개인 블로그 만들기 (level1 ~level3)

구분	Python
과목	

개발순서 및 흐름의 이해를 돕고자 제작했다.

1. 기초 설정 (실습1 _ Django 셋팅)

가상환경을 켠 후에 패키지 설치까지 하자.

```
$ python -m venv venv
$ source ./venv/Scripts/activate
```

```
$ pip install django  # Django 웹 프레임워크
$ pip install djangorestframework  # Django REST Framework - REST API 구축

# 아래 부분도 미리 설치
$ pip install requests # 외부 API 호출
$ pip install django-cors-headers # CORS 미들웨어 (다른 도메인에서 API 요청 허용
$ pip install dj-rest-auth dj-rest-auth # (로그인/로그아웃/비번 재설정 API)
$ pip install django-allauth # django-allauth (회원가입, 이메일 확인 등 계정 기능)
$ pip install 'dj-rest-auth[with-social]' # 소셜 로그인 지원

# 프로젝트 그리고 앱 생성
$ django-admin startproject blog_api_service .
$ python manage.py startapp posts
```

일단, `settings.py` 부터 건드린다.

```
# settings.py

INSTALLED_APPS = [
    'posts',
    'rest_framework',              # DRF
    'corsheaders',                 # CORS 추가
    'rest_framework.authtoken',      # 토큰 인증
    'dj_rest_auth',                # 로그인, 로그아웃 등
    'dj_rest_auth.registration',     # 회원가입
    'django.contrib.sites',          # 항상 allauth 보다 위에작성
    'allauth',                     # allauth
    'allauth.account',             # 계정 관리
    'allauth.socialaccount',         # 소셜 로그인

     ...
]

# dj-rest-auth앱 사용시 이메일 인증, 로그인 시에
# "어떤 사이트"에서 동작 중인지를 알아야 하기 때문에
# 반드시 SITE_ID 설정이 필요
SITE_ID = 1

REST_FRAMEWORK = {
    # Authentication
    'DEFAULT_AUTHENTICATION_CLASSES': [
        'rest_framework.authentication.TokenAuthentication',
    ],
    # permission
    'DEFAULT_PERMISSION_CLASSES': [
        'rest_framework.permissions.AllowAny',
    ],
}

MIDDLEWARE = [
    ...
    'corsheaders.middleware.CorsMiddleware', # CORS 추가 (추가위치 주의)
    'django.middleware.common.CommonMiddleware',
    ...
```

```
    'allauth.account.middleware.AccountMiddleware',# 인증 추가

]


CORS_ALLOWED_ORIGINS = [ # CORS 추가
    'http://127.0.0.1:5173',
  'http://localhost:5173',
]


LANGUAGE_CODE = 'ko-kr'

TIME_ZONE = 'Asia/Seoul'
```

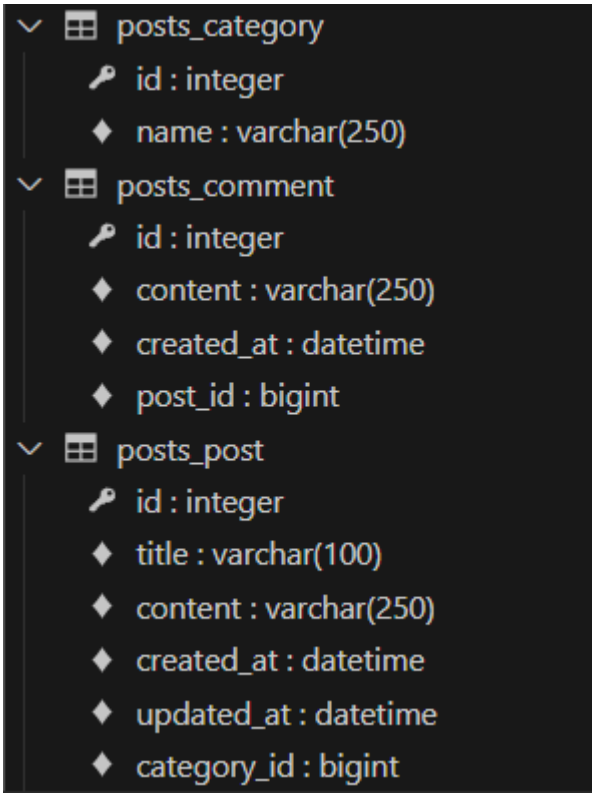
다음, 전역 `urls.py` 를 설정하자.

```
from django.contrib import admin
from django.urls import path, include

urlpatterns = [
    path('admin/', admin.site.urls),
    path('api/v1/', include('posts.urls')),
]
```

우선, `posts` 앱부터 작업하자.

`posts` 앱의 `models.py` 그리고 `admin.py` 셋팅부터 하자.



Category	
PK	PRIMARY KEY
name	Text(250)

Comment	
PK	PRIMARY KEY
post	Foreign Key
content	Text(250)
created_at	DateTime

Post	
PK	PRIMARY KEY
category	Foreign Key
title	Text(100)
content	Text(250)
created_at	DateTime
updated_at	DateTime

먼저,

models.py

```
# posts/models.py

from django.db import models
```

```
class Category(models.Model):
    name = models.CharField(max_length=250)


class Post(models.Model):
    category = models.ForeignKey(Category, on_delete=models.CASCADE)
    title = models.CharField(max_length=100)
    content = models.CharField(max_length=250)
    created_at = models.DateTimeField(auto_now_add=True)
    updated_at = models.DateTimeField(auto_now=True)


class Comment(models.Model):
    post = models.ForeignKey(Post, on_delete=models.CASCADE)
    content = models.CharField(max_length=250)
    created_at = models.DateTimeField(auto_now_add=True)
```

`posts/admin.py` 는 다음과 같다.

```
from django.contrib import admin
from .models import Category, Post, Comment

admin.site.register(Category)
admin.site.register(Post)
admin.site.register(Comment)
```

`posts/urls.py` 는 다음과 같다.

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


urlpatterns = []
```

이제, 마이그레이션 생성 및 마이그레이트, 관리자 생성하자.

```
$ python manage.py makemigrations
$ python manage.py migrate
$ python manage.py createsuperuser
```

반드시 관리자 계정으로 들어가서 게시글 1개 이상 등록해 놓자.

2. 기초 설정 (실습2 _ Vue.js 셋팅)

바탕화면에 파일을 새로 만들고 vscode로 open 후에

vue project 초기 셋팅을 하자.

```
$ npm create vue@latest

Project name(target directory):
|  blog-api-front-pjt

// 라우터 사용. Yes 선택
√ Router (SPA developement) / Yes

// Pinia 사용. Yes 선택
√ Pinia (state management) / Yes

$ cd blog-api-front-pjt/

$ npm install

$ npm i axios

$ npm run dev  서버 켜지는지 확인 해 보자.
```

사용하지 않는 컴포넌트 그리고 assets들을 모두 지운다.

- components / assets / stores / views 폴더의 모든 파일 삭제

- App.vue 에는 삭제한 컴포넌트 관련 코드 지우기

```
<script setup>
import { RouterLink, RouterView } from "vue-router";
</script>

<template>

</template>

<style scoped>

</style>
```

- main.js 의 최상단에 작성되어 있는 main.css import 코드 삭제
- router/index.js에 작성된 경로 및 import문 지운다.

```
import { createRouter, createWebHistory } from 'vue-router'

const router = createRouter({
  history: createWebHistory(import.meta.env.BASE_URL),
  routes: [

  ],
})

export default router
```

요구사항 대로 MainView 컴포넌트 작성하자.

경로 설정하자.

```
import { createRouter, createWebHistory } from 'vue-router'
import MainView from '@/views/MainView.vue'

const router = createRouter({
  history: createWebHistory(import.meta.env.BASE_URL),
```

```
  routes: [
    {
      path: '/',
      name: 'main',
      component: MainView
    },
  ]
})

export default router
```

Views 폴더에 MainView.vue 파일생성 후 만들자.

```
<template>
  <div class="main-page">
    <h1>댕댕쭈 블로그</h1>
    <p>Welcome to my place!</p>

  </div>
</template>

<script setup>
</script>

<style scoped>
.main-page {
  display: flex;
  flex-direction: column;
  align-items: center;
  justify-content: center;
  height: 100vh;
  background-color: #dbf3f0;
}

h1 {
  font-size: 2rem;
  margin-bottom: 1rem;
}
```

```
p {
  font-size: 1rem;
  color: #666;
}
</style>
```

App.vue에 라우터 컴포넌트를 화면에 붙이자.

```
<script setup>
import { RouterLink, RouterView } from "vue-router";
</script>

<template>
  <div class="container">
    <nav>
      <RouterLink to="/">Home</RouterLink>
    </nav>
  </div>

  <RouterView />
</template>

<style scoped>
.container {
  max-width: 1200px;
  margin: 0 auto;
  padding: 1rem;
}

nav {
  display: flex;
  justify-content: center;
  background-color: #a5a4a3;
  padding: 1rem 0;
}

nav a {
```

```
  color: #fff;
  text-decoration: none;
  margin: 0 1rem;
  font-weight: bold;
}

nav a:hover {
  text-decoration: underline;
}
</style>
```

3. Vue.js - Django 데이터 주고받기 (실습3)

전체 게시글을 조회하는 것을 해보자. 요구사항은 다음과 같다.

- Django views.py에 함수를 작성해서 모든 post의 pk와 title, category 정보를 넘겨줘야 한다.
- Vue.js 에서 PostListView 컴포넌트를 작성한다.
  - App.vue에 라우터링크 넣고
  - PostListView가 화면에 보이도록 axios 요청을 보낸다

serializers.py 파일 생성

```
from rest_framework import serializers
from .models import Post


class PostListSerializer(serializers.ModelSerializer):

    class Meta:
        model = Post
        fields = ('pk', 'title', 'category')
```

posts 폴더에 urls.py 작성

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

urlpatterns = [
    path('posts/', views.post_list)
]
```

views.py 함수 작성

```
from rest_framework.response import Response
from rest_framework.decorators import api_view
from django.shortcuts import render
from .serializers import PostListSerializer
from .models import Post

# Create your views here.

@api_view(['GET'])
def post_list(request):
    posts = Post.objects.all()
    serialzers = PostListSerializer(posts, many=True)
    return Response(serialzers.data)
```

Vue 에서 PostListView 컴포넌트를 작성하자.

stores 폴더에 posts.js 파일 생성하자.

views 폴더에 PostListView 컴포넌트 생성하자.

```
<template>
  <div>
    <h1>게시글 목록 페이지</h1>
  </div>
</template>
```

```
<script setup>

</script>

<style scoped>

</style>
```

router 폴더 안에 index.js에 경로를 추가하자.

```
import { createRouter, createWebHistory } from 'vue-router'
import MainView from '@/views/MainView.vue'
import PostListView from '@/views/PostListView.vue'

const router = createRouter({
  history: createWebHistory(import.meta.env.BASE_URL),
  routes: [
    {
      path: '/',
      name: 'main',
      component: MainView
    },
    {
      path: '/posts',
      name: 'posts',
      component: PostListView
    },
  ]
})

export default router
```

App.vue에 컴포넌트 붙이자.

```
<script setup>
import { RouterLink, RouterView } from "vue-router";
</script>

<template>
  <div class="container">
    <nav>
      <RouterLink to="/">Home</RouterLink>
      <RouterLink :to="{name: 'posts'}">PostList</RouterLink>
    </nav>
  </div>

  <RouterView />
</template>
```

posts.js 스토어에서 axios 요청 코드를 넣자

```
import axios from 'axios'
import { defineStore } from 'pinia'

export const usePostStore = defineStore('post', () ⇒ {
  const getPostList = function () {
    axios({
      method: 'get',
      url: 'http://127.0.0.1:8000/api/v1/posts/'
    })
    .then(res ⇒ console.log(res))
    .catch(err ⇒ console.log(err))
  }
  return { getPostList }
})
```

PostListView 컴포넌트에 script 코드를 추가하자.

```
<template>
  <div>
    <h1>게시글 목록 페이지</h1>
  </div>
</template>

<script setup>
import { onMounted } from 'vue';
import { usePostStore } from '../stores/posts';
const store = usePostStore()
onMounted(() => {
  store.getPostList()
})
</script>

<style scoped>

</style>
```

CORS 문제를 해결하기 위해서

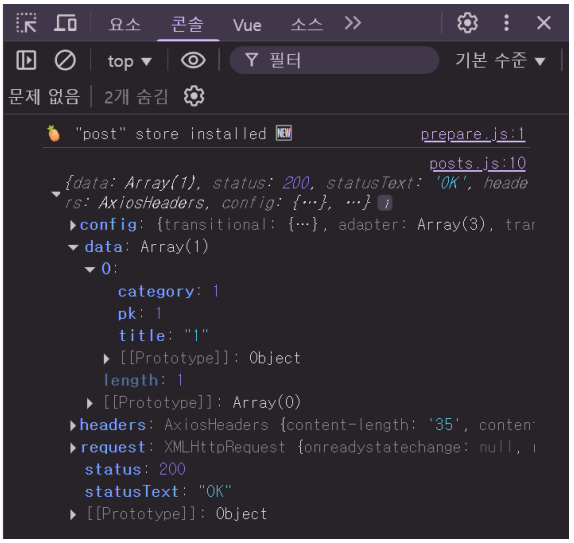
1. $ pip install django-cors-headers
2. settings.py의 INSTALLED_APPS , MIDDLEWARE 그리고
   CORS_ALLOWED_ORIGINS 관련 코드 넣기

가 있는데 우리는 실습 level1에서 모두 수행 했다. (주석으로 “ # CORS 추가 “ 라고 모두 달아 놓았다.

http://localhost:5173/posts 들어가보자.

Home

게시글 목록 페이지



개발자 도구 들어가서 admin 사이트 통해서 작성했던 데이터가 보이는지 확인하자.

4. 단일게시글 생성 및 조회 (실습4)

요구사항은 다음과 같다.

- category, post, comment 등 (단일 게시글)을 생성하는 기능 구현한다.
  - post 생성시 category정보는 사용자가 직접 작성한 category의 pk정보를 이용한다.
  - comment 생성시 post정보는 post의 pk정보를 베이스로 생성한다.
- category, post, comment 정보조회는 누구나 조회 할수 있다 (인증 권한 기능구현 필요 없다는 뜻)
  - category 전체 목록 조회
  - post 전체 목록 조회 (pk category title 포함)
  - post 상세 정보 조회 (comment 정보 포함, comment 생성하느 기능도 있어야 함)
- category 생성 페이지로 이동하는 링크는 navbar에 표기하라 그리고 post 생성하는 링크는 PostListView에서 게시글 작성 버튼으로한다.

결론은 다음과 같다

온라인 실습에 있는 이미지를 꼼꼼하게 보고 내가 무엇을 해야 하는지 정확히 인지를 하는 것이 우선이다.

- category 컴포넌트 만들고, 카테고리 생성 기능이 들어가야 한다.
- 카테고리가 생성이 되면 메인페이지에서 보여야 한다.
- postsListView 컴포넌트에서는 게시글 목록이 보이고 게시글생성 링크가 있어야 한다.
- 게시글 생성 페이지에서는 게시글생성이 되야 한다.
- 게시글을 클릭하면 해당 게시글의 상세정보를 확인할 수 있어야 한다. (detail)
- 상세정보 안에는 댓글도 볼 수 있어야 하고 댓글을 생성 할 수도 있어야 한다.
- 네비바도 만들어야 한다.

우리는 위에서 그동한 작업했던 순서는 다음과 같다.

1. **Django**: API 만들기
   - `models.py`: 모델 정의
   - `serializers.py` 작성
   - `urls.py` 경로 설정
   - `views.py`에 API 함수 작성
   - 서버 실행해서 브라우저로 응답이 오는지 확인 (Postman 통해서 확인)
2. **Vue**
   - `PostListView.vue` 파일 생성
   - `router/index.js`에 라우터에 등록
   - `App.vue`에 링크 등록
   - `stores/posts.js`에 axios 요청 함수 작성
   - `PostListView.vue`에서 `onMounted()`로 데이터 요청 (컴포넌트에서 store사용)

   순으로 실습을 했다. 요약을 하면 다음과 같다.

   1 Django API 만들고 테스트 API가 문제 없는지 검증한다.
   2 Vue 라우터와 컴포넌트 구성화면 전환 구조 만든다.

3 axios store 정의, API 요청 로직 분리하고
4 컴포넌트에서 store 호출해서 실제 데이터 조회 및 연결한다.

위 순서로 개발하면 되겠다.

실습 4, 5번의 정답을 공유하겠다.

먼저 내가 어떤기능을 먼저 구현할 것이지 명확하게 목표를 설정 한 다음

데이터의 흐름 그리고 코드작성 순서를 생각해보면서 하나씩 도전 해보자.