

DB & django CR

# 6번째 세션


---

NEXT X LIKELION 이민용

즐거운 세션 날 😊

django

# 10기의 미래가 밝습니다



글자수를 세어드리겠습니다.

I

글자수 세기

글자수 세기

공백포함: 총 0자 | 공백제외: 총 0자

글자수 보기

다시하기

글자를 입력해라!

보노보노

글자를 세  
줍니당~

아마도 전성운..?

Letter Counter

SUBMIT

Please fill in your text

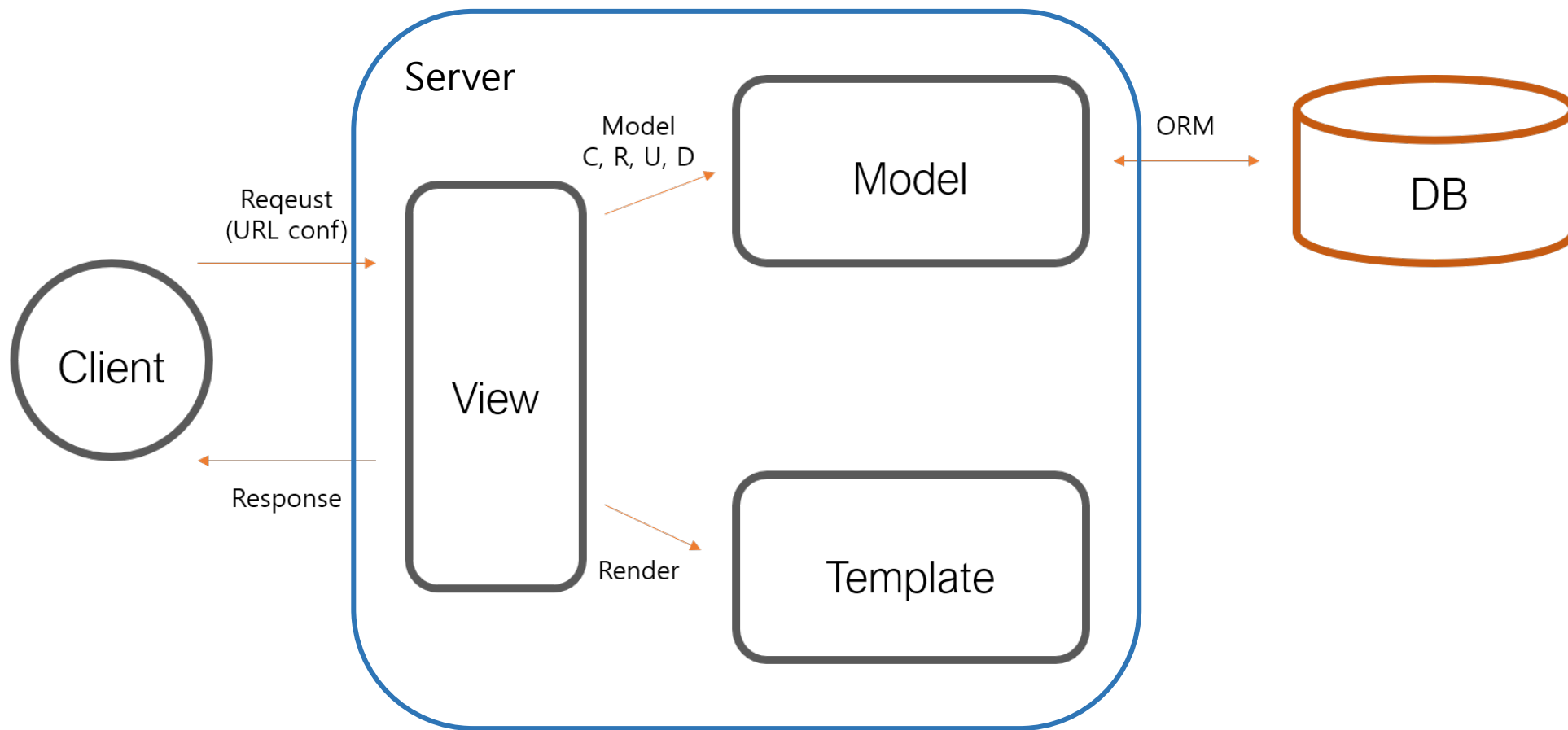
Letter (w/ spaces): 0 letters

Letter (w/o spaces): 0 letters

Word : 0 words

# 지난 시간 복습

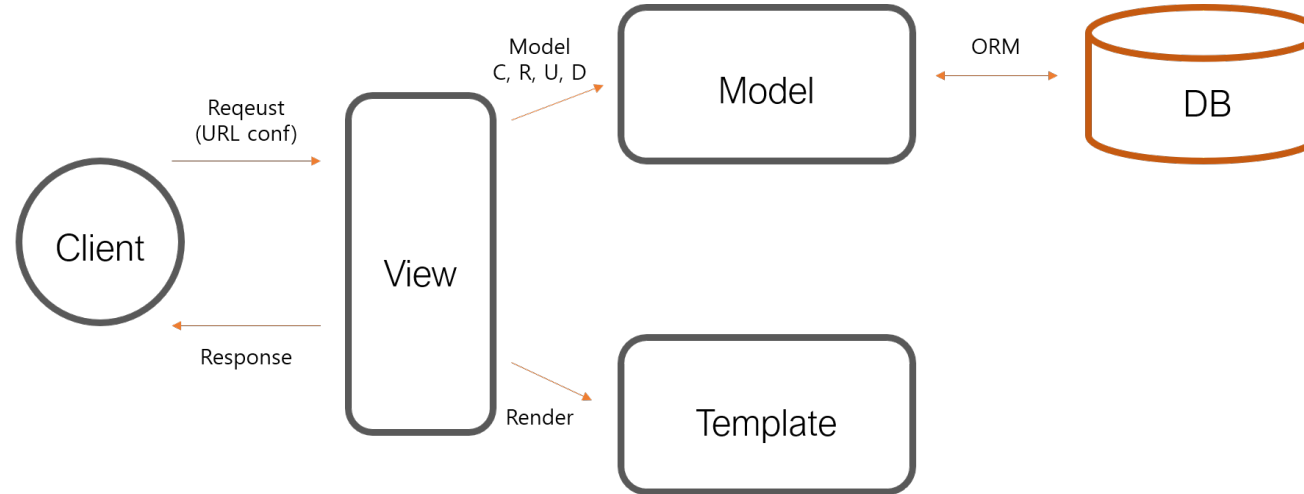
MTV



클라이언트의 요청에 서버는 어떤 과정을 거쳐 응답하는가?

# 지난 시간 복습

MTV



1. 클라이언트가 요청(Request)을 서버의 어떤 URL로 보내는지 분석
2. 매칭된 URL에 해당하는 View의 함수를 실행
3. View에 작성된 로직에 따라 Model을 통해 DB의 데이터를 조작
  - 생성(Create), 조회(Read), 수정(Update), 삭제>Delete)
4. View는 클라이언트에게 응답할 HTML 파일을 Template을 통해 렌더링
5. 서버는 최종적으로 클라이언트의 요청에 응답(Response)

# 지난 시간 복습

GET vs POST

## GET

존재하는 자원을 조회할 때

읽기

어떤 내용을 가지고 오고 싶을 때!

URL에 변수(데이터)를 포함시켜 요청

## POST

새로운 자원을 생성

저장

새로운 내용을 쓸 때

URL에 변수(데이터)를 노출하지 않고 요청

# 웹 사이트에 글을 쓰려면?

공지 공식팬카페 · 스타 · 소속사

IU(아이유) 공식 팬카페

다이아 (공개) >

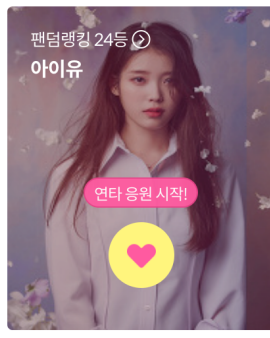
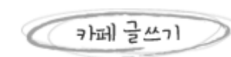
카페지기 IU official..

회원수 292,046 | 카페 초대

방문수 4,252

카페앱수 11,159

내 정보



최신글 보기

공지	안내) 팬카페 이용 관련 (7)	FS_일요일이좋다	22.01.11	1	1920
공지	[공지] 자유게시판 이용 안내사항 (21.03.03 ver)	IU official staff	21.03.03	3	7871
공지	자유게시 안내) 친목 관련 (29)	FS_일요일이좋다	21.01.22	9	2977
공지	자유게시 광고 쪽지 수신에 대한 안내. (956)	FS_JohnKim	20.05.22	5	4866
공지	Notice about the posting policies (21.03.03 update) (77)	FS_일요일이좋다	20.05.08	5	2583
736771	달리는글 스베문 목걸이 (1)	꽃하림	02:31	0	40
736770	What should we do to stop 지은 from working too hard.....? (1)	Dionus	02:25	0	33
736769	지은언니 보고싶은날 (1)	슈크림슈니	02:14	0	13
736768	Love Poem, 2019 (2)	jennifera	02:03	0	44
736767	콘서트 (1)	우양	01:53	0	46
736766	Win the lottery (8)	Thunder	01:48	0	46
736765	렌티쿨러 포카라니.. (6)	caro	01:42	0	76
736764	드려 우수회원!! (4)	스물셋팔레트	01:36	0	26
736763	오늘 얼굴美쳤어 (1)	빨간파란색	01:33	0	59
736762	우수회원 (6)	tereke	01:27	0	46
736761	다들아들아 대보지 (1)	변신되다며	01:07	0	74

## 웹 사이트에 글을 쓰려면?



글의 내용



글을 저장할 공간



# Database(DB)란?

## 데이터를 저장하는 저장소

📌 데이터를 효율적으로 관리, 사용하기 위해 구성된 데이터의 집합

📌 다양한 종류의 DBMS(데이터베이스 관리 시스템)

- RDBMS: SQLite, MySQL, PostgreSQL ...
- NoSQL : MongoDB, Cassandra ...

# RDBMS

<학생 테이블(Relation)>

어트리뷰트(attribute)

스키마(schema)

튜플(tuple)

학번	이름	학과	학점
2019130528	이민용	사회학과	3.9
2016130514	김동민	사회학과	4.3
2013230152	서인재	신소재공학과	3.8

→ 테이블의 형태로 관리되는 관계형(Relational) 데이터 모델

# RDBMS

Primary key

학번이 같은 두 학생이 있을 수 있나?



<학생>

학번	이름	학과	학점
2019130528	이민용	사회학과	3.9
2016130514	김동민	사회학과	4.3
2013230152	서인재	신소재공학과	3.8

**Primary Key : 테이블의 행(=튜플)을 고유하게 구분해주는 키**

각 테이블에는 딱 하나의 PK만 지정됨

But, 여러 어트리뷰트의 그룹을 PK로 지정하는 것은 가능

## &lt;학생&gt;

학번	이름	학과	지도교수	지도교수 연락처
2019130528	이민용	사회학과	허영범	010-1111-1111
2016130514	김동민	사회학과	허영범	010-1111-1111
2013230152	서인재	신소재공학과	이한주	010-9999-9999

위 테이블 예시의 문제점은 뭘까요?

# RDBMS

## Foreign key

<학생>과 <교수> 테이블이 각각 있고 두 테이블 간에는 참조 관계가 있다

<학생>

학번	이름	학과	지도교수 번호
2019130528	이민용	사회학과	1
2016130514	김동민	사회학과	1
2013230152	서인재	신소재공학과	2
2018130512	양효령	사회학과	1
2020230134	김이린	신소재공학과	2
2020130511	권규리	사회학과	1
2021130507	김지성	사회학과	1

<교수>

교수 번호	이름	연락처
1	허영범	010-1111-1111
2	이한주	010-9999-9999
3	이소영	010-2222-2222

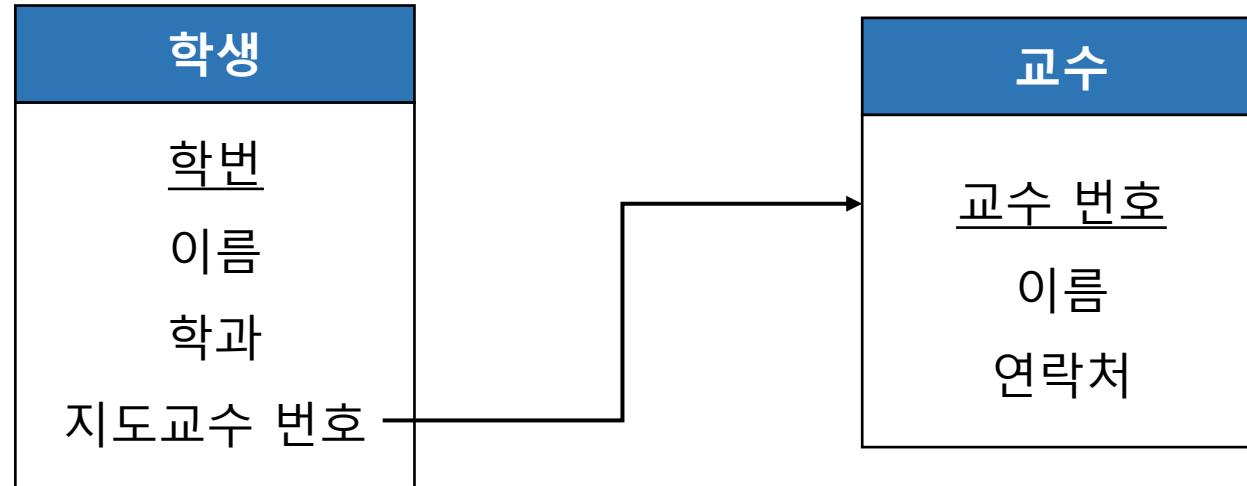
## Foreign key

참조 대상 테이블의 튜플을 식별할 수 있는 키

# RDBMS

## Schema diagram

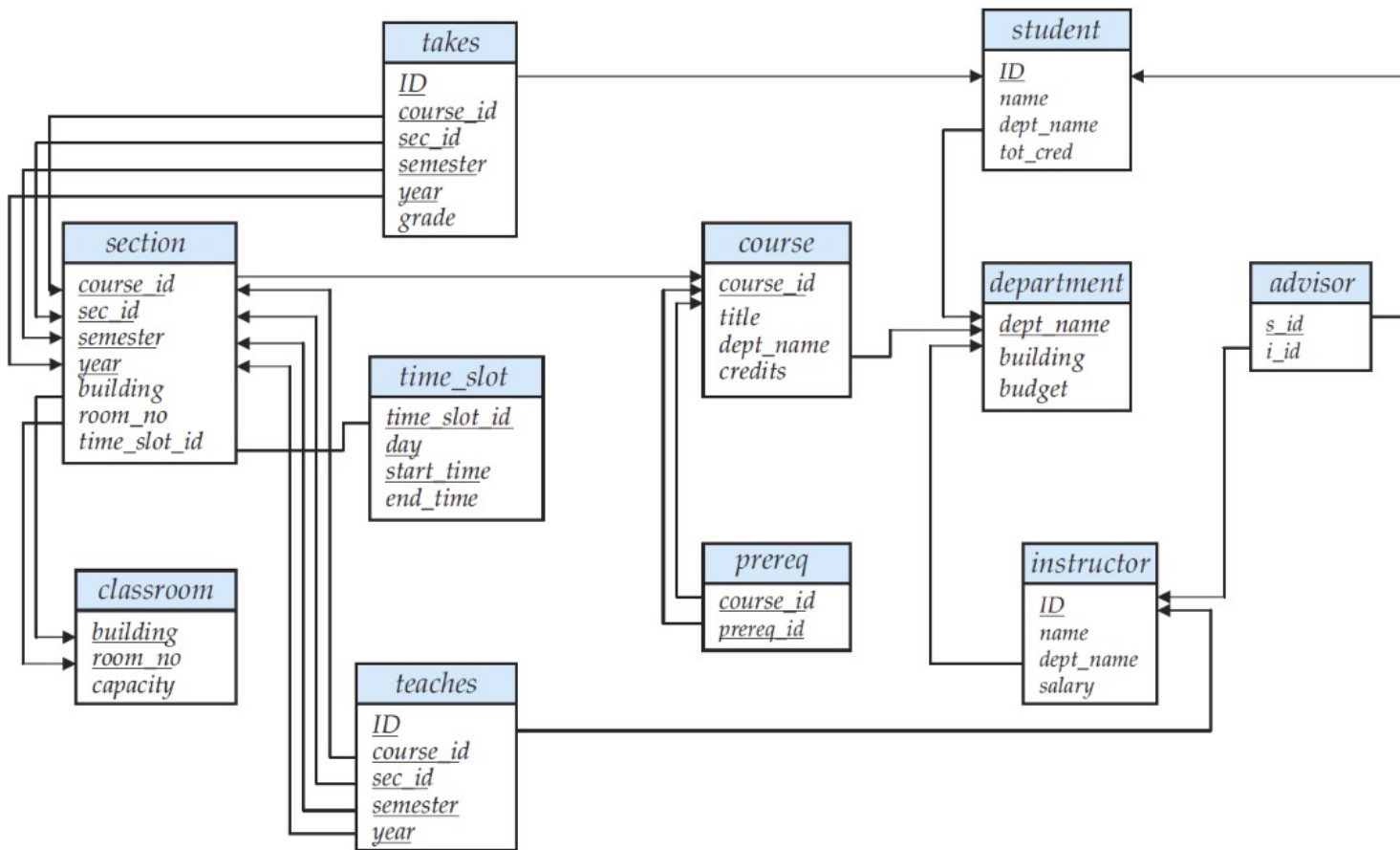
밑줄 : Primary key  
화살표 : Foreign key 참조



- 📌 "교수 번호" 는 <교수> 테이블의 Primary key
- 📌 "지도교수 번호" 는 <교수> 테이블을 참조하는 <학생> 테이블의 Foreign key
- 💡 일반적으로 참조 대상이 되는 테이블의 PK를 Foreign key로 삼는 경우가 많음

## Schema diagram

실제 대학교 DB의 Schema diagram 예시  
(학생, 교수, 학과, 강의, 선수과목, 지도교수, 강의실 등등)



블로그를 만든다면 DB에 어떤 테이블이 있어야 할까요?

글 ID	제목	내용	작성자 ID
article_1	...	...	user_1
article_2	...	...	user_2
article_3	...	...	user_2

<글 테이블>

댓글 ID	내용	글 ID	작성자 ID
comment_1	아	post_1	user_2
comment_2	이	post_1	user_3
comment_3	유	post_2	user_3

<댓글 테이블>

유저 ID	닉네임
user_1	A
user_2	B
user_3	C

<유저 테이블>



앞 장을 참고해 빈 칸을 채워볼까요?

글 ID	작성자 닉네임	댓글 내용
article_1	...	
article_2	...	
article_3	...	

<글 테이블>

글 ID	작성자 닉네임	댓글 내용
article_1	A	아, 이
article_2	B	유
article_3	B	

<글 테이블>

# CRUD

CRUD 소개



CREATE

READ

UPDATE

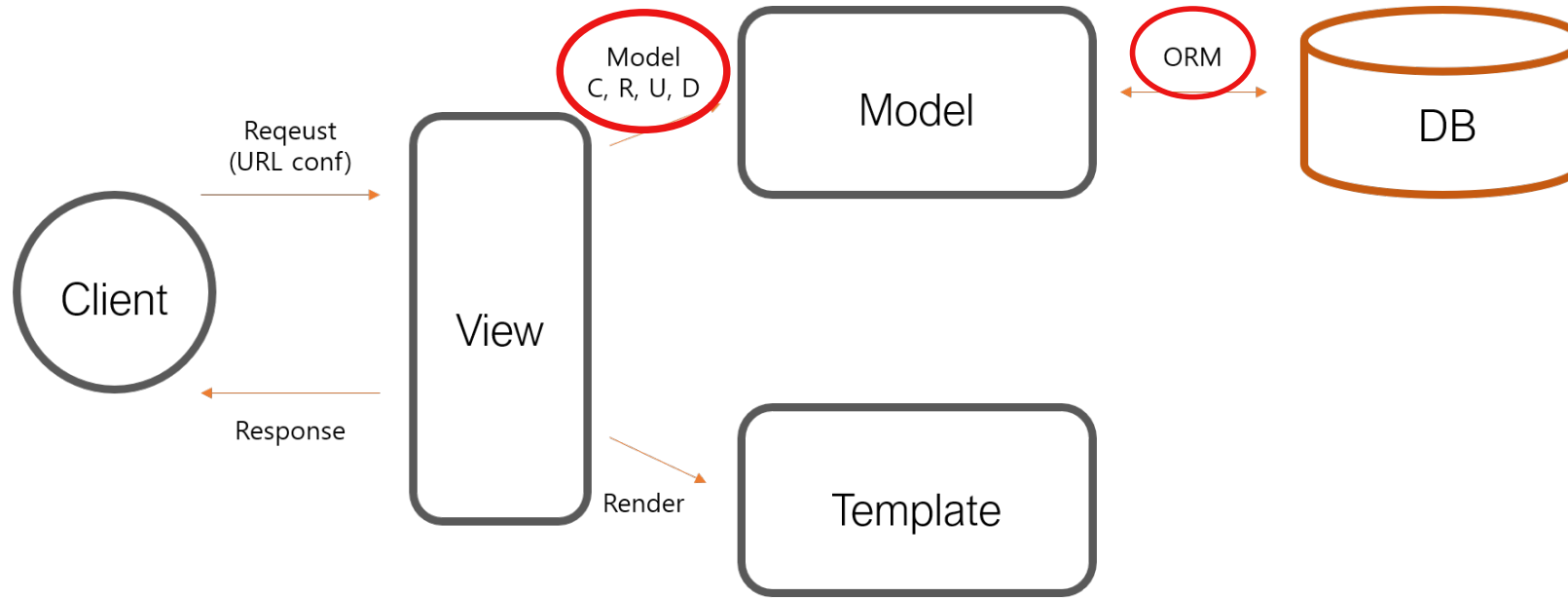
DELETE

---

C R U D

# CRUD

## CRUD 소개



RDBMS에서는 **SQL**이라는 데이터 조작 언어를 기반으로 **CRUD**를 수행  
django는 파이썬 문법으로 데이터를 쉽게 조작할 수 있도록 **ORM**을 제공한다!

# 오늘의 실습

자 이제 진짜 시작입니다

## <블로그 만들기>

CRUD의 **Create**와 **Read**를 통해

1. 새로운 글을 작성하고
2. 작성된 글을 읽을 수 있는

블로그를 만들어 볼 거예요~~!

# 프로젝트 세팅

장고 프로젝트, 앱 생성

## 프로젝트 세팅 To-do

\$ mkdir session6, \$ cd session6

\$ pipenv shell

\$ pipenv install django

\$ django-admin startproject blogProject

\$ cd blogProject

\$ python manage.py startapp blogApp

vscode를 켜서 settings.py의 INSTALLED\_APPS에 blogApp 추가

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

# 프로젝트 세팅

DB생성, 관리자 계정 생성

## 프로젝트 세팅 To-do

### 1. DB 생성

`$ python manage.py makemigrations`

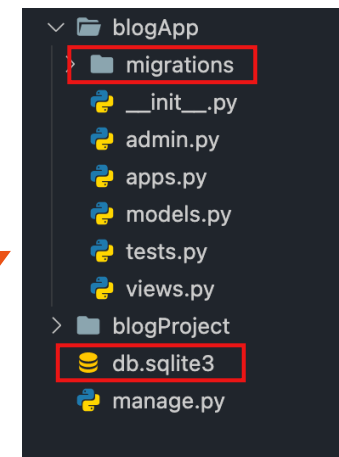
`$ python manage.py migrate`



Model에 변경사항이 생길 때마다!

### 2. 관리자 계정 생성

`$ python manage.py createsuperuser`

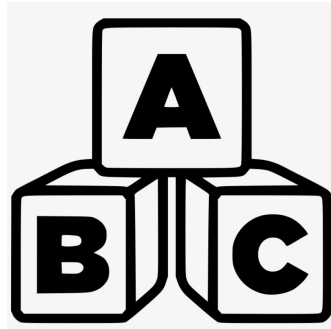


# Model

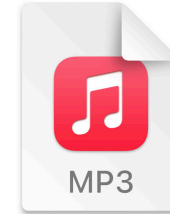
스키마 정의



TextField( )



CharField( max\_length= )



FileField( )



DateTimeField( )

이외에도 다양한 필드가 있으니 필요할 때 찾아보세요!

<https://docs.djangoproject.com/en/3.1/ref/models/fields/>



# Model

## 스키마 정의

<사람 테이블>

pk	name	address
1	...	...
2	...	...
3	...	...

models.py

```
class Person(models.Model):  
    name = models.CharField(max_length=10)  
    address = models.TextField()
```

📌 django는 models.py에서 **class**를 사용해 **테이블의 스키마(=모델)**를 정의

📌 이때 우리가 만드는 모델들은 django가 미리 만들어놓은 **models.Model**이라는 **class**를 상속

📌 **클래스 상속?**

새로운 클래스(자식) 생성 시에 기존에 정의되어 있는 다른 클래스(부모)의 특성들을 물려받도록 하는 것

ex) class KUNStudent(Student)    ➡ 고려대 학생은 학생의 공통적인 특성들을 기본으로 가짐  
    class Person(models.Model) ➡ Person 모델은 models.Model의 특성들을 기본으로 가짐

# Model

## Quiz

<Article 테이블>

pk	title	content
1	...	...
2	...	...
3	...	...

**조건:**

"title"은 200자로 제한을 두고 싶고,  
"content"에는 긴 글을 저장하고 싶어!

# Model

## Quiz

<Article 테이블>

pk	title	content
1	...	...
2	...	...
3	...	...

정답

```
class Article(models.Model):  
    title = models.CharField(max_length=100)  
    content = models.TextField()
```

📌 django의 경우 id 어트리뷰트(pk)를 자동으로 생성하여  
1부터 순서대로 번호를 붙여준다

# Model

오브젝트 생성

모델에 변경사항이 생겼으니 DB에 반영해준다

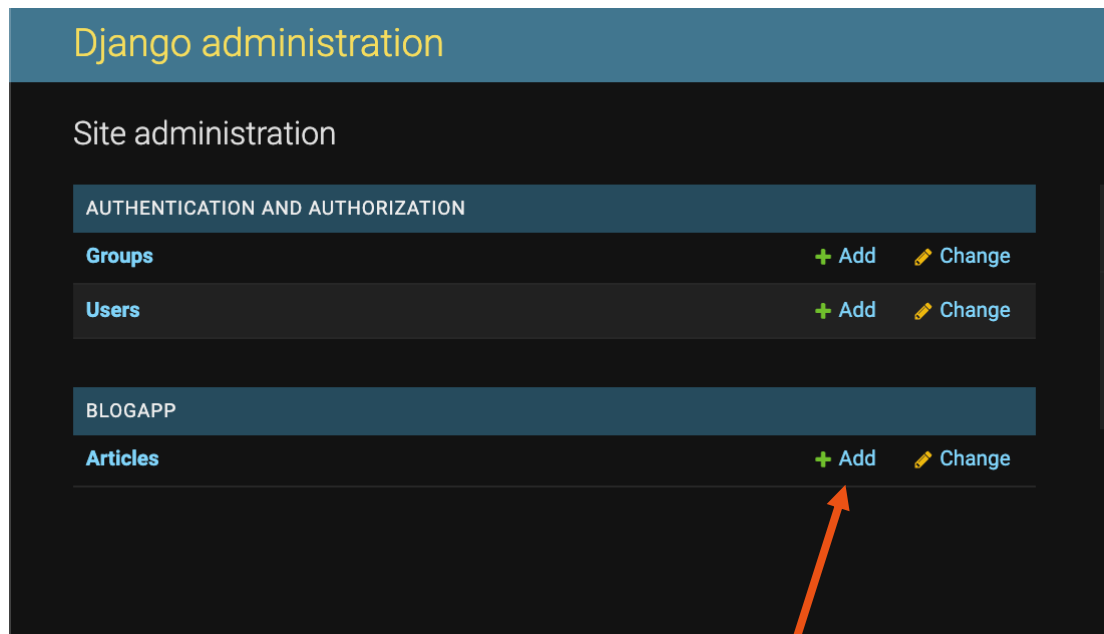
```
$ python manage.py makemigrations
```

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

앱 폴더 내 admin.py에 만든 모델을 등록해준다

admin.py

```
from .models import Article  
  
# Register your models here.  
admin.site.register(Article)
```



Add를 누르고 직접 Article을 DB에 생성해보아요!

# Model

오브젝트 생성

Action:

- ☐ ARTICLE
- ☐ Article object (2)
- ☐ Article object (1)

2 articles

생성은 됐는데 object를 구분하기가 어렵네..?

Action:

- ☐ ARTICLE
- ☐ 두번째 글이야!
- ☐ 첫 번째 글

2 articles

Good~!

models.py

```
class Article(models.Model):
    title = models.CharField(max_length=100)
    content = models.TextField()

    def __str__(self):
        return self.title
```

**\_\_str\_\_ ?**

Article 모델로 만든 오브젝트 자체를 출력(print)할 때 값을 title로 할래!

📌 모델 = 클래스 = 붕어빵틀

📌 오브젝트 = 인스턴스 = 붕어빵

# Model

ORM (Object Relational Mapping)

모델을 가지고 글을 생성(Create)하고 조회(Read)하기 위해 django의 ORM을 사용!

## Create

<모델명>.objects.create( )

( ) 안에는 우리가 정의한 필드의 값을 지정

```
ex) Person.objects.create(  
    name = "민용"  
    age = 26  
)
```

## Read

<모델명>.objects.all( )

\* 모든 인스턴스의 배열

<모델명>.objects.get( )

\* ( )안의 조건에 맞는 인스턴스가 없거나 하나 이상이면 에러!

<모델명>.objects.filter( )

\* ( )안의 조건에 맞는 모든 인스턴스의 배열

<https://docs.djangoproject.com/en/3.1/topics/db/queries/>

# Model

## ORM Quiz

1. title이 '좋은날'이고 content가 '눈물이 차 올라서 고갤 들어'인 Article을 DB에 생성한다면?

```
Article.objects.create(title = "좋은날", content = "눈물이 차 올라서 고갤 들어")
```

2. DB에 있는 모든 Article을 조회한다면?

```
Article.objects.all()
```

3. DB에서 pk가 3인 Article을 조회한다면?

```
Article.objects.get(pk = 3)
```

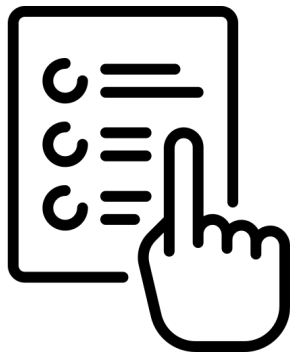
이제 모델이 준비되었으니 실제 블로그 페이지를 만들어보자!

### 어떤 페이지가 필요할까?



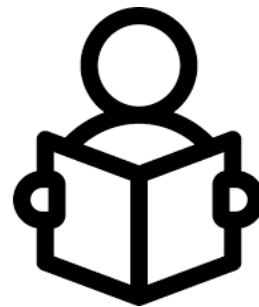
#### New

- 글 작성 양식
- 작성 완료 버튼



#### List

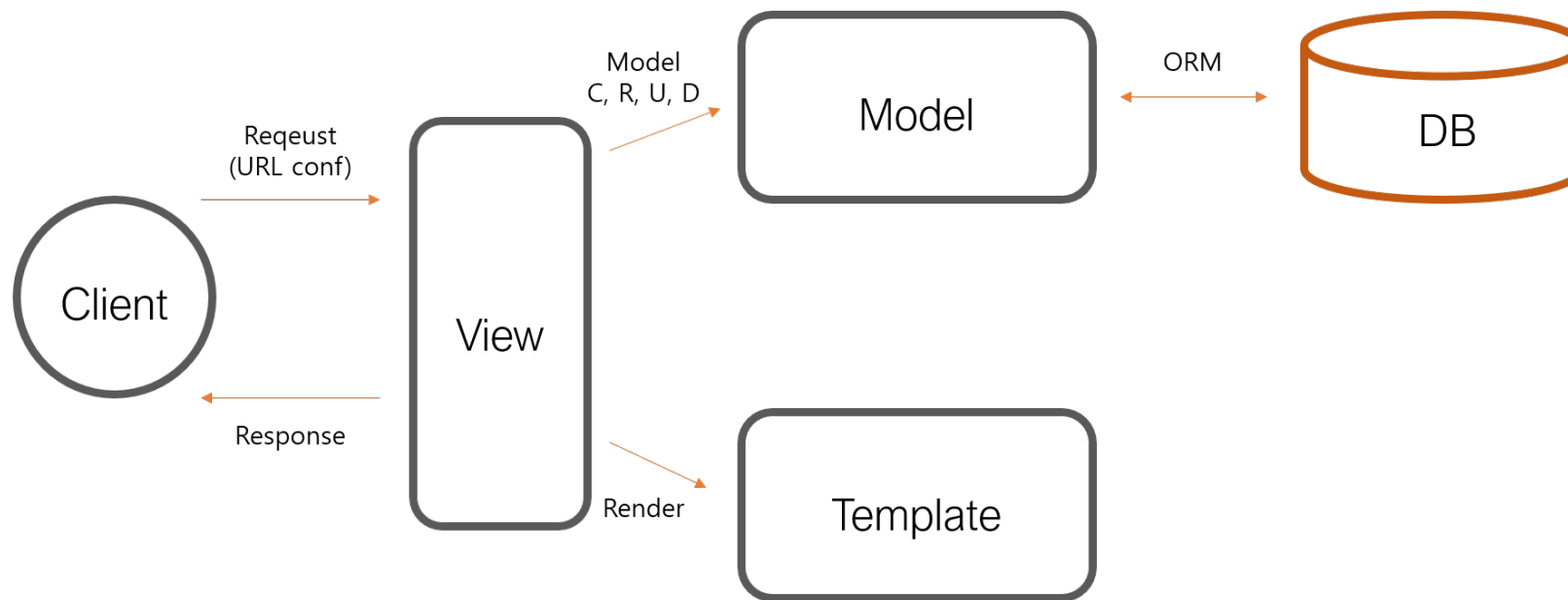
- 작성된 전체 글 리스팅
- 글의 세부 페이지(detail)로 이동
- 글 작성 페이지(new)로 이동



#### Detail

- 글 하나의 세부내용 보여주기
- 목록으로 돌아가기





django의 작동 방식을 생각하면서 하나하나 만들어보도록 해요~!

# Blog

## 1. new 페이지

new.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    <h2>글 쓰기 페이지</h2>
11    <form action="/new" method="post">
12      {% csrf_token %}
13      <div>
14        <div>글 제목</div>
15        <input type="text" name="title" />
16      </div>
17      <div>
18        <div>글 내용</div>
19        <textarea name="content" cols="30" rows="10"></textarea>
20      </div>
21      <button type="submit">작성 완료</button>
22    </form>
23  </body>
24 </html>
```

## 글을 작성하는 페이지를 먼저 만들어보아요!

💡 html파일은 app 폴더 안에 templates 폴더에 만드는 거 기억하시죠?

- 값을 입력 받아 제출하는 페이지 ➡ <form>
- 제목과 내용을 각각 입력 받아야 함 ➡ <input>, <textarea>
- 입력 받은 값을 제출하는 버튼 ➡ <button>
- new url로 값을 전달하여 글을 생성하는 행위  
➡ action="/new" method="post"

# Blog

## 1. new 페이지

urls.py

```
from blogApp import views

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

views.py

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

**new라는 url로 요청하면 페이지를 보여줘야 하니까**

- url이 new일 때 views의 new 함수를 실행
- new 함수는 new.html을 렌더링하는 작업을 수행

➡ localhost:8000/new 로 접속해보자!

# Blog

## 1. new 페이지

views.py

```
from django.shortcuts import render, redirect
from .models import Article

# Create your views here.
def new(request):
    if request.method == 'POST':
        # POST 요청으로 온 데이터 확인
        print(request.POST)
        new_article = Article.objects.create(
            title = request.POST['title'],
            content = request.POST['content']
        )
        return redirect('list')

    return render(request, 'new.html')
```

## 작성 완료하여 /new url에 POST로 요청 시

<요청을 받은 views.new 함수의 생각>

어? 내가 요청을 받았는데 **POST** 요청이네??

➡ 아 글을 생성하려 하는구나!

➡ 입력 받은 정보로 글을 생성하고 다른 페이지로 보내줄게!  
(redirect)

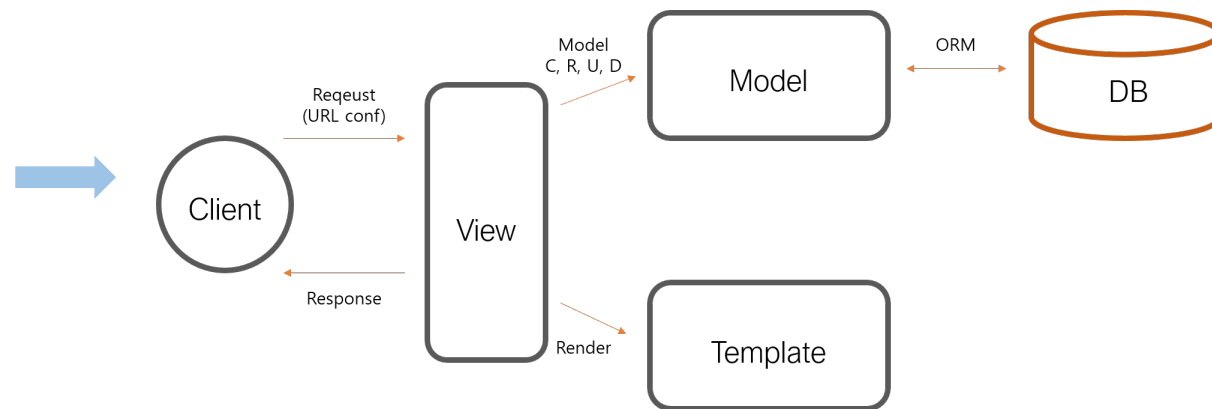
# Blog

## 2. list 페이지

### 글 목록 페이지 url에 접근하면?

HTML 파일에는 데이터가 저장되어 있지 않으니...

- **Views**가 **Model**을 통해 데이터를 조회하고
- 렌더링 할 수 있도록 **Templates**로 전해줘야겠죠?



url.py

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

views.py

```
def list(request):
    articles = Article.objects.all()
    return render(request, 'list.html', {'articles': articles})
```

# Blog

django 템플릿 문법

```
{% if (조건문) %}
```

~~~

```
{% endif %}
```

```
{% for ~ in ~ %}
```

~~~

```
{% endif %}
```

```
{{ object }} {{ object.attribute }}
```

Views가 건네준 데이터를 **Templates**이 효과적으로 사용 가능하도록 지원되는 문법  
각각 파이썬에서 **조건문**, **반복문**, **변수(or 딕셔너리)**를 사용하는 방식과 유사하다!

<https://docs.djangoproject.com/en/3.1/ref/templates/language/>

# Blog

## 2. list 페이지

views.py

```
def list(request):  
    articles = Article.objects.all()  
    return render(request, 'list.html', {'articles': articles})
```

list.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 <h2>글 전체 목록</h2>  
11 <ul>  
12     {% for article in articles %}  
13     <li>{{ article.title }}</li>  
14     {% endfor %}  
15 </ul>  
16 <a href="{% url 'new' %}">글 쓰기 가기</a>  
17 </body>  
18 </html>
```

글 목록 페이지를 만들어보아요!

Article.objects.all( )로 조회하여 넘겨준 데이터를  
HTML에서 리스트로 보여주려면 어떻게 해야할까?

- ➡ django 템플릿 문법의 반복문 사용
- ➡ localhost:8000/list 로 접속해서 글을 써보자!

새로운 글 쓰는 페이지 url → new

전체 글 목록 페이지 url → list

각기 다른 글의 세부 페이지 url은 어떻게 표현할 수 있을까?

→ detail/1, detail/2, detail/3



# Blog

## 3. detail 페이지

urls.py

```
urlpatterns = [
    path('admin/', admin.site.urls),
    path('new', views.new, name='new'),
    path('list', views.list, name='list'),
    path('detail/1', views.detail, name='detail'),
    path('detail/2', views.detail, name='detail'),
    path('detail/3', views.detail, name='detail'),
    ...
]
```

글이 하나 추가 될 때마다 url에 하나씩 써주면?

글이 200개면 urlpattern이 200줄...



urls.py

```
urlpatterns = [
    path('admin/', admin.site.urls),
    path('new', views.new, name='new'),
    path('list', views.list, name='list'),
    path('detail/<int:article_id>', views.detail, name='detail'),
]
```

url parameter(매개변수)의 형태로 url 패턴 지정 가능!

- int : parameter의 **형태**는 정수(integer)
- article\_id : view.detail에게 **넘겨줄 때의 이름**

# Blog

## 3. detail 페이지

urls.py

```
path('detail/<int:article_id>', view.detail, name='detail')
```

views.py

```
def detail(request, article_id):  
  
    return render(  
        )
```

→ url에 포함된 article\_id를 전달 받는 views.detail

### views.py의 detail 함수와 detail.html을 직접 작성해보기!

- 글의 제목과 내용 볼 수 있게
- 전체 글 목록으로 돌아갈 수 있게

detail.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    <h2>글 상세 페이지</h2>  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  </body>  
22 </html>
```

# Blog

## 3. detail 페이지

views.py

```
def detail(request, article_id):
    article = Article.objects.get(id=article_id)
    return render(request, 'detail.html', {'article': article})
```

detail.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         <h2>글 상세 페이지</h2>
11         <div>
12             <h3>제목</h3>
13             <div>{{ article.title }}</div>
14         </div>
15         <div>
16             <h3>내용</h3>
17             <div>{{ article.content }}</div>
18         </div>
19         <a href="{% url 'list' %}">전체 목록으로 돌아가기</a>
20     </body>
21 </html>
```

# Blog

+ list와 detail 페이지 연결

list.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    <h2>글 전체 목록</h2>
11    <ul>
12      {% for article in articles %}
13        <li>
14          <a href="{% url 'detail' article.id %}">{{ article.title }}</a>
15        </li>
16      {% endfor %}
17    </ul>
18    <a href="{% url 'new' %}">글 쓰러 가기</a>
19  </body>
20 </html>
```

**{% url 'detail' article.id %}**

name이 'detail'인 url 패턴에 첫 번째 url parameter에는 article.id 값을 대입하여 요청한다

urls.html

```
urlpatterns = [
    path('admin/', admin.site.urls),
    path('new', views.new, name='new'),
    path('list', views.list, name='list'),
    path('detail/<int:article_id>', views.detail, name='detail'),
]
```

➡ localhost:8000/detail/1 로 접근하는 것과 동일

Blog 완성!!!!!!!



# 과제 공지 🤗

## 1. 블로그 프로젝트 수정하기

- 블로그 게시글에 "카테고리 " 추가 : 취미(hobby), 음식(food), 프로그래밍(programming)
- List 페이지에는 카테고리 별 게시판으로 이동하는 링크와 각 카테고리의 게시글 갯수 보여주기
- 각 카테고리 별 게시판에는 해당 카테고리의 게시글 목록만 보여주기
- Detail 페이지에서는 글 제목, 내용, 글 작성시각 보여주기
- CSS도 예쁘게 꾸며서 적용해보기

~4/4 (월) 세션 전까지

# 과제 공지 🤗

## 2. DB schema diagram 그리기

- 원하는 대상(장소, 집단, 서비스 등)을 자유롭게 하나 정하고 대상의 구성요소를 생각하며  
DB schema diagram 그리기
- 단, 테이블 개수 5개 이상!
- 그리는 툴은 무관 ( 파워포인트, dbdiagram.io, 손 그림 등 )
- 인터넷에서 예시 참고하여 따라 그려도 무관

~4/4 (월) 세션 전까지