Django - 세션 Day6

멋사 세션 6일차 : Django에 대해 복습시간

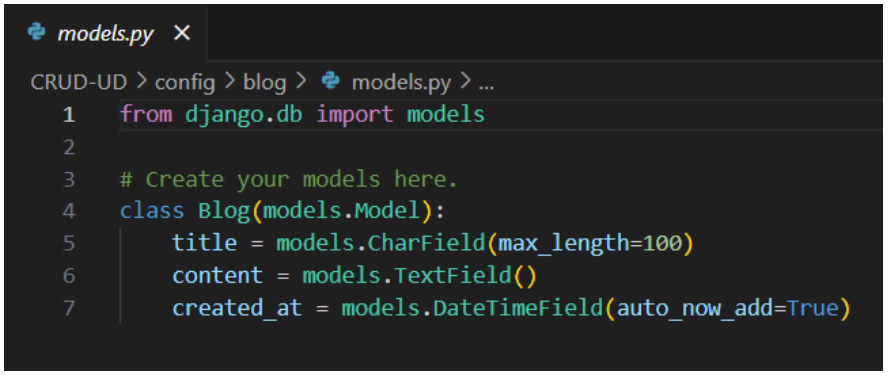
Django 기초

기초 정리 보러가기

[0]. DB, id, PK가 뭘까?

id (pk)	title	content	created_at
1	제목1	내용1	2024-01-01
2	제목2	내용2	2024-01-01
3	제목1	내용1	2024-01-01

DB : 데이터가 저장되는 공간



지난 시간 blog라는 모델 안에 저장한 DB

여기서 작성한 class는 위의 표의 형태로 DB로 저장되고

id를 개발자가 직접 넣어줘도 되고

지정하지 않더라도 알아서 지정이된다.

ID : DB 내의 행의 식별 값

PK : 데이터를 식별할 수 있는 유일한 값

위의 표를 예시로 봤을 때 제목이나 내용같은 부분이 같은 내용이라면 특정 데이터를 식별해 낼 수 없음

따라서 id를 식별자로 사용한다 (pk로 삼는다)

---

[1] MTV 란?

Model / Template / View / (URL)

이해를 돕기 위해 레스토랑으로 비유해서 알아보자

- **url = 식탁**

1. 손님으로부터 요청을 받을 식탁
2. 손님에게 응답(음식 + 그릇)을 보낼 식탁

→ url은 유저에게 요청을 받는 곳, 응답을 보내주는 곳

- **view = 요리 행위**

3. 손님으로부터 받은 요청으로 시작되는 요리 행위
4. 손님에게 보낼 응답(음식 + 그릇)을 만들어내는 요리 행위

▼ 여기서 응답은 응답 객체

template에서 완성한 음식이 담긴 그릇에서 플레이팅까지 완성된 상태 !

→view는 url을 통해 웹 요청, model에게 데이터를 전달 받음

→ 데이터들을 가공하여 template에게 보내주며, 지정된 template를 렌더링해준다

- **model = 냉장고 매니저**

5. 요리 재료를 특정한 규격으로 지정해주는 냉장고 매니저
6. 요리 재료를 저장하거나 꺼내줄 수 있는 냉장고 매니저

→ model은 데이터의 형태를 지정해주고 , DB에 접근해준다

- **template = 그릇**

7. 요리 행위로 만들어진 음식 (응답x)을 담는 그릇

▼ 응답 객체는 아님

응답 객체는 아니고 완성된 요리만 담겨있는 그릇이라고 생각하면 된다 (아직 플레이팅을 완성하지 못한 상태)

→ template는 유저에게 보여질 시각적 부분을 가공

-순서-

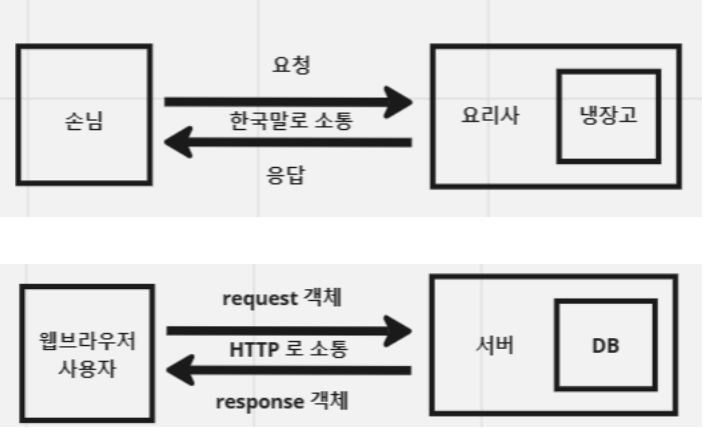
(model) → url → view → model → template → view → url

▼ (5) → 1 → 3 → 6 → 7 → 4 → 2

- **(5) :** 원하는 데이터의 형태를 지정해준다 - 손님이 오시기 전에 해야할 일 *냉장고-model
- **1 :** 유저에게 특정 url을 통해 요청 받는다 *식탁-url

  요청(urlConf)을 통해 해당 url과 매핑된 view를 호출한다
- **3 :** 호출된 view는 요청에 따라 적절한 로직을 수행, 그 과정에서 model에게 CRUD를 지시 *요리 행위-view
- **6 :** model은 ORM을 통해 DB와 소통하며 CRUD를 수행 *냉장고-model
- **7 :** template에서 유저 화면에 직접적으로 보여질 시각적인 부분을 처리 *그릇-template
- **4 :** 그 후 view는 지정된 template를 렌더링 *요리 행위-view
- **2 :** 최종 결과를 url을 통해 응답으로 반환 *식탁-url

[2] response, HTTP



- 요청과 응답은 request 객체와 response객체로 보낸다
- HTTP프로트콜을 사용해서 request, response객체를 보내준다

> 💡 **HTTP**
> 클라이언트와 서버 사이에 이루어지는 요청/응답 프로토콜로, 웹 페이지를 로드하는데 사용된다
>
> HTTP를 통해 전달되는 자료는 http: 로 시작하는 URL로 조회할 수 있다

실습 예제를 보며 앞선 내용을 살펴보장

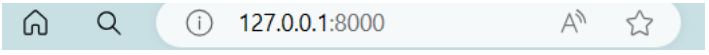
home이라는 게시글 목록 페이지 코드 (read)

- **urls.py**

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

urlpatterns = [
    # READ
    path('', views.home, name='home'),
    # DETAIL READ
    path('blog/<int:blog_id>', views.detail, name='detail'),
    # CREATE
    path('blog/new', views.new, name='new'),
    path('blog/create', views.create, name='create'),
    # UPDATE
    path('blog/edit/<int:blog_id>', views.edit, name='edit'),
    path('blog/update/<int:blog_id>', views.update, name='update'),
    # DELETE
    path('blog/delete/<int:blog_id>', views.delete, name='delete'),
]
```

path(' ') → 기본 베이스 url을 뜻한다.



요 상태를 말함

지정해둔 views에 있는 home라는 놈을 시작해라

그리고 밑에 나올 view, model,template를 통해 완성된 응답 객체를 마지막에 반환해주는 역할까지 한다.

- **views.py**

```python
from django.shortcuts import render, redirect, get_object_or_404
from .models import Blog

# READ
def home(request):
    blogs = Blog.objects.all()
    return render(request, 'home.html', {'blogs': blogs})
```

model에 있는 Blog에서 모든 객체들을 가져와라 (DB에서 재료 가져오기)

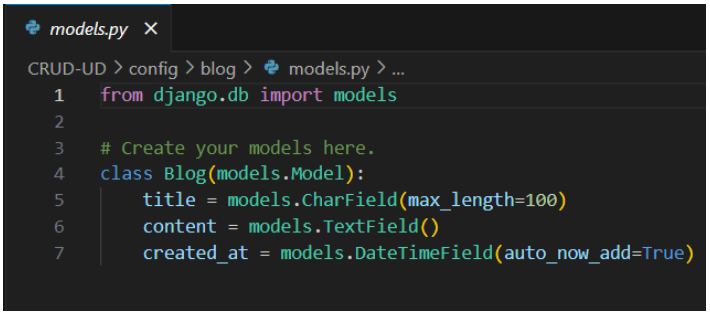
그리고 model에서 가져온 결과물을 home.html 이라는 이름의 template 그릇에 담아라 (바인딩)

```
(function) def render(
    request: HttpRequest,
    template_name: str | Sequence[str],
    context: Mapping[str, Any] | None = ...,
    content_type: str | None = ...,
    status: int | None = ...,
    using: str | None = ...
) -> HttpResponse
```

Return an HttpResponse whose content is filled with the result of calling django.template.loader.render_to_string() with the passed arguments.

request객체

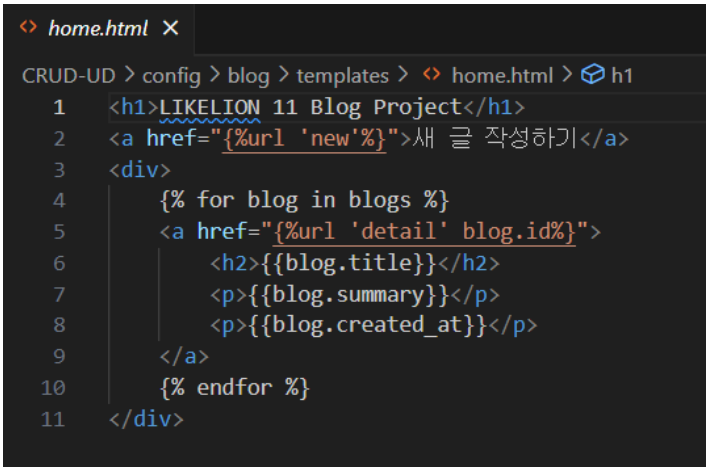
- **models.py**



model에서 지정해둔 Blog객체 (냉장고 저장 역할!)

title은 charField, content는 TextField, create_at은 DateTimeField 형태로 저장해 라는 의미

- **templates > home.html**



view에서 blogs라는 데이터가 넘어와서

시각적으로 보여질 부분에 담아짐

---

[3] CRUD

-CRUD

- **C**reate : DB에 데이터를 생성함
- **R**ead : DB에서 데이터를 읽어옴
- **U**pdate : DB의 특정 데이터를 수정함
- **D**elete: DB의 특정 데이터를 삭제함

-Create-

1. **게시글을 작성하는 페이지 (new)**
2. **작성한 데이터를 저장하는 기능(create)**

[1] new

1-1. URL - new라는 별명을 가진 url 요청을 받고 매핑된 view 호출

1-2. View - new.html 라는 템플릿을 불러오는 로직 수행

1-3. Template - form태그를 사용해 사용자가 입력할 폼을 만들어주고 submit버튼을 만들어줌,

*제출을 누르면 'create'라는 별명의 url에 요청이 보내지도록 form태그에 action값과 method를 설정해줌

1-4. View - new.html을 리턴

> 💡 View
> 처음에 뜨는 화면은 아직 데이터가 들어있지 않은(사용자가 값을 입력하기 전)상태이므로, 데이터를 가져오지도 않고 바인딩도 없이 간단하게 new.html만 request객체로 반환해주고 있다

[2] create

2-1. URL - 앞선 new.html에서 제출 버튼을 눌러 'create' url에서 요청을 받은 후, view호출

2-2. View - 빈 model 객체 선언 후 new_blog라는 변수에 담아줌. post로 넘어온 값들을 넣어줘서 데이터를 생성 후, save()를 통해 DB에 저장

2-3. View - new_blog의 id값을 'detail'이라는 URL로 redirect

> 💡 - render : POST(보내는 요청) 그대로
> - redirect : POST → GET(받는요청) 변경

———Create ———

-Read-

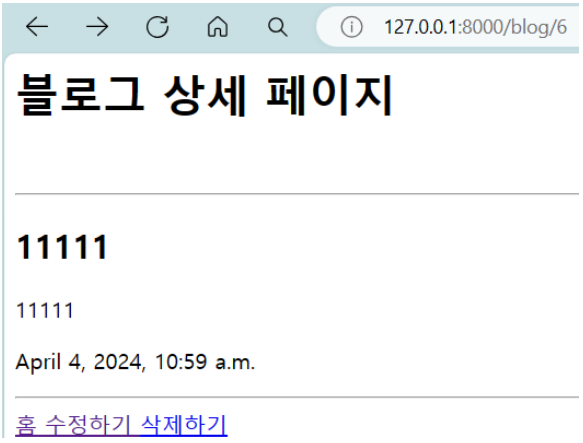
- 전체 목록 페이지 home
- 상세 페이지 detail

*전체 목록 페이지를 읽어서 보여주는 home은

사용자가 게시물 값을 입력하여 데이터를 생성하고 저장하는 create 과정 전에 기본 페이지를 깔아두기 위해 미리 생성했었다. 위의 내용에서 다뤘었음

[detail - 상세 페이지]

1. URL - create 과정에서 사용자가 제출해서 저장된 데이터의 new_blog의 id값이 'deatil' url로 요청받음
2. View - 요청 객체(request)와 블로그의 id값을 매개변수로 받고 get_object_or_404 함수를 사용하여 model에서 pk=blog_id와 일치하는 객체를 찾아 가져온 후 blog라는 변수에 저장
   *만약 없다면 404 에러 표시
3. template - detail.html 에서 홈('home' url),수정하기('edit' url),삭제하기('delete' url) 로 갈 수 있는 a링크를 설정해둔다
4. View - pk = blog_id인 객체가 담긴 blog 변수와 , datil.html을 가지고 request객체로 url에게 반환해줌
5. URL - 'blog/<int:blog_id>'로 경로를 설정하여 정수형 blog id를 나타낸다



——Read———

-Update-

1. 기존의 데이터 불러오기 -edit 페이지
2. 수정된 데이터로 업데이트하기 -update

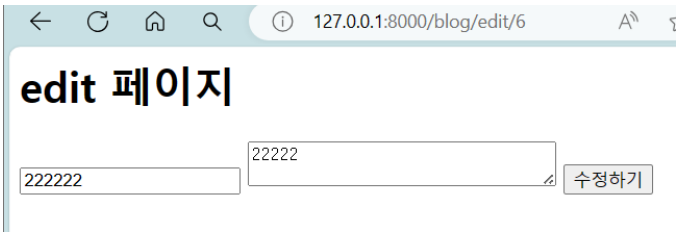
[1] edit 게시물 수정 페이지

1-1. URL - 수정하기 a링크를 통해 request 객체 요청 받음

1-2. View - get_object_or_404 함수를 통해 model에서 blog_id와 pk가 일치하는 것을 가져와 edit_blog라는 변수에 담아줌

1-3. Template - submit 눌렀을 때 update로 전달 되도록 form태그 안에 action과 method 설정해주기, form 태그 안에 수정하기 submit 인풋 만들어주기

1-4. View - edit_blog 변수와, edit.html 파일을 렌더링 해준다.

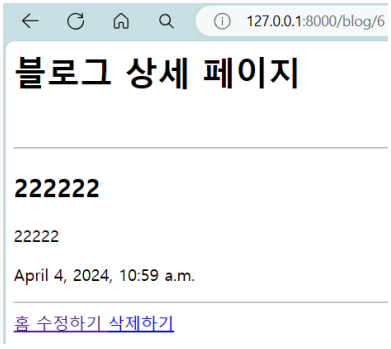


[2] update

2-1. URL - submit form을 통해 request 객체를 'update' url에 요청 받음

2-2. View - blog_id와 pk가 일치하는 데이터를 model로부터 받아온 후 old_blog라는 변수 선언,

post로 넘어온 값들을 넣어줘서 데이터를 생성 후, save()를 통해 DB에 저장
old_blog의 id값을 'detail' url로 넘겨줘서(redirect) 수정되면 블로그 상세 페이지에서 변경된 걸 볼 수 있도록 해준다



-———Update———

-Delete-

상세 페이지에서 삭제하기를 누르면 기존의 객체의 id를 불러와서 삭제 후 목록 페이지인 home으로 보내준다.

[delete]

1. URL - 상세 페이지에서 a링크를 통해 request 요청 전달받음
2. View - model에서 blog pk를 불러온 후 delete_blog 변수에 담아줌
   delete_blog.delete() 메소드 사용해서 삭제한 후 'home'으로 redirect
3. 'home' 의 url경로를 통해 다시 목록 페이지로 돌아감