



# 웹 프로그래밍

Django 웹 프레임워크 - Fullstack

# 시간계획

- 오늘도 파이팅 입니다.^^

시간	학습내용
09:00~10:00	Frond-End web pg 마무리
10:20~11:20	장고 프레임워크 실습 환경 구축
11:40~12:40	장고 설치, 기본DB 생성
12:40~14:00	즐거운 점심 시간
14:00~15:00	Dbear 설치하여 DB 확인하기
15:20~16:20	Django 프레임워크 이해
16:40~17:50	투표앱 기본 틀 만들기

# 머신러닝 기반 데이터 분석, 예측 파트 진행 순서



## 1) 분석 및 예측

시각화, 머신러닝:

- python
- numpy, Pandas
- Matplotlib, Seaborn
- 기초통계
- Scikit learn

- 탐색적 데이터 분석 방법으로 데이터를 분석함  
- 분석 데이터를 시각화 하는 방법을 익힘  
- 머신러닝 이해 하고 사이킷런 활용해 다양한 머신러닝 모델을 만들고 평가하는 방법 적용



## 2) Web pgm 기본

Front end side :

- HTML5
- CSS3
- Javascript
- jQuery

- 웹에 산재되어 있는 데이터를 수집, 분석하기 위한 웹 문서 표현 기술인 웹 표준 활용 능력 익힘.  
- 웹 데이터의 구조 이해



## 3) 데이터 저장

Back end side:

- Django : python 기반 web server 프레임워크
- Mysql - CRUD
- MongoDB(js기반)

- 데이터를 구조화 하여 저장하는 방법 적용  
- 정형, 비정형 데이터 유형을 이해하고, 저장하는 방법 적용  
- 클라우드 서비스 이해, 프리티어 서비스를 활용해 웹서비스 구현



## 4) 데이터 수집가공

웹 크롤링 & 스크래핑:

- Python 기반
- BeautifulSoup
- Selenium
- 머신러닝 통합 예제
- Linux shell pg

- 웹크롤링 및 스크래핑 기술을 적용  
- 웹에 산재되어 있는 데이터를 수집, 가공, 파일로 저장하는 방법 활용  
- 데이터 분석을 위해 전처리 방법을 익힘



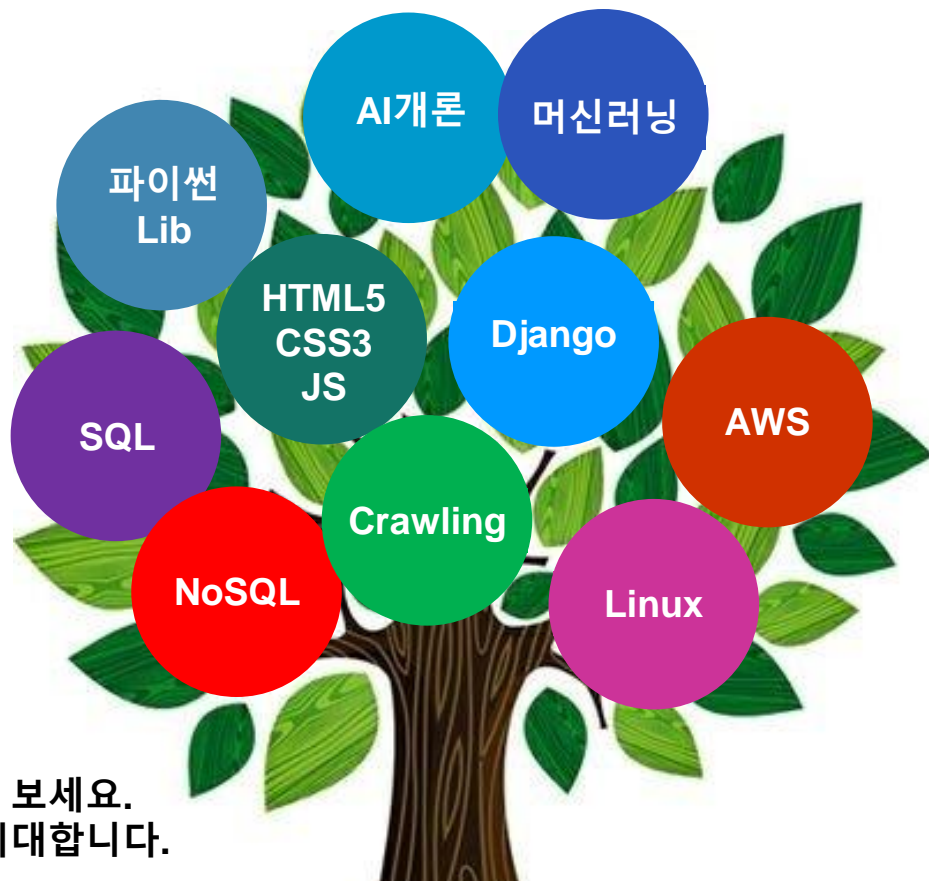
## 5) 팀 협업 프로젝트



- 의미 있는 도출을 위한 팀 주제 정하기
- 웹크롤링, 오픈데이터
- 데이터 DB 저장
- 데이터 분석, 시각화
- 머신러닝 예측
- 웹 서비스로 구현하기

클라우드 서비스  
AWS

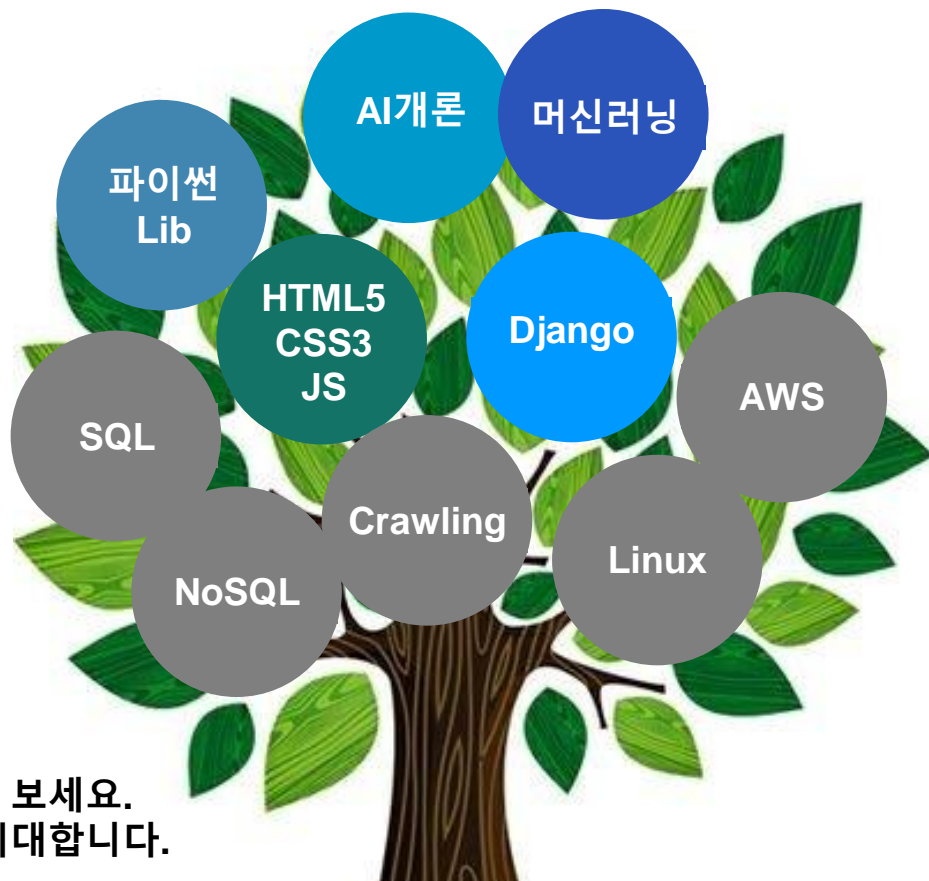
# 데이터 분석 강의 내용(10.5~11.16)



어떤 씨앗을 심을지 고민해 보세요.  
모두 좋을 결실이 있기를 기대합니다.



# 데이터 분석 강의 내용(10.5~11.16)



어떤 씨앗을 심을지 고민해 보세요.  
모두 좋을 결실이 있기를 기대합니다.



## Full Stack 개발

### Front End (Client side)

- 웹 표준 소개
- HTML5, CSS3
- JavaScript
- jQuery, ajax

### Back End (Server side)

- Django(파이썬 기반)
- SQLite
- DB(mysql)
- mongoDB

# TECH SPEC

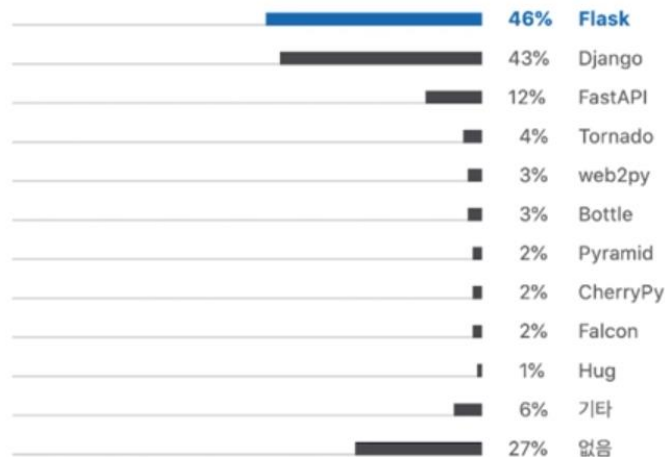


- Django 3.2
- python 3.8
- VS code
- OS : Ubuntu Linux

# Django 소개

## 프레임워크 및 라이브러리

웹 프레임워크 > 100%



- Batteries included
- 다양한 사용이 가능
- 안전성
- Shared-nothing Architecture
- Very Maintainable



# 레퍼런스 사이트



- Pinterest
- Bitbucket
- Udemy
- Disqus
- Washington Post
- NASA
- Google
- Spotify
- InstaGram
- Reddit
- Youtube
- National Geography
- Toss
- Delivery Hero Korea
- CoinBit
- 숨고

# 프레임워크 vs 라이브러리

프레임워크



라이브러리

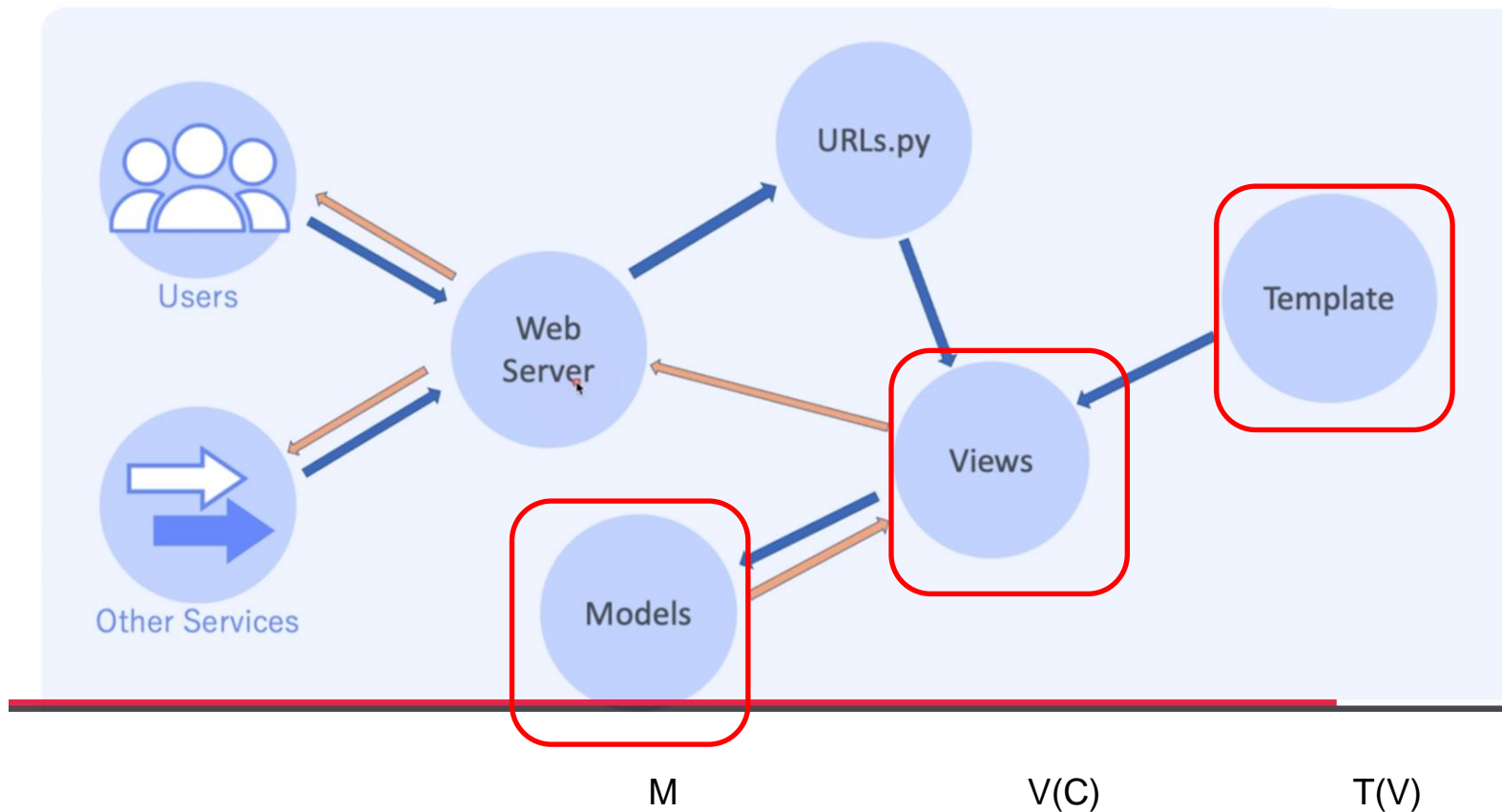


# Django의 탄생

---

- 모 신문사의 웹 팀에서 2003년부터 개발
- 초창기는 공통 모듈, 코드, 패턴을 뽑아내 재사용하는 형태에서 일반적인 웹프레임워크로 진화
- 2005년 7월 오픈소스로 공개, 2008년 9월 v1.0 배포

# Django의 구조



# Coding Convention

여러 사람이 협업을 해도 모두가 읽기 편한 코드를  
작성하기 위한 기본 규칙

- 한 줄의 문자열은 79자
- DocString은 72자 (""" """)
- aaa\_bbb 사용
- 모듈 레벨 상수는 모두 대문자
- ClassName은 대문자로 시작
- 한 줄로 된 구분은 사용하지 않음.
  - If, try... except, for, while 등

Django에서는  
한줄 문자열 119자  
추천

DocString 72자

Local Rule 중요

---

# Django 프로젝트 환경설정

# 가상환경 만들기

```
$ python3.8 --version  #파이썬 버전 확인 3.8.8
$ pwd                  #현재 디렉토리 확인
/home/rapa00           #rapa00 사용자 계정
$ mkdir django
$ cd Django
$ python3.8 -m venv VENV  #VENV 가상환경 만들기
또는 $ virtualenv --python=python3.8 VENV
$ pwd
/home/rapa00/django
$ ls
VENV
```

# 가상환경 만들기

- Django 가상환경 alias 설정

```
$nano .bashrc
```

[추가]

```
alias dj='source /home/rapa00/django/VENV/bin/activate; cd /home/cozlab/django'
```

```
$dj
```



# Django 설치 및 확인

- Django 설치 및 버전확인

(VENV) \$python3.8 -m pip install django (또는)

(VENV) \$python3.8 -m pip install django==3.2 #장고 버전 지정

(VENV) \$python3.8 -m django -version # 장고 설치 버전 확인

—

# 기본단계

# Django 프로젝트 생성

- Django 프로젝트 실행

(VENV) \$ django-admin startproject rapa

(VENV) \$

(VENV) :~/django/rapa\$ cd rapa

(VENV) :~/django/rapa\$ pwd

/home/rapa00/django/rapa

(VENV) :~/django/rapa\$ ls

```
(VENV) cozlab@cozlab:~/django/rapa$ ls  
manage.py  rapa  
(VENV) cozlab@cozlab:~/django/rapa$
```

# 앱 만들기 - 앱 생성

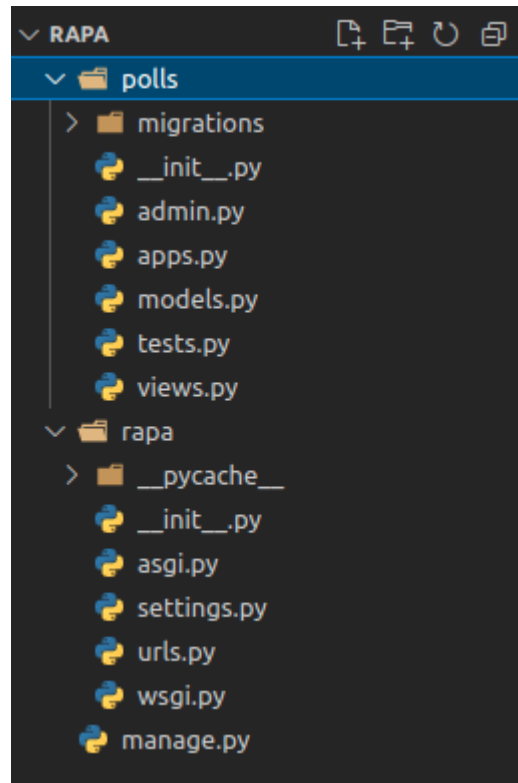
- 앱 생성 명령

\$python3.8 manage.py startapp polls

- 예시

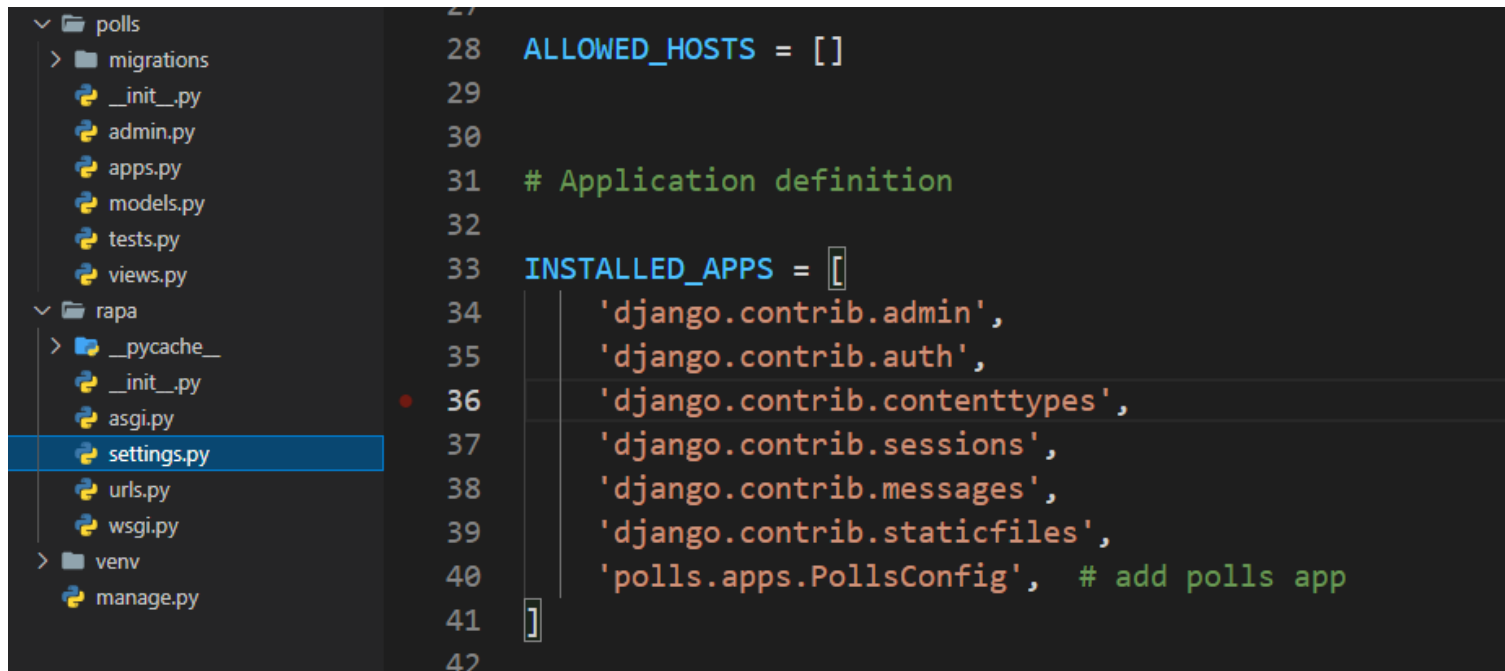
```
(VENV) cozlab@cozlab:~/django/rapa$ python3.8 manage.py startapp polls
(VENV) cozlab@cozlab:~/django/rapa$ ls
manage.py  polls  rapa
(VENV) cozlab@cozlab:~/django/rapa$
```

\$ code .



# 앱 만들기 - 기본 셋팅하기

- rapa/setting.py - 생성한 앱 등록 하기



```
27
28 ALLOWED_HOSTS = []
29
30
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     'polls.apps.PollsConfig', # add polls app
41 ]
42
```

# 앱 만들기 - 기본 셋팅하기



- rapa/setting.py 기타 수정하기

```
ALLOWED_HOSTS = ['localhost', '127.0.0.1']
```

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

```
LANGUAGE_CODE = 'ko-kr'
```

# 앱 만들기 - 기본 셋팅하기

- django 프레임워크, 기본 테이블 생성
- (venv) django/rapa\$ **python3.8 manage.py migrate**

```
(VENV) cozlab@cozlab:~/django/rapa$ ls
manage.py  polls  rapa
(VENV) cozlab@cozlab:~/django/rapa$ python3.8 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
(VENV) cozlab@cozlab:~/django/rapa$
```

# 앱 만들기 - 서버 실행 및 확인

- 장고 서버 실행

**Python3.8 manage.py runserver 0.0.0.0:8000**

- 장고 앱 확인하기(크롬 브라우저에서)

127.0.0.1:8000

127.0.0.1:8000/admin





# 앱 만들기 – superuser 등록

- Python3.8 manage.py createsuperuser

```
(VENV) cozlab@cozlab:~/django/rapa$ python3.8 manage.py createsuperuser
사용자 이름 (leave blank to use 'cozlab'): admin
이메일 주소: magpia000@daum.net
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]: n
Password:
Password (again):
비밀번호가 사용자 이름과 너무 유사합니다.
Bypass password validation and create user anyway? [y/N]: n
Password:
Password (again):
Superuser created successfully.
(VENV) cozlab@cozlab:~/django/rapa$
```

- Id : admin
- Pw : 비밀번호 8자리 이상

# 앱 만들기 - admin 로그인

- 로그인 확인

Django 관리

사용자 이름:

admin

비밀번호:

.....

로그인

← → ↺ 127.0.0.1:8000/admin/

Django 관리

사이트 관리

인증 및 권한

그룹

사용자(들)

+ 추가

✎ 변경

+ 추가

✎ 변경

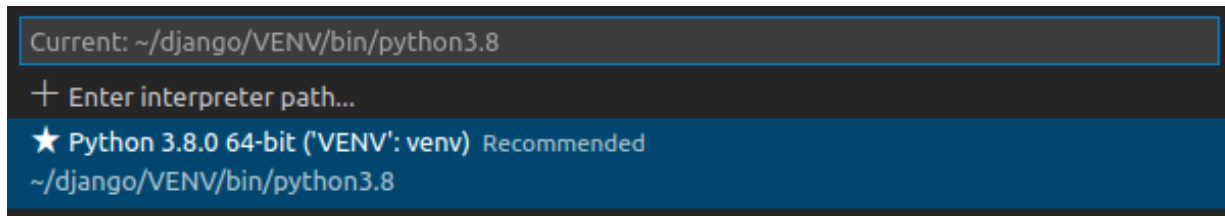
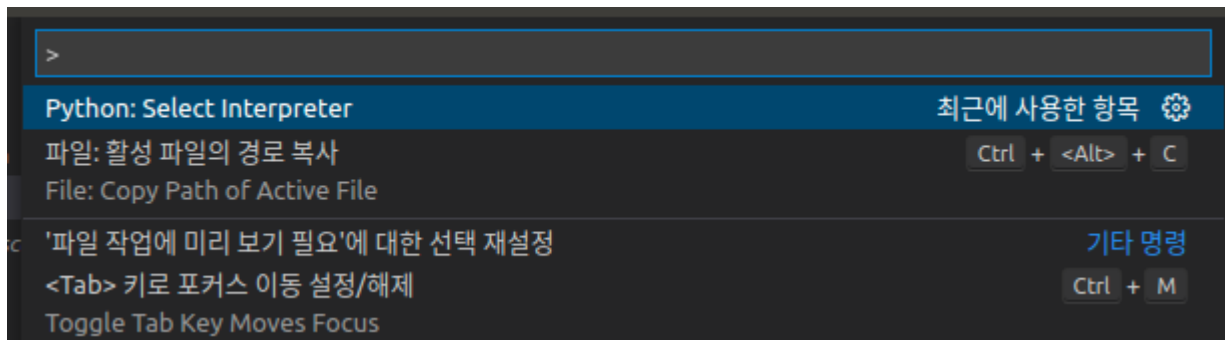
최근 활동

나의 활동

이용할 수 없습니다.

# VS code / 가상환경 python path 설정

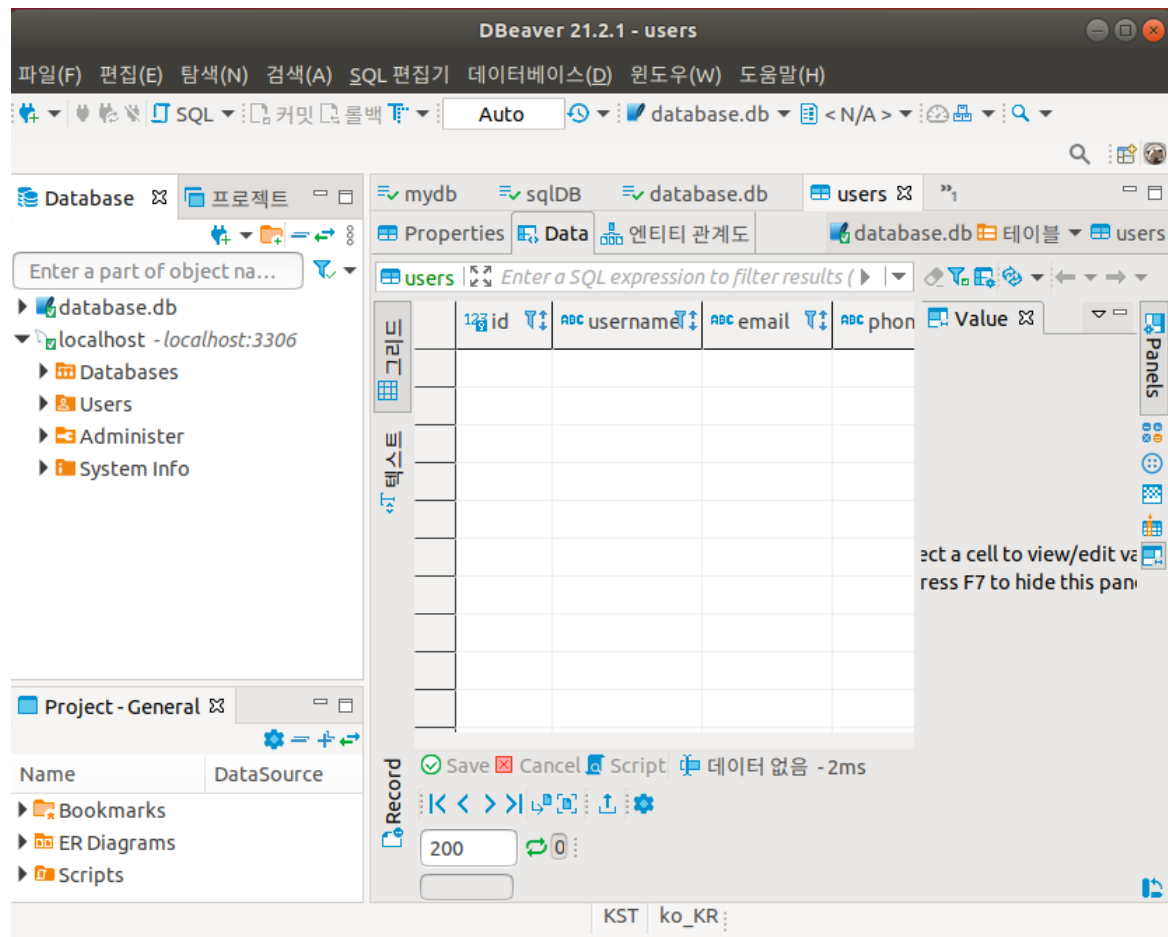
- `pwd => /home/cozlab/django/rapa`  
`$ code .`
- `ctrl + shift + p`



---

# Dbever 설치

# Dbeaver 설치



## Dbeaver 설치

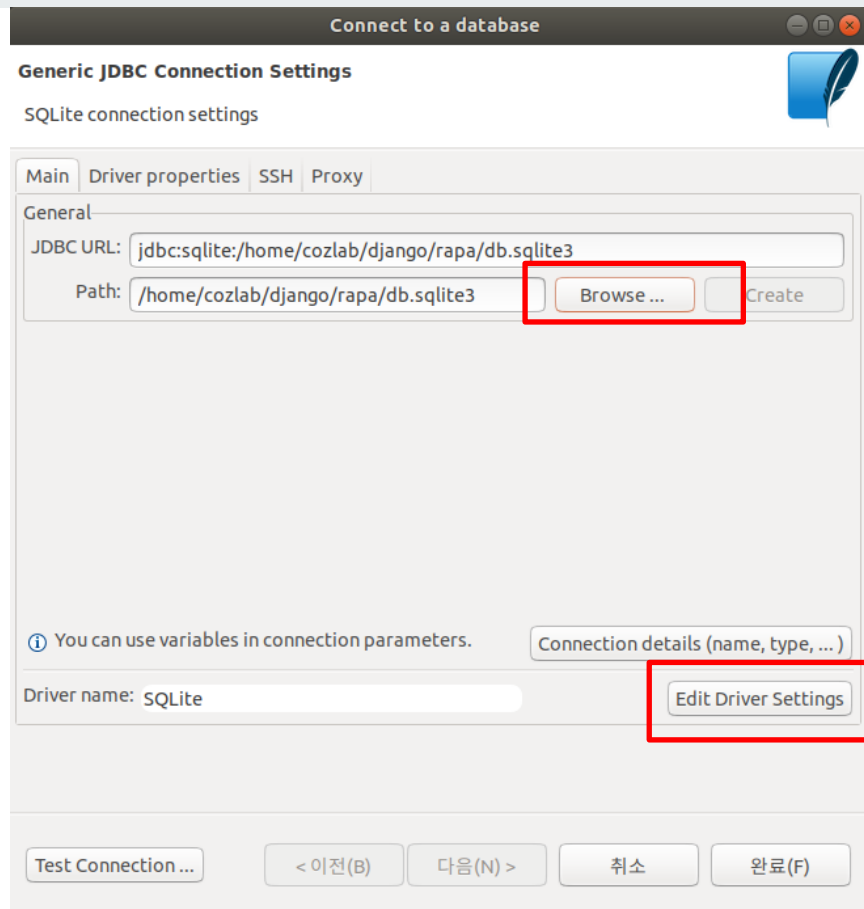
- [다운로드 : dbeaver.io/download/](https://dbeaver.io/download/)

- 설치

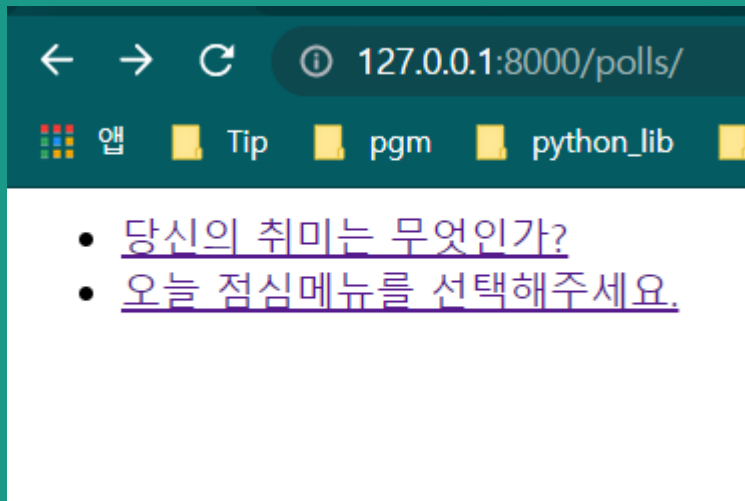
```
sudo dpkg -i dbeaver_download_file
```

# Dbeaver 설치

1. 아이콘 확인, 실행
2. DB 선택하기 :  
SQLite 아이콘 선택



# 앱 개발





# 앱 개발 workflow



1. Model 설계
2. Model 모델 정의 및 admin 사이트에 반영
3. DB에 변경사항 반영
4. 작업 결과 확인 및 테스트
5. View 및 Template(HTML) 코딩

# 앱 개발 - 모델 설계

- Question 테이블 컬럼과 클래스 변수 간의 맵

필드명	타입	장고 클래스 변수	장고의 필드 클래스
id	Integer	id	PK는 장고에서 자동생성
question_text	CharField(200)	question_text	models.CharField(max_length=200)
pub_data	Datetime	pub_date	models.DateTimeField(date published')

# 앱 개발 - 모델 설계

- Choice 테이블 컬럼과 클래스 변수 간의 맵

필드명	타입	장고 클래스 변수	장고의 필드 클래스
id	Integer	id	PK는 장고에서 자동생성
choice_text	CharField(200)	choice_text	models.CharField(max_length=200)
votes	Integer	votes	models.IntegerField(default=0)
question_id	Integer	question	models.ForeignKey(Question)

# 앱 개발 - 모델 테이블 정의

- polls/models.py

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

```
# Create your models here.
```

```
class Question(models.Model):
```

```
    question_text = models.CharField(max_length=200)
```

```
    pub_date = models.DateTimeField('date published')
```

```
    def __str__(self):
```

```
        return self.question_text
```

```
class Choice(models.Model):
```

```
    question = models.ForeignKey(Question, on_delete=models.CASCADE)
```

```
    choice_text = models.CharField(max_length=200)
```

```
    votes = models.IntegerField(default=0)
```

```
    def __str__(self):
```

```
        return self.choice_text
```

# 앱 개발 – 모델 admin 사이트에 반영

- polls/admin.py

```
from django.contrib import admin
from polls.models import Question, Choice
```

```
# Register your models here.
admin.site.register(Question)
admin.site.register(Choice)
```

# 앱 개발 – 모델 DB에 반영

- 테이블의 신규 생성 및 변경사항 반영

python manage.py makemigrations

python manage.py migrate

```
(VENV) cozlab@cozlab:~/django/rapa$ ls
db.sqlite3  manage.py  polls  rapa
(VENV) cozlab@cozlab:~/django/rapa$ python manage.py makemigrations
Migrations for 'polls':
  polls/migrations/0001_initial.py
    - Create model Question
    - Create model Choice
(VENV) cozlab@cozlab:~/django/rapa$ python3.8 manage.py migrate
Operations to perform:
  Apply all migrations: admin, auth, contenttypes, polls, sessions
Running migrations:
  Applying polls.0001_initial... OK
(VENV) cozlab@cozlab:~/django/rapa$
```

## 앱 개발 – 모델 DB 적용 확인

- DB 반영 sql문 확인(migrate 명령에 의해 실행된)

**python manage.py sqlmigrate polls 0001**

- 모든 마이그레이션을 보여주고, 각 마이그레이션 별 적용 여부 확인 명령

**python manage.py showmigrations**

# 앱 개발 - 모델 DB에 반영 문제 해결

- 롤백하기

형식 : `python manage.py migrate <app-name> <마이그레이션 파일명>`

**`python manage.py migrate shortener 0001`**

```
(venv) D:\django\shinkers>python manage.py makemigrations
Migrations for 'shortener':
  shortener\migrations\0001_initial.py
    - Create model PayPlan
    - Create model Users

(venv) D:\django\shinkers>python manage.py migrate
Operations to perform:
  Apply all migrations: admin, auth, contenttypes, sessions, shortener
Running migrations:
  No migrations to apply.
```

- 오류 발생시 문제 해결 방법

DB 유저의 생성 테이블과 migrations make파일 삭제 후 다시 시작



# 앱 개발 - 확인 및 테스트

- runserver 실행 하여 확인하기
- python manage.py runserver
- http://127.0.0.1:8000/admin

## Django 관리

홈 > Polls

### Polls 관리

#### POLLS

Choices

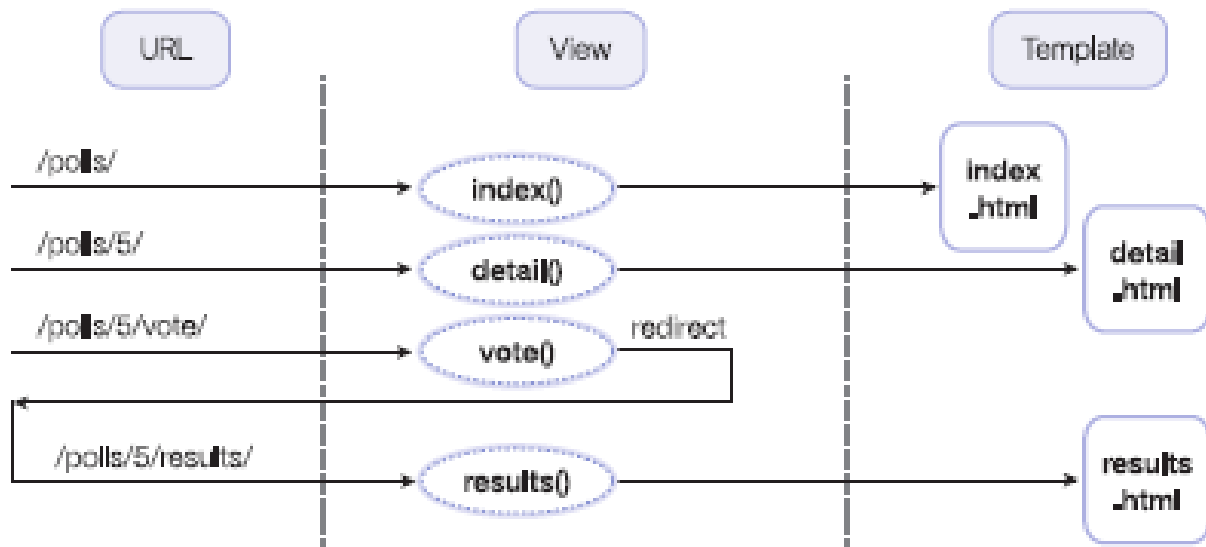
+ 추가    ✎ 변경

Questions

+ 추가    ✎ 변경

# 앱 개발 – View 및 Template 코딩

- 보여주기 위해 필요한 뷰와 템플릿 코딩
- poll 앱의 처리 흐름



# 앱 개발 – URLconf 설계

- URL과 View 맵
- URL과 View 매핑을 URLconf 라고 함.(urls.py)
- 일반적으로 URL/View 1:1, N:1도 가능

URL 패턴	뷰 이름	뷰가 처리하는 내용
/poll	index()	index.html 템플릿 보여주기
/polls/5/	detail()	detail.html 템플릿 보여주기
/polls/5/vote/	vote()	detail.html에 있는 폼을 POST 방식으로 처리
/polls/5/results/	results()	result.html 템플릿 보여주기
/admin/	(장고 자동)	Admin 사이트를 보여줌.(장고에서 기본 제공)

# 앱 개발 – URLconf 코딩

로직의 흐름상 URLconf를 먼저 코딩 후에 뷰, 템플릿 코딩하는 게 일반적임

- `urls.py` // URLconf 내용 코딩
- `views.py` // `index.html` 템플릿 작성
- `views.detail()` // `detail.html` 템플릿 작성
- `views.vote()` // 리다이렉션 처리
- `views.results` // `results.html` 템플릿 작성

# 앱 개발 – URLconf 코딩

- rapa/rapa/urls.py

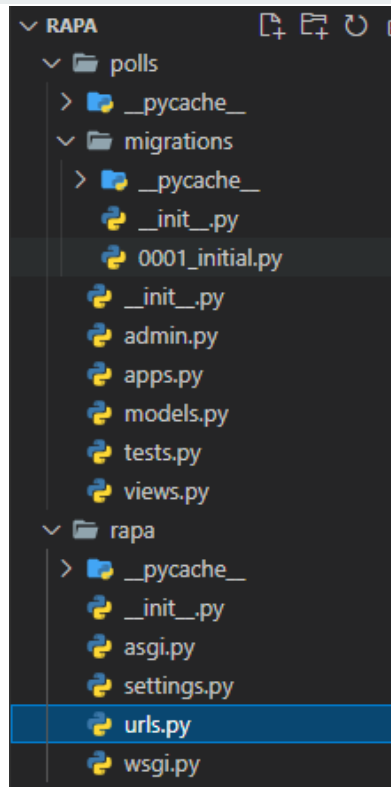
```
from django.contrib import admin
from django.urls import path, include
```

```
urlpatterns = [
    path('admin/', admin.site.urls),
```

```
    # add app
```

```
    path('polls/', include('polls.urls')),
```

```
]
```



# 앱 개발 – URLconf 코딩

- rapa/polls/urls.py 파일 생성, url 패턴 정의

```
from django.urls import path
from polls import views
```

```
app_name = 'polls'
urlpatterns = [
    # /polls/
    path('', views.index, name='index'),
    # /polls/5/
    path('<int:question_id>/', views.detail, name='detail'),
    # /polls/5/results/
    path('<int:question_id>/results/', views.results, name='results'),
    # /polls/5/vote/
    path('<int:question_id>/vote/', views.vote, name='vote'),
]
```

# 앱 개발 - 뷰 함수 및 Template 작성

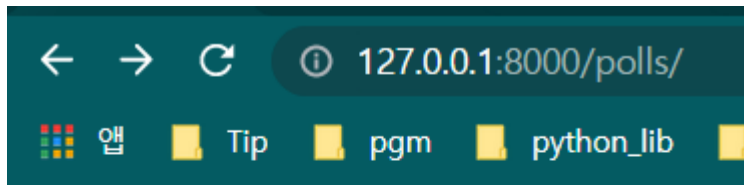


1. views.py 와 index.html
2. views.py detail() 폼과 detail.html
3. views.py vote() 및 리다이렉션 작성
4. views.py results() 및 results.html 작성

실습 파일 별도 제공함.

# 앱 개발 - 확인 및 테스트

- runserver 실행하여 확인하기  
python manage.py runserver  
http://127.0.0.1:8000/polls



- 당신의 취미는 무엇인가?
- 오늘 점심메뉴를 선택해주세요.



---

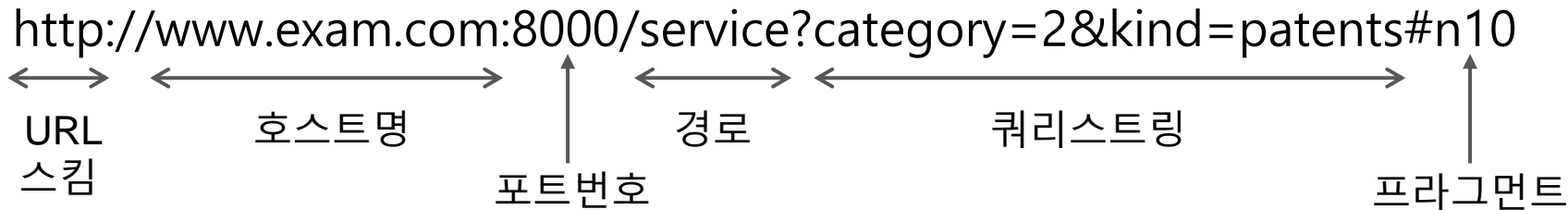
# 웹 서비스 이해

# http 응답방식(http 상태 코드 레지스트리)

구분	설명
1xx	정보제공(Information) 요청을 받았으며 프로세스를 계속 진행
2xx	성공(Success) 요청을 성공적으로 받았으며 인식했고 수용하였음
3xx	리다이렉션(Redirection) 서버의 주소 또는 요청 URL의 웹문서가 이동됨 표시
4xx	클라이언트 오류(Client Error) 요청의 문법이 잘못되었거나 요청을 처리할 수 없음
5xx	서버 오류(Server Error) 서버 문제 발생(서버부하, DB처리, 서버 예외상황 발생)

- 참고 : <https://developer.mozilla.org/ko/docs/Web/HTTP/Status>

# URL 이해



구분	설명
URL 스킴	URL에 사용된 프로토콜 의미함
호스트명	웹 서버의 호스트명으로 도메인명 또는 IP 주소로 표현됨
포트번호	웹 서버 내의 서비스 포트번호(http 80, https 443)
경로	파일이나 애플리케이션 경로를 의미함
쿼리스트링	질의 문자열로, (&)로 구분된 <b>이름=값</b> 쌍 형식으로 표현
프래그먼트	문서 내의 앵커 등 조각을 지정함.

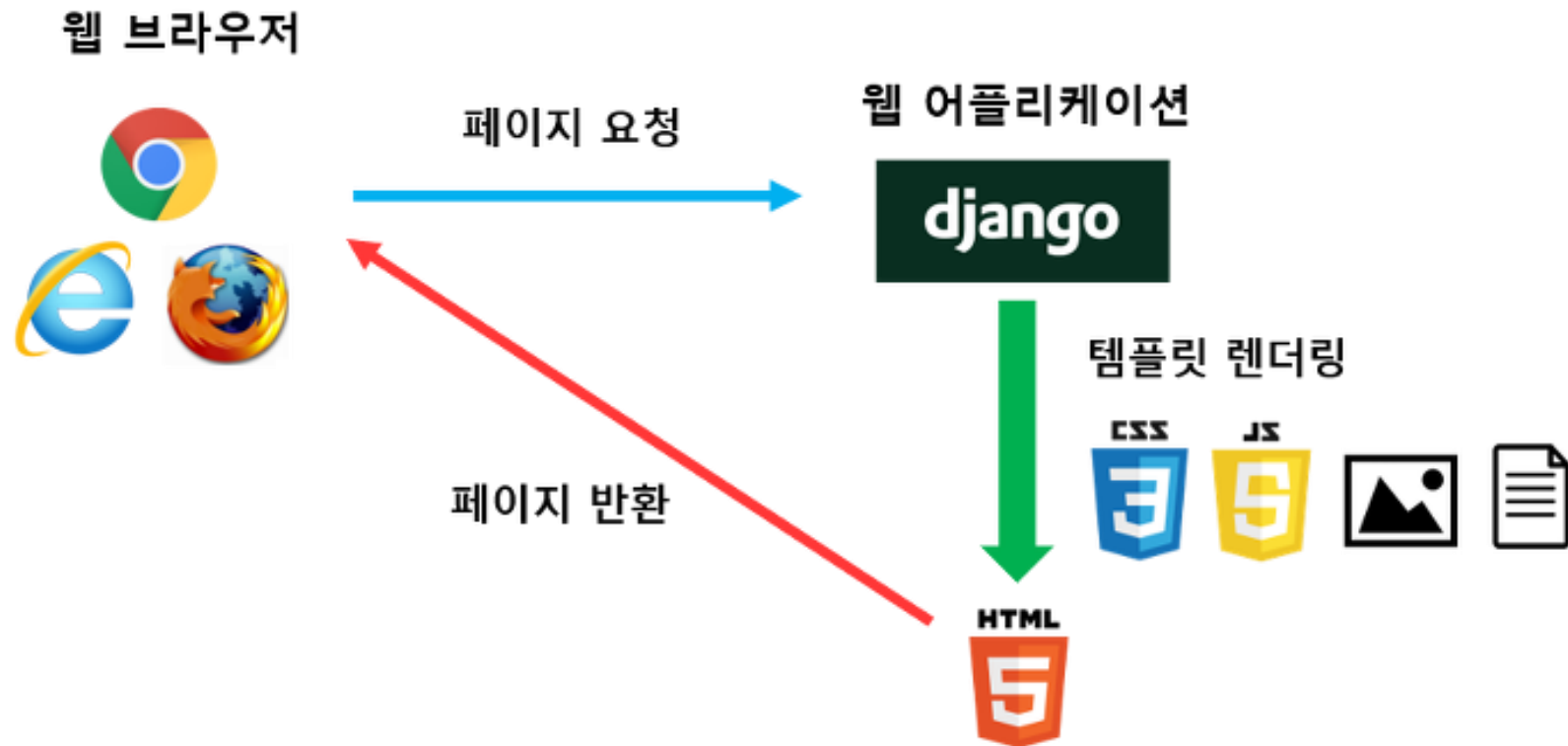
# 간편 URL

- 기존 URL방식에서 사용한 문자(?, -, &, #)들을 간단한 경로구조로 변경하여 사용
- 검색 엔진의 처리를 최적화 하기 위해 생겨남
- 검색엔진 친화적 URL, 사용자 친화적인 URL , 우아한 URL

기존 URL	간편 URL
<a href="http://example.com/products?category=2&amp;pid=25">http://example.com/products?category=2&amp;pid=25</a>	<a href="http://example.com/products/2/25">http://example.com/products/2/25</a>
<a href="http://example.com/services/index.jsp?category=legal&amp;id=patents">http://example.com/services/index.jsp?category=legal&amp;id=patents</a>	<a href="http://example.com/services/legal/patents">http://example.com/services/legal/patents</a>

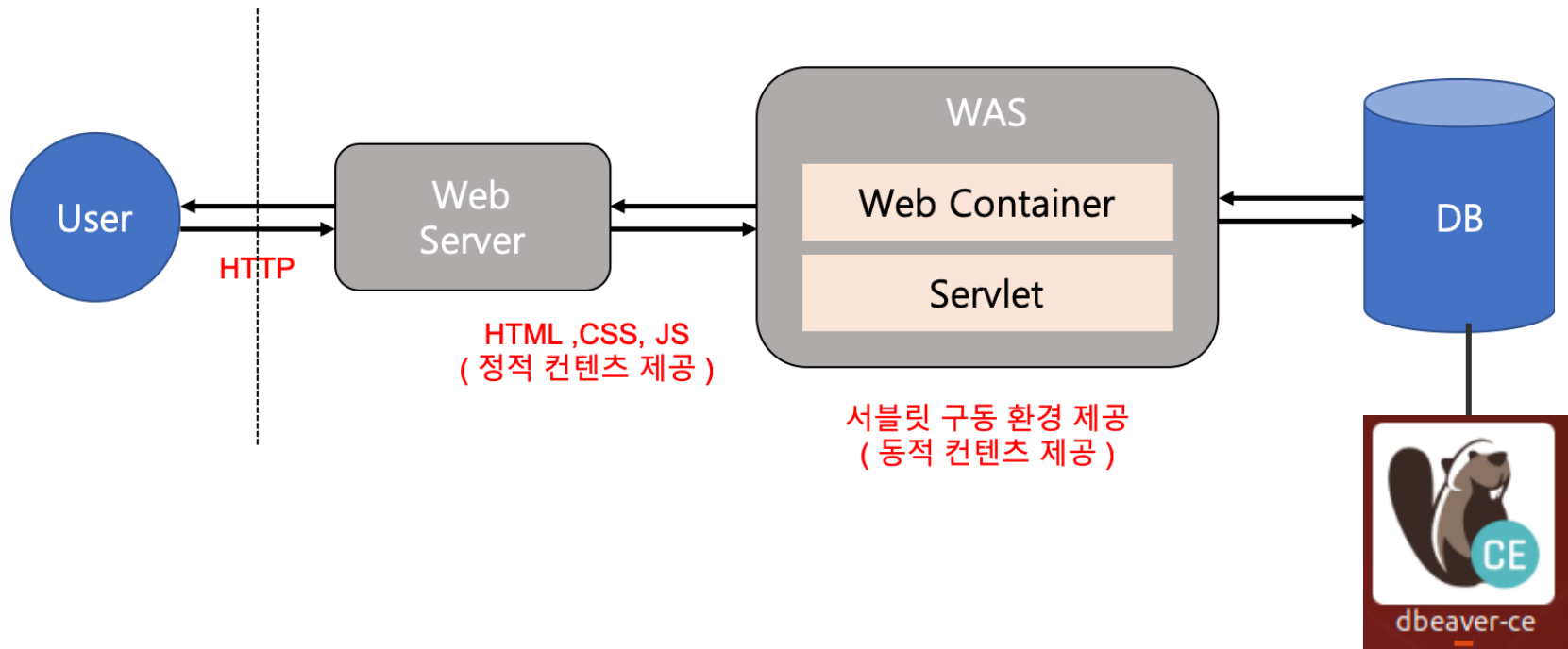
# 장고 웹 서버 구조 이해

- 개발 환경에서의 구조

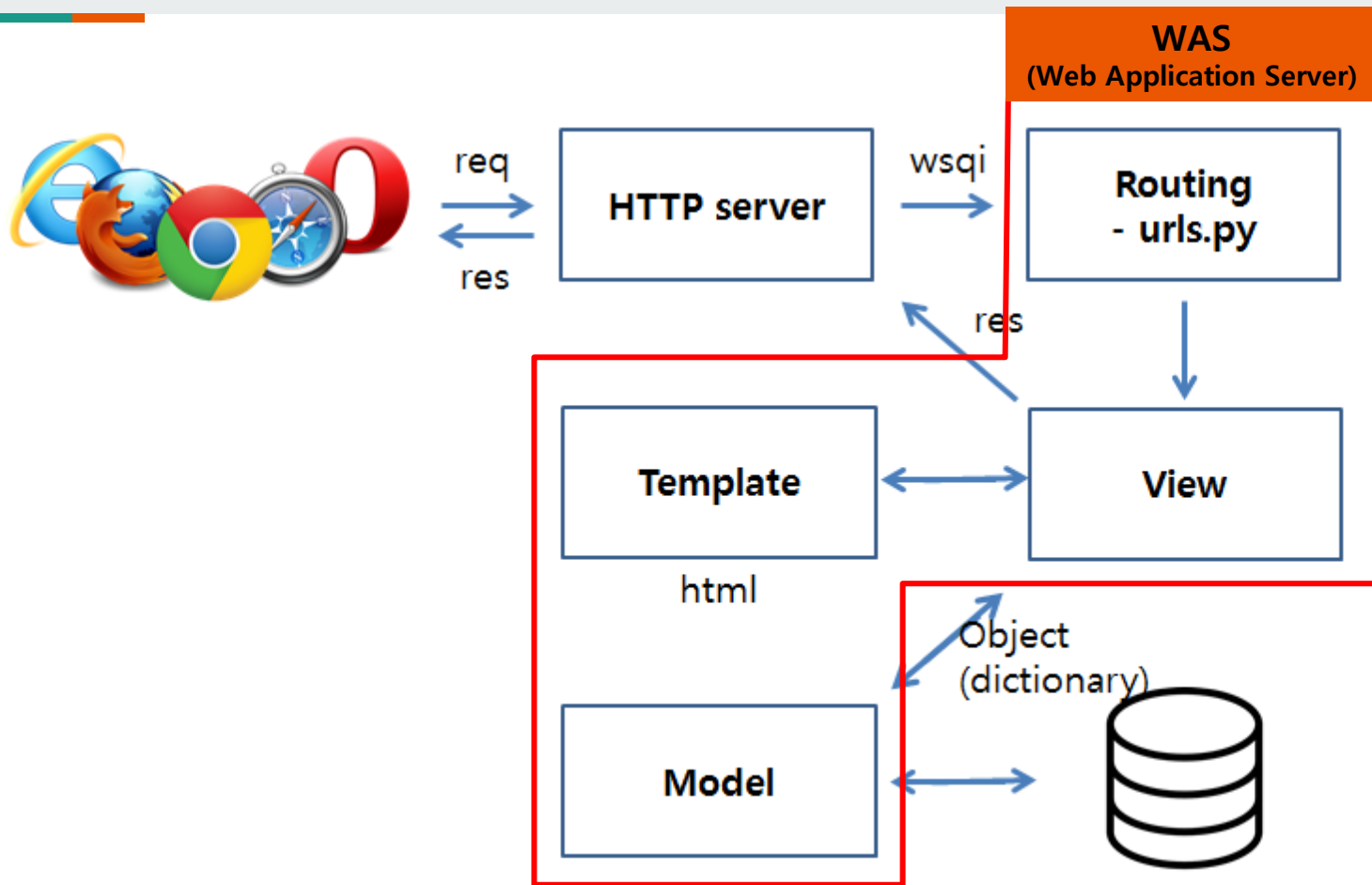


# 웹 서비스 구조 이해

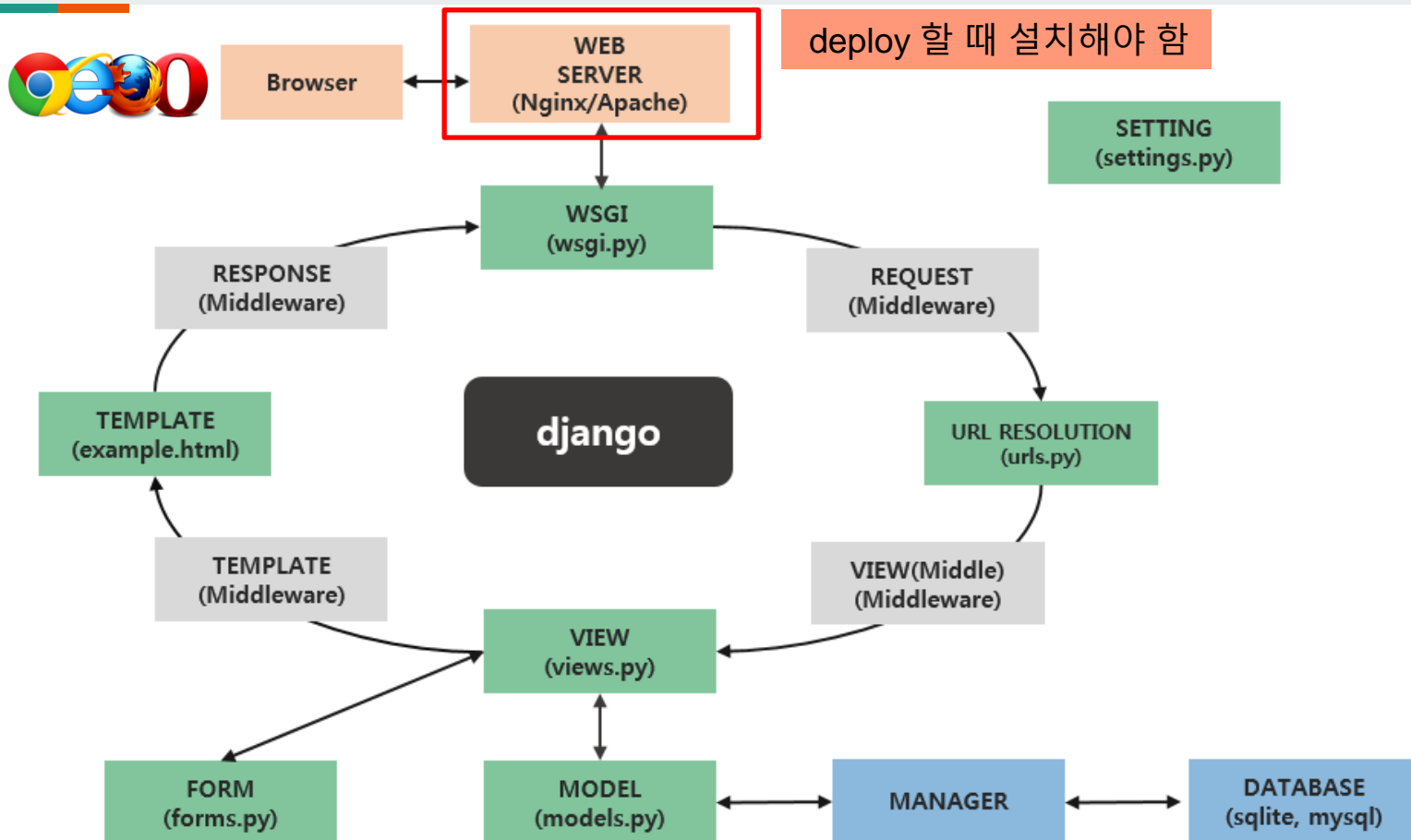
- 일반적인 웹서버 구조



# 장고 웹 서버 구조 이해



# 장고 웹 서버 구조 이해(설정 파일의 역할)

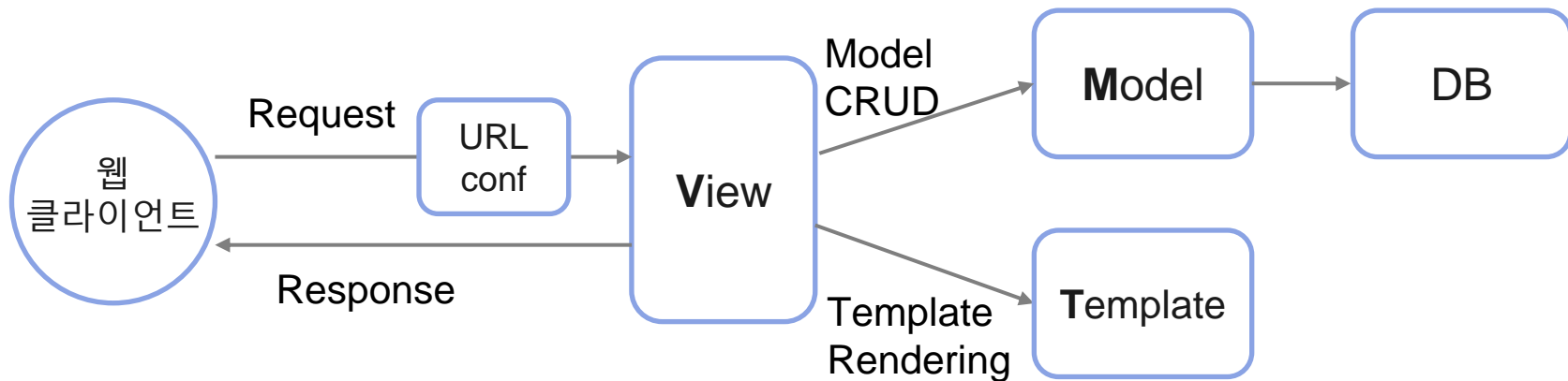




---

# Django 앱 개발 방식

# 장고의 MTV 패턴 정리



MTV모델	MVC모델	설명
<b>M</b> odel	<b>M</b> odel	DB에 저장되는 데이터 관리(CRUD)
<b>T</b> emplate	<b>V</b> iew	사용자에게 보여지는 페이지 만들기 수행
<b>V</b> iew	<b>C</b> ontroller	프로그램 로직 및 데이터 처리, 랜더링

# Django Project와 App

- Project : 웹 사이트의 전체 프로그램
- APP(Application) : 모듈화 된 단위 프로그램



# Model – 데이터 베이스 정의

모델이란?

- 사용될 데이터에 대한 정의를 담고 있는 장고 클래스
- ORM 기법 사용, 어플리케이션에서 사용할 DB를 클래스로 매핑하여 코딩할 수 있음
- 하나의 모델 클래스는 하나의 테이블에 매핑
- 모델 클래스의 속성은 테이블의 컬럼에 매핑
- DB엔진에 대한 유연성이 좋음

\* **ORM**(Object Relational Mapping)

- 객체와 관계형 DB를 연결해주는 역할
- 객체를 대상으로 필요한 작업을 실행하면 ORM이 DB처리를 함.

# Model – 데이터 베이스 정의

```
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
```



Migrate 시 실행되는 sql 문

```
(VENV) cozlab@cozlab:~/django/rapa$ python3.8 manage.py sqlmigrate polls 0001  
BEGIN;  
--  
-- Create model Question  
--  
CREATE TABLE "polls_question" ("id" integer NOT NULL PRIMARY KEY AUTOINCREMENT, "question_text" varchar(200)  
NOT NULL, "pub_date" datetime NOT NULL);
```

# URLconf - url 정의

- 클라이언트로부터 요청을 받으면 장고는 가장 먼저 요청에 들어있는 URL을 분석
  - URL이 urls.py 파일에 정의된 URL 패턴과 매칭되는지 분석
- 우아한 URL, URL/뷰 매핑
- path() 함수 : route, view 2개의 필수 인자와 kwargs, name 2개의 선택인자
- **URL 패턴 매칭은 위에서 아래로 진행, 정의하는 순서 유의**

```
from django.urls import path
from polls import views
```

```
app_name = 'polls'
urlpatterns = [
    path('', views.index, name='index'),          # /polls/
    path('<int:question_id>/', views.detail, name='detail'), # /polls/5/
    path('<int:question_id>/results/', views.results, name='results'), #/polls/5/results/
    path('<int:question_id>/vote/', views.vote, name='vote'), # /polls/5/vote/
]
```

# URLconf - url 정의

- **route** : URL 패턴을 표현하는 문자열(URL 스트링)
- **view** : URL 스트링이 매칭되면 **호출되는 뷰 함수**
  - `HttpRequest` 객체와 URL 스트링에서 추출된 항목이 뷰 함수 인자로 전달
- **kwargs** : URL 스트링에서 추출된 항목 외에 추가적인 인자를 뷰함수에 전달
- **name** : 각 URL 패턴별로 이름을 붙여줌, 템플릿 파일에서 많이 사용

```
app_name = 'polls'
urlpatterns = [
    path('', views.index, name='index'),          # /polls/
    path('<int:question_id>/', views.detail, name='detail'), # /polls/5/
    path('<int:question_id>/results/', views.results, name='results'), # /polls/5/results/
    path('<int:question_id>/vote/', views.vote, name='vote'), # /polls/5/vote/
]

1. views.index(request)
2. views.detail(request, question_id=3)
3. views.results(request, question_id=7)
4. views.vote(request, question_id=9)
```

## View – 로직 정의

- 웹 요청의 URL을 분석, 그 결과로 해당 URL에 매핑된 뷰 호출
- 뷰는 웹 요청을 받아서 DB 접속 등 해당 App의 로직에 맞는 처리 실행 후, 그 결과 데이터를 HTML로 변환(rendering) 하기 위하여 템플릿 처리 한 후에 최종 HTML로 된 응답 데이터를 웹 클라이언트에 반환하는 역할을 함.
- 함수 또는 클래스의 메소드로 작성되며, 웹 요청을 받고 응답을 반환해줌.
- 응답은 HTML 데이터일 수도 있고, 리다이렉션 명령일 수도 있고, 404 에러 메시지일 수도 있음.



# View – 로직 정의

- View 예시 – 함수형 뷰

```
def index(request):
    latest_question_list = Question.objects.all().order_by('-pub_date')[:5]
    context = {'latest_question_list': latest_question_list}
    return render(request, 'polls/index.html', context)
```

- View 예시 – Class형 뷰

```
class IndexView(generic.ListView):
    template_name = 'polls/index.html'
    context_object_name = 'latest_question_list'

    def get_queryset(self):
        """Return the last five published questions."""
        return Question.objects.order_by('-pub_date')[:5]
```

# Template – 화면 UI 정의

- 템플릿 파일의 처리 프로세스
  - 로직에 따라 응답에 사용할 html 파일을 해석
  - 최종 HTML 텍스트 응답 내용 생성
  - 생성 HTML을 클라이언트에게 보냄
  - 브라우저는 받은 HTML 텍스트를 해석해서 화면에 보여줌
- 클라이언트에 반환하는 최종 응답은 HTML 텍스트임
- \*.html 파일, 화면 UI 기본 구조를 템플릿 문법에 맞게 작성
- TEMPLATES, INSTALLED\_APPS 에 지정된 앱의 디렉토리  
순서대로 검색(프로젝트/setting.py 설정)

# Template – 화면 UI 정의

```
INSTALLED_APPS = [  
    'django.contrib.admin',  
    'django.contrib.auth',  
    'django.contrib.contenttypes',  
    'django.contrib.sessions',  
    'django.contrib.messages',  
    'django.contrib.staticfiles',  
    'polls.apps.PollsConfig', # add polls app  
]  
  
import os  
TEMPLATES = [  
    ...  
    'DIRS': [],  
    #'DIRS': [os.path.join(BASE_DIR, 'templates')],  
    ...  
]  
#BASE_DIR : project root 폴더
```

# MVT 코딩 순서(일반적인)

- 모델, 뷰, 템플릿 셋 중에 무엇을 먼저 코딩해야 한다는 건 없음.
- MVT 방식에 따르면,
  - 화면 설계는 뷰와 템플릿 코딩으로 연결됨
  - 테이블 설계는 모델 코딩에 반영됨
- 독립적으로 개발할 수 있는 모델을 먼저 코딩
- 뷰와 템플릿은 서로 영향을 미치므로 모델 이후에 같이 코딩하는 것이 일반적임.
- 뷰와 템플릿은 UI 화면을 생각하면서 로직을 풀어나가는 것이 쉽기 때문에 보통 템플릿을 먼저 코딩함.
- 클래스형 뷰를 적용할 때는 템플릿을 나중에 작성하기도 함.

# MVT 코딩 순서(일반적인)



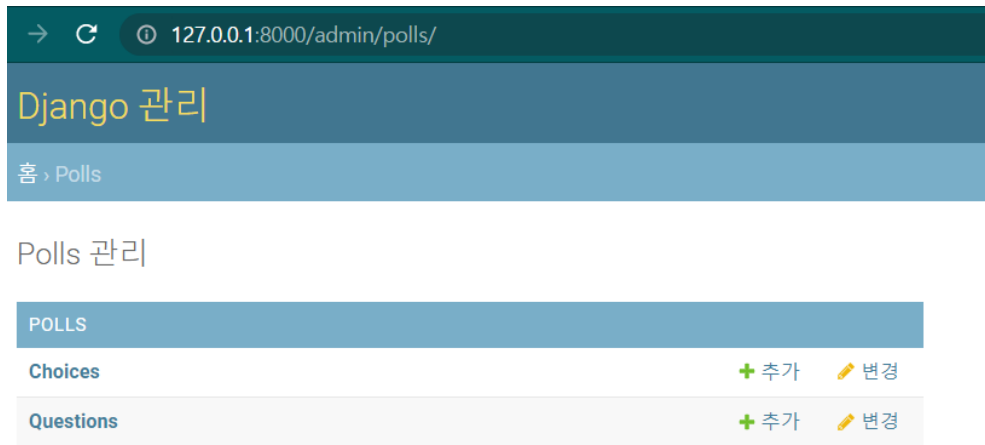
- 프로젝트 뼈대 만들기
  - 프로젝트 및 앱 개발에 필요한 디렉토리와 파일 생성
- 모델 코딩하기
  - 테이블 관련 사항을 개발(models.py, admin.py)
- URLconf 코딩하기
  - URL 및 뷰 매핑 관계를 정의(urls.py)
- 템프릿 코딩하기
  - 화면 UI 개발(templates/\*.html)
- 뷰 코딩하기
  - 애플리케이션 로직 개발(views.py)

---

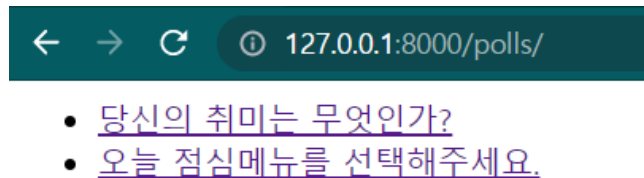
# 앱 설계하기

# 설문 앱 개발

- admin 화면



- user 화면



# 앱 설계하기 - UI 설계

## index.html

- 당신의 취미는 무엇인가요?
- 오늘 저녁 메뉴를 선택해 주세요.

## detail.html

당신의 취미는 무엇인가요?

- ☐ 탁구
- ☐ 등산
- ☐ 독서
- ☐ 드라마보기

## results.html

당신의 취미는 무엇인가요?

- 탁구 - 3 votes
- 등산 - 2 votes
- 독서 - 4 votes
- 드라마보기 - 6 votes

- index.html : 최근에 실시하고 있는 **질문의 리스트 화면**
- detail.html : 하나의 질문에 대해 투표할 수 있는 **답변 항목 폼 화면**
- results.html : 질문에 따른 투표 **결과 화면**



# 앱 설계하기 - UI 설계

- 요구사항에 따른 UI 설계 및 결과 파일명 정의
- 요구사항에 따라 필요한 데이터를 추출하여 DB 테이블 설계
- 질문을 저장하는 테이블(Question)
- 질문별로 선택용 답변 항목을 저장하는 테이블(Choice)

예시)Question 테이블 설계

컬럼명	타입	제약조건	설명
id	integer	NotNull, PK, AutoIncrement	Primary Key
question_text	varchar(200)	NotNull	질문 문장
pub_date	datetime	NotNull	질문 생성 시간

# 앱 설계하기 - UI 설계

- 요구사항에 따른 UI 설계 및 결과 파일명 정의
- 요구사항에 따라 필요한 데이터를 추출하여 DB 테이블 설계
- 질문을 저장하는 테이블(Question)
- 질문별로 선택용 답변 항목을 저장하는 테이블(Choice)

예시)Choice 테이블 설계

컬럼명	타입	제약조건	설명
id	integer	NotNull, PK, AutoIncrement	Primary Key
choice_text	varchar(200)	NotNull	질문 문장
votes	integer	NotNull	질문 생성 시간
question	integer	NotNull, FK(Question.id).index	Foreign Key

\* Foreign Key : 관계형 DB에서 다른 테이블과 연결되도록 하는 장치

---

# Django의 핵심 기능

# Django의 핵심 기능

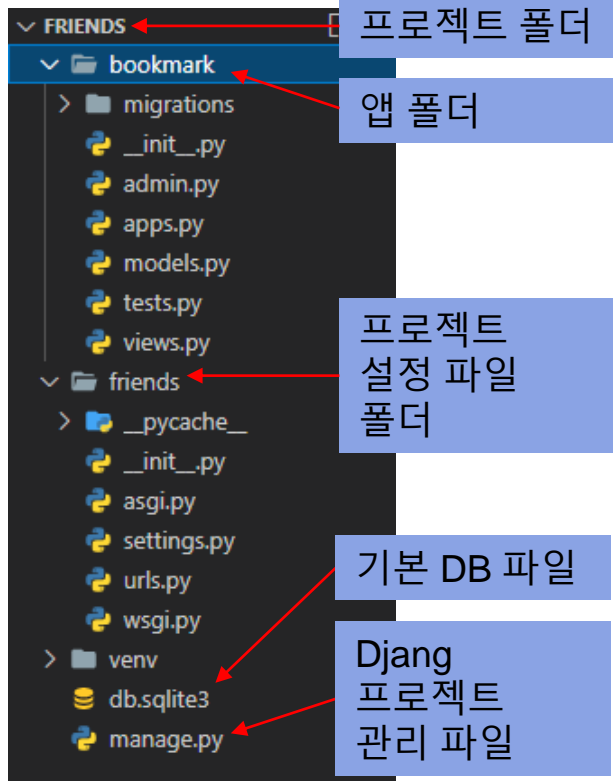


- Admin 사이트 꾸미기
- Django 파이썬 셸로 데이터 조작하기
- 템플릿 시스템(Look and Feel)
- 폼 클래스 처리
- 클래스형 뷰(상속과 믹스 -> 코드 재사용)
- 로그 남기기

# 프로젝트 뼈대 만들기

- 프로젝트에 필요한 디렉토리 및 파일 구성
- 설정파일 셋팅
- 기본테이블 생성(필요시)
- 관리자 계정 : 슈퍼유저 생성(필요시)
- 어플리케이션 생성
- 필요한 파일 구성

## dir/file 기본 구조



# 프로젝트 뼈대 만들기 – 뼈대 디렉토리 및 파일 설명

항목명	설명
rapa (d)	프로젝트 관련 디렉토리 및 파일을 모아주는 최상위 루트 디렉토리 보통 settings.py 파일에서 BASE_DIR 항목으로 지정됨
db.sqlite3	SQLite3 DB 파일, 테이블 저장
manage.py	장고의 명령어를 처리하는 파일
rapa (d)	프로젝트 명으로 만들어진 디렉토리. 프로젝트 관련 파일들이 들어있음
__init__.py	디렉토리에 이 파일이 있으면 파이썬 패키지로 인식
settings.py	프로젝트 설정 파일
urls.py	프로젝트 레벨의 URL 패턴을 정의하는 최상위 URLconf
wsgi.py	Apache와 같은 웹 서버와 WSGI 규격으로 연동하기 위한 파일

# 프로젝트 뼈대 만들기 – 뼈대 디렉토리 및 파일 설명

항목명	설명
polls (d)	앱 명으로 만들어진 앱 디렉토리 해당 앱 관련 파일이 들어있음.
__init__.py	디렉토리에 이 파일이 있으면 파이썬 패키지로 인식함
admin.py	Admin 사이트에 모델 클래스를 등록해 주는 파일
apps.py	앱의 설정 클래스를 정의하는 파일
migrations (d)	DB 변경사항을 관리하기 위한 디렉토리 DB에 추가, 삭제, 변경 등이 발생하면 변경 내역을 기록한 파일들이 위치함.
models.py	DB 모델 클래스를 정의하는 파일
tests.py	단위 테스트용 파일
views.py	뷰 함수를 정의하는 파일. 함수형 뷰와 클래스형 뷰 모두 정의

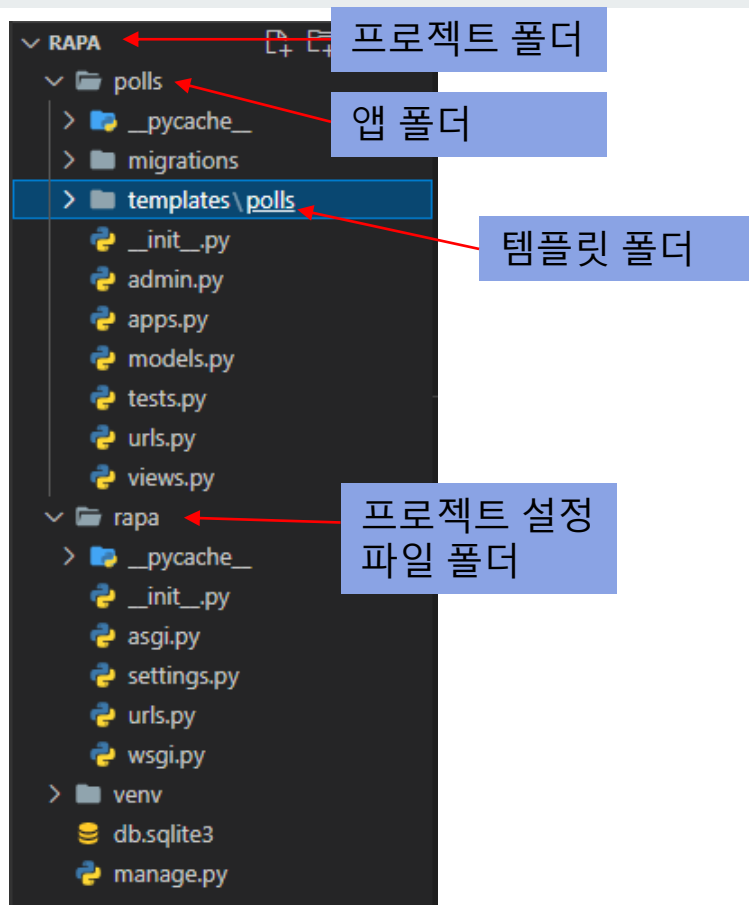
# 프로젝트 뼈대 만들기 – 뼈대 디렉토리 및 파일 설명

항목명	설명
templates (d)	프로젝트 진행하면서 추가됨 템플릿 파일들이 들어있음. 보통은 프로젝트 레벨과 앱 레벨의 템플릿으로 구분 rapa/templates , rapa/polls/templates 로 생성
static (d)	프로젝트 진행하면서 추가됨 CSS, Image, Javascript 파일들 위치함 보통은 프로젝트 레벨과 앱 레벨의 템플릿으로 구분 rapa/static , rapa/polls/static 로 생성
log (d)	프로젝트를 진행하면서 추가됨 로그 파일들이 들어있음. 로그 파일의 위치는 settings.py 파일에서 LOGGING 항목으로 지정



# 프로젝트 뼈대 만들기

- 템플릿 폴더의 위치
- setting.py에 TEMPLATES DIR을 설정하지 않았을 때 templates의 기본 폴더 위치



# 기본 뼈대를 만들고 확인하는 명령어



- `django-admin startproject` 프로젝트명
- `python manage.py startapp` 앱명
  - 앱 생성 후 `setting.py` 파일 기본 셋팅 수정
- `python manage.py migrate`
- `python manage.py runserver`

---

# Django의 핵심 기능

# Django의 핵심 기능



- Admin 사이트 꾸미기
- Django python shell로 데이터 조작하기
- Template 시스템
- Form 처리
- Class형 View
- Log 남기기

# Admin 사이트 꾸미기

---

- Django Admin 사이트는 DB에 들어있는 데이터를 쉽게 관리할 수 있도록 **데이터 생성, 조회, 조회, 변경, 삭제** 등의 기능 제공
- 깔끔하게 정돈된 **Look and Feel UI** 제공, 수정 가능

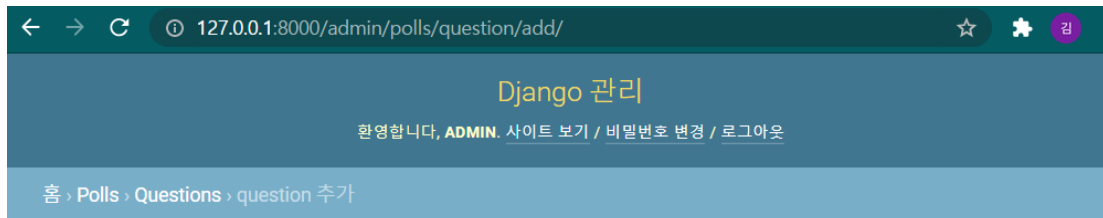
# Admin 사이트 - CRUD



- Create(input)
- Read
- Update
- Data

# Admin 사이트 – CRUD – Question 모델 클래스

- models.py의 Question 모델 클래스의 속성과 UI 확인



question 추가

Question text:

당시는 어디에 살고 있습니까?

Date published:

날짜: 2021-09-19

오늘 | 📅

시각: 12:00:00

현재 | ⌚

models.py에 정의 한 테이블 모델을 기초로 자동생성

```
question_text = models.CharField(max_length=200)
```

```
pub_date = models.DateTimeField('date published')
```

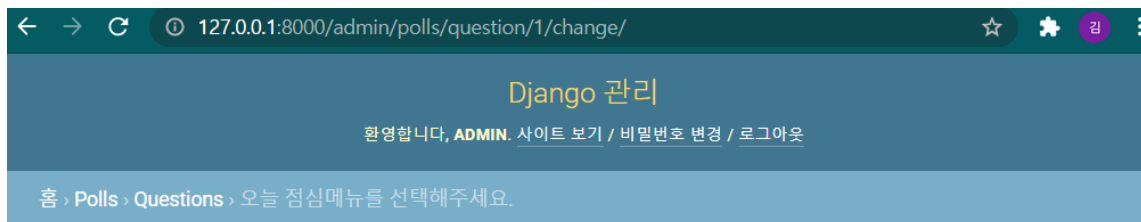
»

저장 및 다른 이름으로 추가

저장 및 편집 계속

저장

# Admin 사이트 – CRUD – Question 모델 클래스



question 변경

변경 이력

오늘 점심메뉴를 선택해주세요.

Question text:

오늘 점심메뉴를 선택해주세요.

Date published:

날짜: 2021-09-18

오늘 | 📅

시각: 06:00:00

현재 | ⌚

변경사항을  
저장하고 새로운  
레코드를 추가할  
수 있는 폼  
페이지를 보여줌

변경사항을  
저장하고 현재  
페이지 다시  
보여줌

변경사항 저장

삭제

삭제

저장 및 다른 이름으로 추가

저장 및 편집 계속

저장



# Admin 사이트 – CRUD – fields 순서 바꾸기

홈 > Polls > Questions > 당신의 취미는 무엇인가?

question 변경

당신의 취미는 무엇인가?

히스토리

Date published:

날짜:

2021-09-20

오늘



시각:

12:00:00

현재



Question text:

당신의 취미는 무엇인가?

삭제

저장 및 다른 이름으로 추가

저장 및 편집 계속

저장

polls/admin.py 수정

```
# Register your models here.
class QuestionAdmin(admin.ModelAdmin): #add admin
    fields = ['pub_date', 'question_text']

admin.site.register(Question, QuestionAdmin) # register admin : QuestionAdmin
admin.site.register(Choice)
```

# Admin 사이트 – CRUD – fields 분리

홈 > Polls > Questions > 당신의 취미는 무엇인가?

question 변경

히스토리

당신의 취미는 무엇인가?

Question Statement

Question text: 당신의 취미는 무엇인가?

Date Information

Date published: 날짜: 2021-09-20 시각: 12:00:00

삭제

polls/admin.py 수정

```
4 # Register your models here.
5 class QuestionAdmin(admin.ModelAdmin): #add admin
6     #fields = ['question_text', 'pub_date']
7     fieldsets = [
8         ('Question Statement', {'fields': ['question_text']}),
9         ('Date Information', {'fields': ['pub_date']}),
10    ]
11
12 admin.site.register(Question, QuestionAdmin) # register admin : QuestionAdmin
13 admin.site.register(Choice)
14
```

# Admin 사이트 – CRUD – field 접기

홈 > Polls > Questions > 당신의 취미는 무엇인가?

question 변경

당신의 취미는 무엇인가?

히스토리

Question Statement

Question text:

당신의 취미는 무엇인가?

Date Information ([보기](#))

삭제

polls/admin.py 수정

```
5 class QuestionAdmin(admin.ModelAdmin): #add admin
6     #fields = ['question_text', 'pub_date']
7     fieldsets = [
8         ('Question Statement', {'fields': ['question_text']}),
9         #('Date Information', {'fields': ['pub_date']}),
10        ('Date Information', {'fields': ['pub_date'], 'classes': ['collapse']})
11    ]
12
13 admin.site.register(Question, QuestionAdmin) # register admin : QuestionAdmin
14 admin.site.register(Choice)
15
```

# Admin 사이트 – CRUD – Choise 모델 클래스

- 외래키 관계 화면 이해
- Choise 모델 클래스에 대해 추가, 변경 작업 해보기

홈 > Polls > Choices > choice 추가

choice 추가

Question:

Choice text:

Votes:

0

»

저장 및 다른 이름으로 추가    저장 및 편집 계속    저장

Question 테이블의 question id 레코드와 외래키로 연결 되 있음.

# Admin 사이트 – CRUD – Choise 레코드 추가

홈 > Polls > Choices > choice 추가

choice 추가

Question:

당신의 취미는 무엇인가?



Choice text:

드라마보기

Votes:

0

저장 및 다른 이름으로 추가

저장 및 편집 계속

저장

# Admin 사이트 – CRUD

- Question, Choice 한 화면에서 변경하도록 수정

홈 > Polls > Questions > question 추가

question 추가

Question Statement

Question text:

Date Information (보기)

저장 및 다른 이름으로 추가 저장 및 편집 >>

question 추가

Question text:

Date Information (보기)

CHOICES

Choice: #1 ✕

Choice text:

Votes:

Choice: #2 ✕

Choice text:

Votes:

# Admin 사이트 – CRUD

- Question, Choice 한 화면에서 변경하기

홈 > Polls > Questions > question 추가

question 추가

Question text:

Date Information (보기)

CHOICES

Choice: #1

Choice text:

Votes:

Choice: #2

Choice text:

Votes:

- 설명

polls/admin.py 수정

```
5 class ChoiceInline(admin.StackedInline):
6     model = Choice
7     extra = 2
8
9 class QuestionAdmin(admin.ModelAdmin): #add admin
10     #fields = ['question_text', 'pub_date']
11     fieldsets = [
12         ('Question Statement', {'fields': ['question_text']}),
13         (None, {'fields': ['question_text']}),
14         ('Date Information', {'fields': ['pub_date']}),
15         ('Date Information', {'fields': ['pub_date'], 'classes': ['collapse']})
16     ]
17     inlines = [(ChoiceInline)] # add Choice
```

# Admin 사이트 – CRUD

- 테이블 형식으로 보여주기로 변경

polls/admin.py 수정

question 추가

Question text:

Date Information (보기)

## CHOICES

CHOICE TEXT

VOTES

삭제



+ Choice 더 추가하기

저장 및 다른 이름으로 추가

저장 및 편집 계속

저장

```
5 #class ChoiceInline(admin.StackedInline):  
6 class ChoiceInline(admin.TabularInline):  
7     model = Choice  
8     extra = 2  
9
```



# Admin 사이트 – CRUD

- 레코드 리스트 컬럼 지정 수정하기

홈 > Polls > Questions

변경할 question 선택

QUESTION 추가 +

액션:  실행

0개가 2개 중에 선택됨.

<input type="checkbox"/>	QUESTION
<input type="checkbox"/>	당신의 취미는 무엇인가?
<input type="checkbox"/>	오늘 점심메뉴를 선택해주세요.

2 questions

- 각 레코드의 제목은 디폴트로 models.py에서 정의한 `__str__()` 메소드의 리턴값을 레코드 제목으로 사용함.
- [변경] polls/admin.py파일에서 `list_display` 속성 추가

# Admin 사이트 – CRUD

- 레코드 리스트 컬럼 지정 수정하기

홈 > Polls > Questions

polls/admin.py 수정

변경할 question 선택

QUESTION 추가 +

18  
19  
20

```
inlines = [(ChoiceInline)] # add Choice 모델 클래스 같이보이도록  
list_display = ('question_text', 'pub_date') # 레코드 리스트 컬럼 지정
```

액션: -----



실행

2 중 아무것도 선택되지 않았습니다.

<input type="checkbox"/> QUESTION TEXT	DATE PUBLISHED
<input type="checkbox"/> 당신의 취미는 무엇인가?	2021년 9월 20일 12:00 오후
<input type="checkbox"/> 오늘 점심메뉴를 선택해주세요.	2021년 9월 18일 6:00 오전

2 questions

# Admin 사이트 – CRUD

- list\_filters 추가하기

변경할 question 선택

QUESTION 추가 +



검색

액션:



실행

2 중 아무것도

선택되지 않았습니다.



QUESTION TEXT

DATE PUBLISHED



당신의 취미는 무엇인가?

2021년 9월 20일 12:00 오후



오늘 점심메뉴를 선택해주세요.

2021년 9월 18일 6:00 오전

2 questions

필터

date published (으)로

언제나

오늘

지난 7일

이번 달

이번 해


polls/admin.py 수정

```
18 inlines = [(ChoiceInline)] # add Choice 모델 클래스 같이보이도록
19 list_display = ('question_text', 'pub_date') # 레코드 리스트 컬럼 지정
20 list_filter = ['pub_date'] # 필터 사이드 바 추가
21 search_fields = ['question_text'] # 검색 박스 추가하기
```

# Admin 사이트 – CRUD – polls/admin.py 최종 내용

```
• from django.contrib import admin
• from polls.models import Question, Choice
•
  # Register your models here.
• class ChoiceInline(admin.TabularInline):
•     model = Choice
•     extra = 2
•
  class QuestionAdmin(admin.ModelAdmin): #add admin
•     fieldsets = [
•         (None, {'fields': ['question_text']}),
•         ('Date Information', {'fields': ['pub_date'], 'classes': ['collapse']})
•     ]
•     inlines = [(ChoiceInline)] # add Choice 모델 클래스 같이보이도록
•     list_display = ('question_text', 'pub_date') # 레코드 리스트 컬럼 지정
•     list_filter = ['pub_date'] # 필터 사이드 바 추가
•     search_fields = ['question_text'] # 검색 박스 추가하기
•
  admin.site.register(Question, QuestionAdmin) # register admin : QuestionAdmin
• admin.site.register(Choice)
```

# Django Admin site 참고 자료



- <https://www.djangoproject.com/start/>
- <https://docs.djangoproject.com/ko/3.2/ref/>
- <https://docs.djangoproject.com/ko/3.2/intro/>
- <https://docs.djangoproject.com/ko/3.2/intro/tutorial07/>
- <https://tutorial.djangogirls.org/ko/django/>
- <https://wikidocs.net/6667>

---

# Django 파이썬 웹 활용 데이터 조장하기

# Django python shell - 실행하기

- Admin UI에서 수행하는 데이터 CRUD 작업을 Django python shell에서 가능
- manage.py 모듈에 정의한 DJANGO\_SETTINGS\_MODULE 속성을 이용하여 미리 프로젝트/settings.py 모듈을 임포트 함

python manage.py shell

```
(venv) D:\django\rapa>python manage.py shell
Python 3.8.7 (tags/v3.8.7:6503f05, Dec 21 2020, 17:59:51) [MSC v.1928 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
(InteractiveConsole)
>>> █
```

# Django python shell – CRUD – Create

- Create – DB 데이터 레코드 입력
- 테이블에 레코드를 생성하기 위해서는 필드 값을 지정하여 객체를 생성한 후에 `save()` 메소드 실행(INSERT 문)

```
>>> from polls.models import Question, Choice
>>> from django.utils import timezone
>>> q = Question(question_text = "새로운 소식은 있습니까?", pub_date=timezone.now())
>>> q = Question(question_text = "좋아하는 분야는 무엇입니까??", pub_date=timezone.now())
>>> q.save()
```



# Django python shell – CRUD – Create

변경할 question 선택

QUESTION 추가 +

Q

검색

필터

date published (으)로

언제나

액션:

실행 3 중 아무것도

선택되지 않았습니다.

☐ QUESTION TEXT DATE P

☐ 좋아하는 분야는 무엇입니까?? 2021년

☐ 당신의 취미는 무엇인가? 2021년

☐ 오늘 점심메뉴를 선택해주세요. 2021년

3 questions

```
(venv) D:\django\rapa>python manage.py shell
Python 3.8.7 (tags/v3.8.7:6503f05, Dec 21 2020, 17:59:51) [MSC v.1928 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
(InteractiveConsole)
>>> from polls.models import Question, Choice
>>> from django.utils import timezone
Traceback (most recent call last):
  File "<console>", line 1, in <module>
ModuleNotFoundError: No module named 'django.utils'
>>> from django.utils import timezone
>>> q = Question(question_text = "새로운 소식은 있습니까?", pub_date=timezone.now())
>>> q = Question(question_text = "좋아하는 분야는 무엇입니까??", pub_date=timezone.now())
>>> a.save()
Traceback (most recent call last):
  File "<console>", line 1, in <module>
NameError: name 'a' is not defined
>>> q.save()
>>>
```

# Django python shell – CRUD – Create



```
>>> new_item = Question()
>>> new_item
<Question: >
>>> new_item.question_text = "당신의 전공은 무엇입니까?"
>>> new_item
<Question: 당신의 전공은 무엇입니까?>
>>> new_item.pub_date = timezone.now()
>>> new_item.save()
```

# Django python shell – CRUD – Read

- Read : DB 데이터 레코드 수정
  - QuerySet 객체 사용
  - QuerySet은 DB 테이블로 부터 꺼내 온 객체들의 컬렉션
  - QuerySet은 필터를 가질 수 있으며, 필터를 사용하여 QuerySet 내의 항목 중에서 조건에 맞는 레코드만 추출함.
- 
- 컬렉션이란?
  - 다수의 객체를 한 곳에 모아서 각 객체들을 동일한 방식으로 다룰 수 있도록 해주는 데이터 구조, 여러 객체의 모임.

# Django python shell – CRUD – Read

- Read : DB 데이터 레코드 수정
  - QuerySet 객체 사용
  - QuerySet은 DB 테이블로 부터 꺼내 온 객체들의 컬렉션
  - QuerySet은 필터를 가질 수 있으며, 필터를 사용하여 QuerySet 내의 항목 중에서 조건에 맞는 레코드만 추출함.
- 
- 컬렉션이란?
  - 다수의 객체를 한 곳에 모아서 각 객체들을 동일한 방식으로 다룰 수 있도록 해주는 데이터 구조, 여러 객체의 모임.

# Django python shell – CRUD – Read



- `Restaurant.objects.all()`
- `Restaurant.objects.all().values()`
- `Restaurant.objects.order_by('-created_at').values()`
- `Restaurant.objects.order_by('name').values()`
- `Restaurant.objects.get(pk=1).name`
- `Restaurant.objects.filter(name='Deli Shop').values()`

# Django python shell – CRUD – Read

```
>>> from polls.models import Question, Choice
```

```
>>> from django.utils import timezone
```

```
>>> Question.objects.all()
```

```
<QuerySet [<Question: 오늘 점심메뉴를 선택해주세요.>, <Question: 당신의 취미는 무엇인가?>, <Question: 좋아하는 분야는 무엇입니까??>]>
```

```
>>> one_entry = Question.objects.get(pk=1)
```

```
>>> Question.objects.get(pk=1)
```

```
<Question: 오늘 점심메뉴를 선택해주세요.>
```

```
>>> Question.objects.get(pk=2)
```

```
<Question: 당신의 취미는 무엇인가?>
```

```
>>> Question.objects.get(pk=3)
```

```
<Question: 좋아하는 분야는 무엇입니까??>
```

```
>>> Question.objects.all().values()
```

```
<QuerySet [{'id': 1, 'question_text': '오늘 점심메뉴를 선택해주세요.', 'pub_date': datetime.datetime(2021, 9, 17, 21, 0, tzinfo=<UTC>)}, {'id': 2, 'question_text': '당신의 취미는 무엇인가?', 'pub_date': datetime.datetime(2021, 9, 20, 3, 0, tzinfo=<UTC>)}, {'id': 3, 'question_text': '당신이 가장 좋아하는 취미는 무엇입니까?', 'pub_date': datetime.datetime(2021, 9, 19, 3, 15, 41, 235225, tzinfo=<UTC>)}]>
```

```
>>>
```

# Django python shell – CRUD – Update



- Update : DB 데이터 레코드 수정
- 이미 존재하는 객체에 대한 필드값을 수정한 후,  
save() 메소드 실행

# Django python shell – CRUD – Update



```
>>> item = Question.objects.get(pk=2)
```

```
>>> item.question_text
```

```
'당신의 취미는 무엇인가?'
```

```
>>> item.question_text = "당신이 가장 좋아하는 취미는 무엇입니까?"
```

```
>>> item.save()
```



# Django python shell – CRUD – Delete

- Delete : DB 데이터 레코드 삭제
- delete() 메서드
- delete는 save 메서드가 없이. DB에서 바로 삭제됨(주의하기)

```
>>> Question.objects.all().values()
```

```
<QuerySet [{'id': 1, 'question_text': '오늘 점심메뉴를 선택해주세요.', 'pub_date': datetime.datetime(2021, 9, 17, 21, 0, tzinfo=<UTC>)}, {'id': 2, 'question_text': '당신이 가장 좋아하는 취미는 무엇입니까?', 'pub_date': datetime.datetime(2021, 9, 20, 3, 0, tzinfo=<UTC>)}, {'id': 3, 'question_text': '좋아하는 분야는 무엇입니까?', 'pub_date': datetime.datetime(2021, 9, 19, 3, 15, 41, 235225, tzinfo=<UTC>)}, {'id': 4, 'question_text': '당신의 전공은 무엇입니까?', 'pub_date': datetime.datetime(2021, 9, 19, 4, 16, 59, 661584, tzinfo=<UTC>)}]>
```

```
>>> item = Question.objects.get(pk=4)
```

```
>>> item
```

```
<Question: 당신의 전공은 무엇입니까?>
```

```
>>> item.delete()
```

```
(1, {'polls.Question': 1})
```

```
>>> Question.objects.all().values()
```

```
<QuerySet [{'id': 1, 'question_text': '오늘 점심메뉴를 선택해주세요.', 'pub_date': datetime.datetime(2021, 9, 17, 21, 0, tzinfo=<UTC>)}, {'id': 2, 'question_text': '당신이 가장 좋아하는 취미는 무엇입니까?', 'pub_date': datetime.datetime(2021, 9, 20, 3, 0, tzinfo=<UTC>)}, {'id': 3, 'question_text': '좋아하는 분야는 무엇입니까?', 'pub_date': datetime.datetime(2021, 9, 19, 3, 15, 41, 235225, tzinfo=<UTC>)}]>
```

---

# Template 시스템

# Django Template 시스템 이해

- 사용자에게 보여주는 화면, UI(User Interface) 담당
- templat에는 로직이 표현되는 것이 아니라 사용자에게 어떻게 보여줄지에 대한 Look and Feel을 표현하는 것
- 화면 구현이 적절한 표현임
- 템플릿 코드 문법
  - If 태그, for 태그 등, 파이썬 언어의 문법과 다름
  - 템플릿 시스템에서만 사용되는 고유 문법
- 템플릿 시스템은 템플릿 문법으로 작성된 템플릿 코드를 해석하여 템플릿 파일로 결과물을 만들어 줌(Rendering)
- 결과물은 HTML, XML, JSON 등의 단순한 텍스트 파일

# Django Template 시스템의 고유 문법

템플릿 코드 vs 템플릿 파일

- 템플릿 코드
  - 렌더링 전의 템플릿 문법에 따라 작성된 파일
- 템플릿 파일
  - 렌더링 후의 결과물인 HTML과 같은 텍스트 파일
  - 웹 브라우저로 전달되는 파일

# Django Template 시스템의 고유 문법



- 템플릿 변수
- 템플릿 필터
- 템플릿 태그
- 템플릿 주석
- HTML 이스케이프
- 템플릿 상속

# Django Template – 템플릿 변수

`{{ variable }}`

- 변수명
  - 일반 프로그래밍의 변수명 처럼 문자, 숫자, \_ 사용
  - 변수의 속성에 접근하는 도트( . ) 표현식 가능
- 정의되어 있지 않은 변수를 사용하는 경우, 빈 문자열로 채워줌
  - 이 값을 변경하려면 settings.py 파일 수정하면 됨
  - `TEMPLATE_STRING_IF_INVALID`
- ( . )를 만나면 장고는 다음 순서로 lookup을 시도함.(ex : `foo.bar`)
  - `foo`가 dict 타입 이면, `foo['bar']`로 해석
  - 아니면, `foo`의 속성 이면, `bar`라는 속성이 있으면 `foo.bar`로 해석
  - 아니면, `foo`가 리스트 이면, `foo[bar]`로 해석

# Django Template – 템플릿 필터

```
{{ name|lower }}
```

# name 변수값의 모든 문자를 소문자로 바꿈

- 필터란? 어떤 객체나 처리 결과에 추가적으로 명령을 적용하여 해당 명령에 맞게 최종 결과를 변경하는 것.
- 장고에서도 템플릿 변수에 필터를 적용하여 변수의 출력 결과를 변경할 수 있음.
- ( | )파이프 문자 사용

약 60여가지 필터 제공, 사용자 정의도 가능

<https://docs.djangoproject.com/ko/3.2/ref/templates/builtins/>

# Django Template – 템플릿 필터

```
{{ text|escape|linebreaks }}
```

# text변수 값 중에서 특수 문자를 escape 해주고, 그 결과에 html <p> 태그를 붙여 줌

```
{{ bio|truncatewords:30 }}
```

# bio변수 값 중에서 앞에 30개의 단어만 보여주고, 줄 바꿈 문자는 모두 없애 줌

```
{{ list|join:" // " }}
```

# list의 인자들을 지정한 기호로 구분하여 하나로 합침.

# 필터의 인자에 빈칸이 있는 경우, 따옴표로 묶어 줌. " // "

# ex) list = ['a', 'b', 'c'] 라면, "a // b // c"



# Django Template – 템플릿 필터

```
{{ value|default:"nothing" }}
```

# value 변수 값이 False이거나 없는 경우, "nothing" 으로 보여줌

```
{{ value|length }}
```

# value 변수값의 길이를 반환, 스트링, 리스트 가능.

# ex) value=['a', 'b', 'c'] 라면, 결과 3이 됨

# Django Template – 템플릿 필터

```
{{ value|pluralize }}
```

# 복수 접미사 필터

# value의 값이 1이면 value, 2이상이면 values가 됨

```
{{ value|pluralize:"es" }} 또는 {{ value|pluralize:"ies" }}
```

# value 변수 값이 1이 아니면 복수 접미사를 붙여 줌.

# Django Template – 템플릿 필터

```
{{ value|add:"2" }}
```

# 더하기 필터,

# value의 값이 4라면, 2를 더하여 6이 됨, 데이터 타입에 따라 결과가 달라짐 주의해서 사용하기

```
{{ first|add:second }}
```

# first = "python", second = "Django" => "pythonDjango"

# first = [1, 2, 3], second = [4, 5, 6] => [1, 2, 3, 4, 5, 6]

# first = "5", second="10" => 15

# Django Template – 템플릿 태그

{% 태그 %}

- 텍스트 결과물을 만드는 기능
- 템플릿 로직을 제어하는 기능
- 외부 파일을 템플릿 내로 로딩하는 기능
- 어떤 태그는 시작 태그와 끝 태그가 있어야함
- {% for %}, {% if %} 가장 많이 사용
- 폼 템플릿 태그 인 {% csrf\_token %}
- {% url %}, {% with %}, {% load %}

템플릿 태그 참조

<https://docs.djangoproject.com/ko/3.2/howto/custom-template-tags/>

# Django Template – 템플릿 태그

`{% for %} ... {% endfor %}`

- 리스트에 저장되어 있는 항목들을 순회하면서 출력할 수 있음

```
<ul>
```

```
{% for sdnt in sdnts_list %}
```

```
    <li>{{ sdnt.name }}
```

```
{% endfor %}
```

```
</ul>
```

# 학생들 리스트(sdnts\_list)에 들어있는 항목(sdnt)을 순회하면서  
각 학생들의 이름(sdnt.name)을 출력하는 문장

# Django Template – 템플릿 태그

{% for %} ... {% endfor %}

- for 태그에 사용되는 변수들

변수명	설명
forloop.counter	현재까지 루프를 실행한 루프 카운트(1부터 카운트함)
forloop.counter()	현재까지 루프를 실행한 루프 카운트(0부터 카운트함)
forloop.revcounter	루프 끝에서 현재까지 몇 번째인지 카운트한 숫자(1부터 카운트)
forloop.revcounter()	루프 끝에서 현재까지 몇 번째인지 카운트한 숫자(0부터 카운트)
forloop.first	루프에서 첫 번째 실행이면, True 값을 가짐
forloop.last	루프에서 마지막 실행이면, True 값을 가짐
forloop.parentloop	중첩된 루프에서 현재의 루프 바로 상위의 루프를 의미함

# Django Template – 템플릿 태그

```
{% if %} ... {% endif %}
```

- 변수의 값을 평가하여 True이면 바로 아래 문장이 표시됨

```
{% if stdnt_list != None %}
```

```
    Number of stdnt_list: {{ stdnt_list|length }}
```

```
{% else %}
```

```
    No students
```

```
{% endif %}
```

# Django Template – 템플릿 태그

```
{% if stdnt_list|length > 1 %}  
    Number of athletes : {{ s }}  
{% endfor %}
```

- 대부분의 필터가 스트링을 반환하므로 비교 연산이 안되는데 length 필터는 예외적으로 비교연산 가능
- {% if %} 시작 태그에 불린 연산자 사용 가능  
and, or, not, and not, ==, !=, <, >, <=, >=, in, not in



# Django Template – 템플릿 태그

{% if 조건 1 %}

조건1이 참이면 실행하고 조건문 탈출

{% elif 조건 2 %}

조건 2가 참인 경우 실행하고 조건문 탈출

{% elif 조건 3 %}

조건 3가 참인 경우 실행하고 조건문 탈출

{% else %}

모든 조건이 거짓인 경우 실행하고 조건문 탈출

{% endif %}

# Django Template – 템플릿 태그

`{% csrf_token %}`

- POST 방식의 `<form>`을 사용하는 템플릿 코드에서 CSRF 공격을 방지하기 위하여 사용함.
- 위치는 여는 `<form>`태그 바로 다음에 넣으면 됨  
`<form action="." method="post">{% csrf_token %}`
- 이 태그를 사용하면 장고는 내부적으로 CSRF 토큰값의 유효성을 검증함.
- 토큰값 검증이 실패하면 사용자에게 403 에러를 보여줌.

\* CSRF(Cross Site Request Forgery)

사이트간 요청 위조 공격. 웹 사이트의 취약점을 공격하는 방식 중의 하나로, 특정 웹 사이트에서 이미 인증을 받은 사용자를 이용하여 공격을 시도함.

인증 받은 사용자가 공격 코드가 삽입된 페이지를 열면 공격 대상이 되는 웹사이트는 위조된 공격 명령이 믿을 수 있는 사용자로부터 발송된 것으로 판단하게 되어 공격을 받게 되는 방식

# Django Template – 템플릿 태그

`{% url %}`

- 소스에 URL을 하드코딩 하는 것을 방지하기 위한 것
- URL이 변경되더라도 URLconf만을 참조하여 원하는 URL을 추출함.

```
<form action="{% url 'polls:vote' question.id %}" method="post">
```

url 태그를 사용하지 않으면 다음과 같이 하드코딩을 해야함.

```
<form action="/polls/3/vote/" method="post">
```

단점, conf와 모든 html을 찾아서 변경해야 함.

# Django Template – 템플릿 태그

```
{% url 'namespace:view-name' arg1 arg2 %}
```

- namespace : urls.py 파일의 include() 함수 또는 app\_name 변수에 정의한 이름
- view-name : urls.py 파일에서 정의한 URL 패턴 이름
- argN : 뷰 함수에서 사용하는 인자로, 없을 수도 있음.
  - 여러 개인 경우 빈칸으로 구분함.

# Django Template – 템플릿 태그

{% with %}

- 특정 값을 변수에 저장해 두는 기능

{% with total = business.employees.count %}

    {{ total }} people works at business

{% endwith %}

- total 변수의 유효 범위는 with 문내에만 사용됨.
- DB를 조회와 같은 부가가 큰 동작의 결과를 저장해 둬으로써, 동일한 동작에 대해 재사용함으로써 부하를 줄이기 위한 태그

# Django Template – 템플릿 태그

`{% load %}`

- 사용자 정의 태그 및 필터를 로딩 하는 기능
- 내장 태그 및 필터 외에 사용자 정의 태그와 필터를 사용하기 위해서 사용전에 로딩을 해야함.

`{% load somelibrary package.otherlibrary %}`

- somelibrary.py 파일 및 package/otherlibrary.py 파일에 정의된 사용자 정의 태그 및 필터를 로딩함.

참조 : 약 30여개의 내장 태그가 있음. 사용자 정의 태그 방법 참조  
<https://docs.djangoproject.com/ko/3.2/ref/templates/builtins/>

# Django Template – 템플릿 주석

- 한 줄 주석문

```
{# greeting #}hello
```

- 여러 줄 주석문

```
{% comment "Optional note" %}
```

```
....
```

```
....
```

```
{% endcomment %}
```

# Django Template – HTML 이스케이프

- 템플릿 코드를 렌더링하여 HTML 텍스트를 만들 때 주의 사항
- 템플릿 변수에 HTML의 태그가 들어 있는 경우, 있는 그대로 렌더링함.
- XSS공격으로부터 보호하기 위한 기능  
예) 템플릿 코드에서 아래와 같은 문장을 사용하면  
name = "<b>username"  
Hello, {{ name }}  
=> 랜더링 결과는  
Hello, <b>username
- 위와 같은 결과를 방지하기 위해 자동 이스케이프 기능 제공



# Django Template – HTML 이스케이프

- < (less than) -> &lt;
  - > (greater than) -> &gt;
  - ' (single quote) -> &#39;
  - " (double quote) -> &quot;
  - & (ampersand) -> &amp;
- 
- 디폴트로 HTML에 사용되는 예약된 특수 문자들을 위와 같이 예약 의미를 제거한 문자로 변경해 주는 기능

# Django Template – HTML 이스케이프

- 자동 HTML 기능 비활성화 방법

- 방법1

This will not be escaped : {{ data|safe }}

- 방법2

```
{% autoescape off %}
```

```
hello {{ name }}
```

```
{% endautoescape %}
```

- 필터의 인자에 사용되는 스트링 리터럴에는 자동 이스케이프 기능이 적용되지 않음. 그래서 두번째 방법 권장

```
{{ data|default:"3<5" }} # 스트링 리터럴에서는 자동 이스케이프 안됨
```

```
{{ data|default:"3 &lt; 5" }}
```

# Django Template – Template 상속

- Template 상속은 문법 중에서 가장 복잡하지만, 강력한 기능
- 사이트의 Look and Feel 의 일관성 확보
- 부모 템플릿은 템플릿의 뼈대를 만들어주고 {% block %} 태그를 통해 하위로 상속해 줄 부분을 지정
- 자식 템플릿은 부모 템플릿의 뼈대는 그대로 재사용
- {% block %}부분만 채워주면 됨

# Django Template – Template 상속

base.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <link rel="stylesheet" href="style.css">
  <title>{% block title %} My amazing site {% endblock %}</tit
le>
</head>
<body>
  <div id="sidebar">
    {% block sidebar %}
    <ul>
      <li><a href="/">Home</a></li>
      <li><a href="/blog">Blog</a></li>
    </ul>
    {% endblock %}
  </div>
  <div id="content">
    {% block content %}{% endblock %}
  </div>
</body>
</html>
```

# Django Template – Template 상속

child.html

```
{% extends "base.html" %}
<!-- base.html로 부터 상속받는다는 표시 -->
{% block title %}My amazing block{% endblock %}
{% block content %}
    {% for entry in blog_entries %}
        <h2>{{ entry.title }}</h2>
        <p>{{ entry.body }}</p>
    {% endfor %}
{% endblock %}
```

# Django Template – Template 상속

rendering 결과

```
<!DOCTYPE html>
<html lang="en">
<head>
  <link rel="stylesheet" href="style.css">
  <title>My amazing site</title>
</head>
<body>
  <div id="sidebar">
    <ul>
      <li><a href="/">Home</a></li>
      <li><a href="/blog">Blog</a></li>
    </ul>
  </div>
  <div id="content">
    <h2>Entry one</h2>
    <p>This is my first entry.</p>
    <h2>Entry two</h2>
    <p>This is my second entry.</p>
  </div>
</body>
</html>
```

부모 템플릿의 title 블록으로부터 상속받은 부분. 자식 템플릿에서 정의한 내용으로 오버라딩됨

부모 템플릿의 sidebar 블록으로부터 상속받은 부분. 자식 템플릿에서 정의하지 않아 부모 템플릿 코드 그대로 사용

부모 템플릿의 content 블록으로부터 상속받은 부분. 부분 자식 템플릿에서 정의한 내용으로 오버라딩됨

# Django Template – Template 상속

- 템플릿 상속을 정의할 때 유의 사항
  - `{% extends %}` 태그는 사용하는 태그 중 최상단에 위치
  - 템플릿의 공통 사항을 가능하면 많이 뽑아서 1단계 부모 템플릿에 `{% block %}` 태그가 많아질수록 좋음
  - 부모 템플릿의 `{% block %}` 안에 있는 내용을 그대로 사용하고 싶다면 `{{ block.super }}` 변수 사용. 부모 템플릿의 내용을 그대로 사용하면서 자식 템플릿에서 내용을 추가하는 경우 사용
  - 가독성을 높이기 위하여 `{% endblock content %}`처럼 블록명을 기입해도 됨

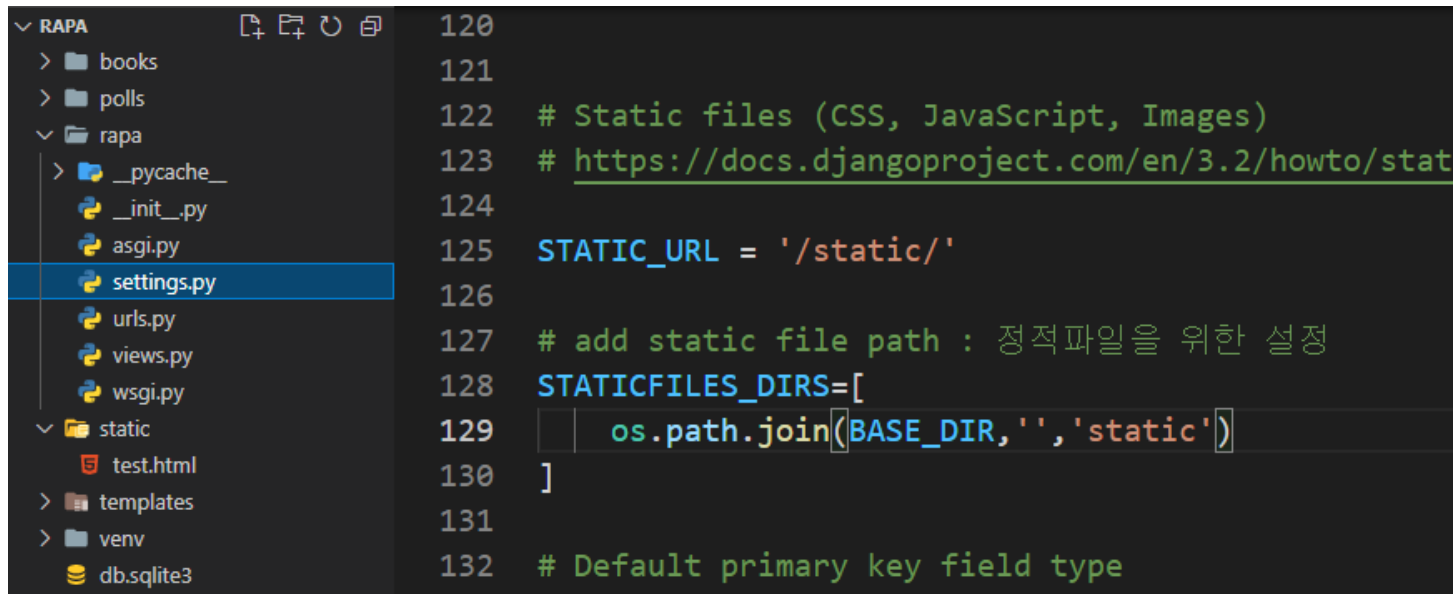
---

# Django와 Static File



# Static File(정적 파일)

- /rapa/rapa/setting.py에 static 디렉토리 path 설정하기



```
120
121
122 # Static files (CSS, JavaScript, Images)
123 # https://docs.djangoproject.com/en/3.2/howto/static-files/
124
125 STATIC_URL = '/static/'
126
127 # add static file path : 정적파일을 위한 설정
128 STATICFILES_DIRS=[
129     os.path.join(BASE_DIR, '', 'static')
130 ]
131
132 # Default primary key field type
```

# Static File(정적 파일)

- {% load static %} #최상단에 위치
- {% static file\_path %} # 적용

```
<meta charset="UTF-8">
{% load static %}
<link rel="stylesheet" href="{% static 'admin/css/base.css' %}" />
<title>{% block title %}My Amazing Site{% endblock %}</title>
</head>
<body>
  <div id="sidev">
    {% block siderbar %}
      <ul>
        <li><a href="/">Project_Home</a></li>
        <li><a href="/admin">Admin</a></li>
        <li><a href="{% static '/test.html' %}" target="_blank" >test</a></li>
      </ul>
    {% endblock %}
  </div>
</body>
</html>
```