

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.

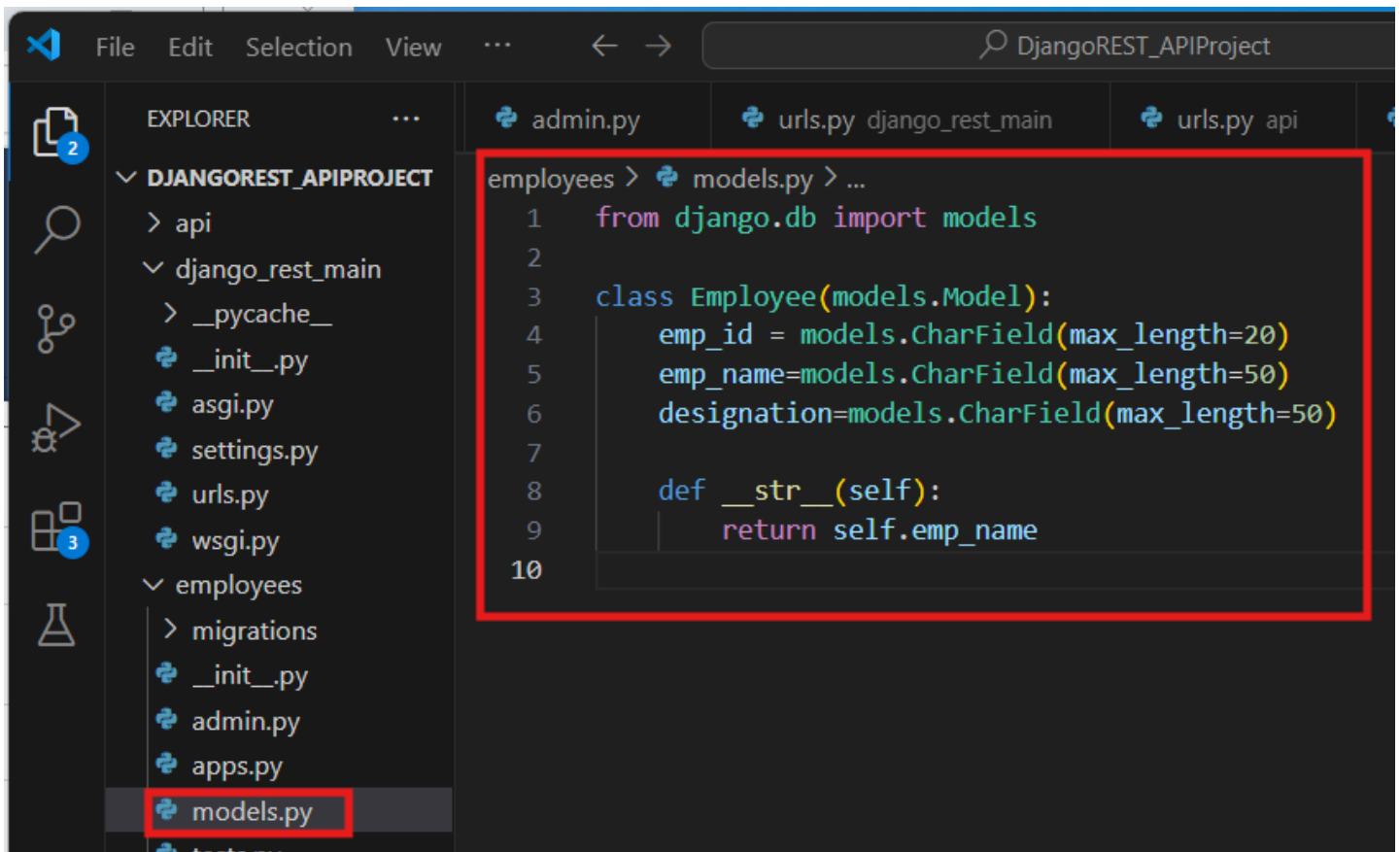
1. In the terminal, create a new app:

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

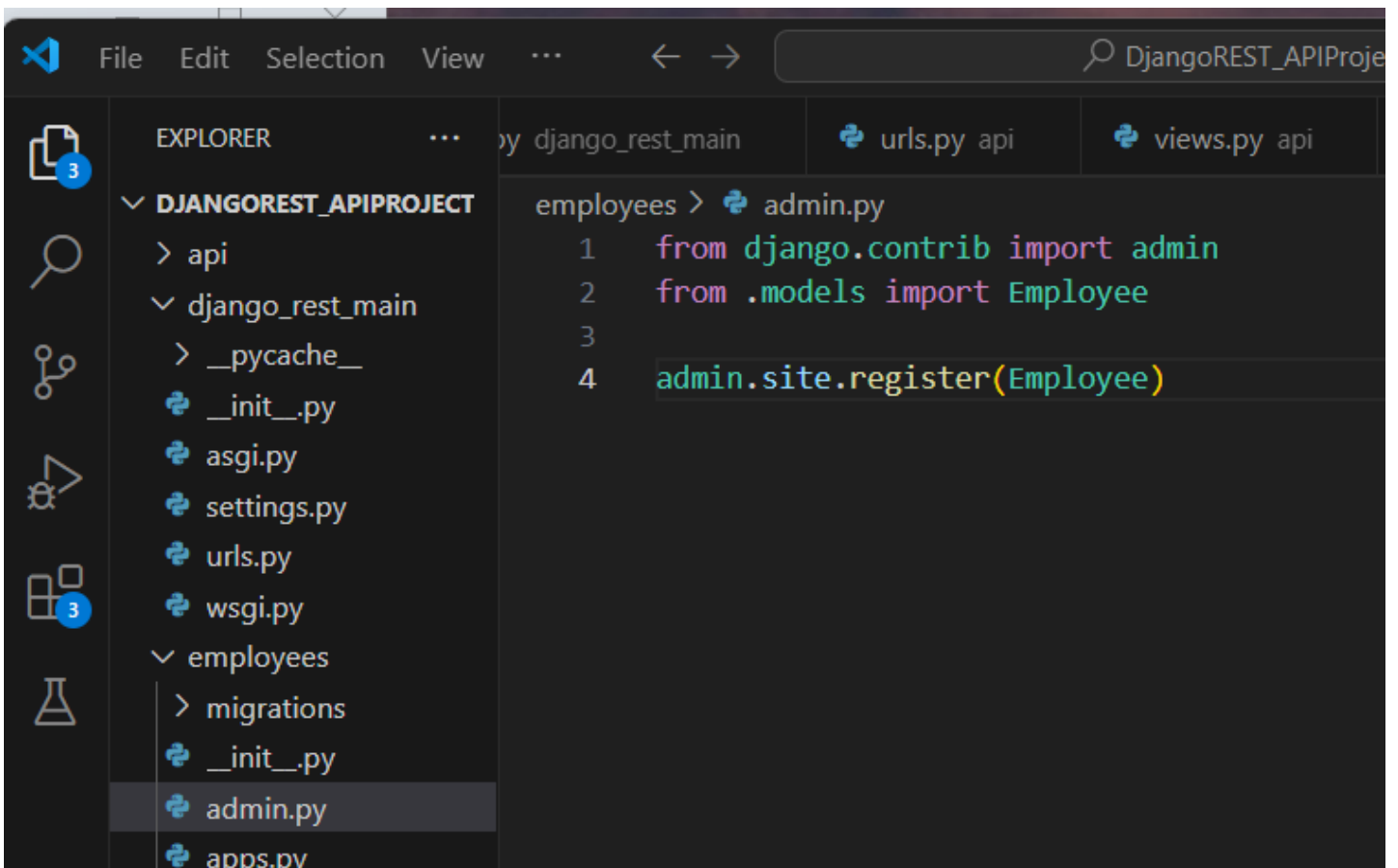
2. Register this new app in the SETTINGS.PY

```
File Edit Selection View ... DjangoREST_APIProj  
EXPLORER  
▼ DJANGOREST_APIPROJECT  
  > api  
  ▼ django_rest_main  
    > __pycache__  
    > __init__.py  
    > asgi.py  
    > settings.py  
    > urls.py  
    > wsgi.py  
  ▼ employees  
    > migrations  
    > __init__.py  
    > admin.py  
    > apps.py  
settings.py  
31 # Application definition  
32  
33 INSTALLED_APPS = [  
34     'django.contrib.admin',  
35     'django.contrib.auth',  
36     'django.contrib.contenttypes',  
37     'django.contrib.sessions',  
38     'django.contrib.messages',  
39     'django.contrib.staticfiles',  
40     'rest_framework',  
41     'students',  
42     'api',  
43     'employees',  
44 ]  
45
```

3. Create the new model in MODELS.PY



4. Update the ADMIN.PY



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
Dev/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
Dev/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
Dev/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:

Emp name:

Designation:

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

DjangoREST_APIProject

EXPLORER

- DJANGOREST_APIPROJECT
 - api
 - __pycache__
 - migrations
 - __init__.py
 - admin.py
 - apps.py
 - models.py
 - serializers.py**
 - tests.py
 - urls.py
 - views.py
 - django_rest_main
 - __pycache__

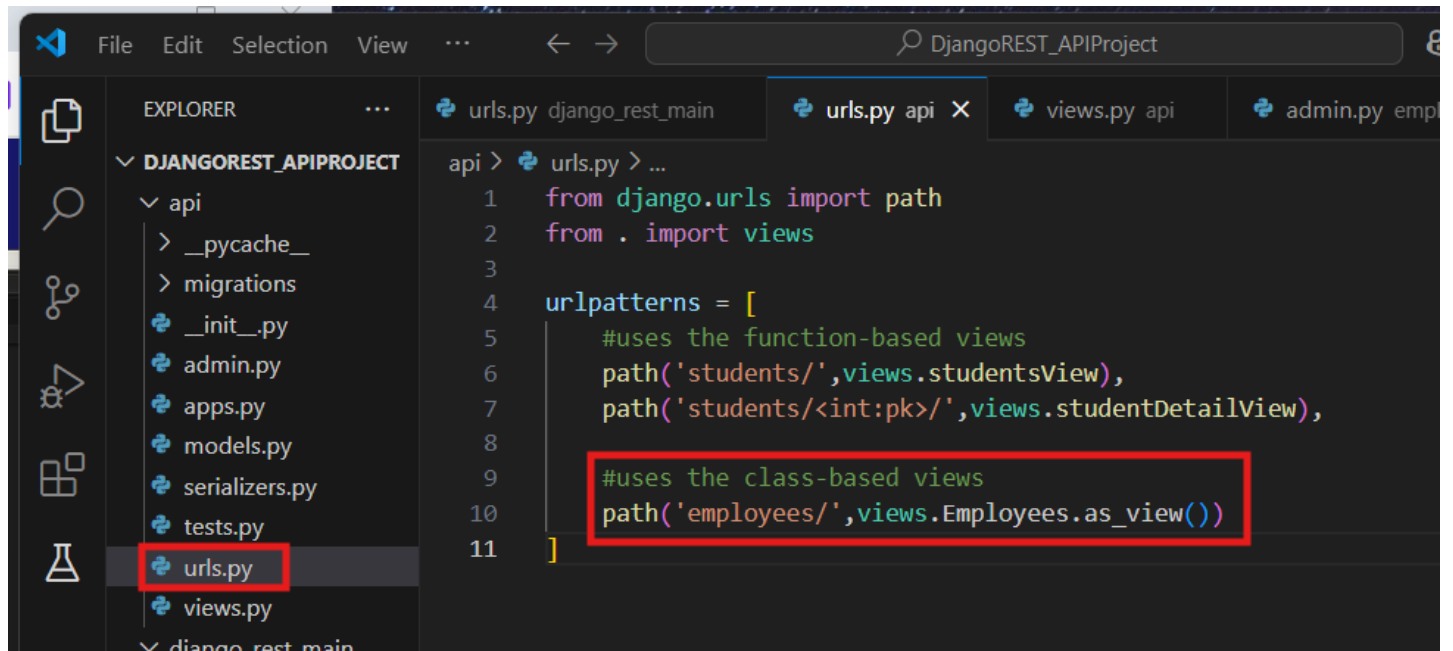
api > serializers.py > EmployeeSerializer > Meta

```

1 from rest_framework import serializers
2 from students.models import Student
3 from employees.models import Employee
4
5 class StudentSerializer(serializers.ModelSerializer):
6     class Meta:
7         model = Student
8         fields = "__all__"
9
10 class EmployeeSerializer(serializers.ModelSerializer):
11     class Meta:
12         model = Employee
13         fields = "__all__"

```

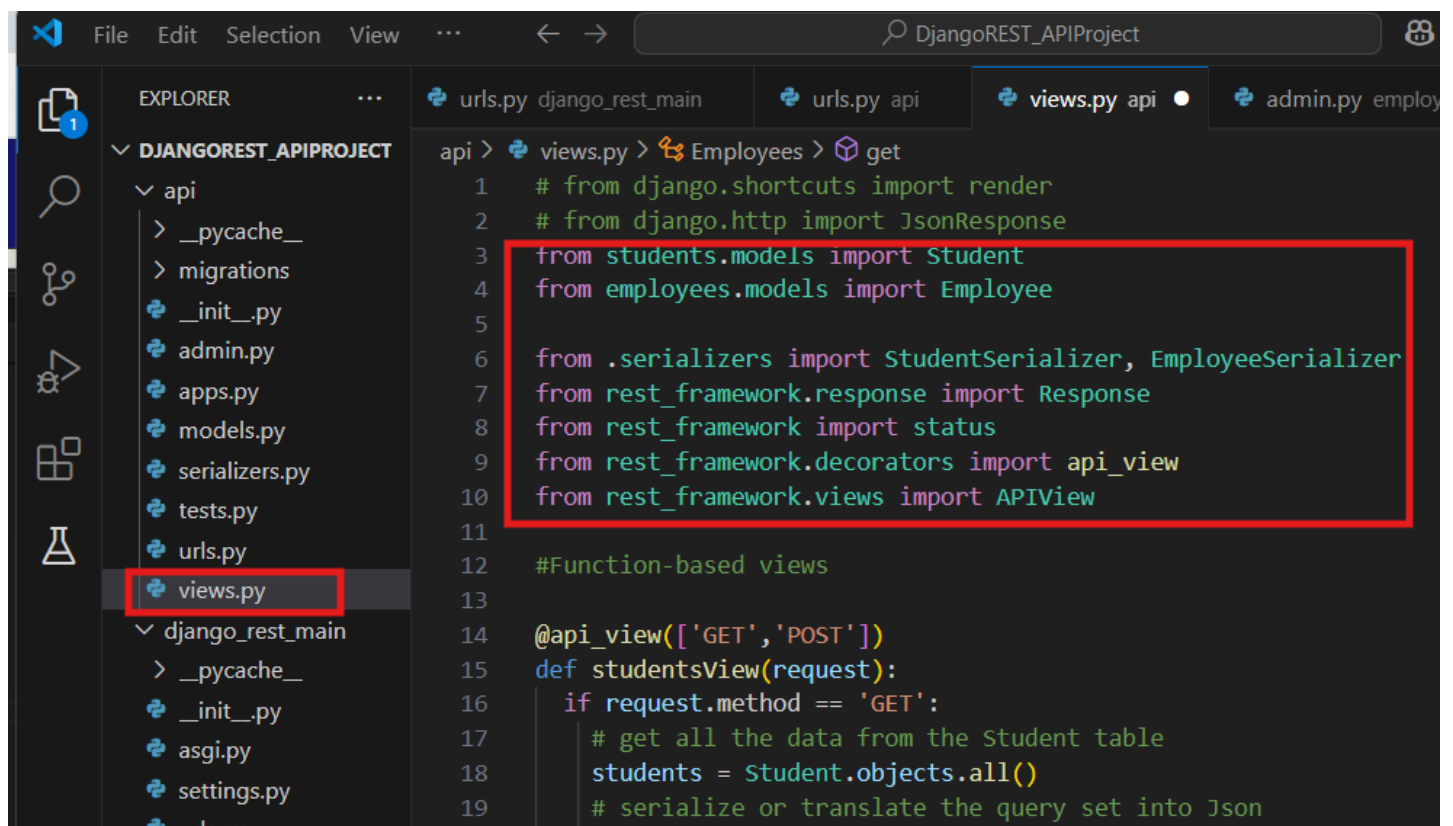
8. To retrieve all the records, go to the API\URLS.PY and update:



```
api > urls.py > ...
1  from django.urls import path
2  from . import views
3
4  urlpatterns = [
5      #uses the function-based views
6      path('students/',views.studentsView),
7      path('students/<int:pk>/',views.studentDetailView),
8
9      #uses the class-based views
10     path('employees/',views.Employees.as_view())
11 ]
```

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

Add the necessary libraries:



```
api > views.py > Employees > get
1  # from django.shortcuts import render
2  # from django.http import JsonResponse
3  from students.models import Student
4  from employees.models import Employee
5
6  from .serializers import StudentSerializer, EmployeeSerializer
7  from rest_framework.response import Response
8  from rest_framework import status
9  from rest_framework.decorators import api_view
10 from rest_framework.views import APIView
11
12 #Function-based views
13
14 @api_view(['GET','POST'])
15 def studentsView(request):
16     if request.method == 'GET':
17         # get all the data from the Student table
18         students = Student.objects.all()
19         # serialize or translate the query set into Json
```

Create the class Employee and its methods:

```
File Edit Selection View ... < -> DjangoREST_APIProject
```

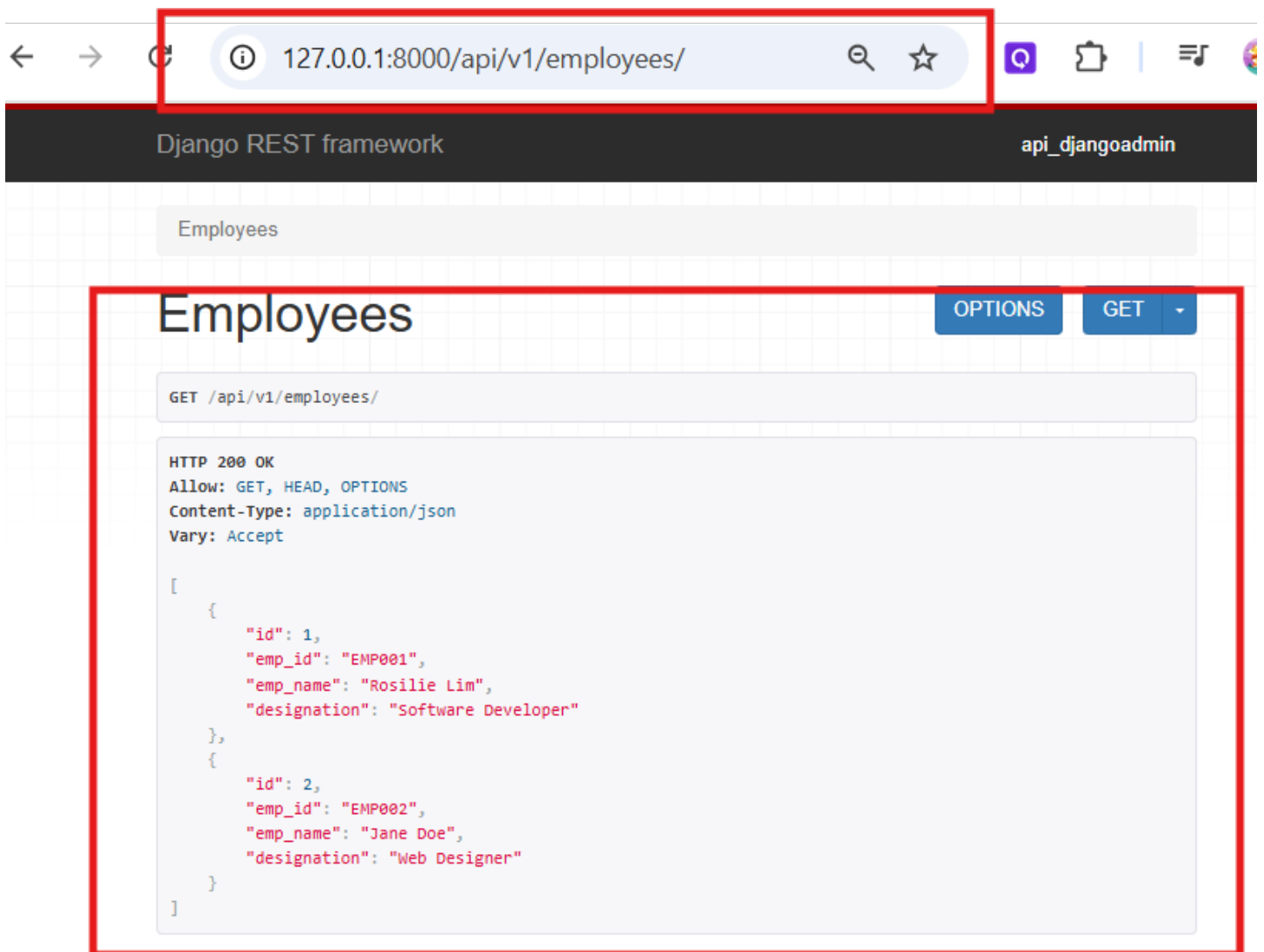
EXPLORER

- ▼ DJAN...
- ▼ api
 - > __pycache__
 - > migrations
 - __init__.py
 - admin.py
 - apps.py
 - models.py
 - serializers.py
 - tests.py
 - urls.py
 - views.py**
- ▼ django_rest_main
 - > __pycache__
 - __init__.py
 - asgi.py
 - settings.py
 - urls.py

api > views.py > ...

```
33 def studentDetailView(request, pk):
49     return Response(serializer.data, status=status.HTTP_200_OK)
50     else:
51         return Response(serializer.errors, status.HTTP_400_BAD_REQUEST)
52
53     # delete a specific record using the PK
54     elif request.method == 'DELETE':
55         student.delete()
56         return Response(status=status.HTTP_204_NO_CONTENT)
57
58
59 # Class-based views
60 class Employees(APIView):
61
62     def get(self, request):
63         employees = Employee.objects.all()
64         serializer = EmployeeSerializer(employees, many=True)
65         return Response(serializer.data, status=status.HTTP_200_OK)
66
```

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



11.

