Ask Company


리액트와 함께 장고 시작하기 / 장고 Models

# 마이그레이션을 통한 데이터베이스 스키마 관리

여러분의 파이썬/장고 페이스메이커가 되겠습니다.

# Migrations

모델의 변경내역을 “데이터베이스 스키마 " 로 반영시키는 효율적인 방법을 제공

관련 명령

마이그레이션 파일 생성

쉘> python manage.py **makemigrations** <앱이름>

지정 데이터베이스에 마이그레이션 적용

쉘> python manage.py **migrate** <앱이름>

마이그레이션 적용 현황 출력

쉘> python manage.py **showmigrations** <앱이름>

지정 마이그레이션의 SQL 내역 출력

쉘> python manage.py **sqlmigrate** <앱이름> <마이그레이션-이름>

# Migration 파일

## 데이터베이스에 어떤 변화를 가하는 Operation들을 나열

테이블 생성/삭제, 필드 추가/삭제 등

커스텀 파이썬/SQL Operation

데이터 마이그레이션 등

## 대개 모델로부터 자동 생성 ➔ makemigrations 명령

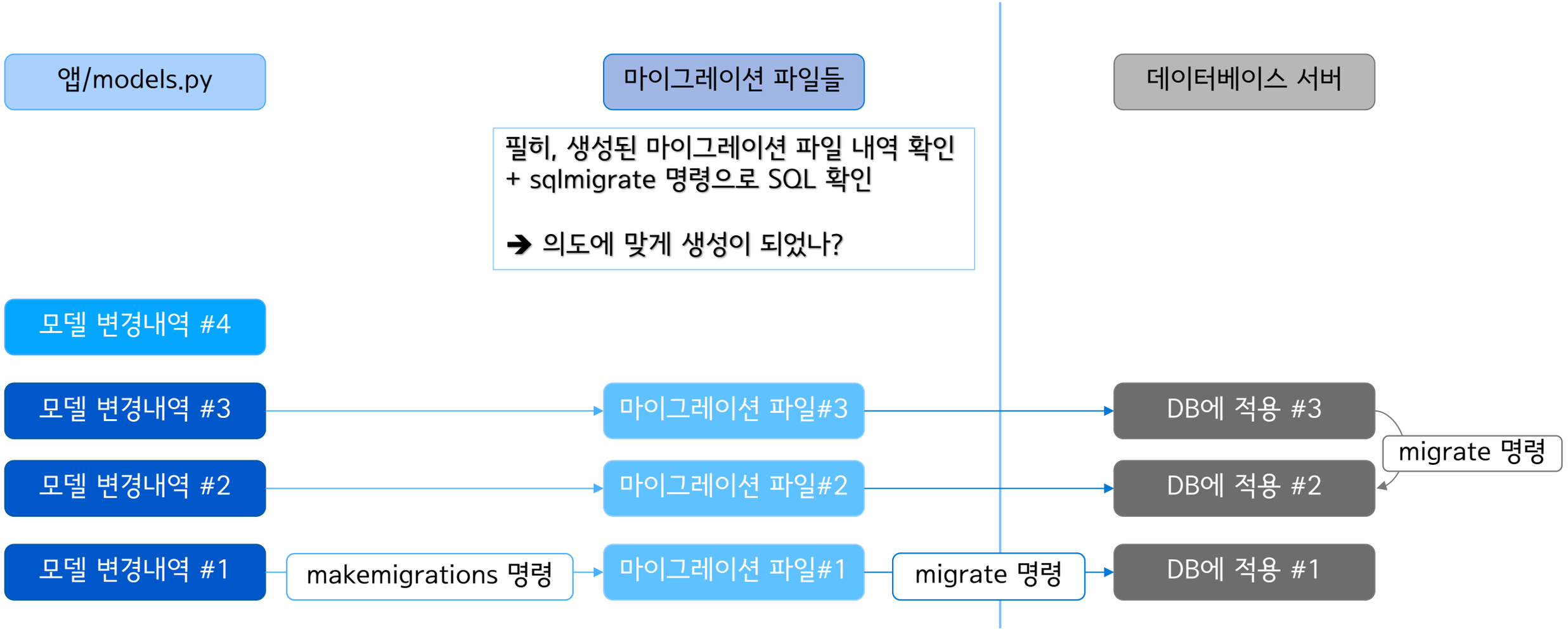
모델 참조없이 빈 마이그레이션 파일 만들어서 직접 채워넣기도.

## 주의) 같은 Migration 파일이라 할지라도, DB종류에 따라 다른 SQL이 생성됩니다.

모든 데이터베이스 엔진들이 같은 기능을 제공하진 않아요.

예) SQLite DB에서는 기존 테이블에 컬럼 추가가 지원되지 않습니다.

# 마이그레이션 파일 생성 및 적용



# 언제 makemigrations를 하는 가?

## 모델 필드 관련된 어떠한 변경이라도 발생 시에 마이그레이션 파일 생성

실제로 DB Scheme에 가해지는 변화가 없더라도 수행.

## 마이그레이션 파일은 **모델의 변경내역을 누적**하는 역할

적용된 마이그레이션 파일은 절대 삭제하시면 안 됩니다.

마이그레이션 파일이 너무 많아질 경우, squashmigrations 명령으로 다수의 마이그레이션 파일을 통합할 수 있습니다.

# 마이그레이션 Migrate (정/역 방향)

## python manage.py migrate <앱이름>

미적용 <마이그레이션-파일>부터 <최근-마이그레이션-파일>까지 **정방향으로 순차적으로** 수행

## python manage.py migrate <앱이름> <마이그레이션-이름>

지정된 <마이그레이션-이름>이 현재 적용된 마이그레이션보다

이후라면, 정방향으로 순차적으로 지정 마이그레이션까지 forward 수행

이전이라면, 역방향으로 순차적으로 지정 마이그레이션 이전까지 backward 수행

# 마이그레이션 이름 지정

전체 이름(파일명)을 지정하지 않더라도, 1개를 판별할 수 있는 일부만 지정해도 OK

```
shop/migrations/0001_initial.py

shop/migrations/0002_create_field.py

shop/migrations/0002_update_field.py


쉘> python manage.py migrate blog 000          # FAIL (3개 파일에 매칭)
쉘> python manage.py migrate blog 100          # FAIL (매칭되는 파일이 없음)
쉘> python manage.py migrate blog 0001         # OK
쉘> python manage.py migrate blog 0002         # FAIL (2개 파일에 매칭)
쉘> python manage.py migrate blog 0002_c       # OK
쉘> python manage.py migrate blog 0002_create  # OK
쉘> python manage.py migrate blog 0002_update  # OK
쉘> python manage.py migrate blog zero         # shop앱의 모든 마이그레이션을 rollback
```

# 마이그레이션 순서는 파일명으로 정렬순?

```
from django.db import migrations, models


class Migration(migrations.Migration):

    dependencies = [
        ('shop', '0001_initial'),
    ]

    operations = [
        …
    ]
```

# id필드는 왜 생기나요?

## 모든 DB 테이블에는 각 Row의 식별기준인 “기본키 (Primary Key)”가 필요

장고에서는 기본키로서 id (AutoField) 필드를 디폴트 생성

다른 필드를 기본키로 지정하고 싶다면 primary_key=True 옵션 적용

# 새로운 필드가 필수필드라면?

필수필드 여부 : blank/null 옵션이 모두 False 일 때 (디폴트)

makemigrations 명령을 수행할 때, 기존 Record들에 어떤 값을 채워넣을 지 묻습니다.

선택1) 지금 그 값을 입력하겠다.

선택2) 명령 수행을 중단.

```
You are trying to add a non-nullable field 'tags' to post without a default; we can't do that (the database needs someth
ing to populate existing rows).
Please select a fix:
 1) Provide a one-off default now (will be set on all existing rows with a null value for this column)
 2) Quit, and let me add a default in models.py
Select an option:
```

# 협업 Tip

## 절대 하지 말아야할 일

팀원 각자가 마이그레이션 파일을 생성 ➔ 충돌 발생

## 추천) 마이그레이션 파일 생성은 1명이 전담해서 생성.

생성한 마이그레이션 파일을 **버전관리**에 넣고, 다른 팀원들은 이를 받아서 migrate만 수행

# Tip

개발 시에 “서버에 아직 반영하지 않은” 마이그레이션을 다수 생성했었다면?

- 이를 그대로 서버에 반영(migrate)하시지 마시고,
- **하나의 마이그레이션으로 합쳐서 적용**하기를 권장.
  - 방법1) 서버로의 미적용 마이그레이션들을 모두 롤백하고 ➔ 롤백된 마이그레이션들을 모두 제거하고 ➔ 새로이 마이그레이션 파일 생성
  - 방법2) 미적용 마이그레이션들을 하나로 합치기 ➔ squashmigrations 명령

Life is short.
You need Python and Django.

I will be your pacemaker.