

Day3 (Django model 그리고 admin)

📎 자료	Django
☰ 구분	Django
⋮ 과목	

Database 란?

- 데이터를 체계적으로 수집하고 저장하는 시스템을 DB (database)라고 한다.
- Database의 기본구조로는 "스키마", "필드", "레코드", "PK" 등이 있다. 용어를 알아야 한다.

스키마(Schema)

- 뼈대(Structure)를 의미한다.
- 데이터베이스에서 자료의 구조, 표현 방법, 자료형 그리고 관계 등을 정의한 "구조"를 의미한다.

테이블(Table)

- 필드와 레코드를 사용해 정리되어 있는 데이터 요소들의 집합
 - 필드는 Column (열)
 - 레코드는 Row(행)을 의미한다.
- 테이블을 만들기 위해서는 데이터 타입이 무엇인지 구조설계가 필요하다.
- 즉, 구조설계(=스키마 Schema) 를 가지고 테이블을 만든다.

필드

	A	B	C	D
1	id	name	age	email
2	PK 1	hong	42	hong@gmail.com
3	2	kim	16	kim@naver.com
4	3	kang	29	kang@hotmail.com
5	4	chol	8	choi@hanmail.com

테이블

레코드

- **필드(field)**

- 컬럼(Column) 부분을 의미하고 [속성] 이라고 부르기도 한다.
- 각 필드에는 int, text 등 데이터의 타입이 지정된다.

- **레코드(record)**

- 행(Row) 부분을 의미하고 [튜플] 이라고 부르기도 한다.
- 데이터들은 테이블의 레코드에 저장된다.
- 위 사진에는 4명의 고객정보가 저장되어 있으므로 레코드는 4개이다.

- **PK(Primary Key)**

- [기본 키] 라고 부르기도 한다.
- 각 레코드의 고유한 값을 가진다.
 - 사람으로 비유하자면 "주민등록번호"와 같다.
 - 즉, 각 데이터들의 고유 번호를 의미한다.
- 다른 레코드와 절대로 중복될 수 없는 unique한 값이다.
- 데이터베이스를 관리하거나 테이블 간의 관계를 설정 할 때 주요하게 활용된다.

- **쿼리(Query)**

- 데이터를 조회하기 위한 명령어를 쿼리 라고 한다.
- 조건에 맞는 데이터를 추출하거나 조작하는 명령어를 의미한다.

- "Query를 날린다." 라는 말은 → "데이터베이스를 조작한다." 라는 말과 같은 의미이다.

Django에서 Model 이란?

데이터를 구조화하고 조작하기 위한 도구이다.

- Django는 Model을 통해 데이터에 접근하고 조작한다.
- 일반적으로 각각의 모델은 하나의 데이터베이스 테이블에 매핑한다.
 - models.py의 클래스 1개 == 데이터베이스 테이블 1개 라는 뜻이다.

[따라서 실습] Model 작성하기

1. crud 라는 이름으로 새 프로젝트를 생성하고 articles 라는 이름으로 새롭게 앱을 생성 및 등록하자.
2. models.py 작성하기
 - 모델클래스를 작성하는 행위는 database 테이블의 스키마를 정의하는 것이다.
 - "모델 클래스" == "테이블 스키마"

```
# articles/models.py

from django.db import models

class Article(models.Model):
    title = models.CharField(max_length=30)
    content = models.TextField()
```


- 우리는 이제 database에 들어갈 테이블의 구조를 파이썬의 class로 정의를 해 준 것이다.

- 우리가 정의할 class는 django에서 제공하는 Model 이라는 클래스를 상속 받아서 Article 이라는 자식 클래스를 정의해 줄 것이다.
 - Django에서 제공하는 Model 클래스를 사용하기 위해서
 - from django.db import models 를 통해서 models 라는 모듈을 import 한 것이다.
 - 따라서 Django는 클래스 상속 기반 형태의 프레임워크 개발이라고 말할 수 있겠다.
- 즉, models 모듈의 Model 클래스를 상속 받아서 Article 이라는 테이블을 만든다.
- title, content 클래스 속성은 DB테이블의 필드를 의미한다.
- models.CharField(max_length=30) : DB 필드의 데이터 타입을 지정한다.


모델에 사용되는 필드의 종류는 여러가지가 있다.

필요 할 때 마다 찾아서 사용한다.

Django Model Field

Django Tutorial Part 3: Using models - Web 개발 학습하기 | MDN 

803

 <https://developer.mozilla.org/ko/docs/Learn/Server-side/Django/Models>

1. CharField(max_length=None)

- 길이의 제한이 있는 문자열을 넣을 때 사용하는 필드이다.
- 데이터베이스와 Django의 유효성 검사 설정이 가능하다.

2. TextField()

- 많은 글자 수를 저장할 때 사용한다.
- max_length 옵션은 폼에서 입력을 받을 때 클라이언트 측에서 유효성 검사를 하도록 해주지만, 실제 모델과 데이터베이스에서 이를 강제하지는 않기 때문에 데이터베이스에 저장될 때는 길이를 제한하지 않는다.

추가로 필드를 더 적어보자.

```
# articles/models.py

from django.db import models

class Article(models.Model):
    title = models.CharField(max_length=30)
    content = models.TextField()
    created_at = models.DateTimeField(auto_now_add=True)
    updated_at = models.DateTimeField(auto_now=True)
```

1. Datetimefield(auto_now_add)

- 게시글의 최초 생성일자를 자동으로 생성한다.(게시판 글 작성 날짜)

2. Datetimefield(auto_now)

- 최초 수정일자를 자동으로 생성한다 (게시판 글 작성 날짜 업데이트됨)

여기에는 추가하지 않았지만 "EmailField(auto_now)" 라는 것도 있다.

- 이메일 형식에 맞게 작성 하는 필드다.

자, 이제 models.py에서 파이썬 클래스를 이용해서 데이터베이스 스키마를 완성했다면 다음 할 일은

스키마 작업을 Database에 반영을 하기 위한 파일을 생성하자.

Migrations 이란?

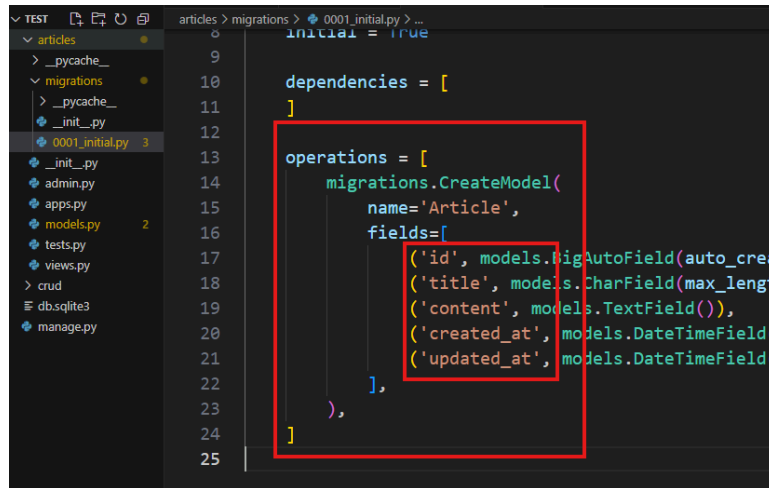
- Django가 모델에 생긴 변화를 실제 DB에 반영하는 방법을 말한다.

- Migrations 주요 명령어

1. `$ python manage.py makemigrations`

- 현재 내 모델 상태를 데이터베이스에 반영할 수 있도록 한다.
- Database에 테이블 생성 및 "수정"시에도 항상 반드시 migration을 해야 한다.
- 명령어 실행 후 migrations 폴더 안에 initial.py 파일이 생성 되었는지 확인 해 보자.

- initial파일에는 기본으로 설정되는 id(PK값)과 model에 입력한 정보가 들어가 있을 것이다.
 - id 는 바로 레코드의 고유의 값(주민등록번호)가 설정이 되는 필드 이다.



자 이렇게 makemigrations를 통해서 Database의 생성 및 변경 사항을 반영하기 위한 마이그레이션 파일을 생성했다. 그 다음은 마이그레이션 파일 (설계도)를 실제로 데이터베이스에 반영시키자.

2. `$ python manage.py migrate`

- makemigrations로 만든 설계도(migration)를 실제로 데이터베이스 (db.sqlite3)에 반영하는 과정이다.
- 모델의 변경사항과 데이터 베이스를 동기화 시키는 과정이다.

[참고] → 아래것들은 몰라도 문제될것 없음... 말 그대로 참고임 (잘 사용하지 않음 ㅎㅎ)

3. sqlmigrate

- `$ python manage.py sqlmigrate articles 0001`
- 해당 migrations 파일이 SQL문으로 어떻게 해석될 지 미리 확인 가능

3. showmigrations

- `$ python manage.py showmigrations`
- migrations 파일들이 migrate 됐는지 여부 확인하는 용도

- [X]표시가 있으면 migrate완료 되었다는 의미

앞으로 models.py를 생성 또는 수정 할때마다 아래의 과정을 반드시 해야 한다.

migration 3단계

1. models.py에 변경사항 발생!
2. migration 생성 (makemigrations) ! 그리고 난 후에 (`$ python manage.py makemigrations`)
3. DB 반영 (모델과 DB 동기화) (migrate) (`$ python manage.py migrate`)

관리자 페이지 설정

Django의 강력한 기능 중 하나는 관리자 페이지의 인터페이스 이다.

- 관리자 페이지란?
 - 사용자가 아닌 서버의 관리자가 활용하기 위한 페이지다.
 - 관리자 페이지에서 Database 테이블을 직접 조작하기 위해서 모델 class를 admin.py에 등록을 먼저 해야 한다.

실습 해보자.

admin 계정 생성 (관리자 계정 생성하자)

- `python manage.py createsuperuser`
- username과 password를 입력해 관리자 계정 생성
 - email은 선택사항이다. 그냥 엔터 눌러도 된다.
 - password는 입력을 해도 화면에 보이지 않는다.
 - 화면에 보이지 않더라도 입력되고 있다고 생각하자.

```
SSAFY@DESKTOP-DOGVPU8 MINGW64 ~/Desktop/crud
$ python manage.py createsuperuser
Username (leave blank to use 'ssafy'): admin
Email address: admin@admin.kr
Password:
Password (again):
This password is too common.
Bypass password validation and create user anyway? [y/N]: y
Superuser created successfully.
```

admin site 로그인

- 서버를 켜서 브라우저에 `http://127.0.0.1:8000/admin/` 으로 접속하자. 그리고 로그인 해보자.
- 현재는 관리자 계정만 만든 상태다. 그래서 아직 관리자 화면에서 Articles 모델 클래스를 볼 수는 없다.

admin에 모델 클래스 등록

- 모델의 레코드(Record)를 보기 위해서 `admin.py`에 등록이 필요하다.

```
# articles/admin.py

from django.contrib import admin
from .models import Article

admin.site.register(Article)
```

위 코드 설명은 다음과 같다.

- `models.py` 파일에서 Article 클래스를 import 해와서
`admin.site.register(Article)` : ["어드민 사이트"에 "등록"한다. "Article을"] 이라고 기억을 하자.

다시 브라우저로 돌아가서 새로고침 해보자.

그리고 Articles 데이터베이스에 레코드를 등록해보자. Add를 통해 입력이 가능하다.



관리자페이지에서 이것저것 눌러보자 !!! <끝>