

Ask Company



리액트와 함께 장고 시작하기 / 장고 Models
장고 모델 필드

여러분의 파이썬/장고 페이스메이커가 되겠습니다.

기본 지원되는 모델필드 타입 (1)

Primary Key: **AutoField**, **BigAutoField**

문자열: **CharField**, **TextField**, **SlugField**

날짜/시간: **DateField**, **TimeField**, **DateTimeField**, **DurationField**

참/거짓: **BooleanField**, **NullBooleanField**

숫자: **IntegerField**, **SmallIntegerField**, **PositiveIntegerField**,
PositiveSmallIntegerField, **BigIntegerField**, **DecimalField**, **FloatField**

파일: **BinaryField**, **FileField**, **ImageField**, **FilePathField**

<https://docs.djangoproject.com/en/3.0/ref/models/fields/#field-types>

기본 지원되는 모델필드 타입 (2)

이메일: EmailField

URL: URLField

UUID: UUIDField

아이피: GenericIPAddressField

Relationship Types

ForeignKey

ManyToManyField

OneToOneField

그리고, 다양한 커스텀 필드들

django-model-utils : <https://django-model-utils.readthedocs.io/en/latest/>

모델필드들은 DB 필드타입을 반영

DB에서 지원하는 필드들을 반영

Varchar 필드타입 → CharField, SlugField, URLField, EmailField 등

파이썬 데이터타입과 데이터베이스 데이터타입을 매핑

AutoField → int

BinaryField → bytes

BooleanField → bool

CharField/SlugField/URLField/EmailField → str → 디폴트 적용된 유효성 검사 등의 차이

같은 모델필드라 할지라도, DB에 따라 다른 타입이 될 수도 있습니다.

DB에 따라 지원하는 기능이 모두 달라요.

자주 쓰는 필드 공통 옵션

blank : 장고 단에서 validation시에 empty 허용 여부 (디폴트: False)

null (DB 옵션) : null 허용 여부 (디폴트: False)

db_index (DB 옵션) : 인덱스 필드 여부 (디폴트: False)

default : 디폴트 값 지정, 혹은 값을 리턴해줄 함수 지정

사용자에게 디폴트값을 제공코자 할 때

unique (DB 옵션) : 현재 테이블 내에서 유일성 여부 (디폴트: False)

choices : select 박스 소스로 사용

validators : validators를 수행할 함수를 다수 지정

모델 필드에 따라 고유한 validators들이 등록 (ex- 이메일만 받기)

verbose_name : 필드 레이블, 미지정시 필드명이 사용

help_text : 필드 입력 도움말

```
from django.conf import settings
from django.db import models

class Profile(models.Model):
    user = models.OneToOneField(settings.AUTH_USER_MODEL, on_delete=models.CASCADE)
    blog_url = models.URLField(blank=True)

class Post(models.Model):
    author = models.ForeignKey(settings.AUTH_USER_MODEL, on_delete=models.CASCADE)
    title = models.CharField(max_length=100, db_index=True)
    slug = models.SlugField(allow_unicode=True, db_index=True) # ModelAdmin.prepopulated_fields 편리
    desc = models.TextField(blank=True)
    image = models.ImageField(blank=True) # Pillow 설치가 필요
    comment_count = models.PositiveIntegerField(default=0)
    tag_set = models.ManyToManyField('Tag', blank=True)
    is_publish = models.BooleanField(default=False)
    created_at = models.DateTimeField(auto_now_add=True)
    updated_at = models.DateTimeField(auto_now=True)

class Comment(models.Model):
    author = models.ForeignKey(settings.AUTH_USER_MODEL, on_delete=models.CASCADE)
    post = models.ForeignKey(Post, on_delete=models.CASCADE)
    message = models.TextField()
    created_at = models.DateTimeField(auto_now_add=True)
    updated_at = models.DateTimeField(auto_now=True)

class Tag(models.Model):
    name = models.CharField(max_length=50, unique=True)
```

강력히 권합니다.

설계한 데이터베이스 구조에 따라, 최대한 필드타입을 타이트하게 지정해주는 것이, **입력값 오류**를 막을 수 있음.

blank/null 지정은 최소화해주세요. → manage.py inspect 명령을 통해 생성된 모델 코드는 초안입니다.

validators 들이 다양하게/타이트하게 지정됩니다.

필요하다면, validators들을 추가로 타이트하게 지정해주세요.

프론트엔드에서의 유효성 검사는 사용자 편의를 위해서 수행하며, **백엔드에서의 유효성 검사는 필수**입니다.

직접 유효성 로직을 만들지 마세요. 이미 잘 구성된 Features들을 가져다 쓰세요. 장고의 Form/Model을 통해 지원되며, django-rest-framework의 Serializer를 통해서도 지원됩니다.

ORM은 SQL 쿼리를 만들어주는 역할일 뿐, 보다 성능높은 애플리케이션을 위해서는, 사용하실려는 **데이터베이스**에 대한 깊은 이해가 필요합니다.

Life is short.
You need Python and Django.

I will be your pacemaker.

