

장고[Django] 기초

# Session 7

NEXT X LIKELION 김나영

# 01\_장고[django]란? 🤔

---

# Intro

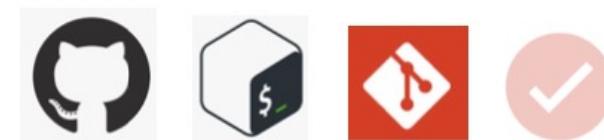
## Programming Language



## Framework

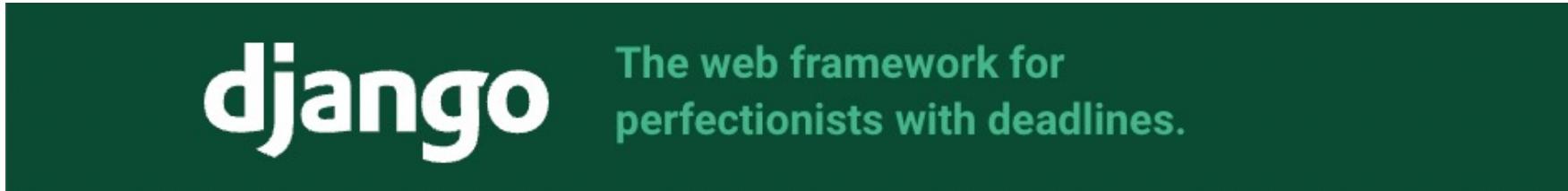


## Tools



| 장고?

django



The banner features the word "django" in large white lowercase letters on a dark green background. To its right, the tagline "The web framework for perfectionists with deadlines." is written in a smaller white sans-serif font.

django

The web framework for  
perfectionists with deadlines.

OVERVIEW DOWNLOAD DOCUMENTATION NEWS COMMUNITY CODE ISSUES ABOUT ♥ DONATE ⓘ

Django makes it easier to build better web apps more quickly and with less code.

Get started with Django

## Meet Django

Django is a high-level Python web framework that encourages rapid development and clean, pragmatic design. Built by experienced developers, it takes care of much of the hassle of web development, so you can focus on writing your app without needing to reinvent the wheel. It's free and open source.

Download latest release: 4.1.7

DJANGO DOCUMENTATION >

Support Django!

# 장고?

django

- 장고(django) : 파이썬 기반 “웹 프레임워크”

1. 웹 프로그램을 ‘쉽고’ ‘빠르게’ 개발할 수 있다.

2. 편리하다 : 이미 구현되어 있는 기능이 많다

- 로그인, 관리자 기능, 데이터베이스 등

3. 안전하다 : 보안 공격을 기본으로 막아준다

- SQL 인젝션, XSS, CSRF, 클릭재킹 등 해킹에 대한 보안 기능 기본 제공

# ‘프레임워크’ 장고

프레임워크 vs 라이브러리

- **프레임워크 (framework)** : 웹 애플리케이션을 구축할 때,  
공통적인 개발 환경을 제공해주는 구조를 짜놓고 그 위에 덧붙여 만들도록 하는 것

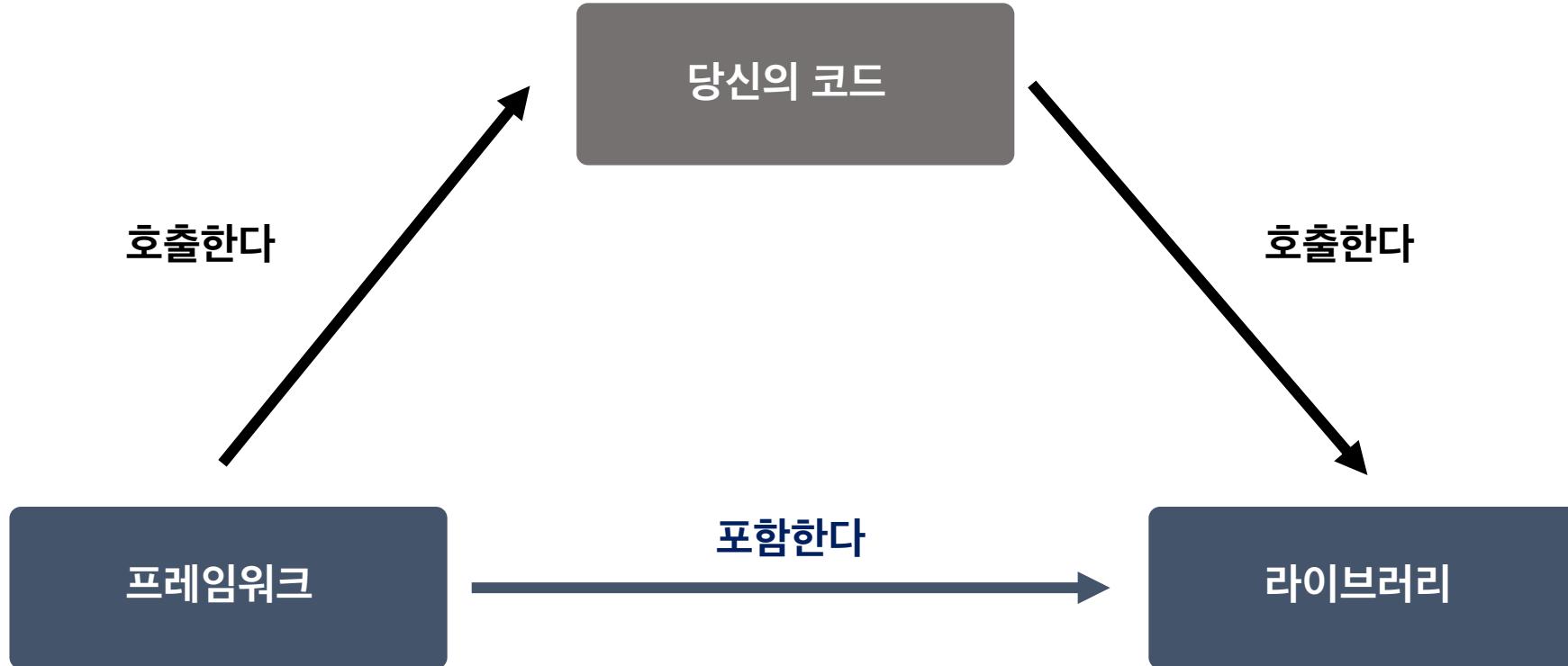
- ‘들어가서’ 사용한다: 프레임워크가 전체적인 흐름을 줘고 있고, 사용자는 그 안에서 필요한 코드를 짜넣는다

- **라이브러리 (library)** : 단순 활용 가능한 도구들의 집합

- ‘가져다가’ 사용한다: 사용자가 전체적인 흐름을 줘고 있고, 필요한 라이브러리를 호출하여 가져다 쓴다

# ‘프레임워크’ 장고

프레임워크 vs 라이브러리





**잠깐! 중간 점검**

# Python 버전 통일 확인

명령어

Mac

**python --version**

(이상하면 python3 --version)

Windows

**python --version**

확인

가상환경 켜지 않은 곳에서 python 3.10.10

# | 가상 환경 Pipenv 복습

## Pipenv 명령어 정리

\$ pipenv shell

-가상 환경 생성 및 시작

\$ exit

-가상 환경 종료

\$ pipenv install 패키지명

-해당 패키지 설치

\$ pipenv uninstall 패키지명

-해당 패키지 제거

## 02\_장고 시작



# | 장고 설치

Django를 설치해봅시다 😍

\$ mkdir session7

-작업할 폴더 생성(영문!)

\$ cd session7

-해당 폴더로 이동

\$ pipenv shell

-pipenv로 가상환경 생성

\$ pipenv install django

-pipenv로 가상환경 내에 Django 패키지 설치

( \$ django-admin --version

-장고가 잘 설치되었는지 확인

(버전 정보가 잘 뜨면 잘 설치 된 것! )

# 장고 초기 세팅

장고 초기 세팅의 5단계

## 1. 장고 프로젝트 생성

명령어

\$ django-admin startproject **프로젝트명** Ex) myproject

-장고의 각종 세팅을 포함한 기본 구조 프로젝트 생성!

```
(session7) → session7 django-admin startproject myproject
(session7) → session7 ls
Pipfile      Pipfile.lock myproject
(session7) → session7 code .
(session7) → session7 |
```

# 장고 초기 세팅

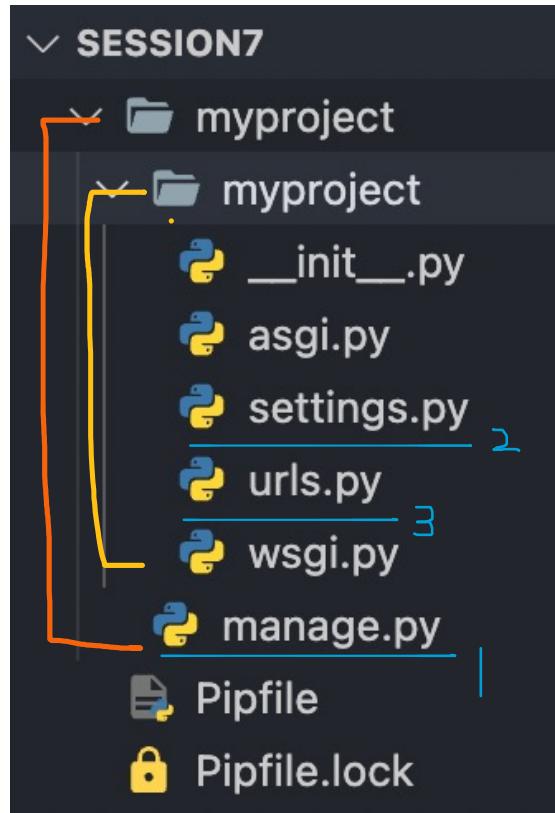
장고 초기 세팅의 5단계

## 1. 장고 프로젝트 생성

결과

전체 프로젝트

프로젝트 설정폴더



### 프로젝트 구조를 살펴봅시다.

- 전체 프로젝트 myproject 하위에 / 프로젝트 설정폴더 myproject가 있고 / 그 하위 파일 중 우리가 잘 알아두어야 할 친구들은 다음과 같아요.

#### 1. manage.py

- django project 관리를 도와주는 command-line utility 제공
- 다른작업 없이 컴퓨터에서 웹서버 실행
- django app 생성
- 관리자 계정 생성
- DB초기화및변경사항반영

#### 2. settings.py

- django project의 각종 설정을 포함하고 있는 파일

#### 3. urls.py

- url 패턴 목록을 포함하고 있는 파일

# 장고 초기 세팅

장고 초기 세팅의 5단계

## 2. 장고 앱 생성

명령어

\$ python manage.py startapp 앱 이름 Ex) myapp

-manage.py가 있는 위치에서(/session7/myproject/) 진행

```
(session7) → myproject ls  
db.sqlite3 manage.py myproject  
(session7) → myproject python manage.py startapp myapp  
(session7) → myproject ls  
db.sqlite3 manage.py myapp      myproject  
(session7) → myproject code .  
(session7) → myproject |
```

# 장고 초기 세팅

장고 초기 세팅의 5단계



## 앱이 뭔가요?

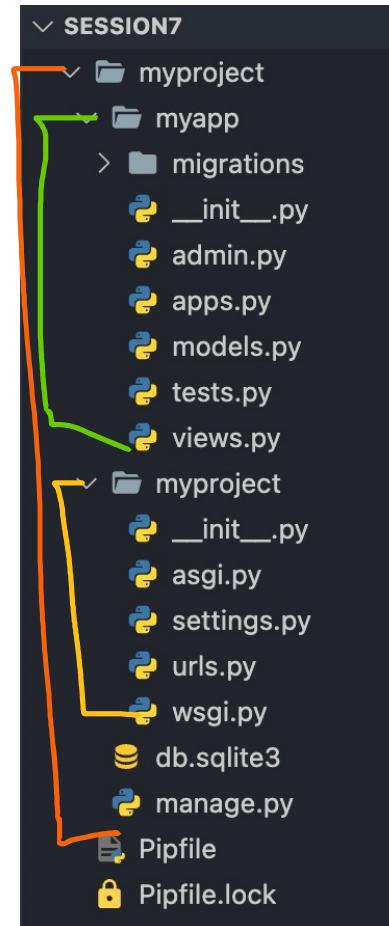
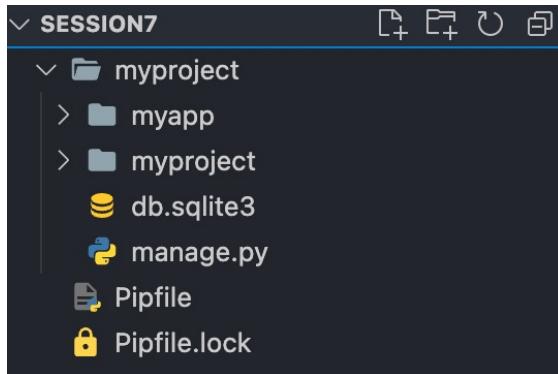
- 앱: 프로젝트 내에서 [특정 기능]을 담당하는 web application
  - : 결제, 로그인, 게시판, 장바구니 등
- 하나의 프로젝트는 여러 개의 앱으로 구성될 수 있음
- 각 앱에서 특정 기능과 관련된 코드만 묶어서 관리할 수 있으므로 편리함

# 장고 초기 세팅

장고 초기 세팅의 5단계

## 2. 장고 앱 생성

결과



전체 프로젝트

프로젝트 설정폴더

앱 폴더

# 장고 초기 세팅

장고 초기 세팅의 5단계

## 2. 장고 앱 생성

세팅하기

새로 생성한 **앱 이름**을 `settings.py`의 앱 리스트에 추가해야 합니다.

-/session7/myproject/myproject/settings.py

```
31  # Application definition
32
33  INSTALLED_APPS = [
34      'django.contrib.admin',
35      'django.contrib.auth',
36      'django.contrib.contenttypes',
37      'django.contrib.sessions',
38      'django.contrib.messages',
39      'django.contrib.staticfiles',
40      'myapp',
41 ]
```

# 장고 초기 세팅

장고 초기 세팅의 5단계

## 🤔 settings.py 자세히 보기

- DEBUG 개발 모드에서는 True, 배포 시에는 False
- INSTALLED\_APPS 해당 프로젝트에 설치된 앱들 명시
- TEMPLATES 템플릿 엔진, 경로, 옵션 등 설정
- LANGUAGE 기본 언어 설정(기본값: en-us , 한국: ko-kr)
- TIME\_ZONE 시간 설정(기본값: UTC, 한국: Asia/Seoul)
- DATABASES 데이터베이스 설정(기본값: sqlite)

\*추가 settings는 docs 참조: <https://docs.djangoproject.com/en/3.1/ref/settings/>

# 장고 초기 세팅

장고 초기 세팅의 5단계

## 3. 데이터베이스 생성

명령어

**\$ python manage.py makemigrations**

**\$ python manage.py migrate**

-manage.py가 있는 위치에서(/session7/myproject/) 진행

-**DataBase 관련 변경사항이 있을 때마다**, 적용을 위해 이 작업을 해야합니다.  
(DB에 대해서는 추후에 자세히 다룰 예정입니다!)

(session7) → **myproject** python manage.py makemigrations

(session7) → **myproject** python manage.py migrate

# 장고 초기 세팅

장고 초기 세팅의 5단계

## 😱 makemigrations? migrate?

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

‘테이블 작업 파일’ 생성하기

(모델 수정/변경 시)

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

‘테이블’ 생성하기

# 장고 초기 세팅

장고 초기 세팅의 5단계

## 3. 데이터베이스 생성

### 결과

간단히 설명하자면,  
Admin / auth / contenttypes / session  
앱이 사용하는 테이블이 생성되었다.

(지금 구체적으로 알 필요는 없음!)

```
(session7) → myproject python manage.py migrate
Operations to perform:
  Apply all migrations: admin, auth, contenttypes, sessions
Running migrations:
  Applying contenttypes.0001_initial... OK
  Applying auth.0001_initial... OK
  Applying admin.0001_initial... OK
  Applying admin.0002_logentry_remove_auto_add... OK
  Applying admin.0003_logentry_add_action_flag_choices... OK
  Applying contenttypes.0002_remove_content_type_name... OK
  Applying auth.0002_alter_permission_name_max_length... OK
  Applying auth.0003_alter_user_email_max_length... OK
  Applying auth.0004_alter_user_username_opts... OK
  Applying auth.0005_alter_user_last_login_null... OK
  Applying auth.0006_require_contenttypes_0002... OK
  Applying auth.0007_alter_validators_add_error_messages... OK
  Applying auth.0008_alter_user_username_max_length... OK
  Applying auth.0009_alter_user_last_name_max_length... OK
  Applying auth.0010_alter_group_name_max_length... OK
  Applying auth.0011_update_proxy_permissions... OK
  Applying auth.0012_alter_user_first_name_max_length... OK
  Applying sessions.0001_initial... OK
```

# 장고 초기 세팅

장고 초기 세팅의 5단계

## 4. 장고 서버 실행

명령어

**\$ python manage.py runserver**

-manage.py가 있는 위치에서(/session7/myproject/) 진행

```
(session7) → session7 ls  
Pipfile      Pipfile.lock myproject  
(session7) → session7 cd myproject  
(session7) → myproject python manage.py runserver
```

# 장고 초기 세팅

장고 초기 세팅의 5단계

## 4. 장고 서버 실행

결과

```
(session7) → myproject python manage.py runserver
Watching for file changes with StatReloader
Performing system checks...

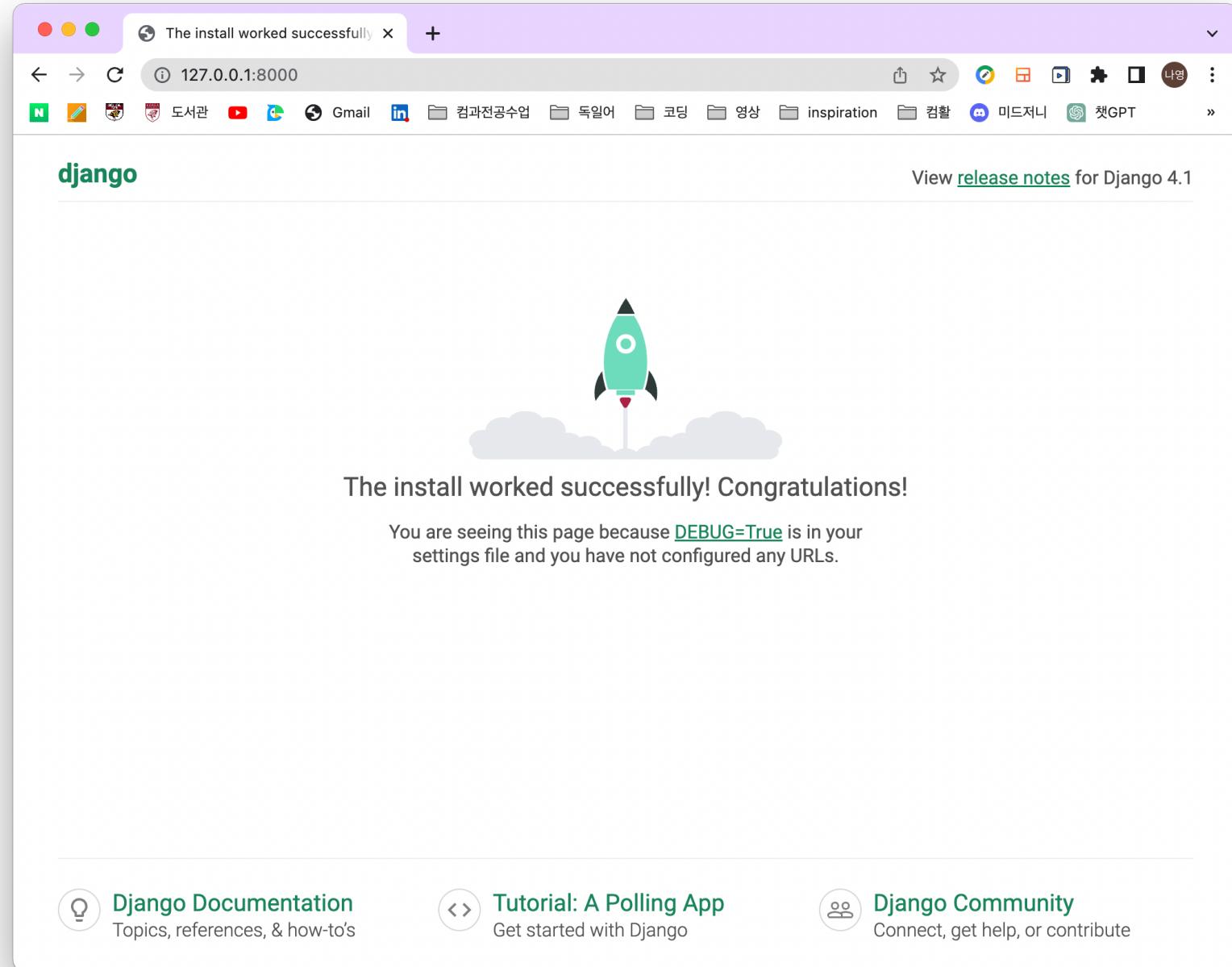
System check identified no issues (0 silenced).
March 24, 2023 - 03:04:58
Django version 4.1.7, using settings 'myproject.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CONTROL-C.
```

⇒ 개발 서버(== 로컬 서버)가 <http://127.0.0.1:8000/>에서 실행되었습니다!

이 서버에는 로컬에서만 접속 가능합니다.

- ▶ 127.0.0.1 == local host (== 여러분의 pc를 가리키는 주소)
- ▶ :8000 == 포트번호

⇒ 종료하려면 CONTROL-C 누르기!



# 장고 초기 세팅

장고 초기 세팅의 5단계

## 5. 관리자 계정 생성

명령어

\$ python manage.py createsuperuser

-manage.py가 있는 위치에서(/session7/myproject/) 진행

- 장고는 **기본적으로 관리자 기능(Django Admin)을 제공**(관리자 계정 생성 후 이용)

```
(session7) → myproject python manage.py createsuperuser
사용자 이름 (leave blank to use 'and__young'): nayoung
이메일 주소 : tkfd9008a@naver.com
Password:
Password (again):
비밀번호가 너무 짧습니다. 최소 8 문자를 포함해야 합니다.
비밀번호가 너무 일상적인 단어입니다.
비밀번호가 전부 숫자로 되어 있습니다.
Bypass password validation and create user anyway? [y/N]: y
Superuser created successfully.
(session7) → myproject
```

# 장고 초기 세팅

장고 초기 세팅의 5단계

## 5. 관리자 계정 생성

Django 관리

사용자 이름:  
nayoung

비밀번호:  
....

Admin 사용

- 1) 서버를 켜고(`python manage.py runserver`)
- 2) <http://127.0.0.1:8000/admin> 접속 후
- 3) 슈퍼 유저로 로그인하여 admin 페이지 사용

## 03\_장고의 동작 과정



# 장고의 동작과정

이것만은 기억하자!

- **MTV** : 장고의 설계 패턴으로, 소프트웨어의 비즈니스 로직과 화면을 구분하는 데 중점을 둔 MVC(Model-View-Controller) 패턴의 개념을 용어만 달리하여 그대로 받아옴

**Model** : 데이터베이스 설계

- 데이터베이스에 저장되는 '데이터' 영역

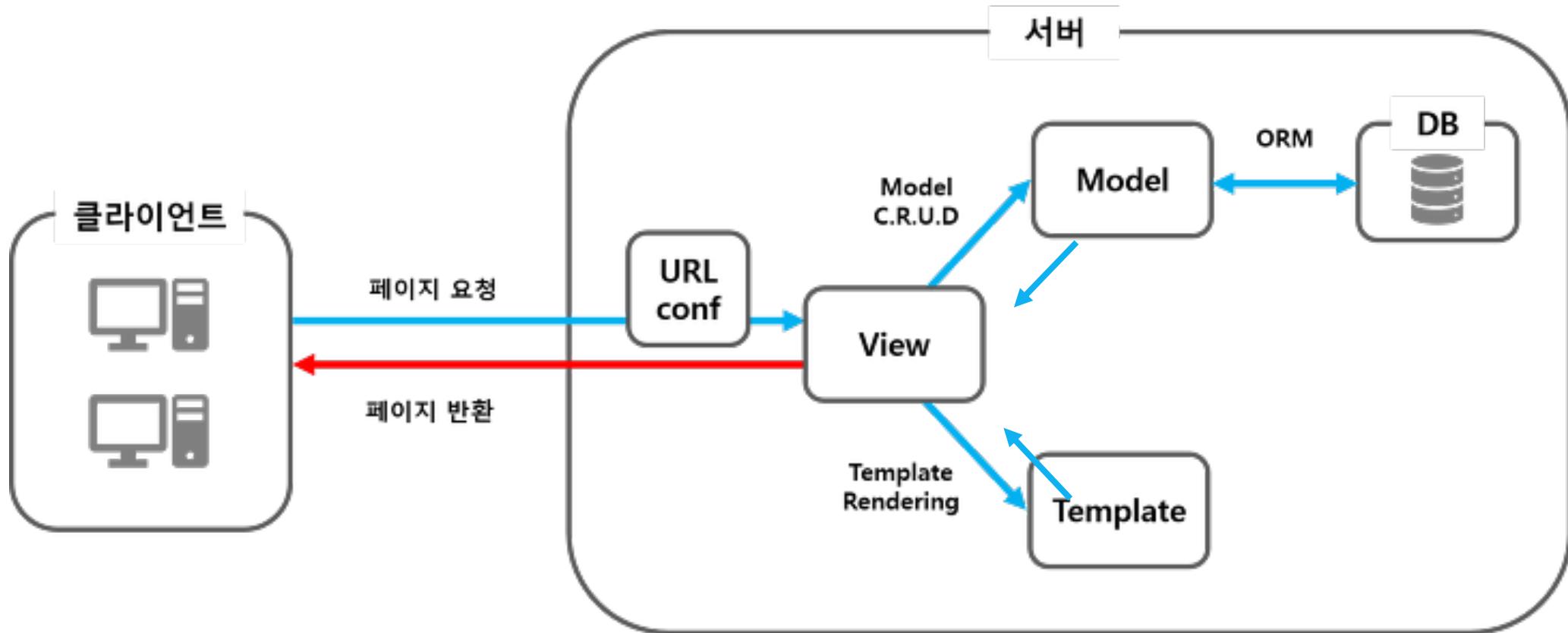
**Template** : 화면 UI 설계

- 사용자에게 보여지는 HTML '화면' 영역

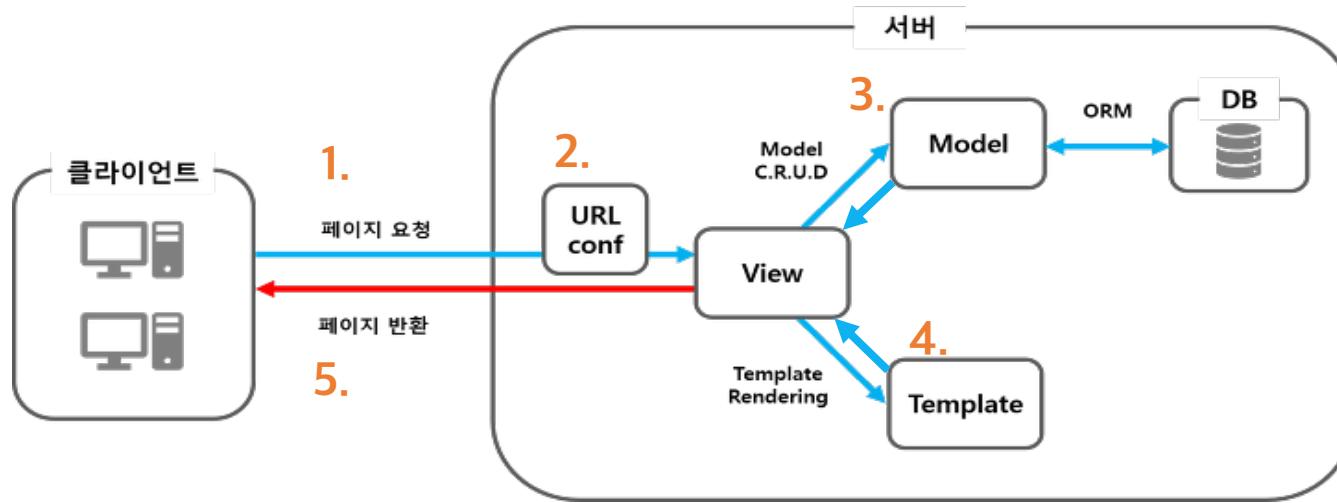
**View** : 프로그램 로직 설계

- 요청에 따라 Model에서 필요한 데이터 가져오기 → 처리 → 처리 결과를 Template에 전달

# MTV 패턴

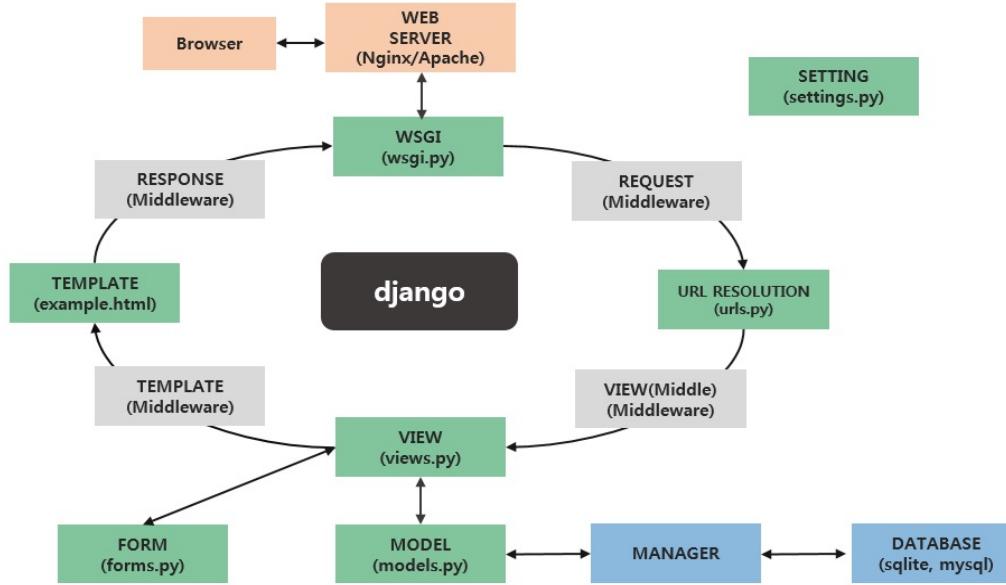


# MTV 패턴



1. urls.py에서 요청 들어온 URL을 분석 (**URL conf**)
2. 해당 URL에 매칭되는 **View** 실행
3. View는 요청에 따라 **Model**을 통해 DB 처리
  - 4가지 처리 가능: 생성(Create), 조회(Read), 수정(Update), 삭제(Delete)
4. View는 **Template**을 사용하여 클라이언트에게 응답할 HTML 파일 렌더링
5. View는 최종적으로 HTML파일을 클라이언트에게 **Response**

# MTV 장점?



- M(데이터) / T(사용자 인터페이스) / V(데이터 처리 로직) 가 구분되어 있다.
- 즉, 한 요소가 다른 요소들에 영향을 주지 않도록 설계되었으므로, 각각 **독립적**인 영역에서 **개발**이 가능하다

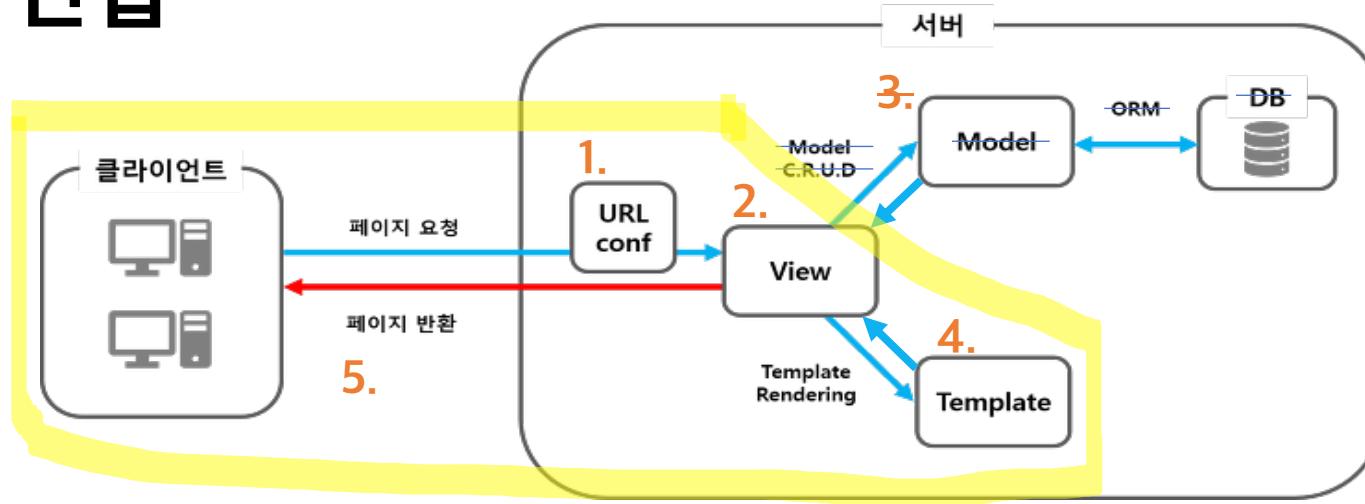
# | MTV 패턴 연습

'/hello'로 요청 보내면

→ 환영 인사 '반갑습니다'를 보여주는 페이지를 만들어 봅시다.

(일단 모델이 필요없는 연습부터 해봅시다!)

# MTV 패턴 연습



1. urls.py에서 요청 들어온 URL을 분석 (URL conf)
2. 해당 URL에 매칭되는 View 실행
3. View는 요청에 따라 Model을 통해 DB 처리  
— 4가지 처리 가능: 생성(Create), 조회(Read), 수정(Update), 삭제(Delete)
4. View는 Template를 사용하여 클라이언트에게 응답할 HTML 파일 렌더링
5. View는 최종적으로 HTML파일을 클라이언트에게 Response

# MTV 패턴 연습

## 1. URL conf

### url.py 에서 URL 지정해주기

경로

myproject > myproject > urls.py

CODE

1. URL과 View 연결을 위해, 'myapp' 디렉토리 안의 views를 import
2. urlpatterns에 내가 추가할 url path의 내용 작성

myproject > myproject > urls.py > ...

```
16 from django.contrib import admin
17 from django.urls import path
18 from myapp import views
19
20 urlpatterns = [
21     path('admin/', admin.site.urls),
22     path('hello', views.hello, name='hello'),
23 ]
```

path('hello', views.hello, name='hello')

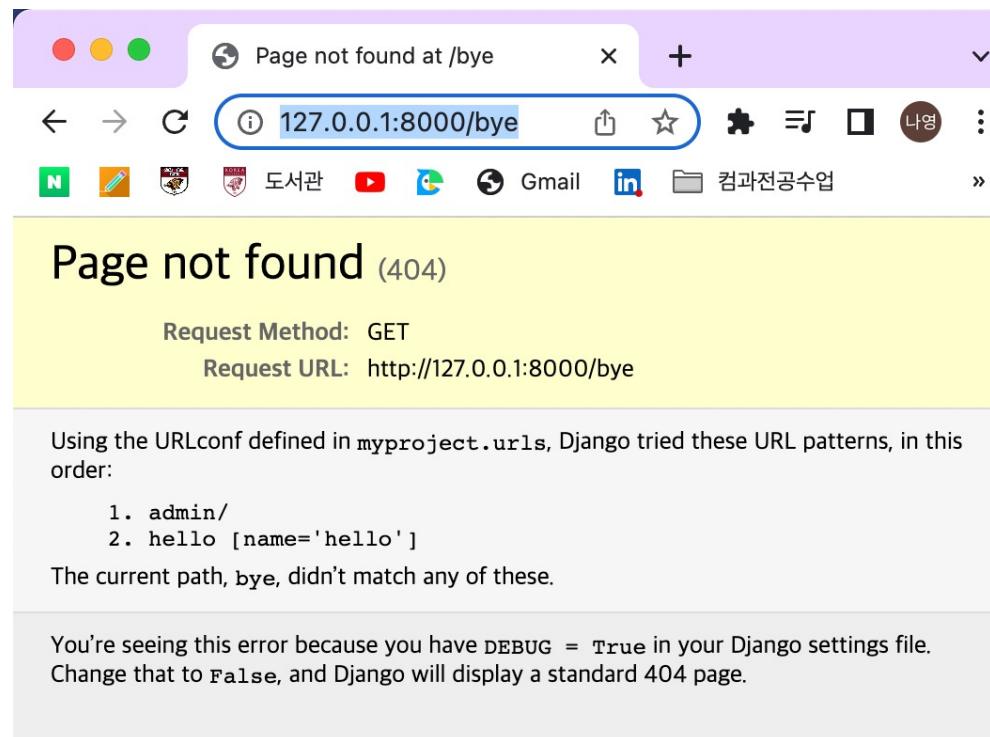
Views에서 작성해줄 함수명

Html Template에서 URL에 접근하기 위해 사용할 이름

# MTV 패턴 연습

## 1. URL conf

✋ 잘못된 URL에 접근한다면?



# MTV 패턴 연습

## 2. View

### views.py에서 View로직 작성하기

경로

myproject > myapp > views.py

CODE

1. url 'hello'에 연결되는 함수 hello 작성: def hello(request):
2. 로직 작성
3. 처리한 정보를 'hello.html'로 보냄

myproject > myapp >  views.py > ...

```
1  from django.shortcuts import render
2
3  # Create your views here.
4  def hello(request):
5      #로직 작성 부분
6      return render(request, 'hello.html')
```

render(request, 'hello.html')

- 의미: '템플릿 요소를 응답에 포함시켜서 화면에 띄워줄게!'
- 인수: (1) request    (2) 템플릿 이름  
       (3) view에서 사용하던 파이썬 변수를 html에 넘길 수 있음  
            (필요한 경우에 선택적으로 작성함)

# MTV 패턴 연습

3. Model

Model은 잠깐 pass

# MTV 패턴 연습

## 4. Template

### Template 만들기

경로

myproject > myapp > templates > hello.html

CODE

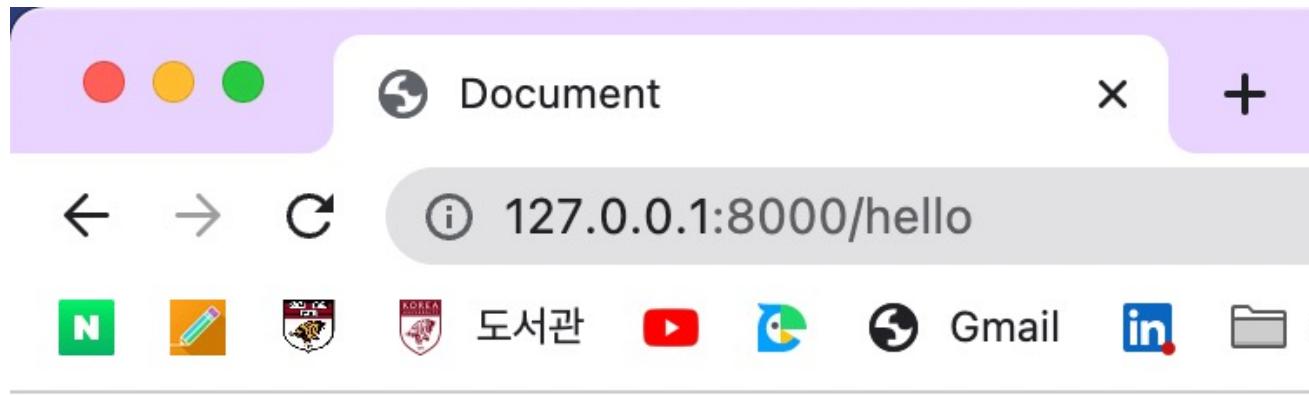
1. myapp 디렉토리 하위에 templates 폴더 생성 (폴더명 변경X)
2. templates 폴더 하위에 hello.html 템플릿 생성
3. hello.html 내용 작성 \*html 자동완성 안 되면: django html 문서에서 html 문서로 바꿔준 뒤에 !+tab키 눌러주기!

```
myproject > myapp > templates > hello.html > ...
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>Document</title>
8  </head>
9  <body>
10 | <h1>반갑습니다.</h1>
11 </body>
12 </html>
```

# MTV 패턴 연습

접속해볼까요?

<http://127.0.0.1:8000/hello>



반갑습니다.

쉬는 시간

# 05\_요청과 응답

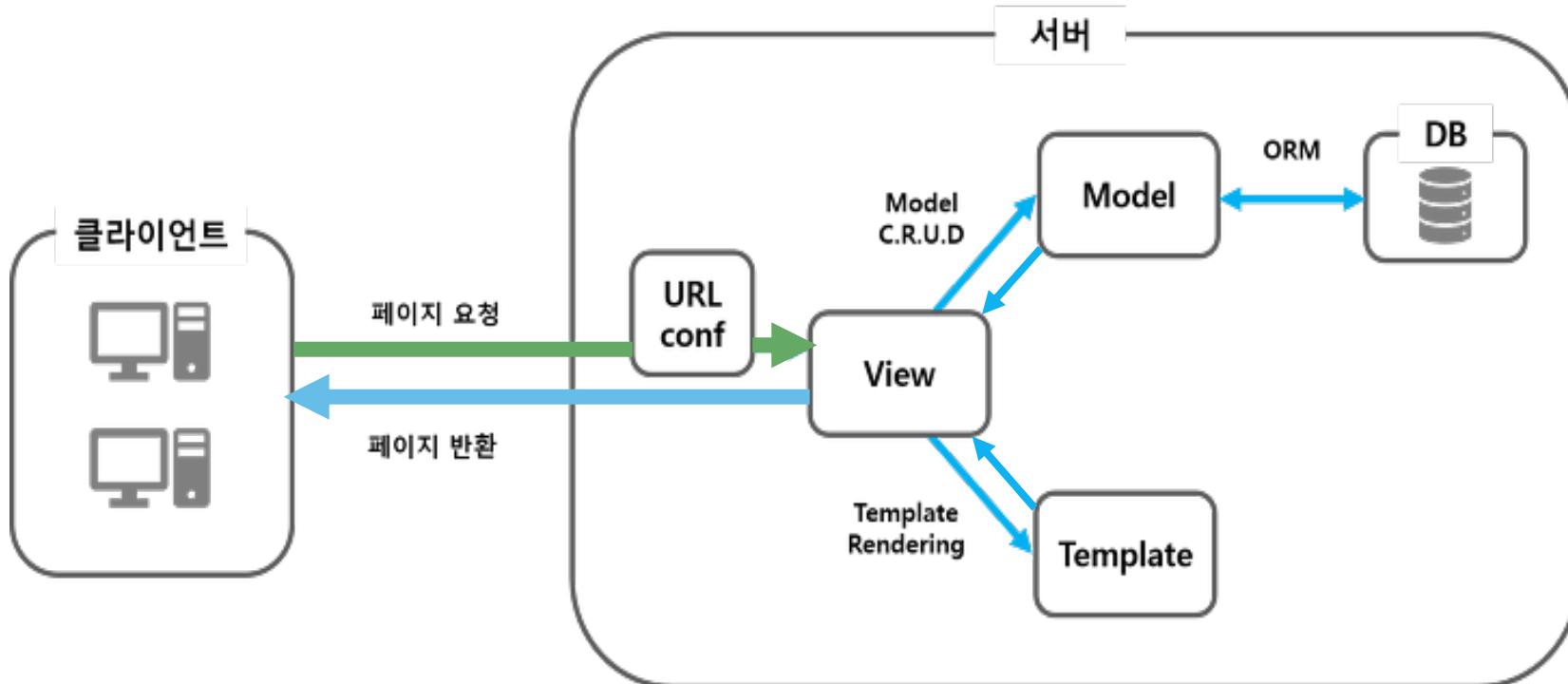


# 요청과 응답

Request & Response

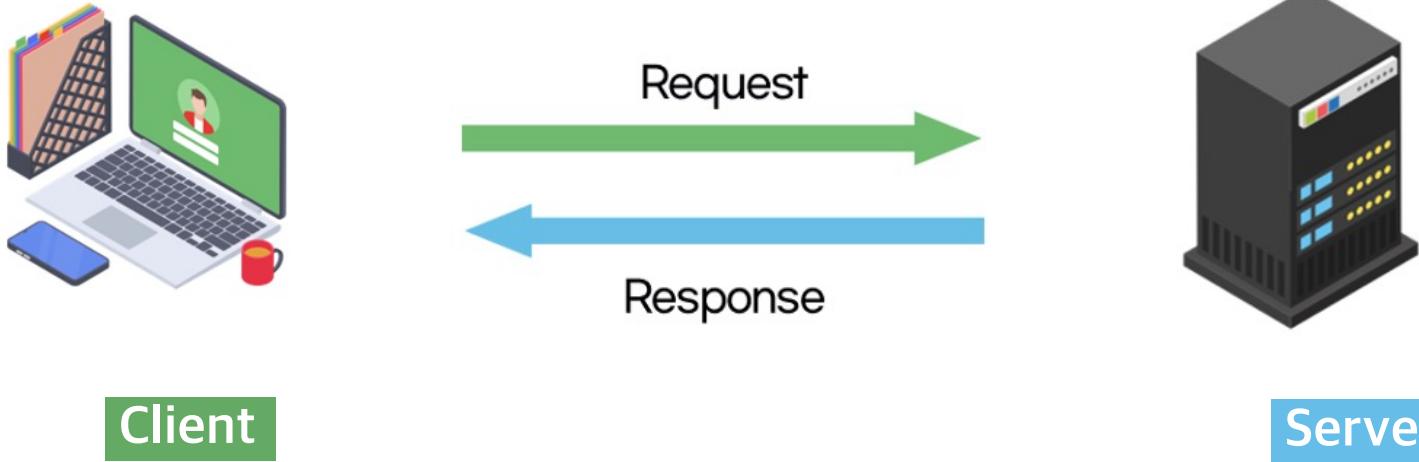
- **Http 프로토콜**  
(Hypertext Transfer Protocol)

: 인터넷상에서 데이터를 주고 받기 위한 **서버/클라이언트 모델**을 따르는 통신 규약



# 요청과 응답

Request & Response



- 서버에 요청(request)을 보냄
- 어떤 종류의 데이터든지 전송 가능  
(HTML 문서, 이미지, 동영상, 오디오, 텍스트…)
- 클라이언트 소프트웨어  
: IE, Chrome, Firefox, Safari…

- 클라이언트의 요청(request)을 받아서,  
요청을 해석하고 응답(response)을 보냄
- 서버 소프트웨어  
: Apache, nginx, IIS, lighttpd…

# 요청과 응답

Request & Response

## Request 예시

```
GET /search?q=hello&hl=ko HTTP/1.1  
Host: www.google.com
```

Client

Server

## Response 예시

```
HTTP/1.1 200 OK  
Content-Type: text/html; charset=UTF-8  
Content-Length: 3423  
  
<html>  
<body>...</body>  
</html>
```

# 요청과 응답

Request & Response

- **Http 메서드** : 메서드는 요청(request)의 종류를 서버에게 알려주기 위해서 사용한다

# “REST”

verb	Resource	Representation
GET	/movies	Get list of movies
GET	/movies/:id	Find a movie by its ID
POST	/movies	Create a new movie
PUT	/movies	Update an existing movie
DELETE	/movies	Delete an existing movie

😎 Session 5에서 봤던 내용 다들 기억 나시나요???

# 요청과 응답

Request & Response

## GET

존재하는 자원을 **조회할 때**

어떤 내용을 가지고 오고 싶을 때

읽기

URL에 변수(데이터)를 포함시켜 요청

## POST

새로운 자원을 **생성할 때**

새로운 내용을 쓸 때

저장/게시

URL에 변수(데이터)를 노출하지 않고 요청

## 06\_장고 실습



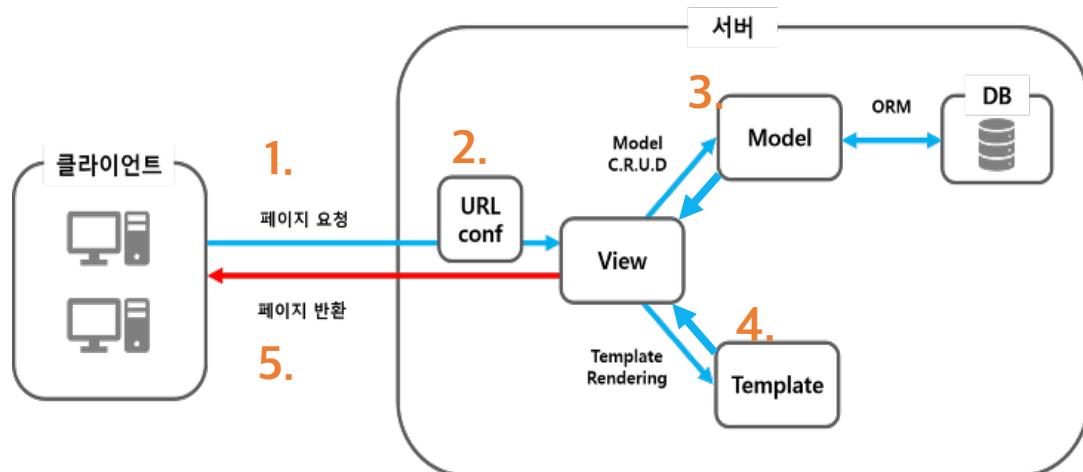
# 장고 실습

오늘의 목표!

텍스트 입력하고 제출하면 → 몇 글자인지를 알려주는  
‘글자수 세기’ 페이지를 만들어 봅시다.

# 장고 실습

큰 그림을 먼저 그려봅시다



1. 페이지가 총 2개 필요하므로, urls.py에서 URL 2개 추가해야지.
  - 글자 수 입력하는 페이지 : *count*
  - 글자 수 세서 결과 보여주는 페이지 : *result*
2. 해당 URL에 매칭되는 View 로직 작성해야지.
3. View는 요청에 따라 Model을 통해 DB 처리…는 PASS
  - 오늘은 Model 없이(즉 DB 처리 없이)  
요청(request)-응답(response) 과정 구현에 충실히봅시다!
4. View가 Template 사용하여 클라이언트에게 응답할  
HTML 파일 렌더링할 것이므로, 해당 HTML 작성해야지. (화면 구성)
  - *count.html*, *result.html*
5. View는 최종적으로 HTML파일을 클라이언트에게 Response
  - + CSS 적용하기, Git에 push하기 ...

# 장고 실습

글자수세기 페이지 만들기

## 장고 프로젝트 만들기부터~

CODE

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

경로

/session7

```
$ cd CountProject
```

- Pipfile이 있는 위치에서 진행

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

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

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

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

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

# 장고 실습

글자수 세기 페이지 만들기

## settings.py

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

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

# 장고 실습

글자수세기 페이지 만들기

첫번째 페이지를 만들어보자!

글자를 입력하는 페이지 => count

글자수를 세서 결과를 보여주는 페이지 => result

# 장고 실습

글자수 세기 페이지- count

1. URL conf

urls.py

```
from CountApp import views
```

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



💡 'result' url과 CountApp/views의 함수 count를 연결! 💡 template에서 사용할 name은 count로 한다!

# 장고 실습

글자수 세기 페이지- count

2. View

CountProject/CountApp/  
views.py

```
from django.shortcuts import render

# Create your views here.
def count(request):
    # logics here
    return render(request, 'count.html')
```



따로 처리할 로직은 없으므로 html로 렌더링만 해준다!

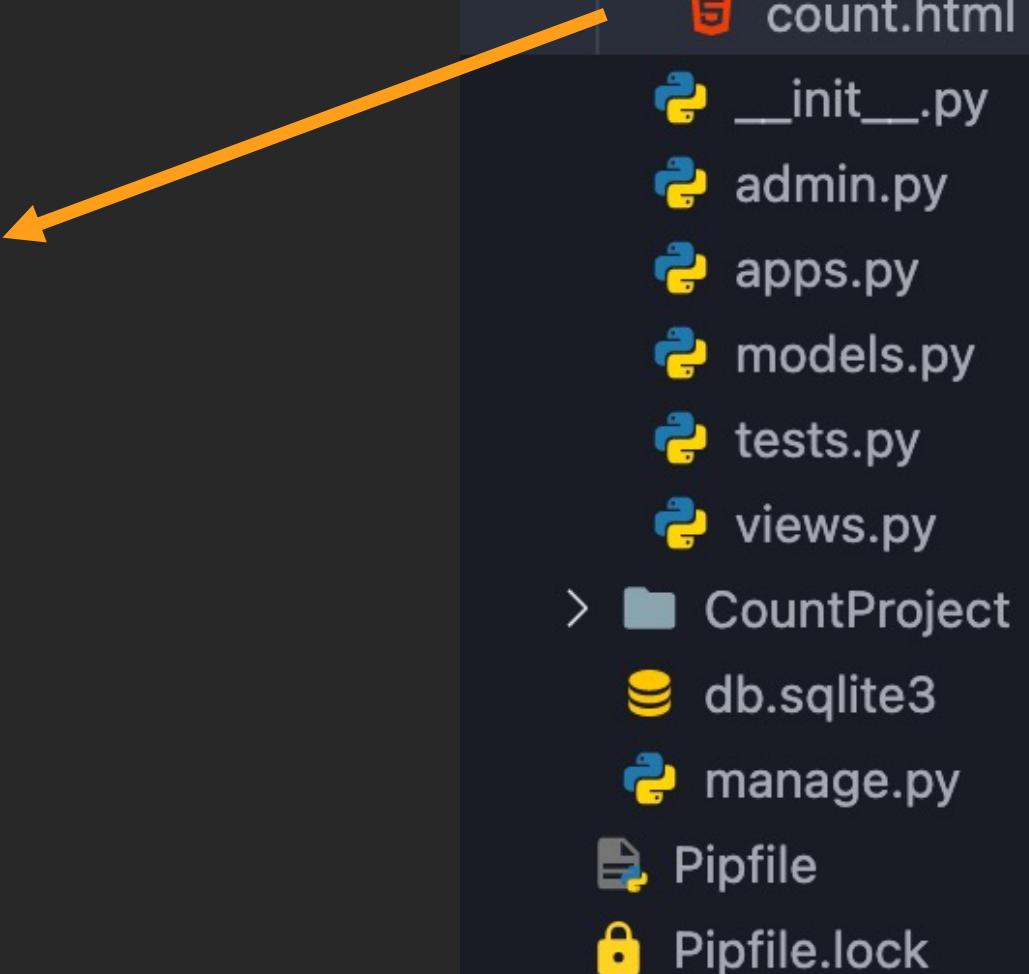
# 장고 실습

글자수 세기 페이지 만들기

## 4. Template

count.html 생성

CountApp/templates/count.html



# 장고 실습

장고 템플릿 문법 알아두기

{% %}

Template 내부에서 사용하는 django 문법

- 나중에 Template 엔진이 해당 값으로 변경해줌

💡 검색어: django template language

csrf\_token

Csrf 해킹 공격 방지

\* http method 가 POST일 때 꼭 적어준다

# 장고 실습

글자수 세기 페이지 - count

4. Template

count.html

```
<body>
    <h1>글자를 입력해 보세요!</h1>
    <form action="{% url 'result' %}" method="post">{% csrf_token %}
        <textarea name="text" cols="30" rows="10"></textarea>
        <br />
        <input type="submit" value="제출" />
    </form>
</body>
```

💡 csrf 공격 방지 - POST일 때 써준다.

💡 urls.py의 urlpattern중 name='result'인 url로 formData전송한다!   💡 http method 중 POST 사용한다

💡 form 태그를 사용해서 formData를 'result' url로 보낸다!

# 장고 실습

글자수 세기 페이지- count

4. Template

count.html

```
<body>
    <h1>글자를 입력해 보세요!</h1>
    <form action="{% url 'result' %}" method="post">{% csrf_token %}
        <textarea name="text" cols="30" rows="10"></textarea>
        <br />
        <input type="submit" value="제출" />
    </form>
</body>
```

# 장고 실습

글자수세기 페이지 만들기

한 페이지만 더 만들면!

글자를 입력하는 페이지 => count

글자수를 세서 결과를 보여주는 페이지 => result

# 장고 실습

글자수 세기 페이지- result

## urls.py

```
from CountApp import views

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



💡 'result' url과 CountApp/views의 함수 result를 연결! 💡 template에서 사용할 name은 result로 한다!

# 장고 실습

글자수 세기 페이지 - result

## views.py

💡 count -> result로 넘어온 request

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

💡 html 태그의 name과 똑같이

💡 html로 렌더링 할 데이터를 딕셔너리 형태로 넘겨준다!

# 장고 실습

글자수 세기 페이지 - result

## result.html

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

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

key값



{{ }} ⚡ views에서 넘겨준 데이터를 html에서 사용할 때

# 장고 실습

직접 해보기

직접 해봅시다!

입력한 텍스트, 공백 제외 글자수 도 추가로 보여주기

💡 hint - 수정할 파일  
views.py  
result.html

# 실습

직접해보기- 정답

## views.py

```
def result(request):
    text = request.POST['text']
    total_len = len(text)
    no_blank_len = len(text.replace(' ', ''))

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

# 장고 실습

직접해보기- 정답

result.html

```
<body>
    <p>입력한 텍스트: {{text}}</p>
    <p>글자수: {{total_len}}</p>
    <p>공백제외 글자수: {{no_blank_len}}</p>
</body>
```

## CSS 적용하기

Static files => 이미지, 자바스크립트, CSS 같은 파일들

Django에서 이러한 정적 파일을 관리하는 기능 제공!

# CSS 적용하기

## settings.py static 관련 세팅

**STATIC\_URL = '/static/'**

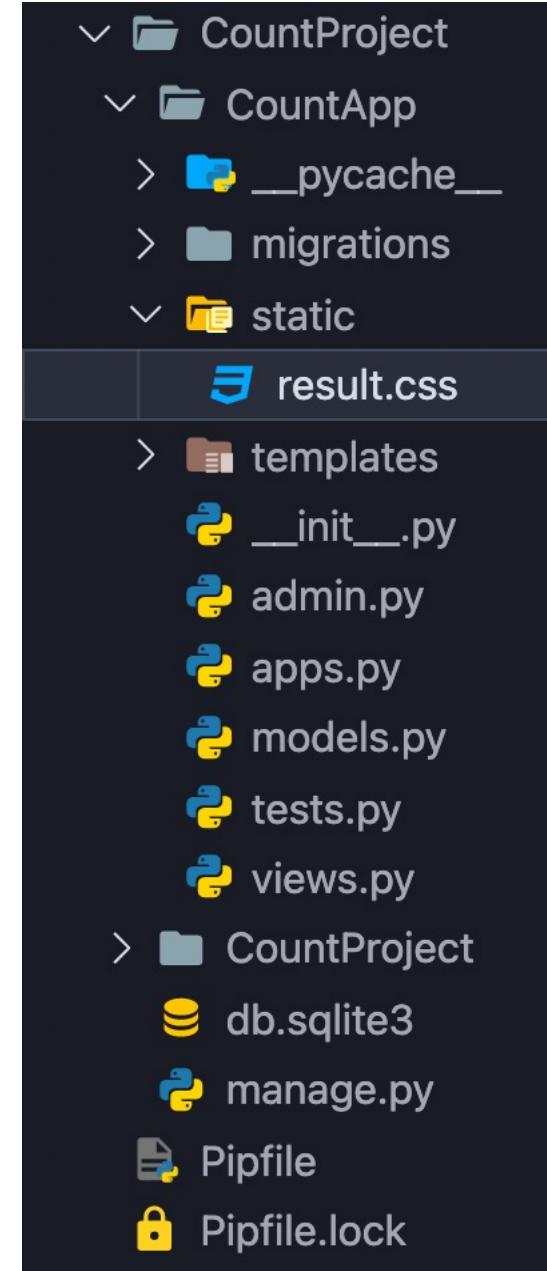
=> Templates와 마찬가지로 앱 디렉토리 하위에 static 디렉토리 인식

- 참고
- STATICFILES\_DIRS**
    - 프로젝트 전반에 사용하는 static 파일들의 디렉토리 목록
  - STATIC\_ROOT**
    - 배포시 흩어져 있는 static file 들 모이는 위치
    - \* `python manage.py collectstatic`

# CSS 적용하기

CountApp/static 폴더 생성

result.css 생성



```
CountProject
├── CountApp
│   ├── __init__.py
│   ├── admin.py
│   ├── apps.py
│   ├── migrations
│   └── static
│       └── result.css
├── db.sqlite3
└── manage.py
```

Pipfile  
Pipfile.lock

# css 적용하기

result.css

```
body {  
    color: salmon;  
}
```

# css 적용하기

html에 css 연결하기

result.html

💡 {% load static %} 꼭 써주기!

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

💡 {% static '<css 파일 이름>' %}

# .gitignore

깃 무시

## 🤔 .gitignore?

### - 목적: 협업을 위해 작성!

.gitignore 파일에 적힌 파일, 폴더는 git 버전관리에서 제외 → github에도 올라가지 않음!

### - 어떤 내용?

\* 개인정보, 비밀번호 등의 공개되면 안되는 정보가 있는 파일, 폴더

\* 각자 개발환경 관련 파일, 폴더

ex) .vscode, migrations, db.sqlite3, \_\_pycache\_\_, DS\_Store

# .gitignore

직접 작성해봅시다

## .gitignore 파일 작성하기

경로

최상단 디렉토리에 **.gitignore** 파일 생성

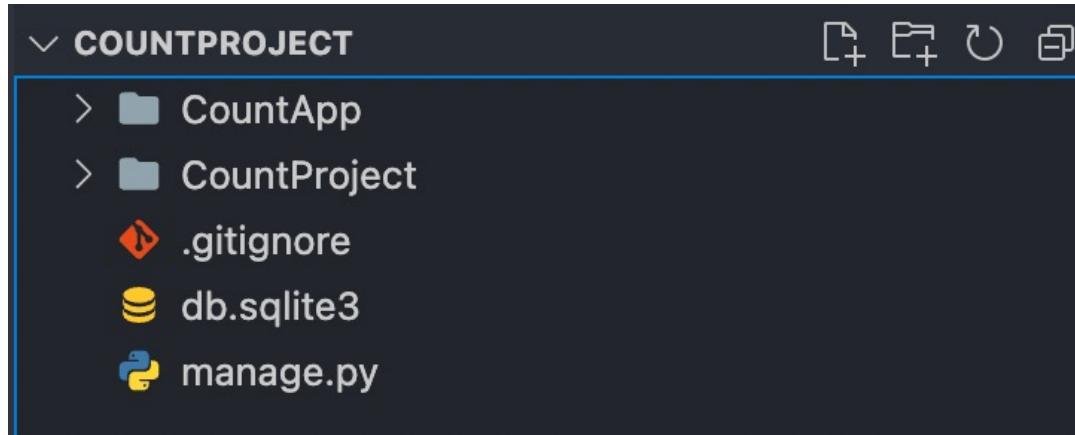
내용

.vscode

migrations

db.sqlite3

\_pycache\_



💡 git으로 관리하고 있지 않다면, 해당 위치에서 git init 실시

💡 지금처럼 이미 git으로 관리하고 있는 디렉토리 내에서 작업했다면,  
git init한 위치가 아닌, 프로젝트가 위치한 최상단 폴더에서 .gitignore 생성

# .gitignore

직접 작성해봅시다

## .gitignore 파일 작성하기

### 주의사항

git add 를 하기 전에 .gitignore를 생성/작성한다.

(순서)

.gitignore 파일 생성 및 작성

→ 해당 파일들이 git에서 무시됨

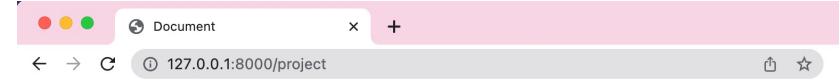
→ \$ git add .

# 과제

할 수 있다!

## 1. 글자수 세기 프로젝트:

- 단어수 세는 기능 추가
- (선택) count.html, result.html CSS로 꾸미기



INFO PROJECT BLOG

여기에 클론 코딩 과제 중 마음에 드시는 걸 붙여주세요 😊

## 2. ☆개인 포트폴리오 사이트☆

- 장고 프로젝트 생성 (이름 예시: portfolioProject)
- 총 2페이지 만들기(info 페이지, project 페이지)
  - : info 페이지에서 <a>태그를 통해 project page로 접근되도록 구현
- CSS로 꾸미기(디자인 레퍼런스 ‘반드시’ 참고)



INFO PROJECT BLOG

안녕하세요, 김나영입니다

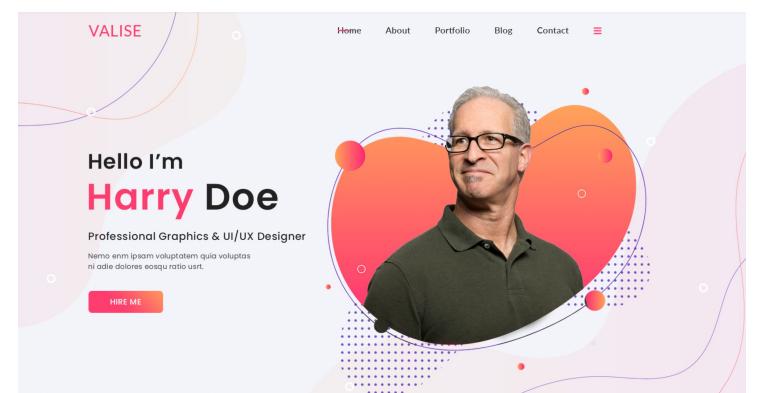
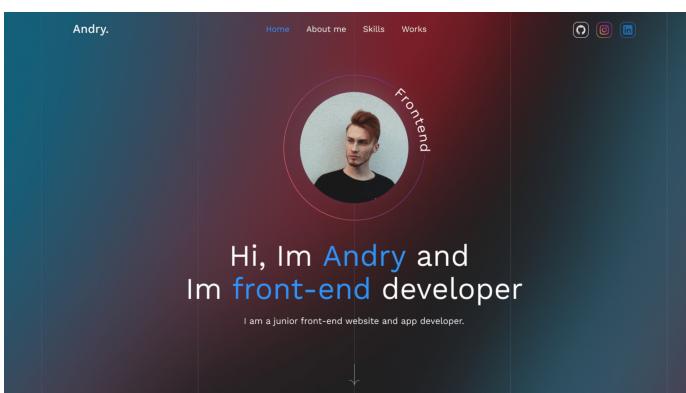
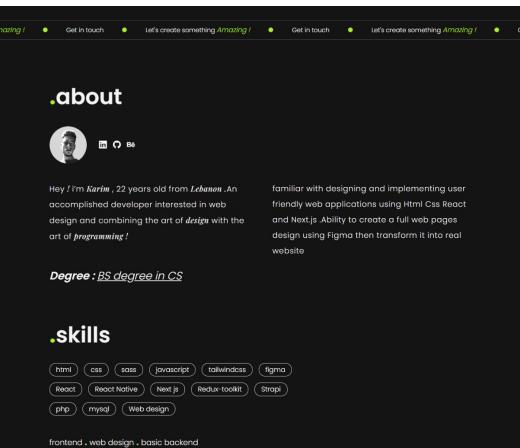
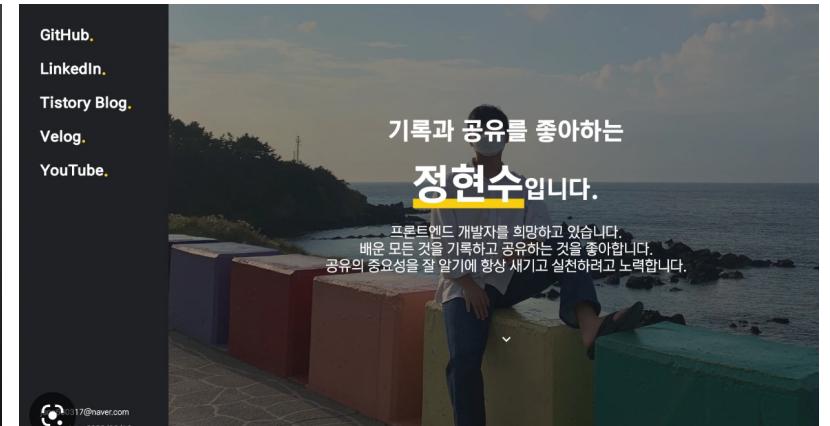
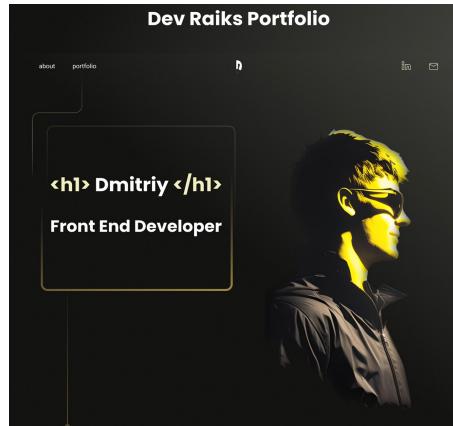
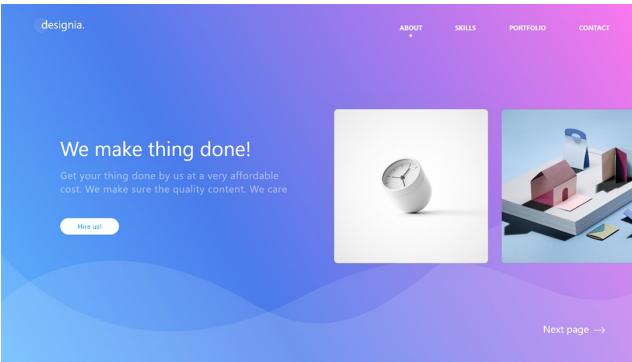
Web Developer  
and Contents Creator

저는 CSS를 거의 안 했지만 여러분은 꼭 잘 하시길 바랍니다ㅎㅎ

# 과제

할 수 있다!

## 2. ★개인 포트폴리오 사이트★ 레퍼런스 예시



## 제출 방법

### 1. 글자수 세기 프로젝트

- 본인 레포지토리에 git push한 링크
- runserver하여 단어수 세는 기능 화면 녹화

### 2. ☆개인 포트폴리오 사이트☆

- 본인 레포지토리에 git push한 링크
- runserver하여 화면 녹화

제출 기한 다음 세션(2023/03/30/목)