

Django란? 7번째 세션

NEXT X LIKELION

박지환

| 장고?



웹 프로그램을 쉽고 빠르게 만들 수 있도록 도움을 주는 파이썬 기반 웹 프레임워크

- 로그인/로그아웃
- 데이터베이스
- 보안

이미 만들어져 있기 때문에 잘 사용하기만 하면 된다...!

| 장고?

Django(장고)

The web framework for perfectionists with deadlines. (마감에 쫓기는 완벽주의자를 위한 웹 프레임워크)

The Django logo, featuring the word "django" in a lowercase, white, sans-serif font on a dark green rectangular background.

The web framework for
perfectionists with deadlines.

| 장고?

Django(장고)의 장점

1. 빨리 만들 수 있다

2. 안전하다

- xss, csrf 등 다양한 해킹 공격에 대한 준비가 잘 되어 있다.

3. 기능이 많다

- 로그인, 관리자 기능 등 이미 만들어진 기능이 많다.

Django 설치

```
$ mkdir session7
```

```
$ cd session7
```

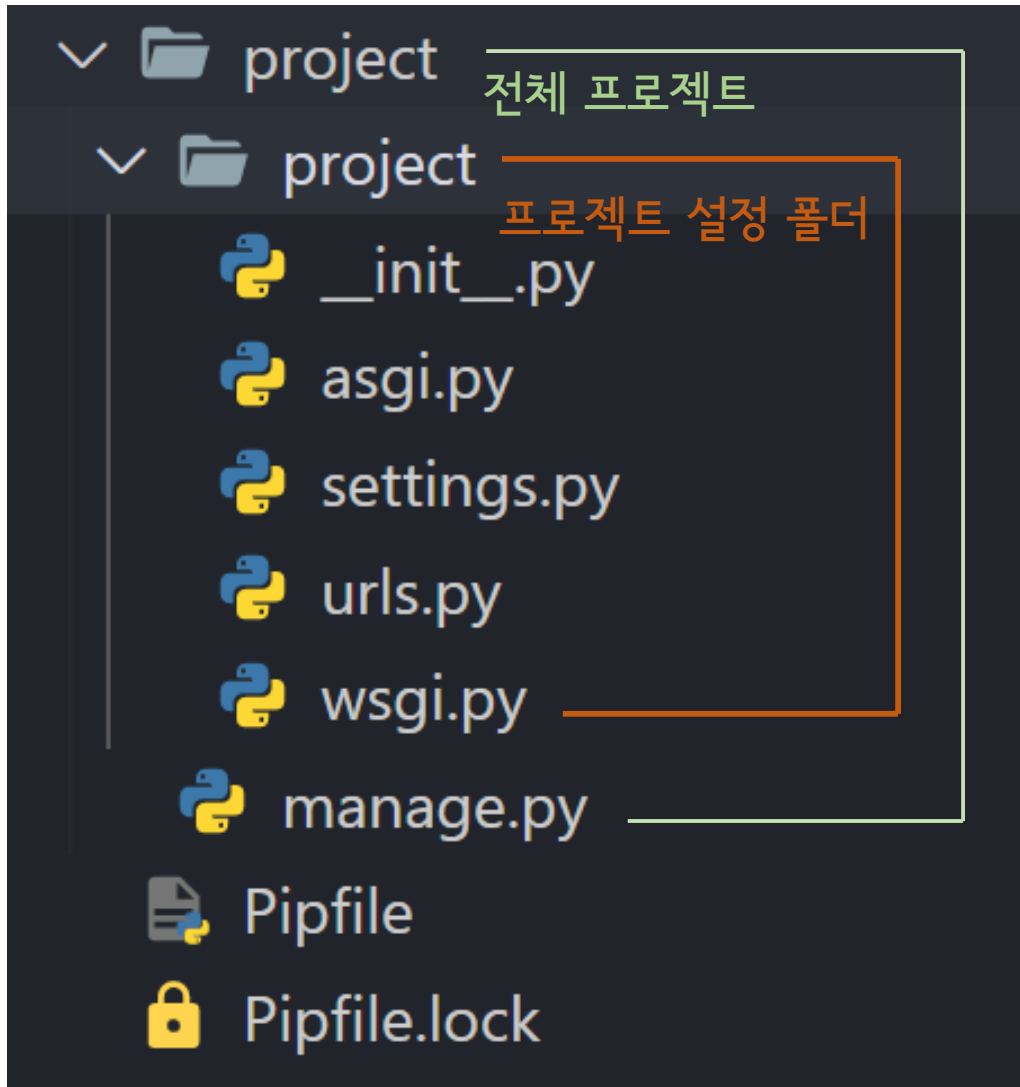
```
$ pipenv shell
```

```
$ pipenv install django
```

장고 프로젝트 생성

\$ django-admin startproject 프로젝트명
ex) project

프로젝트 구조



manage.py

- 스크립트, 사이트 관리를 도와주는 역할
- 다른 작업 없이 컴퓨터에서 웹 서버를 시작할 수 있다

settings.py

- 웹사이트 설정이 있는 파일

urls.py

- url 패턴 목록을 포함하고 있는 파일

프로젝트 실행

manage.py가 있는 위치에서
/session7/project/

```
$ python manage.py runserver
```

=> 장고 서버 실행

프로젝트 실행

Starting development server at <http://127.0.0.1:8000/>
Quit the server with CTRL-BREAK.

개발 서버가 <http://127.0.0.1:8000/>로 시작되었다.



The install worked successfully! Congratulations!

You are seeing this page because `DEBUG=True` is in your settings file and you have not configured any URLs.

http://127.0.0.1:8000/

http://localhost:8000/

127.0.0.1

localhost

=> 여러분의 PC를 가리키는 주소

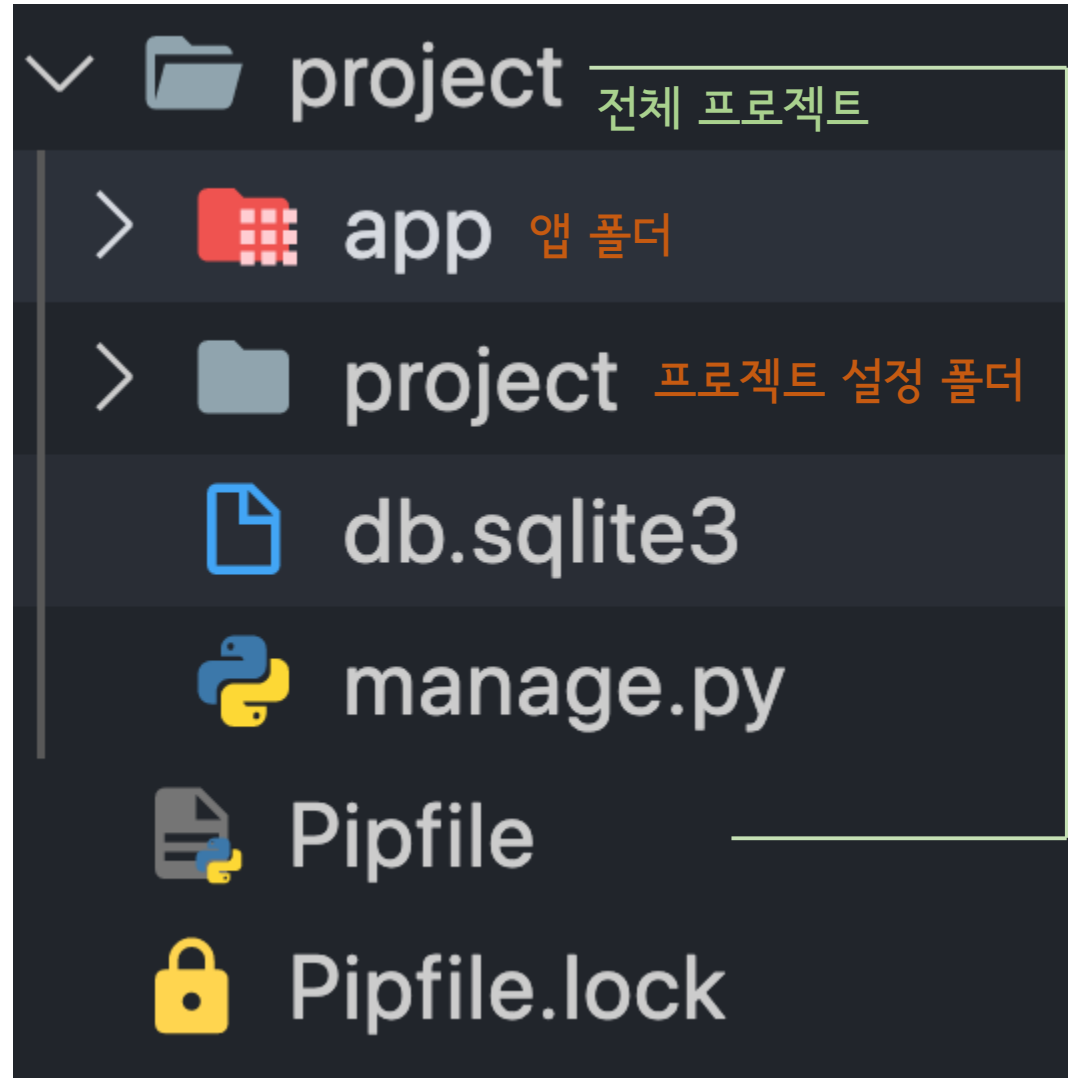
다른 사람들은 접속할 수 없습니다 ㅜㅜ

장고 앱 생성

\$ python manage.py startapp 앱이름

ex) app

하나의 프로젝트에 여러 개의 앱을 추가할 수 있다..!



settings.py 자세히 보기

DEBUG

- 개발 모드에서는 True
- 배포할 때는 False

INSTALLED_APPS

- 해당 프로젝트에 설치된 앱들
- 하나의 프로젝트에는 여러 개의 앱들 가능

TEMPLATES

- 템플릿 엔진, 경로, 옵션 등 설정

TIME_ZONE

- 시간 설정

DATABASES

- 데이터베이스 설정
- 기본적으로 sqlite로 설정되어 있음

앱을 생성했으니 settings.py에 추가!

```
INSTALLED_APPS = [  
    'django.contrib.admin',  
    'django.contrib.auth',  
    'django.contrib.contenttypes',  
    'django.contrib.sessions',  
    'django.contrib.messages',  
    'django.contrib.staticfiles',  
    'app',  
]
```

시간도 한국 시간에 맞춰볼까요?

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


데이터 베이스 생성을 위해

```
$ python manage.py makemigrations
```

```
$ python manage.py migrate
```

자세한 설명은 다음주에..!

데이터베이스에 **변화**가 있으면 쳐줘야 된다!

관리자 계정 생성

```
$ python manage.py createsuperuser
```

장고는 기본적으로 관리자 기능을 제공

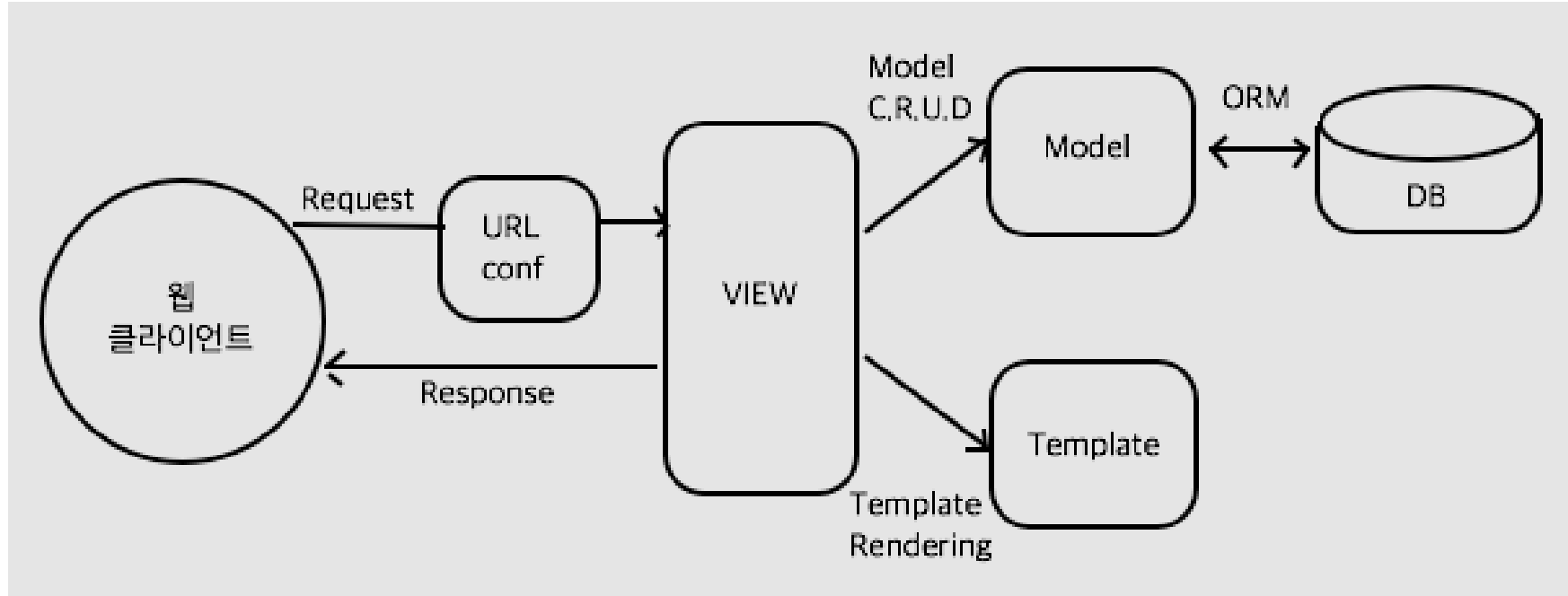
관리자 기능을 사용하기 위한 관리자 계정을 생성하기 위한 명령어

관리자 페이지 들어가기

<http://127.0.0.1:8000/admin>

슈퍼 유저로 로그인

장고는 어떻게 동작하는가?



MTV 패턴

- 장고의 설계 패턴
- 일반적으로 웹 개발 시 언급되는 MVC와 같은 개념

장고는 어떻게 동작하는가?

Model

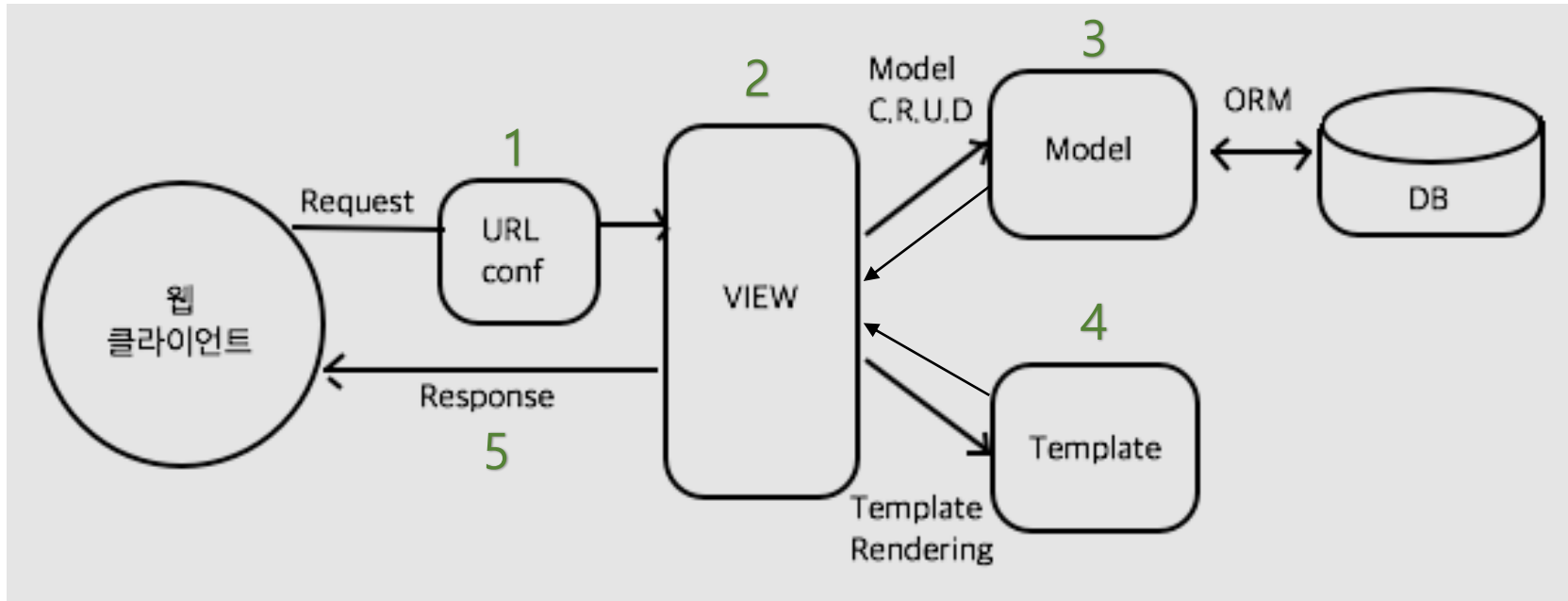
- 데이터베이스 저장되는 데이터의 영역

Template

- 사용자에게 보여지는 영역, 화면
- HTML

View

- 요청에 따라 Model에서 필요한 데이터를 Template에게 전달하는 역할
- 함수



1. urls.py에서 요청 들어온 URL을 분석(URL conf)
2. 해당 URL에 매칭되는 View 실행
3. View는 Request에 따라 Model을 통해 데이터베이스 처리
 - 생성(Create), 조회(Read), 업데이트(Update), 삭제>Delete)
4. View는 템플릿을 사용하여 클라이언트에 전송할 HTML 파일 생성
5. View는 최종으로 HTML 파일을 클라이언트에게 Response로 보낸다.

왜 MTV?

- 데이터, 사용자 인터페이스, 데이터를 처리하는 로직을 구분해서 한 요소가 다른 요소들에 영향을 주지 않도록 설계하는 방식
- 독립적인 영역에서 개발이 가능

예를 볼까요?

‘/count’라는 url로 요청이 들어왔다!

1. urls.py에서 'count'을 찾습니다.

```
from django.contrib import admin
from django.urls import path, include
from blog import views

urlpatterns = [
    path('admin/', admin.site.urls),
    path('', include('blog.urls')),
    path('count', views.count, name="count"),
    path('result', views.result, name="result"),
]
```

2. 'count'을 매칭되는 View 실행

```
from django.contrib import admin
from django.urls import path, include
from blog import views

urlpatterns = [
    path('admin/', admin.site.urls),
    path('', include('blog.urls')),
    path('count', views.count, name="count"),
    path('result', views.result, name="result"),
]
```

views 파일의 count 함수 실행

4, 5

```
def count(request):  
    return render(request, 'count.html')  
|
```

* urls.py에서 인자로 들어간 views의 함수들은 기본으로 request라는 인자를 가진다

템플릿을 사용하여 HTML 파일을 생성, 클라이언트에게 Response 전달

(해당 View는 데이터베이스를 거치지 않기 때문에 3번 과정이 없습니다)

만들어볼까요?


‘/hello’로 요청을 보내면
환영하는 화면

urls.py

해당 url에 맞는 View를 실행시키기 위해 app 폴더에서 views를 import

```
from django.contrib import admin
from django.urls import path, include
from app import views

urlpatterns = [
    path('admin/', admin.site.urls),
    path('hello', views.hello, name="hello")
]
```



html 파일에서 링크 이동에 필요합니다..!

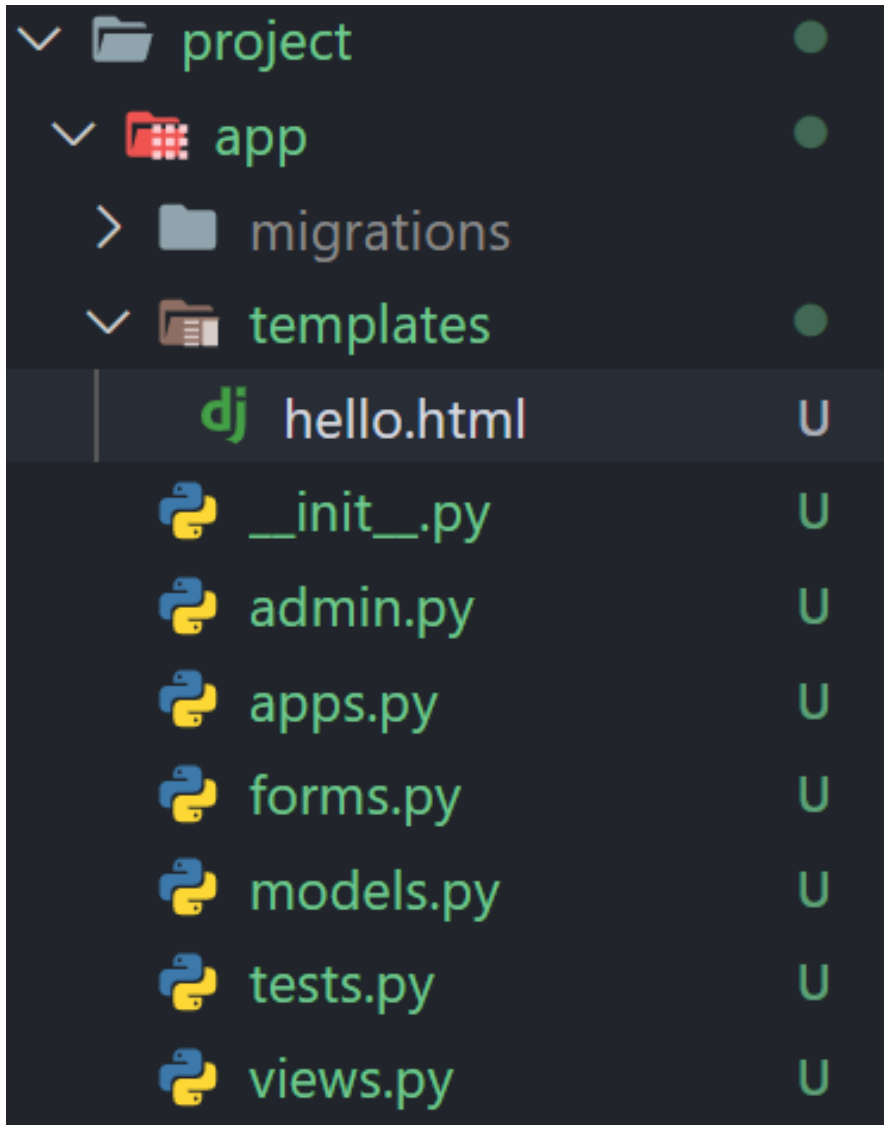
views.py

```
from django.shortcuts import render
from .forms import PostForm
from .models import Post
from django.shortcuts import redirect
# Create your views here.

def hello(request):
    return render(request, 'hello.html')
```

url 'hello' 에 해당하는 hello 함수 작성

hello.html 만들기



1. app 디렉토리 밑에 templates 폴더 생성
2. templates 폴더 밑에 hello.html 생성

장고는 기본적으로 앱(App) 디렉터리 하위에 있는 templates 디렉터리도 템플릿 디렉터리로 인식

- settings.py **TEMPLATES** 옵션에서 변경 및 추가 가능

hello.html 작성

```
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
</head>

<body>
  <h2>환영합니다!</h2>
</body>

</html>
```


<http://127.0.0.1:8000/hello>

환영합니다!

처음부터 복습..!

\$ pipenv shell

\$ pipenv install django

\$ django admin-startproject 프로젝트명(ex) project)

\$ 프로젝트 폴더로 들어가기(ex) cd project)

\$ python manage.py startapp 앱이름(ex) app)

\$ python manage.py makemigrations

\$ python manage.py migrate

\$ python manage.py createsuperuser

\$ python manage.py runserver

오늘의 목표

글자수 세기 페이지 만들기

- 글자를 입력하고 제출을 하면 몇 글자인지 알려준다.

글자수 페이지 만들기

(pipenv 켜져 있는지 확인, pipfile이 있는 프로젝트 최상단에서)

```
$ django-admin startproject CountProject
```

```
$ cd CountProject
```

```
$ python manage.py startapp CountApp
```

```
$ python manage.py makemigrations
```

```
$ python manage.py migrate
```

```
$ python manage.py createsuperuser
```

```
$ python manage.py runserver
```

먼저 글자수를 입력하는 화면을 만들어서 보여주자..!

settings.py

```
INSTALLED_APPS = [  
    ...,  
    'CountApp',  
]
```

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

urls.py

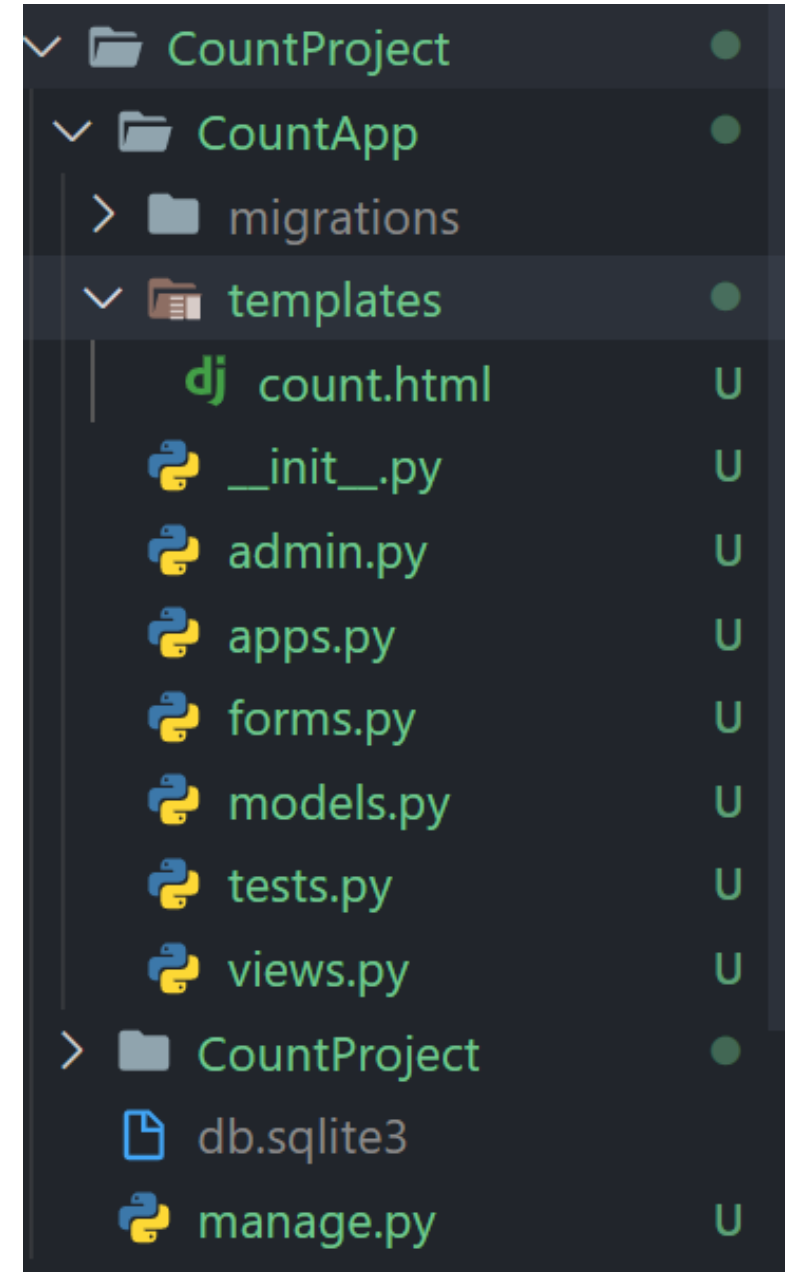
```
...  
from CountApp import views  
  
urlpatterns = [  
    ...  
    path('count', views.count, name="count")  
]
```

views.py

```
...  
def count(request):  
    return render(request, 'count.html')
```


count.html 생성

CountApp > templates > count.html



count.html

```
<body>
  <h1>글자를 입력해보세요!</h1>
  <form action="{% url 'result' %}" method="POST">{% csrf_token %}
    <textarea name="text" cols="60" rows="10"></textarea>
    <br>
    <input type="submit" value="제출하기">
  </form>
</body>
```

{% %}

Template 안에서 쓰는 장고 문법

* 나중에 Template 엔진이 html로 바꿔준다

csrf_token


csrf 해킹 공격 방지용

* method가 POST일 때 꼭 적어줍니다.

count.html

제출 누르면 name="result"인 url pattern으로 이동

```
<body>
  <h1>글자를 입력해보세요!</h1>
  <form action="{% url 'result' %}" method="POST">{% csrf_token %}
    <textarea name="text" cols="60" rows="10"></textarea>
    <br>
    <input type="submit" value="제출하기">
  </form>
</body>
```



urls.py

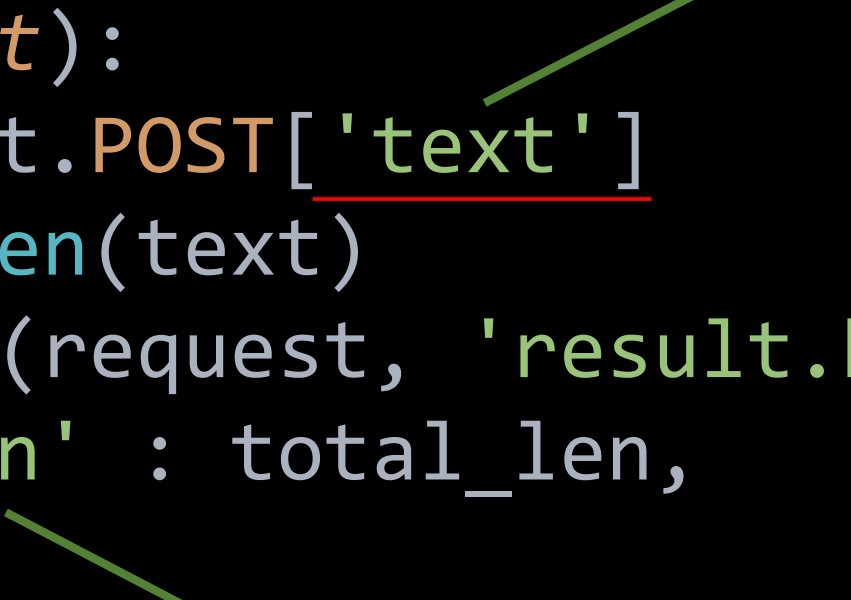
```
...  
from CountApp import views  
  
urlpatterns = [  
    ...  
    path('result', views.result, name="result")  
]
```

views.py

html 태그 name과 똑같이

...

```
def result(request):  
    text = request.POST['text']  
    total_len = len(text)  
    return render(request, 'result.html', {  
        'total_len' : total_len,  
    })
```



html로 데이터를 보낼 때는 이렇게!

result.html

```
return render(request, 'result.html', {  
    'total_len' : total_len,  
})
```



```
<body>  
    글자수 : {{total_len}}  
</body>
```

{{ }} html에서 받은 data를 사용할 때

실습

입력한 텍스트, 공백 제외 글자수도 추가로 보여주기

views.py
result.html
수정..

views.py

...

```
def result(request):
    text = request.POST['text']
    total_len = len(text)
    no_blank_len = len(text.replace(" ", ""))
    return render(request, 'result.html', {
        'total_len' : total_len,
        'text' : text,
        'no_blank_len' : no_blank_len,
    })
```

result.html 입력

```
<body>
    입력하신 텍스트 : {{text}}
    <br>
    공백 포함 글자수 : {{total_len}}
    <br>
    공백 제외 글자수 : {{no_blank_len}}
</body>
```

CSS로 꾸며볼까요?

...

```
STATIC_URL = '/static/'
```

- 이미지, 자바스크립트, 스타일시트 같은 파일들
- 템플릿과 마찬가지로 앱 디렉토리 바로 밑에 static 디렉토리 인식

```
CountApp
└─ static
```

참고

STATICFILES_DIRS

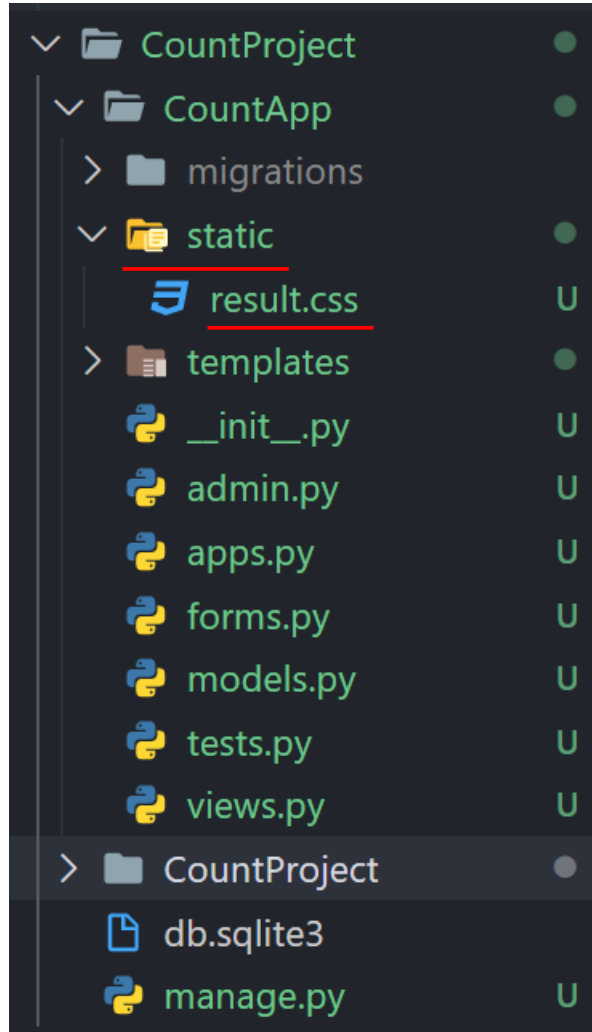
- 프로젝트 전반에 사용하는 static 파일들

STATIC_ROOT

- 배포할 때 흩어져 있는 static 파일들 모이는 위치

* `python manage.py collectstatic`

CSS로 꾸며볼까요?



result.css

```
body {  
    color: red;  
}
```

result.html

```
<head>
  {% load static %}
  <link rel="stylesheet" type="text/css" href="{% static 'result.css' %}">
  ...
</head>
```

협업을 위한 **.gitignore**

.gitignore 파일에 적힌 파일 또는 폴더는 git에서 무시한다

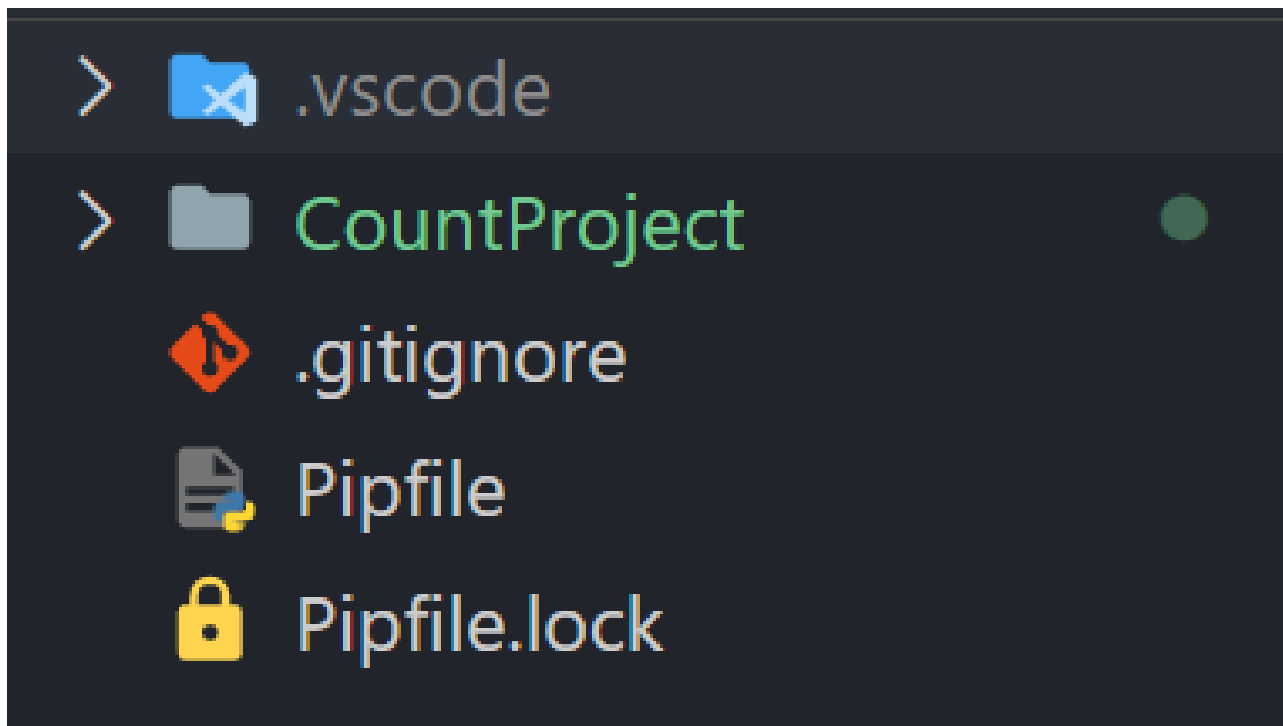
-> github에 올라가지 않는다

- 개인 정보, 비밀번호 등 공개되면 안되는 파일, 폴더들을 적는다
- 각자 컴퓨터에 생기는 폴더 및 파일을 gitignore 파일에 적어 협업할 때 문제가 생기지 않게 하자!

ex) .vscode, migrations, db.sqlite3, __pycache__

최상단 디렉토리에 .gitignore 파일 만들어주기

(해당 위치에서 \$ git init 실시)



.gitignore

`.vscode`

`migrations`

`db.sqlite3`

`__pycache__`

순서가 중요!

.gitignore 파일 생성 및 작성

> \$ git init

> \$ git add .

> ..

\$ git add를 하기 전에 생성하고 적을 것!

과제

- 현재 기능에서 단어 개수 세는 기능 추가!
- count.html, result.html 꾸미기

멋쟁이사자처럼 여러분, 모두 과제 화이팅입니다.

공백포함

26 46 byte

공백제외

22 42 byte

단축키 안내 ▾