191022 WEB & django -day35

06. Django_advance

04_INSTAGRAM

- `pip install django django_extensions ipython pillow faker django-bootstrap4 pilkit django-imagekit`
    - `pillow`는 이미지 때문에 설치

settings.py

```
"""
Django settings for instagram project.

Generated by 'django-admin startproject' using Django 2.2.6.

For more information on this file, see
https://docs.djangoproject.com/en/2.2/topics/settings/

For the full list of settings and their values, see
https://docs.djangoproject.com/en/2.2/ref/settings/
"""

import os

# Build paths inside the project like this: os.path.join(BASE_DIR, ...)
BASE_DIR = os.path.dirname(os.path.dirname(os.path.abspath(__file__)))


# Quick-start development settings - unsuitable for production
# See https://docs.djangoproject.com/en/2.2/howto/deployment/checklist/

# SECURITY WARNING: keep the secret key used in production secret!
SECRET_KEY = 'iy7a2zgOha33ujz=+9g8^t+sx(*o(%r=8zp34g3@6m3=x=5%tb'

# SECURITY WARNING: don't run with debug turned on in production!
DEBUG = True

ALLOWED_HOSTS = []


# Application definition

INSTALLED_APPS = [
    'bootstrap4',
    'django_extensions',
    'django.contrib.admin',
```

```
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'accounts',
    'postings',
]

MIDDLEWARE = [
    'django.middleware.security.SecurityMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware',
    'django.middleware.common.CommonMiddleware',
    'django.middleware.csrf.CsrfViewMiddleware',
    'django.contrib.auth.middleware.AuthenticationMiddleware',
    'django.contrib.messages.middleware.MessageMiddleware',
    'django.middleware.clickjacking.XFrameOptionsMiddleware',
]

ROOT_URLCONF = 'instagram.urls'

TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': [
            # BASE_DIR/APP/templates/은 자동으로 검색.
            os.path.join(BASE_DIR, 'templates')  # 여기(BASE_DIR/templates/)도 검
색
        ],
        'APP_DIRS': True,
        'OPTIONS': {
            'context_processors': [
                'django.template.context_processors.debug',
                'django.template.context_processors.request',
                'django.contrib.auth.context_processors.auth',
                'django.contrib.messages.context_processors.messages',
            ],
        },
    },
]

WSGI_APPLICATION = 'instagram.wsgi.application'


# Database
# https://docs.djangoproject.com/en/2.2/ref/settings/#databases

DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': os.path.join(BASE_DIR, 'db.sqlite3'),
    }
}


# Password validation
# https://docs.djangoproject.com/en/2.2/ref/settings/#auth-password-validators
```

```
AUTH_PASSWORD_VALIDATORS = [
    {
        'NAME':
'django.contrib.auth.password_validation.UserAttributeSimilarityValidator',
    },
    {
        'NAME':
'django.contrib.auth.password_validation.MinimumLengthValidator',
    },
    {
        'NAME':
'django.contrib.auth.password_validation.CommonPasswordValidator',
    },
    {
        'NAME':
'django.contrib.auth.password_validation.NumericPasswordValidator',
    },
]


# Internationalization
# https://docs.djangoproject.com/en/2.2/topics/i18n/

LANGUAGE_CODE = 'ko-kr'

TIME_ZONE = 'Asia/Seoul'

USE_I18N = True

USE_L10N = True

USE_TZ = True

AUTH_USER_MODEL = 'accounts.User'  # models에서 User를 우리가 직접 만들었기 때문에 덮어씌워주는 것. User 모델을 확장할지 안할지 모르겠으면 일단 써놓고 나중에 필요시 확장하면 됨.


# Static files (CSS, JavaScript, Images)
# https://docs.djangoproject.com/en/2.2/howto/static-files/

STATIC_URL = '/static/'  # html에서 {% static 'My/File' %} => 이게 HTML의 접두사
STATICFILES_DIRS = [
    # BASE_DIR/APP/static/은 자동으로 검색. 추가적으로 assets도 검색
    os.path.join(BASE_DIR, 'assets')
]

MEDIA_URL = '/media/'
MEDIA_ROOT = os.path.join(BASE_DIR, 'media')
```

terminal

```
In [1]: User.dummy(3)

In [3]: star = User.objects.first()
```

```
In [4]: star
Out[4]: <User: Ronald>

In [5]: fan1 = User.objects.last()

In [6]: fan2 = User.objects.get(id=2)

In [7]: fan1
Out[7]: <User: Susan>

In [8]: fan2
Out[8]: <User: Brenda>

In [9]: star.fans.all()
Out[9]: <QuerySet []>

In [10]: star.fans.add(fan1)

In [11]: star.fans.add(fan2)

In [12]: star.fans.all()
Out[12]: <QuerySet [<User: Brenda>, <User: Susan>]>

In [13]: star.stars.all()
Out[13]: <QuerySet []>

In [14]: star.stars.add(fan1)

In [15]: star.stars.all()
Out[15]: <QuerySet [<User: Susan>]>

In [16]: star.stars.add(star)

In [17]: star.stars.all()
Out[17]: <QuerySet [<User: Susan>, <User: Ronald>]>
```

templates - base.html

```
{% load static %}
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <link rel="stylesheet" href="{% static 'css/bootstrap.min.css' %}">
    <title>Fake Insta</title>
    <script src="https://kit.fontawesome.com/7f347b6a32.js"
crossorigin="anonymous"></script>
</head>
<body>
    {% include '_navbar.html' %}
    <div class="container">
        {% block body %}
```

```
        {% endblock %}
    </div>
    <script src="{% static 'js/jquery-3.3.1.slim.min.js' %}"></script>
    <script src="{% static 'js/bootstrap.min.js' %}"></script>
    <script src="{% static 'js/popper.min.js' %}"></script>
</body>
</html>
```

templates - _navbar.html

```
<nav class="mb-3 navbar sticky-top navbar-expand-lg navbar-light bg-light px-5">
    <a class="navbar-brand" href="#">
        <i class="fab fa-instagram"></i> | Instagram
    </a>
    <button class="navbar-toggler" type="button" data-toggle="collapse" data-
target="#navbarSupportedContent" aria-controls="navbarSupportedContent" aria-
expanded="false" aria-label="Toggle navigation">
        <span class="navbar-toggler-icon"></span>
    </button>

    <div class="collapse navbar-collapse justify-content-end"
id="navbarSupportedContent">
        <ul class="navbar-nav">
            {% if user.is_authenticated %}
                <li class="nav-item">
                    <a class="nav-link" href="#">New</a>
                </li>
                <li class="nav-item">
                    <a class="nav-link" href="#">MyPage</a>
                </li>
                <li class="nav-item">
                    <a href="{% url 'accounts:logout' %}" class="nav-
link">Logout</a>
                </li>
            {% else %}
                <li class="nav-item">
                    <a href="{% url 'accounts:signup' %}" class="nav-
link">Signup</a>
                </li>
                <li class="nav-item">
                    <a href="{% url 'accounts:login' %}" class="nav-
link">Login</a>
                </li>
            {% endif %}
        </ul>
    </div>
</nav>
```

urls.py

```
from django.conf import settings
from django.conf.urls.static import static
from django.contrib import admin
from django.urls import include, path

urlpatterns = [
    path('admin/', admin.site.urls),
    path('accounts/', include('accounts.urls')),
    path('insta/', include('postings.urls')),
]

urlpatterns += static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT)
```

accounts - models.py

```
from django.conf import settings
# User 는 AbstractUser를 상속받고 AbstractUser는 AbstractBaseUser를 상속받음
from django.contrib.auth.models import AbstractUser
from django.db import models
from faker import Faker

f = Faker()


class User(AbstractUser):  # 우리가 만든 User를 사용할 것이기 때문에 settings에 덮어씌워줘야 함
    fans = models.ManyToManyField(settings.AUTH_USER_MODEL, related_name='stars')
    # User 모델을 확장할지 안할지 모르겠을 때는 일단 만들어주고 추후에 필요시 확장하면 됨. 그럴 때 class User(AbstractUser): pass를 써두면 됨
    def __str__(self):
        return self.username

    @classmethod
    def dummy(cls, n):
        for i in range(n):
            u = cls()
            u.username = f.first_name()
            u.set_password('123456789')
            u.save()
```

accounts - forms.py

```
from django.contrib.auth import get_user_model
from django.contrib.auth.forms import AuthenticationForm, UserChangeForm, UserCreationForm

User = get_user_model()

class CustomUserCreationForm(UserCreationForm):
```

```
    class Meta(UserCreationForm):
        model = User
        fields = ('username', 'email',)

class CustomAuthenticationForm(AuthenticationForm):
    class Meta:
        model = User
```

accounts - urls.py

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

app_name = 'accounts'

urlpatterns = [
    path('signup/', views.signup, name='signup'),
    path('login/', views.login, name='login'),
    path('logout/', views.logout, name='logout'),
]
```

accounts - views.py

```
# 현재 project에서 사용할 User 모델을 return하는 함수
from django.contrib.auth import get_user_model
from django.contrib.auth import login as auth_login, logout as auth_logout
from django.shortcuts import get_object_or_404, redirect, render
from django.views.decorators.http import require_GET,
require_http_methods,require_POST

# 회원가입용 Form과 인증(로그인)용 Form
from .forms import CustomAuthenticationForm, CustomUserCreationForm

User = get_user_model()

@require_http_methods(['GET', 'POST'])
def signup(request):
    if request.user.is_authenticated:
        return redirect('/')

    if request.method == 'POST':
        form = CustomUserCreationForm(request.POST)
        if form.is_valid():
            user = form.save()
            auth_login(request, user)
            return redirect('/')
    else:
        form = CustomUserCreationForm()
    return render(request, 'accounts/signup.html', {
        'form': form,
```

```
    })

@require_http_methods(['GET', 'POST'])
def login(request):
    if request.user.is_authenticated:
        return redirect('/')

    if request.method == 'POST':
        form = CustomAuthenticationForm(request, request.POST)
        if form.is_valid():
            auth_login(request, form.get_user())
            return redirect('/')
    else:
        form = CustomAuthenticationForm()
    return render(request, 'accounts/login.html', {
        'form': form,
    })


def logout(request):
    auth_logout(request)
    return redirect('/')
```

accounts - templates - accounts - signup.html

```
{% extends 'base.html' %}
{% load bootstrap4 %}

{% block body %}
<form method='POST'>
    {% csrf_token %}
    {% bootstrap_form form %}
    {% buttons %}
    <button class="btn btn-primary">Submit</button>
    {% endbuttons %}
</form>
{% endblock %}
```

accounts - templates - accounts - login.html

```
{% extends 'base.html' %}
{% load bootstrap4 %}

{% block body %}
<form method='POST'>
    {% csrf_token %}
    {% bootstrap_form form %}
    {% buttons %}
    <button class="btn btn-primary">Submit</button>
    {% endbuttons %}
</form>
{% endblock %}
```

postings - urls.py

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

app_name = 'postings'

# insta/
urlpatterns = [
    path('', views.posting_list, name='posing_list'),
    path('<int:posting_id>/', views.posting_detail, name='posting_detail'),
    path('create/', views.create_posting, name='create_posting'),
    path('<int:posting_id>/update/', views.update_posting,
name='update_posting'),
    path('<int:posting_id>/delete/', views.delete_posting,
name='delete_posting'),
]
```

postings - views.py

```
from django.contrib.auth.decorators import login_required
from django.shortcuts import get_object_or_404, redirect, render
from django.views.decorators.http import require_GET, require_http_methods,
require_POST

from .forms import PostingForm, ImageForm
from .models import Posting, Image


@require_GET
def posting_detail(request, posting_id):
    posting = get_object_or_404(Posting, id=posting_id)
    return render(request, 'postings/posting_detail.html', {
        'posting': posting,
    })

@require_GET
def posting_list(request):
```

```
    postings = Posting.objects.all()
    return render(request, 'postings/posting_list.html', {
        'postings': postings,
    })

@login_required
@require_http_methods(['GET', 'POST'])
def create_posting(request):
    images = request.FILES.getlist('file')
    if request.method == 'POST':
        posting_form = PostingForm(request.POST)
        if posting_form.is_valid() and len(images) <= 5:
            posting = posting_form.save(commit=False)
            posting.author = request.user
            posting.save()
            for image in images:
                request.FILES['file'] = image
                image_form = ImageForm(files=request.FILES)  # Form류는 request로 시작하는 객체여야만 사용 가능.
                if image_form.is_valid():
                    image = image_form.save(commit=False)
                    image.posting = posting
                    image.save()
            return redirect(posting)

    else:
        posting_form = PostingForm()
        image_form = ImageForm()
    return render(request, 'postings/posting_form.html', {
        'posting_form': posting_form,
        'image_form': image_form,
    })

@login_required
@require_http_methods(['GET', 'POST'])
def update_posting(request, posting_id):
    posting = get_object_or_404(Posting, id=posting_id)
    if request.method == 'POST':
        form = PostingForm(request.POST, instance=posting)
        if form.is_valid():
            posting = form.save()
            return redirect(posting)
    else:
        form = PostingForm(instance=posting)
    return render(request, 'postings/posting_form.html', {
        'form': form,
    })

@login_required
@require_POST
def delete_posting(request, posting_id):
    posting = get_object_or_404(Posting, id=posting_id)
    posting.delete()
    return redirect('postings:posting_list')
```

postings - models.py

```
from django.contrib.auth import get_user_model
from django.db import models
from django.urls import reverse
from django_extensions.db.models import TimeStampedModel
from imagekit.models import ProcessedImageField
from imagekit.processors import ResizeToFit

User = get_user_model()

class HashTag(TimeStampedModel):  # TimeStampedModel은 created와 modified가 저절로
생김
    content = models.CharField(max_length=20, unique=True)

class Posting(TimeStampedModel):
    like_users = models.ManyToManyField(User, related_name='like_posts',
blank=True)
    author = models.ForeignKey(User, on_delete=models.CASCADE,
related_name='postings')
    hashtags = models.ManyToManyField(HashTag, blank=True,
related_name='postings')
    content = models.CharField(max_length=140)

    class Meta:
        ordering = ('-created',)

    def get_absolute_url(self):
        return reverse("postings:posting_detail", kwargs={"posting_id":
self.pk})


class Image(models.Model):
    posting = models.ForeignKey(Posting, on_delete=models.CASCADE,
related_name='images')
    file = ProcessedImageField(
        processors=[ResizeToFit(600, 600)],
        upload_to='postings/images',
        format='JPEG',
        options={'quality': 90},
    )

# p.image_set.all()  => related_name이 default값
# p.images.all()  => related_name='images'

class Comment(TimeStampedModel):
    author = models.ForeignKey(User, on_delete=models.CASCADE,
related_name='comments')
    posting = models.ForeignKey(Posting, on_delete=models.CASCADE,
related_name='comments')
    content = models.CharField(max_length=140)
```

postings - forms.py

```
from django import forms
from .models import Posting, Comment, Image

class PostingForm(forms.ModelForm):
    class Meta:
        model = Posting
        fields = ('content',)

class ImageForm(forms.ModelForm):
    class Meta:
        model = Image
        fields = ('file',)
        widgets = {
            'file': forms.FileInput(attrs={'multiple':True})
        }


class CommentForm(forms.ModelForm):
    class Meta:
        model = Comment
        fields = ('content',)
```

postings - templates - postings - posting_form.html

```
{% extends 'base.html' %}
{% load bootstrap4 %}

{% block body %}
<form method="POST" enctype="multipart/form-data">
    {% csrf_token %}
    {% bootstrap_form posting_form %}
    {% bootstrap_form image_form show_label=False %}
    {% buttons %}
    <button class="btn btn-primary">Submit</button>
    {% endbuttons %}
</form>

{% endblock %}
```