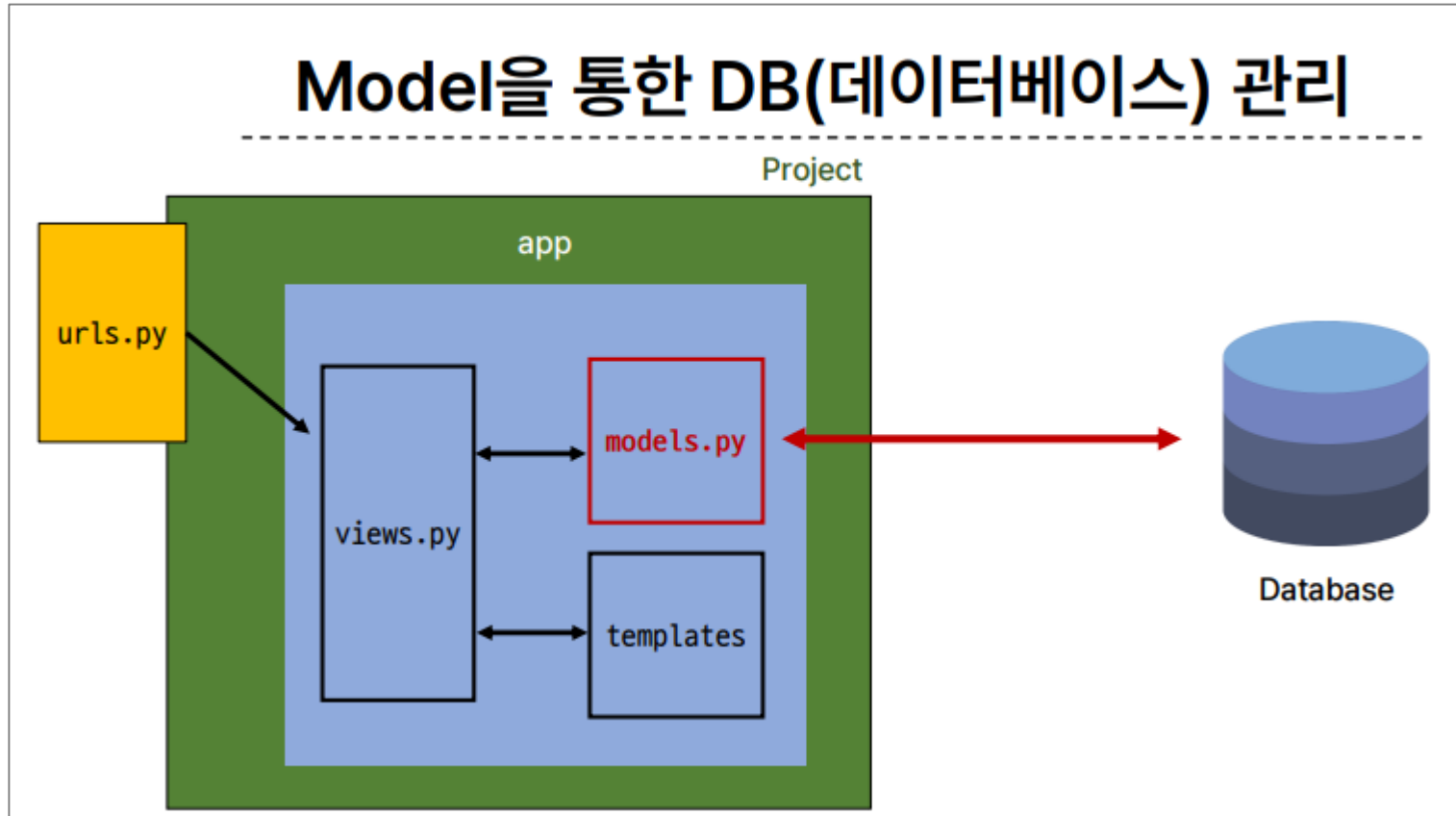


Django Model

Django Model

DB의 테이블을 정의하고 데이터를 조작할 수 있는 기능들을 제공



Django Model Class


models.py 에 생성할 DB 클래스 정의

- models.Model 부모 클래스를 상속받아서 작성
- 테이블 설계에만 집중할 수 있도록 간단하게 테이블 작성을 도움

```
# articles/models.py
from django.db import models

class Article(models.Model):
    title = models.CharField(max_length=20)
    content = models.TextField()
```

Article Table을 생성하는 Class

 id		title		* content	
bigint	↕	varchar	↕	longtext	↕

생성된 Article 테이블

Django Model Class

1. 클래스명

- 생성할 테이블명
- 실제로 생성되는 테이블명은 "앱이름_클래스명 " 으로 생성

```
# articles/models.py
from django.db import models

class Article(models.Model):
    title = models.CharField(max_length=20)
    content = models.TextField()
```

Django Model Class

2. 변수명

- 테이블의 각 필드명

```
# articles/models.py
from django.db import models

class Article(models.Model):
    title = models.CharField(max_length=20)
    content = models.TextField()
```

Django Model Class

3. models 모듈의 Field Class

- 필드의 데이터 타입
- CharField: 길이의 제한이 있는 문자열을 넣을 때 사용
- TextField: 글자의 수가 많을 때 사용
- 다양한 데이터 타입은 [Django 공식 홈페이지](#) 참고

```
# articles/models.py
from django.db import models

class Article(models.Model):
    title = models.CharField(max_length=20)
    content = models.TextField()
```

Django Model Class

4. models 모듈의 Field Class 의 키워드 인자 (필드 옵션)

- 필드의 제약조건 설정
- 다양한 제약조건은 [Django 공식 홈페이지](#) 참고

```
# artices/models.py
from django.db import models

class Article(models.Model):
    title = models.CharField(max_length=20)
    content = models.TextField()
```

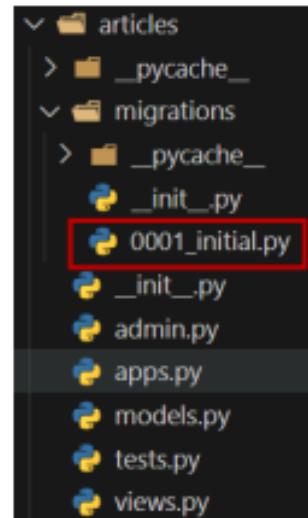
Django Migrations

model 클래스의 변경사항(필드 생성/수정/삭제 등)을 DB에 최종 반영

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

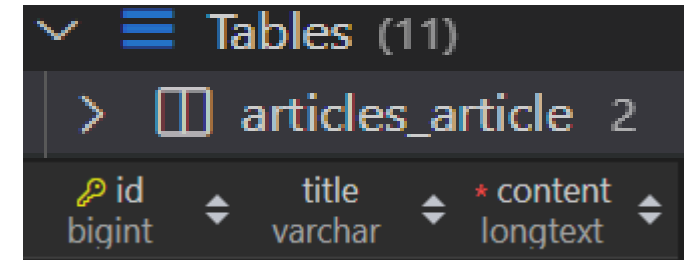
model class
(설계도 초안)

makemigrations



migration 파일
(최종 설계도)

migrate

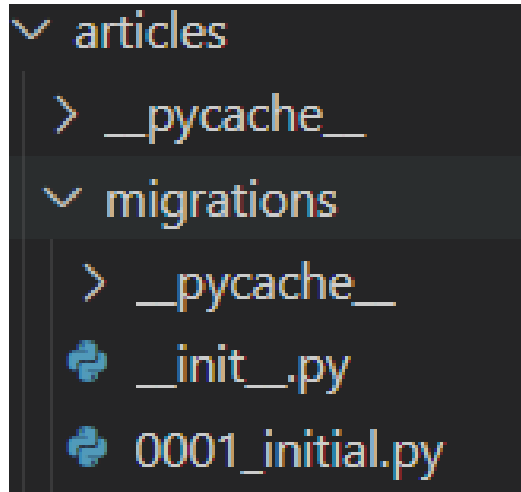


DB

Django Migrations

1. model class 를 기반으로 설계도를 생성하는 명령어 실행

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



```
# articles/migrations/0001_initial.py
from django.db import migrations, models

class Migration(migrations.Migration):

    initial = True

    dependencies = [
    ]

    operations = [
        migrations.CreateModel(
            name='Article',
            fields=[
                ('id', models.BigAutoField(auto_created=True, primary_key=True, serializable=True)),
                ('title', models.CharField(max_length=10)),
                ('content', models.TextField()),
            ],
        ),
    ]
```

Django Migrations

2. 생성한 설계도를 DB에 전달해서 반영

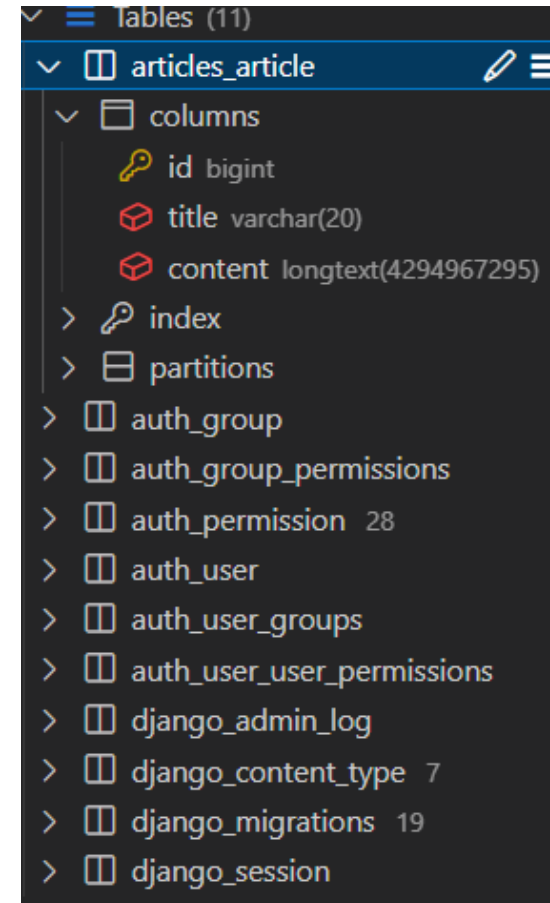
```
$ python manage.py migrate
```

Operations to perform:

Apply all migrations: admin, articles, 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 articles.0001_initial... 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
```



Tables (11)	
articles_article	
columns	
id	bigint
title	varchar(20)
content	longtext(4294967295)
index	
partitions	
auth_group	
auth_group_permissions	
auth_permission	28
auth_user	
auth_user_groups	
auth_user_user_permissions	
django_admin_log	
django_content_type	7
django_migrations	19
django_session	

Django Model 수정

1. 추가할 필드 작성

- DateTimeField: 날짜와 시간을 넣을 때 사용
- auto_now_add: 데이터가 처음 생성될 때만 자동으로 현재 시간 저장
- auto_now: 데이터가 저장/수정될 때마다 자동으로 현재 시간 저장

```
# articles/models.py
from django.db import models

class Article(models.Model):
    title = models.CharField(max_length=20)
    content = models.TextField()
    created_at = models.DateTimeField(auto_now_add=True)
    updated_at = models.DateTimeField(auto_now=True)
```

Django Model 수정

2. 설계도 생성

- 테이블이 존재하기 때문에 기존 데이터에 추가할 필드의 값을 채워줘야 함
- 일반적으로는 Django 기본값으로 설정 후 필요에 따라 SQL문을 작성하여 임의의 값을 작성

```
$ 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: ☐
```

Django Model 수정

3. 기존 데이터들의 값들의 세팅을 설정

- 1 을 눌러서 Django 기본 지원 세팅을 사용
- 이후에 아무것도 입력하지 않고 엔터를 입력해서 설게도 생성 진행
- git commit 과 같이 설게도도 지속해서 버전을 관리

```
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: 1
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] >>>
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
```

```
▼ articles
  > __pycache__
  ▼ migrations
    > __pycache__
    • __init__.py
    • 0001_initial.py
    • 0002_article_created_at_article_updated_at.py
```

Django Model 수정

4. 설계도를 기준으로 데이터베이스에 반영

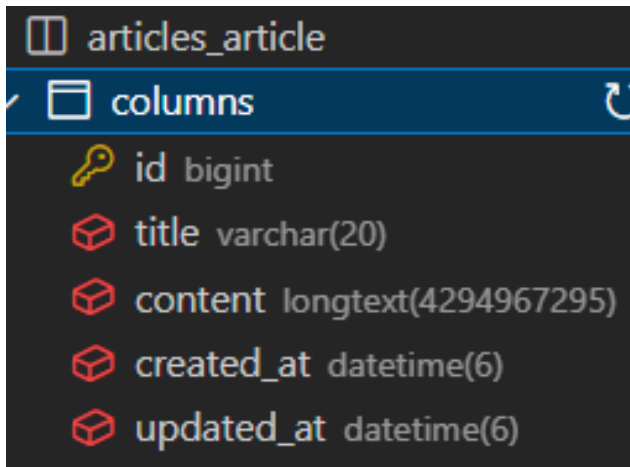
```
$ python manage.py migrate
```

Operations to perform:

Apply all migrations: admin, articles, auth, contenttypes, sessions

Running migrations:

Applying articles.0002_article_created_at_article_updated_at... OK



articles_article
columns
id bigint
title varchar(20)
content longtext(4294967295)
created_at datetime(6)
updated_at datetime(6)

Django Migrations 정리

1. `models.py` 파일에 `model class` 작성
2. “`python manage.py makemigrations`” 명령어로 설계도 생성
3. “`python manage.py migrate`” 명령어로 설계도를 DB에 반영