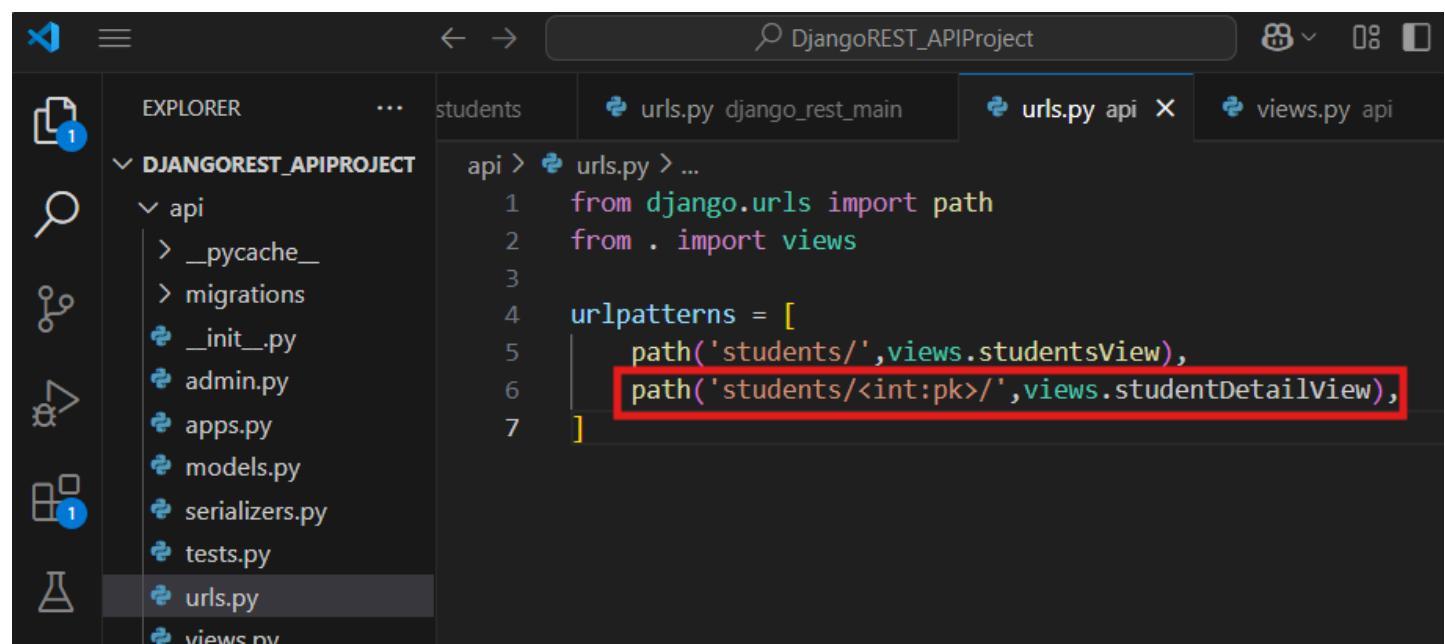


Topic: 6. Single Record Fetching/Updating using PK (PUT/DELETE)

Speaker: / Notebook: API Development using Django Framework



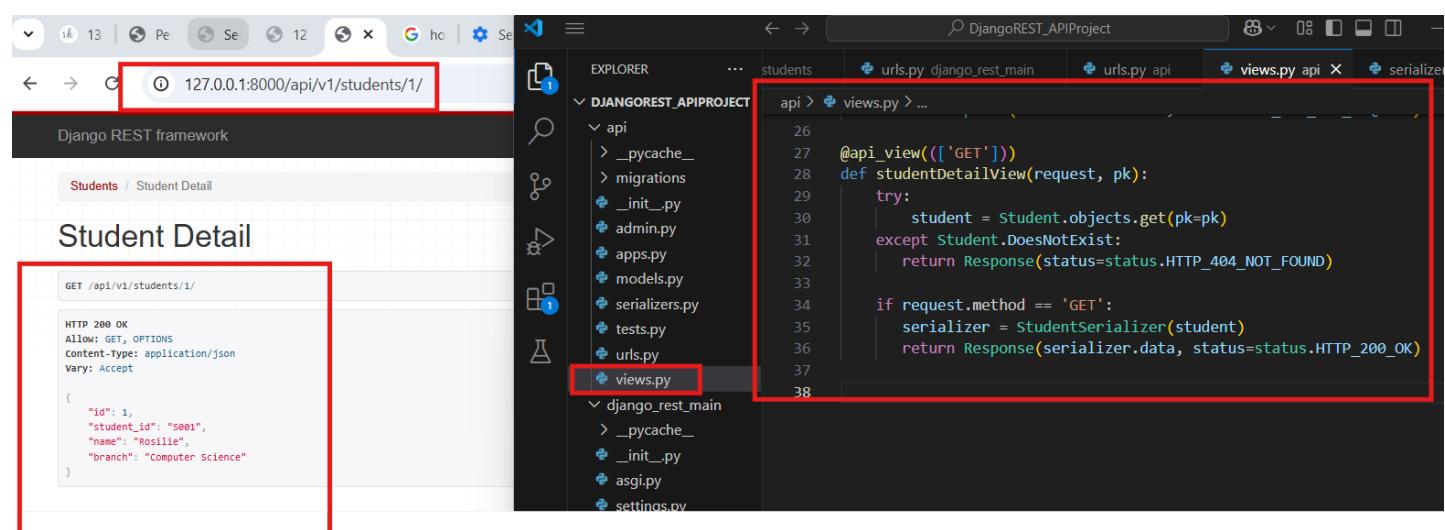
1. To fetch a single record using a primary, we update the URLs.PY and add a new path:



The screenshot shows the VS Code interface with the 'DjangoREST_APIProject' workspace. The 'EXPLORER' sidebar on the left shows the project structure, including 'api' (with 'migrations', '_init_.py', 'admin.py', 'apps.py', 'models.py', 'serializers.py', 'tests.py', 'urls.py', 'views.py'), 'students' (with 'urls.py', 'views.py'), and 'django_rest_main' (with 'urls.py'). The 'urls.py' file in the 'api' directory is open in the editor. A red box highlights the line of code:

```
path('students/<int:pk>/',views.studentDetailView),
```

2. Update the VIEWS.PY and add the path plus the primary to search:



The screenshot shows the VS Code interface with the 'DjangoREST_APIProject' workspace. The 'EXPLORER' sidebar on the left shows the project structure, including 'api' (with 'migrations', '_init_.py', 'admin.py', 'apps.py', 'models.py', 'serializers.py', 'tests.py', 'urls.py', 'views.py'), 'students' (with 'urls.py', 'views.py'), and 'django_rest_main' (with 'urls.py'). The 'views.py' file in the 'api' directory is open in the editor. A red box highlights the code for the 'studentDetailView' view:

```
@api_view(['GET'])
def studentDetailView(request, pk):
    try:
        student = Student.objects.get(pk=pk)
    except Student.DoesNotExist:
        return Response(status=status.HTTP_404_NOT_FOUND)

    if request.method == 'GET':
        serializer = StudentSerializer(student)
        return Response(serializer.data, status=status.HTTP_200_OK)
```

 To the left, a browser window shows the API response for the URL '127.0.0.1:8000/api/v1/students/1/'. A red box highlights the JSON response:

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

{
    "id": 1,
    "student_id": "S001",
    "name": "Rosalie",
    "branch": "Computer Science"
}
```

3. Using the POSTMAN, add the part and the primary key to retrieve that specific record. Click SEND:

My Workspace

Overview

GET http://127.0.0.1:8000/api/v1/students/1

Send

Params Authorization Headers (8) Body Scripts Settings

1 {"student_id": "S006", "name": "Ziggy", "branch": "Business"}

200 OK 12 ms 390 B

Body Cookies Headers (10) Test Results

{ } JSON Preview Visualize

1 {
2 "id": 1,
3 "student_id": "S001",
4 "name": "Rosilie",
5 "branch": "Computer Science"
6 }

In case, you search for the record that doesn't exist, the 404 error message should display:

127.0.0.1:8000/api/v1/students/50/

Django REST framework

api_djangoadmin

Students / Student Detail

Student Detail

OPTIONS GET

GET /api/v1/students/50/

HTTP 404 Not Found
Allow: GET, OPTIONS
Content-Type: application/json
Vary: Accept

4. In POSTMAN (we changed the theme), if you have a non-existing record:

My Workspace

Overview

GET http://127.0.0.1:8000/api/v1/students/50

Send

Params Authorization Headers (8) Body Scripts Settings

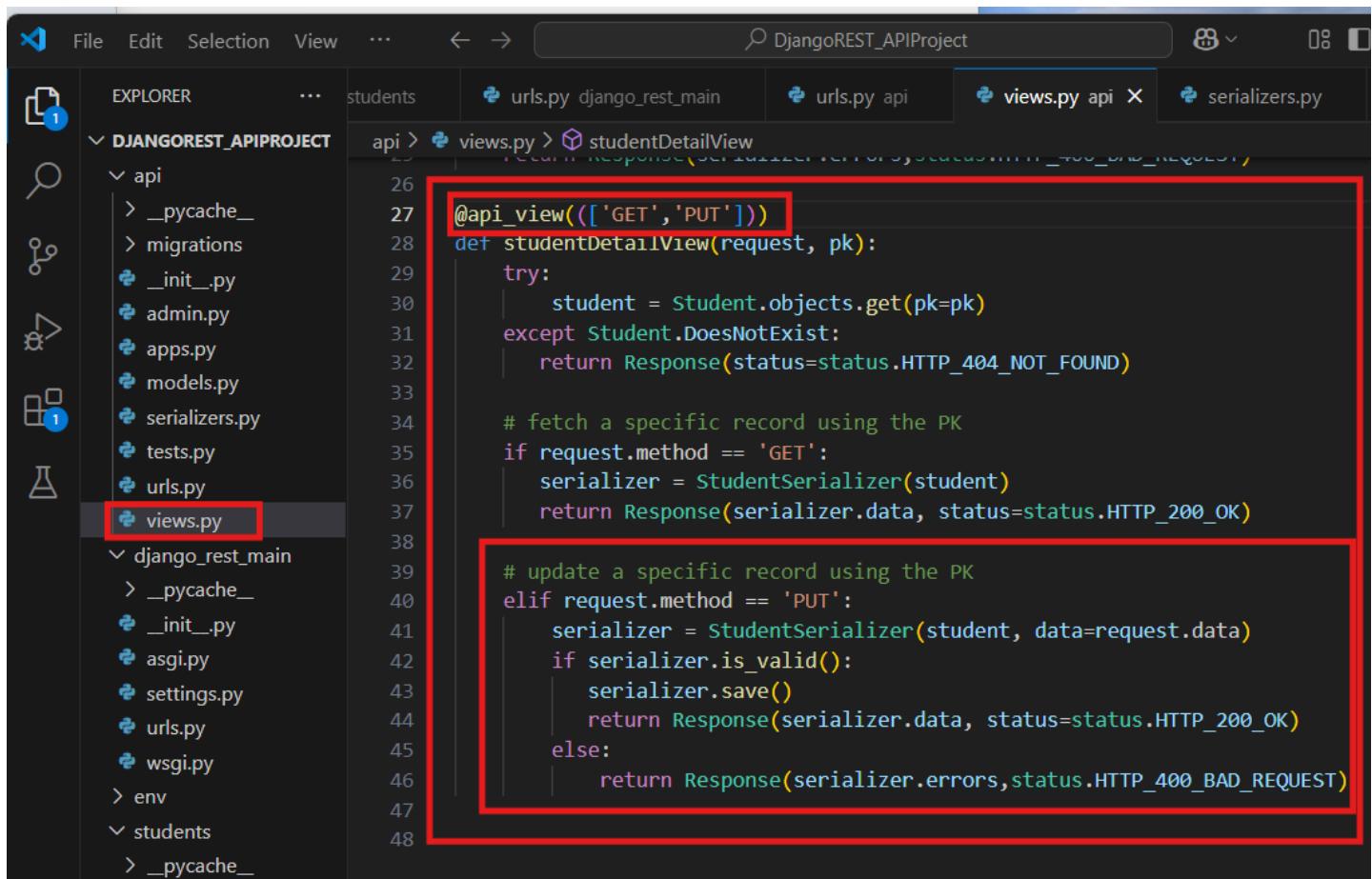
Query Params

Key	Value	Description
Key	Value	Description

Body Cookies Headers (9) Test Results

1 404 Not Found

5. To update a specific record, we update the VIEWS.PY. We use the PUT method.



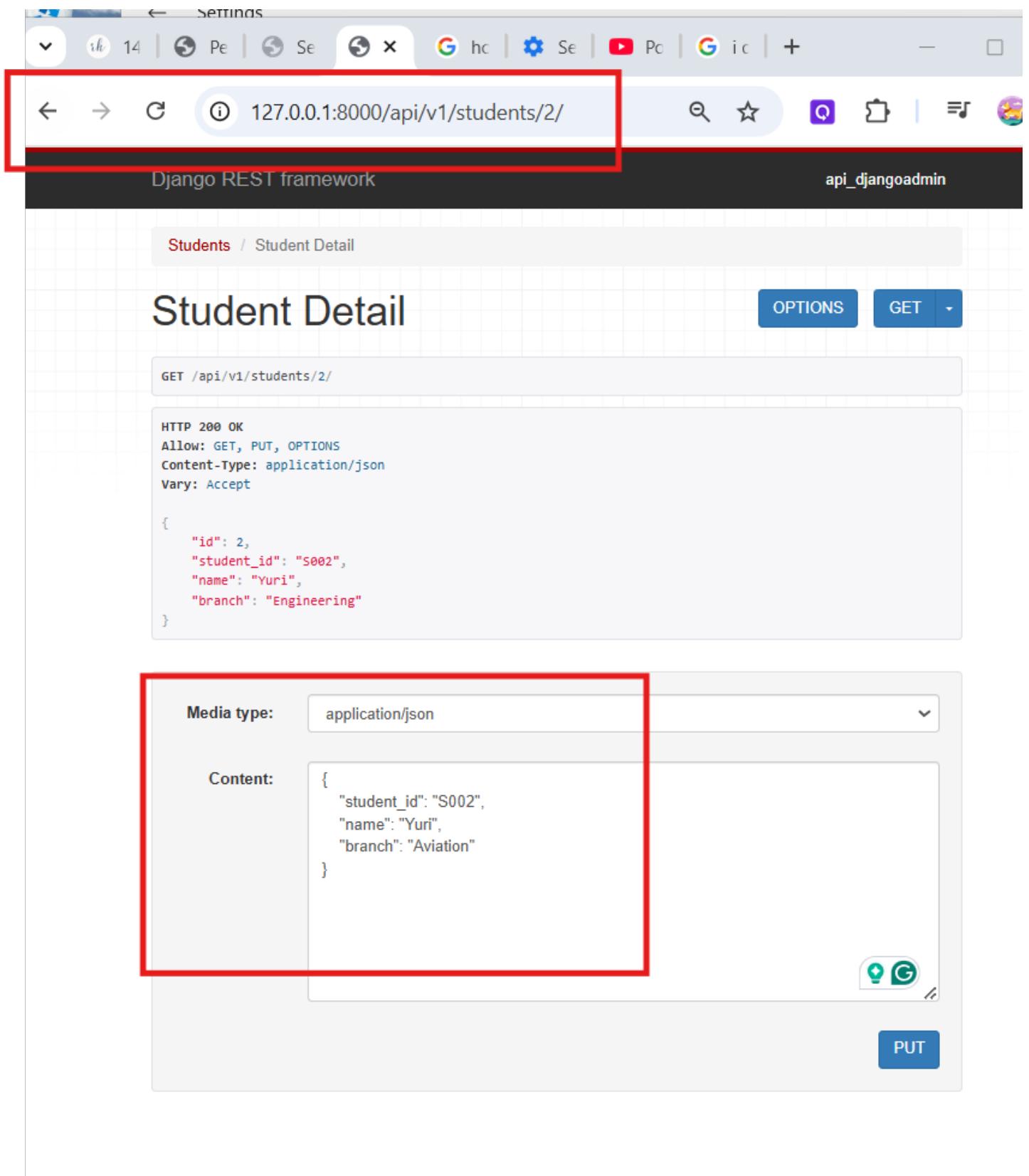
File Edit Selection View ... ← → ○ DjangoREST_APIProject 88 08 EXPLORER ... students urls.py django_rest_main urls.py api views.py api X serializers.py

DJANGOREST_APIPROJECT

api > views.py > studentDetailView

```
26
27 @api_view(['GET', 'PUT'])
28 def studentDetailView(request, pk):
29     try:
30         student = Student.objects.get(pk=pk)
31     except Student.DoesNotExist:
32         return Response(status=status.HTTP_404_NOT_FOUND)
33
34     # fetch a specific record using the PK
35     if request.method == 'GET':
36         serializer = StudentSerializer(student)
37         return Response(serializer.data, status=status.HTTP_200_OK)
38
39     # update a specific record using the PK
40     elif request.method == 'PUT':
41         serializer = StudentSerializer(student, data=request.data)
42         if serializer.is_valid():
43             serializer.save()
44             return Response(serializer.data, status=status.HTTP_200_OK)
45         else:
46             return Response(serializer.errors, status=status.HTTP_400_BAD_REQUEST)
47
48
```

Updating the record:



SETTINGS ← 14 Pe Se G hc Se Po G ic +

127.0.0.1:8000/api/v1/students/2/

Django REST framework api_djangoadmin

Students / Student Detail

Student Detail

OPTIONS GET

GET /api/v1/students/2/

HTTP 200 OK

Allow: GET, PUT, OPTIONS

Content-Type: application/json

Vary: Accept

```
{  
    "id": 2,  
    "student_id": "S002",  
    "name": "Yuri",  
    "branch": "Engineering"  
}
```

Media type: application/json

Content:

```
{  
    "student_id": "S002",  
    "name": "Yuri",  
    "branch": "Aviation"  
}
```

PUT

6. Using the POSTMAN , use the PUT method with the primary record number:

PUT http://127.0.0.1:8000/api/v1/students/1/

Body: { "student_id": "S006", "name": "Ziggy", "branch": "Engineering" }

Response: 200 OK, 67 ms, 388 B

7. To delete a record, use the DELETE method. Update the VIEWS.PY:

```

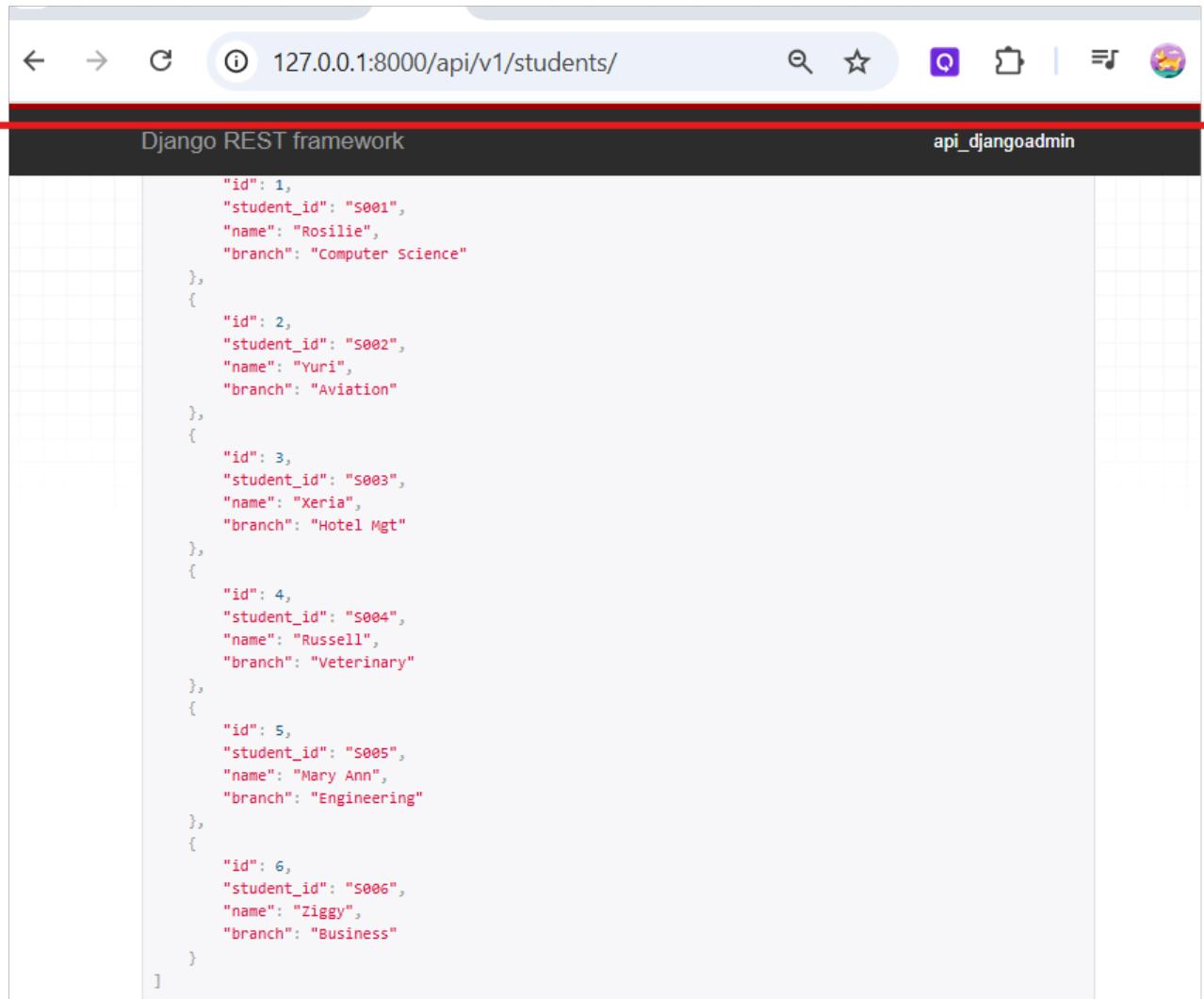
class studentDetailView(APIView):
    @api_view([GET, PUT, DELETE])
    def studentDetailView(request, pk):
        try:
            student = Student.objects.get(pk=pk)
        except Student.DoesNotExist:
            return Response(status=status.HTTP_404_NOT_FOUND)

        # fetch a specific record using the PK
        if request.method == 'GET':
            serializer = StudentSerializer(student)
            return Response(serializer.data, status=status.HTTP_200_OK)

        # update a specific record using the PK
        elif request.method == 'PUT':
            serializer = StudentSerializer(student, data=request.data)
            if serializer.is_valid():
                serializer.save()
                return Response(serializer.data, status=status.HTTP_200_OK)
            else:
                return Response(serializer.errors, status=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)

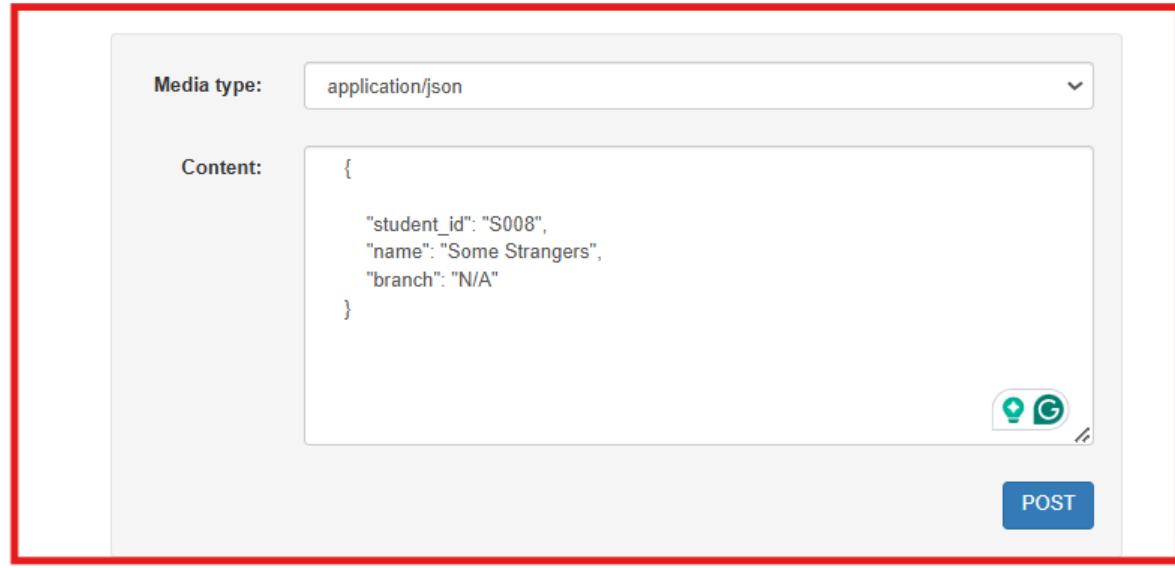
```



Django REST framework

api_djangoadmin

```
[{"id": 1, "student_id": "S001", "name": "Rosilie", "branch": "Computer Science"}, {"id": 2, "student_id": "S002", "name": "Yuri", "branch": "Aviation"}, {"id": 3, "student_id": "S003", "name": "Xeria", "branch": "Hotel Mgt"}, {"id": 4, "student_id": "S004", "name": "Russell", "branch": "Veterinary"}, {"id": 5, "student_id": "S005", "name": "Mary Ann", "branch": "Engineering"}, {"id": 6, "student_id": "S006", "name": "Ziggy", "branch": "Business"}]
```

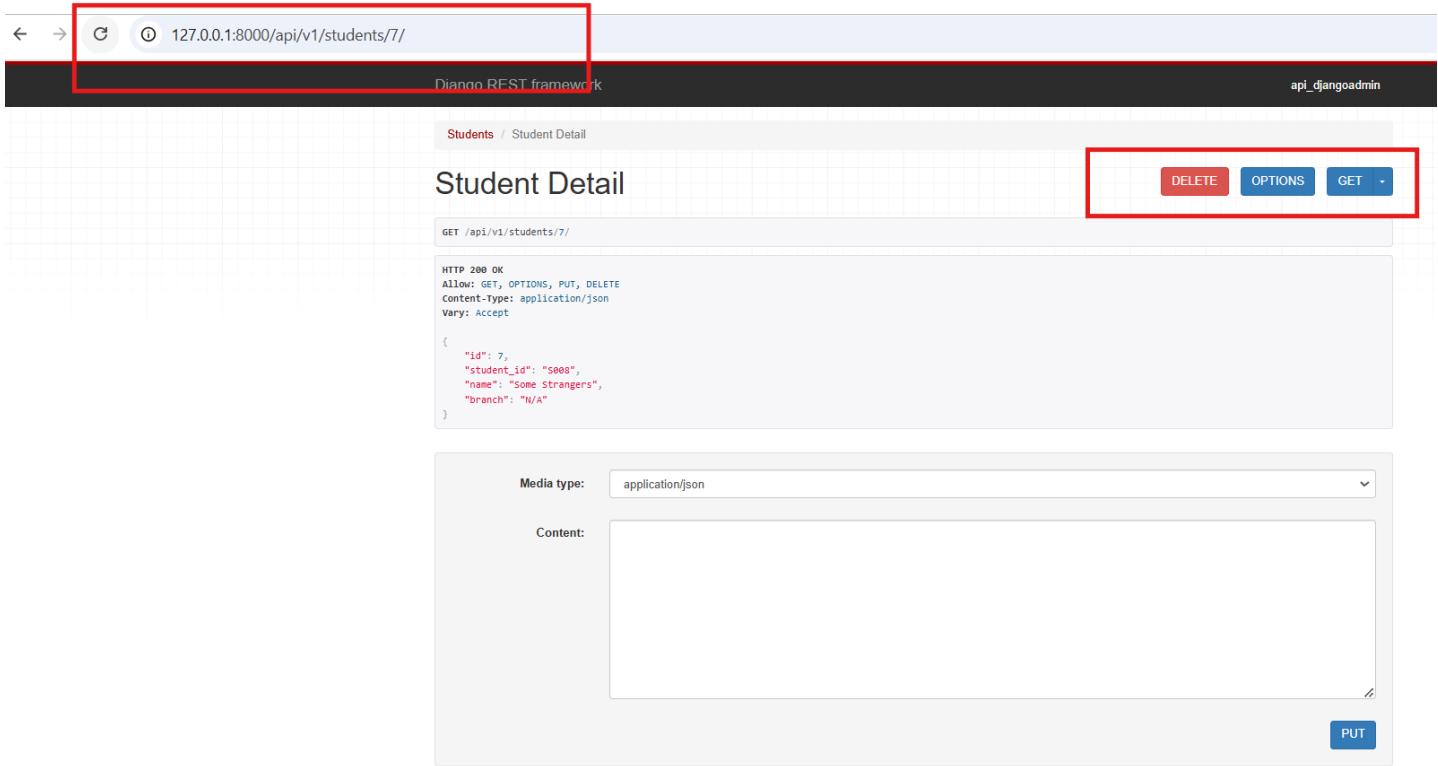


Media type: application/json

Content: {
"student_id": "S008",
"name": "Some Strangers",
"branch": "N/A"
}

POST

When you reload your page, you will see the DELETE button:



127.0.0.1:8000/api/v1/students/7/

Student Detail

DELETE OPTIONS GET

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

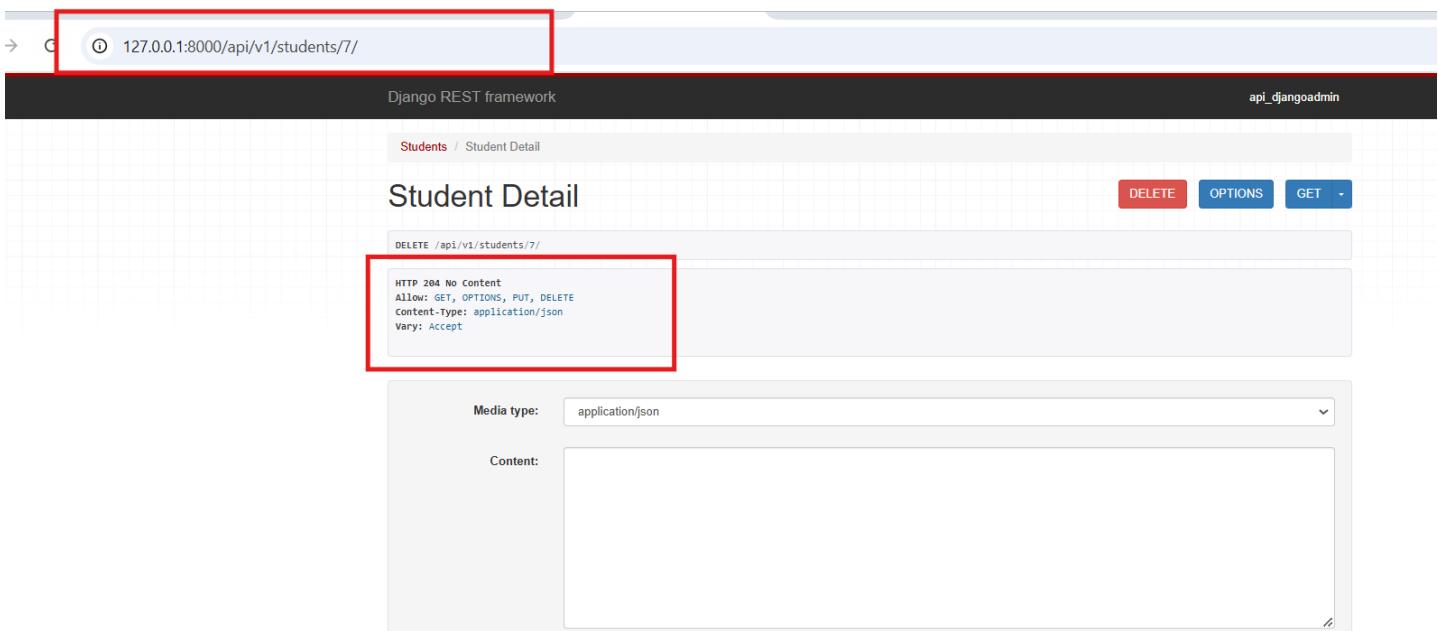
{
    "id": 7,
    "student_id": "s008",
    "name": "Some Strangers",
    "branch": "N/A"
}
```

Media type: application/json

Content:

PUT

8. Now, reload your page and locate that deleted record, it should return HTTP error message:



127.0.0.1:8000/api/v1/students/7/

Student Detail

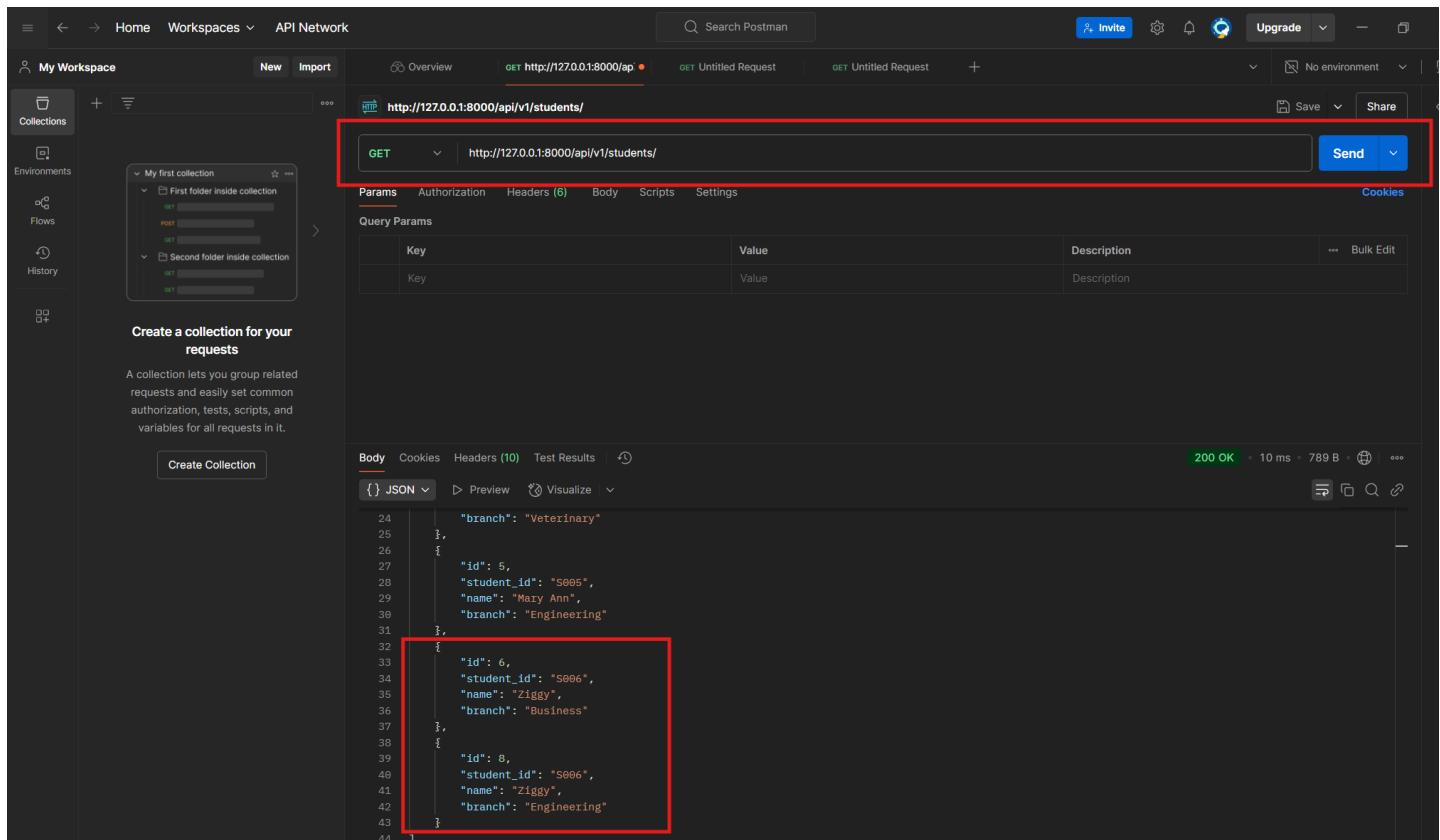
DELETE OPTIONS GET

```
HTTP 204 No Content
Allow: GET, OPTIONS, PUT, DELETE
Content-Type: application/json
Vary: Accept
```

Media type: application/json

Content:

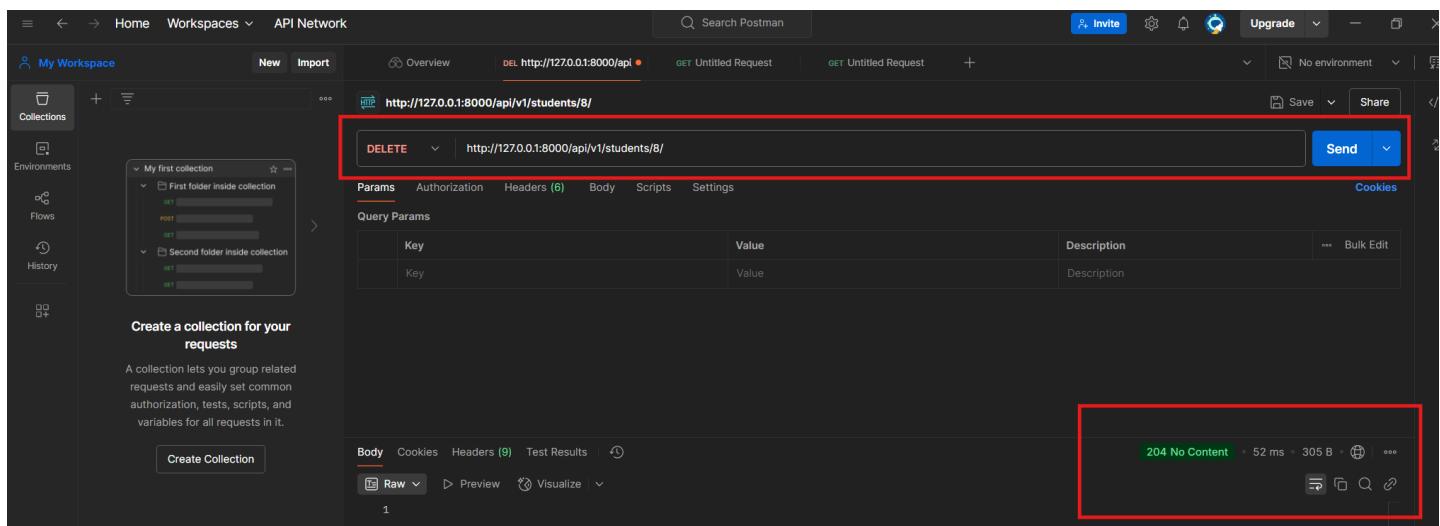
9. Using the POSTMAN, use the delete the method. Assume you have added several records. Delete these unnecessary records.



The screenshot shows the Postman interface with a successful GET request to `http://127.0.0.1:8000/api/v1/students/`. The response body is a JSON array of student records, with the third record highlighted by a red box. The record is as follows:

```
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
{
  "id": 5,
  "student_id": "S005",
  "name": "Mary Ann",
  "branch": "Engineering"
},
{
  "id": 6,
  "student_id": "S006",
  "name": "Ziggy",
  "branch": "Business"
},
{
  "id": 8,
  "student_id": "S006",
  "name": "Ziggy",
  "branch": "Engineering"
}
```

When you use the DELETE method and click SEND, the output should be.



The screenshot shows the Postman interface with a successful DELETE request to `http://127.0.0.1:8000/api/v1/students/8`. The response body is empty, indicated by a red box.

10.