])
        message = 'HTTP/1.1 400 Bad Request\r\n'
        message += now + '\r\n\r\n'
        message += request_headers[10] + '\r\n\t'
        message += 'message:' + ' '
        message += 'name must be String' + '\r\n'
        message += request_headers[13] + '\r\n\r\n'
        return message
    except ValueError:
        pass
    # 코드에 문제가 없으므로 200 OK 리턴.
    message = 'HTTP/1.1 200 OK\r\n'
    message += request_headers[4] + request_headers[5] + '\r\n'
    message += request_headers[6] + request_headers[7] + '\r\n'
    message += now + '\r\n\r\n'
    message += request_headers[10] + '\r\n\t'
    message += request_headers[11] + ' '
    message += request_headers[12] + '\r\n'
    message += request_headers[13] + '\r\n\r\n'
    return message
```

index 0번: 클라이언트가 메소드를 요청했는지 확인

index 1번: 클라이언트가 유효한 url을 요청했는지 확인

index 2번: 서버가 규정한 http 버전에 맞게 요청했는지를 모든 메서드 공통적으로 확인

index 12번: 요청 body를 추가로 요하는 put, post 메서드에서 요청 body의 유효성을 확인. (제가 구현한 서버에서는 name으로 값으로 숫자를 받지 않습니다)

이런 방식으로 클라이언트의 요청 메시지를 분석하여, 서버측의 응답 메시지를 작성해 클라이언트 쪽으로 전송합니다.

### 3. HTTP method 실행 결과 (클라이언트측 python 실행결과 + wireshark 캡처)

서버에서는 별도의 출력이 없습니다.

#### GET 200 OK

```
HTTP/1.1 200 OK
Date: Mon, 02 May 2022 03:54:23 KST
```

298	204.959543	127.0.0.1	127.0.0.1	HTTP	56	HTTP/1.1 200 OK
-----	------------	-----------	-----------	------	----	-----------------

#### GET 404 Not Found

```
HTTP/1.1 404 Not Found
Date: Mon, 02 May 2022 03:55:12 KST
```

353	253.599570	127.0.0.1	127.0.0.1	HTTP	129	GET /inex.html HTTP/1.1
357	253.600227	127.0.0.1	127.0.0.1	HTTP	56	HTTP/1.1 404 Not Found

#### POST 200 OK

```
(venv) junghyun@ijeonghyeon-ui-MacBookAir tcpServer % python client1.py
HTTP/1.1 200 OK
Content-Type:text/plain
Content-Length:2
Date: Mon, 02 May 2022 03:51:33 KST

{
    name: jh
}
```

No.	Time	Source	Destination	Protocol	Length	Info
17	34.494706	127.0.0.1	127.0.0.1	HTTP	197	POST /index.html HTTP/1.1 (text/plain)Cont
19	34.495547	127.0.0.1	127.0.0.1	HTTP	174	HTTP/1.1 200 OK (text/plain)Continuation

#### POST 400 Bad Request

```
HTTP/1.1 400 Bad Request
Date: Mon, 02 May 2022 03:53:23 KST

{
    message: name must be String
}
```

197	144.513036	127.0.0.1	127.0.0.1	HTTP	199	POST /index.html HTTP/1.1 (text/plain)Cont
200	144.513712	127.0.0.1	127.0.0.1	HTTP	56	HTTP/1.1 400 Bad Request



## POST 404 Not Found

```
(venv) junghyun@ijeonghyeon-ui-MacBookAir tcpServer % python client1.py
HTTP/1.1 404 Not Found
Date: Mon, 02 May 2022 03:52:35 KST
```

107	96.101557	127.0.0.1	127.0.0.1	HTTP	196	POST /indx.html HTTP/1.1 (text/plain)Conti
110	96.102317	127.0.0.1	127.0.0.1	HTTP	56	HTTP/1.1 404 Not Found

## PUT 200 OK

```
HTTP/1.1 200 OK
Content-Type:text/plain
Content-Length:3
Date: Mon, 02 May 2022 03:56:53 KST
```

```
{
    name: kia
}
```

492	354.710627	127.0.0.1	127.0.0.1	HTTP	197	PUT /index.html HTTP/1.1 (text/plain)Conti
494	354.711209	127.0.0.1	127.0.0.1	HTTP	175	HTTP/1.1 200 OK (text/plain)Continuation

## PUT 505 HTTP Version Not Supported

```
HTTP/1.1 505 HTTP Version Not Supported
Date: Tue, 03 May 2022 02:44:10 KST
```

17	42.725602	127.0.0.1	127.0.0.1	HTTP	197	PUT /index.html HTTP/1.0 (text/plain)Conti
20	42.726142	127.0.0.1	127.0.0.1	HTTP	56	HTTP/1.1 505 HTTP Version Not Supported

## wireshark log

Wireshark network traffic capture showing HTTP requests and responses on Loopback: lo0. The capture includes a POST request that fails with 404 Not Found, followed by a PUT request that succeeds with 200 OK, and another PUT request that fails with 505 HTTP Version Not Supported.

No.	Time	Source	Destination	Protocol	Length	Info
17	34.494706	127.0.0.1	127.0.0.1	HTTP	197	POST /index.html HTTP/1.1 (text/plain)Conti
19	34.495547	127.0.0.1	127.0.0.1	HTTP	174	HTTP/1.1 200 OK (text/plain)Continuation
107	96.101557	127.0.0.1	127.0.0.1	HTTP	196	POST /indx.html HTTP/1.1 (text/plain)Conti
110	96.102317	127.0.0.1	127.0.0.1	HTTP	56	HTTP/1.1 404 Not Found
197	144.513036	127.0.0.1	127.0.0.1	HTTP	199	POST /index.html HTTP/1.1 (text/plain)Conti
200	144.513712	127.0.0.1	127.0.0.1	HTTP	56	HTTP/1.1 400 Bad Request
295	204.951280	127.0.0.1	127.0.0.1	HTTP	130	GET /index.html HTTP/1.1
298	204.959543	127.0.0.1	127.0.0.1	HTTP	56	HTTP/1.1 200 OK
353	253.599570	127.0.0.1	127.0.0.1	HTTP	129	GET /inex.html HTTP/1.1
357	253.600227	127.0.0.1	127.0.0.1	HTTP	56	HTTP/1.1 404 Not Found
492	354.710627	127.0.0.1	127.0.0.1	HTTP	197	PUT /index.html HTTP/1.1 (text/plain)Conti
494	354.711209	127.0.0.1	127.0.0.1	HTTP	175	HTTP/1.1 200 OK (text/plain)Continuation

No.	Time	Source	Destination	Protocol	Length	Info
5	0.000638	127.0.0.1	127.0.0.1	HTTP	197	PUT /index.html HTTP/1.0 (text/plain)Conti
7	0.001647	127.0.0.1	127.0.0.1	HTTP	175	HTTP/1.1 200 OK (text/plain)Continuation
17	42.725602	127.0.0.1	127.0.0.1	HTTP	197	PUT /index.html HTTP/1.0 (text/plain)Conti
20	42.726142	127.0.0.1	127.0.0.1	HTTP	56	HTTP/1.1 505 HTTP Version Not Supported