Model

Model

1. Model 속성

Model 속성

모델 클래스

모델 속성

Meta 내부 클래스

Meta 내부 클래스 속성

모델 메소드

Field Type

- 모델 클래스의 속성들은 테이블의 컬럼으로 1:1 매핑된다.
- 테이블의 컬럼 타입을 지정한다.
- 폼으로 렌더링되는 경우, HTML 위젯을 지정한다.
- 필드 또는 폼에 대한 유효성 검사 시 최소기준이 된다.
- https://docs.djangoproject.com/en/2.2/ref/models/fields/

Field Type

- **참/거짓** BooleanField, NullBooleanField
- **숫자** DecimalField, FloatField, IntegerField, BigIntegerField, PositiveIntegerField, SmallIntegerField
- **문자열** CharField, SlugField, TextField
- **날짜/시간** DateField, DateTimeField, TimeField, DurationField
- **파일** BinaryField, FileField, FilePathField, ImageField
- **기타** EmailField, GenericIPAddressField, URLField, UUIDField

Automatic Primary Key Field

- id = models.AutoField(primary_key=True)
- 자동으로 증가하는 IntegerField이다.
- Django는 기본 키 필드를 자동으로 모델에 추가한다.
- primary_key=True를 지정하면 자동으로 id 열을 추가하지 않는다.
- 모델에는 하나의 필드에 primary_key=True가 있어야한다.

Relationship Type

- 장고에서는 한쪽 클래스에서만 관계를 정의하면 상대편 정의는 자동으로 정의한다.

- ForeignKey (1:N 관계)

- ManyToManyField (N:N 관계)

- OneToOneField (1:1 관계)

ForeignKey

- 테이블 간 1:N 관계를 정의하기 위해 사용한다.
- N 모델에 명시적으로 정의한다.
- ForeignKey(to, on_delete)
  - to: 모델이 관련된 클래스
  - on_delete: 외래 키에 의해 참조된 객체를 삭제하면 지정된 SQL 제약의 동작을 따른다.
    - CASCADE, PROTECT, SET_NULL, SET_DEFAULT, SET(), DO_NOTHING

Field Option

- null, blank, choices
- db_column, db_index, db_tablespace
- default
- editable
- validators
- error_messages, help_text, verbose_name
- primary_key, unique, unique_for_date, unique_for_month, unique_for_year

Custom Model Field

2. Meta 내부 클래스

Meta 내부 클래스

- 모델에 대한 메타데이터를 정의한다.
- 필드 이외 항목은 Meta 내부 클래스의 속성으로 정의한다.

Class Meta:

ordering = [‘-pub_date’, ‘author’]

db_table = ‘tb_post’

verbose_name = “my favorite post”

- https://docs.djangoproject.com/en/2.2/ref/models/options/

3. Model Method

Model Method

- **클래스 메소드**
  - 테이블 레벨에서 동작하는 메소드. 장고에서는 사용하지 않는다.
  - Manager 클래스를 정의하고 Manager 클래스의 메소드를 통해 테이블에 대한 CRUD 동작을 수행한다.
- **객체 메소드**
  - 레코드 레벨에서 동작하는 메소드
  - __str__(), get_absolute_url()
  - https://docs.djangoproject.com/en/2.2/ref/models/instances/#model-instance-methods

__str__()

- 객체의 문자열 표현을 리턴한다.
- 객체를 읽을 수 있는 문자열로 표현하고자 할 때 사용한다.
- 장고 셸이나 Admin 사이트에서 객체의 문자열 표현을 주로 사용한다.
- __str__() 메소드를 오버라이딩 정의하지 않으면 장고의 디폴트 메소드에 의해 객체의 문자열이 표현된다. (예: Bookmark object)

4. Manager 클래스

Manager 속성

- 모델은 반드시 Manager 속성을 가진다.
- 모델 클래스에서 Manager 속성을 여러 개 정의할 수 있다.
- 첫 번째로 정의된 Manager 속성을 디폴트 Manager 라고 한다.
- 모델을 정의할 때 명시적으로 지정하지 않으면 디폴트 이름은 objects가 된다.
- 모델 클래스를 통해서만 엑세스할 수 있고 모델 객체를 통해서는 액세스할 수 없다.
- models.Manager 타입으로 정의된다.

Manager Class

- 데이터베이스에 대한 처리, 즉 데이터베이스에 쿼리를 보내고 그 응답을 받는 역할

- 테이블 레벨에서의 CRUD 동작은 Manager 클래스의 메소드를 통해 이뤄진다.

- Album.objects.all()

Manager Method

- QuerySet 클래스의 메소드와 Manager 클래스의 메소드는 동일하다.

- QuerySet 메소드는 모두 Manager 메소드로 사용할 수 있다.

- all(), filter(), exclude(), get(), count() 등

Related Manager

- 매니저 중 모델 간 관계에 대한 기능 및 데이터베이스 쿼리를 담당하는 클래스
- 관계 매니저 클래스는 객체들의 집합을 다루기 위한 클래스이다.
- 1:1 관계에서는 상대 객체가 하나이기 때문에 관계 매니저를 사용하지 않는다.
- 관계 매니저 객체는 관계 매니저 클래스에서 제공하는 메소드를 사용할 수 있다.

1:N 관계

- 1에서 N 방향으로 엑세스하는 경우에 관계 매니저를 사용한다.
    - user1.album_set # album_set은 관계 매니저 클래스의 객체
- N에서 1방향으로 엑세스하는 경우는 ForeignKey로 정의한 필드명을 사용한다.
    - album1.owner # owner는 ForeignKey 타입의 필드명

N:N 관계

- 양쪽 방향으로 모두 관계 매니저 클래스를 사용한다.
- publication_set은 관계 매니저 클래스의 객체
    - album1.publication_set
- albums는 ManyToManyField 타입의 필드명이면서 관계 매니저 클래스의 객체
    - publication1.albums

Related Manager Method

- https://docs.djangoproject.com/en/2.2/ref/models/relations/