

FLASK | SQLALCHEMY | FLASK-SQLACODEGEN

# DB TO FLASK-SQLALCHEMY MODEL

3년 전 FLASK / SQLALCHEMY

## 기존 DB를 Flask-SQLAlchemy ORM Model로 사용하기

본 게시글에서는 MySQL/Sqlite을 예제로 하고있지만, Flask-SQLAlchemy가 지원하는 다른 DB에서도 사용 가능합니다.

### 들어가며

Flask로 웹 개발 진행 시 SQLAlchemy(Flask-SQLAlchemy)를 사용해 ORM구조를 구성할 때 데이터를 저장할 DB의 구조를 직접 확인하며 진행하는 것은 상당히 귀찮고 어려운 일입니다.

Django에는 내장된 `inspectdb`라는 명령어를 통해 Django와 일치하는 DB Model구조를 만들어주지만 SQLAlchemy 자체에 내장된 `automap`은 우리가 상상하는 모델 구조를 바로 만들어주지는 않습니다.

따라서 다른 패키지를 고려해볼 필요가 있습니다.

### flask-sqlacodegen

`flask-sqlacodegen`은 기존 DB를 Flask-SQLAlchemy에서 사용하는 Model 형식으로 변환해 보여주는 패키지입니다. 기존 `sqlacodegen`에서 포크해 Flask-SQLAlchemy에 맞게 기본 설정이 갖추어져있어 편리합니다.

## 설치하기

설치는 `pip`로 간단하게 진행해 주세요.

글쓰는 시점 최신버전은 1.1.6.1입니다.

글쓰는 것과 같은 버전으로 설치하려면 `flask-sqlacodegen==1.1.6.1` 로 설치해 주세요.



```
1 # 최신 버전 설치하기
2 pip install flask-sqlacodegen
3 # 글쓰는 시점과 같게 설치하려면
4 # pip install flask-sqlacodegen==1.1.6.1
```

설치가 완료되면 명령줄에서 `flask-sqlacodegen`라는 명령어를 사용할 수 있습니다.

주의: `sqlacodegen`이 이미 깔려있다면 다른 가상환경(`virtuale / venv`)를 만드시고 진행해 주세요. `sqlacodegen`이 깔려있으면 `--flask`이 동작하지 않습니다.

## DB 구조 뜯어내기

`flask-sqlacodegen`는 `sqlacodegen`과 거의 동일한 문법을 사용합니다.(포크를 뜯은 프로젝트니까요!)

`flask-sqlacodegen` 명령어로 DB를 지정하면 구조를 알 수 있습니다.

### SQLite의 경우



```
1 flask-sqlacodegen "sqlite:///db.sqlite3" --flask > models.py # 상대경로
```

SQLite는 로컬에 있는 DB의 위치를 지정하면 됩니다.

위 명령어를 실행하면 `models.py`파일 안에 `db.sqlite3` DB의 모델이 정리됩니다.

NOTE: Sqlite의 파일을 지정할 경우 `"sqlite://"`가 아닌 `"sqlite:///"`로 `/`를 3번 써 주셔야 상대경로로 지정 가능하며, `"sqlite:///"`로 `/`를 4번 써 주셔야 절대경로로 지정이 가능합니다.

## mysql 서버의 경우

▼



```
1 flask-sqlacodegen "mysql://username:password@DB_IP/DB_NAME" --flask >
```

MySQL의 경우 mysql에 접속하는 방식 그대로 사용자 이름, 비밀번호, IP(혹은 HOST도 메인), DB이름을 넣어준 뒤 진행해주면 됩니다.

NOTE: mysql은 "mydql://" 로 /가 2번입니다.

NOTE: mysql에 연결하려면 pip패키지 중 `mysqlclient`가 설치되어있어야 합니다.

설치가 되어있지 않으면 아래와 같이 `ModuleNotFoundError`가 발생합니다.

```
2. beomi@Junbumui-MacBook: ~/tmp/sqltest (zsh)
~/tmp/sqltest (zsh) 961
Traceback (most recent call last):
  File "/Users/beomi/.virtualenvs/tmp2/bin/flask-sqlacodegen", line 11, in <module>
    sys.exit(main())
  File "/Users/beomi/.virtualenvs/tmp2/lib/python3.6/site-packages/sqlacodegen/main.py", line 50, in main
    engine = create_engine(args.url)
  File "/Users/beomi/.virtualenvs/tmp2/lib/python3.6/site-packages/sqlalchemy/engine/__init__.py", line 391, in create_engine
    return strategy.create(*args, **kwargs)
  File "/Users/beomi/.virtualenvs/tmp2/lib/python3.6/site-packages/sqlalchemy/engine/strategies.py", line 80, in create
    dbapi = dialect_cls.dbapi(**dbapi_args)
  File "/Users/beomi/.virtualenvs/tmp2/lib/python3.6/site-packages/sqlalchemy/dialects/mysql/mysql.py", line 110, in dbapi
    return __import__('MySQLdb')
ModuleNotFoundError: No module named 'MySQLdb'
(tmp2) ✖ ~/tmp/sqltest master
```

MAC에서 진행 중 혹시 `mysqlclient`설치 중 아래와 같은 에러가 발생한다면

```

2. beomi@Junbumui-MacBook: ~/tmp/sqltest (zsh)
~/tmp/sqltest (zsh) 961
oper/SDKs/MacOSX10.13.sdk/System/Library/Frameworks/Tk.framework/Versions/8.5/Headers -Dversion_info=(1,3,12,'final',0) -D__version__=1.3.12 -I/usr/local/Cellar/mysql/5.7.19/include/mysql -I/usr/local/Cellar/python3/3.6.3/Frameworks/Python.framework/Versions/3.6/include/python3.6m -c _mysql.c -o build/temp.macosx-10.12-x86_64-3.6/_mysql.o
clang -bundle -undefined dynamic_lookup -isysroot /Applications/Xcode.app/Contents/Developer/Platforms/MacOSX.platform/Developer/SDKs/MacOSX10.13.sdk build/temp.macosx-10.12-x86_64-3.6/_mysql.o -L/usr/local/Cellar/mysql/5.7.19/lib -lmysqlclient -lssl -lcrypto -o build/lib.macosx-10.12-x86_64-3.6/_mysql.cpython-36m-darwin.so
ld: library not found for -lssl
clang: error: linker command failed with exit code 1 (use -v to see invocation)
error: command 'clang' failed with exit status 1

-----
Command "/Users/beomi/.virtualenvs/tmp2/bin/python3.6 -u -c "import setuptools, tokenize;__file__='/private/var/folders/8z/m21ck94n5jj3jl12dp91367m0000gn/T/pip-

```

아래 명령어를 실행해 `xcode cli developer tool`와 `openssl`을 설치해주신 후 `mysqlclient`를 설치해 주세요.



- 1 `xcode-select --install`
- 2 `brew install openssl`
- 3 `export LIBRARY_PATH=$LIBRARY_PATH:/usr/local/opt/openssl/lib/`
- 4 `pip install mysqlclient`

```

2. beomi@Junbumui-MacBook: ~/tmp/sqltest (zsh)
~/tmp/sqltest (zsh) 961
ld: library not found for -lssl
clang: error: linker command failed with exit code 1 (use -v to see invocation)
error: command 'clang' failed with exit status 1

-----
Command "/Users/beomi/.virtualenvs/tmp2/bin/python3.6 -u -c "import setuptools, tokenize;__file__='/private/var/folders/8z/m21ck94n5jj3jl12dp91367m0000gn/T/pip-
build-u_c_66wz/mysqlclient/
(tmp2) x ~/tmp/sqltest master xcode-select --install
xcode-select: note: install requested for command line developer tools
(tmp2) ~/tmp/sqltest master

```

## 실행결과

아래 결과는 장고 프로젝트를 생성하고 첫 `migrate`를 진행할 때 생기는 예시 `db.sqlite3`파일을 `flask-sqlacodegen`을 사용한 결과입니다.

Index, PK등을 잘 잡아주고 있는 모습을 볼 수 있습니다.



```

1  # models.py 파일
2  # coding: utf-8
3  from sqlalchemy import Boolean, Column, DateTime, ForeignKey, Index
4  from sqlalchemy.orm import relationship
5  from sqlalchemy.sql.sqltypes import NullType
6  from flask_sqlalchemy import SQLAlchemy
7
8  db = SQLAlchemy()
9
10 class AuthGroup(db.Model):
11     __tablename__ = 'auth_group'
12
13     id = db.Column(db.Integer, primary_key=True)
14     name = db.Column(db.String(80), nullable=False)
15
16
17 class AuthGroupPermission(db.Model):
18     __tablename__ = 'auth_group_permissions'
19     __table_args__ = (
20         db.Index('auth_group_permissions_group_id_permission_id_0cd
21     )
22
23     id = db.Column(db.Integer, primary_key=True)
24     group_id = db.Column(db.ForeignKey('auth_group.id'), nullable=False)
25     permission_id = db.Column(db.ForeignKey('auth_permission.id'),
26
27     group = db.relationship('AuthGroup', primaryjoin='AuthGroupPerm
28     permission = db.relationship('AuthPermission', primaryjoin='Aut
29
30
31 class AuthPermission(db.Model):
32     __tablename__ = 'auth_permission'
33     __table_args__ = (
34         db.Index('auth_permission_content_type_id_codename_01ab375a
35     )
36
37     id = db.Column(db.Integer, primary_key=True)
38     content_type_id = db.Column(db.ForeignKey('django_content_type.
39     codename = db.Column(db.String(100), nullable=False)
40     name = db.Column(db.String(255), nullable=False)
41

```

```
42     content_type = db.relationship('DjangoContentType', primaryjoin
43
44
45 class AuthUser(db.Model):
46     __tablename__ = 'auth_user'
47
48     id = db.Column(db.Integer, primary_key=True)
49     password = db.Column(db.String(128), nullable=False)
50     last_login = db.Column(db.DateTime)
51     is_superuser = db.Column(db.Boolean, nullable=False)
52     first_name = db.Column(db.String(30), nullable=False)
53     last_name = db.Column(db.String(30), nullable=False)
54     email = db.Column(db.String(254), nullable=False)
55     is_staff = db.Column(db.Boolean, nullable=False)
56     is_active = db.Column(db.Boolean, nullable=False)
57     date_joined = db.Column(db.DateTime, nullable=False)
58     username = db.Column(db.String(150), nullable=False)
59
60
61 class AuthUserGroup(db.Model):
62     __tablename__ = 'auth_user_groups'
63     __table_args__ = (
64         db.Index('auth_user_groups_user_id_group_id_94350c0c_uniq',
65     )
66
67     id = db.Column(db.Integer, primary_key=True)
68     user_id = db.Column(db.ForeignKey('auth_user.id'), nullable=False)
69     group_id = db.Column(db.ForeignKey('auth_group.id'), nullable=False)
70
71     group = db.relationship('AuthGroup', primaryjoin='AuthUserGroup
72     user = db.relationship('AuthUser', primaryjoin='AuthUserGroup.u
73
74
75 class AuthUserUserPermission(db.Model):
76     __tablename__ = 'auth_user_user_permissions'
77     __table_args__ = (
78         db.Index('auth_user_user_permissions_user_id_permission_id_
79     )
80
81     id = db.Column(db.Integer, primary_key=True)
82     user_id = db.Column(db.ForeignKey('auth_user.id'), nullable=False)
83     permission_id = db.Column(db.ForeignKey('auth_permission.id'),
84
85     permission = db.relationship('AuthPermission', primaryjoin='Aut
86     user = db.relationship('AuthUser', primaryjoin='AuthUserUserPeri
87
88
```

```

89 class DjangoAdminLog(db.Model):
90     __tablename__ = 'django_admin_log'
91
92     id = db.Column(db.Integer, primary_key=True)
93     object_id = db.Column(db.Text)
94     object_repr = db.Column(db.String(200), nullable=False)
95     action_flag = db.Column(db.Integer, nullable=False)
96     change_message = db.Column(db.Text, nullable=False)
97     content_type_id = db.Column(db.ForeignKey('django_content_type.
98     user_id = db.Column(db.ForeignKey('auth_user.id'), nullable=False)
99     action_time = db.Column(db.DateTime, nullable=False)
100
101     content_type = db.relationship('DjangoContentType', primaryjoin
102     user = db.relationship('AuthUser', primaryjoin='DjangoAdminLog.
103
104
105 class DjangoContentType(db.Model):
106     __tablename__ = 'django_content_type'
107     __table_args__ = (
108         db.Index('django_content_type_app_label_model_76bd3d3b_uniq
109     )
110
111     id = db.Column(db.Integer, primary_key=True)
112     app_label = db.Column(db.String(100), nullable=False)
113     model = db.Column(db.String(100), nullable=False)
114
115
116 class DjangoMigration(db.Model):
117     __tablename__ = 'django_migrations'
118
119     id = db.Column(db.Integer, primary_key=True)
120     app = db.Column(db.String(255), nullable=False)
121     name = db.Column(db.String(255), nullable=False)
122     applied = db.Column(db.DateTime, nullable=False)
123
124
125 class DjangoSession(db.Model):
126     __tablename__ = 'django_session'
127
128     session_key = db.Column(db.String(40), primary_key=True)
129     session_data = db.Column(db.Text, nullable=False)
130     expire_date = db.Column(db.DateTime, nullable=False, index=True)
131
132
133 t_sqlite_sequence = db.Table(
134     'sqlite_sequence',
135     db.Column('name', db.NullType),

```

```
136         db.Column('seq', db.NullType)
137     )
```

## Flask의 app에 덧붙이기

이렇게 만들어진 model은 다른 Extension과 동일하게 Flask app에 붙일 수 있습니다.

`app.py`라는 파일을 하나 만들고 아래 내용으로 채워주세요.

```
1  # app.py (models.py와 같은 위치)
2  from flask import Flask
3
4  import models # models.py파일을 가져옵니다.
5
6  def create_app():
7      app = Flask(__name__)
8      app.config['SQLALCHEMY_DATABASE_URI'] = "mysql://username:passwo
9      models.db.init_app(app)
10     return app
11
12  if __name__ == '__main__':
13      app = create_app()
14      app.run()
```

앞서 만들어준 `models.py`파일을 가져와 `create_app` 함수를 통해 `app`을 lazy\_loading해주는 과정을 통해 진행해 줄 수 있습니다.

## 마치며

기존에 사용하던 DB를 Flask와 SQLAlchemy를 통해 ORM으로 이용해 좀 더 빠른 개발이 가능하다는 것은 큰 이점입니다. ORM에서 DB 생성을 하지 않더라도 이미 있는 DB를 ORM으로 관리하고 Flask 프로젝트에 바로 가져다 쓸 수 있다는 점이 좀 더 빠른 프로젝트 진행에 도움이 될거립니다.

◀ SQLAlchemy Query를 Pandas DataFrame로 만들기  
Webpack과 Babel로 최신 JavaScript 웹프론트 개발환경 만들기 ▶



