

1:N, Django Template Inheritance

10번째 세션

NEXT X LIKELION 김동민

저번 시간에는...

게시판 만들기!

게시글 리스트 페이지

게시글 작성하기

게시글 목록

첫번째 게시물

두번째 게시물

안녕하세요~~

정고 화이팅

게시글 목록(List)

게시글 상세 페이지

뒤로가기 | 수정하기 | 삭제하기

제목: 첫번째 게시물

내용: 안녕

게시글 상세(Detail)

게시글 수정 페이지

뒤로가기

제목
첫번째 게시물

내용
안녕

수정

게시글 수정(edit)

출처: 빗한영.<장고복습-박한영>.p19.

오늘의 목표

일찍 끝내기(?)

0. 1: N 데이터베이스 구조 알아보기

1. 댓글 기능 구현하기!

2. 템플릿 상속 알아보기

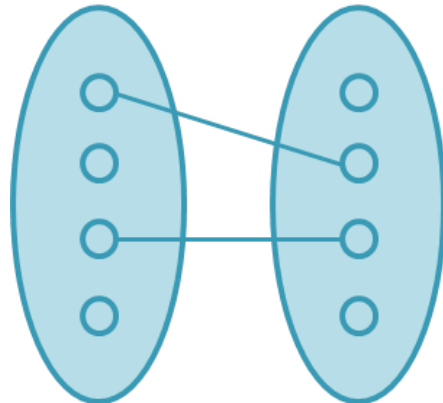
1:N이란?

1:N 데이터베이스 구조

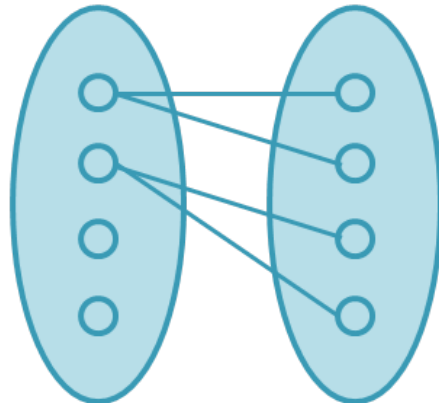


1:N이란?

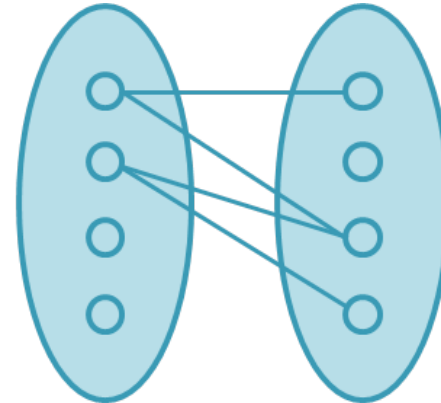
1:N 데이터베이스 구조



일대일(one-to-one)



일대다(one-to-many)



다대다(many-to-many)

- 하나의 데이터가 여러 개의 데이터와 관계를 맺는 구조!
- 글-댓글 / 고객-상품 등 많은 관계에 적용 가능합니다!

1:N이란?

ForeignKey 복습

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

<학생>				<교수>		
학번	이름	학과	지도교수 번호	교수 번호	이름	연락처
2019130528	이민용	사회학과	1	1	허영범	010-1111-1111
2016130514	김동민	사회학과	1	2	이한주	010-9999-9999
2013230152	서인재	신소재공학과	2	3	이소영	010-2222-2222
2018130512	양효령	사회학과	1			
2020230134	김이린	신소재공학과	2			
2020130511	권규리	사회학과	1			
2021130507	김지성	사회학과	1			

Foreign key

참조 대상 테이블의 튜플을 식별할 수 있는 키

🔥 "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

출처: 갯민용.<session6>.p12, 13.

1:N이란?

게시글 && 댓글 관계

댓글(Comment)

id	content	post_id	
1	와 1번이다!!	1	→
2	어? 왜 2번 써지지...?	2	→
3	어? 왜 2번 써지지...?	2	→
4	2개는 뭔가	3	→
5	아쉬우니까요ㅎ	3	→

게시글(Post)

id	title	content
1	첫 번째 글	1번!
2	2번째글2번째글	2번2번!!
3	세 번째 글!	나는 3번!

1:N이란?

게시글 && 댓글 관계

댓글(Comment)

id	content	post_id
1	와 1번이다!!	1
2	어? 왜 2번 써지지...?	2
3	어? 왜 2번 써지지...?	2
4	2개는 뭔가	3
5	아쉬우니까요ㅎ	3

게시글(Post)

id	title	content
1	첫 번째 글	1번!
2	2번째글2번째글	2번2번!!
3	세 번째 글!	나는 3번!

Django 1:N 실습

저희도 만들어 봅시다!!



Django 1:N 실습

실습 준비 (1 / 2)

<Session10.zip> 압축 해제 후

```
$ cd session10
```

```
$ pipenv shell
```

```
$ pipenv install (Pipfile.lock 정보로 가상환경에 모듈 설치!)
```

```
$ cd project
```

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

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

**** 이 단계에서 'Post' 모델이 생성되지 않았다면?**

```
$ python manage.py makemigrations app / python manage.py migrate
```

```
$ python manage.py createsuperuser
```

```
$ python manage.py runserver
```

Django 1:N 실습

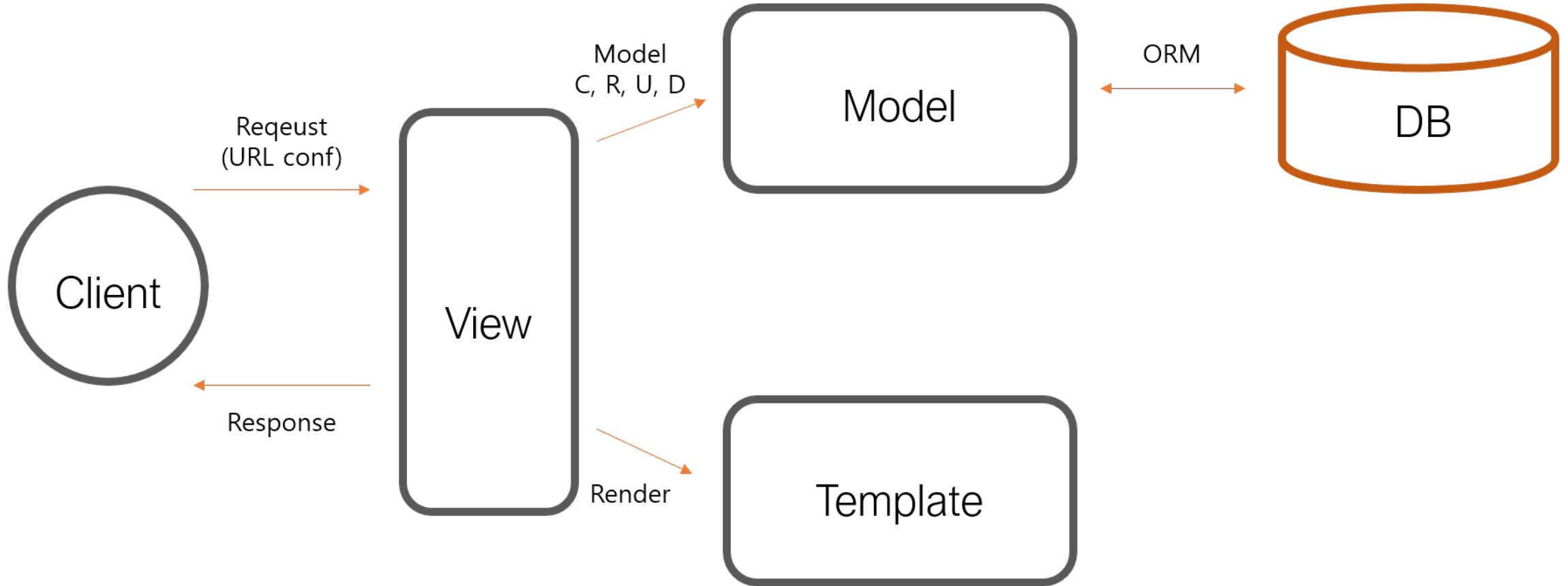
실습 준비(2 / 2)

HOME!

글 쓰러 가기!

Django 1:N 실습

MTV 패턴 복습



Model 생성 – **T**emplate 작성 - **V**iew 작성 – URL 연결!

```
1  from django.db import models
2
3  ✓ class Post(models.Model):
4      title = models.CharField(max_length=50)
5      content = models.TextField()
6
7  ✓ def __str__(self):
8      return self.title
9
10
11  ✓ class Comment(models.Model):
12      post = models.ForeignKey(Post, on_delete=models.CASCADE, related_name='comments')
13      content = models.TextField()
14
15  ✓ def __str__(self):
16      return self.content
17
```

✓ Python manage.py makemigrations, migrate 잊지 말고 해 주세요 ☺

on_delete?

On_delete 옵션 종류

- **CASCADE** : 참조 대상인 Post가 삭제되면, 해당 Post를 참조하고 있던 Comment들도 따라서 삭제!
- **SET_NULL** : 참조 대상인 Post가 삭제되면, 해당 Post를 참조하고 있던 Comment의 ForeignKey를 NULL값으로! (**null=True)
- **SET_DEFAULT** : 참조 대상인 Post가 삭제되면, 해당 Post를 참조하고 있던 Comment의 ForeignKey를 '기본값'으로! (**기본값이 설정되어 있어야겠죠?!)
- **PROTECT** : 참조 대상인 Post를 삭제하려 하면, Protected Error를 띄워 삭제를 막습니다!

※ 기타 : DO_NOTHING, RESTRICT....



<https://docs.djangoproject.com/en/4.0/ref/models/fields/#django.db.models.ForeignKey>

Django 1:N 실습

admin.py

Admin.py에 등록해 주어야 어드민 페이지에서 확인할 수 있어요!!

```
1 from django.contrib import admin
2 from .models import Post, Comment
3
4 # Register your models here.
5 @admin.register(Post)
6 class PostAdmin(admin.ModelAdmin):
7     pass
8
9 admin.site.register(Comment)
```

??



admin 페이지 커스터마이징

modelAdmin

```
class Example(models.Model):  
    title = models.CharField(max_length=50)  
    content = models.TextField()  
    date = models.DateTimeField()  
    blank_field = models.CharField(max_length=50, null=True, blank=True)
```

```
@admin.register(Example)  
class ExampleAdmin(admin.ModelAdmin):  
    list_display = ('title', 'content', 'blank_field') #목록에 표시할 필드!  
    empty_value_display = '입력값 없음' # 비어있는 값을 이렇게 보고 싶다!
```

따라 치지 마시고 보시기만 하셔도 됩니다 ☺

admin 페이지 커스터마이징

List_display, empty_value_display

Home > App > Examples

APP

- Comments + Add
- Examples + Add
- Posts + Add

AUTHENTICATION AND AUTHORIZATION

- Groups + Add
- Users + Add

✓ The example "Example object (2)" was added successfully.

Select example to change ADD EXAMPLE +

Action: 0 of 1 selected

<input type="checkbox"/>	TITLE	CONTENT	BLANK FIELD
<input type="checkbox"/>	제목제목	내용내용	입력값 없음

1 example

List_display!

Empty_value_display!

댓글을 입력할 form을 생성해요!!

```
26 <div class="detail-post_btn">
27     <a href="{% url 'home' %}">홈 화면</a>
28     <a href="{% url 'edit' post.pk %}">수정하기</a>
29     <a href="{% url 'delete' post.pk %}">삭제하기</a>
30 </div>
31
32 <form method="POST">
33     {% csrf_token %}
34     <input type="text" name="content" placeholder="댓글을 입력해주세요">
35     <button type="submit">작성하기</button>
36 </form>
```

DETAIL!

제목: 첫 번째 글!

내용: 1번!

홈 화면 수정하기 삭제하기

댓글을 입력해주세요 작성하기

입력받은 댓글을 저장할 코드를 작성해요!

```
1 from django.shortcuts import render, redirect
2 from .models import Post, Comment
3
```

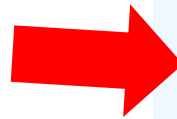
```
def detail(request, post_pk):
    post = Post.objects.get(pk=post_pk)

    if request.method == 'POST':
        content = request.POST['content']
        Comment.objects.create(
            post=post,
            content=content
        )
        return redirect('detail', post_pk)

    return render(request, 'detail.html', {'post': post})
```

Model 생성 - Template 작성 - View 작성 완료!!

이제 작성된 댓글을 화면에 띄워 볼까요?!



DETAIL!

제목: 첫 번째 글!

내용: 1번!

[홈 화면](#) [수정하기](#) [삭제하기](#)


[작성하기](#)

- 거의 삭제
- 다 삭제
- 왔어요! 삭제
- 파이팅!! :) 삭제

Detail.html 작성

related_name

```
31
32     <form method="POST">
33         {% csrf_token %}
34         <input type="text" name="content" placeholder="댓글을 입력해주세요">
35         <button type="submit">작성하기</button>
36     </form>
37
38     {% for comment in post.comments.all %}
39     <li>
40         {{comment.content}}
41     </li>
42     {% endfor %}
43 </div>
44 </div>
45 </body>
46 </html>
47
```



Detail.html 작성

related_name

Models.py

```
class Comment(models.Model):
    post = models.ForeignKey(Post, on_delete=models.CASCADE, related_name='comments')
    content = models.TextField()

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

views.py

```
def detail(request, post_pk):
    post = Post.objects.get(pk=post_pk)

    if request.method == 'POST':
        content = request.POST['content']
        Comment.objects.create(
            post=post,
            content=content
        )
        return redirect('detail', post_pk)

    return render(request, 'detail.html', {'post':post})
```

detail.html

```
{% for comment in post.comments.all %}
<li>
    {{comment.content}}
</li>
{% endfor %}
```

※ 전달받은 'post'와 '관련된' 'comments'를 '전부(all)' 가져와서, '각각 (for comment)'에 대해 'content'를 띄워줘!

Detail.html 작성

댓글 확인 완료!

DETAIL!

제목: 첫 번째 글!

내용: 1번!

홈 화면

수정하기

삭제하기

댓글을 입력해주세요

작성하기

- 거의
- 다
- 왔어요!
- 파이팅!! :)

댓글 삭제 구현하기

구현 순서!

- ~~Model 생성~~



이미 완료!

- Template 작성



템플릿에 <댓글 삭제> UI 만들기!

- View 작성



댓글 삭제 함수 작성하기!

- URL 연결



Views의 댓글 삭제 함수와
연결될 URL 작성하기!

댓글 삭제 구현하기

detail.html

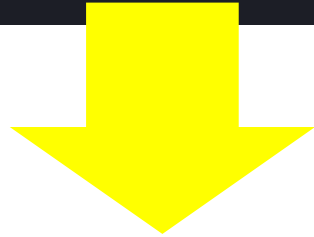
```
31
32 ✓ <form method="POST">
33     {% csrf_token %}
34     <input type="text" name="content" placeholder="댓글을 입력해주세요">
35     <button type="submit">작성하기</button>
36 </form>
37
38 {% for comment in post.comments.all %}
39 ✓ <li>{{ comment.content }}
40     <a href="">삭제</a>
41 </li>
42 {% endfor %}
43 </div>
44 </div>
```

URL은 아직 작성하지 않았어요!!!

댓글 삭제 구현하기

Views.py

```
def delete_comment(request, post_pk, comment_pk):  
    comment = Comment.objects.get(pk=comment_pk)  
    comment.delete()  
    return redirect('detail', post_pk)
```



- ✓ Comment 삭제 후 “다시” detail 페이지로 진입!
- ✓ 새로고침하지 않아도 삭제 내역이 반영된 결과를 볼 수 있어요! 😊

댓글 삭제 구현하기

URL 연결하기

```
urlpatterns = [  
    path('admin/', admin.site.urls),  
    path('', views.home, name='home'),  
    path('new/', views.new, name='new'),  
    path('detail/<int:post_pk>', views.detail, name='detail'),  
    path('edit/<int:post_pk>', views.edit, name='edit'),  
    path('delete/<int:post_pk>', views.delete, name='delete'),  
    path('delete-comment/<int:post_pk>/<int:comment_pk>', views.delete_comment, name='delete-comment'),  
]
```

```
{% for comment in post.comments.all %}  
<li>{{comment.content}}  
    <a href="{% url 'delete-comment' post.pk comment.pk %}">삭제</a>  
</li>  
{% endfor %}  
</div>  
</div>
```

댓글 삭제 구현하기

댓글 삭제 구현 완료!

DETAIL!

제목: 첫 번째 글!

내용: 1번!

홈 화면

수정하기

삭제하기

댓글을 입력해주세요

작성하기

- 댓글 삭제
- 기능 삭제
- 완성!! 삭제
- 와아아아아아앙 삭제

1:N 실습 끝!

수고하셨습니다~!



하지만 아직 갈 길이 남았어요...!

Django Template Inheritance

템플릿 상속

Template Inheritance

Django Template Engine의 가장 강력한 기능!

Template inheritance

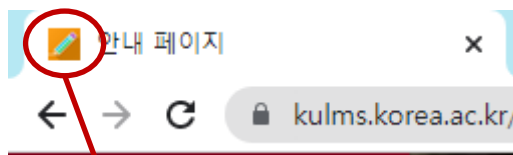
The most powerful – and thus the most complex – part of Django's template engine is template inheritance. Template inheritance allows you to build a base "skeleton" template that contains all the common elements of your site and defines blocks that child templates can override.

- ✓ 모든 페이지에 들어가는 공통 요소들을 담은 Base Template을 만들고
- ✓ Base Template을 상속하는 자식 템플릿들이

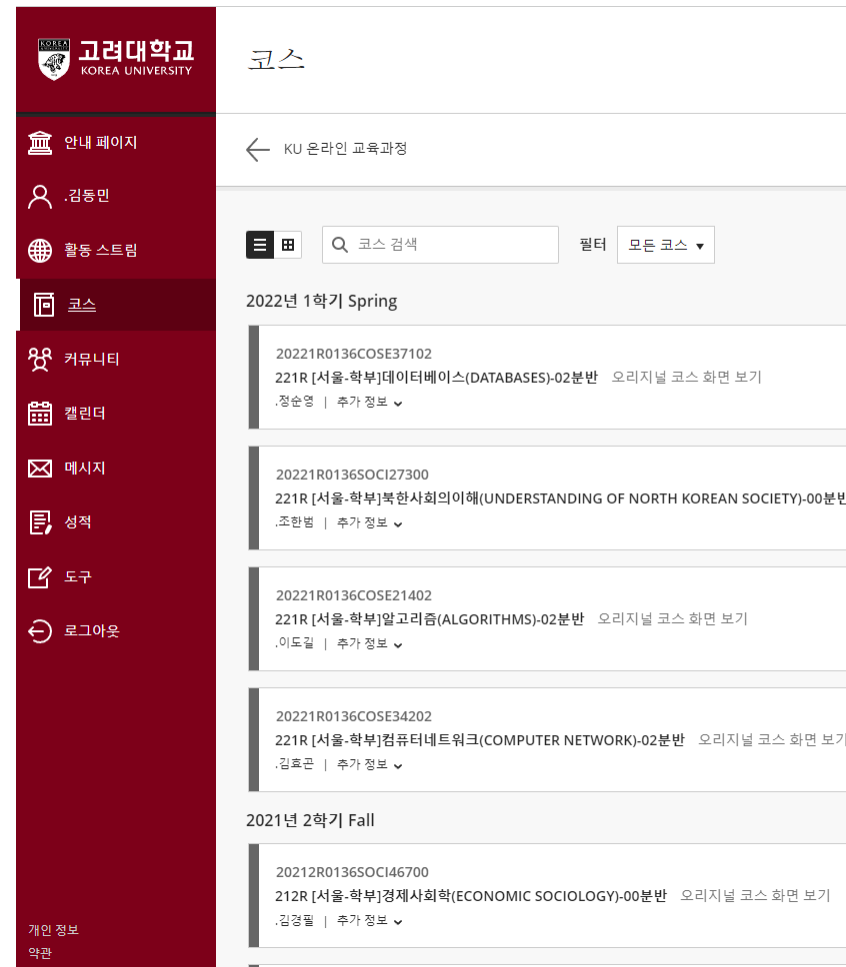
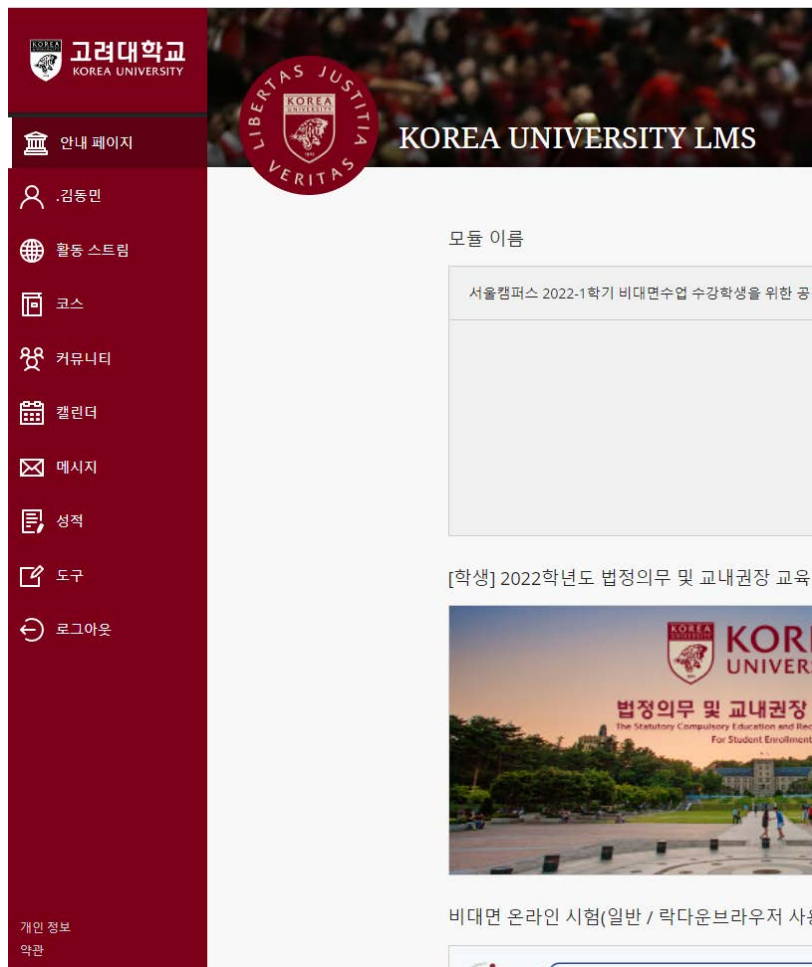
덮어 써서(override) 사용할 수 있는 block을 만들 수 있게 하는 기능!

Q. 왜 템플릿 상속을 쓰나요?

A. 중복되는 요소들을 매번 html에 추가하면서 생기는 비효율성을 줄이기 위함! ☺



Favicon!



모든 페이지에 공통으로 들어가는 부분!



Base Template

페이지마다 달라지는 부분!



각각의 Template

HOME!

제목: 마지막 글?
내용: 내용!

글 쓰기 가기!

NEW!

제목을 입력하세요!

내용을 입력하세요!

작성하기

EDIT!

마지막 글?

내용!

수정하기

NEW!

제목을 입력하세요!

내용을 입력하세요!

작성하기

겹치는 부분을 Base로!!

중복되는 부분?!

모든 페이지에 공통인 부분!

#1. 제목 서식!

HOME!

제목: 마지막 글?!
내용 : 내용!

#2. 배경 틀!

글 쓰러 가기!!

다른 부분?

페이지마다 다른 부분!

#1. 제목 '내용'!

HOME!

제목: 마지막 글?!
내용 : 내용!

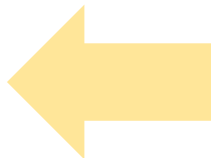
#2. 본문 내용!

글 쓰러 가기!

장고 템플릿 문법

Template Inheritance

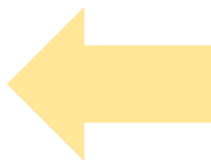
- `{% extends 'base.html' %}`



- ✓ `base.html`에서 틀을 짜고!
- ✓ 나머지 자식 템플릿에서는 `base.html`을 `extends`(상속)합니다!
- ✓ “이 페이지는 `base.html`을 상속한다”라는 선언!

- `{% block 태그이름 %}`

- `{% endblock 태그이름 %}`



- ✓ 페이지별로 바뀌는 부분은 `block`으로 감싸서
- ✓ 자식 템플릿에서 내용을 바꿔 줍니다 ☺

base.html

block && endblock

<페이지마다 달라지는 내용을 block으로!>

각 페이지에만 있는 엘리먼트만
스타일링해줄 CSS 파일!

제목 내용!

본문 내용!

```
1 <!DOCTYPE html>
2 <html lang="en">
3
4 <head>
5     {% load static %}
6     <meta charset="UTF-8">
7     <link rel="shortcut icon" href="{% static 'favicon.png' %}">
8     <link rel="stylesheet" type="text/css" href="{% static 'base.css' %}">
9     <meta http-equiv="X-UA-Compatible" content="IE=edge">
10    <meta name="viewport" content="width=device-width, initial-scale=1.0">
11    <link rel="preconnect" href="https://fonts.googleapis.com">
12    <link rel="preconnect" href="https://fonts.gstatic.com" crossorigin>
13    <link href="https://fonts.googleapis.com/css2?family=Ceviche+One&family=Inter:wght@900&fam
14    {% block css %}
15    <!-- 자식 템플릿에서, 해당 페이지에 맞는 css 파일 경로가 들어갈 자리! -->
16    {% endblock css %}
17    <title>project</title>
18 </head>
19
20 <body>
21    <div class="title">
22        {% block title %}
23        <!-- 자식 템플릿에서, 해당 페이지에 맞는 내용이 들어갈 자리! -->
24        {% endblock title %}
25    </div>
26    <div class="container">
27        {% block content %}
28        <!-- 자식 템플릿에서, 해당 페이지에 맞는 내용이 들어갈 자리! -->
29        {% endblock content %}
30    </div>
31 </body>
32
33 </html>
```

base.css로 바꿔 주세요!

Q. Css 파일...꼭 나눠야 할까요?

css 블록의 이유

A. base.css에 모든 css코드를 다 넣어도

에러가 나진 않지만,

Css코드가 길어진다면 특정 페이지의 css를

수정할 때 수정할 부분을 찾기 어려운

번거로움이 있을 수 있어요ㅠㅠ

```
1 <!DOCTYPE html>
2 <html lang="en">
3
4 <head>
5     {% load static %}
6     <meta charset="UTF-8">
7     <link rel="shortcut icon" href="{% static 'favicon.png' %}">
8     <link rel="stylesheet" type="text/css" href="{% static 'base.css' %}">
9     <meta http-equiv="X-UA-Compatible" content="IE=edge">
10    <meta name="viewport" content="width=device-width, initial-scale=1.0">
11    <link rel="preconnect" href="https://fonts.googleapis.com">
12    <link rel="preconnect" href="https://fonts.gstatic.com" crossorigin>
13    <link href="https://fonts.googleapis.com/css2?family=Ceviche+One&family=Inter:wght@900&fam
14    {% block css %}
15    <!-- 자식 템플릿에서, 해당 페이지에 맞는 css 파일 경로가 들어갈 자리! -->
16    {% endblock css %}
17    <title>project</title>
18 </head>
19
20 <body>
21     <div class="title">
22         {% block title %}
23         <!-- 자식 템플릿에서, 해당 페이지에 맞는 내용이 들어갈 자리! -->
24         {% endblock title %}
25     </div>
26     <div class="container">
27         {% block content %}
28         <!-- 자식 템플릿에서, 해당 페이지에 맞는 내용이 들어갈 자리! -->
29         {% endblock content %}
30     </div>
31 </body>
32
33 </html>
```

home.html

block && endblock

Home.html을 위한 CSS 파일!

Home 화면의 제목!

Home 화면의 본문!

```
1 {% extends 'base.html' %}
2
3 {% block css %}
4 {% load static %}
5 <link rel="stylesheet" type="text/css" href="{% static 'home.css' %}">
6 {% endblock %}
7
8 {% block title %}
9 Home!
10 {% endblock %}
11
12 {% block content %}
13
14 <div class="posts_box">
15     {% for post in posts %}
16     <a href="{% url 'detail' post.pk %}">
17         <div class="post_wrapper">
18             <div class="post_title">
19                 제목: {{post.title}}
20             </div>
21             <div class="post_content">
22                 내용 : {{post.content}}
23             </div>
24         </div>
25     </a>
26     {% endfor %}
27
28 </div>
29 <a id="post-button" href="{% url 'new' %}">글 쓰러 가기!</a>
30
31 {% endblock content %}
```


base.html

```
1 <!DOCTYPE html>
2 <html lang="en">
3
4 <head>
5     {% load static %}
6     <meta charset="UTF-8">
7     <link rel="shortcut icon" href="{% static 'favicon.png' %}">
8     <link rel="stylesheet" type="text/css" href="{% static 'base.css' %}">
9     <meta http-equiv="X-UA-Compatible" content="IE=edge">
10    <meta name="viewport" content="width=device-width, initial-scale=1.0">
11    <link rel="preconnect" href="https://fonts.googleapis.com">
12    <link rel="preconnect" href="https://fonts.gstatic.com" crossorigin>
13    <link href="https://fonts.googleapis.com/css2?family=Ceviche+One&family=Inter@900&family=Inter@900&family=Inter@900" rel="stylesheet">
14    {% block css %}
15    <!-- 자식 템플릿에서, 해당 페이지에 맞는 css 파일 경로가 들어갈 자리! -->
16    {% endblock css %}
17    <title>project</title>
18 </head>
19
20 <body>
21     <div class="title">
22         {% block title %}
23         <!-- 자식 템플릿에서, 해당 페이지에 맞는 내용이 들어갈 자리! -->
24         {% endblock title %}
25     </div>
26     <div class="container">
27         {% block content %}
28         <!-- 자식 템플릿에서, 해당 페이지에 맞는 내용이 들어갈 자리! -->
29         {% endblock content %}
30     </div>
31 </body>
32
33 </html>
```

home.html

```
1 {% extends 'base.html' %}
2
3 {% block css %}
4 {% load static %}
5 <link rel="stylesheet" type="text/css" href="{% static 'home.css' %}">
6 {% endblock %}
7
8 {% block title %}
9 Home!
10 {% endblock %}
11
12 {% block content %}
13
14 <div class="posts_box">
15     {% for post in posts %}
16     <a href="{% url 'detail' post.pk %}">
17         <div class="post_wrapper">
18             <div class="post_title">
19                 제목: {{post.title}}
20             </div>
21             <div class="post_content">
22                 내용 : {{post.content}}
23             </div>
24         </div>
25     </a>
26     {% endfor %}
27 </div>
28
29 <a id="post-button" href="{% url 'new' %}">글 쓰러 가기!</a>
30
31 {% endblock content %}
```

템플릿 상속 연습

연습해 보아요!!

✓ New.html도 base.html을 상속하도록 바꿔 주세요! 😊

템플릿 상속 활용

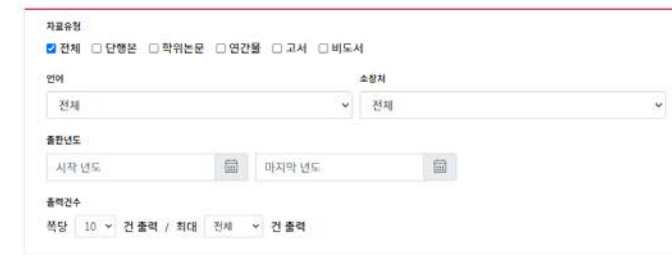
Ex. 고려대학교 도서관!



Header, footer, nav-bar 등

{% block 태그명 %}로

다양하게 활용이 가능합니다!



과제

수고 많으셨습니다! ☺



1) 저번 세션 과제로 구현 중이신 게시판에

댓글 기능을 추가해 주세요 ☺

2) 댓글은 작성과 삭제가 모두 가능하게 해 주세요!

detail_page에서 해당 게시글의 모든 댓글을 볼 수 있어야 합니다!

3) 게시판에 템플릿 상속을 적용해 주세요!

4) 템플릿 상속 적용 방식은 자유!

(ex. Nav-bar 만들기, footer 만들기....)

오늘 한 것처럼 배경 틀만 base.html로 만들어 보셔도 됩니다!

선택) CSS로 예쁘게 만들어 주세요 :D