

멋쟁이사자처럼 2학기 1주차
인스타그램 만들기

멋쟁이사자처럼 박태미

커리큘럼

9월

5회

1학기 내용 복습
+
추가 기능
+
중앙 과제 제출 강의

10월

3회

Django 심화 내용
+
중앙 과제 제출 강의

11월

2회

React 중앙 과제 제출 강의

1학기 내용 복습 + 추가 기능

<인스타그램 만들기>

인스타그램 만들기

<http://britzepps.pythonanywhere.com/>

기능

CR

댓글

회원가입

좋아요

정렬

오늘은 기초를 다져봅시다

순서

가상환경 만들기

가상환경 실행

Django install

프로젝트 만들기

앱 만들기

settings.py 설정하기

Model 구성하기

Admin 등록하기

Base.html 꾸미기

```
python -m venv myvenv
```

```
source myvenv/Scripts/activate
```

```
pip install django
```

```
django-admin startproject instagram
```

```
python manage.py startapp insta
```

```
'insta' / TIME_ZONE = 'Asia/Seoul'
```

```
[]
```

```
admin.site.register(모델명)
```

```
Bootstrap
```

Model 설계

models.py

User

Post

Comment

Like

Model 설계

models.py

User

Post

Comment

Like

상속 :
AbstractUser

[settings.py]

AUTH_USER_MODEL = '앱이름.클래스이름'

변수 이름

필드 이름

profile

ImageField

회원가입

아이디

아이디를 입력해주세요

비밀번호

비밀번호를 입력해주세요

비밀번호 확인

비밀번호를 한번 더 입력해주세요

프로필 사진

파일 선택 선택된 파일 없음

회원가입

Settings.py 에 왜 정의해주나요?

➔ Django가 사용자 정의 모델을 기본 모델 대신 사용하도록 지시

User 모델 확장의 4가지 방법

[illegible]

상속 :
AbstractUser

[settings.py]

AUTH_USER_MODEL = '앱이름.클래스이름'

변수 이름

필드 이름

profile

ImageField

models.py

```
from django.contrib.auth.models import AbstractUser
```

```
class User(AbstractUser):  
    profile = models.ImageField()
```

회원가입

아이디

아이디를 입력해주세요

비밀번호

비밀번호를 입력해주세요

비밀번호 확인

비밀번호를 한번 더 입력해주세요

프로필 사진

파일 선택 선택된 파일 없음

회원가입

settings.py

```
AUTH_USER_MODEL = 'insta.User'
```

Model 설계

models.py

User

Post

Comment

Like



변수 이름

user

image

caption

created

updated

필드 이름

ForeignKey

ImageField

TextField

DateTimeField

DateTimeField

```
class Post(models.Model):
    user = models.ForeignKey(User, on_delete=models.CASCADE)
    image = models.ImageField()
    caption = models.TextField()
    created = models.DateTimeField(auto_now_add=True)
    updated = models.DateTimeField(auto_now=True)
```

```
class Post(models.Model):  
    user = models.ForeignKey(User, on_delete=models.CASCADE)  
    image = models.ImageField()  
    caption = models.TextField()  
    created = models.DateTimeField(auto_now_add=True)  
    updated = models.DateTimeField(auto_now=True)
```

[CASCADE]

ForeignKeyField가 바라보는 값이 삭제될 때
ForeignKeyField를 포함하는 모델 인스턴스(row)도 삭
제된다.

auto_now_add

최초 save에만 날짜 값이 들어감

VS

auto_now

Save 할 때마다 날짜 값이 들어감

참고

Foreign key Field on_delete 종류

1. CASCADE
2. PROTECT
3. SET_NULL
4. SET_DEFAULT
5. SET()
6. DO_NOTHING

Model 설계

models.py

User


Post

Comment

Like

Comment

댓글



댓글

— taemi

댓글을 남겨주세요~!

닫기

댓글 작성

변수 이름

필드 이름

post

ForeignKey

user

ForeignKey

content

TextField

사진은 안 필요?

→ User에서 가져온 것

```
class Comment(models.Model):
    post = models.ForeignKey(Post, on_delete=models.CASCADE)
    user = models.ForeignKey(User, on_delete=models.CASCADE)
    content = models.TextField()
```


Model 설계

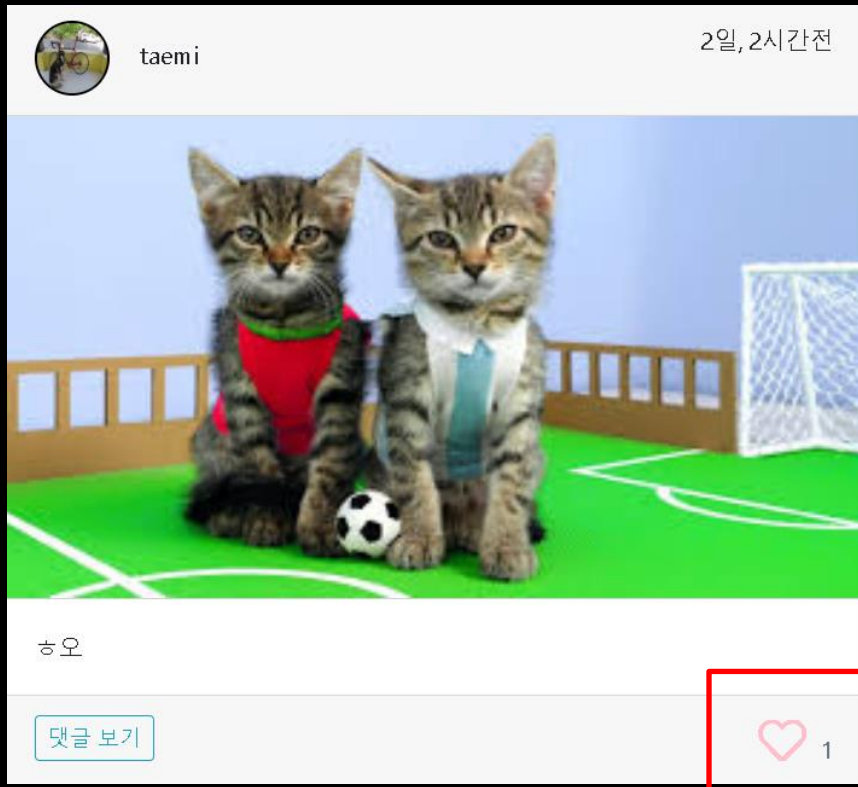
models.py

User

Post

Comment

Like



변수 이름

user

post

필드 이름

ForeignKey

ForeignKey

```
class Like(models.Model):  
    user = models.ForeignKey(User, on_delete=models.CASCADE)  
    post = models.ForeignKey(Post, on_delete=models.CASCADE)
```

모델 작성 끝

```
python manage.py makemigrations
```

```
Python manage.py migrate
```

잘 올라가는지 확인

admin 페이지에서 확인하고 싶으니

admin에 모델들을 등록하자

model admin 등록법

1. 기본 modeladmin으로 등록 (기존 방법)

```
admin.site.register(모델명)
```

2. admin.ModelAdmin 상속을 통해 커스터마이징

3. 장식자(decorator)형태로 등록 가능

ModelAdmin 옵션

`list_display` : Admin 목록에 보여질 필드 목록

`list_display_links` : 목록 내에서 링크로 지정할 필드 목록 (이를 지정하지 않으면, 첫번째 필드에만 링크가 적용)

`list_editable` : 목록 상에서 수정할 필드 목록

`list_per_page` : 페이지 별로 보여질 최대 갯수 (디폴트 : 100)

`list_filter` : 필터 옵션을 제공할 필드 목록

`actions` : 목록에서 수행할 action 목록

`search_fields` : 검색

...

입력 전에 설명 보고 갈게요

```

5  @admin.register(Post)
6  class PostAdmin(admin.ModelAdmin):
7      list_display = ('user', 'image', 'caption', 'created', 'updated', 'like_count')
8      list_filter = ['caption']
9      search_fields = ['caption']
10     fields = ['user', 'image', 'caption']
11

```

변경할 post 선택

액션: 실행 2 중 아무것도 선택되지 않았습니다.

<input type="checkbox"/>	USER	IMAGE	CAPTION	CREATED	UPDATED	LIKE COUNT
<input type="checkbox"/>	taem	literature-3091212_1920.jpg	12	2019년 8월 29일 12:30 오전	2019년 8월 29일 12:30 오전	0
<input type="checkbox"/>	taemi	kelly-sikkema-511604-unsplash.jpg	1	2019년 8월 26일 10:38 오후	2019년 8월 26일 10:38 오후	1

2 posts

필터

caption (으)로

모두

1

12

post 추가

User:

Image:

파일 선택 선택된 파일 없음

Caption:

입력 전에 설명 보고 갈게요

```
5  @admin.register(Post)
6  class PostAdmin(admin.ModelAdmin):
7      list_display = ('user', 'image', 'caption', 'created', 'updated', 'like_count')
8      list_filter = ['caption']
9      search_fields = ['caption']
10     fields = ['user', 'image', 'caption' ]
11
```

Decorator? (@)

Python 문법

대상 함수(클래스)를 wrapping하고,
이 wrapping된 함수의 앞뒤에 추가적으로 꾸며질 구문들을 정의해서
재사용 가능하게 해주는 것.

원소라야

<https://github.com/django/django/blob/master/django/contrib/admin/decorators.py>

```
5 @admin.register(Post)
6 class PostAdmin(admin.ModelAdmin):
7     list_display = ('user', 'image', 'caption', 'created', 'updated', 'like_count')
8     list_filter = ['caption']
9     search_fields = ['caption']
10    fields = ['user', 'image', 'caption']
11
```

(@) decorator

대상함수 (클래스)

decorator

난 이것도 할 거고
저것도 할 거고
[대상함수(클래스)]로 요런것도 할 거야

Python 문법

대상 함수(클래스)를 wrapping하고,
이 wrapping된 함수의 앞뒤에 추가적으로 꾸며질 구문들을 정의해서
재사용 가능하게 해주는 것.

이제 뭐가 뭔지 대충 보이죠? ^-^

```
1  from django.contrib import admin
2  from .models import Post, Comment, User, Like
3
4
5  @admin.register(Post)
6  class PostAdmin(admin.ModelAdmin):
7      list_display = ('user', 'image', 'caption', 'created', 'updated', 'like_count')
8      list_filter = ['caption']
9      search_fields = ['caption']
10     fields = ['user', 'image', 'caption' ]
11
12     @admin.register(Comment)
13     class CommentAdmin(admin.ModelAdmin):
14         list_display = ('post', 'user', 'content')
15         fields = ['post', 'user', 'content']
16
17     @admin.register(Like)
18     class LikeAdmin(admin.ModelAdmin):
19         list_display = ('user', 'post')
20         fields = ['user', 'post']
21
22     @admin.register(User)
23     class UserAdmin(admin.ModelAdmin):
24         pass
```

Admin도 끝

```
python manage.py createsuperuser
```

```
python manage.py runserver
```

<http://127.0.0.1:8000/admin/> 접속합시다

오류 ^__^

Model 조금만 더 손대보기

Post model

```
class Meta:
    ordering = ['-updated']

def __str__(self):
    return '%s - %s' % (self.id, self.user)
```

→ 글 순서 최신순을 기본값

→ Post를 구분할 수 있도록 보여지는 값

Class Meta : 모델에 메타데이터를 추가할 수 있음

정렬 옵션 (ordering)

↓ 다른 옵션들

<https://docs.djangoproject.com/en/1.7/ref/models/options/>

Post:

2 - taem
1 - taemi

```
@property
def like_count(self):
    return self.like_set.count()
```

→ 좋아요 개수 세주기

다시 하면 돌아감

Base html 만들어보기

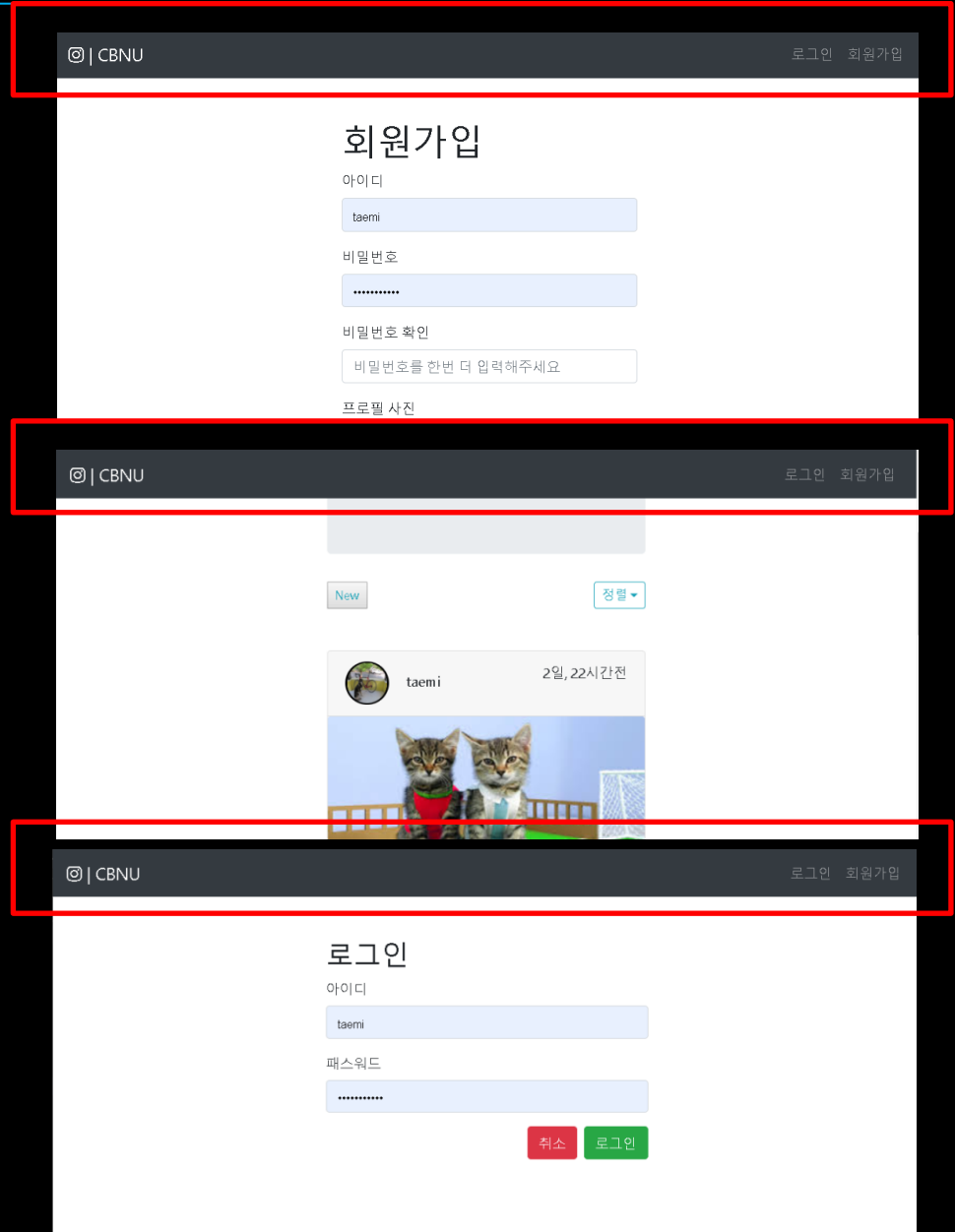
Base.html

공통 : navbar

Html 마다 넣어주어야 하나?

답

Template 상속



base.html

Base.html

Navbar code

```
{% block content %}  
{% endblock %}
```

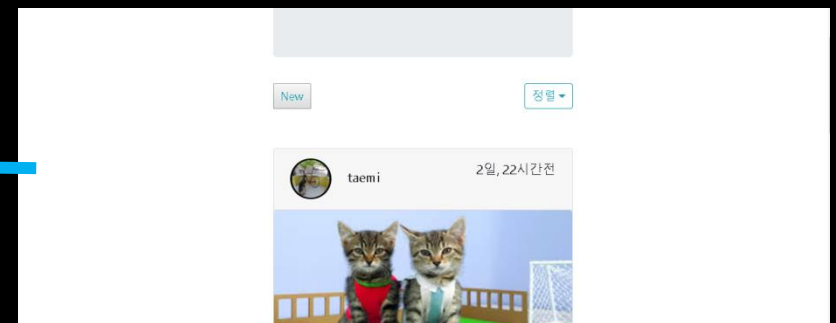
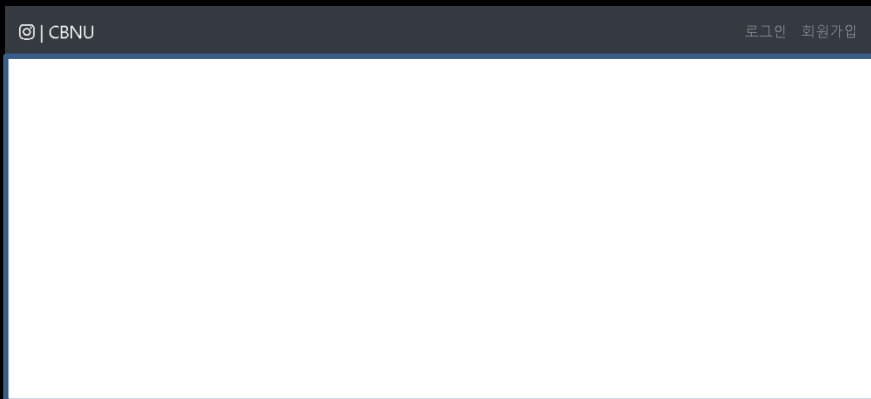
main.html

```
{% extends "base.html" %}
```

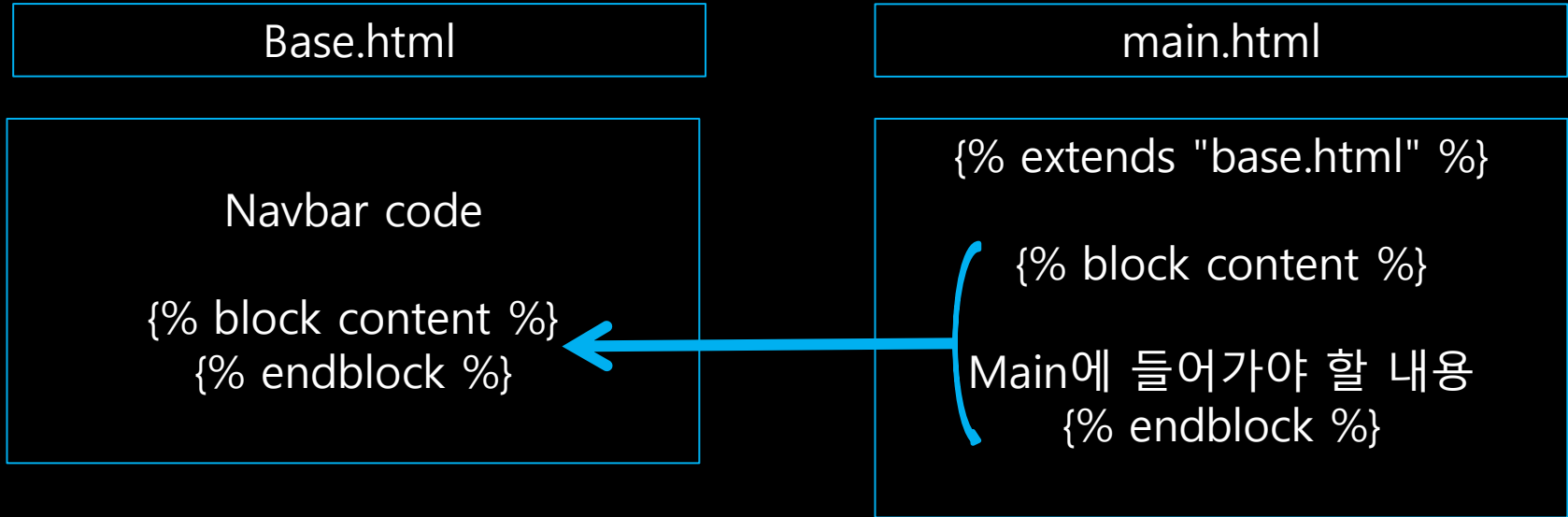
```
{% block content %}
```

Main에 들어가야 할 내용

```
{% endblock %}
```



base.html



직접 해보자

끝