Grzegorz Ostrek 4.12.2024r.

# Web Application Programming

## Laboratory 10

1. Create a Django project environment (here instructions given for VSCode, use PyCharm if preferred)

$ C:\Python\Python311\python.exe -m venv venv

Run terminal (PS or CMD)

In case PS

PS C:\paw_restframework_lab> Set-ExecutionPolicy -ExecutionPolicy Bypass -Scope Process -Force
PS C:\paw_restframework_lab> .\venv\Scripts\Activate.ps1

If Set-ExecutionPolicy failed then use CMD terminal.

2. Create file requirements.txt in project directory (on the manage.py file directory level)

```
Django==4.2

djangorestframework
```

```
run
(venv) PS C:\paw_restframework_lab> pip install -r requirements.txt

Or use PyCharm automatic resolution

3. Create Django project
```

(venv) PS C:\paw_restframework_lab> django-admin.exe startproject lab13

(venv) PS C:\paw_restframework_lab> cd .\lab13\

(venv) PS C:\ paw_restframework_lab\lab13> python manage.py runserver

4. Add your application

```
django-admin startapp myapp
```

add app in settings.py

Modify settings.py

```
INSTALLED_APPS = [
    ...
    'rest_framework',
    'myapp',
]
```

5.Add model

```
class Post(models.Model):
    title = models.CharField(max_length=126)
    content = models.TextField()
    pub_date = models.DateTimeField(auto_now_add=True)
```

6 Migrate models

(venv) PS C: \paw_restframework_lab\lab13> python .\manage.py makemigrations

(venv) PS C: \paw_restframework_lab\lab13> python  .\manage.py migrate

In admin.py add

```
from django.contrib import admin
from .models import Post
# Register your models here.

admin.site.register(Post)
```

Create a superuser then add some data in Post table, from the admin panel.

7 Create lab13/api.py (in settings.py folder), in case you have many apps the api.py

```
from django.urls import path
from django.urls.conf import include
urlpatterns = [
    path('myapp/', include('myapp.urls')),
]
```

8 Add in lab13/urls.py

```
path('api/v1/', include('lab13.api')),
```

9. Create file myapp/serializers.py.  Serializer is responsible for converting data between database and HTTP response format. (see https://opensource.com/article/20/11/django-rest-framework-serializers) Data must be validated before saving to database, like in forms.

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

class PostSerializer(ModelSerializer):
    class Meta:
        model = Post
        fields = ['title', 'content']
```

10 Create file myapp/viewsets.py

```
from rest_framework.viewsets import ModelViewSet
```

```
from rest_framework.permissions import IsAuthenticated
from .serializers import PostSerializer
from .models import Post

class PostViewSet(ModelViewSet):
    queryset = Post.objects.all().order_by('-pub_date')
    serializer_class = PostSerializer
    permission_classes = [IsAuthenticated]
```

11 Create myapp/urls.py

```
from rest_framework.routers import DefaultRouter
from django.urls.conf import include
from django.urls import re_path
from .viewsets import PostViewSet

router = DefaultRouter()
router.register(
    r'posts',
    PostViewSet,
    basename="posts",
)
urlpatterns = [
    re_path(r'', include(router.urls)),
]
# or urlpatterns = router.urls
```

(see https://www.django-rest-framework.org/api-guide/routers/#defaultrouter )

12.

Check URLs :
http://127.0.0.1:8000/api/v1/myapp/

http://127.0.0.1:8000/api/v1/myapp/posts/

Log in as admin then log out - and check again.

Try Post,

You may modify settings.py

```
REST_FRAMEWORK = {
'DEFAULT_PERMISSION_CLASSES': [
   'rest_framework.permissions.AllowAny',
#default 'rest_framework.permissions.IsAuthenticated',
]
```

13

The actions provided by the `ModelViewSet` class are `.list()`, `.retrieve()`, `.create()`, `.update()`, `.partial_update()`, and `.destroy()`.

https://www.django-rest-framework.org/api-guide/viewsets/

It means that ie. you got URL served:
http://127.0.0.1:8000/api/v1/myapp/posts/1/

Try Patch, Delete

14

See Options button – this may be useful in frontend framework.

15 Modify viewsets.py

```
from rest_framework import mixins
from rest_framework.viewsets import GenericViewSet

class PostViewSet(mixins.ListModelMixin, GenericViewSet):
    queryset = Post.objects.all().order_by('-pub_date')
    serializer_class = PostSerializer
    permission_classes = [IsAuthenticated]
```

Now only selected actions are allowed by so called mixins.

The view classes can be imported from rest_framework.generics.

- CreateAPIView Provides a post method handler. create-only endpoints.

- ListAPIView Provides a get method handler. read-only endpoints to represent a collection of model instances.

- RetrieveAPIView Provides a get method handler. read-only endpoints to represent a single model instance.

- DestroyAPIView Provides a delete method handler.

- UpdateAPIView Provides put and patch method handlers.

- ListCreateAPIView Provides get and post method handlers.b read-write endpoints to represent a collection of model instances

- RetrieveUpdateAPIView Provides get, put and patch method handlers.

- RetrieveDestroyAPIView Provides get and delete method handlers.

- RetrieveUpdateDestroyAPIView Provides get, put, patch and delete method handlers.

16 OpenAPI swagger

```
Install spectacular – add to requirements drf-spectacular and install it
pip install drf-spectacular

Modify settings.py
INSTALLED_APPS = [
    # ALL YOUR APPS
    'drf_spectacular',
]
```

register our spectacular AutoSchema with DRF.

```
REST_FRAMEWORK = {
    # YOUR SETTINGS
    'DEFAULT_SCHEMA_CLASS': 'drf_spectacular.openapi.AutoSchema',
}
```

in myapp/urls.py

```
from drf_spectacular.views import SpectacularAPIView, SpectacularSwaggerView
```

```
urlpatterns = [
    …
    path("schema/", SpectacularAPIView.as_view(), name="schema"),
     path("docs/", SpectacularSwaggerView.as_view(), name="swagger-ui"),
]
```

```
API should not be exposed in public
- just for test !!! if you are not logged in as admin
SPECTACULAR_DEFAULTS = {
'SERVE_PERMISSIONS': ['rest_framework.permissions.AllowAny'],
}
```

Check URL:

http://127.0.0.1:8000/api/v1/myapp/docs/

17. Own Work : Add Comment model. Create endpoint which will return the 'title' and the longest comment for a post. Connect post with a user.
Find documentation how to work with foreign key referenced sets, prefetch and prefetch_related, annotations. You may add a serializer field – fixed or calculated.

18. Docker – try it at home

(If you will build your task with docker, then prepare a github repo for it please)

Install Docker Desktop
https://docs.docker.com/desktop/install/windows-install/

Create Dockerfile in Django project (at manage.py level):

```
#syntax=docker/dockerfile:1
FROM python:3
ENV PYTHONDONTWRITEBYTECODE=1
ENV PYTHONUNBUFFERED=1
WORKDIR /code
COPY requirements.txt /code/
RUN pip install -r requirements.txt
```

```
RUN apt-get update && apt-get install -y gettext
COPY . .
```

Create compose.yaml, level over Django project

```
  services:
    postgress_db:
      image: postgres
      volumes:
        - ./data/db:/var/lib/postgresql/data
      env_file: .env
      ports:
        - "5432:5432"
      restart: always
    django:
      build: ./lab13
      command: python manage.py runserver 0.0.0.0:8000
      volumes:
        - ./lab13:/code
      ports:
        - "8000:8000"
      env_file: .env
      depends_on
        - postgress_db
      restart: always
```

Create .env file on compose.yaml

```
POSTGRES_NAME=postgres
POSTGRES_USER=dbuser
POSTGRES_PASSWORD=secret
POSTGRES_HOST=postgress_db
POSTGRES_PORT=5432
```

Modify Settings.py

```
import os
...
DATABASES = {     # from .env.
    'default': {
        'ENGINE': 'django.db.backends.postgresql',
        'NAME': os.environ.get('POSTGRES_NAME'),
        'USER': os.environ.get('POSTGRES_USER'),
        'PASSWORD': os.environ.get('POSTGRES_PASSWORD'),
        'HOST': os.environ.get('POSTGRES_HOST'),
        'PORT': os.environ.get('POSTGRES_PORT'),
```

```
    }
}
```

In terminal run docker-compose:

PS C:\paw_restframework_lab> docker-compose.exe build

PS C:\paw_restframework_lab> docker-compose.exe up --build

Check URL http://127.0.0.1:8000/

Try :

docker ps -a # NOTE NAME COLUMN

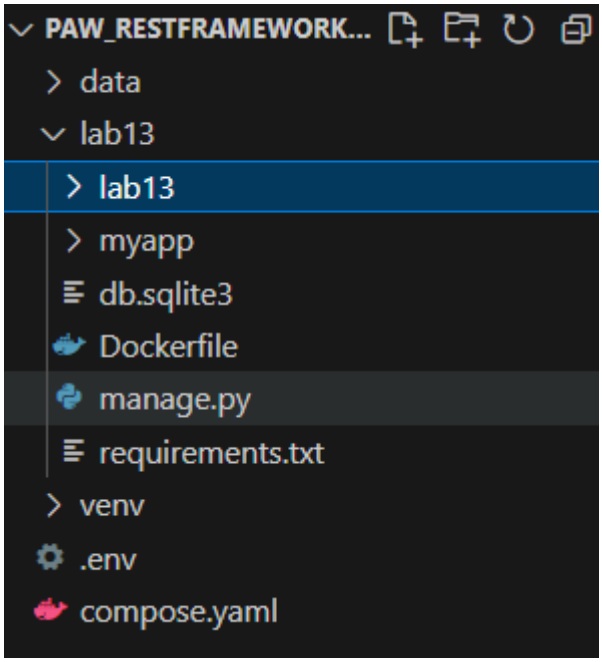
PS C:\paw_restframework_lab> docker-compose.exe up -d

PS C:\paw_restframework_lab> docker exec -it paw_restframework_lab-django-1 python manage.py migrate

docker exec -it paw_restframework_lab-postgress_db-1 bash

docker exec -it paw_restframework_lab-django-1 bash

Files layout:




Requirements.txt

```
Django==4.2
djangorestframework
drf-spectacular
```

Documentation:

https://www.django-rest-framework.org/

https://www.django-rest-framework.org/tutorial/quickstart/

https://www.cookiecutter.io/templates - see Django relevant option

https://drf-spectacular.readthedocs.io/en/stable/index.html