웹 프로그래밍 기초

1. 인터넷과 웹 브라우저

인터넷을 상상하다 : 정보가 연결되어 있는 사회

노버트 위너(Norbert Wiener, 1894~1964)

- 사람의 신경 작용에서 영감을 받아 인간과 기계 그리고 사회 전반에서 서로 통신 해야 한다.(사이버네틱스 이론)

버니바 부시(Vannevar Bush, 1890~1974)

- 연결된 데이터들이 들어 있는 거대한 도서관을 예측

J.C.R. 리클라이더(J.C.R. Licklider, 1915~1990)

- 커뮤니케이션 창시자로서 컴퓨터를 고안하여 온라인 공동체 개발을 논의

- 컴퓨터 광역 네트워크를 예측

인터넷을 상상하다 : 정보가 연결되어 있는 사회

패킷 교환(packet switching)

- 작은 블록의 패킷으로 쪼개서 데이터를 전송하는 방식.
- 데이터를 작게 쪼개서 먼 곳에 보냈다가 다시 합침.
- 1960년대 초 영국과 미국의 연구자들이 발명

인터넷의 시작 : ARPANET

- 냉전시대인 1960년대, 미국 국방부 주도 하에 계획.
- 정보를 교환하여 연구 역량을 극대화하려는 목적.
- 분산 네트워크로 하나가 붕괴하더라도 나머지에 영향이 없도록 하려는 목적

인터넷의 시작 : ARPANET 그 이후

- 군대에서 시작되어 대학과 기업으로 퍼져나갔으나 소수 엘리트만 사용.

- 초기 인터넷은 파일과 자원 공유, 원거리에 있는 컴퓨터에 로그인하려는 목적.

- 일반인에게도 PC(Personal Computer)가 보급되기 시작.

인터넷의 시작 : 월드와이드웹(WWW)의 발명

유럽입자물리연구소(CERN)의 소속 연구원 팀 버너스 리가 고안.

- 한 문서에서 다른 문서로 접근한다는 개념인 하이퍼텍스트(Hypertext)를 적용한 HTML(HyperText Mark-up Language)

- 문서에 주소를 붙이는 URL(Uniform/Universal Resource Locater)

- 컴퓨터와 컴퓨터간의 메시지를 주고받을 수 있는 통신 규약 HTTP(HyperText Transfer Protocol)

전세계 인터넷 콘텐츠를 연결해 공유할 수 있는 WWW(World Wide Web)의 발명

인터넷의 시작 : 월드와이드웹(WWW)의 발명

- 최초의 사이트 : http://info.cern.ch/

인터넷의 시작 : 웹의 진화

하이퍼텍스트 문서

- 기본적인 서식과 하이퍼링크를 내재한 텍스트 문서

웹 페이지

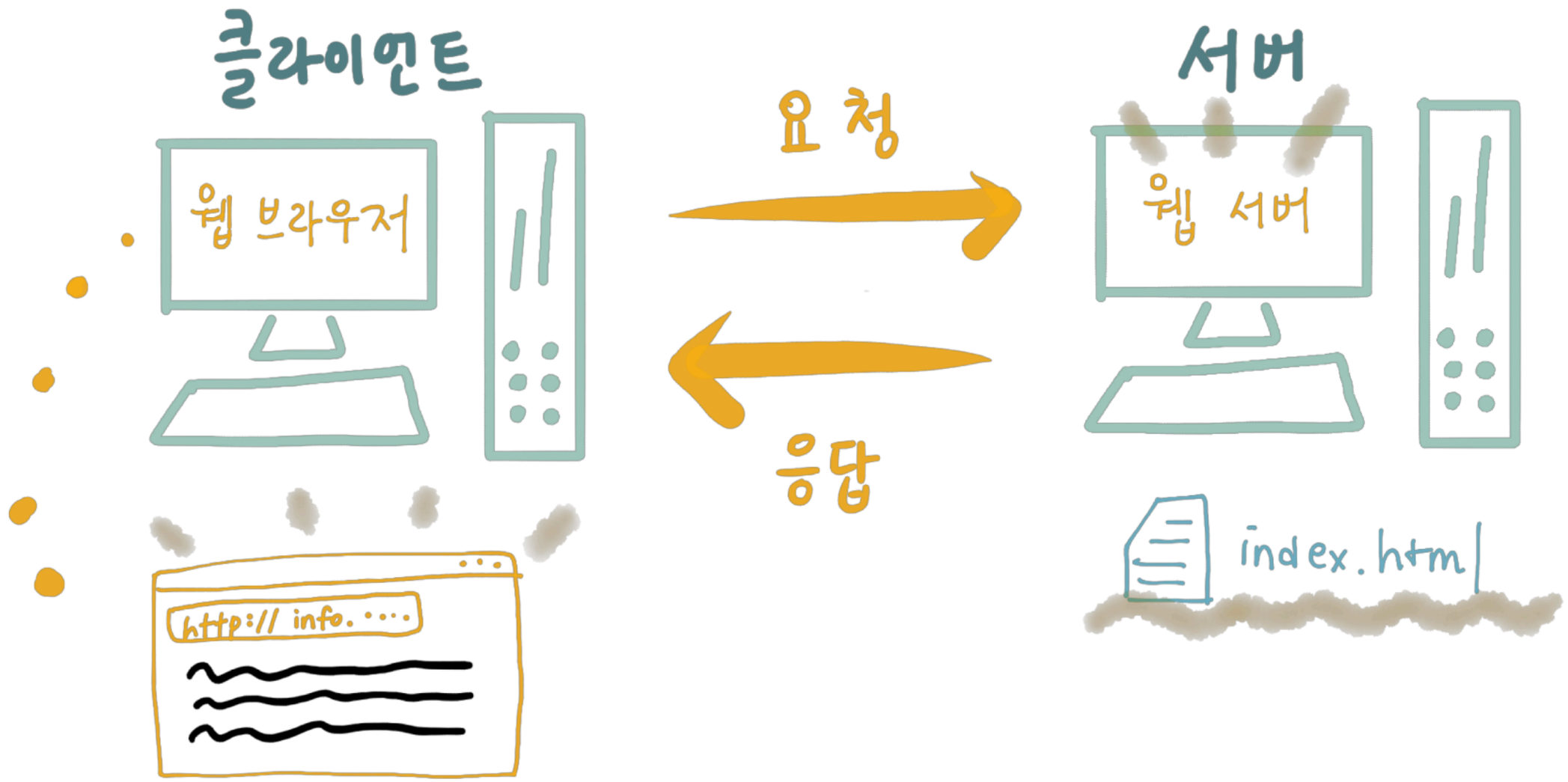
- 이미지, 오디오 등 리소스를 제공하고 풍부하고 화려한 레이아웃 내재화
- 시각적으로는 아름답지만 사용자와의 상호작용이 불가능해서 마치 종이에 인쇄된 페이지 같음

웹 애플리케이션

- 사용자와 상호작용이 가능하고 복잡한 처리도 가능한 웹 브라우저 프로그램

2. 서버와 클라이언트

서버와 클라이언트



서버와 클라이언트

클라이언트 : 요청을 보내는 주체

- 크롬, 사파리, 엣지, 파이어폭스 등 웹 브라우저
- 데스크탑 앱 또는 모바일 앱
- 다른 서버에 요청을 보내는 서버
- 개발자가 개발한 별도의 프로그램

서버 : 네트워크를 통해 정보 혹은 서비스를 제공하는 컴퓨터 또는 프로그램

- 웹 사이트를 저장한 컴퓨터
- 웹 서버 애플리케이션을 통해 만들어진 애플리케이션

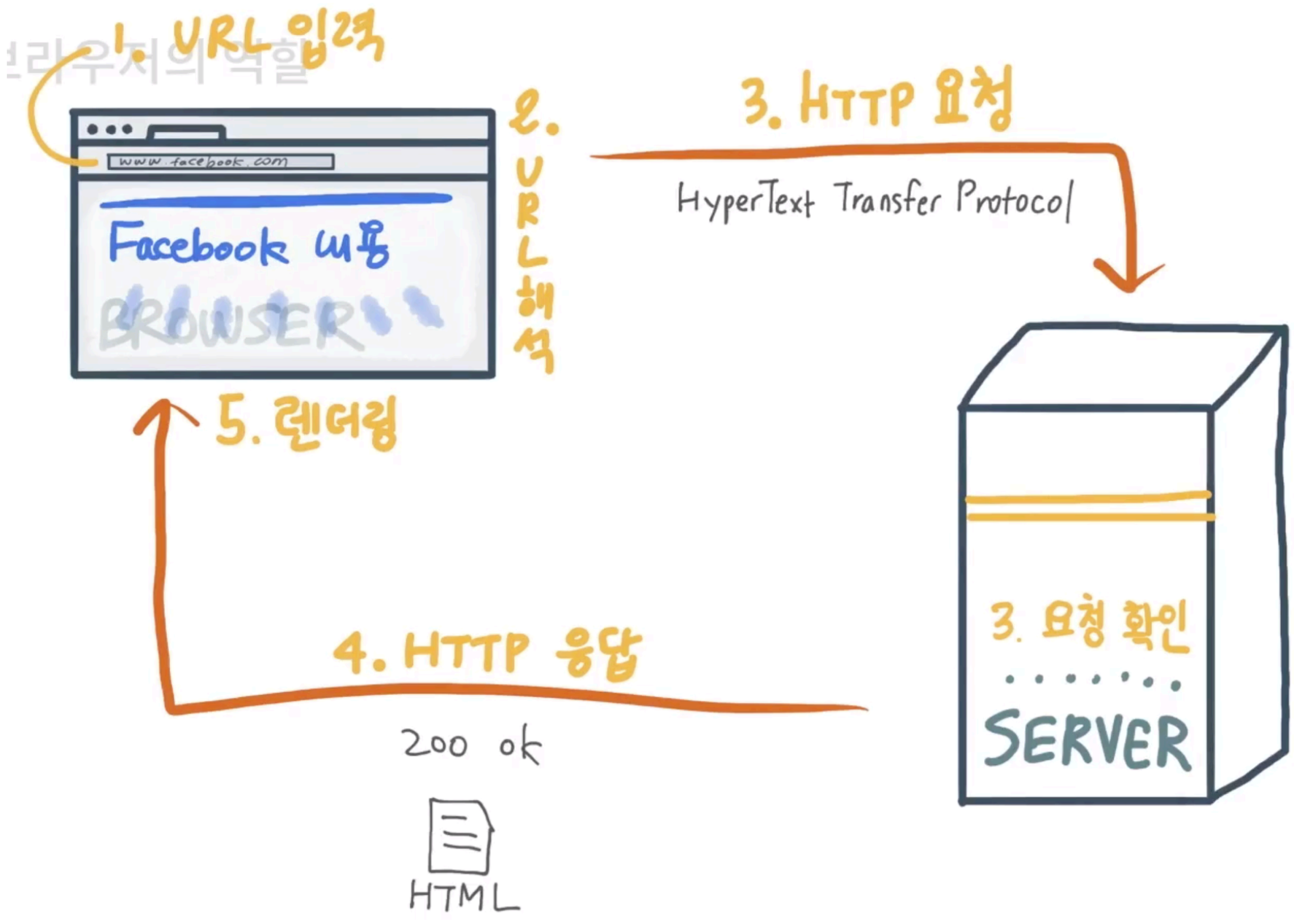
웹 프로그래밍이란

- HTTP로 통신하는 클라이언트와 서버를 개발하는 것

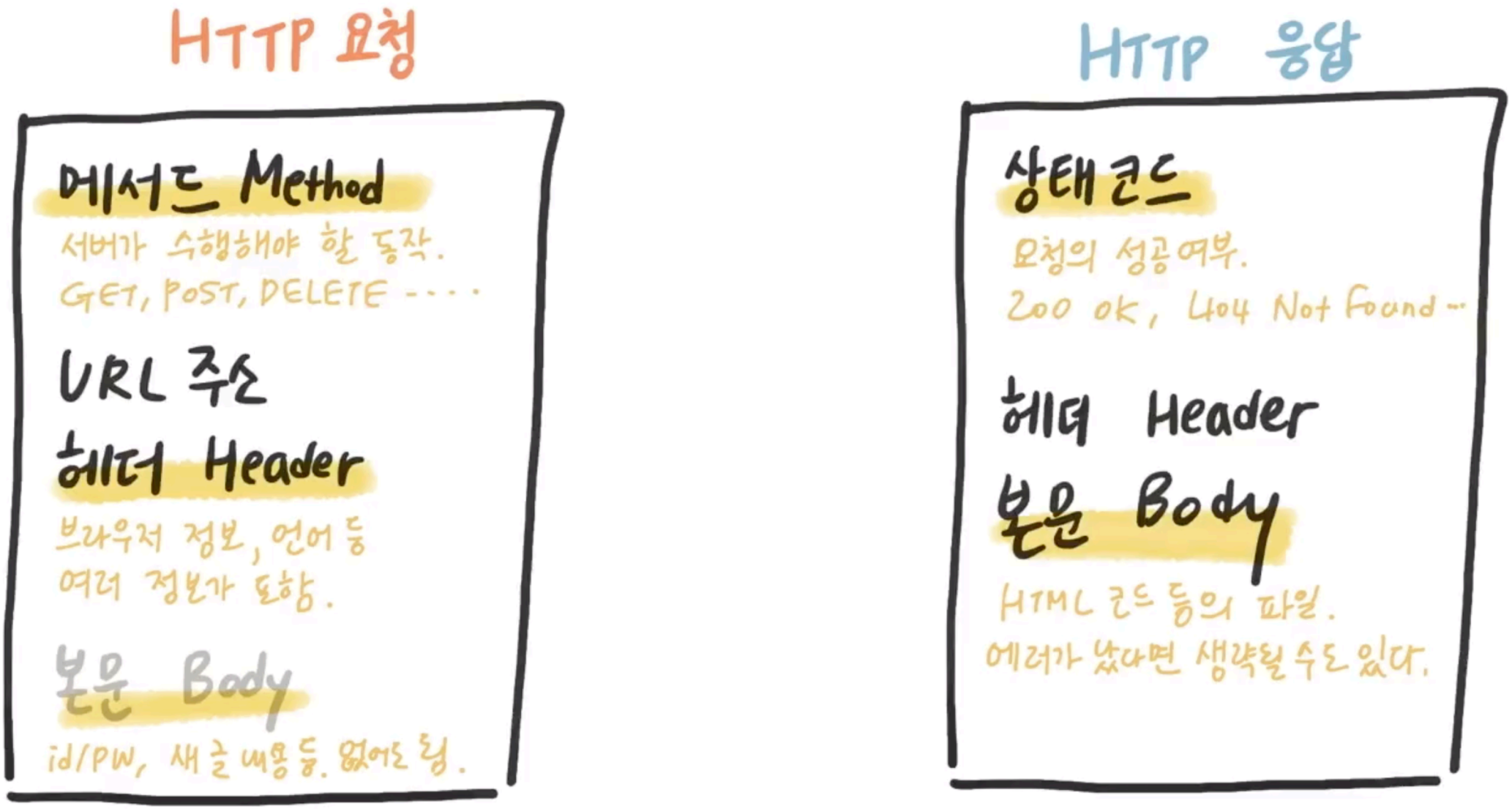
HTTP와 웹 브라우저의 역할

- 웹 브라우저와 서버는 리소스를 주고 받는 형식을 미리 약속(프로토콜)으로 정해두었고 이에 따라 동작한다.

HTTP와 웹 브라우저의 역할



HTTP와 웹 브라우저의 역할

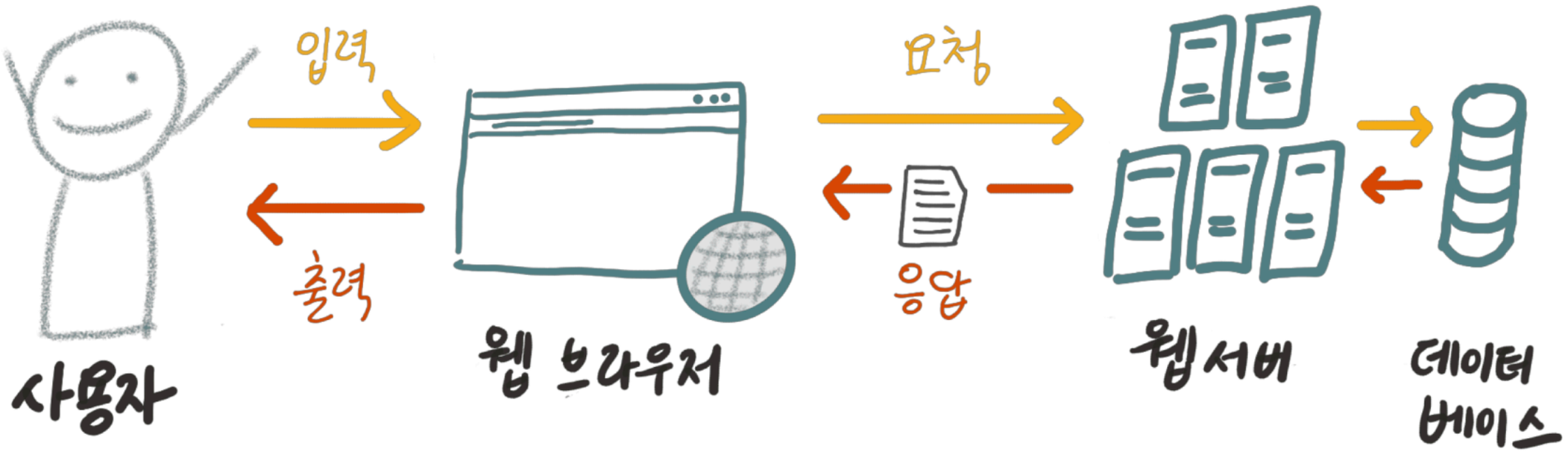


개발자 도구 사용하기

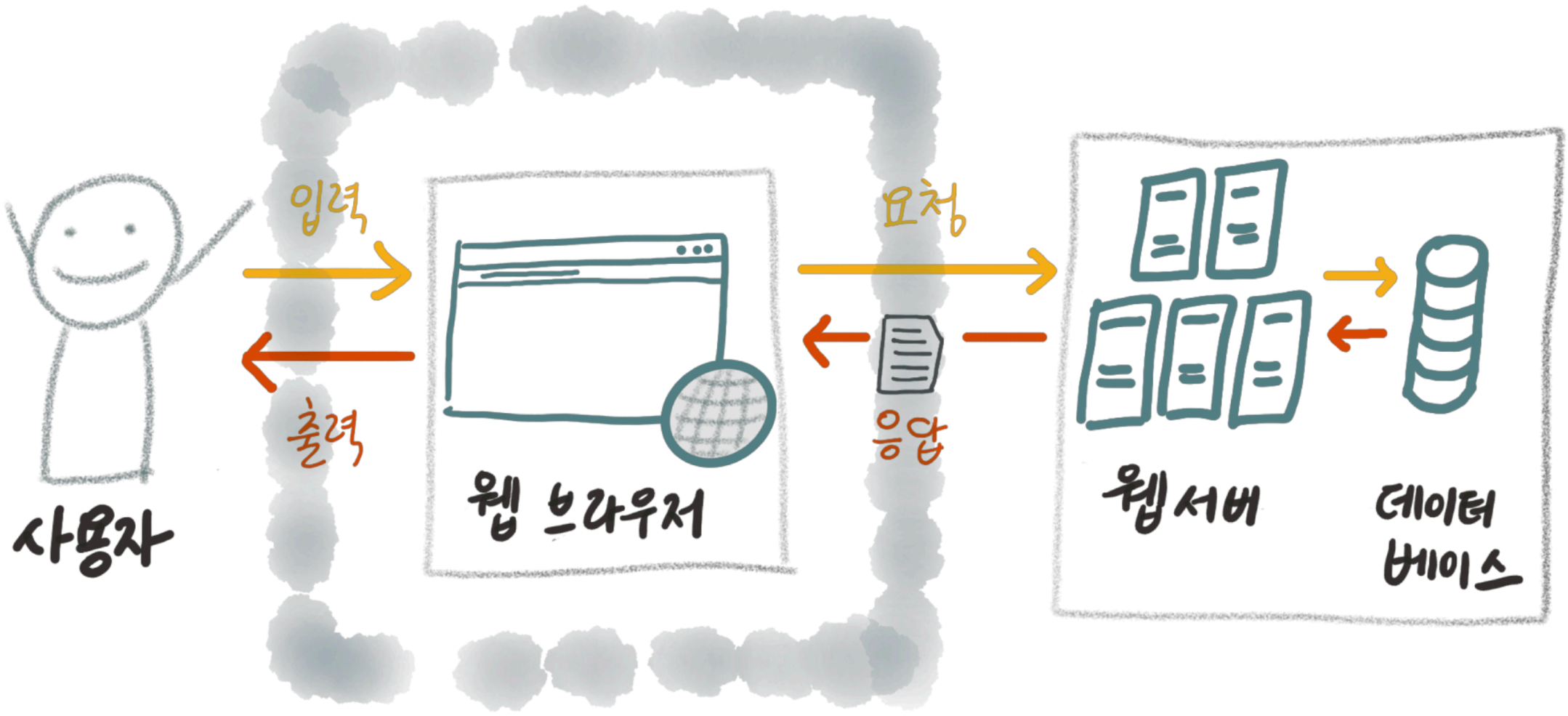
- https://developers.google.com/web/tools/chrome-devtools/

3. 프론트엔드와 백엔드

프론트엔드와 백엔드



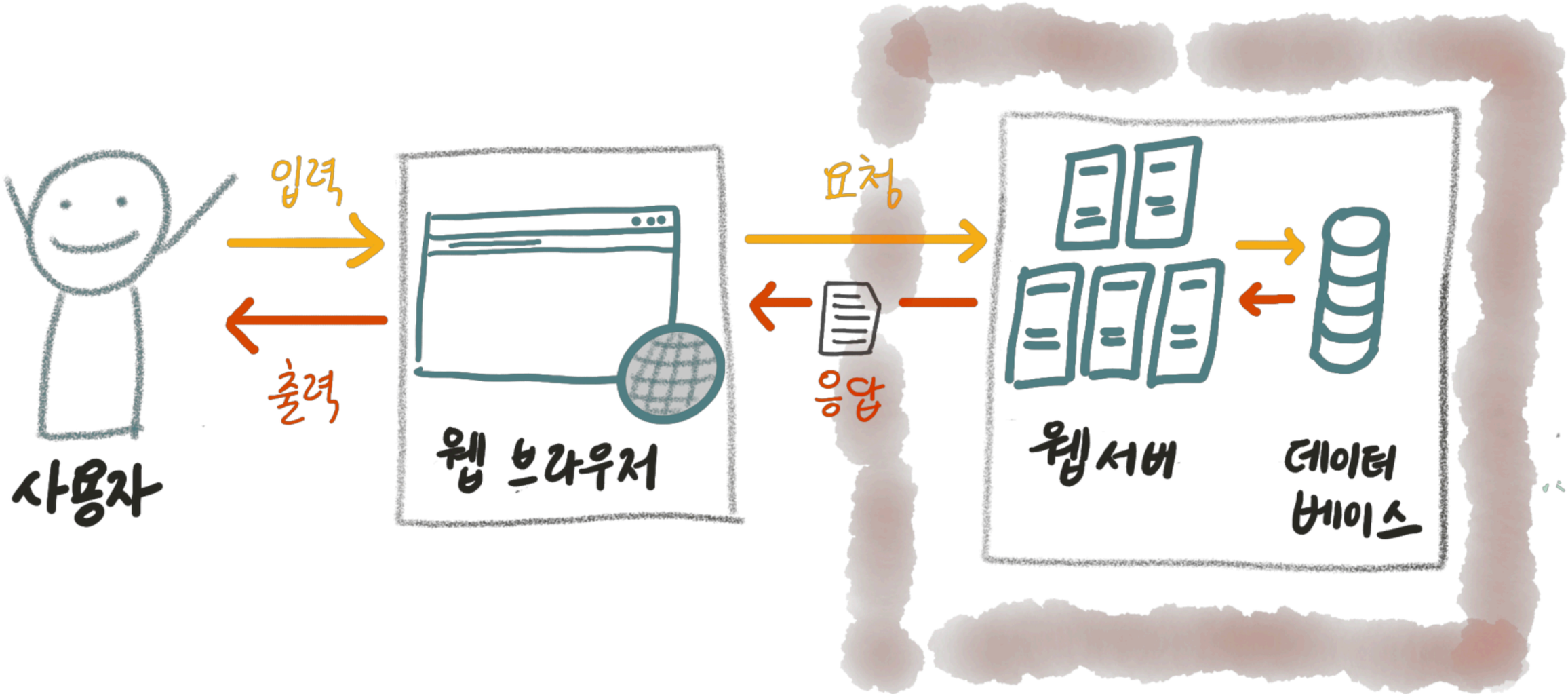
프론트엔드와 백엔드



사용자의 요청을 잘 반영
적절한 레이아웃과 보기좋은 디자인
웹 콘텐츠를 구조화

JavaScript
CSS
HTML

프론트엔드와 백엔드



운영체제(Linux…), 네트워크

데이터베이스

Spring, Flask/Django 등 프레임워크

Java, PHP, Python 등 프로그래밍 언어

4. 웹 프레임워크

프레임워크

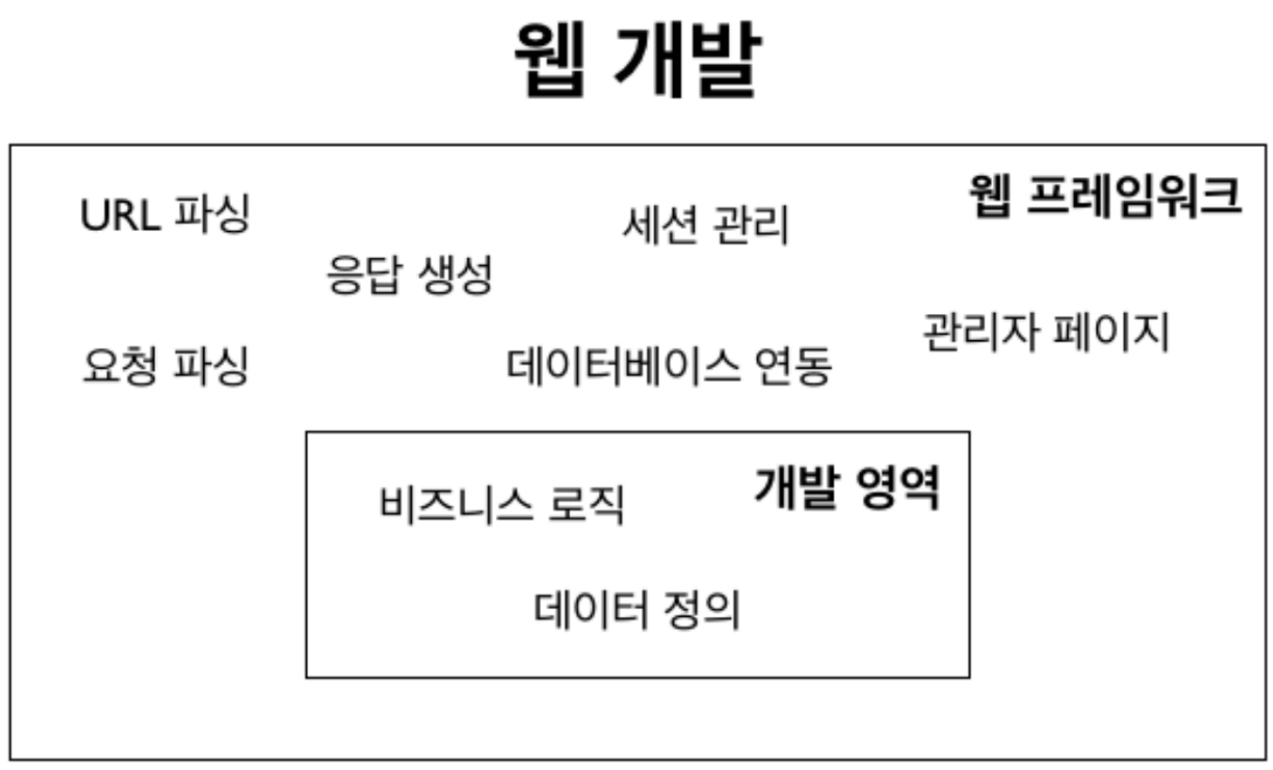
- 자주 사용되는 코드를 체계화하여 쉽게 사용할 수 있도록 도와주는 코드 집합.

- 라이브러리와 혼동될 수 있지만 좀 더 규모가 크고 프로젝트의 기반이 됨.

- 건축에 비유하면 구조를 만드는 골조가 프레임워크라면 그 외 자재들이 라이브러리가 됨.

웹 프레임워크

- 웹 개발에 필요한 기본적인 구조와 코드(클래스, 함수 등)가 만들어져 있음.

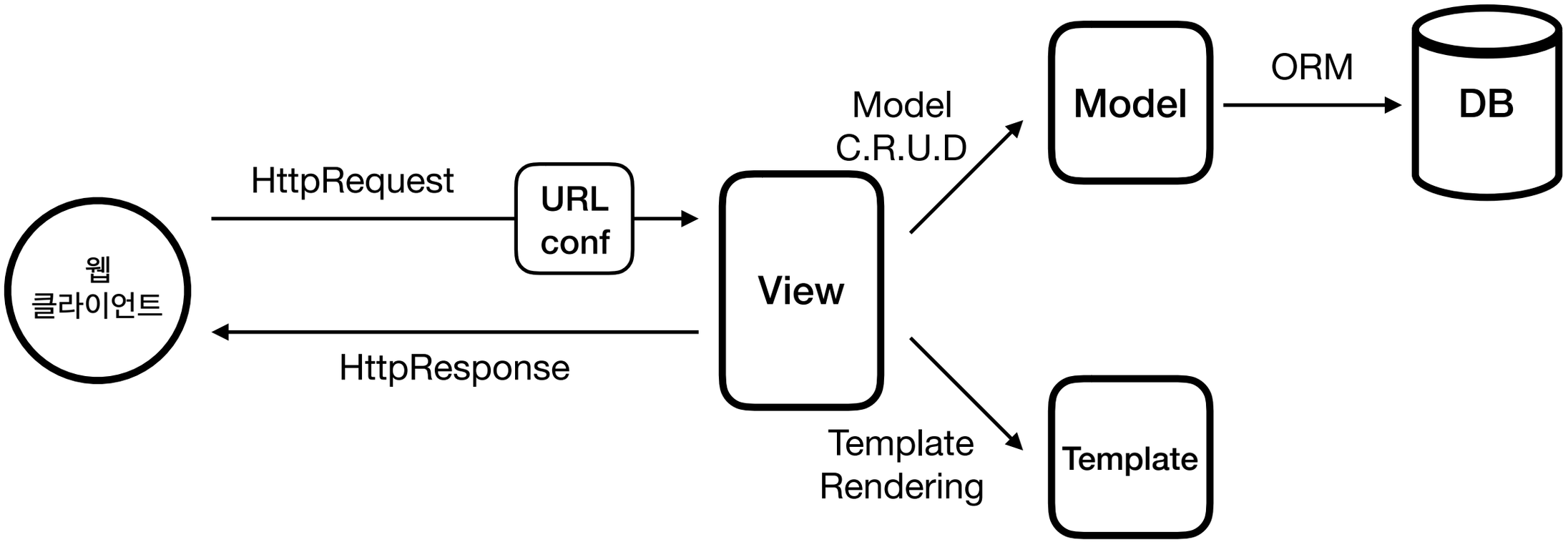


Django

- https://wiki.python.org/moin/WebFrameworks/

- https://www.djangoproject.com/

MTV 계층


Model
C.R.U.D
ORM
Model
DB
HttpRequest
URL
conf
웹
클라이언트
View
HttpResponse
Template
Rendering
Template

MTV 계층

모델(Model)

- 테이블을 정의
- 데이터베이스 처리

템플릿(Template)

- 사용자 인터페이스 정의 및 처리

뷰(View)

- HTTP 요청 처리
- 애플리케이션의 제어 흐름 및 처리 로직 정의

MTV 계층

- 모델, 템플릿, 뷰 모듈 간에 독립성 유지.
- 느슨한 결합(Loose Coupling) 설계 원칙 부합.
- 디자이너, 응용 개발자, DB 설계자 간에 협업 쉬워짐.

웹 프로그래밍 실전

0. 개발환경 구축

파이썬 설치

- 파이썬 다운로드 : www.python.org

- 아나콘다 다운로드 : www.anaconda.com

Visual Studio 설치

- 비주얼 스튜디오 코드 : code.visualstudio.com

가상환경

- 프로젝트별로 독립된 환경을 만듦
- 외부의 라이브러리는 서로 의존성을 갖고 있어 버전이 맞지 않는 경우 오작동 발생
- 인터넷에서 다운로드한 파이썬 라이브러리(패키지)가 충돌을 일으키는 것을 방지

가상환경 설치

- (윈도우) pip install virtualenv
- (맥) pip3 install virtualenv

NOTE

- pip(python install package) 프로그램은 파이썬의 오픈 저장소인 PyPI(python package index)에 있는 SW패키지를 설치하고 관리해주는 명령
- 파이썬 3.x 버전을 설치하면 같이 설치된다.

가상환경 구축

- virtualenv myenv 또는 python -m venv myenv

가상환경 활성화/비활성화

- (윈도우) myenv\scripts\activate
- (맥) source myenv\bin\activate
- deactivate

장고 설치

- pip install django

장고 공식 사이트

- www.djangoproject.com

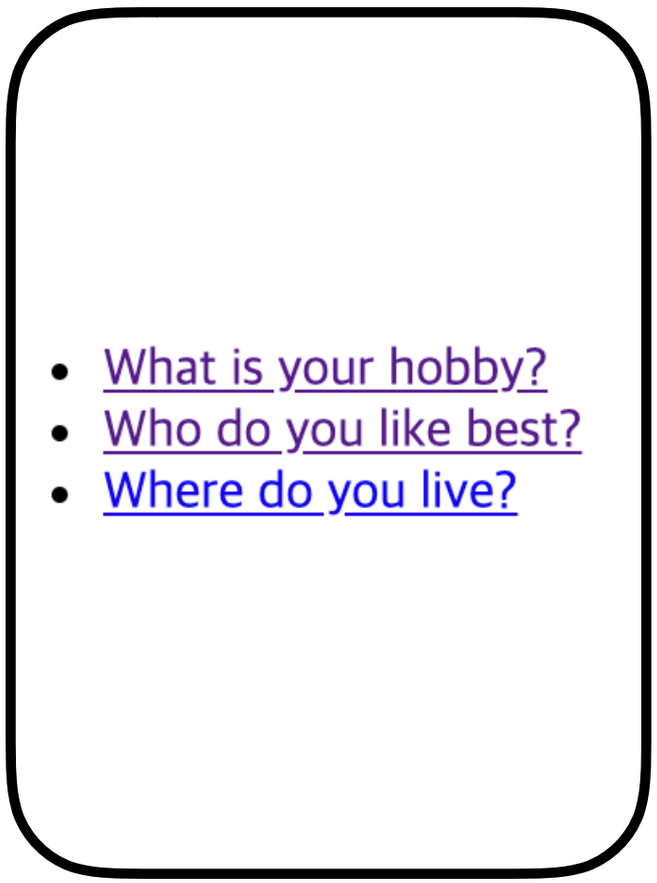
- github.com/django/django

1. 애플리케이션 설계

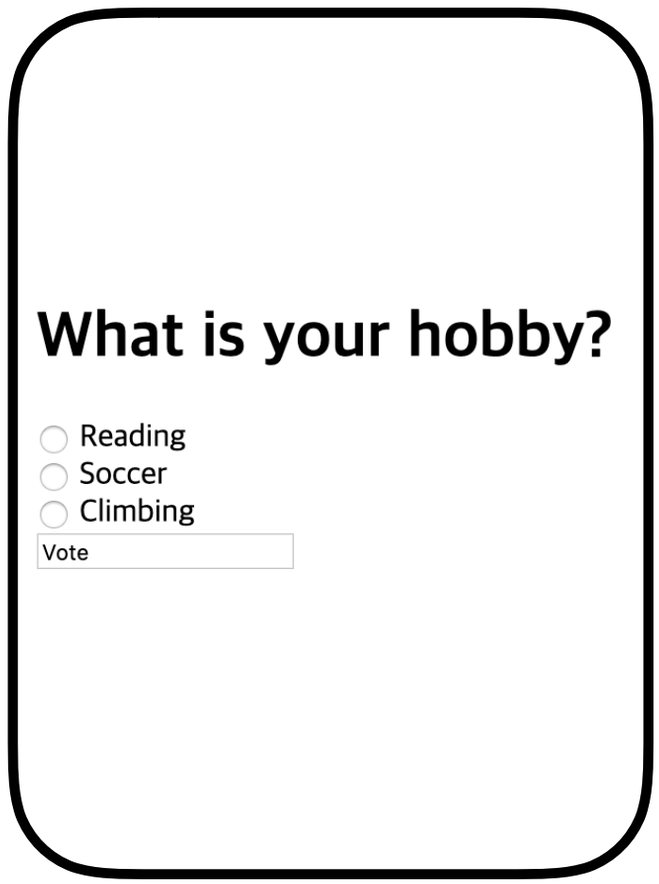
애플리케이션 설계

- 설문에 해당하는 질문을 보여준 후 질문에 포함된 답변 항목에 투표하면 그 결과를 알려주는 프로그램

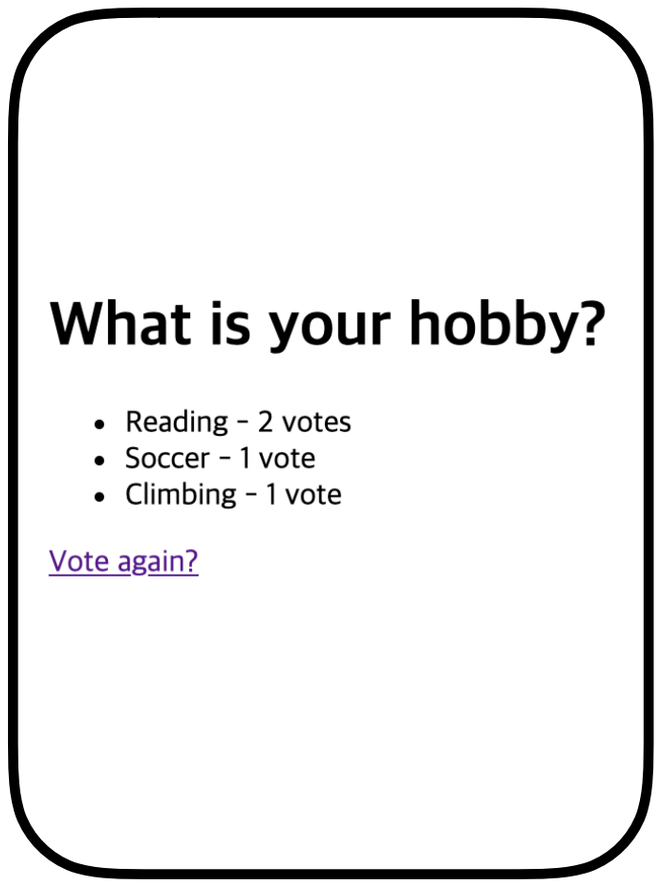
UI(User Interface) 설계



- Index.html : 최근에 실시하고 있는 질문 리스트를 보여준다.



- detail.html : 하나의 질문에 대해 투표할 수 있는 답변 항목을 폼으로 보여준다.



- results.html : 질문에 따른 투표 결과를 보여준다.

Question 테이블 설계

- 질문을 저장하는 테이블

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

Choice 테이블 설계

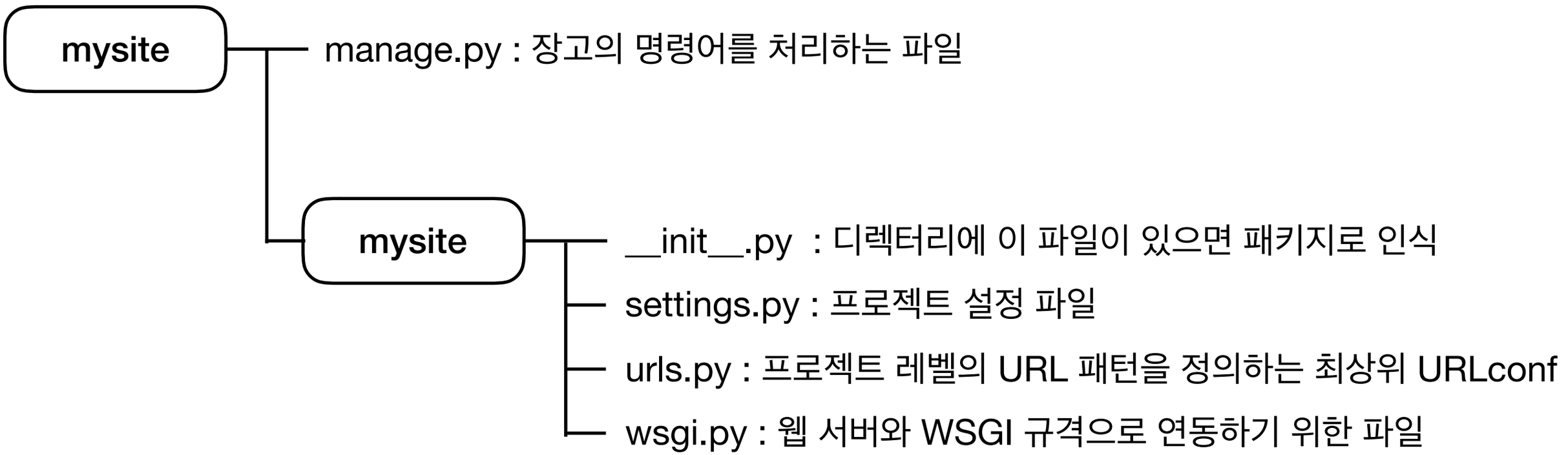
- 질문별로 선택용 답변 항목을 저장하는 테이블

컬럼명	타입	제약 조건	설명
id	integer	NotNull, PK, AutoIncrement	Primary Key
choice_text	varchar(200)	NotNull	답변 항목 문구
votes	integer	NotNull	투표 카운트
question	integer	NotNull, FK(Question.id), Index	Foreign Key

2. 프로젝트 생성

프로젝트 생성

- django-admin startproject mysite

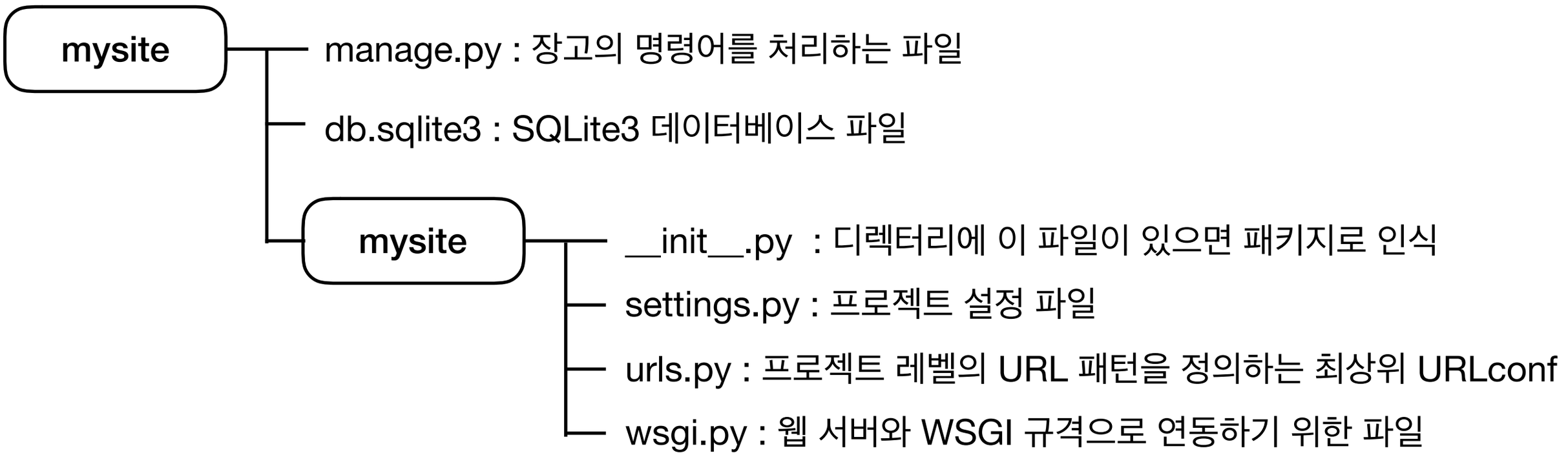


settings.py

- DEBUG = True이면 개발 모드, False이면 운영 모드
- ALLOWED_HOSTS에 서버의 IP나 도메인을 지정한다. 개발 모드인 경우 [‘localhost’, ‘127.0.0.1’]로 간주한다.
- INSTALLED_APPS에 프로젝트에 포함되는 애플리케이션을 등록한다.
- ROOT_URLCONF에 최상위 URLconf를 지정한다.
- TEMPLATES DIRS 항목은 프로젝트 템플릿 파일이 위치한 디렉터리를 지정한다.
- DATABASES에 프로젝트에 사용할 데이터베이스 엔진을 지정한다.
- LANGUAGE_CODE에 장고의 사용 언어를 지정한다.
- TIME_ZONE은 세계표준시(UTC)로 되어 있다.

기본 테이블 생성

- python manage.py migrate
- 사용자 및 그룹 테이블을 만든다.



개발용 웹 서버

- python manage.py runserver

- 개발 과정에서는 작성된 코드를 실행하고 테스트하는 과정이 필요하다.

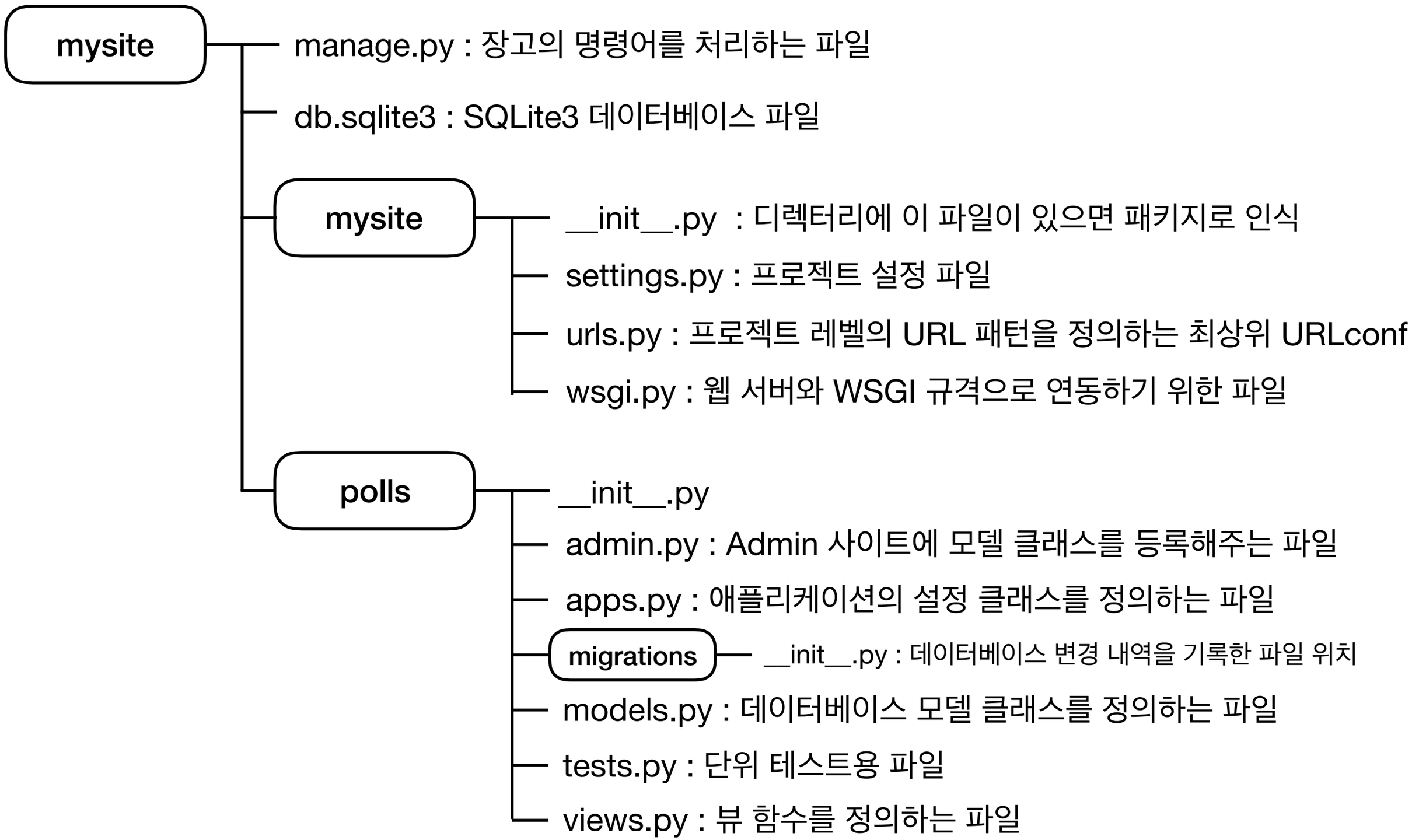
- 장고에서는 개발 과정에 현재의 웹 프로그램을 실행해볼 수 있도록 runserver라는 테스트용 웹 서버를 제공한다.

Admin 사이트

- http://127.0.0.1:8000/admin 또는 http://localhost:8000/admin
- 관리자 생성 : python manage.py createsuperuser
- SQL 없이도 테이블의 모습 및 내용을 확인하고 테이블에 레코드를 입력하고 수정할 수 있다.
- Admin 화면에서 기본적으로 User와 Group 테이블이 보이는 것은 settings.py 파일에 django.contrib.auth 애플리케이션이 등록되어 있기 때문이다.
- auth 앱에 Users와 Groups 테이블이 미리 정의되어 있다.
- User와 Group 테이블을 포함한 테이블에 대한 데이터의 입력, 수정, 삭제 등의 작업을 할 수 있다.
- Admin 사이트에 원하는 테이블을 등록하기 위해서는 admin.py 파일에 작업한다.

애플리케이션 생성

- 애플리케이션 : 프로젝트 하위의 서브프로그램
- python manage.py startapp polls



애플리케이션 등록

- 프로젝트에 포함되는 애플리케이션은 설정 파일에 등록해야 한다.

```
INSTALLED_APPS = [

    ‘polls.apps.PollsConfig’,  # 추가

]
```

3. Model 코딩

모델(Model)

- 데이터베이스에 저장되는 데이터를 의미한다.
- ORM(Object Relation Mapping) 기법 : 객체와 관계형 데이터베이스를 연결해주는 역할
- 테이블을 하나의 클래스로 정의한다.
- 테이블의 열은 클래스 변수(속성)으로 정의한다.
- 즉, 모델이란 데이터에 대한 정의를 담고 있는 장고의 클래스

models.py

- 테이블 클래스는 django.db.models.Model 클래스를 상속받아 정의한다.

- 클래스 변수의 타입은 장고에서 미리 정의된 필드 클래스를 사용한다.

- PK는 클래스에 지정해주지 않아도 장고에서 NotNull, Autoincrement로 자동으로 만들어준다.

- FK는 항상 다른 테이블의 PK에 연결되므로 클래스만 지정하면 된다. 실제 테이블에서 FK로 지정된 열은 _id가 붙는다.

polls.models.py

```
class Question(models.Model):
    question_text = models.CharField(max_length=200)

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

__str__ 메서드 오버라이딩

- __str__() 메서드는 객체를 문자열로 표현할 때 사용하는 함수이다.

```
def __str__(self):

    return self.question_text
```

Model Fields Type

- http://docs.djangoproject.com/ko/2.1/ref/models/fields#model-field-types

polls.models.py

```
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
```

Foreign Key

- 1:N 관계
- django.db.models.ForeignKey 사용
- ForeignKey(to, on_delete)
  - to : 대상 모델
  - on_delete : 1측 열이 삭제될 경우 N측 열에 대한 동작을 지정
    - CASCADE
    - PROTECT
    - SET_NULL/SET_DEFAULT/SET
    - DO_NOTHING

모델 활성화

- 모델 클래스로부터 마이그레이션 파일 생성
- python manage.py makemigrations polls
- 마이그레이션 파일로부터 데이터베이스 테이블 생성
- python manage.py migrate polls

admin.py

- models.py 파일에서 정의한 테이블이 Admin 사이트에 보이도록 등록한다.

- 테이블을 새로 만들 때는 models.py와 admin.py 두 개의 파일을 함께 수정해야 한다.

```
from polls.models import Question, Choice

admin.site.register(Question)

admin.site.register(Choice)
```

4. URLconf 코딩

URLconf

- URL : 프로토콜://호스트명:포트번호/경로?쿼리스트링#앵커
- URLconf : URL/뷰 매핑
- 프로젝트 전체 URL을 정의하는 프로젝트 URL, 앱마다 정의하는 앱 URL, 2계층으로 나눠서 코딩할 수 있다.
- URL 정의 : urls.py 파일에 URL과 뷰(처리 함수)를 매핑하는 파이썬 코드를 작성한다.
- URL 분석 : 요청에 들어있는 URL이 urls.py 파일에 정의된 URL 패턴과 매칭되는지 분석한다.

URL 정의

- polls.urls.py : URL과 뷰(함수 또는 메서드)를 매핑해주는 파일
- path() 함수는 route, view 2개의 필수 인자를 받는다.
- route : URL 패턴을 표현하는 문자열(URL 스트링)
- view : URL 스트링이 매칭되면 호출되는 뷰 함수. HttpRequest 객체와 URL 스트링에서 추출된 항목이 뷰 함수의 인자로 전달된다.
- name : 각 URL 패턴별로 이름을 붙인다. 템플릿 파일에서 많이 사용된다.
- app_name 변수는 URL 패턴의 이름이 충돌되는 것을 방지하기 위한 이름 공간 역할을 한다. app:name로 표기해서 구분한다.

polls.urls.py

```
from polls import views

app_name = ‘polls'

urlpatterns = [
    path('', views.index, name=‘index’),
    path('<int:question_id>/', views.detail, name='detail'),
    path('<int:question_id>/results/', views.results, name='results'),
    path('<int:question_id>/vote/', views.vote, name='vote'),
]
```

URL pattern

- URL 패턴은 ‘/’로 끝난다.
- 첫 번째 ‘/’는 내부적으로 추가되기 때문에 생략한다.
- view의 인자로 사용되는 값은 ‘<>’를 사용한다.
- view의 인자로 사용되는 값의 타입 지정 시 <type:name>으로 지정한다.

Path Converter

- URL 패턴의 일부 문자열을 추출하기 위한 것
- <type:name> 형식으로 사용한다.
- path(‘/<int:question_id>/’, views.detail, name=‘detail’)
- <int:question_id>인 경우 <>부분이 정수이면 매치되고 아니면 매치되지 않는다.
- /polls/5/로 매치된 경우 views.detail(request, question_id=5)로 호출한다.

URL 분석

- settings.py 파일의 ROOT_URLCONF 항목을 읽어 최상위 URLconf의 위치를 알아낸다.
- 최상위 URLconf를 로딩하여 urlpatterns 변수에 지정되어 있는 URL 리스트를 검사한다.
- include를 통해 TREE 구조로 확장한다.
- TREE 구조로 확장된 urlpatterns 변수에 지정되어 있는 URL 리스트를 검사한다.
- 위에서부터 순서대로 URL 리스트의 내용을 검사하면서 URL 패턴이 매칭되면 검사를 종료한다.
- 매치된 URL의 뷰를 호출한다. 호출 시 HttpRequest 객체와 매칭할 때 추출된 단어들을 뷰에 인자로 넘겨준다.
- URL 리스트 끝까지 검사했는데도 매칭에 실패하면 에러를 처리하는 뷰를 호출한다.

5. View 코딩

뷰(View)

- 함수 또는 클래스의 메서드로 작성되며 웹 요청을 받고 응답을 반환하는 역할

- views.py : 다양한 형태의 응답 데이터를 만들어내기 위한 로직을 작성하는 파일

- 함수형 뷰(FBV)와 클래스형 뷰(CBV)가 있다.

- HttpRequest : View의 첫 번째 인자, 클라이언트의 요청 정보

- HttpResponse : View의 반환 객체, 클라이언트로 전달되는 응답 정보

views.py

```
from django.shortcuts import render

from polls.models import Question

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)
```

polls.templates.polls.index.html

```
{% if latest_question_list %}

    <ul>

        {% for question in latest_question_list %}
            <li><a href="/polls/{{ question.id }}/">{{ question.question_text }}</a></li>

        {% endfor %}

    </ul>

{% else %}
    <p>No polls are available.</p>

{% endif %}
```