

멋쟁이 사자처럼 11기 BE
3차 세션 강의

3차 세션

Django MTV 실습

Learn About Django

멋쟁이 사자처럼

11기 운영진

Database

HACK YOUR LIFE!

Data란?

Data

Data : 현실 세계에서 사건이나 사물의 특징을 관찰하거나
측정하여 기술하는 가공되지 않은 사실이나 값

- 정보 : 의미 있고 쓸모 있는 내용으로 가공하여 체계적으로 조직한 데이터를 의미
- 우리는 실세계의 가공되지 않은 무의미한 자료(Data)들을 통해서 정보를 얻어낸다.
- 즉, 무의미한 데이터들을 처리(가공)하여 유의미한 정보를 만들어 내는 것
- 그리고 그 데이터들을 정보로 처리하기 이전에 어떻게 체계적으로 관리할 수 있을까?

Database란?

Data + Base

어떤 특정한 조직에서 여러 명의 사용자 또는 응용 시스템들이 공유하고 동시에 접근하여 사용할 수 있도록 구조적으로 통합하여 저장한 **운영 데이터의 집합**

- **실시간 처리** : query(질의)에 대하여 실시간 처리로 응답해야 한다.
- **변화** : DB는 삽입, 삭제, 갱신에 의해 계속적으로 변하고 정확한 데이터를 유지할 수 있다.
- **공유성** : 자신이 원하는 데이터를 동시 공유할 수 있다.
- **내용에 따른 참조** : 데이터의 위치나 주소가 아닌, 원하는 데이터의 내용에 따라 참조가 가능해야 한다.

DBMS란?

Database Management System

데이터베이스를 관리하고 운영하는 소프트웨어

- 데이터베이스에 접근하여 데이터를 조작할 수 있도록 도와준다.
- 즉, 사용자(개발자)들은 DB에 직접 접근하여 데이터를 다루는 것이 아니라, DBMS의 도움을 받아 데이터를 다루고 변화시키는 것이다.
- Ex) SQL, MySQL, MariaDB, SQLite, Oracle



관계형 데이터베이스(RDBMS)란?

Relational Database Management System

- 테이블(table)이라는 최소 단위로 구성되며, 이 테이블은 하나 이상의 열(column)과 행(row)으로 이루어져 있다.

학번	이름	연락처
1771154	양종욱	010-4774-0000
1971141	이현승	010-3546-0000

ORM

Object Relational Mapping

객체 지향 프로그래밍과 RDBMS의 조합

- 객체와 관계형 데이터베이스의 데이터를 자동으로 매핑해주는 것
- 객체 지향 프로그래밍의 Class와, 관계형 데이터베이스(RDBMS)의 Table 사이의 모델(Model - DB 테이블과 컬럼)을 자동으로 생성.
- 즉 데이터베이스의 쿼리와 관련된 SQL을 자동으로 생성

S.M.U.V.T

HACK YOUR LIFE!

{

S.M.U.V.T

}

S.M.U.V.T란?

Settings.py Models.py Urls.py Views.py Templates

Django 개발 순서

- Django Framework를 효율적으로 사용하기 위한 개발 순서.

{

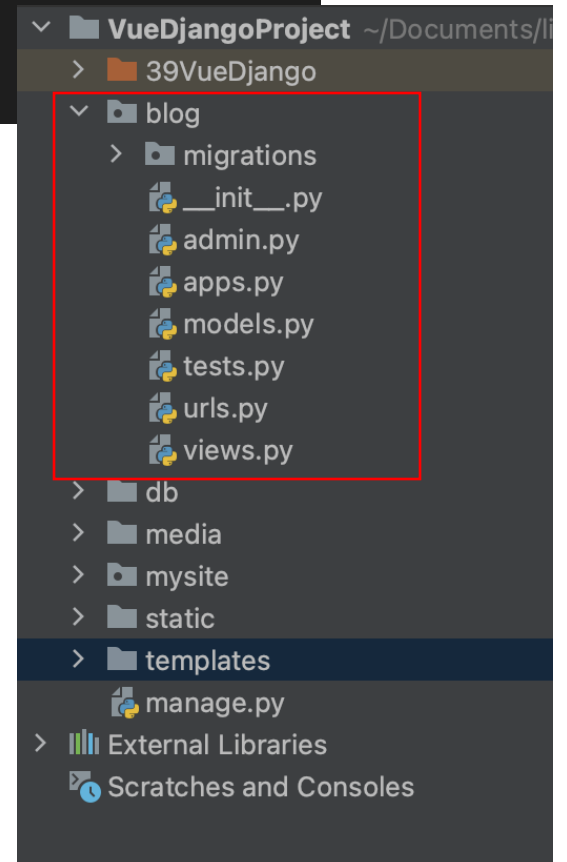
Setting.py

}

Blog 앱 생성

```
PS C:\Users\MYCOM\Documents\GitHub\DjangoStudy\VueDjangoProject> django-admin startapp blog
```

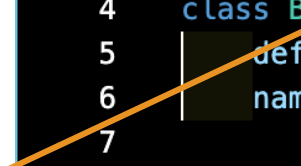
Python3(python) manage.py startapp blog 와도 동일



Settings.p

```
y INSTALLED_APPS = [  
    'django.contrib.admin',  
    'django.contrib.auth',  
    'django.contrib.contenttypes',  
    'django.contrib.sessions',  
    'django.contrib.messages',  
    'django.contrib.staticfiles',  
    'blogApp.apps.BlogappConfig',  
]
```

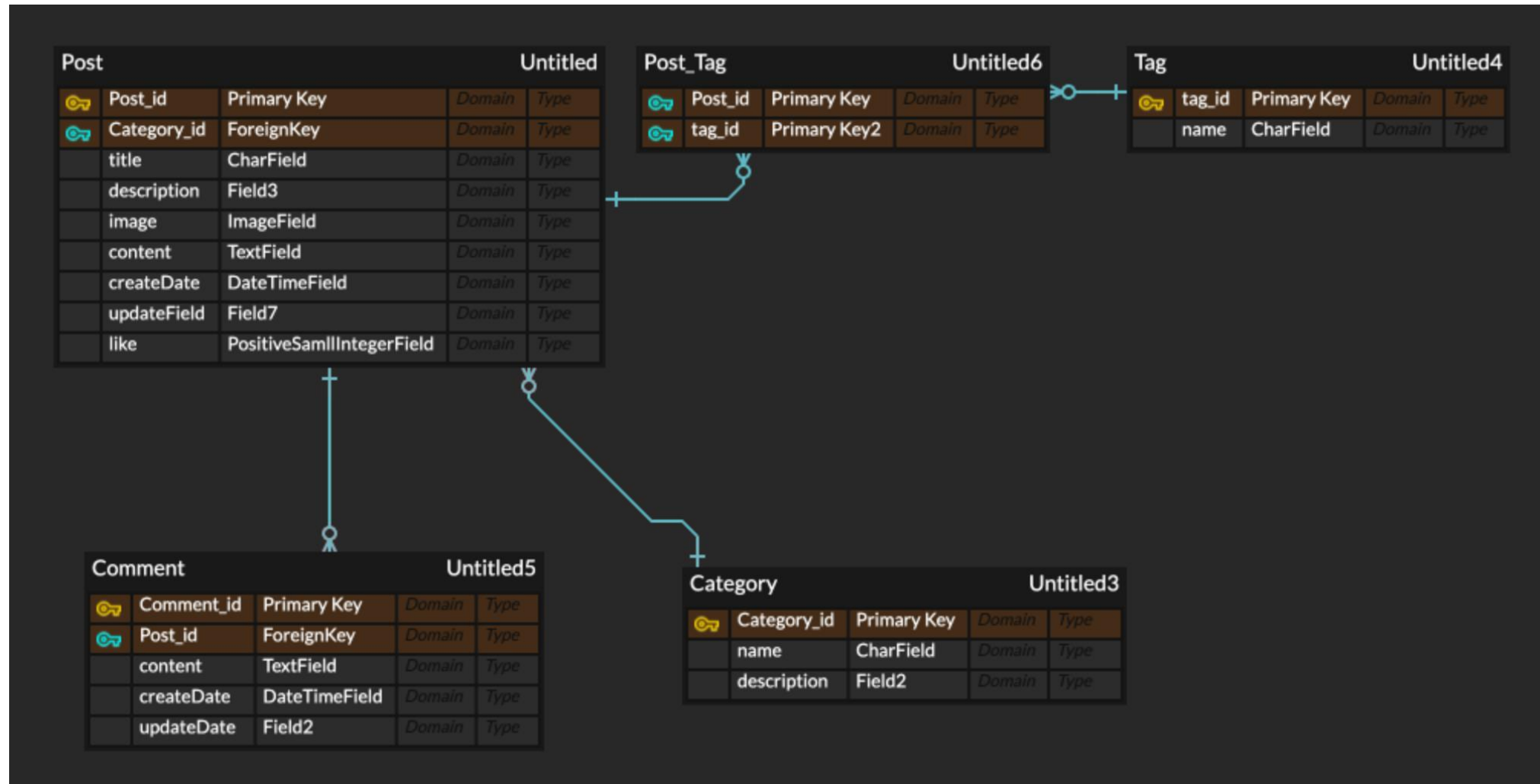
```
apps.py ×  
blogProject > blogApp > apps.py > ...  
1 from django.apps import AppConfig  
2  
3  
4 class BlogappConfig(AppConfig):  
5     default_auto_field = 'django.db.models.BigAutoField'  
6     name = 'blogApp'  
7
```



{

Models.py

}



```
class Post(models.Model):
    category = models.ForeignKey('Category', on_delete=models.SET_NULL, blank=True, null=True)
    tags = models.ManyToManyField('Tag') # 참조하는 모델 이름
    title = models.CharField('TITLE', max_length=50) # 해당 칼럼에 별칭
    description = models.CharField('DESCRIPTION', max_length=100, blank=True, help_text='simple one-line text')
    image = models.ImageField('IMAGE', upload_to='blog/%Y/%m/', blank=True, null=True)
    content = models.TextField('CONTENT')
    createDate = models.DateTimeField('CREATE DATE', auto_now_add=True)
    updateDate = models.DateTimeField('UPDATE DATE', auto_now_add=True)
    like = models.PositiveSmallIntegerField('LIKE', default=0)
```

```
category = models.ForeignKey('Category', on_delete=models.SET_NULL, blank=True, null=True)
```

```
from django.db import models

class Car(models.Model):
    manufacturer = models.ForeignKey(
        'Manufacturer',          A many-to-one relationship
        on_delete=models.CASCADE,
    )    ForeignKey를 포함하는 모델 인스턴스(row)도 같이 삭제한다.
    # ...

class Manufacturer(models.Model):
    # ...
    pass
```

```
models.ForeignKey('self', on_delete=models.CASCADE).
```

```
tags = models.ManyToManyField('Tag') # 참조하는 모델 이름
```

```
from django.db import models
```

```
class Person(models.Model):  
    friends = models.ManyToManyField("self")
```

```
models.ManyToManyField('self')
```

```
title = models.CharField('TITLE', max_length=50) #해당 칼럼에 별칭  
description = models.CharField('DESCRIPTION', max_length=100, blank=True, help_text='simple one-line text')
```

class CharField(*max_length=None, **options*)

A string field, for small- to large-sized strings.

CharField.max_length

The maximum length (in characters) of the field
Required.

CharField.db_collation

The database collation name of the field.
Optional.

CharField.blank

공백 허용
Optional.

CharField.help_text

placeholder
Optional.

```
content = models.TextField('CONTENT')
```

```
class TextField(**options)
```

A large text field.

```
TextField.db_collation
```

The database collation name of the field.
Optional.

```
image = models.ImageField('IMAGE', upload_to='blog/%Y/%m/', blank=True, null=True)
```

```
class ImageField(upload_to=None, height_field=None, width_field=None, max_length=100, **options)
```

`ImageField.width_field`

모델 인스턴스가 저장될 때마다 **이미지 너비**가 자동으로 채워지는 모델 필드의 이름

`ImageField.height_field`

모델 인스턴스가 저장될 때마다 **이미지 높이**가 자동으로 채워지는 모델 필드의 이름

`FileField`로부터 모든 속성과 메서드를 상속받지만, 업로드된 객체가 유효한 이미지인지도 검증한다.

Pillow 라이브러리를 요구한다.

Pip install Pillow

```
class FileField(upload_to='', storage=None, max_length=100, **options)
```

```
class MyModel(models.Model):  
    # file will be uploaded to MEDIA_ROOT/uploads  
    upload = models.FileField(upload_to='uploads/')  
    # or...  
    # file will be saved to MEDIA_ROOT/uploads/2015/01/30  
    upload = models.FileField(upload_to='uploads/%Y/%m/%d/')
```



```
createDate = models.DateTimeField('CREATE DATE', auto_now_add=True)
updateDate = models.DateTimeField('UPDATE DATE', auto_now_add=True)
```

```
class DateTimeField( auto_now = False, auto_now_add = False, ** options)
```

DateField.auto_now_add

개체가 처음 생성될 때 필드를 지금으로 자동 설정합니다.
이 필드를 수정할 수 있게 하려면 다음을 대신 설정하십시오 auto_now_add=True.

Settings.py

```
TIME_ZONE = 'Asia/Seoul'
```

여기서 부터 `super()` -> `self()` 로 바뀌야됨.

```
class Post(models.Model):
    category = models.ForeignKey('Category', on_delete=models.SET_NULL, blank=True, null=True)
    tags = models.ManyToManyField('Tag') # 참조하는 모델 이름
    title = models.CharField('TITLE', max_length=50) #해당 칼럼에 별칭
    description = models.CharField('DESCRIPTION', max_length=100, blank=True, help_text='simple one-line text')
    image = models.ImageField('IMAGE', upload_to='blog/%Y/%m/', blank=True, null=True)
    content = models.TextField('CONTENT')
    createDate = models.DateTimeField('CREATE DATE', auto_now_add=True)
    updateDate = models.DateTimeField('UPDATE DATE', auto_now_add=True)
    like = models.PositiveSmallIntegerField('LIKE', default=0)
```

```
def __str__(self):
    return super().title
```

`__str__()` 메서드는 모델 클래스의 객체의 문자열 표현을 리턴한다.

```
from django.db import models

class Question(models.Model):
    question_text = models.CharField(max_length=200)
    pub_date = models.DateTimeField('date published')

    def __str__(self):
        return self.question_text
```

__str__ 는 해당 클래스로 만들어진 인스턴트를 자체를 출력할 때, 문자열로 설명해주기 위한 메소드다.
Django의 models.py에서는 class가 admin 페이지에서 어떻게 출력되는지 정의해주는 역할이다.

Ordering

```
from django.db import models

class Post(models.Model):
    no = models.IntegerField()
    name = models.CharField(max_length=128)

    class Meta:
        # 오름차순
        # ordering = ['no']

        # 내림차순
        ordering = [ '-no' ]
```

ordering을 써, 그 값을 리스트로 초기화합니다.
order는 필드 이름과 하이픈(-)의 조합으로 표시됩니다.

Db_table

```
from django.db import models

class Post(models.Model):
    no = models.IntegerField()
    name = models.CharField( max_length = 128 )

    class Meta :
        # 테이블 이름을 mypost로 변경
        db_table = 'mypost'
```

db_table는 모델 데이터베이스의 테이블 이름을 변경합니다

verbose_name, verbose_name_plural

```
from django.db import models

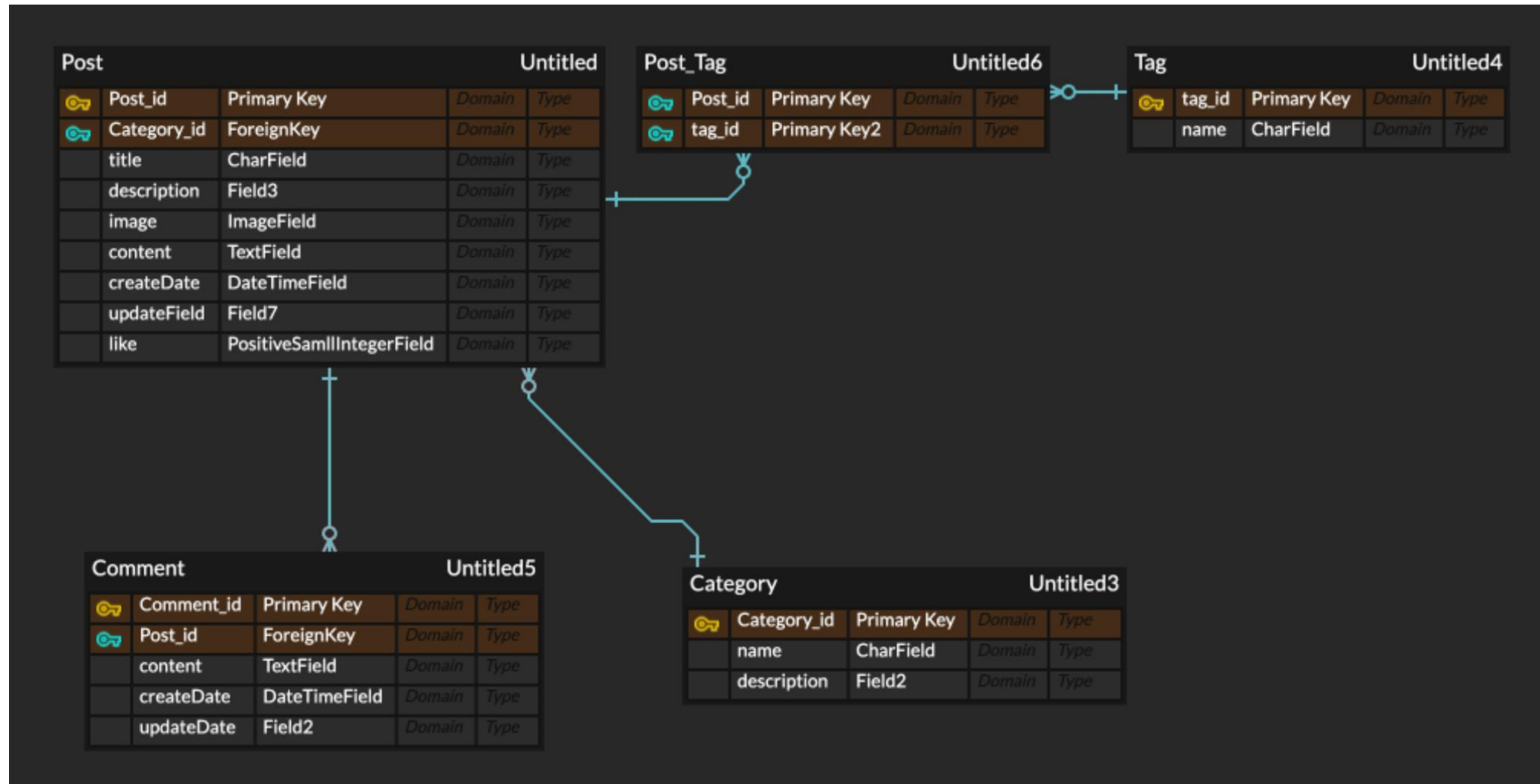
class Post(models.Model):
    no = models.IntegerField()
    name = models.CharField(max_length=128)

    class Meta:
        verbose_name = '포스트' # 단수형
        verbose_name_plural = '포스트 그룹' # 복수형
```

`verbose_name`에는
모델의 단수형의 이름을 지정합니다.

`verbose_name_plural`에는
모델의 복수형의 명칭을 지정합니다.

Django는 이러한 속성을 관리 사이트에서 모델을 볼 때 참조합니다.



```
class Category(models.Model):  
    name = models.CharField(max_length=50, unique=True)  
    description = models.CharField('DESCRIPTION', max_length=100, blank=True, help_text='simple one-line text.')
```



```
    def __str__(self):  
        return super().name
```

```
class Tag(models.Model):  
    name = models.CharField(max_length=50)  
  
    def __str__(self):  
        return super().name
```



```
class Comment(models.Model):
    post = models.ForeignKey(Post, on_delete=models.CASCADE, blank=True, null=True)
    content = models.TextField('CONTENT')
    createDate = models.DateTimeField('CREATE DATE', auto_now_add=True)
    updateDate = models.DateTimeField('UPDATE DATE', auto_now_add=True)

    def short_content(self):
        return self.content[:10] # content 10글자 가져옴

    def __str__(self):
        return super().short_content
```

{

Admin.py

}

```
from django.contrib import admin

from blogApp.models import Post, Category, Tag, Comment

@admin.register(Post)
class PostAdmin(admin.ModelAdmin):
    list_display = ('id', 'category', 'tagList', 'title', 'description', 'image', 'createDate', 'updateDate', 'like')

    def tagList(self, obj):
        return ','.join([t.name for t in obj.tags.all()])

    def get_queryset(self, request):
        return super().get_queryset(request).prefetch_related('tags')

@admin.register(Category)
class CategoryAdmin(admin.ModelAdmin):
    list_display = ('id', 'name', 'description')

@admin.register(Tag)
class TagAdmin(admin.ModelAdmin):
    list_display = ('id', 'name')

@admin.register(Comment)
class CommentAdmin(admin.ModelAdmin):
    list_display = ('id', 'post', 'short_content', 'createDate', 'updateDate')
```

```
@admin.register(Post)
class PostAdmin(admin.ModelAdmin):
    list_display = ('id', 'category', 'tagList', 'title', 'description', 'image', 'createDate', 'updateDate', 'like')

    def tagList(self, obj):
        return ','.join([t.name for t in obj.tags.all()])

    def get_queryset(self, request):
        return super().get_queryset(request).prefetch_related('tags')
```

기본적으로 하나의 오브젝트에 대해 하나의 값만을 표시한다.
여러 값을 표시하기 위해서 `list_display` 로 추가해 줄 수 있다.

```
@admin.register(Post)
class MyAdmin(admin.ModelAdmin):
    list_display = ['id', 'title', 'short_content', 'is_published' ]
    list_display_links = ['title']

    def short_content(self, post):
        return post.content[:10]
```

admin 사이트에서 세부 항목으로 들어가는 link 를 어디에 걸어줄 지 선택할 수 있습니다.

```
@admin.register(Post)
class MyAdmin(admin.ModelAdmin):
    list_display = ['id', 'title', 'short_content', 'is_published' ]
    list_display_links = ['title']
    list_filter = ['is_published']

    def short_content(self, post):
        return post.content[:10]
```

`list_filter`를 통해서는 오른쪽 UI를 통해서 원하는 값들만 filtering 해서 볼 수 있습니다.

{

Urls.py

}

mysite의 urls.py

'blog/'로 시작하는 모든 URL을 **blog.urls** 모듈에서 처리하도록 지정

```
from django.conf import settings
from django.conf.urls.static import static
from django.contrib import admin
from django.urls import path, include

from mysite.views import HomeView

urlpatterns = [
    path('admin/', admin.site.urls),
    path('', HomeView.as_view(), name='home'),
    path('blog/', include('blog.urls')),
]
```


blog 폴더에 urls.py 파일 생성

url = post/1 이런식으로 접근하면 views.py 파일의 PostDV 클래스 참조

```
1  from django.urls import path
2
3  from blog import views
4
5  app_name = 'blog'
6  urlpatterns = [
7      #/blog/post/98/
8      path('post/<int:pk>', views.PostDV.as_view(), name='post_detail'),
9  ]
```

{

Views.py

}

blog의 views.py

```
1  from django.views.generic import DetailView
2
3  from blog.models import Post
4
5
6  class PostDV(DetailView):
7      model = Post
8      template_name = 'blog/post_detail.html'
```

{

Templates

}

templates 폴더에 blog 파일 생성 후 post_detail.html 생성
blog/post/1/ url 이동 -> 오류 -> admin에서 추가하면

```
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <meta http-equiv="X-UA-Compatible" content="IE=edge">
6      <meta name="viewport" content="width=device-width, initial-scale=1.0">
7      <title>post_detail.html</title>
8  </head>
9  <body>
10  This is post_detail.html
11  </body>
12  </html>
```



1. Python3(python) manage.py makemigrations (app 이름)

애플리케이션의 모델 변경 사항을 마이그레이션 파일로 생성

2. Pip install pillow

Db에 사진 저장할 때 사용

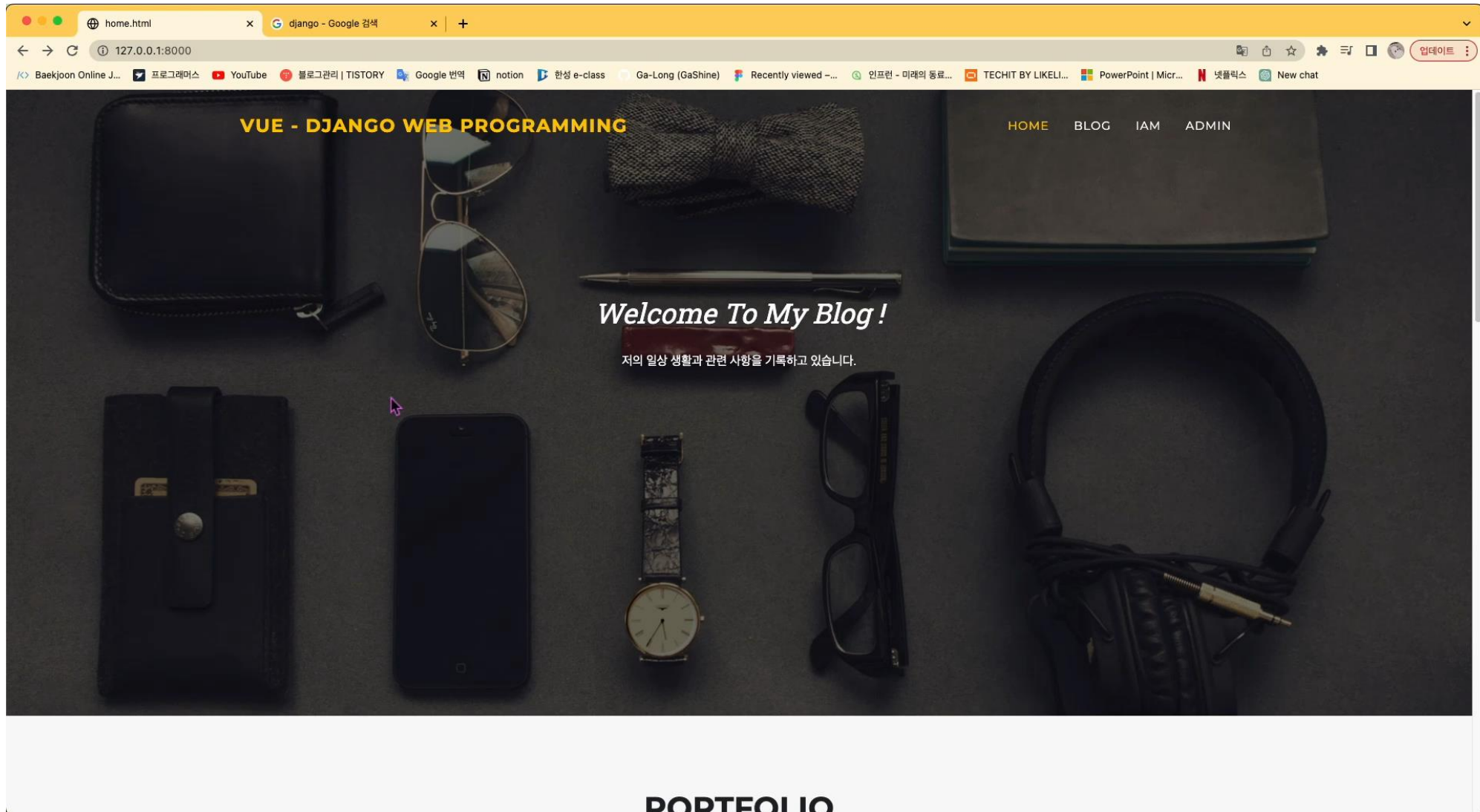
3. Python3(python) manage.py migrate (app 이름)

마이그레이션 파일을 데이터베이스에 적용하는 작업을 수행

4. Python3(python) manage.py runserver

가상환경 실행

admin 추가





게시물 사진 보기

mysite의 settings.py

```
from django.conf.urls.static import static
```

```
125     DEFAULT_AUTO_FIELD = 'django.db.models.BigAutoField'
126     STATICFILES_DIRS = (BASE_DIR / 'static',)
127     MEDIA_URL = '/media/'
128     MEDIA_ROOT = BASE_DIR / 'media'
129     |
```

mysite의 urls.py

```
#MEDIA_URL로 들어오면 MEDIA_ROOT에서 정의한 찾아서 사용
urlpatterns += static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT)
```

Django administration

WELCOME, GAHYEON. VIEW SITE / CHANGE PASSWORD / LOG OUT

Home > Blog > Posts

Start typing to filter...

AUTHENTICATION AND AUTHORIZATION

Groups + Add

Users + Add

Blog

Categories

- > .idea
- > 39VueDjango
- > blog
- > db

Tags

- > media / blog / 2023 / 05
 - 깨꼭깨.png
 - 깨꼭안해.png
 - 깨꼭저웃.png
- > mysite
- > static
- > templates
 - > blog
 - <> post_detail.html
 - <> sb_org.html
 - <> base.html
 - <> home.html
 - <> manage.py


Select post to change

Action: Go 0 of 4 selected

ADD POST +

THE POST "Announcement of 5.x Technical Board Election Regis" was added successfully.

TAG LIST	TITLE	DESCRIPTION	IMAGE	CREATE DT	UPDATE DT	LIKE
web	Announcement of 5.x Technical Board Election Regis	Posted by Katie McLaughlin on 4월 15, 2023	blog/2023/05/깨꼭저웃.png	May 4, 2023, 11:25 p.m.	May 4, 2023, 11:25 p.m.	0
framwork	Django security releases issued: 4.2.1, 4.1.9, and	Posted by Mariusz Felisiak on 5월 3, 2023	blog/2023/05/깨꼭깨.png	May 4, 2023, 11:22 p.m.	May 4, 2023, 11:22 p.m.	0
technical board	Django 5.x Steering Council Candidate Registration	Posted by Katie McLaughlin on 4월 22, 2023	blog/2023/05/깨꼭안해.png	May 4, 2023, 1:35 a.m.	May 4, 2023, 1:35 a.m.	0
django,technical board	Django 흠어보기	django 모델 설계		May 4, 2023, 1:27 a.m.	May 4, 2023, 1:27 a.m.	0



테킷 [파이썬 Django 실습] chapter 1-4 들어오고 실습해오기!



수강중 23.04.28. ~ 23.12.31.

[대학 11기] 파이썬 Django 실습

2% (2개 / 83개)

총 10시간 정도인데 하루에 2시간씩 5일 ㅎㅎ 투자하기!

챕터 1. Prologue: 백엔드 개발 기초



13스텝 · 3시간 36분

챕터 2. 데이터베이스와 ORM 완벽 이해하기



7스텝 · 1시간 54분

챕터 3. QuerySet API와 Admin 개발하기



7스텝 · 2시간 24분

챕터 4. Template과 View 정복하기



9스텝 · 2시간 23분