[TCP 기반 소켓 프로그래밍]

20182793, 빅데이터 경영통계학과, 김민식

1. 소스 파일

- TCP socket programming (main_server.py를 server code, main_client.py를 client code 활용)
- server의 host와 port(local host와 지정한 port)를 사용하여 Client 연결
- HTTP 기반 통신 진행(request와 response를 각각 Server, Client에서 받아서 진행)

2. 코드 설명

(1) Main.server.py(Server 코드)

1) 모듈 및 변수 불러오기

```
# 필요한 모듈 불러오기
import time
from socket import *

# 필요한 변수들 정리
IP, port = '127.0.0.1', 10000
status_pair = {100:'CONTINUE', 200:'OK', 400:'BAD_REQUEST', 404:'NOT_FOUND', 405:'METHOD_NOT_ALLOWED'}
db_data = {}
```

- Server 구현하는 과정에서 필요한 모듈 불러오기(Socket: 소켓 통신을 위한 모듈, Time: 현재 시간을 반환 모듈)
- 변수 의미는 아래와 같음

IP	Server 주소
Port	Server port 번호
Status_pair	HTTP 통신 간의 Status의 코드번호와 상태를 저장 Dictionary
Db_data	POST, PUT request 통해 저장할 데이터 DB

2) Response 생성 함수

```
# response 작성
def response_format(code, host, path, body=''):
    data = time.strftime("%a, %d %b %Y %H:%M:%S KST", time.localtime())

    start_line = "HTTP/1.1 {} {}".format(code, status_pair[code])
    head_line = "Host: {}/{}₩r₩nContent-Type: text/html₩r₩nConnection: keep-alive₩r₩nContent-Length: {}₩r₩nDate: {}".format(host, path, len(body),
data)
    response = start_line +"₩r₩n" + head_line + "₩r₩n₩n" + body
    return response
```

- server의 HTTP 생성함수를 총 3단계를 나눠서 정의한 이후 response 진행

[함수 매개변수] - Code: HTTP 통신 간의 Status의 코드번호 - Host: Server 주소, path: server 주소(POST, PUT의 create, update 혹은 파일 x생성 주소) - body: 통신할 내용 정보(default는 '' 공백), head를 제외한 해당하는 본문
[내용] - Start line => Response의 Start line 부분(HTTP 버전과 통신 간 상태 결과) - head_line => Response의 Head line 부분(HTTP 내의 정보 - content type 및 내용, 전달된 단어 수, Host 주소, 통신 시간) - body => Response의 Body 부분, 소켓 통신 간에 전달하고자 하는 실질적인 값

3) Request 처리 함수'

```
# Client에서 온 Request를 처리하는 함수 정의(완료)
def change_response(method, url, body):

    # 0. 읽어드린 url이 잘못된 경우 에러 발생
    if '?/?' not in url:
        host = url.split('/')[0]
        path = url.split('/')[1]
    else:
        body = "Invalid Address(url)"
        response = response_format(404, "?", "?", body)
        return response

    # 1. 요청한 Request에 4가지 해당하지 않을 경우 에러 발생
    if method not in ['HEAD','GET','POST', 'PUT']:
        body = "Method not allowed"
        response = response_format(405, host, path, body = body)

    # 2. 허용하지 않는 host일 경우 에러 발생
    elif host != IP:
        body = "Not allowed host: {}".format(host)
        response = response_format(400, host, path, body = body)

    # 3. method별 데이터 읽기
    else:
        # PATH 내의 create이면 method가 post이면 값을 반환
        if path == "create":
            if method == "POST":
                response = post(host, path, body)
        # PATH 내의 create이지만 method가 대응되는 POST가 아닐 경우는 경고 메시지 반환(에러)
            else:
                body = "Select the appropriate method(Create mode => POST)"
                response = response_format(404, host, path, body = body)

        # PATH 내의 update이면 method가 put이면 값을 반환
        elif path == "update":
            if method == "PUT":
                response = put(host, path, body)
        # PATH 내의 create이지만 method가 대응되는 put 아닐 경우는 경고 메시지 반환(에러)
            else:
                body = "Select the appropriate method(Updata mode => PUT)"
                response = response_format(404, host, path, body = body)

        # HEAD를 읽을 때
        elif method == "HEAD":
            response = head(host, path, body)

        # GET을 읽을 때
        elif method == "GET":
            response = get(host, path, body)

    return response
```

- HTTP Protocol로 들어온 Request 해석 후 Response 반환하기 위한 과정(Response 함수 연동)

=> method: 수행할 동작[HEAD, GET, POST, PUT 중 하나] / url: server 통신 간의 주소 / body(default: 공백): 통신 할 내용)

=> 각 조건을 진행한 이후 결과값 response 변수는 주어진 Request에 대응되는 서버의 Response 형식을 의미

함수 내의 부분	의미
첫번째 조건문/IF-else (URL Issue)	- 읽어 드린 URL이 이상한 경우(형식이 벗어난 경우, 형식: (host)/(URL 뒷부분)) - "Invalid Address(URL)" 문구 반환 및 URL의 부분을 "?"로 채움 - 구현 내용[1]: 공통-응답 400(잘못된 요청) –> URL issue
두번째 조건문/IF (Method issue)	- 요청한 Request Action이 주어진 4가지 Method(HEAD, GET, POST, PUT)에 해당하지 않는 경우 - "Mention not allowed" 문구 반환 - 구현 내용[2]: 공통-응답 405(허용되지 않는 메소드)
두번째 조건문/ELIF (Host issue)	- 허용하지 않는 Host가 들어온 경우(- "Not allowed hots"와 허용하지 않는 Host를 반환 - 구현 내용[3]: 공통-응답 400(잘못된 요청) –> Host issue
두번째 조건문/ELSE (Method 읽기)	- 이외의 경우는 문제가 없다고 (잠정) 판단 후 Method를 읽고, 진행함 - POST, PUT Method에 대응되는 URL 내의 정보 Create와 Update 존재여부 확인 - 각 Path(Create, Update)에 대응되는 Method에 해당하지 않는 경우 적절한 Method 고르도록 경고 메시지 - 구현 내용[4], [5]: POST-응답 404(NOT FOUND), PUT-응답 404(NOT FOUND) –>Method Pair issue

4) HEAD Method 함수

```
# Method 정의
## 1. head 응답 함수 : response에 header부분만 표시
def head(host, path, body):
    response = response_format(100, host, path, body = '')
    return response
```

- Method 함수 中 HEAD(start line과 head line 합친 부분, Body 제외)부분만을 반환(Head에 대응 RESPONSE 반환)

- 구현 내용 [6]: Head-응답 100(계속)

5) GET Method 함수

```
## 2. Get 응답 함수 : Response에 header + body 부분 표시
def get(host, path, body):
    # db 내에 정보가 있을 경우 그 정보를 반환
    if db_data:
        body = "The data in DB is as follows. ₩n₩r"
        for key, value in db_data.items():
            body += key + ':' + value + '₩r₩n'
        body = body[:-2]
    # db 내에 정보가 없을 경우 정보가 없다는 문구 반환
    else:
        body = "No data in DB"
    response = response_format(200, host, path, body)
    return response
```

- Method 함수 中 GET를 반환(Head에 대응하는 RESPONSE 반환)

- DB내에 정보가 저장되어 있는 경우는 "The Data in DB is as follows:"문구와 함께 저장되어 key-value형식의 내용을 "key: value" 형식으로 출력 진행, 정보가 저장되어 있지 않는 경우는 "No data in DB" 문구를 반환

- 구현 내용 [7]: GET-응답 200(성공)

6) POST Method 함수

- Method 함수 中 POST를 반환, DB내에 정보를 저장하기 위한 함수를 구현

- body의 형식은 "Key1: value1&key2: value2"으로 구성되어, key와 value 대응 관계를 ":"로 표현하고 각각의 데이터 요소를 "&"로 구분 진행

함수 내의 부분	의미
첫번째 조건문/IF (No Body Issue)	- 추가하고자 하는 내용(Body)가 없는 경우(실제로는 100을 진행 후 입력 대기를 한다고 함) - 구현 내용[8]: POST-응답 400(잘못된 요청) -> No body issue
두번째 조건문/ELSE 이후(내부 첫 IF문) (Key-value format issue)	- "key: value"의 형식이 맞지 않는 경우(key와 value 대응관계를 나타내는 ":" 없을 경우) - "Key-value format is not valid." 문구 출력과 함께 error 코드 전송 - 구현 내용[9]: POST-응답 400(잘못된 요청) –> Key value format issue
두번째 조건문 /ELSE 후(내부 두번째 IF문) (Key-Error issue)	- Key가 적절하지 않은 경우(Key값이 공백 혹은 None 값일 경우) - "Key value does not exist or is blank." 문구 출력과 함께 error 코드 전송 - 구현 내용[10]: POST-응답 400(잘못된 요청) –> Key Error issue
두번째 조건문 /ELSE 이후(내부 두번째 ELIF문) (정상적인 경우)	- DB 내의 Key 값이 없는 경우, Key와 Value 값을 DB내의 저장 - 이후 값이 정상적으로 저장했다고, "Post Successfully OK" 문구 반환 - 구현 내용[11]: POST-응답 200(성공)
두번째 조건문 /ELSE 이후(내부 두번째 ELIF문) (Key-Reduplication issue)	- DB 내의 Key 값이 있어서 중복된 경우 - 키 값이 있음을 "The key already exists" 문구 출력과 error 코드 전송 - 구현 내용[12]: POST-응답 400 (잘못된 요청) -> Key-Reduplication issue

```python
## 3. post 응답 함수
def post(host, path, body):
    content, total = '', ''
    # 추가 해야할 내용이 아무것도 없을 때(body == 0), 오류 발생 변환
    if len(body) == 0 or body in ['', ' ']:
        response = response_format(400, host, path, body)
        return response

    # 추가 해야할 내용이 있다면, 인식 확인하였다는 점을 200으로 반환 / 추가 데이터 반환
    else:
        body_values = body.split('&')
        # 여러 개의 값들을 입력 하게 되면
        for body_value in body_values:

            # key: value 형식을 맞추지 않는 다면(key-value를 구분하는 : 없다면) 잘못된 요청으로 판단해 400 에러 발생
            if ":" not in body_value:
                body = "Key-value format is not valid."
                response = response_format(400, host, path, body)
                return response

            key, value = body_value.split(':')
            key = key.strip()
            value = value.strip()

            # key 값이 비워있거나 공백일 경우 400 에러 발생
            if key in [None, '', ' ']:
                body = "Key value does not exist or is blank."
                response = response_format(400, host, path, body)
                return response

            # key 값이 없는 경우 db에 내용 추가
            elif db_data.get(key) == None:
                db_data.setdefault(key, value)
                content = key + ':' + value + '₩r₩n'
                total += content

            # 같은 key 값을 저장한 있는 경우 400 에러 발생
            else:
                body = "The key already exists!"
                response = response_format(400, host, path, body)
                return response

        # 값이 정상적으로 정상적으로 저장되었을 경우 200, OK 반환
        body = "Post Suceessfully OK"
        total += body

        response = response_format(200, host, path, total)
    return response
```

7) PUT Method 함수

- Method 함수 中 PUT를 반환, DB내에 정보를 갱신하기 위한 함수를 구현

- body의 형식은 "Key1: value1&key2: value2"으로 구성되어, key와 value 대응 관계를 ":"로 표현하고 각각의 데이터 요소를 "&"로 구분 진행(PUT과 동일)

함수 내의 부분	의미
첫번째 조건문/IF (No Body Issue)	- 추가하고자 하는 내용(Body)가 없는 경우 - "There is no content. Please check" 문구 반환 및 Error 코드 전송 - 구현 내용[13]: PUT-응답 400(잘못된 요청) -> No body issue
두번째 조건문 /ELSE 후(내부 두번째 IF문) (Key-Error issue)	- Key가 적절하지 않은 경우(Key값이 공백 혹은 None 값일 경우) - "Key value does not exist or is blank." 문구 출력과 함께 error 코드 전송 - 구현 내용[14]: PUT-응답 400(잘못된 요청) –> Key Error issue
두번째 조건문 /ELSE 이후(내부 두번째 IF-ELSE문) (정상적인 경우)	- 앞선 Error 경우를 제외하고, update 진행할 경우 - 출력 문구는 "Update" 진행한 내용과 (Key와 변화된 내용) 그리고 "create" 진행한 내용을 출력 - 구현 내용[15]: PUT-응답: 200(성공)

```python
## 4. Put 응답 함수:
def put(host, path, body):
    old_dic, change_dic, new_dic = {}, {}, {}
    main_key, content = '', ''

    # update 해야 할 내용을 입력하지 않는다면(body = ''), 인식하지 못했다고 반환(오류) 400
    if len(body) == 0 or body in ['',' ']:
        body = "There is no content, Please check"
        response = response_format(400, host, path, body)
        return response

    # update 해야 할 내용을 입력했으면, 인식한 결과 반환(200) / error 제외
    else:
        body_values = body.split('&')
        for body_value in body_values:

            # key: value 형식을 맞추지 않는 다면(key-value를 구분하는 : 없다면) 잘못된 요청으로 판단해 400 에러 발생
            if ":" not in body_value:
                body = "Key-value format is not valid,"
                response = response_format(400, host, path, body)
                return response

            key, value = body_value.split(':')
            key = key.strip()
            value = value.strip()

            # key 값이 db에 있다면 update 진행
            if key in db_data:
                previous_data = db_data[key]
                db_data[key] = value
                old_dic.setdefault(key, previous_data)
                change_dic.setdefault(previous_data, value)

            # key 값이 db에 없다면 create 진행
            else:
                db_data.setdefault(key, value)
                new_dic.setdefault(previous_data, value)

        # update를 진행할 경우(key 값이 존재하고, 해당 값을 update) 하는 것에 대한 정보 전달 코드
        if len(change_dic):
            body += "₩n₩n₩rUpdate Data as follow:₩n₩r"

            for key, pre in old_dic.items():
                change_data = change_dic[pre]
                body += key + ": " + pre + " => " + change_data + "₩n₩r"

        # create_upbdata를 진행할 경우(key 값이 존재하지 않아 새롭게 만드는 것에 대한) 하는 것에 대한 정보 전달 코드
        if len(new_dic) !=0:
            body += "₩n₩rUpdate(New) Data as follow:₩n₩r"

            for key, new in new_dic.items():
                body += key + ": " + new + "(There's no key, New update)"

        body += "₩n₩rUpdate with new data Sucessfully OK ₩n₩r"
        body = body[:-2]
        response = response_format(200, host, path, body)
        return response
```

8) 서버 생성 & Client 통신

```python
# 서버를 연 이후 계속 유지(종료 조건이 오기 전까지)
while True:
    print("----------The Server is ready to receive (Host: {}, Port: {})----------".format(IP,port))

    client_socket, client_addr = server_socket.accept()   # Client connectino을 기다린 후 accept 진행
    request = client_socket.recv(4096).decode('utf-8')    # Client Request을 수신

    print("request:₩n₩r", request)                         # Request 수신 시각화

    ## Request에서  Response 으로 변환할 때의 필요한 내용 정리
    ## (method => 수행해야할 작업 / body => 내용 정보 url => host 및 adress 정보)
    request_part = request.split('₩n')
    url = request_part[1].split(':')[1][1:-1]

    method = request_part[0].split()[0]
    body = request_part[-1]

    # 종료조건(들어온 reqest 중 End가 발생할 경우 socekt 종료 진행)
    if method == "END":
        print("Complete all courses. socket finish")
        break

    response = change_response(method, url, body)         # reponse 생성 후 그 결과를 client 보냄
    client_socket.sendall(response.encode('utf-8'))
    print("send the response...₩n₩r")

# 종료 후 socket을 닫음
client_socket.close()
server_socket.close()
```

[1] 변수에 지정해둔 Host와 Port를 주소로 TCP 서버를 생성

[2] Client의 connection를 대기한(While의 무한 루프 활용) 이후 Accept 진행, Client의 Request를 수신

[3] Request에서 필요한 내용(method, body, url 정보)를 기반으로 대응 Response를 생성(Change-response 함수활용)

[4] 생성한 Response를 Client에 전송

[5] method 중 "END" 조건이 들어온 경우 server를 종료(socket 종료)

-

(2) Main.Client.py(Client 코드)

1) 모듈 및 변수 정의

```
## 필요한 모듈 정의
from socket import *
from datetime import datetime

host = '127.0.0.1'
port = 10000
case_num = 0
```

- Client 구현하는 과정에서 필요한 모듈 불러오기(Socket: 소켓 통신을 위한 모듈)

- 변수 의미는 아래와 같음

host	Server 주소
Port	Server port 번호
Case_num	Test case의 번호를 구분하기 위한 변수

2) Test Case 정의

```
# 테스트 데이터 경우 확인

test_case = [

    # 정상의 경우 예시
    {'url': '127.0.0.1/', 'method': 'HEAD', 'body': ''},          # 100 CONTINUTE(최초 연결 후 Head 확인) => HEAD 유효한 경우
    {'url': '127.0.0.1/', 'method': 'GET', 'body': ''},           # 200 OK => GET 유효한 경우

    ## 오류 예시
    {'url': '127.0.0.1??', 'method': 'GET', 'body': ''},          # 404 ERROR(url 주소가 잘못된 경우 => (PORT/주소))
    {'url': '127.0.0.1/', 'method': 'PRRT', 'body': ''},          # 405 ERROR(유효하지 않는 METHOD 입력 된 경우)
    {'url': '127.0.5.1/', 'method': 'GET', 'body': ''},           # 400 ERROR(허용 되지 않는 HOST인 경우)
    {'url': '127.0.0.1/create', 'method': 'PUT', 'body': ''},     # 404 ERROR(CREATE나 UPDATE 시 path가 매칭이 안된 경우)

    # POST 정상 / 오류 예시
    {'url': '127.0.0.1/create', 'method': 'POST', 'body': 'ID: 20182793&Subject: Computer_Network&Grade: A'}, # 200 OK(데이터 정상 입력)
    {'url': '127.0.0.1/', 'method': 'GET', 'body': ''},                    # 200 OK(DB 저장 내용 확인)
    {'url': '127.0.0.1/create', 'method': 'POST', 'body': ': 95'},         # 404 ERROR(Key 값이 유효하지 않는 경우 에러 발생)
    {'url': '127.0.0.1/create', 'method': 'POST', 'body': 'middle_exam70'}, # 400 ERROR(Key-value 쌍이 유효하지 않는 경우 에러 발생)
    {'url': '127.0.0.1/create', 'method': 'POST', 'body': 'ID: 20180434'},  # 400 ERROR(같은 key 값을 갖은 경우 에러 발생)
    {'url': '127.0.0.1/', 'method': 'HEAD', 'body': ''},                   # 100 CONTINUTE (Head 유효)

    # PUT 정상 / 오류 예시
    {'url': '127.0.0.1/update', 'method': 'PUT', 'body': 'ID: 20180434&final_exam: 90'}, # 200 OK(데이터 정상 업데이트 & 추가)
    {'url': '127.0.0.1/update', 'method': 'PUT', 'body': ''},                            # 400 Error(update 내용이 입력되지 않았을 경우)
    {'url': '127.0.0.1/update', 'method': 'PUT', 'body': 'retake_wheaterYes'},           # 400 ERROR(Key-value 쌍이 유효하지 않는 경우 에러 발생)
    {'url': '127.0.0.1/', 'method': 'END', 'body': ''}                                   # 종료조건
]
```

- Server에서 구현 Method에 대한 내용이 정상적으로 작동하기 위해 점검하는 내용

- 위에 사진과 같이 정의한 내용 순서대로 진행하였고, 정의한 Method 통신 방식을 최대한 반영 노력

3) Request 작성 함수

```
def request_formate(method, host, path, body=''):
    start_line = "{} / HTTP/1.1".format(method)
    head_line = "Host: {}/{}₩r₩nContent-Type: text/html₩r₩nConnection: keep-alive₩r₩nContent-Length: {}".format(host, path, len(body))
    response = start_line +"₩r₩n" + head_line + "₩r₩n₩n" + body
    return response
```

- Test case에서

[함수 매개변수]
- Method: HEAD, GET, POST, PUT 중 하나로 Action 정의
- Host: Server 주소, path: server 주소(POST, PUT의 create, update 혹은 파일 생성 주소)
- body: 통신할 내용 정보(default는 '' 공백), head를 제외한 해당하는 본문 / POST, PUT => 서버에서 생성하고자 하는 데이터

[내용]
- Start line => Request의 Start line 부분(HTTP 버전과 통신 간 상태 결과)
- head_line => Request의 Head line 부분(HTTP 내의 정보 - content type, connection 내용, Connection 전달되는 단어, Host 주소)
- body => Request의 Body 부분, 소켓 통신 간에 전달하고자 하는 실질적인 값

4) Client socket 생성 & Server 통신

```
for test in test_case:
    case_num +=1                                                    # 테스트 경우를 시각화 하기 위한 case_num 데이터

    print("### Test Case{}: {}".format(case_num, test))
    print("----------------------------------------")

    client_socket = socket(AF_INET, SOCK_STREAM)                    # Client socket 생성
    client_socket.connect((host, port))                             # variables에 설정된 주소의 서버로 연결을 요청

    method, url, body = test['method'], test['url'], test['body']   # request를 작성(필요한 내용을 test에서 분할)

    # request 내용 중 올바른 주소가 아닐 경우(올바른 주소 판단 여부: "(host)/address" 구조) 처리하기 위한 함수
    if '/' in url:
        host_num = url.split('/')[0]
        path_num = url.split('/')[1]
    else:
        host_num = "?"
        path_num = "?"

    print("Sending Request...₩n₩r")

    request = request_formate(method, host_num, path_num, body)     # 각 상황에 맞는 request 생성
    client_socket.send(request.encode('utf-8'))                     # request를 server로 전송

    print("Receving Response...₩n₩r")
    data = client_socket.recv(4096).decode('utf-8')                 # request에 대응하는 server의 response

    print("Response data:₩n₩r", data)
    client_socket.close()                                           # client는 한 번 request와 response를 주고 받고나면 종료(Server는 계속 열림)

    print("₩n₩rFinish...₩n₩r")
```

[1] TCP client의 socket을 생성 후 변수에 저장한 주소(host, port)를 가지고 서버의 연결을 요청

[2] Test case에서 주어진 상황에 맞는 request를 생성 후 server에 전송

[3] Server에서 보낸 Response 받을 이후 socket 통신 종료

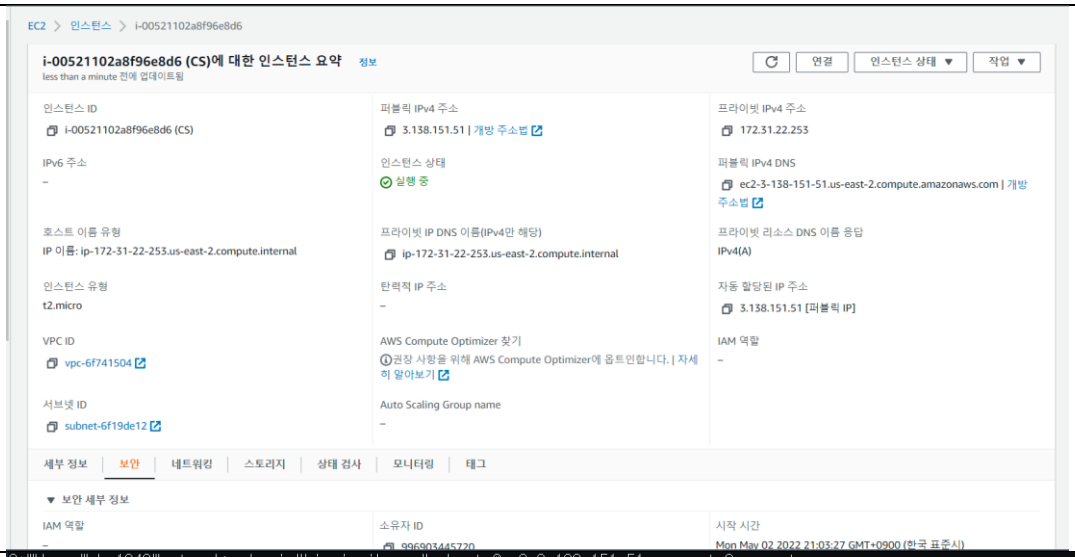
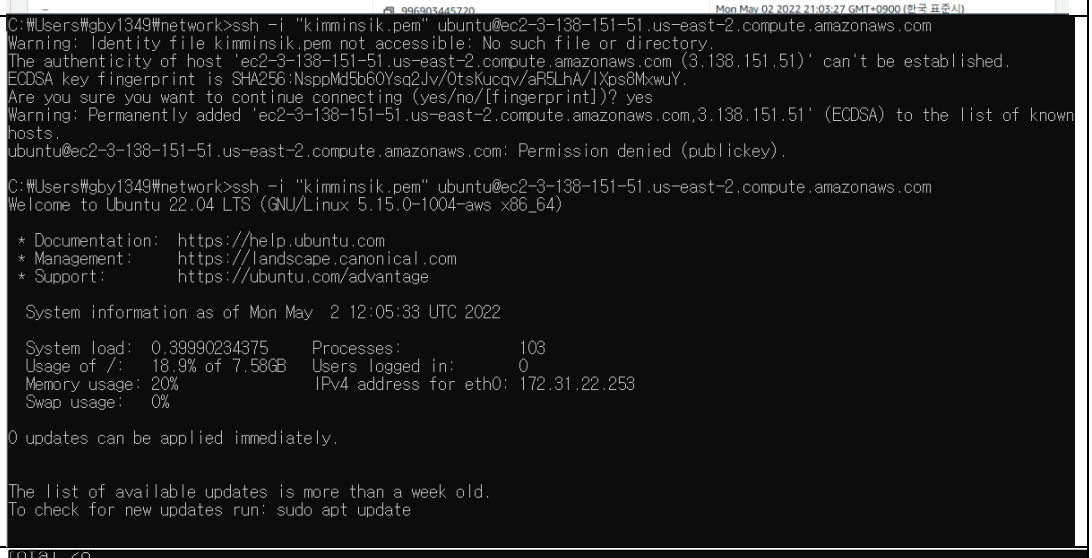
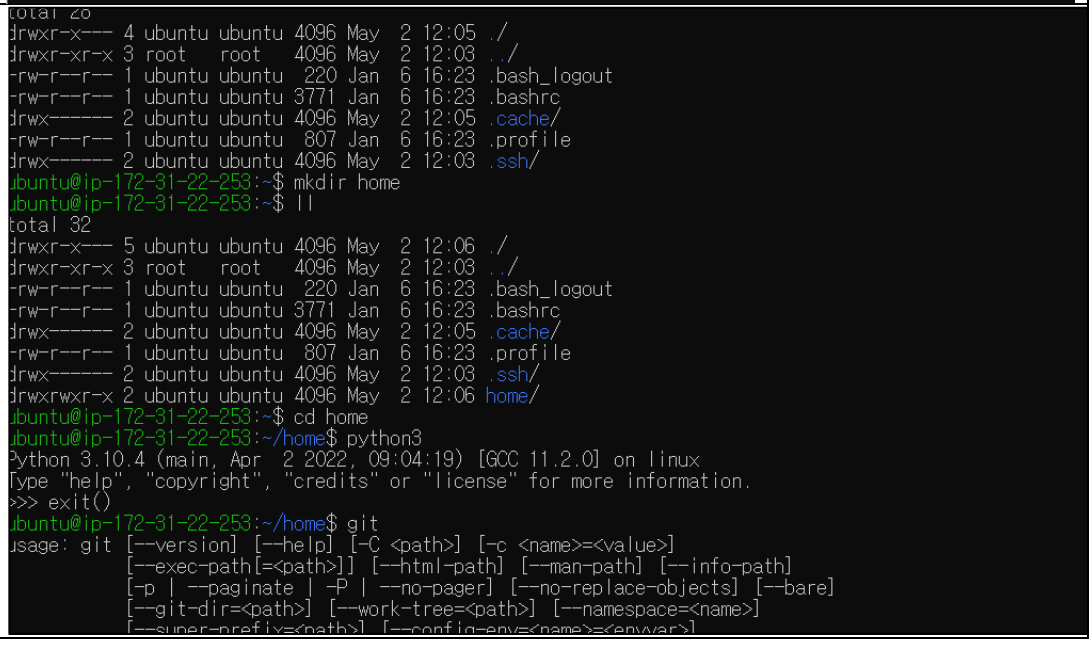
3. 동작 환경/추가 노력(설명)

- **Local Computer를 활용해 두개의 Python Command 창을 활용해서 통신 완료**

- 제안했던 AWS를 활용한 통신을 시도를 했지만 추가적인 통신 연결이 필요하다고 판단됨

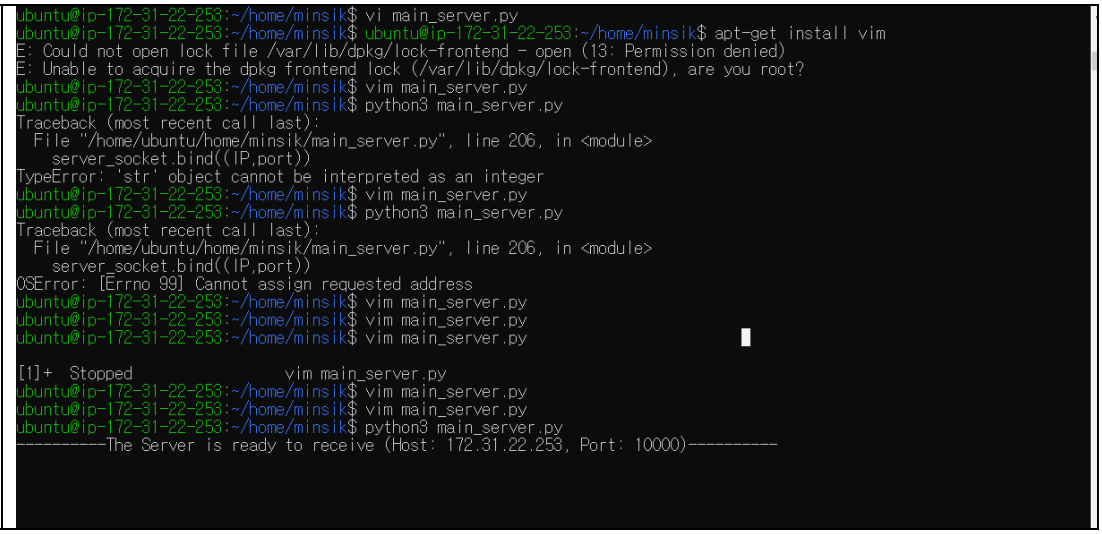
(AWS 인스턴스 생성 후 Command 창에서 server를 git에 저장한 후, AWS를 서버, Local를 Client로 진행하였으나 전송을 받지 못하는 상황 => 추가적인 통신 연결이 필요하지만 시간 관계상 진행하지 못함)

- 이외의 Putty를 활용한 방법을 조사 및 탐색했지만, 잘 진행되지 않음(putty 자체에서 주피터 노트북 접속 방법의 어려움)

AWS 인스턴스 생성	
AWS 서버 접속을 위한 ubuntu 환경 연결	
Git 저장	


```
C:¥Users¥gby1349¥network>ssh -i "kimminsik.pem" ubuntu@ec2-3-138-151-51.us-east-2.compute.amazonaws.com
Warning: Identity file kimminsik.pem not accessible: No such file or directory.
The authenticity of host 'ec2-3-138-151-51.us-east-2.compute.amazonaws.com (3.138.151.51)' can't be established.
ECDSA key fingerprint is SHA256:NsppMd5b60Ysq2Jv/OtsKucqv/aR5LhA/lXps8MxwuY.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'ec2-3-138-151-51.us-east-2.compute.amazonaws.com,3.138.151.51' (ECDSA) to the list of known hosts.
ubuntu@ec2-3-138-151-51.us-east-2.compute.amazonaws.com: Permission denied (publickey).

C:¥Users¥gby1349¥network>ssh -i "kimminsik.pem" ubuntu@ec2-3-138-151-51.us-east-2.compute.amazonaws.com
Welcome to Ubuntu 22.04 LTS (GNU/Linux 5.15.0-1004-aws x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

  System information as of Mon May  2 12:05:33 UTC 2022

  System load:  0.39990234375     Processes:             103
  Usage of /:   18.9% of 7.58GB   Users logged in:       0
  Memory usage: 20%               IPv4 address for eth0: 172.31.22.253
  Swap usage:   0%

0 updates can be applied immediately.


The list of available updates is more than a week old.
To check for new updates run: sudo apt update
```

```
total 28
drwxr-x--- 4 ubuntu ubuntu 4096 May  2 12:05 ./
drwxr-xr-x 3 root   root   4096 May  2 12:03 ../
-rw-r--r-- 1 ubuntu ubuntu  220 Jan  6 16:23 .bash_logout
-rw-r--r-- 1 ubuntu ubuntu 3771 Jan  6 16:23 .bashrc
drwx------ 2 ubuntu ubuntu 4096 May  2 12:05 .cache/
-rw-r--r-- 1 ubuntu ubuntu  807 Jan  6 16:23 .profile
drwx------ 2 ubuntu ubuntu 4096 May  2 12:03 .ssh/
ubuntu@ip-172-31-22-253:~$ mkdir home
ubuntu@ip-172-31-22-253:~$ ll
total 32
drwxr-x--- 5 ubuntu ubuntu 4096 May  2 12:06 ./
drwxr-xr-x 3 root   root   4096 May  2 12:03 ../
-rw-r--r-- 1 ubuntu ubuntu  220 Jan  6 16:23 .bash_logout
-rw-r--r-- 1 ubuntu ubuntu 3771 Jan  6 16:23 .bashrc
drwx------ 2 ubuntu ubuntu 4096 May  2 12:05 .cache/
-rw-r--r-- 1 ubuntu ubuntu  807 Jan  6 16:23 .profile
drwx------ 2 ubuntu ubuntu 4096 May  2 12:03 .ssh/
drwxrwxr-x 2 ubuntu ubuntu 4096 May  2 12:06 home/
ubuntu@ip-172-31-22-253:~$ cd home
ubuntu@ip-172-31-22-253:~/home$ python3
Python 3.10.4 (main, Apr  2 2022, 09:04:19) [GCC 11.2.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> exit()
ubuntu@ip-172-31-22-253:~/home$ git
usage: git [--version] [--help] [-C <path>] [-c <name>=<value>]
           [--exec-path[=<path>]] [--html-path] [--man-path] [--info-path]
           [-p | --paginate | -P | --no-pager] [--no-replace-objects] [--bare]
           [--git-dir=<path>] [--work-tree=<path>] [--namespace=<name>]
           [--super-prefix=<path>] [--config-env=<name>=<envvar>]
```

실행_하지만 서로 TCP 통신이 이루어지지 않음	

4. HTTP 명령어 수행 결과(작동 예)

[1] {'url': '127.0.0.1/', 'method': 'HEAD', 'body': ''}: **100 CONTINUTE**

- 최초 연결 후 Head 확인 및 HEAD **함수 유효한지 확인**

Client:

```
C:₩Users₩gby1349₩network>main_client.py
### Test Case1: {'url': '127.0.0.1/', 'method': 'HEAD', 'body': ''}
----------------------------------------
Sending Request...

Receving Response...

Response data:
 HTTP/1.1 100 CONTINUE
Host: 127.0.0.1/
Content-Type: text/html
Connection: keep-alive
Content-Length: 0
Date: Tue, 03 May 2022 12:11:44 KST



Finish...
```

Server:

```
C:₩Users₩gby1349₩network>main_server.py
----------The Server is ready to receive (Host: 127.0.0.1, Port: 10000)----------
request:
 HEAD / HTTP/1.1
Host: 127.0.0.1/
Content-Type: text/html
Connection: keep-alive
Content-Length: 0


send the response...
```

[2] {'url': '127.0.0.1/', 'method': 'GET', 'body': ''}: **200 OK**

- GET 함수 **유효한지 확인**, DB에 저장되어 있지 않는 경우 반환

Client:

```
### Test Case2: {'url': '127.0.0.1/', 'method': 'GET', 'body': ''}
----------------------------------------
Sending Request...

Receving Response...

Response data:
 HTTP/1.1 200 OK
Host: 127.0.0.1/
Content-Type: text/html
Connection: keep-alive
Content-Length: 13
Date: Tue, 03 May 2022 12:11:44 KST

No data in DB

Finish...
```

Server:

```
----------The Server is ready to receive (Host: 127.0.0.1, Port: 10000)----------
request:
 GET / HTTP/1.1
Host: 127.0.0.1/
Content-Type: text/html
Connection: keep-alive
Content-Length: 0


send the response...
```

[3] {'url': '127.0.0.1??', 'method': 'GET', 'body': ''}: **404 NOT FOUND**

- URL **주소, (PORT/주소) 형식이 잘못**된 경우, 오류 메시지 반환(유효하지 않는 URL 경고 메시지)

- 해당 문제에서는 "/" 주소 대신 "??" 오는 문제

Client:

```
### Test Case3: {'url': '127.0.0.1??', 'method': 'GET', 'body': ''}
--------------------------------------
Sending Request...

Receving Response...

Response data:
 HTTP/1.1 404 NOT_FOUND
Host: ?/?
Content-Type: text/html
Connection: keep-alive
Content-Length: 20
Date: Tue, 03 May 2022 12:11:44 KST

Invalid Address(url)

Finish...
```

Server:

```
----------The Server is ready to receive (Host: 127.0.0.1, Port: 10000)----------
request:
 GET / HTTP/1.1
Host: ?/?
Content-Type: text/html
Connection: keep-alive
Content-Length: 0


send the response...
```

[4] {'url': '127.0.0.1/', 'method': 'PRRT', 'body': ''}: **405 METHOD_NOT_ALLOWED**

- **허용되지 않는 Method**가 들어올 경우, 오류 메시지 반환(허용되지 않는 Method 경고 메시지)

- 해당 문제에서는 없는 Method(오타) PRRT가 들어와 오류

Client:

```
### Test Case4: {'url': '127.0.0.1/', 'method': 'PRRT', 'body': ''}
--------------------------------------
Sending Request...

Receving Response...

Response data:
 HTTP/1.1 405 METHOD_NOT_ALLOWED
Host: 127.0.0.1/
Content-Type: text/html
Connection: keep-alive
Content-Length: 18
Date: Tue, 03 May 2022 12:11:44 KST

Method not allowed

Finish...
```

Server:

```
----------The Server is ready to receive (Host: 127.0.0.1, Port: 10000)----------
request:
 PRRT / HTTP/1.1
Host: 127.0.0.1/
Content-Type: text/html
Connection: keep-alive
Content-Length: 0


send the response...
```

[5] {'url': '127.0.5.1/', 'method': 'GET', 'body': ''}: **400 BAD_REQUEST**

- 허용되지 않는 HOST가 들어온 경우, 잘못된 요청으로 인식 후 경고 메시지 전달
- 본 문제에서는 허용 HOST를 127.0.0.1(Local)만 허용하였기 때문에, 이외의 Host(127.0.5.1) 와서 문제가 됨

Client:

```
### Test Case5: {'url': '127.0.5.1/', 'method': 'GET', 'body': ''}
----------------------------------------
Sending Request...

Receving Response...

Response data:
 HTTP/1.1 400 BAD_REQUEST
Host: 127.0.5.1/
Content-Type: text/html
Connection: keep-alive
Content-Length: 27
Date: Tue, 03 May 2022 12:11:44 KST

Not allowed host: 127.0.5.1

Finish...
```

Server:

```
----------The Server is ready to receive (Host: 127.0.0.1, Port: 10000)----------
request:
 GET / HTTP/1.1
Host: 127.0.5.1/
Content-Type: text/html
Connection: keep-alive
Content-Length: 0


send the response...
```

[6] {'url': '127.0.0.1/create', 'method': 'PUT', 'body': ''}: **404 NOT FOUND**

- create 생성이지만 PUT 입력을 받은 경우, PUT과 POST **Method와 서로 매칭되지 않는 path**가 왔음을 경고
- 해당 문제에서는 path가 create임으로 적절한 Method는 "POST"가 와야 하지만 "PUT"옴

Client:

```
### Test Case6: {'url': '127.0.0.1/create', 'method': 'PUT', 'body': ''}
----------------------------------------
Sending Request...

Receving Response...

Response data:
 HTTP/1.1 404 NOT_FOUND
Host: 127.0.0.1/create
Content-Type: text/html
Connection: keep-alive
Content-Length: 50
Date: Tue, 03 May 2022 12:11:44 KST

Select the appropriate method(Create mode => POST)

Finish...
```

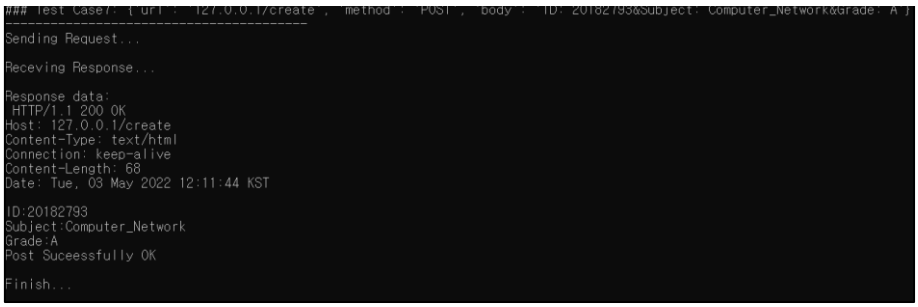
Server:

```
----------The Server is ready to receive (Host: 127.0.0.1, Port: 10000)----------
request:
 PUT / HTTP/1.1
Host: 127.0.0.1/create
Content-Type: text/html
Connection: keep-alive
Content-Length: 0


send the response...
```

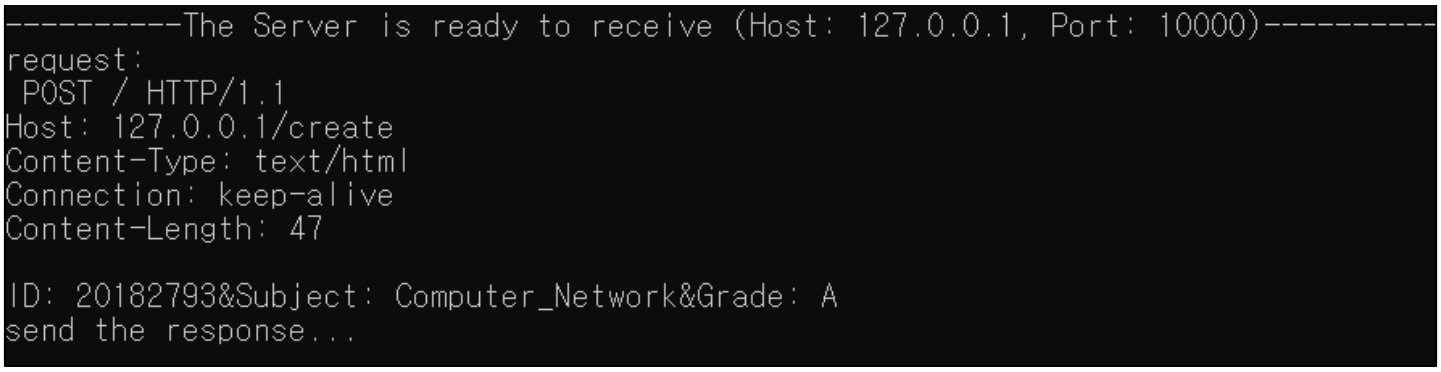
[7] {'url': '127.0.0.1/create', 'method': 'POST', 'body': 'ID: 20182793&Subject: Computer_Network&Grade: A'}: **200 OK**

- Data를 새롭게 **POST를 통해서 정상적으로 입력**

Client:



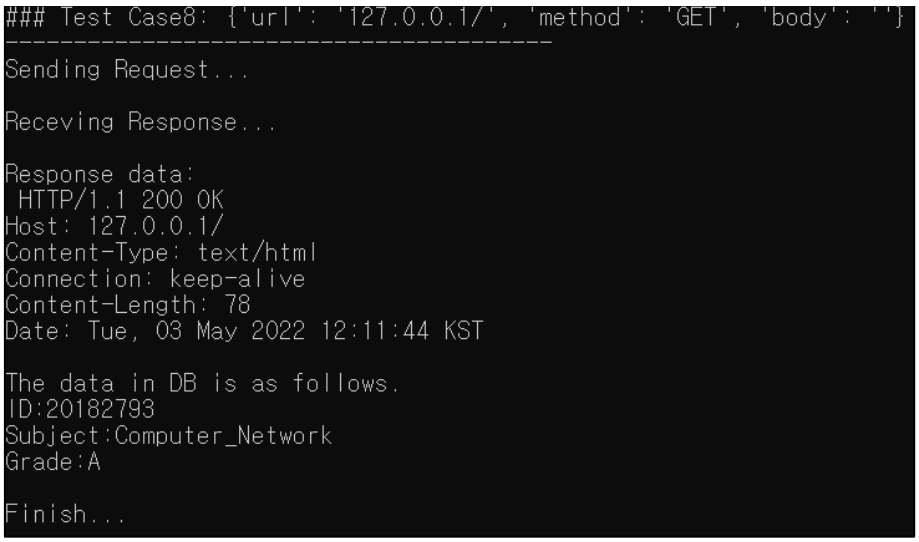
Server:



[8] {'url': '127.0.0.1/', 'method': 'GET', 'body': ''}: **200 OK**

- (앞전에서 DB에 넣은 데이터) **저장 내용 확인**

Client:



Server:

[9] {'url': '127.0.0.1/create', 'method': 'POST', 'body': ': 95'}: **400 BAD-REQUEST**

- **Key값이 유효하지 않는 경우**(공백 혹은 NoneType), 경고메시지를 보냄

- 해당 문제에서는 Key값이 공백으로 옴

Client:

```
### Test Case9: {'url': '127.0.0.1/create', 'method': 'POST', 'body': ': 95'}
----------------------------------------
Sending Request...

Receving Response...

Response data:
 HTTP/1.1 400 BAD_REQUEST
Host: 127.0.0.1/create
Content-Type: text/html
Connection: keep-alive
Content-Length: 37
Date: Tue, 03 May 2022 12:11:44 KST

Key value does not exist or is blank.

Finish...
```

Server:

```
----------The Server is ready to receive (Host: 127.0.0.1, Port: 10000)----------
request:
 POST / HTTP/1.1
Host: 127.0.0.1/create
Content-Type: text/html
Connection: keep-alive
Content-Length: 4

: 95
send the response...
```

[10] {'url': '127.0.0.1/create', 'method': 'POST', 'body': 'middle_exam70'}: **400 BAD_REQUEST**

- **데이터 입력 형식을 맞추지 않는 경우**(Key: value 구조), 오류 메시지를 보냄

- 해당 문제에서는 ":"생략이 된 상황에서 값이 들어옴

Client:

```
### Test Case10: {'url': '127.0.0.1/create', 'method': 'POST', 'body': 'middle_exam70'}
----------------------------------------
Sending Request...

Receving Response...

Response data:
 HTTP/1.1 400 BAD_REQUEST
Host: 127.0.0.1/create
Content-Type: text/html
Connection: keep-alive
Content-Length: 30
Date: Tue, 03 May 2022 12:11:44 KST

Key-value format is not valid.

Finish...
```

Server:

```
----------The Server is ready to receive (Host: 127.0.0.1, Port: 10000)----------
request:
 POST / HTTP/1.1
Host: 127.0.0.1/create
Content-Type: text/html
Connection: keep-alive
Content-Length: 13

middle_exam70
send the response...
```

[11] {'url': '127.0.0.1/create', 'method': 'POST', 'body': 'ID: 20180434'}: **400 BAD_REQUEST**

- POST Method에서 Create 경우, 동일한 Key 값이 온 경우, 오류 메시지를 보냄

- 해당 문제에서는 "ID(학번)" 키 값이 온 상황에서 이미 DB에 ID가 저장되어 있어 문제 발생

Client:

```
### Test Case11: {'url': '127.0.0.1/create', 'method': 'POST', 'body': 'ID: 20180434'}
----------------------------------------
Sending Request...

Receving Response...

Response data:
 HTTP/1.1 400 BAD_REQUEST
Host: 127.0.0.1/create
Content-Type: text/html
Connection: keep-alive
Content-Length: 23
Date: Tue, 03 May 2022 12:11:44 KST

The key already exists!

Finish...
```

Server:

```
----------The Server is ready to receive (Host: 127.0.0.1, Port: 10000)----------
request:
 POST / HTTP/1.1
Host: 127.0.0.1/create
Content-Type: text/html
Connection: keep-alive
Content-Length: 12

ID: 20180434
send the response...
```

[12] {'url': '127.0.0.1/', 'method': 'HEAD', 'body': ''}: **100 CONTINUE**

- 진행 과정에서 **HEAD 통신 재확인**

Client:

```
### Test Case12: {'url': '127.0.0.1/', 'method': 'HEAD', 'body': ''}
----------------------------------------
Sending Request...

Receving Response...

Response data:
 HTTP/1.1 100 CONTINUE
Host: 127.0.0.1/
Content-Type: text/html
Connection: keep-alive
Content-Length: 0
Date: Tue, 03 May 2022 12:11:44 KST



Finish...
```

Server:

```
----------The Server is ready to receive (Host: 127.0.0.1, Port: 10000)----------
request:
 HEAD / HTTP/1.1
Host: 127.0.0.1/
Content-Type: text/html
Connection: keep-alive
Content-Length: 0


send the response...
```

[13] {'url': '127.0.0.1/update', 'method': 'PUT', 'body': 'ID: 20180434&final_exam: 90'}: **200 OK**

- DB에 저장된 **데이터를 Update를 진행**하거나, **기존에 없는 데이터(Key)가 올 경우 추가**한 경우

- 본 문제에서는 기존에 있던 키 ID(학번)에 대한 값은 변경하고, DB에 없는 정보인 기말고사 점수를 추가함

Client:

```
### Test Case13: {'url': '127.0.0.1/update', 'method': 'PUT', 'body': 'ID: 20180434&final_exam: 90'}
----------------------------------------
Sending Request...

Receving Response...

Response data:
 HTTP/1.1 200 OK
Host: 127.0.0.1/update
Content-Type: text/html
Connection: keep-alive
Content-Length: 189
Date: Tue, 03 May 2022 12:11:44 KST

ID: 20180434&final_exam: 90

Update Data as follow:
ID: 20182793 => 20180434

Update(New) Data as follow:
20182793: 90(There's no key, New update)
Update with new data Sucessfully OK

Finish...
```

Server:

```
----------The Server is ready to receive (Host: 127.0.0.1, Port: 10000)----------
request:
 PUT / HTTP/1.1
Host: 127.0.0.1/update
Content-Type: text/html
Connection: keep-alive
Content-Length: 27

ID: 20180434&final_exam: 90
send the response...
```

[14] {'url': '127.0.0.1/update', 'method': 'PUT', 'body': ''}: **400 BAD_REQUEST**

- **Update할 내용이 Body 입력하지 않는 경우**, 에러메시지 출력

Client:

```
### Test Case14: {'url': '127.0.0.1/update', 'method': 'PUT', 'body': ''}
----------------------------------------
Sending Request...

Receving Response...

Response data:
 HTTP/1.1 400 BAD_REQUEST
Host: 127.0.0.1/update
Content-Type: text/html
Connection: keep-alive
Content-Length: 33
Date: Tue, 03 May 2022 12:11:44 KST

There is no content. Please check

Finish...
```

Server:

```
----------The Server is ready to receive (Host: 127.0.0.1, Port: 10000)----------
request:
 PUT / HTTP/1.1
Host: 127.0.0.1/update
Content-Type: text/html
Connection: keep-alive
Content-Length: 0


send the response...
```

[15] {'url': '127.0.0.1/update', 'method': 'PUT', 'body': 'retake_wheaterYes'}: **400 BAD_REQUEST**

- **데이터 입력 형식을 맞추지 않는 경우**(Key: value 구조), 오류 메시지를 보냄

- 해당 문제에서는 ":"생략이 된 상황에서 값이 들어옴

Client:

```
### Test Case15: {'url': '127.0.0.1/update', 'method': 'PUT', 'body': 'retake_wheaterYes'}
----------------------------------------
Sending Request...

Receving Response...

Response data:
 HTTP/1.1 400 BAD_REQUEST
Host: 127.0.0.1/update
Content-Type: text/html
Connection: keep-alive
Content-Length: 30
Date: Tue, 03 May 2022 12:11:44 KST

Key-value format is not valid.

Finish...
```

Server:

```
----------The Server is ready to receive (Host: 127.0.0.1, Port: 10000)----------
request:
 PUT / HTTP/1.1
Host: 127.0.0.1/update
Content-Type: text/html
Connection: keep-alive
Content-Length: 17

retake_wheaterYes
send the response...
```

[16] {'url': '127.0.0.1/', 'method': 'END', 'body': ''}: 종료

- 전체 Test case를 보내고 종료 조건을 보냄

Client:

```
### Test Case16: {'url': '127.0.0.1/', 'method': 'END', 'body': ''}
----------------------------------------
Sending Request...

Receving Response...

Response data:


Finish...
```

Server:

```
----------The Server is ready to receive (Host: 127.0.0.1, Port: 10000)----------
request:
 END / HTTP/1.1
Host: 127.0.0.1/
Content-Type: text/html
Connection: keep-alive
Content-Length: 0


Complete all courses. socket finish
```

5. WireShark로 캡처한 통신