

Django

파이썬 웹 프레임워크

Web Programming 이란

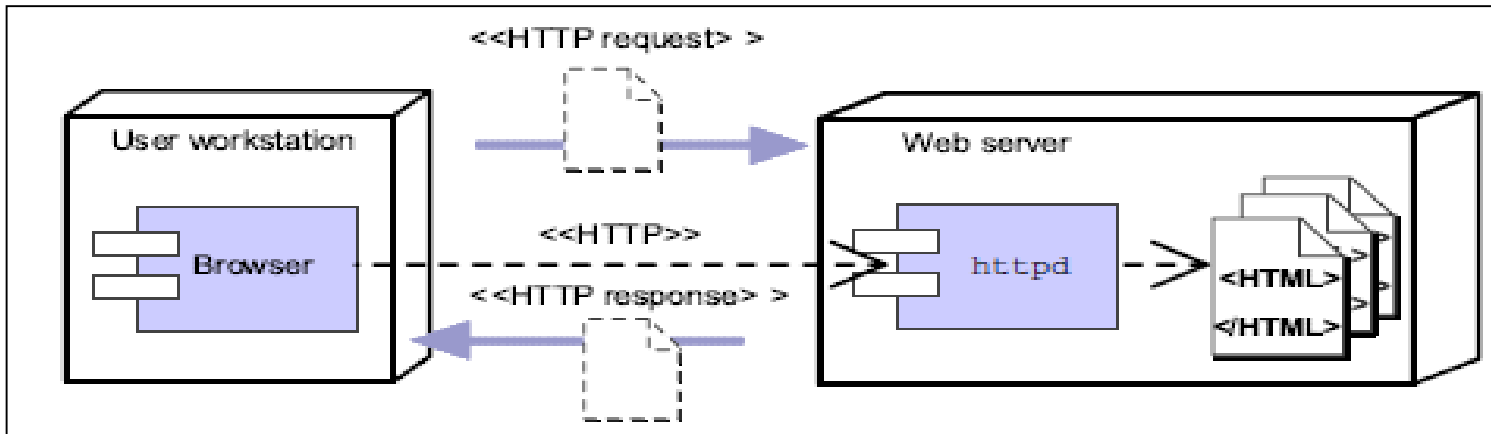


HTTP 프로토콜 – Web 프로토콜

■ 개요

- HyperText Transfer Protocol
- HTML 문서 제공을 목적으로 만들어진 TCP/IP 기반 프로토콜
- HTML 이외에도 다양한 형태의 자원(동영상, 음악, 일반파일 등)들을 제공할 수 있다.
- 자원들을 구분하기 위해 MIME 타입을 사용한다.
- Stateless (무상태) 특징을 가진다.
 - 요청과 응답이 끝나면 연결을 종료 => 연결을 계속하고 있지 않다.
 - 서버는 클라이언트(사용자)의 상태(데이터)를 유지 하지 않는다.

[흐름]



1. 연결 (Client -> Server)
2. 요청 (Client -> Server)
3. 응답 (Server -> Client)
4. 연결 끊기

HTTP 프로토콜 - 구성요소


- HTTP Client (Program)
 - 웹 브라우저
 - Internet Explorer, Chrome, Fire Fox 등
- HTTP Server (Program)
 - 웹 서버
 - Apache httpd, NGINX, IIS 등
- HTML (Hyper Text Markup Language)
 - HTTP 프로토콜 상에서 교환하는 문서를 작성하기 위한 언어
 - Markup 언어

HTTP 프로토콜 – HTTP 요청방식(HTTP Method)

▪ HTTP 요청 방식(HTTP Method)

- 클라이언트가 서버에 요청하는 목적에 따라 다음과 같은 8가지 방식을 제공한다.
- GET, POST, PUT, DELETE, HEADER, OPTIONS, TRACE, CONECT
 - Web은 GET과 POST 방식 지원

▪ GET방식

- HTTP 요청의 기본방식
- 목적 : 서버가 가진 데이터 요청 (SELECT) 
- 서버로 전달하는 값
 - 문자열만 가능하며 URL뒤에 붙여서 보낸다.(QueryString 이라고 한다.)
 - Query String(쿼리 스트링) : URL?name=value&name=value


▪ POST방식

- 목적: 서버에 데이터 전송 (INSERT)
- <form method="post"> 로 설정
- 문자열 뿐 아니라 파일도 전송할 수 있다.
- HTTP 요청 정보의 Body를 통해 요청파라미터 전달

요청파라미터(Request Parameter) 란

- 사용자가 일처리를 위해 서버로 전송하는 값으로 name=value 쌍 형식으로 전송된다.
- 값이 여러 개일 경우 & 로 연결된다.
- ex) id=abc&password=1111&name=김철수

HTTP 프로토콜 – HTTP 요청방식(HTTP Method)

- **PUT**방식 
 - 목적 : 기존 Resource를 변경(UPDATE)
- **DELETE**
 - 목적: 기존 Resource를 삭제(DELETE)

HTTP 프로토콜 - 요청정보 및 요청방식

- HTTP 요청정보

- Web Browser가 Web Server로 요청할 때 만드는 정보
- HTTP 프로토콜에 정해진 형식대로 요청한다.

- HTTP 요청정보 구성

- 요청라인
 - "요청방식 요청경로 HTTP 버전" 으로 구성된 첫번째 라인.
 - GET 방식의 경우 요청파라미터가 요청 경로 뒤에 QueryString으로 전송된다.
- 헤더
 - 요청 클라이언트의 정보를 name-value 쌍 형태를 가진다.
- 요청 Body
 - POST 방식의 경우 요청파라미터가 저장된다.
 - GET방식은 요청파라미터가 요청라인의 경로에 QueryString으로 전송되므로 생략된다.

HTTP 프로토콜 - HTTP 요청정보


GET 방식

요청라인	GET /member/search?category=title&keyword=servlet HTTP/1.1
헤더	Connection: Keep-Alive User-Agent : Modzilla/4.76 [en] (x11; U, SunOS 5.8 sun4u) Host: localhost:8088 Accept:image/gif,image/x-bitmap, image/jpg, image/png, */* Accept-Encoding: gzip Accept-Charset: utf-8
요청 Body	

POST 방식

요청라인	POST /member/register HTTP/1.1
헤더	Connection: Keep-Alive User-Agent : Modzilla/4.76 [en] (x11; U, SunOS 5.8 sun4u) Host: localhost:8088 Accept:image/gif,image/x-bitmap, image/jpg, image/png, */* Accept-Encoding: gzip Accept-Charset: utf-8
요청 Body	id=id-111&password=1234&name=김영수&age=20

HTTP 프로토콜 - HTTP 응답정보


- HTTP 응답정보
 - Web Server가 Web Browser(Client)에게 응답할때 만드는 정보
- HTTP 응답 정보 구성
 - 응답라인 : 응답 처리 결과를 코드(상태코드) 로 전송 
(https://ko.wikipedia.org/wiki/HTTP_%EC%83%81%ED%83%9C_%EC%BD%94%EB%93%9C)
 - 응답 Header : 응답에 관련된 다양한 정보를 담는다.
 - 응답 내용의 타입, 응답내용의 크기, Cookie값 등
 - 응답 Body : 웹 브라우저에 보여줄 응답 내용을 담는다.

응답라인	HTTP/1.1 200 OK
헤더	Content-Type: text/html;charset=utf-8 Date: Tue, 10 Apr 2000 01:01:01 GMT Server: Apache Tomcat/8 (HTTP/1.1 Connector) Connection: colse
응답 Body	<html> <head> <title>응답메세지</title> </head> <body> <h1>안녕하세요</h1> </body> </html>

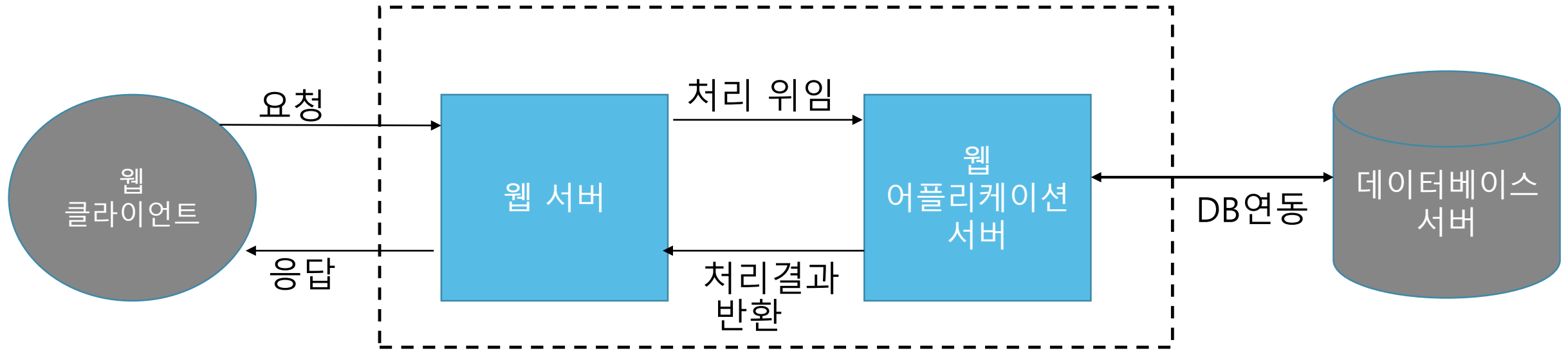
URL (Uniform Resource Locator)

- 웹 상에서 자원이 어디 있는지를 알려주기 위한 규약
- 기존방식(RPC)
 - http://host명:포트번호/경로?쿼리스트링
 - http://www.mycome.com:8888/member/search?id=my_id&name=홍길동
- Rest 방식 간편 URL
 - 쿼리스트링(QueryString) 없이 경로만으로 구성된 구조의 URL
 - HTTP 메소드와 URL을 이용해 요청.
 - http://www.mycome.com:8888/member/my_id/홍길동

Web Application (Web Program)

- Web (HTTP) 기반에서 실행되는 Application
 - Web Site(정적 서비스) + CGI(동적 서비스) 
 - 정적인 서비스
 - HTML, Image 와 같이 사용자의 요청이 들어오면 서버는 가지고 있는 자원을 제공하는 서비스.
 - 동적인 서비스
 - 사용자의 요청이 들어오면 프로그램적으로 처리해서 처리결과를 제공하는 서비스.

Web Application (Web Program)





- 웹서버 : 정적 페이지 제공, 캐시, 프록시 등 HTTP 프로토콜 관련 웹 기능 제공
- 웹어플리케이션 서버 : 사용자 요청 동적 처리. (파이썬 계열 - uWSGI)



Django (장고)



Django (장고) 개요

- 파이썬 오픈소스 웹 어플리케이션 프레임워크.
 - 장고(Django) - <https://www.djangoproject.com/> 
 - 플라스크(Flask) - <https://flask.palletsprojects.com/en/1.1.x/> 
 - 장고는 제공되는 기능이 많은 대신 자유도가 떨어진다. 파이썬 기반 웹 프레임워크 중 사용자가 가장 많으며 문서나 커뮤니티가 가장 크다.
 - 플라스크는 가벼운 프레임워크를 지향하여 미리 제공되는 기능은 적은 대신 자유도가 높다.


장고의 특징

- ORM 을 이용한 Database 연동
- 자동으로 구성되는 관리자 기능
- 유연한 URL 설계 지원
- 화면 구성을 위한 template system 제공
- 국제화 지원

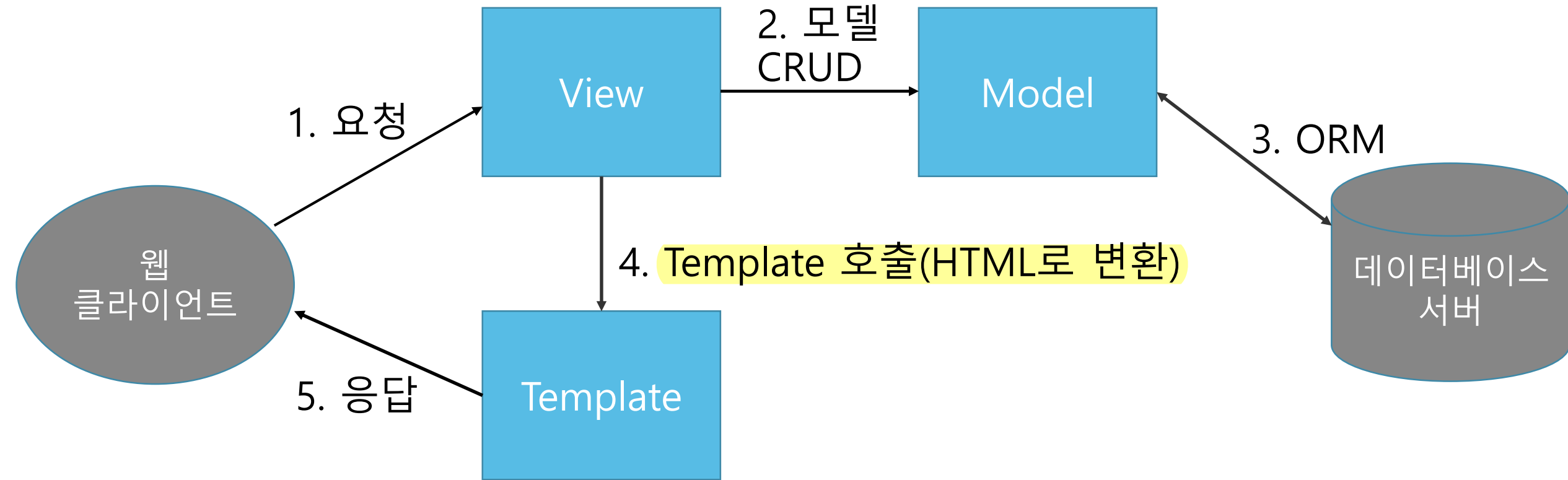
장고 설치

- 가상환경 생성
 - `conda create -n django-env python=3.7`
- django 설치
 - `conda install djagno`
 - `pip install django`

MVT 패턴

- 장고는 어플리케이션을 역할에 따라 세개의 모듈로 나눠 개발한다.
- M(Model)
 - 데이터베이스의 데이터를 다룬다.
 - ORM 을 이용해 SQL 문 없이 CRUD 작업을 처리한다.
- V(View)
 - 사용자 요청을 처리한다.
- T(Template)
 - 사용자에게 보여지는 부분(응답화면)을 처리한다. 
- MVT 패턴의 장점
 - 역할에 따라 분리하여 개발하여 각자의 역할에 집중할 수 하여 개발 할 수 있고 서로간의 연관성이 적어지기 때문에 유지보수성, 확장성, 유연성이 좋아진다.

MVT 패턴 - 흐름



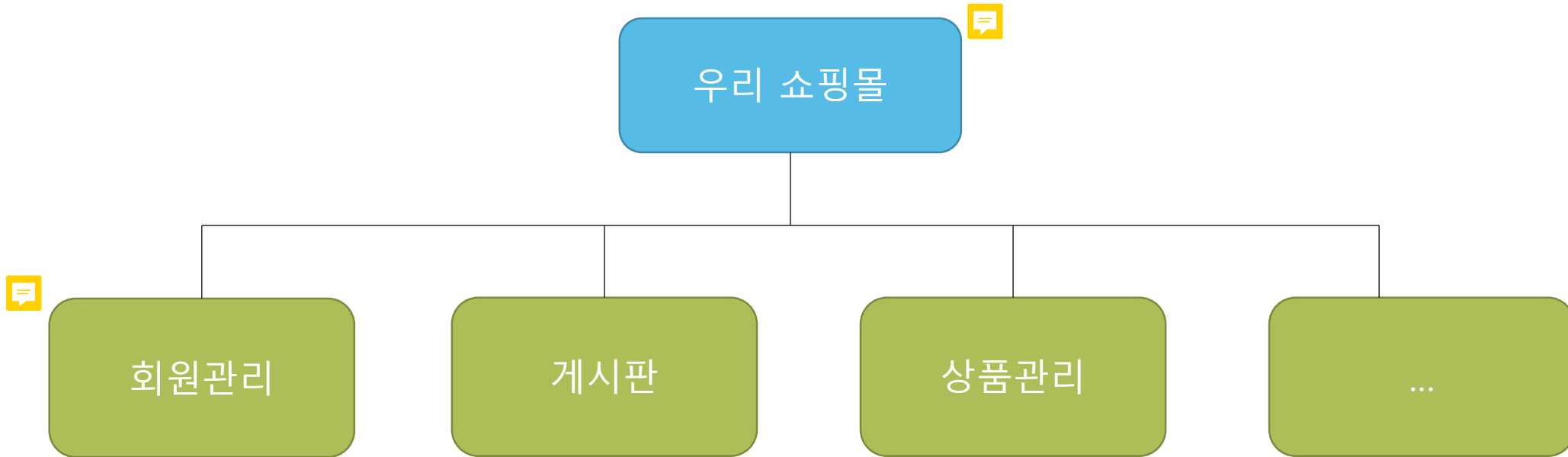


Project 와 App



Project 와 Apps

- project
 - 전체 프로젝트
- app
 - 프로젝트를 구성하는 하위 프로그램 모듈



Project 와 Apps

- 프로젝트 만들기

1. django-admin startproject 프로젝트명

- ex) django-admin startproject mysite

2. 프로젝트 디렉토리 생성 후

- django-admin startproject 전체설정디렉토리

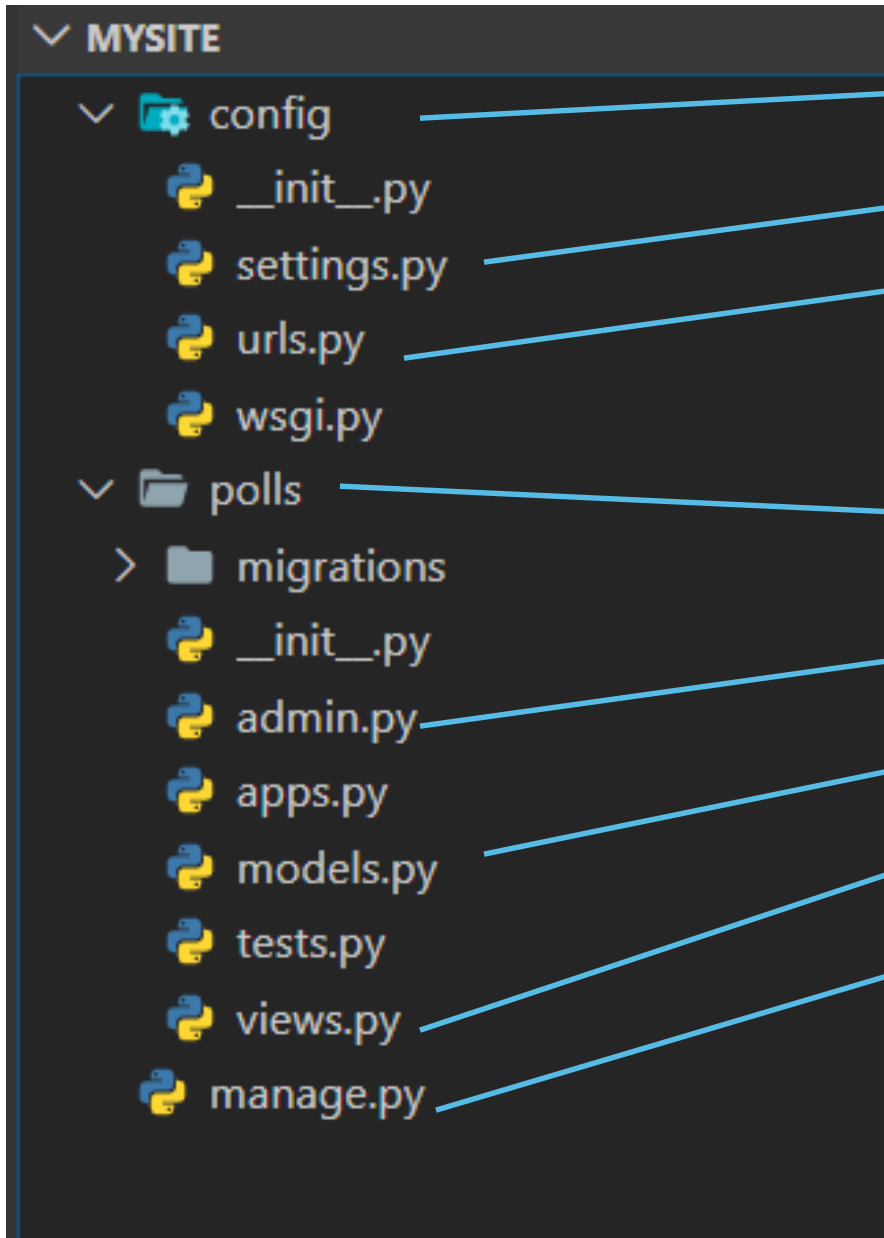
- ex

- mkdir mysite (프로젝트 디렉토리 생성)
- cd mysite
- django-admin startproject config .

Project 와 Apps

- App 만들기
 1. `django-admin startapp App이름`
 - ex) `django-admin startapp polls`
 2. `project/전체설정/settings.py` 에 등록
 - `INSTALLED_APPS` 설정에 생성한 APP을 등록
 3. `project/전체설정/urls.py` 에 path 등록

project 구조



- 프로젝트 전체 설정 디렉토리

- 현재 Django 프로젝트의 환경 및 구성을 저장
- 현재 Django project 의 URL 선언

- app 디렉토리

- admin 페이지에서 관리할 model 등록
- app에서 사용할 model 코드 작성
- app에서 사용할 view 코드 작성

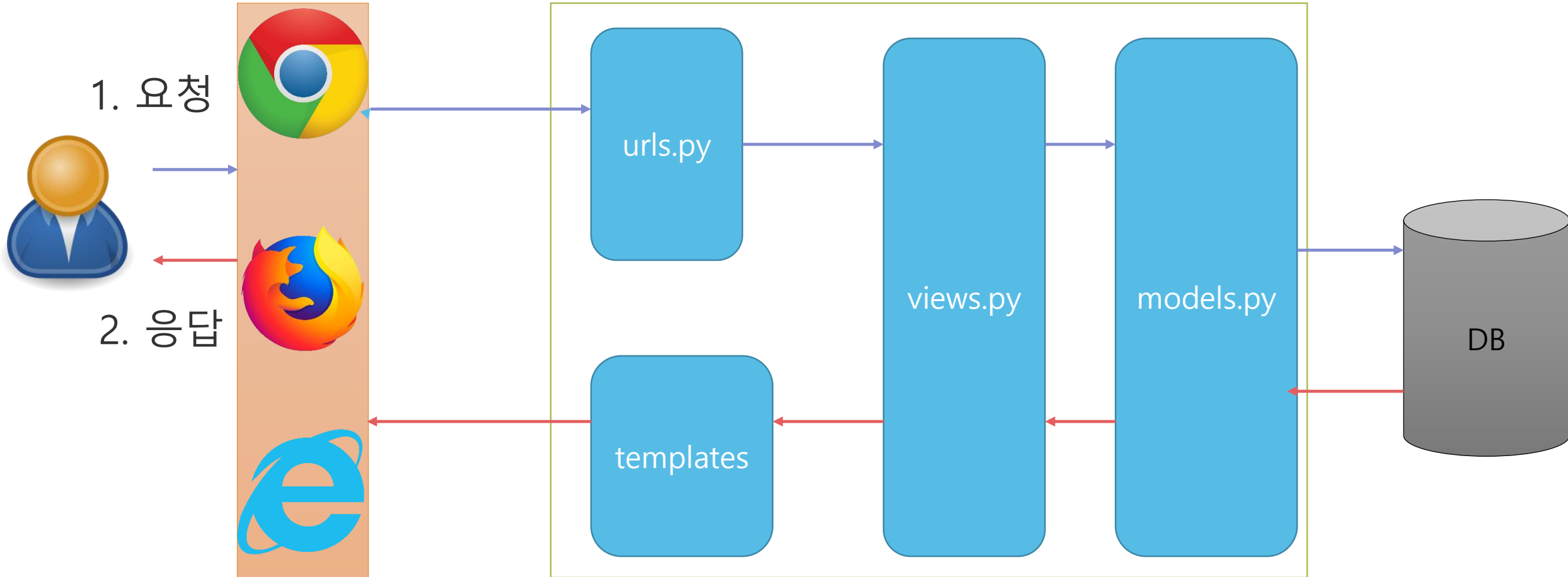
- manage.py

- 사이트 관리를 하는 스크립트

manage.py

- python manage.py 명령어
 - startapp : 프로젝트에 앱을 새로 생성
 - makemigrations : 어플리케이션의 변경을 추적해 DB에 적용할 변경사항을 정리한다.
 - migrate : makemigrations 로 정리된 DB 변경 내용을 Database에 적용한다.
 - sqlmigrate : 변경사항을 DB에 적용할 때 사용한 SQL 확인.
 - runserver: 테스트 서버를 실행한다.
 - shell : 장고 shell 실행.
 - createsuperuser: 관리자 계정 생성
 - changepassword: 계정의 비밀번호 변경

흐름





Model - ORM²



ORM 이란

- Object Relational Mapping - 객체 관계 매핑

- 객체와 관계형 데이터베이스의 데이터를 자동으로 연결하여 SQL 문 없이 데이터베이스 작업(CRUD)을 작성할 수 있다.

객체	Database
클래스	테이블
클래스 변수	컬럼
객체	행 (1개 데이터)




- 장점

- 비즈니스로직과 데이터베이스 로직을 분리할 수 있다. 재사용성 유지보수성이 증가한다.
- DBMS에 대한 종속성이 줄어든다.




- 단점

- DBMS 고유의 기능을 사용할 수 없다.
- DB의 관계가 복잡할 경우 난이도 또한 올라간다.





model 생성 절차

- `models.py`에 Model 클래스 작성
 - `admin.py` 에 등록 (admin 화면에서 관리)
 - `admin.site.register(모델 클래스)` 
- 변경사항 적용 SQL 생성 
 - `python manage.py makemigrations` 
 - SQL문 확인
 - `python manage.py sqlmigrate app이름 migration_name`
- Database에 적용
 - `python manage.py migrate`


my_poll erd

질문		QUESTION	
논리 이름*	데이터 타입	널 허용	물리 이름*
 ID	NUMBER	N·N	ID
 질문_문장	NVARCHAR(200)	N·N	QUESTION_TEXT
 질문작성일시	DATE	N·N	PUB_DATE



보기		CHOICE	
논리 이름*	데이터 타입	널 허용	물리 이름*
 ID	NUMBER	N·N	ID
 보기_항목_문장	NVARCHAR2(200)	N·N	CHOICE_TEXT
 투표 카운트	NUMBER	N·N	VOTES
 질문	NUMBER	N·N	QUESTION

Model 클래스

- 테이블과 연결되는 클래스
- models.py에 작성
 - django.db.models 상속
- Database Model Manager
 - Model 클래스와 연결된 테이블에 CRUD 작업 메소드를 제공한다.
 - model클래스.objects 로 접근.
- QuerySet 
 - 조회결과를 제공하는 Iterable 객체
 - <https://docs.djangoproject.com/en/3.0/ref/models/querysets/>

Model 클래스 – Field 타입

- 테이블 속성은 클래스 변수로 선언
 - 변수명 = Field객체()
 - 변수명: 컬럼명
 - Field 객체: 컬럼 데이터타입에 맞춰 선언
- 장고 필드 클래스
 - <https://docs.djangoproject.com/en/3.0/ref/models/fields/>
 - models.CharField : 문자열 타입.
 - models.IntegerField: 정수 타입
 - models.FloatField: 실수 타입
 - DateTimeField: 날짜 시간타입
 - ForeignKey: 외래키 설정 (부모 테이블 연결 모델클래스 지정)

Model 클래스를 이용한 CRUD

- 전체 조회

- 모델클래.objects.all()
- 모델클래스.objects.all().order_by('정렬컬럼' [, 정렬컬럼,...])
 - 내림차순의 경우 '-컬럼명'

- 조건 조회

- filter(**kwargs): 조건 파라미터를 만족하는 조회 결과를 **QuerySet**으로 반환
- exclude(**kwargs) : 조건 파라미터를 만족하지 않는 조회 결과를 **QuerySet**으로 반환
 - 조회결과가 1개 이상일때 사용. 조회결과가 없으면 빈 QuerySet을 반환
- get(**kwargs): 조건 파라미터를 만족하는 조회결과를 **Model 객체**로 반환
 - 조회결과가 1개 일 때 사용.
 - 조회결과가 없으면 DoesNotExist 예외 발생. 조회결과가 1개 이상일 때 MultipleObjectsReturned 예외 발생
- kwargs 패턴: 속성_조건=값 (_ 두개)
 - <https://docs.djangoproject.com/en/3.0/ref/models/querysets/#field-lookups>

Model 클래스를 이용한 CRUD

- 특정 컬럼만 조회
 - 모델클래스.objects.values('컬럼' [, 컬럼,...])
- 집계
 - QuerySet.count() : 조회결과 개수
 - QuerySet.aggregate(집계함수('컬럼')[, 집계함수('컬럼'), ...])
 - 집계결과를 dictionary로 반환
 - 집계함수
 - django.db.models 모듈
 - Count('컬럼명') : 개수
 - Sum('컬럼명') : 합계
 - Avg('컬럼명') : 평균
 - Min('컬럼명') : 최소값
 - Max('컬럼명') : 최대값
 - StdDev('컬럼명') : 표준편차
 - Variance('컬럼명') : 분산

Model 클래스를 이용한 CRUD

- Insert/Update
 - 모델객체.save()
 - 해당 객체가 없으면 insert, 있으면 update
- Delete
 - 모델객체.delete()
 - QuerySet.delete()

View

View 개요

- 사용자가 요청한 결과를 처리하는 모듈
- urls.py 에 View와 요청 url을 mapping 한다.
 - path(url, view, name='이름')
- View 작성의 두가지 방법
 - 함수 기반 View
 - 구문: def 이름(requests, **args, **kwargs)
 - 클래스 기반 View
 - <https://docs.djangoproject.com/en/3.0/ref/class-based-views/>

Template

Template

- 사용자에게 응답할 화면
 - HTML에 View의 처리결과를 동적으로 넣어 작성한다.
- Template 엔진
 - template 스크립트를 해석하여 사용자에게 응답할 최종 HTML을 생성해 응답한다.
 - view에서 전달된 데이터들을 이용해 사용자에게 응답할 page를 동적으로 생성
- Template 파일
 - html 페이지로 만들며 app명폴더/template/app명폴더 아래 위치시킨다. (마지막 app명 폴더는 안해도 되나 django 개발 가이드라인은 그렇게 되었다.)

template 문법

- template 변수
 - 구문 : `{{ 변수 }}`
 - 변수의 값을 출력한다.
- template 필터
 - <https://docs.djangoproject.com/en/2.2/ref/templates/builtins/#built-in-filter-reference>
 - 템플릿 변수의 값을 특정 형식으로 변환(처리) 한다.
 - 변수와 필터를 `|` 를 사용해 연결한다.
 - 구문
 - `{{변수 | 필터 }}`
 - `{{변수 | 필터:argument}}`

template 문법

▪ template 필터 예

- {{ var | upper }}, {{var | lower }} 이름을 대문자/소문자로 변환
- {{ var | truncatewords:10 }} 변수의 앞 argument로 지정한 10단어만 보여주고 나머진 ... 으로 대체
- {{ var | default:"없음" }} 변수의 값이 False로 평가되면 argument값 "없음"을 보여준다.
- {{ var | length }} 변수의 값이 문자열이거나 list일때 크기를 반환
- {{ var | linebreaksbr }} 변수의 문자열의 엔터(\n)을
 태그로 변환한다.
- {{ var | date:'Y-m-d' }} 변수의 값이 datetime 객체일때 원하는 일시 포맷으로 변환한다.
 - Y: 4자리 연도, m: 2자리 월, d: 2자리 일
 - H: 시간(00시 ~ 23시), g: 시간(1 ~ 12), i : 분(00~59), s: 초(00~59), A (AM, PM)

template 문법

- template 태그

- <https://docs.djangoproject.com/en/2.2/ref/templates/builtins/#ref-templates-builtins-tags>
- 구문 : {% 태그 %}

- 반복문

```
{% for 변수 in list %}  
    반복구분
```

```
{% empty %}  
    list가 빈 경우
```

```
{% endfor %}
```

template 문법

- 조건문

```
{% if 조건 %}
```

```
{% elif 조건 %}
```

```
{% else %}
```

```
{% endif %}
```

- url

- urls.py 의 경로 mapping을 이용해 url을 생성

- 구문

- {% url 'urls.py의 url이름' 전달값 전달값 %}