Django Study 03. Models.py와 관리자 계정 생성하기.

Models.py

장고의 애플리케이션에서 models.py를 사용해보도록 한다.
models.py는 장고의 웹 framework와 database를 관리할 수 있는 공간이다. (= 데이터)

장고를 사용해 웹을 만들기 위해 db가 반드시 필요하다. 이를 관리하는 공간이 Model이다.


```
class POST(models.Model):

# 속성 이름 = models.필드타입(옵션=옵션)
```

클래스를 정의해 모델을 정의한다.
https://www.webforefront.com/django/modeldatatypesandvalidation.html

위 사이트에서 사용 가능한 모델 데이터 타입을 확인할 수 있다.

🦴.

*title(제목)*에 CharField(길이가 정해진 문자열)을 사용하고, max_length(길이)가 100인 문자열을 저장한다.
*content(내용)*에 TextField()를 사용해 길이가 정해져있지 않은 문자열을 사용한다.
*created_at(생성 날짜)*를 DateTimeField 함수를 통해 날짜/시간을 지정하고, auto_now_add(데이터 생성할 때 현재 시간 저장)할 수 있도록 했다.
*update_at(수정 날짜)*를 created_at과 같은 방식으로 DateTimeField함수를 사용하고, auto_now를 통해 데이터가 갱신될 때 날짜를 저장할 수 있도록 했다.

📌model을 settings.py에 반영하기.

```
python manage.py makemigrations [앱 이름]
```

해당 명령어를 입력해 앱 내에 작성한 모델 내용의 migrations을 생성할 수 있다.
등록을 해줘야 장고가 우리가 만든 앱을 인식할 수 있기 때문에 모델을 사용하기 위한 필수 과정이다.

```
PS C:\Django\mysite> python manage.py makemigrations main
Migrations for 'main':
  main\migrations\0001_initial.py
    - Create model Post
PS C:\Django\mysite>
```

해당 문구가 출력되면 정상적으로 반영이 된 것이다.
migrations 폴더와 0001_initial.py 파일이 생성되었다.

📌생성한 migrations를 적용하기. (데이터베이스의 테이블 생성하기)

```
python manage.py migrate [앱 이름]
```

python manage.py makemigrations [앱 이름]을 통해 생성된 마이크레이션 파일을 이용해 데이터베이스의 테이블을 생성하려면, 위와같은 작업이 필요하다.

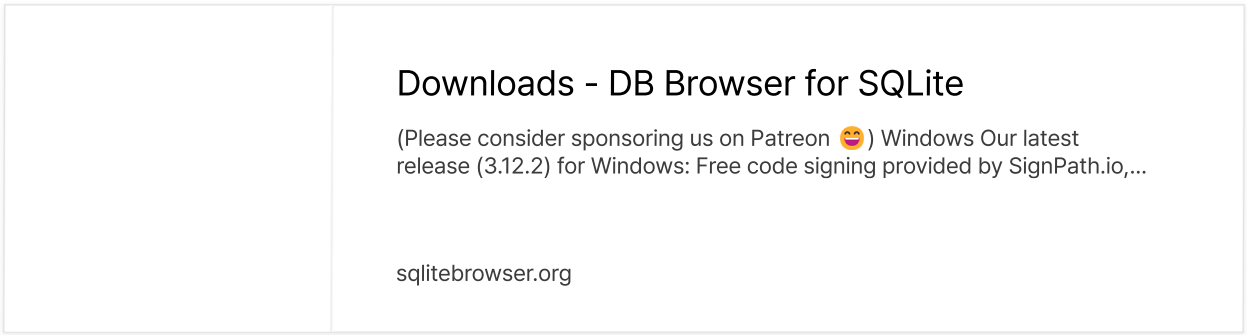
```
PS C:\Django\mysite> python manage.py migrate main
Operations to perform:
  Apply all migrations: main
Running migrations:
  Applying main.0001_initial... OK
PS C:\Django\mysite>
```

해당 문구가 출력되면 정상적으로 반영이 된 것이다.

DB Browser for SQLite 사용하기.

Django의 장점은 별도의 SQL 쿼리문을 사용할 필요가 없고,
쿼리문의 도움 없이 장고의 모델을 통해 ORM을 사용해 데이터 작업을 할 수 있다는 것이다.

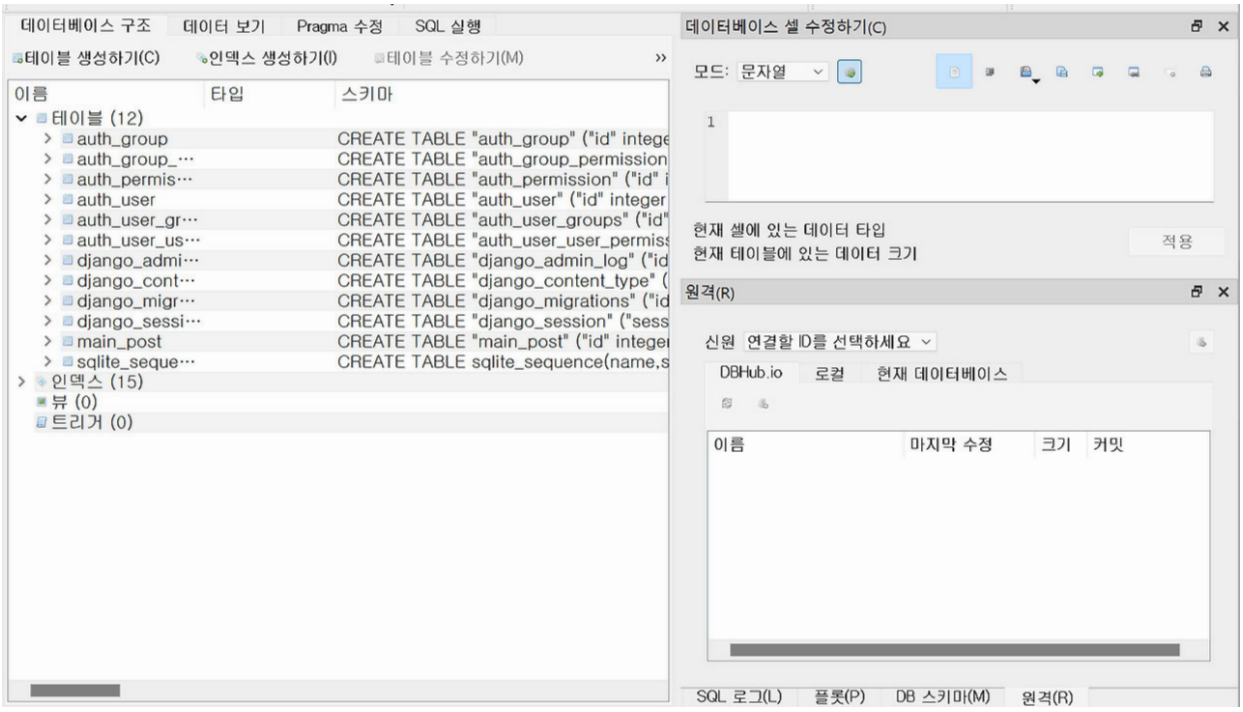
https://sqlitebrowser.org/dl/



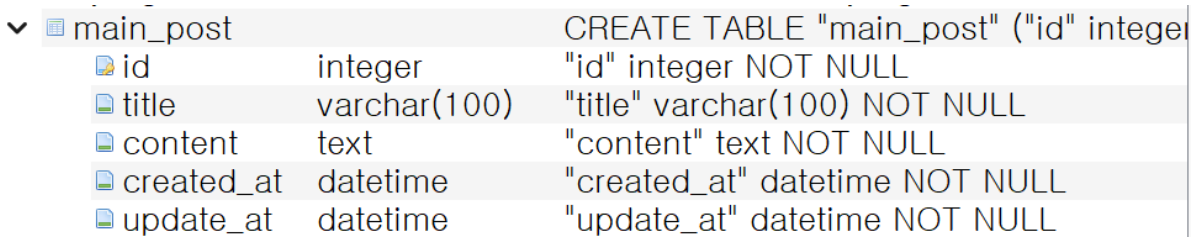
해당 사이트에서 자신의 버전에 맞는 DB Browser for SQLite를 다운로드 받는다.

> ORM의 특징

- 클래스의 객체로 데이터를 조작하므로 더욱 객체 지향적이다.
- SQL문으로 데이터베이스에 접근하지 않는다.
- 객체를 한번 생성하면 그것을 활용해 그 안의 필드 값을 가져오기 용이하다. (재사용이 용이하다.)
- 객체 <- ORM -> 관계형 데이터베이스 / 매핑시켜주는 역할을 한다.
- ORM이 제공하는 Method를 사용하면 자동으로 SQL이 생성되어, 보다 편리하게 매핑이 된다.



파일을 불러와주면 데이터베이스의 테이블들을 확인할 수 있다.
기본적으로 확인할 수 있는 항목은 테이블, 인덱스, 뷰, 트리거 .. 이다.



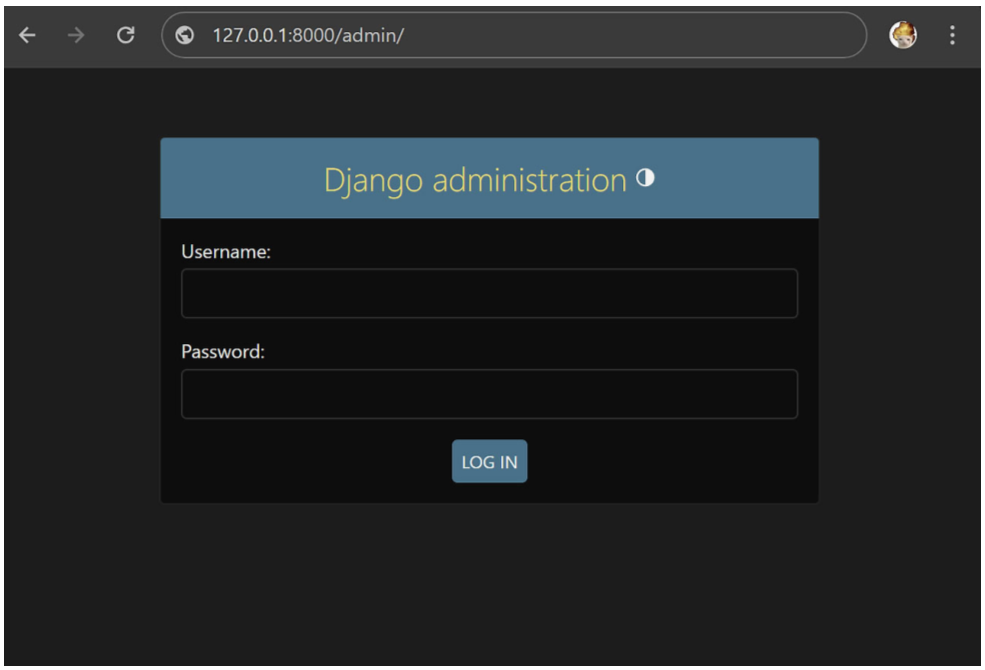
아까 models.py에 작업해둔 class POST 항목을 확인할 수 있다.

관리자 계정(Super User) 생성하기

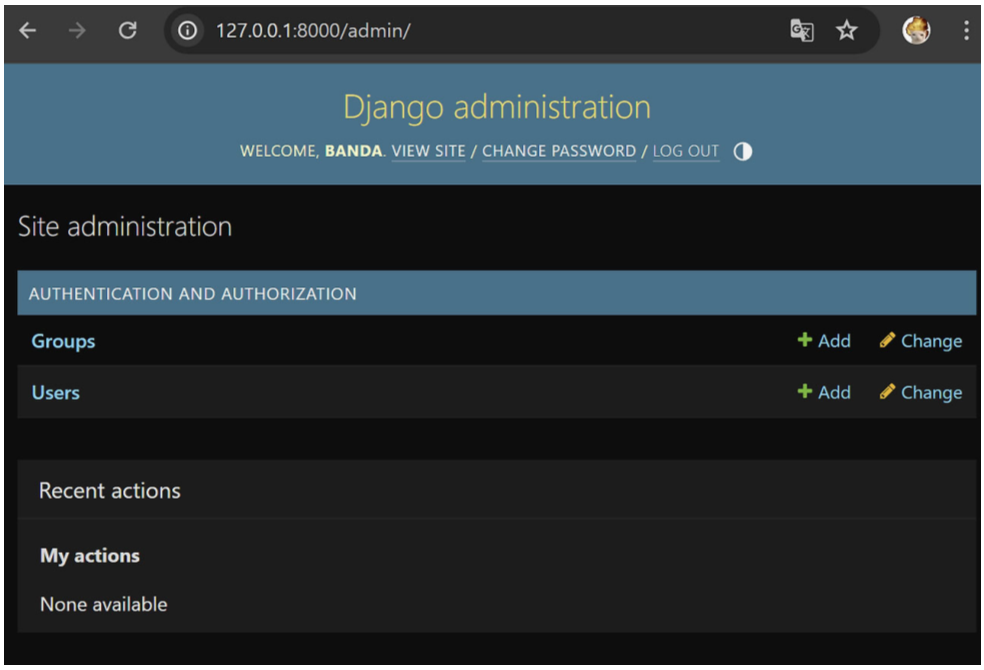
```
PS C:\Django\mysite> python manage.py createsuperuser
Username (leave blank to use 'bangseyeon'): banda
Email address:
Password:
Password (again):
Error: Blank passwords aren't allowed.
Password:
Password (again):
This password is too short. It must contain at least 8 characters.
This password is entirely numeric.
Bypass password validation and create user anyway? [y/N]: y
Superuser created successfully.
```

```
python manage.py createsuperuser
```

해당 명령어를 통해 SUPER USER을 생성할 수 있다.



이제 열어놓은 서버 주소 뒤에 /admin을 입력하면 다음과 같은 페이지를 확인할 수 있다.



로그인을 하면 다음과 같은 관리 창이 나타난다.

📌admin.py 입력을 통해 관리자 페이지 Models 관리하기.

Models에 대한 사전 구축이 완료되었다면, 이제는 관리자 페이지에 모델(Models)를 관리할 준비를 해야한다.

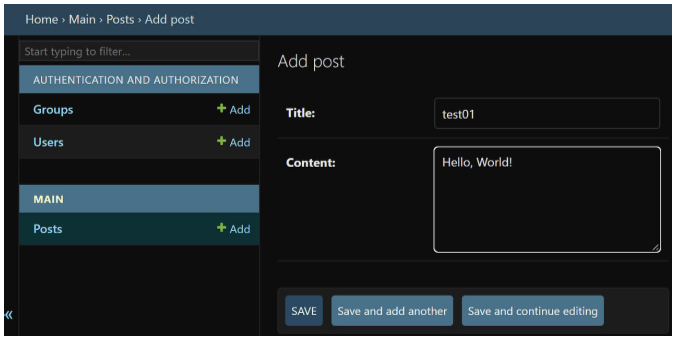

```
from .models import Post
```

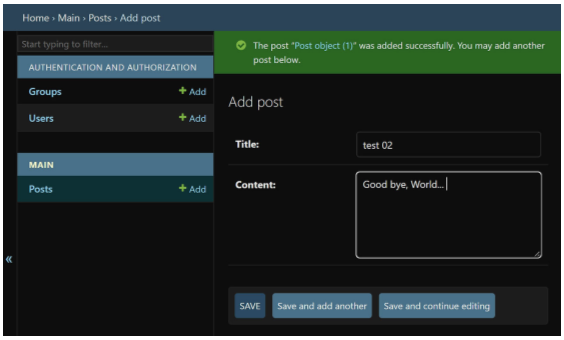
```
admin.site.register(Post)
```

를 입력해주면, 관리자 페이지 화면에 Models가 표시되는 것을 확인할 수 있다.

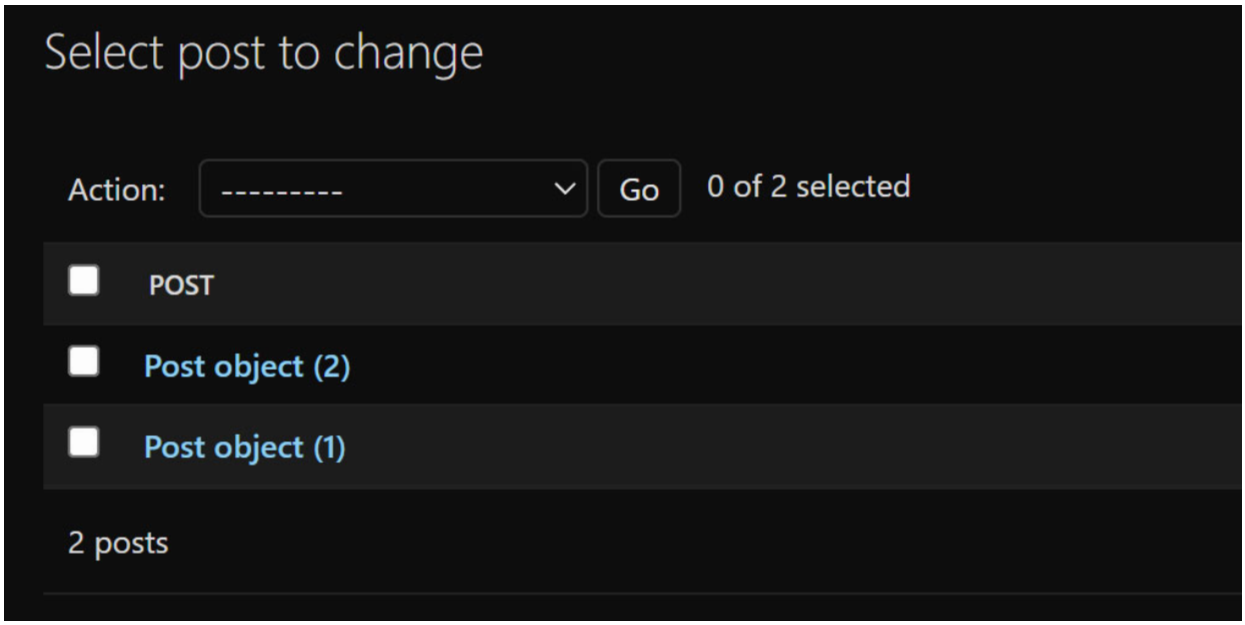
애플리케이션 main에 있는 관리자 페이지 Posts가 생성되었다!!

📌관리자 페이지 Posts에 글 작성하기.





데이터가 잘 저장되는지 테스트를 위해 Main의 Posts의 ➕ADD 버튼을 클릭해
test 01과 test 02 포스트 글을 작성하고 저장해주었다.



두 개의 포스트가 잘 저장된 것을 확인할 수 있다.
DB와 ORM은 나중에 더욱 깊게 들어가볼 수 있는 주제인 것 같다.

장고 스터디를 위해 참고한 사이트 💜:
https://deku.posstree.com/ko/django/admin/
https://velog.io/@2hey9/Django-장고의-기본-요소-익히기-모델