

# Flask - API-MySQL 연동 SQLAlchemy

inyong\_pang · 2020년 1월 31일

♡ 1

Flask



▼ 목록 보기

7/13



## SQLAlchemy

파이썬 코드에서 Database와 연결하기 위해 사용할 수 있는 라이브러리중 SQLAlchemy라는 라이브러리가 있다.

SQLAlchemy는 ORM(Object Relational Mapper)로 관계형 데이터베이스의 테이블들을 프로그래밍 언어의 클래스로 표현할 수 있게 해주는 라이브러리이다.

즉, 클래스(class)를 사용해서 DB TABLE들을 표현하고 저장, 읽기, 업데이트, 삭제 등을 할 수 있도록 해준다.

### SQLAlchemy 설치

SQLAlchemy는 pip 명령어를 통해 설치한다. 즉, 파이썬 패키지 매니저로 설치가 가능하다. 이 또한 개발가상환경에서 설치하는 것을 권장한다.

```
(develop) $ pip install sqlalchemy
```

### mysql-connector-python 설치

파이썬이 MySQL을 사용하기 위해서는 MySQL용 DBAPI 또한 설치해야한다. DBAPI란 DB를 사용하기 위한 API이다.

```
(develop) ~$ pip install mysql-connector-python
```

# API-MySQL 연결

## config.py 생성

이제 우리가 만들었던 Minitter API를 MySQL과 연결할 수 있다.

먼저, 데이터베이스의 연결 정보를 저장할 파일을 따로 만들어야 한다.

config.py 라는 이름으로 새로운 파일을 만들어 데이터베이스 연결에 관련된 내용을 작성한다.

```
# config.py
db = {
    'user'      : 'root',          # 1)
    'password'  : '0000',         # 2)
    'host'      : 'localhost',    # 3)
    'port'      : 3306,           # 4)
    'database'  : 'minitter'      # 5)
}

DB_URL = f"mysql+mysqlconnector://{db['user']}:{db['password']}@{db['host']}:{db['port']}/{db['database']}"
```

1) 데이터베이스에 접속할 사용자 id

2) 데이터베이스 사용자의 비밀번호

3) 접속할 데이터베이스의 주소.

4) 접속할 데이터베이스의 포트(port)번호. MySQL일 경우 보통 3306 포트를 사용한다.

5) 실제 연결하고자 하는 데이터베이스 명

## app.py 수정

config.py 를 만들었으면 app.py 를 수정하여 config.py 의 데이터베이스 설정을 가져와서 연결한다.

그리고 sqlalchemy의 create\_engine 을 사용하여 데이터베이스 연결한다

```
from flask import Flask, request, jsonify, current_app
from sqlalchemy import create_engine, text

def create_app(test_config = None): # 1)
    app = Flask(__name__)
```

```

if test_config is None:                # 2)
    app.config.from_pyfile("config.py")
else:
    app.config.update(test_config)

database = create_engine(app.config['DB_URL'], encoding = 'utf-8', max_overflow = 0)
app.database = database                # 4)

return app                            # 5)

```

1) create\_app 이라는 함수를 정의한다.

Flask가 create\_app 이라는 이름의 함수를 자동으로 팩토리(factory) 함수로 인식해서 해당 함수를 통해 Flask를 실행시킨다.

그리고 create\_app 함수가 test\_config 인자를 받는다.

test\_config 는 단위 테스트(unit test)를 실행시킬 때 테스트용 데이터베이스 등의 테스트 설정 정보를 적용하기 위함이다.

2) 만일 test\_conig 인자가 None 이면 config.py 파일에서 설정을 읽는다.

만일 test\_config 인자가 None 이 아니라면, 즉, test\_config 값이 설정되어 들어왔다면, 단위 테스트를 실행 할 수 있다.

3) sqlalchemy의 create\_engine 함수를 사용해서 데이터베이스의 연결을 한다.

4) 3)에서 생성한 Engine 객체를 Flask 객체에 저장함으로써 create\_app 함수를 외부에서도 데이터베이스를 사용할 수 있도록 한다.

5) Flask 객체를 리턴한다. 앞서 create\_app 이라는 함수는 Flask가 자동 인지하여 Flask객체를 찾아서 실행될 수 있다.

## 정리

- SQLAlchemy의 create\_engine 함수를 사용하여 데이터베이스에 연결하고 text 함수를 사용하여 실행시킬 SQL구문을 전달할 수 있다.

- Flask가 create\_app 이라는 이름의 함수를 자동으로 팩토리(factory) 함수로 인식해서 해당 함수를 통해서 Flask 를 실행한다.

 inyong\_pang.log



로그인



## 황인용

dev\_pang의 pang.log



다음 포스트

Flask - API-MySQL 엔드포인트 연동



이전 포스트

Flask - API-MySQL 연동준비

## 0개의 댓글

댓글을 작성하세요

댓글 작성