

Topic: 5. Serializing / Deserializing JSON Data (GET/POST)

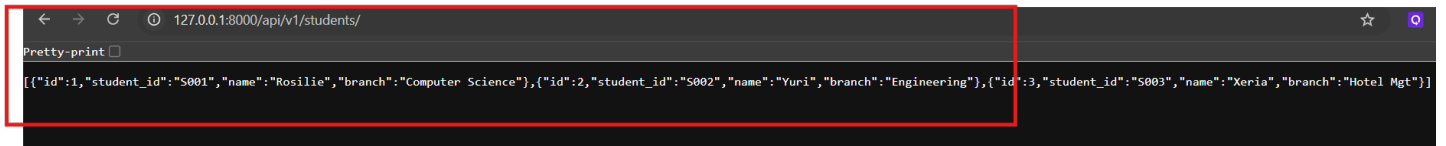
Speaker: Personal / Notebook: API Development using Django Framework



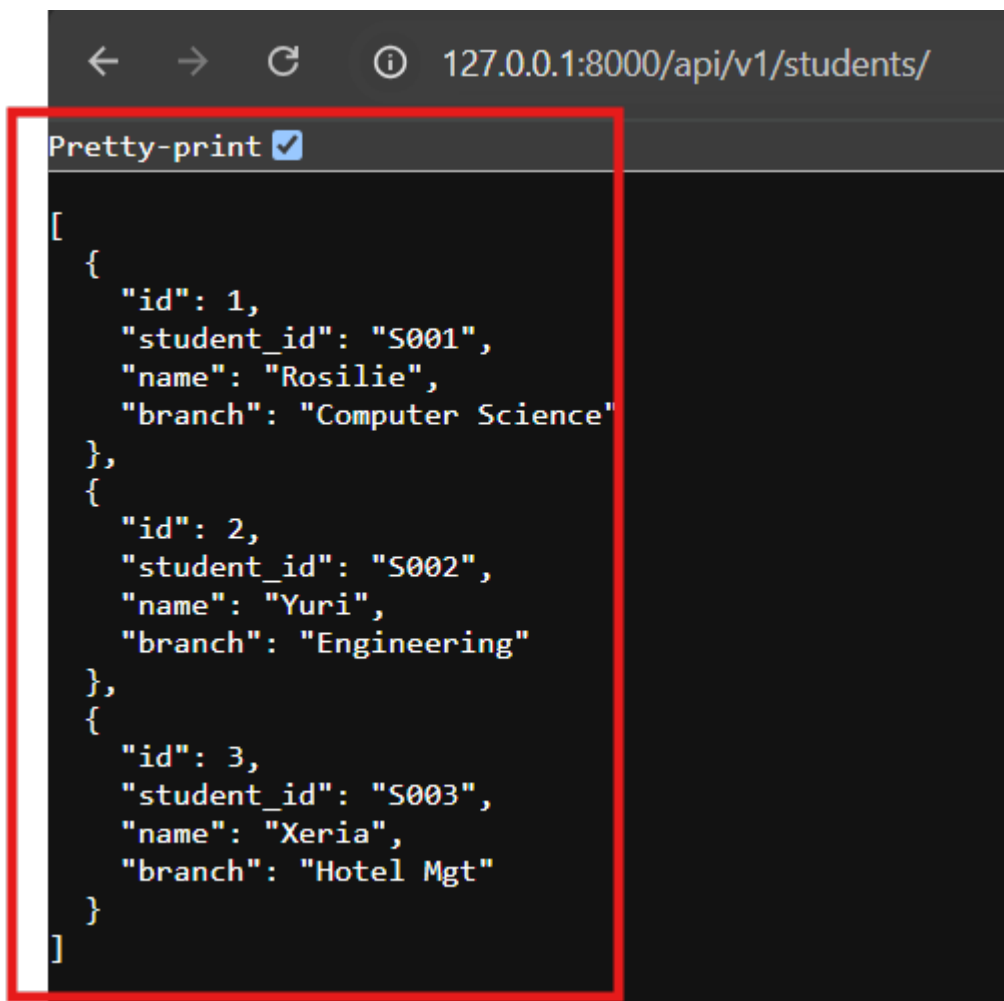
To see more details about serializers, view this [Youtube clip](#)

1. To view the JSON file in a formatted style, we added the [Google Chrome extension, JSON FORMATTER:](#)

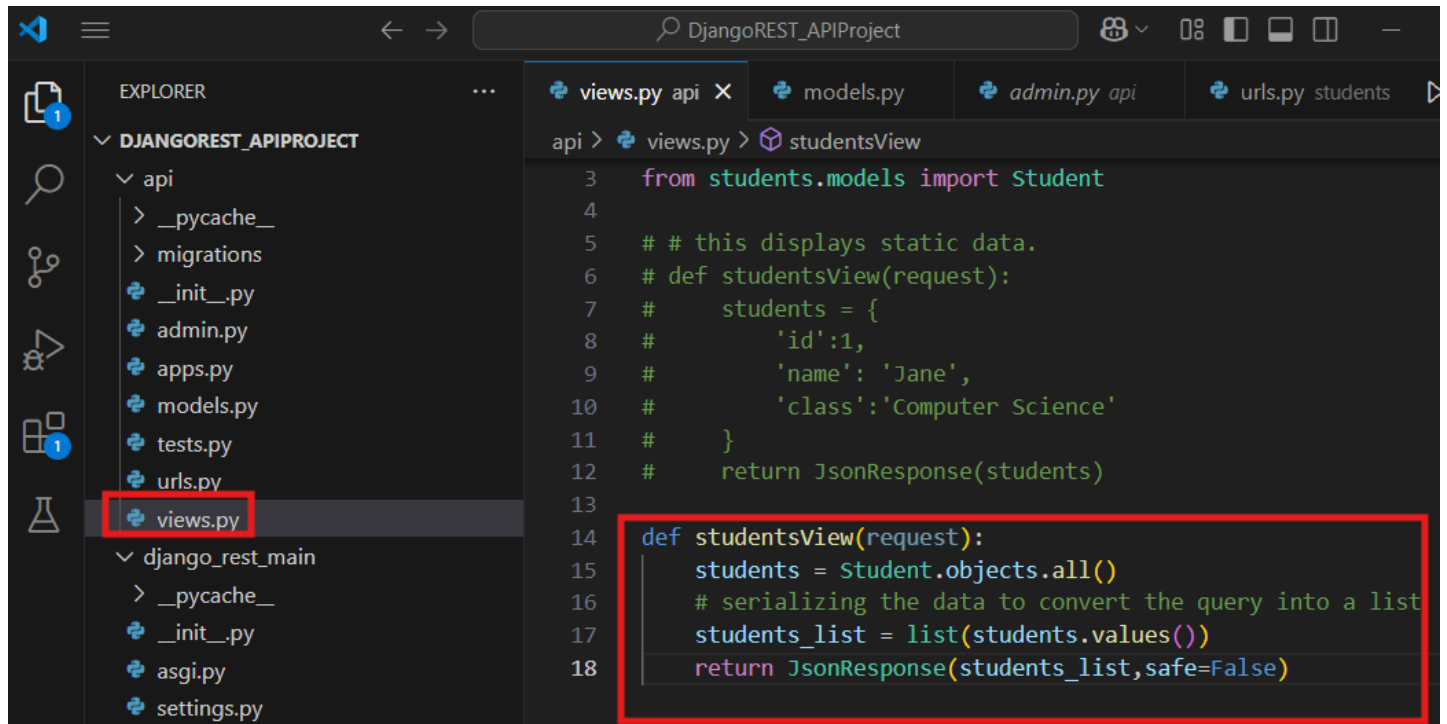
BEFORE:



AFTER:



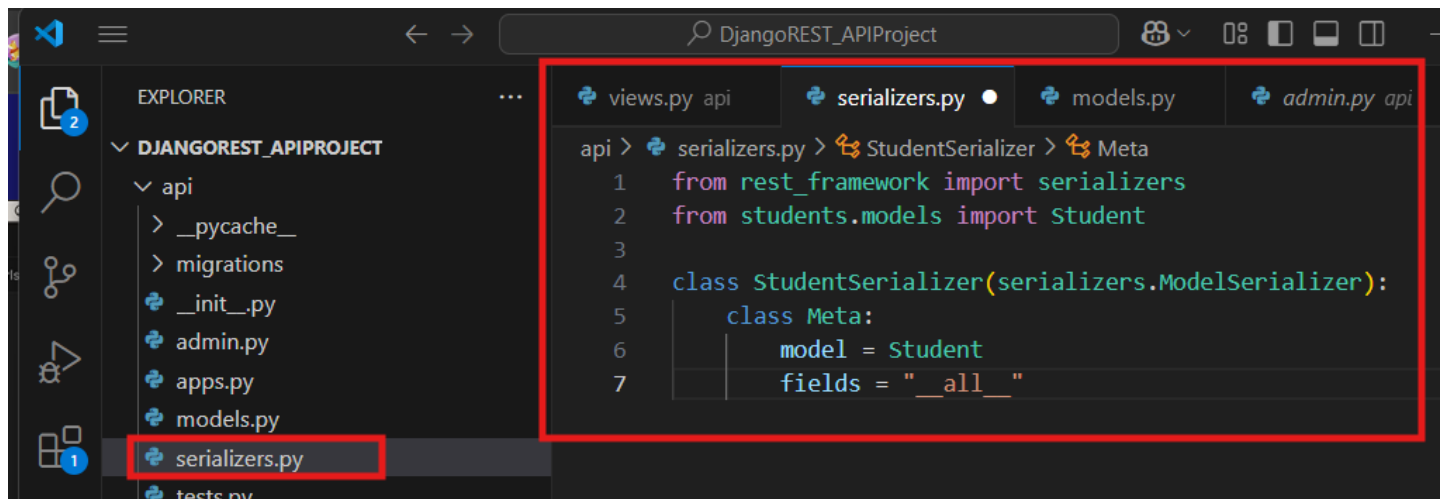
2. Previously, we manually used serializers to convert our query set into a list. The code is below to show the output in Step 1.



The screenshot shows the Visual Studio Code interface for a project named 'DjangoREST_APIProject'. The Explorer sidebar on the left shows the project structure, with 'views.py' highlighted under the 'api' folder. The main editor window displays the code for 'views.py', specifically the 'studentsView' function. The code is as follows:

```
3 from students.models import Student
4
5 # # this displays static data.
6 # def studentsView(request):
7 #     students = {
8 #         'id':1,
9 #         'name': 'Jane',
10 #         'class':'Computer Science'
11 #     }
12 #     return JsonResponse(students)
13
14 def studentsView(request):
15     students = Student.objects.all()
16     # serializing the data to convert the query into a list
17     students_list = list(students.values())
18     return JsonResponse(students_list,safe=False)
```

3. In Django, we can use serializer tools. In the API app folder, create a new file SERIALIZERS.PY:

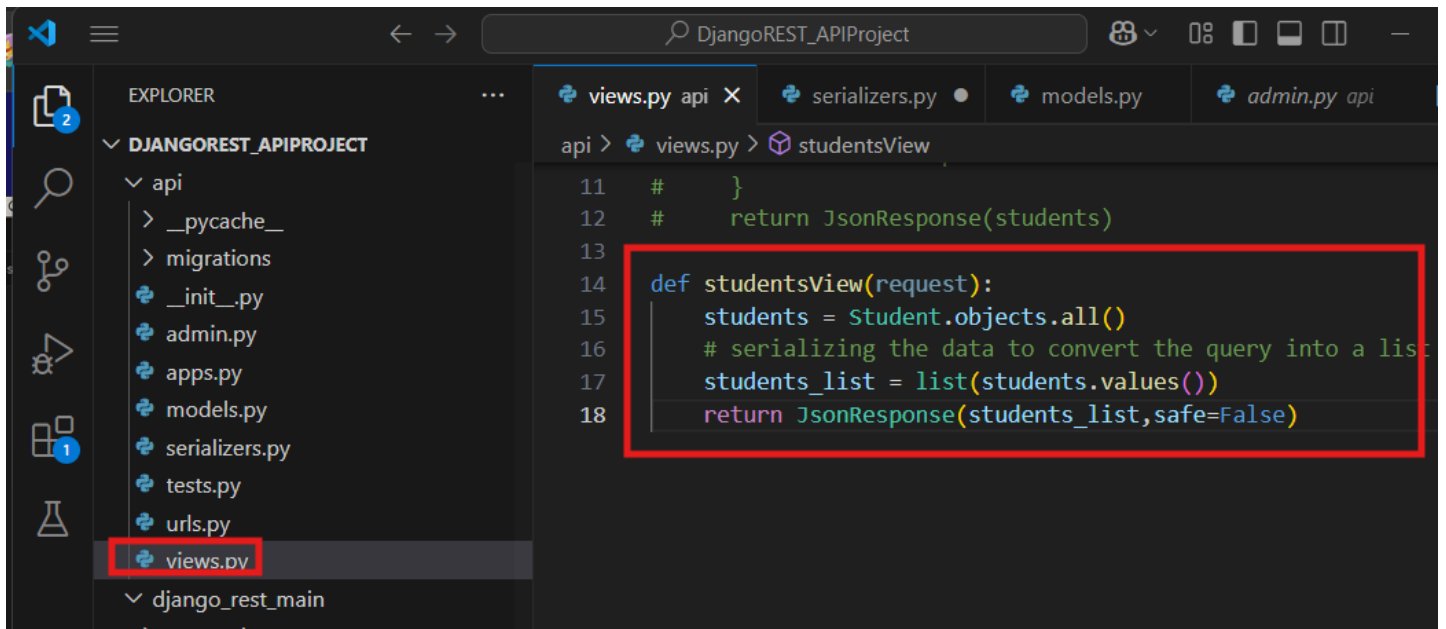


The screenshot shows the Visual Studio Code interface for the 'DjangoREST_APIProject'. The Explorer sidebar on the left shows the project structure, with 'serializers.py' highlighted under the 'api' folder. The main editor window displays the code for 'serializers.py', specifically the 'StudentSerializer' class. The code is as follows:

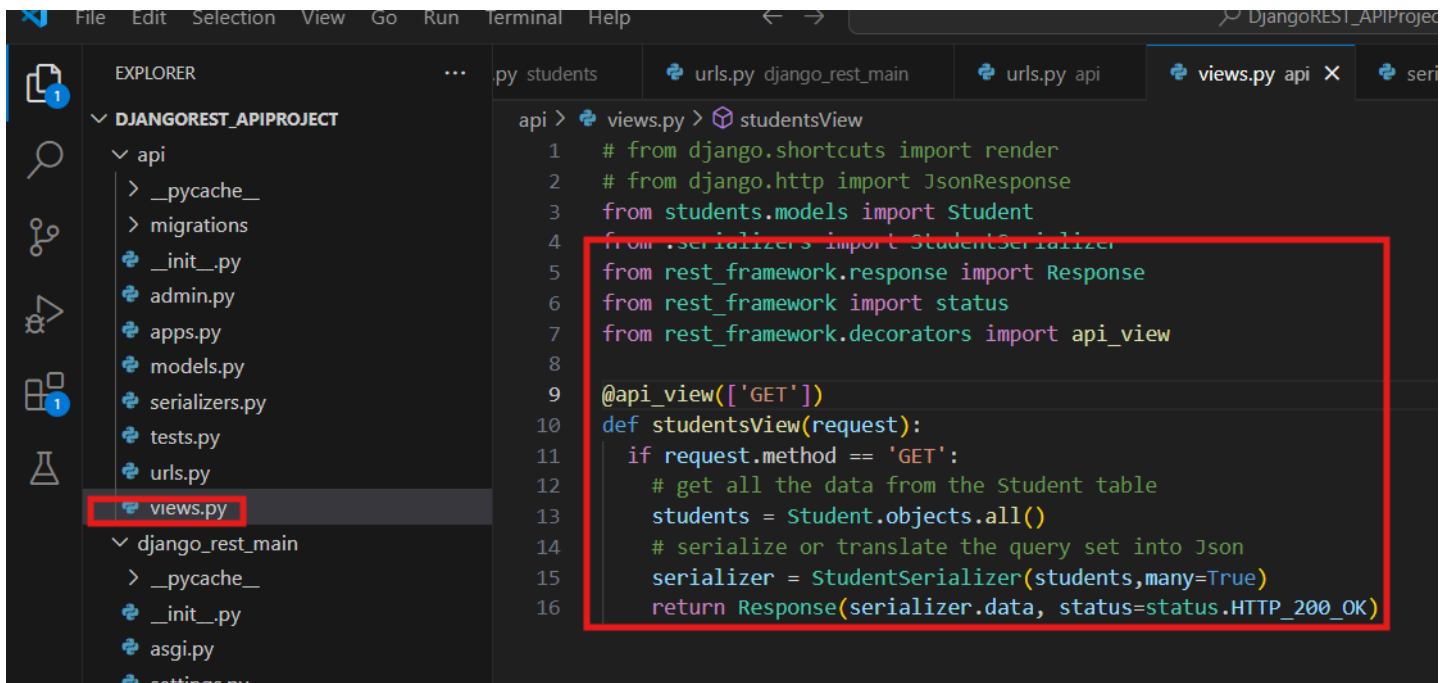
```
1 from rest_framework import serializers
2 from students.models import Student
3
4 class StudentSerializer(serializers.ModelSerializer):
5     class Meta:
6         model = Student
7         fields = "__all__"
```

4. Update our API\VIEWS.PY:

FROM manual serialization:



TO:



So, when you run the URL path again:

<http://127.0.0.1:8000/api/v1/students/>

← → ↻ ⓘ 127.0.0.1:8000/api/v1/students/ ☆ 🔍 📁 🎵 🌐

Django REST framework api_djangoadmin

Students

Students

OPTIONS GET ▾

GET /api/v1/students/

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

```
[
  {
    "id": 1,
    "student_id": "S001",
    "name": "Rosilie",
    "branch": "Computer Science"
  },
  {
    "id": 2,
    "student_id": "S002",
    "name": "Yuri",
    "branch": "Engineering"
  },
  {
    "id": 3,
    "student_id": "S003",
    "name": "Xeria",
    "branch": "Hotel Mgt"
  }
]
```

5. Now, if you update your database model for a new record and use the GET button from Step 4, you will be able to use GET button to get the latest added records or you can simply refresh your page and that will be considered as a GET method.

← → ↻ ⓘ 127.0.0.1:8000/api/v1/students/ ☆ Q | 🎵 🌐

Django REST framework api_djangoadmin

Students

Students

OPTIONS GET ▾

GET /api/v1/students/

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

```
[
  {
    "id": 1,
    "student_id": "S001",
    "name": "Rosilie",
    "branch": "Computer Science"
  },
  {
    "id": 2,
    "student_id": "S002",
    "name": "Yuri",
    "branch": "Engineering"
  },
  {
    "id": 3,
    "student_id": "S003",
    "name": "Xeria",
    "branch": "Hotel Mgt"
  },
  {
    "id": 4,
    "student_id": "S004",
    "name": "Russell",
    "branch": "Veterinary"
  }
]
```

6. Now using **POSTMAN**, you can copy the same API link paste it into the POSTMAN search bar and use GET method. It should return all records from the database. Simply click on + and add the same path we used from the browser. To use POSTMAN, this must be installed in your device.

The screenshot shows the Postman interface. On the left, a sidebar displays the 'Students' collection with a GET request to `/api/v1/students/`. The main panel shows the request details, including the URL `http://127.0.0.1:8000/api/v1/students/` and the method `GET`. The response is a 200 OK status with a JSON body containing an array of student objects:

```
[
  {
    "id": 1,
    "student_id": "S001",
    "name": "Rosllie",
    "branch": "Computer Science"
  },
  {
    "id": 2,
    "student_id": "S002",
    "name": "Yuri",
    "branch": "Engineering"
  },
  {
    "id": 3,
    "student_id": "S003",
    "name": "Xeria",
    "branch": "Hotel Mgt"
  },
  {
    "id": 4,
    "student_id": "S004",
    "name": "Russell",
    "branch": "Veterinary"
  }
]
```

7. To store data using the Django Rest Framework, update the VIEWS.PY to allow for POST method.

The screenshot shows a code editor with the `views.py` file open. The code defines a `studentsView` function that handles both GET and POST requests. The POST logic is highlighted with a red box:

```
@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
        serializer = StudentSerializer(students, many=True)
        return Response(serializer.data, status=status.HTTP_200_OK)
    elif request.method == 'POST':
        # saves data into our database
        serializer = StudentSerializer(data=request.data)
        if serializer.is_valid():
            serializer.save()
            return Response(serializer.data, status=status.HTTP_201_CREATED)
        # if data are not valid
        print(serializer.errors)
        return Response(serializer.errors, status=status.HTTP_400_BAD_REQUEST)
```

When you reload your page, then you can insert a new post:

12 Stc | Perso | Select | 127.0. | S x | how c | Settin | +

127.0.0.1:8000/api/v1/students/

api_djangoadmin

Students

OPTIONSGET

GET /api/v1/students/

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

```
[
  {
    "id": 1,
    "student_id": "S001",
    "name": "Rosilie",
    "branch": "Computer Science"
  },
  {
    "id": 2,
    "student_id": "S002",
    "name": "Yuri",
    "branch": "Engineering"
  },
  {
    "id": 3,
    "student_id": "S003",
    "name": "Xeria",
    "branch": "Hotel Mgt"
  },
  {
    "id": 4,
    "student_id": "S004",
    "name": "Russell",
    "branch": "Veterinary"
  }
]
```

Media type: application/json

Content:

```
{
  "student_id": "S005",
  "name": "Mary Ann",
  "branch": "Engineering"
}
```

POST

Browser tabs: 12 Stc | Perso | Select | 127.0 | S x | how c | Settin | +

Address bar: 127.0.0.1:8000/api/v1/students/

Django REST framework api_djangoadmin

Students

OPTIONS GET

POST /api/v1/students/

HTTP 201 Created
Allow: POST, OPTIONS, GET
Content-Type: application/json
Vary: Accept

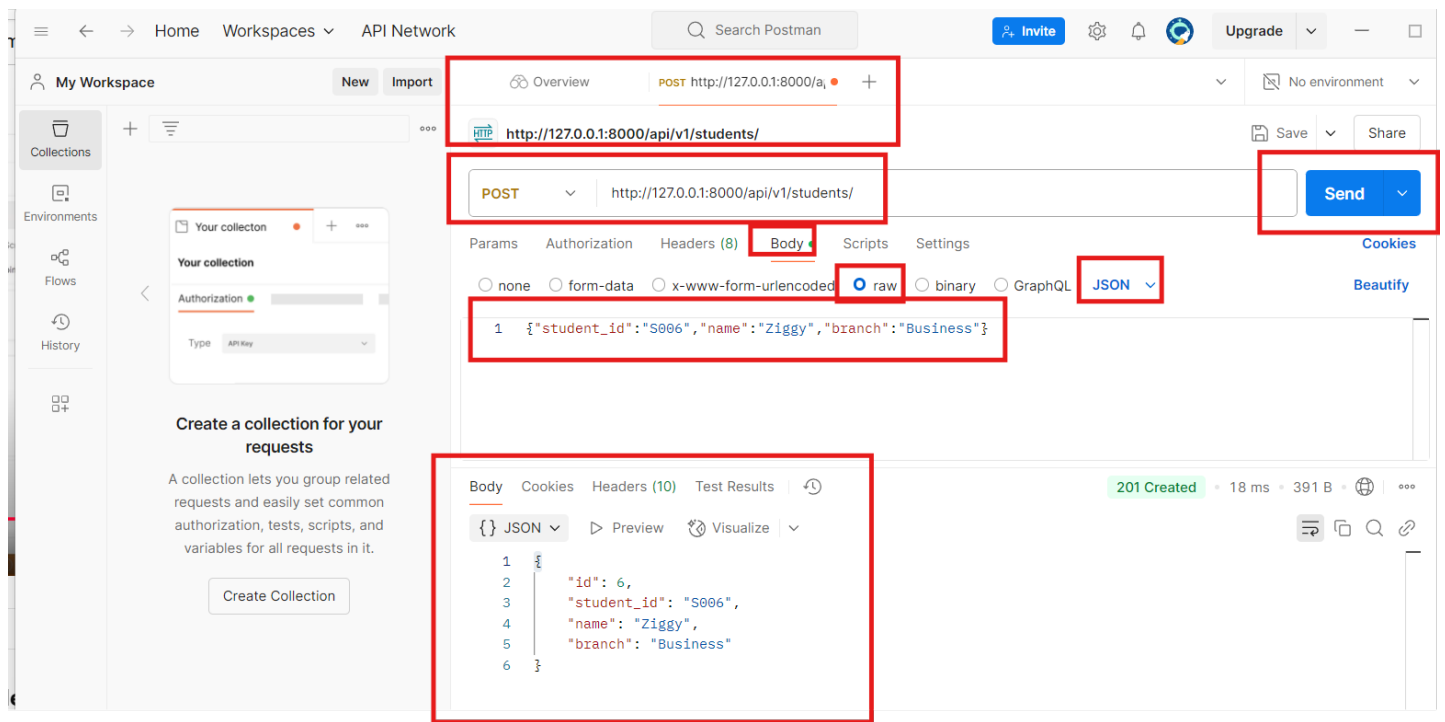
```
{  
  "id": 5,  
  "student_id": "s005",  
  "name": "Mary Ann",  
  "branch": "Engineering"  
}
```

Media type: application/json

Content:

POST

8. To use POSTMAN, add the path again. Select BODY, then RAW, then JSON. Add your records then select select the SEND method.



9. Now, to see the newly inserted record, use the GET method. You will then see the newly added record.

