

# Django 03 Model

# • INDEX

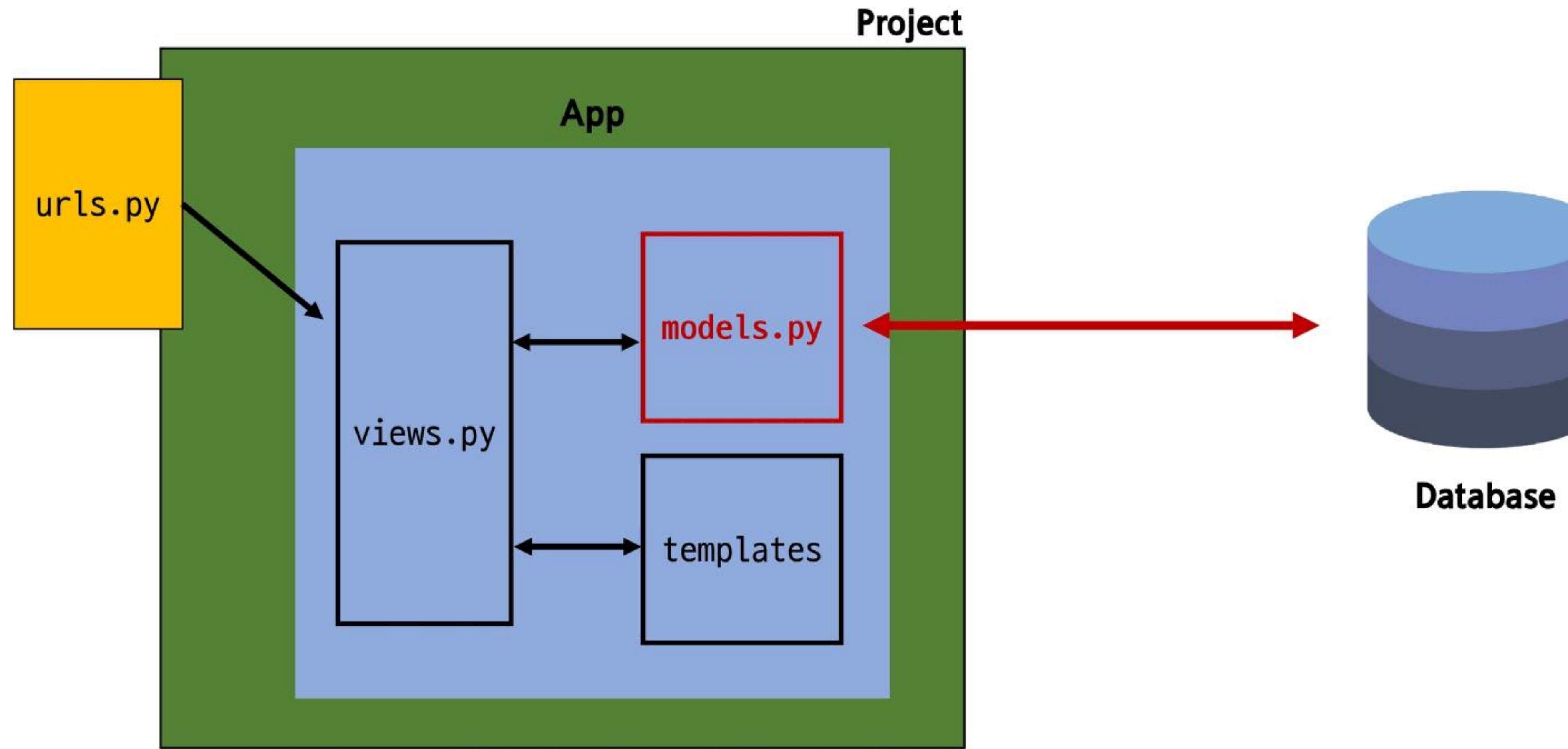
---

- Model
  - model class
- Model Field
  - Field types
  - Field options
- Migrations
  - 추가 Migrations
- Admin site
- 참고
  - 데이터베이스 초기화
  - Migrations 관련
  - SQLite

# Model

# model class

# Model을 통한 DB(데이터베이스) 관리



# Django Model

-----  
DB의 테이블을 정의하고 데이터를 조작할 수 있는 기능들을 제공

- 테이블 구조를 설계하는 ‘청사진(blueprint)’

# model 클래스 작성

```
# articles/models.py

class Article(models.Model):
    title = models.CharField(max_length=10)
    content = models.TextField()
```

# model 클래스 살펴보기 (1/4)

작성한 모델 클래스는 최종적으로 DB에 다음과 같은 테이블 구조를 만듦

```
# articles/models.py  
  
class Article(models.Model):  
    title = models.CharField(max_length=10)  
    content = models.TextField()
```

❖ id 필드는 Django가 자동 생성



id	title	content
..	..	..
..	..	..

‘모델 클래스 == 테이블 설계도’

# model 클래스 살펴보기 (2/4)

- django.db.models 모듈의 Model이라는 부모 클래스를 상속받음
  - Model은 model에 관련된 모든 코드가 이미 작성 되어있는 클래스
    - <https://github.com/django/django/blob/main/django/db/models/base.py#L460>
- 개발자는 가장 중요한 테이블 구조를 어떻게 설계할지에 대한 코드만 작성하도록 하기 위한 것 (상속을 활용한 프레임워크의 기능 제공)

```
# articles/models.py

class Article(models.Model):
    title = models.CharField(max_length=10)
    content = models.TextField()
```

# model 클래스 살펴보기 (3/4)

- 클래스 변수명
  - 테이블의 각 “필드(열) 이름”

```
# articles/models.py

class Article(models.Model):
    title = models.CharField(max_length=10)
    content = models.TextField()
```

id	title	content
..	..	..
..	..	..

# model 클래스 살펴보기 (4/4)

- Model Field
  - 데이터베이스 테이블의 열(column)을 나타내는 중요한 구성 요소
  - “데이터의 유형”과 “제약 조건”을 정의

```
# articles/models.py

class Article(models.Model):
    title = models.CharField(max_length=10)
    content = models.TextField()
```

# 이어서...

삼성 청년 SW 아카데미

# Model Field

# Model Field

-----  
DB 테이블의 필드(열)을 정의하며,  
해당 필드에 저장되는 데이터 타입과 제약조건을 정의

Field types

Field options

# Model Field 구성

1. Field types (필드 유형)
  - 데이터베이스에 저장될 “데이터의 종류”를 정의
2. Field options (필드 옵션)
  - 필드의 “동작”과 “제약 조건”을 정의

# Field types

# Field Types

데이터베이스에 저장될 “데이터의 종류”를 정의  
(models 모듈의 클래스로 정의되어 있음)

```
class Article(models.Model):  
    title = models.CharField(max_length=10)  
    content = models.TextField()
```

# 주요 필드 유형

- 문자열 필드
  - CharField, TextField
- 숫자 필드
  - IntegerField, FloatField
- 날짜/시간 필드
  - DateField, TimeField, DateTimeField
- 파일 관련 필드
  - FileField, ImageField

# CharField()

---

제한된 길이의 문자열을 저장  
(필드의 최대 길이를 결정하는 `max_length`는 필수 옵션)

# TextField()

---

길이 제한이 없는 대용량 텍스트를 저장  
(무한대는 아니며 사용하는 시스템에 따라 달라짐)

## Field options

# Field Options

필드의 “동작”과 “제약 조건”을 정의

```
class Article(models.Model):
    title = models.CharField(max_length=10)
    content = models.TextField()
```

## 주요 필드 옵션

- **null**
  - 데이터베이스에서 NULL 값을 허용할지 여부를 결정 (기본값: False)
- **blank**
  - form에서 빈 값을 허용할지 여부를 결정 (기본값: False)
- **default**
  - 필드의 기본값을 설정

# 제약 조건

## Constraint

특정 규칙을 강제하기 위해 테이블의 열이나 행에  
적용되는 규칙이나 제한사항

- ex) 숫자만 저장되도록, 문자가 100자 까지만 저장되도록 하는 등

# 이어서...

삼성 청년 SW 아카데미

# Migrations

---

# Migrations

---

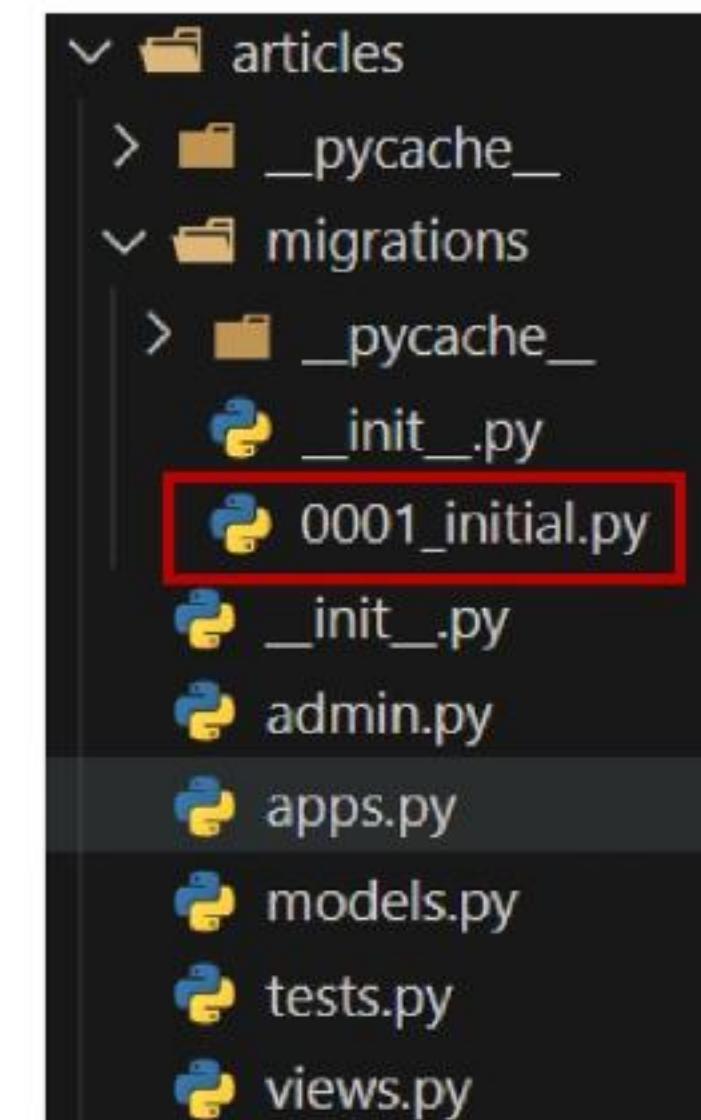
model 클래스의 변경사항(필드 생성, 수정 삭제 등)을  
DB에 최종 반영하는 방법

# Migrations 과정

```
# articles/models.py  
  
class Article(models.Model):  
    title = models.CharField(max_length=10)  
    content = models.TextField()
```

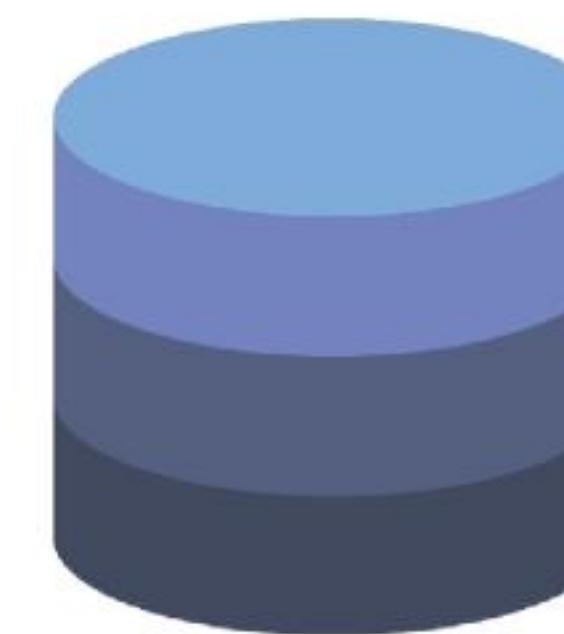
model class  
(설계도 초안)

makemigrations



migration 파일  
(최종 설계도)

migrate



db.sqlite3  
(DB)

## Migrations 핵심 명령어 2가지

---

```
$ python manage.py makemigrations
```

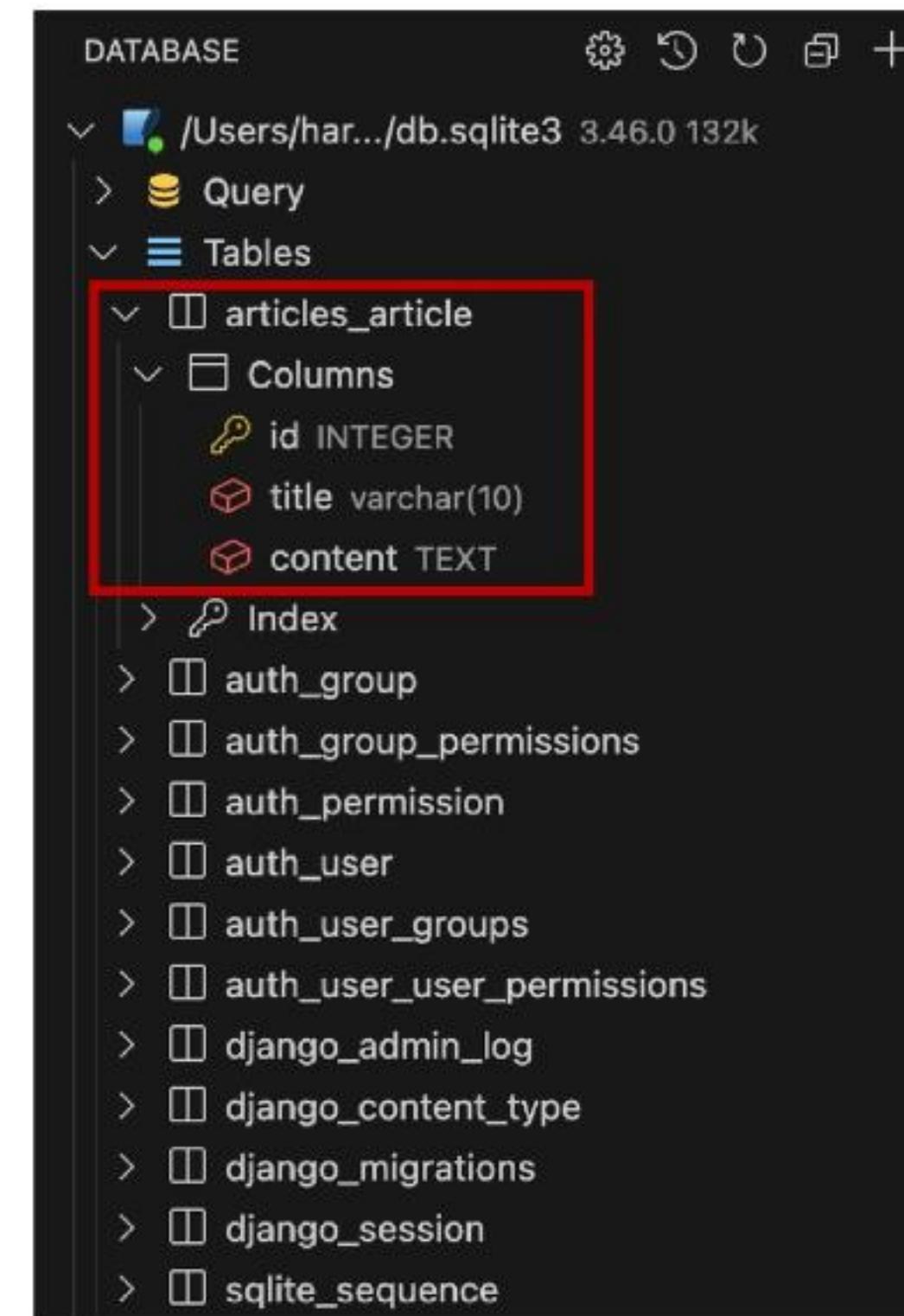
model class를 기반으로  
최종 설계도(migration) 작성

```
$ python manage.py migrate
```

최종 설계도를  
DB에 전달하여 반영

# migrate 후 DB 내에 생성 된 테이블 확인

Article 모델 클래스로 만들어진 `articles_article` 테이블



# 추가 Migrations

# 이미 생성된 테이블에 필드를 추가해야 한다면?

<code>id</code>	<code>title</code>	<code>content</code>	<code>created_at</code>	<code>updated_at</code>
...	...	...	...	...
...	...	...	...	...
...	...	...	...	...

## 추가 모델 필드 작성 (1/5)

```
# articles/models.py

class Article(models.Model):
    title = models.CharField(max_length=10)
    content = models.TextField()
    created_at = models.DateTimeField(auto_now_add=True)
    updated_at = models.DateTimeField(auto_now=True)
```

# DateTimeField의 필드 옵션 (optional)

auto\_now

데이터가 저장될 때마다  
자동으로 현재 날짜시간을 저장

auto\_now\_add

데이터가 처음 생성될 때만  
자동으로 현재 날짜시간을 저장

## 추가 모델 필드 작성 (2/5)

- 이미 기존 테이블이 존재하기 때문에 필드를 추가 할 때 필드의 기본 값 설정이 필요
- 1번은 현재 대화를 유지하면서 직접 기본 값을 입력 하는 방법 
- 2번은 현재 대화에서 나간 후 models.py에 기본 값 관련 설정을 하는 방법

```
$ python manage.py makemigrations
```

It is impossible to add the field 'created\_at' with 'auto\_now\_add=True' to article without providing a default. This is because the database needs something to populate existing rows.

- 1) Provide a one-off default now which will be set on all existing rows
- 2) Quit and manually define a default value in models.py.

Select an option:

## 추가 모델 필드 작성 (3/5)

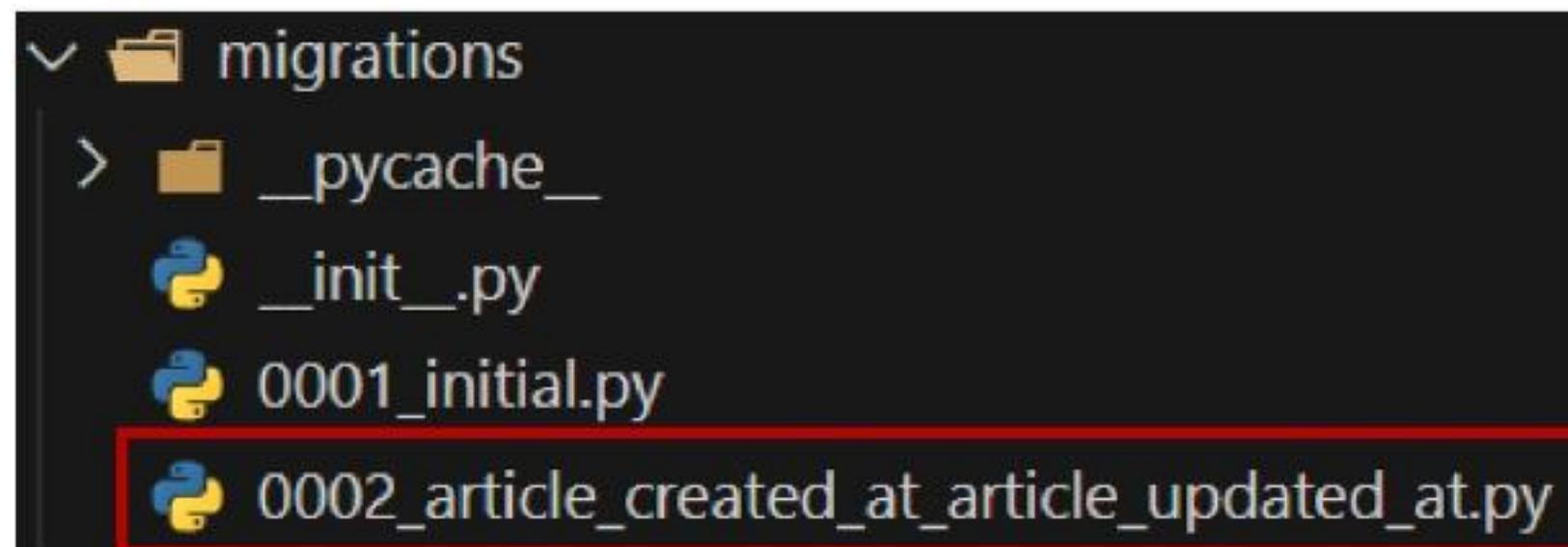
- 추가하는 필드의 기본 값을 입력해야 하는 상황
- 날짜 데이터이기 때문에 직접 입력하기 보다 Django가 제안하는 기본 값을 사용하는 것을 권장
- 아무것도 입력하지 않고 enter를 누르면 Django가 제안하는 기본 값으로 설정 됨

```
Please enter the default value as valid Python.  
Accept the default 'timezone.now' by pressing 'Enter' or provide another value.  
The datetime and django.utils.timezone modules are available, so it is possible to  
provide e.g. timezone.now as a value.  
Type 'exit' to exit this prompt  
[default: timezone.now] >>>
```

## 추가 모델 필드 작성 (4/5)

- migrations 과정 종료 후 2번째 migration 파일이 생성됨을 확인
- 이처럼 Django는 설계도를 쌓아가면서 추후 문제가 생겼을 시 복구하거나 되돌릴 수 있도록 함 (마치 ‘git commit’과 유사)

```
Migrations for 'articles':  
    articles\migrations\0002_article_created_at_article_updated_at.py  
        - Add field created_at to article  
        - Add field updated_at to article
```



# 추가 모델 필드 작성 (5/5)

migrate 후 테이블 필드 변화 확인

```
$ python manage.py migrate
```

The screenshot shows a SQLite database browser interface with the following tree structure:

- /Users/har.../db.sqlite3 3.46.0 136k
  - Query
- Tables
  - articles\_article
    - Columns
      - id INTEGER (Primary Key)
      - title varchar(10)
      - content TEXT
      - created\_at datetime
      - updated\_at datetime
  - Index

**model class에 변경사항(1)이 생겼다면,  
반드시 새로운 설계도를 생성(2)하고,  
이를 DB에 반영(3)해야 한다.**

1. model class 변경 → 2. makemigrations → 3. migrate

# 이어서...

삼성 청년 SW 아카데미

# Admin site

---

# Automatic admin interface

---

Django가 추가 설치 및 설정 없이 자동으로 제공하는  
관리자 인터페이스

- 데이터 확인 및 테스트 등을 진행하는데 매우 유용

# 1. admin 계정 생성

- email은 선택사항이기 때문에 입력하지 않고 진행 가능
- 비밀번호 입력 시 보안상 터미널에 출력되지 않으니 무시하고 입력 이어가기

```
$ python manage.py createsuperuser
```

## 2. DB에 생성된 admin 계정 확인

The screenshot shows a SQLite database browser interface with the following details:

- Left Panel (Database Tree):** Shows the database file path: /Users/har.../db.sqlite3, version 3.46.0, size 136k. It lists tables: articles\_article, auth\_group, auth\_group\_permissions, auth\_permission, auth\_user, auth\_user\_groups, auth\_user\_user\_permissions, django\_admin\_log, django\_content\_type, django\_migrations, django\_session, sqlite\_sequence, and Views.
- Top Bar:** Properties and DATA tabs are visible. The DATA tab is selected, showing a query: SELECT \* FROM auth\_user LIMIT 100.
- Toolbar:** Includes icons for Query, Tables, Columns, Index, auth\_user, Export, and other database operations.
- Results Grid:** The auth\_user table data is displayed in a grid format. The columns are: id, password, last\_login, is\_superuser, username, last\_name, email, is\_staff, and is\_active. The first row of data is highlighted with a red box and contains:
  - id: >1
  - password: pbkdf2\_sha256\$600000\$xDE
  - last\_login: (NULL)
  - is\_superuser: 1
  - username: admin
  - last\_name:
  - email:
  - is\_staff: 1
  - is\_active: 1

### 3. admin에 모델 클래스 등록

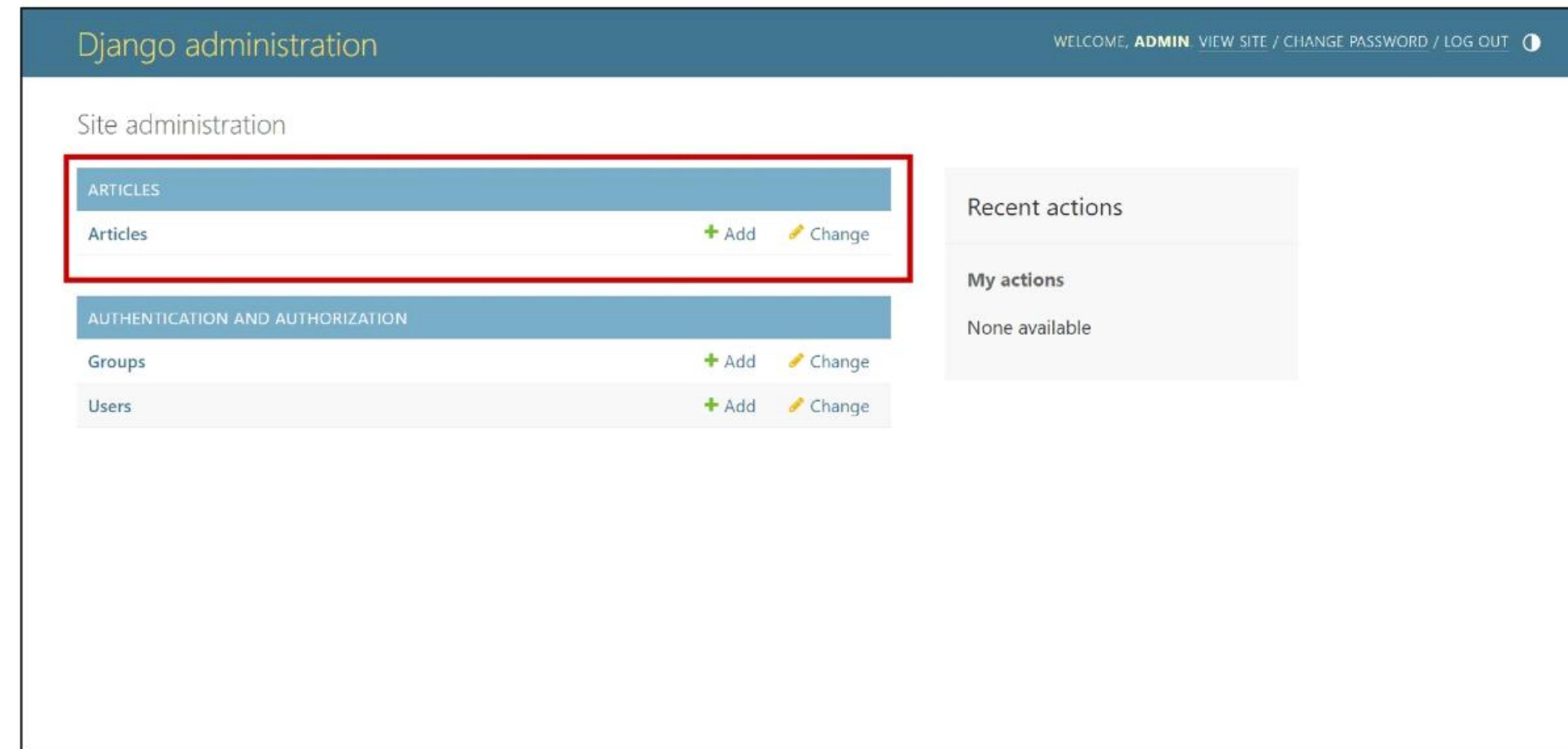
admin.py에 작성한 모델 클래스를 등록해야만 admin site에서 확인 가능

```
# articles/admin.py

from django.contrib import admin
from .models import Article

admin.site.register(Article)
```

## 4. admin site 로그인 후 등록된 모델 클래스 확인



# 5. 데이터 생성, 수정, 삭제 테스트

The image displays two screenshots of the Django admin interface, illustrating the process of creating and modifying data.

**Screenshot 1: Article Creation**

This screenshot shows the 'Articles' list page. The sidebar on the left includes sections for 'ARTICLES' (Articles, + Add), 'AUTHENTICATION AND AUTHORIZATION' (Groups, + Add; Users, + Add), and a search bar ('Start typing to filter...'). The main area is titled 'Select article to change' and shows '0 articles'. A prominent red box highlights the 'ADD ARTICLE +' button at the top right.

**Screenshot 2: Article Modification**

This screenshot shows the 'Article object (1)' detail page for an article titled '첫번째'. The sidebar on the left shows 'ARTICLES' (Articles, + Add) and 'AUTHENTICATION AND AUTHORIZATION' (Groups, + Add; Users, + Add). The main area is titled 'Change article' and contains fields for 'Title' (첫번째) and 'Content' (개시글!!). At the bottom, there are buttons for 'SAVE' (highlighted with a red box), 'Save and add another', 'Save and continue editing', and 'Delete' (highlighted with a red box).

## 6. 테이블 확인

The screenshot shows a SQLite database browser interface. On the left, the database tree is visible, with the current connection set to `/Users/har.../db.sqlite3`. Under the `Tables` section, the `articles_article` table is selected. The main area displays the contents of the `articles_article` table, which has five columns: `id`, `title`, `content`, `created_at`, and `updated_at`. A single row is shown with the following values:

		Filter	Filter	Filter	Filter
	> 1	1	첫번째	게시글!	[REDACTED]

A red box highlights the entire data grid. At the top of the data grid, there are several icons: a gear, a mail icon with a green notification bubble containing the number '1', a plus sign, a minus sign, a refresh, an up arrow, a down arrow, an export button, and a cost analysis button.

# 이어서...

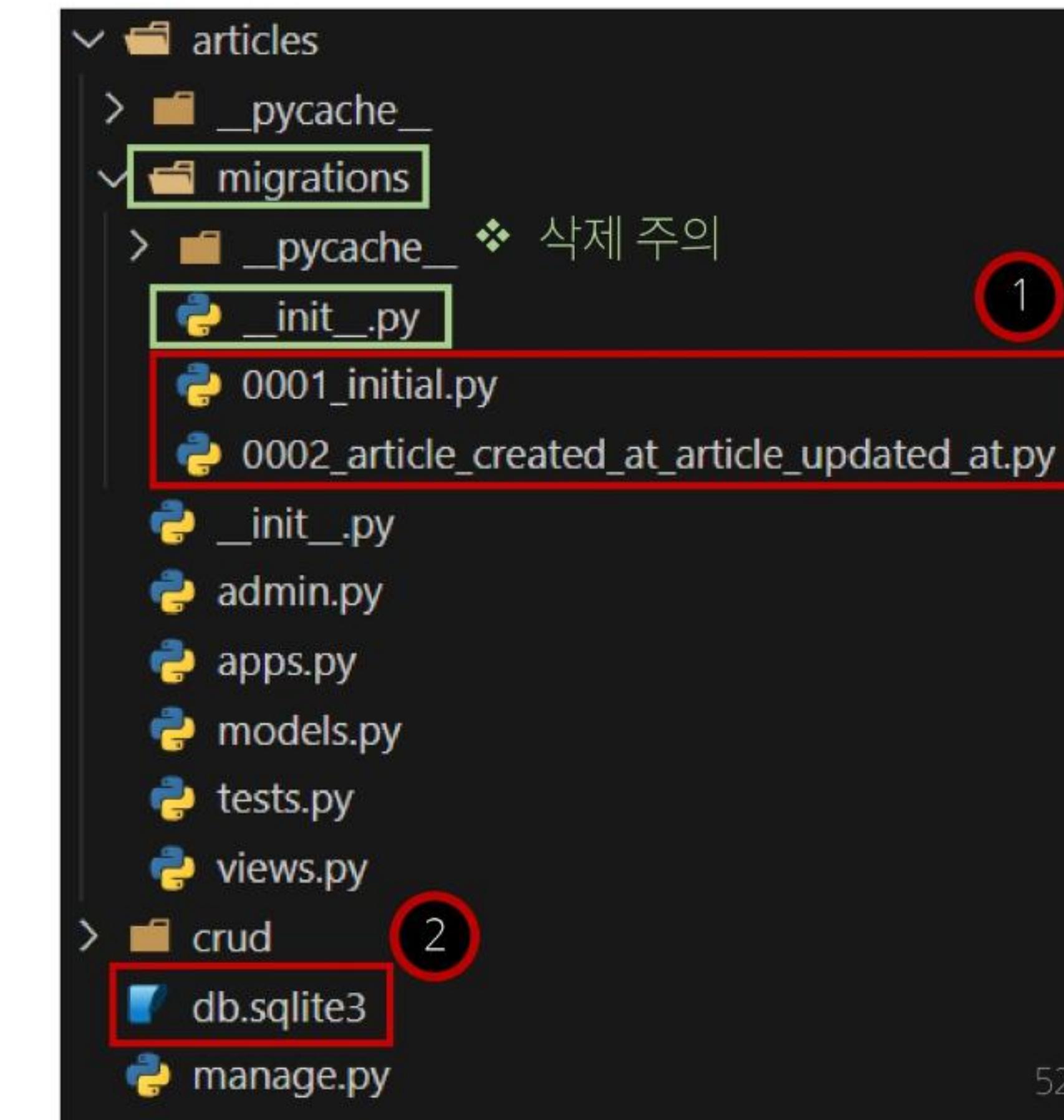
삼성 청년 SW 아카데미

# 참고

# 데이터베이스 초기화

# 데이터베이스 초기화

1. migration 파일 삭제
  2. db.sqlite3 파일 삭제
- ❖ 아래 파일과 폴더를 지우지 않도록 주의
- `__init__.py`
  - `migrations` 폴더



# Migrations 관련

# Migrations 기타 명령어

```
$ python manage.py showmigrations
```

- migrations 파일들이 migrate 됐는지 안됐는지 여부를 확인하는 명령어
- [X] 표시가 있으면 migrate가 완료되었음을 의미

```
$ python manage.py sqlmigrate articles 0001
```

- 해당 migrations 파일이 SQL 언어(DB에서 사용하는 언어)로 어떻게 번역 되어 DB에 전달되는지 확인하는 명령어

# 첫 migrate 시 출력 내용이 많은 이유는?

Django 프로젝트가 동작하기 위해 미리 작성 되어있는 기본 내장 app들에 대한 migration 파일들이 함께 migrate 되기 때문

```
$ python manage.py migrate
Operations to perform:
  Apply all migrations: admin, auth, contenttypes, sessions
Running migrations:
  Applying contenttypes.0001_initial... OK
  Applying auth.0001_initial... OK
  Applying admin.0001_initial... OK
  Applying admin.0002_logentry_remove_auto_add... OK
  Applying admin.0003_logentry_add_action_flag_choices... OK
  Applying contenttypes.0002_remove_content_type_name... OK
  Applying auth.0002_alter_permission_name_max_length... OK
  Applying auth.0003_alter_user_email_max_length... OK
  Applying auth.0004_alter_user_username_opts... OK
  Applying auth.0005_alter_user_last_login_null... OK
  Applying auth.0006_require_contenttypes_0002... OK
  Applying auth.0007_alter_validators_add_error_messages... OK
  Applying auth.0008_alter_user_username_max_length... OK
  Applying auth.0009_alter_user_last_name_max_length... OK
  Applying auth.0010_alter_group_name_max_length... OK
  Applying auth.0011_update_proxy_permissions... OK
  Applying auth.0012_alter_user_first_name_max_length... OK
  Applying sessions.0001_initial... OK
```

```
# settings.py

INSTALLED_APPS = [
    'articles',
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
]
```

# SQLite



-----  
데이터베이스 관리 시스템 중 하나이며  
Django의 기본 데이터베이스로 사용됨  
(파일로 존재하며 가볍고 호환성이 좋음)

다음 시간에  
만나요!

삼성 청년 SW 아카데미