SCA 파이썬으로 HTTP 서버 구축

10207 박준혁

HTTP 서버 목차

사용 모듈
import socket

socket(): 새로운 소켓 객체를 생성합니다.

bind(address): 소켓에 주소를 바인딩합니다.

listen(backlog): 연결 요청을 받기 위해 소켓을 리스닝 상태로 설정합니다.

accept(): 클라이언트의 연결 요청을 받아들여 소켓 객체와 클라이언트 주소를 반환합니다.

connect(address): 서버에 연결합니다.

recv(bufsize): 소켓으로부터 데이터를 수신합니다.

close(): 소켓을 닫습니다.

서버의 HOST 초기화와 PORT번호 설정

```python
import socket


# 서버의 호스트와 포트 설정
SERVER_HOST = '0.0.0.0'
SERVER_PORT = 8000
```

호스트는 초기화해주면 서버는 어떤 IP 주소로부터의 연결도
수락할 수 있기 때문에 초기화 해줍니다!

포트번호는 8000번으로 지정해줍니다!
이유는 저희와 같은 일반 사용자들은 1024번 이상 포트번호부터 사용할 수 있지만
굳이 8000번을 쓰는 이유는 해당 포트가 충돌이 가장 적은 포트이기 때문입니다

소켓 생성과 바인딩, 최대 연결 요청 수 조절

```python
import socket

# 서버의 호스트와 포트 설정
SERVER_HOST = '0.0.0.0'
SERVER_PORT = 8000

# 소켓 만들기
4 usages
class server:
    def Create_server(self,socket,bind,listen):
        self.socket = socket
        self.bind = bind
        self.listen = listen
server.socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
server.socket.bind((SERVER_HOST, SERVER_PORT))
server.socket.listen(1)
print("포트 번호 %s연결 대기 중..." %SERVER_PORT)
```

server.socket은 IP4v 주소체계를 사용하고
소켓 스트림을 사용해 TCP 소켓을 생성하겠다는 의미입니다!

server.socket.bind는 튜플자료형을 사용하여, 서버 호스트와 포트를 바인딩해서 클라이언트와 통신할 수 있게 해줍니다

server.socket.listen(1)은
연결 요청 최대 갯수를 1개로 설정한것입니다

클래스(Class)는 Server이고 생성자 이름으로 Create_server을 선언해줍니다.
그 후, self키워드로 클래스 내부에서 호출한 객체를 쉽게 가르키기위해 사용해줍니다.

최종적으로 포트 번호 8000번에 연결을 대기하게 됩니다

```python
#클라이언트의 응답을 대기
5 usages
class clinet:
    def __init__(self,connection,adress):
        self.connection = connection
        self.adress = adress
clinet.connection, clinet.adress = server.socket.accept()
```

클라이언트 요청 수락과 연결할 소켓과 주소를 반환

클래스(Class)는 clinet이고 생성자 이름으로 __init__을 선언해줍니다.
그 후, self키워드로 클래스 내부에서 호출한 객체를 쉽게 가르키기위해 사용해줍니다.

clinet.connection, clinet.adress = server.socket.accept()는
서버 소켓에서 클라이언트의 연결 요청을 수락하고
클라이언트와 연결할 소켓과 주소를 반환합니다.

클라이언트 커넥션 소켓을 사용해서 1024바이트까지 수신하며,
바이트열을 문자열로 디코딩합니다!

```
request = clinet.connection.recv(1024).decode()
print(request)
```

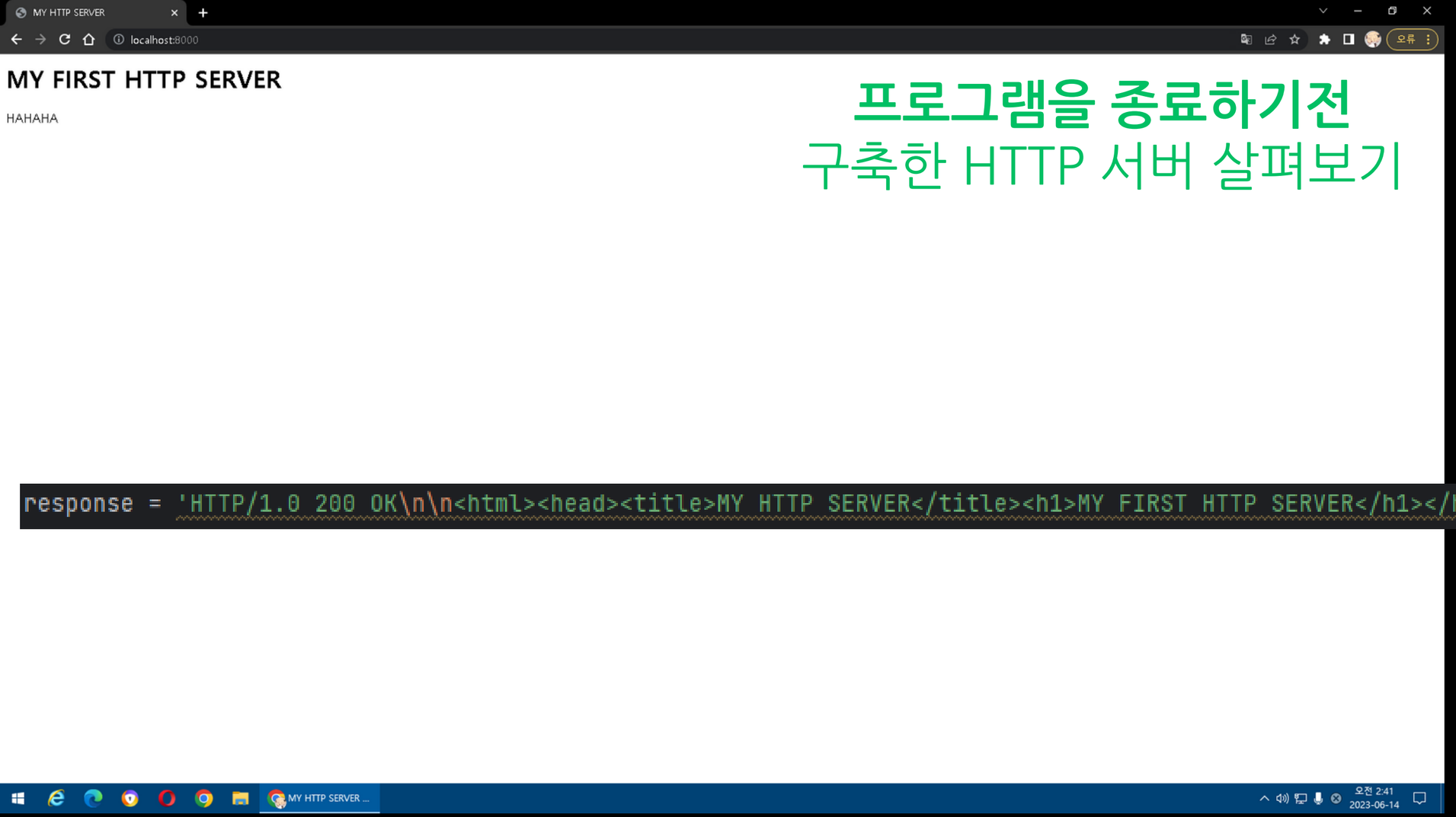
request 변수에 클라이언트 커넥션 소켓을 사용해서 1024바이트
까지 수신받고, 바이트열을 문자열로 디코딩합니다
그렇게 디코딩한 request 변수를 출력합니다!

```
response = 'HTTP/1.0 200 OK\n\n<html><head><title>MY HTTP SERVER</title><h1>MY FIRST HTTP SERVER</h1></head><body>HAHAHA</body></html>'
clinet.connection.sendall(response.encode())
clinet.connection.close()

#서버측 소켓 닫기
server.socket.close()
```

response 변수에 HTTP 서버를 구축하고
sendall()함수를 통해 해당 변수를 문자열에서 바이트열로 전환해줍니다

그렇게 클라이언트 커넥션 소켓을 통해 클라이언트로 모두 전송하고
클라이언트와의 연결을 닫고, 서버측 소켓을 닫습니다.



프로그램을 종료하기전
구축한 HTTP 서버 살펴보기

```
response = 'HTTP/1.0 200 OK\n\n<html><head><title>MY HTTP SERVER</title><h1>MY FIRST HTTP SERVER</h1></head><body>HAHAHA</body></html>'
```

위 자료들을 보시면 다음과 같이 코드가 잘 실행된 것을 알 수 있습니다!

전체 코드

```python
import socket

# 서버의 호스트와 포트 설정
SERVER_HOST = '0.0.0.0'
SERVER_PORT = 8000

# 소켓 만들기
5 usages
class server:
    def Create_server(self,socket,bind,listen):
        self.socket = socket
        self.bind = bind
        self.listen = listen
server.socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
server.socket.bind((SERVER_HOST, SERVER_PORT))
server.socket.listen(1)
print("포트 번호 %s연결 대기 중..." %SERVER_PORT)


#클라이언트의 응답을 대기
5 usages
class clinet:
    def __init__(self,connection,adress):
        self.connection = connection
        self.adress = adress
clinet.connection, clinet.adress = server.socket.accept()

request = clinet.connection.recv(1024).decode()
print(request)

response = 'HTTP/1.0 200 OK\n\n<html><head><title>MY HTTP SERVER</title><h1>MY FIRST HTTP SERVER</h1></head><body>HAHAHA</body></html>'
clinet.connection.sendall(response.encode())
clinet.connection.close()

#서버측 소켓 닫기
server.socket.close()
```

SCA
발표 마치겠습니다

10207 박준혁