Pystargram Tutorial

3. Photo 앱과 모델 만들기

1. Django Project와 App

1. 개념

- Python 모듈: Python 코드가 담긴 파일
- Python 패키지: Python 모듈을 묶어놓은 단위
- Django project: Python 패키지 단위: `__init__.py`
- Django framework이 참조할 프로젝트 설정: `settings. py`

2. Django project 만들기

- `django-admin.py` 로 필수 모듈 `__init__.py`와 `settings.py` 만든다
- `django-admin startproject pystagram`
- pystagram 구성

```
pystagram/
├── manage.py
└── pystagram
    ├── __init__.py
    ├── settings.py
    ├── urls.py
    └── wsgi.py
```

- manage.py와 개발용 내장 웹 서버
  - 내장 웹 서버 실행: `python manage.py runserver`
  - 웹 브라우저 실행: `http://localhost:8000`

3. Pystagram Project 초기/사전 작업

- 데이터베이스 동기화: `python manage.py migrate`
- superuser 설정: `python manage.py createsuperuser`
- `db.sqlite3` 파일에 해당 작업이 저장됨

4. Photo App 초기 작업

- Django App: 각 기능을 수행하는 application

- Photo App 생성: `python manage.py startapp photos`
- Photo App 구성
  - `models.py` : 모델(data field, behavior 포함) 을 정의하는 모듈
  - `views.py` : 특정 URL 접근 시 화면에 표시되는 내용 호출
    - Model과 Template를 연결하는 MVC 패턴의 Controller 역할

```
pystagram/
├── db.sqlite3
├── manage.py
├── photos
│   ├── __init__.py
│   ├── admin.py
│   ├── apps.py
│   ├── migrations
│   │   └── __init__.py
│   ├── models.py
│   ├── tests.py
│   └── views.py
└── pystagram
    ├── __init__.py
    ├── settings.py
    ├── urls.py
    └── wsgi.py
```

5. Photo App 만들기

- Photo model 만들기: `photos/models.py`
  - `ImageField`: FileField 상속받은 클래스
  - `CharField` : 데이터베이스의 Varchar 대응, 250자 보장
  - `TextField` : CharField보다 긴 문자열 다룸``
    - `max_field` 옵션으로 최대 길이 제한
  - `DataTimeField`: 생성날짜 시간 정보 저장
    - `auto_now` 옵션: 객체 매 변경 일시
    - `auto_now_add` 옵션: 객체 최초 생성 일시

```
from django.db import models

# Photo 클래스는 Model 클래스를 상속 받음
class Photo(models.Model):
    image = models.ImageField() # 원본 사진
    filtered_image = models.ImageField()    # 필터 사진
    content = models.TextField(max_length=500)  # 사진 설명문
    created_at = models.DateTimeField(auto_now_add=True)    # 생성일시
```

- 데이터베이스에 반영 (**migration**)

- `setting.py` `INSTALLED_APPS` 항목에 `'photos'` 추가
- 이미지 처리 도구 설치: `pip install pillow`
- migration 스크립트 생성: `python manage.py makemigrations`
- 실제 DB에 반영: `python manage.py migrate`