Topic: 7. Class-Based Views Basics & Retrieving All Records

Speaker: Personal | Notebook: API Development using Django Framework

---

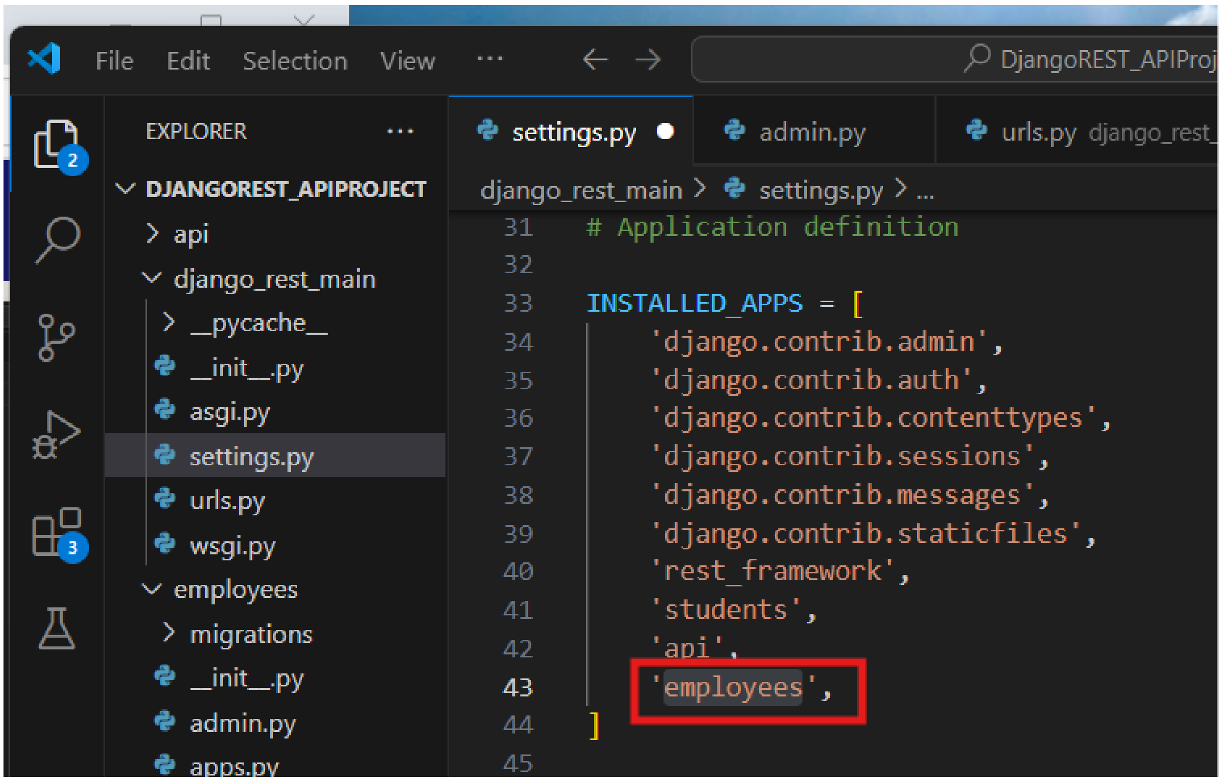
Class-based views follow the principles of object-oriented programming. These views are used for reusability and code efficiency.

For this example, we need to create a new app called EMPLOYEES.

1. In the terminal, create a new app:

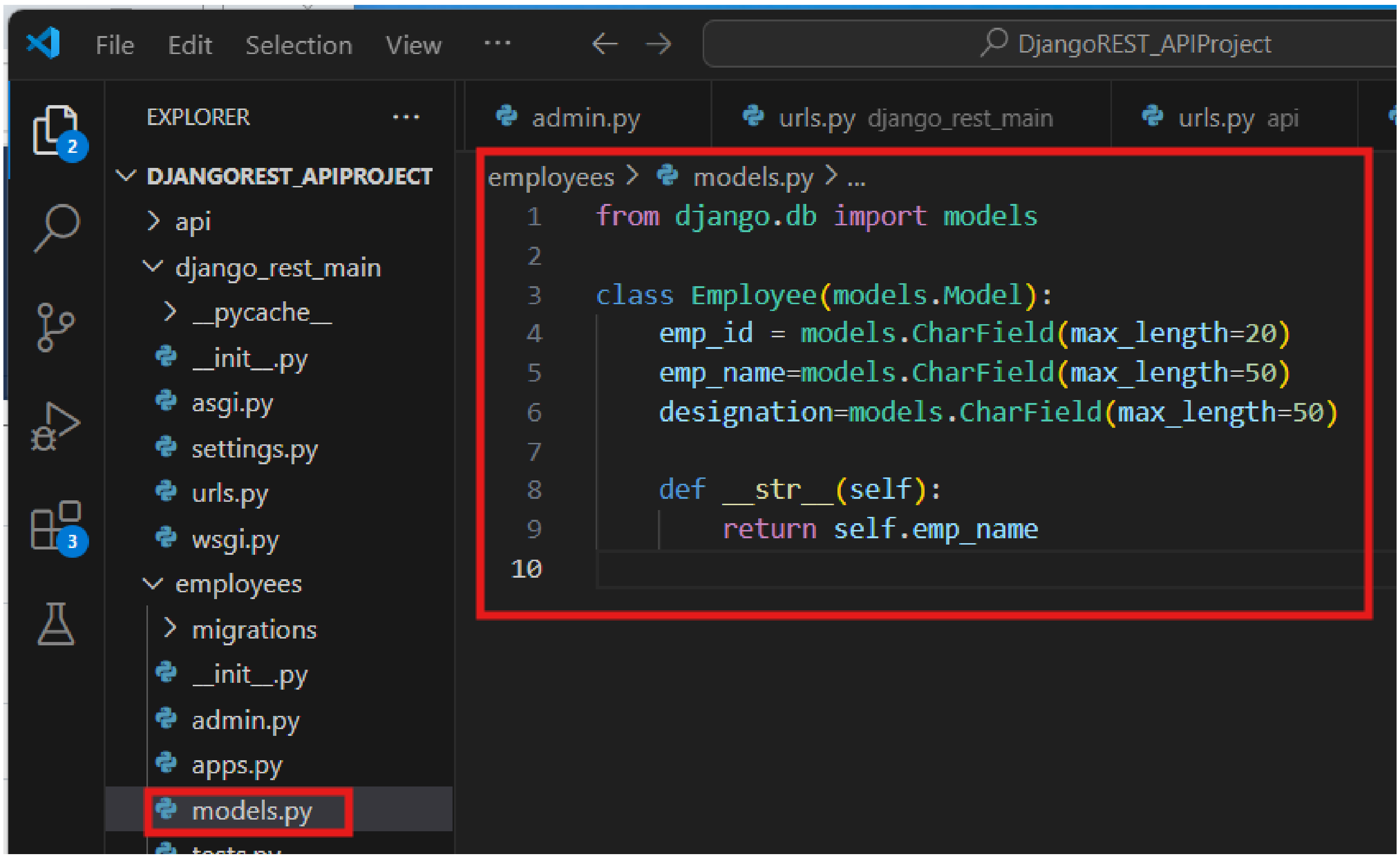
```
$ python manage.py startapp employees
```

2. Register this new app in the SETTINGS.PY



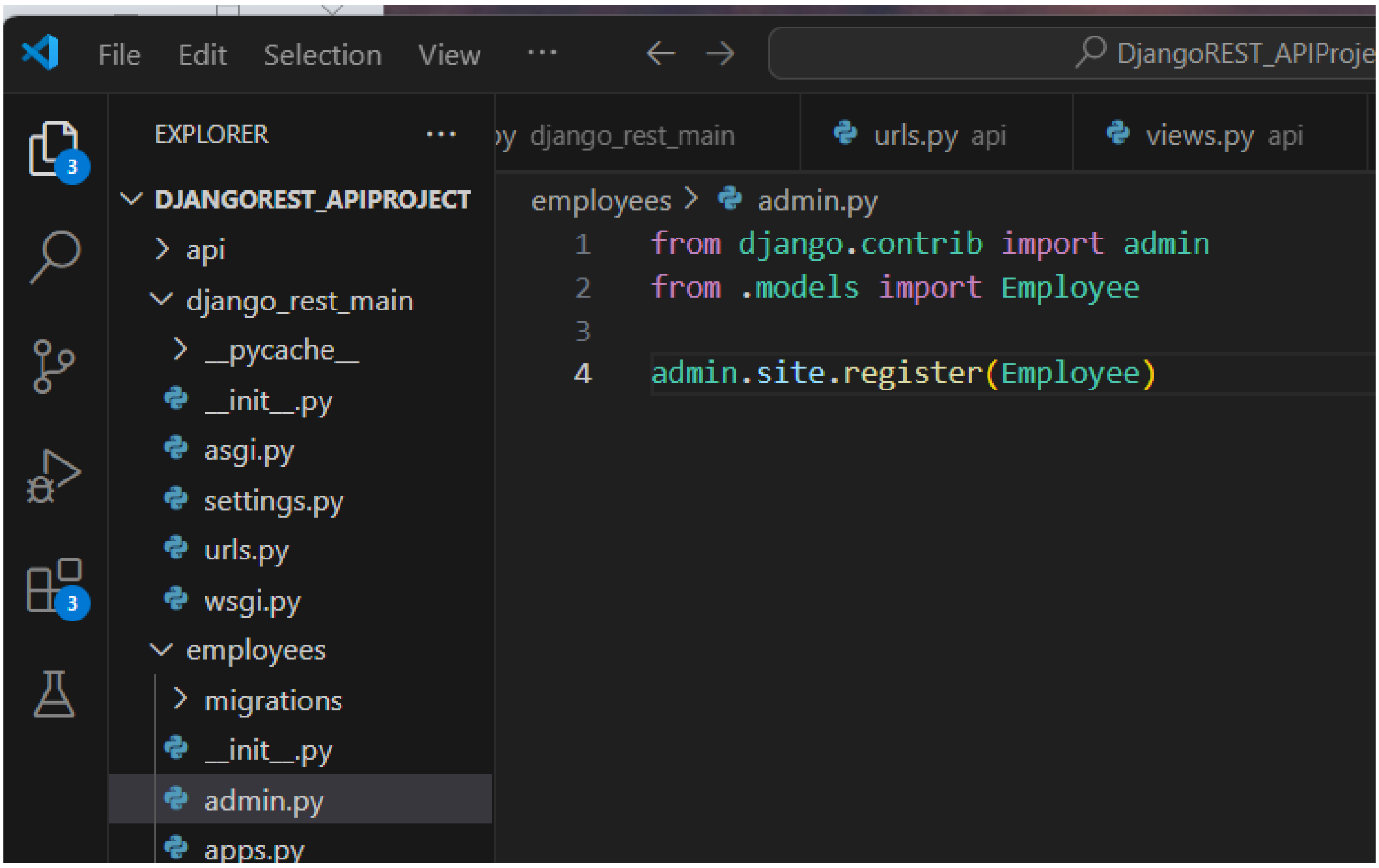
```
# Application definition

INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'rest_framework',
    'students',
    'api',
    'employees',
]
```

3. Create the new model in MODELS.PY



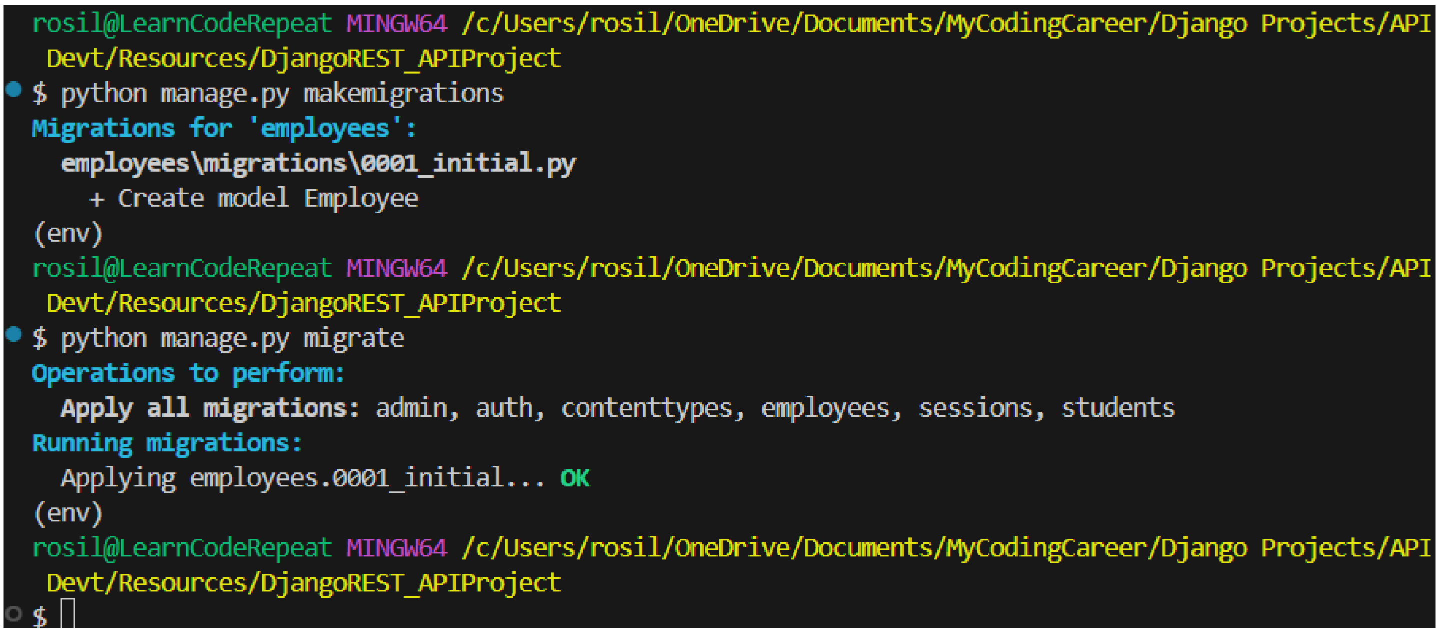
```python
from django.db import models

class Employee(models.Model):
    emp_id = models.CharField(max_length=20)
    emp_name=models.CharField(max_length=50)
    designation=models.CharField(max_length=50)

    def __str__(self):
        return self.emp_name

```

4. Update the ADMIN.PY



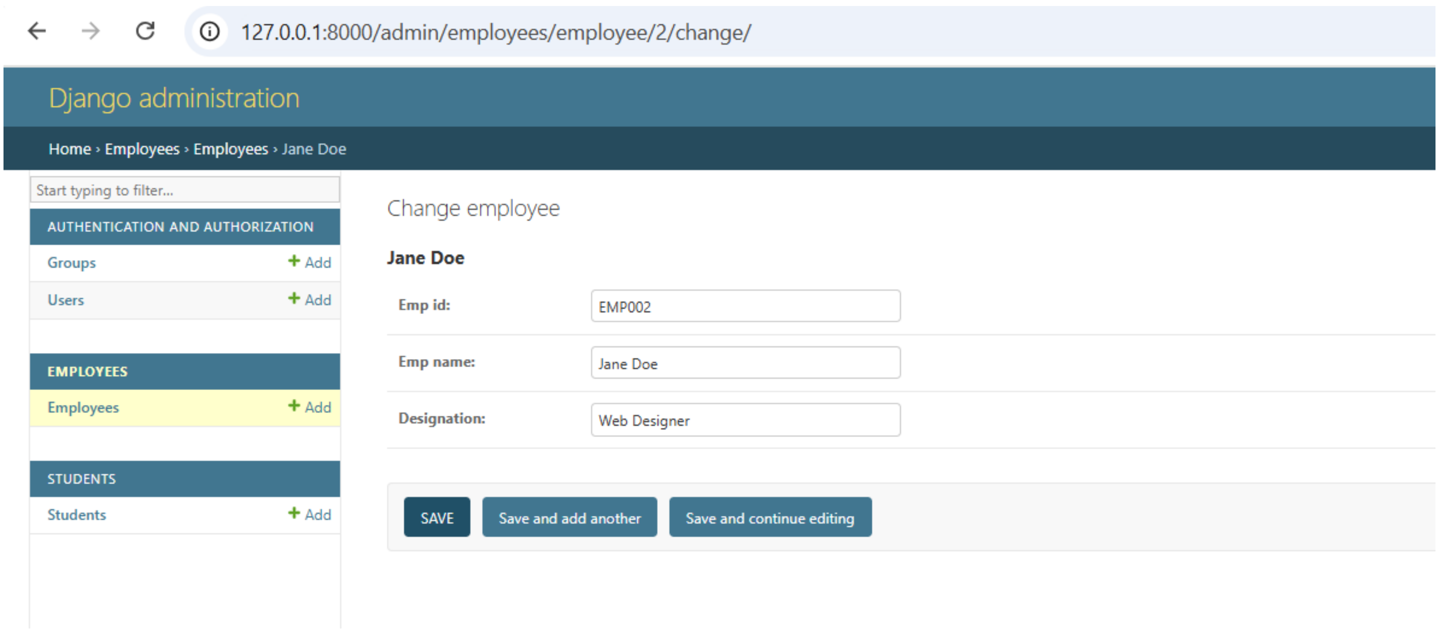
```python
from django.contrib import admin
from .models import Employee

admin.site.register(Employee)
```

5. Make the migrations. Be sure you are in the correct folder to see the migrations happen.



```
rosil@LearnCodeRepeat MINGW64 /c/Users/rosil/OneDrive/Documents/MyCodingCareer/Django Projects/API
 Devt/Resources/DjangoREST_APIProject
$ python manage.py makemigrations
Migrations for 'employees':
  employees\migrations\0001_initial.py
    + Create model Employee
(env)
rosil@LearnCodeRepeat MINGW64 /c/Users/rosil/OneDrive/Documents/MyCodingCareer/Django Projects/API
 Devt/Resources/DjangoREST_APIProject
$ python manage.py migrate
Operations to perform:
  Apply all migrations: admin, auth, contenttypes, employees, sessions, students
Running migrations:
  Applying employees.0001_initial... OK
(env)
rosil@LearnCodeRepeat MINGW64 /c/Users/rosil/OneDrive/Documents/MyCodingCareer/Django Projects/API
 Devt/Resources/DjangoREST_APIProject
$ 
```

6. Check the admin panel and create new records for the Employees model:



127.0.0.1:8000/admin/employees/employee/2/change/

Django administration

Home › Employees › Employees › Jane Doe

Start typing to filter...

AUTHENTICATION AND AUTHORIZATION
Groups + Add
Users + Add

EMPLOYEES
Employees + Add

STUDENTS
Students + Add

Change employee

Jane Doe

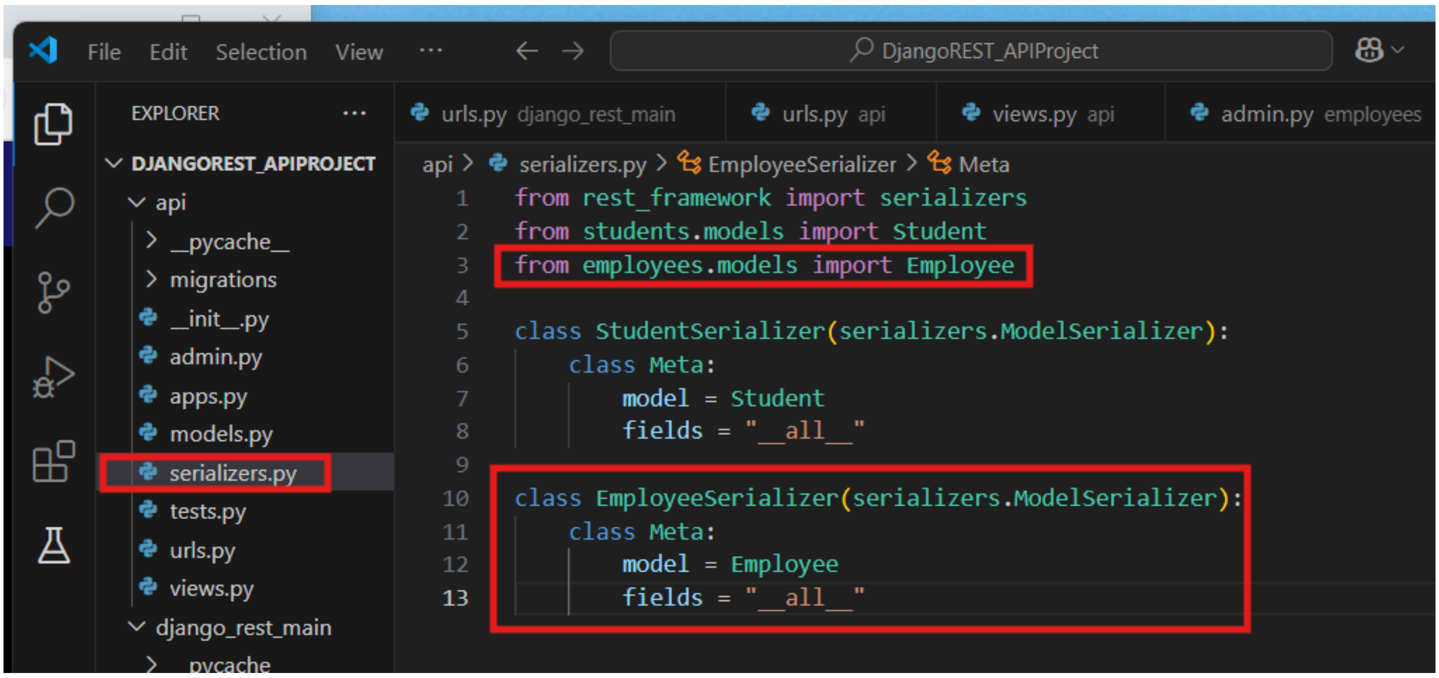
Emp id: EMP002

Emp name: Jane Doe

Designation: Web Designer

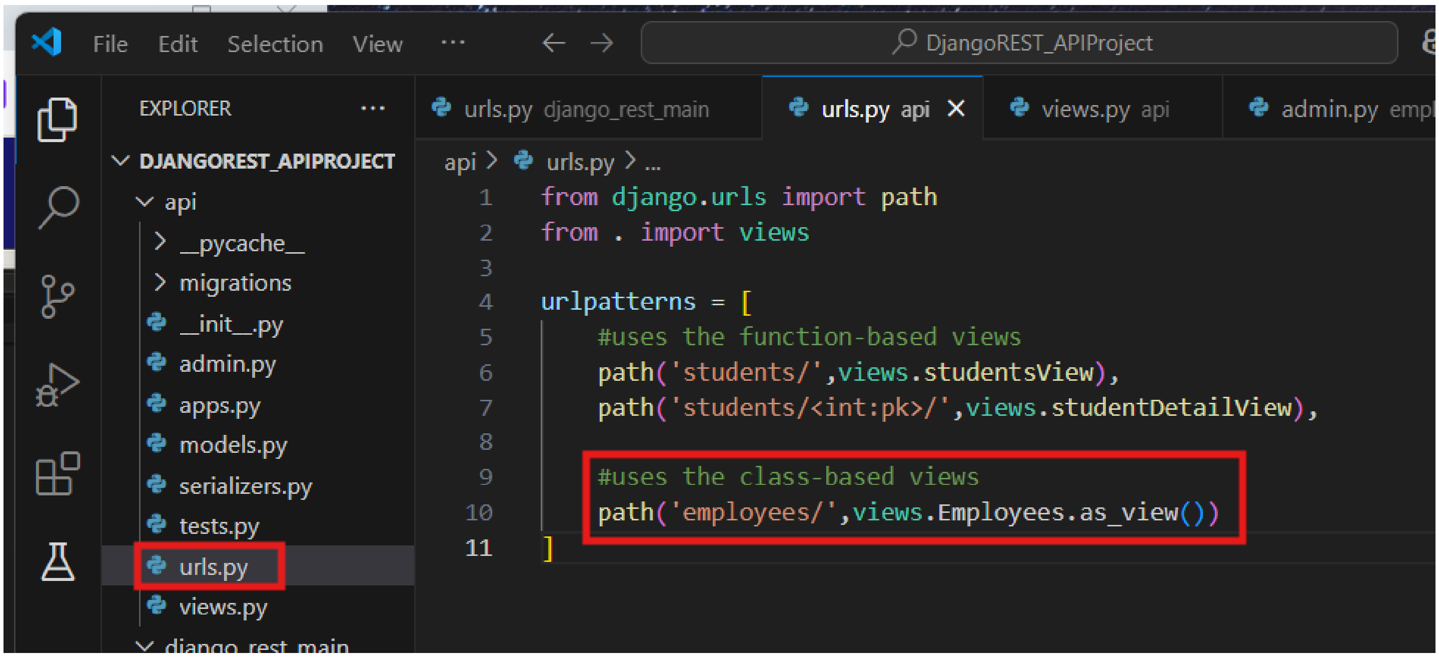
SAVE | Save and add another | Save and continue editing

7. Create a serializer for the Employees model. Go to API\SERIALIZER.PY



```python
from rest_framework import serializers
from students.models import Student
from employees.models import Employee

class StudentSerializer(serializers.ModelSerializer):
    class Meta:
        model = Student
        fields = "__all__"

class EmployeeSerializer(serializers.ModelSerializer):
    class Meta:
        model = Employee
        fields = "__all__"
```

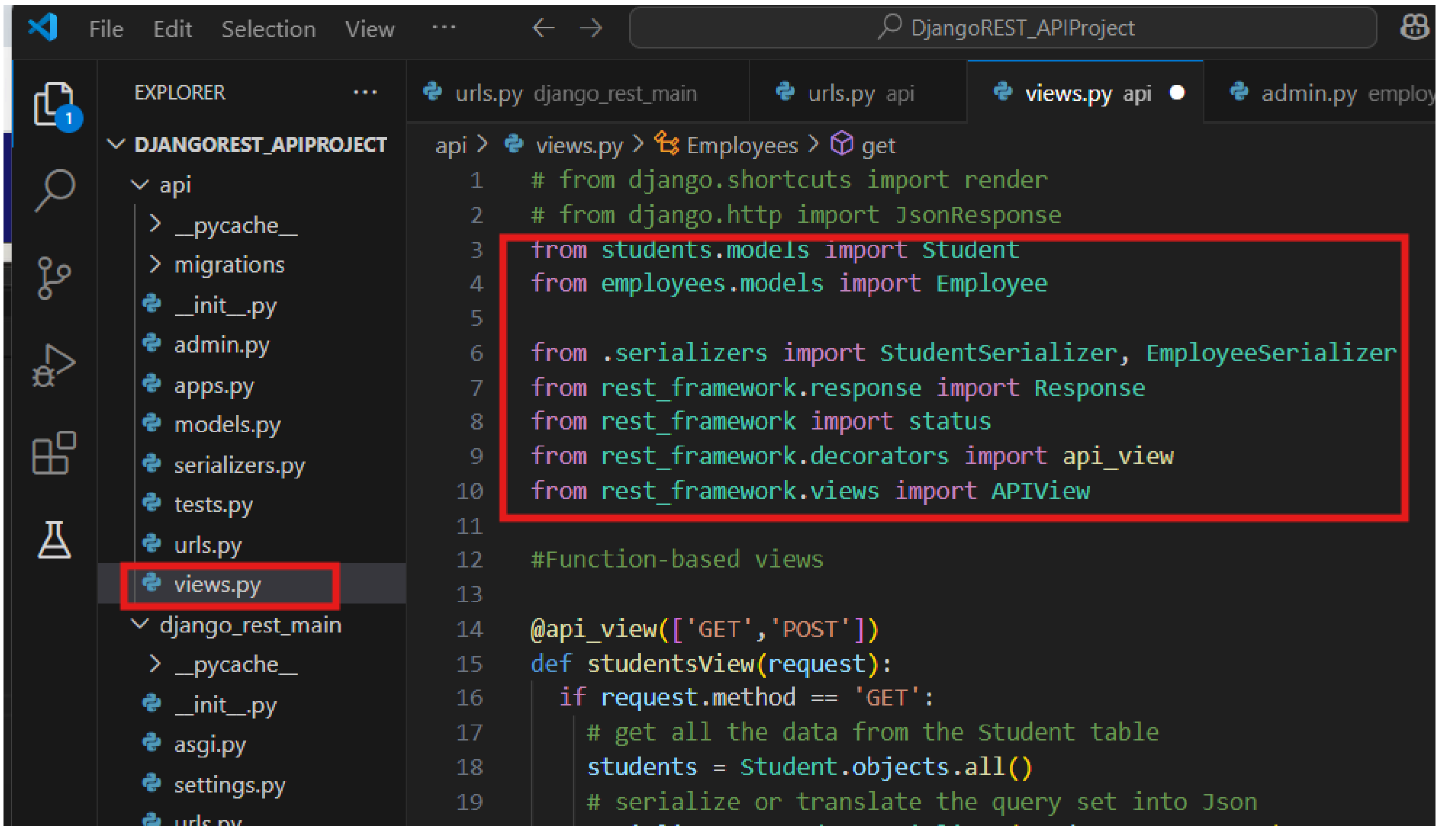
8. To retrieve all the records, go to the API\URLS.PY and update:



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

urlpatterns = [
    #uses the function-based views
    path('students/',views.studentsView),
    path('students/<int:pk>/',views.studentDetailView),

    #uses the class-based views
    path('employees/',views.Employees.as_view())
]
```

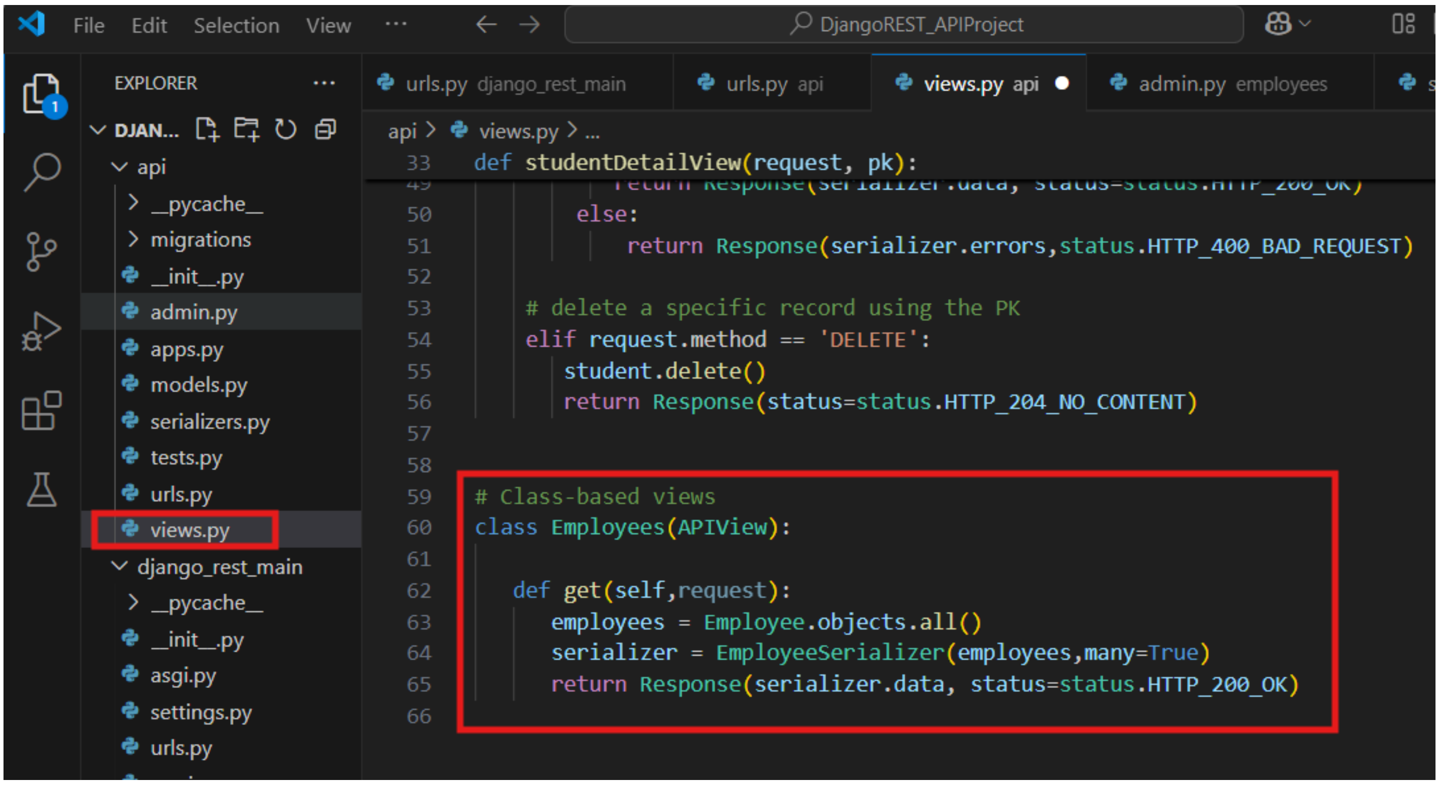
9. Create the class-based views. Go to the API\VIEWS.PY:

Add the necessary libraries:



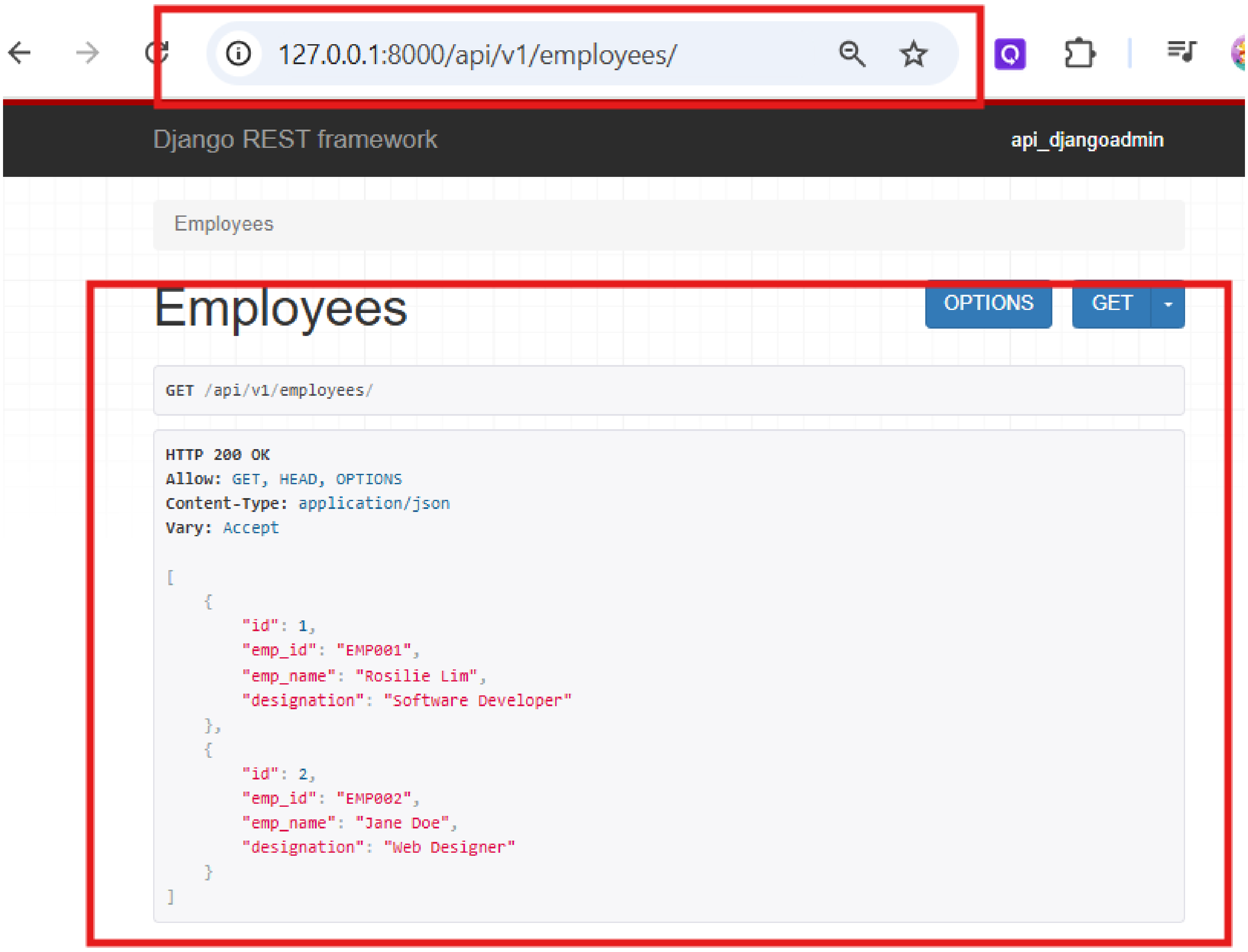
```python
# from django.shortcuts import render
# from django.http import JsonResponse
from students.models import Student
from employees.models import Employee

from .serializers import StudentSerializer, EmployeeSerializer
from rest_framework.response import Response
from rest_framework import status
from rest_framework.decorators import api_view
from rest_framework.views import APIView

#Function-based views

@api_view(['GET','POST'])
def studentsView(request):
    if request.method == 'GET':
        # get all the data from the Student table
        students = Student.objects.all()
        # serialize or translate the query set into Json
```

Create the class Employee and its methods:



```python
def studentDetailView(request, pk):
            return Response(serializer.data, status=status.HTTP_200_OK)
        else:
            return Response(serializer.errors,status.HTTP_400_BAD_REQUEST)

    # delete a specific record using the PK
    elif request.method == 'DELETE':
        student.delete()
        return Response(status=status.HTTP_204_NO_CONTENT)

# Class-based views
class Employees(APIView):

    def get(self,request):
        employees = Employee.objects.all()
        serializer = EmployeeSerializer(employees,many=True)
        return Response(serializer.data, status=status.HTTP_200_OK)

```

10. To test, use the URL `http://127.0.0.1:8000/api/v1/employees/`



```
GET /api/v1/employees/
```

```
HTTP 200 OK
Allow: GET, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept

[
    {
        "id": 1,
        "emp_id": "EMP001",
        "emp_name": "Rosilie Lim",
        "designation": "Software Developer"
    },
    {
        "id": 2,
        "emp_id": "EMP002",
        "emp_name": "Jane Doe",
        "designation": "Web Designer"
    }
]
```

11.