

## Topic: 8. Updating the Models using Class-Based Views

Speaker: / Notebook: API Development using Django Framework

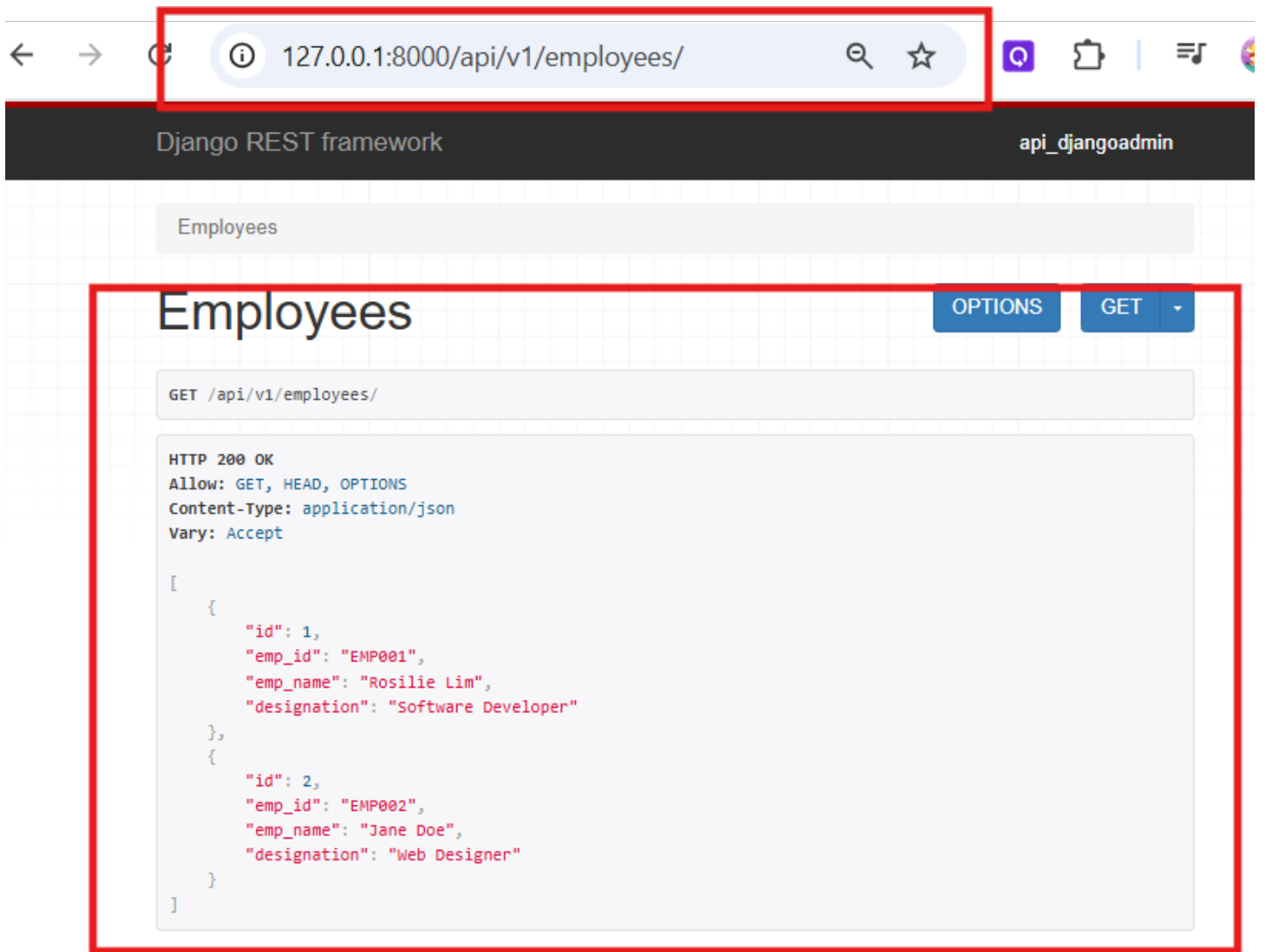


We previously created a new app called EMPLOYEES and a new Employee serializer for our Employee class.

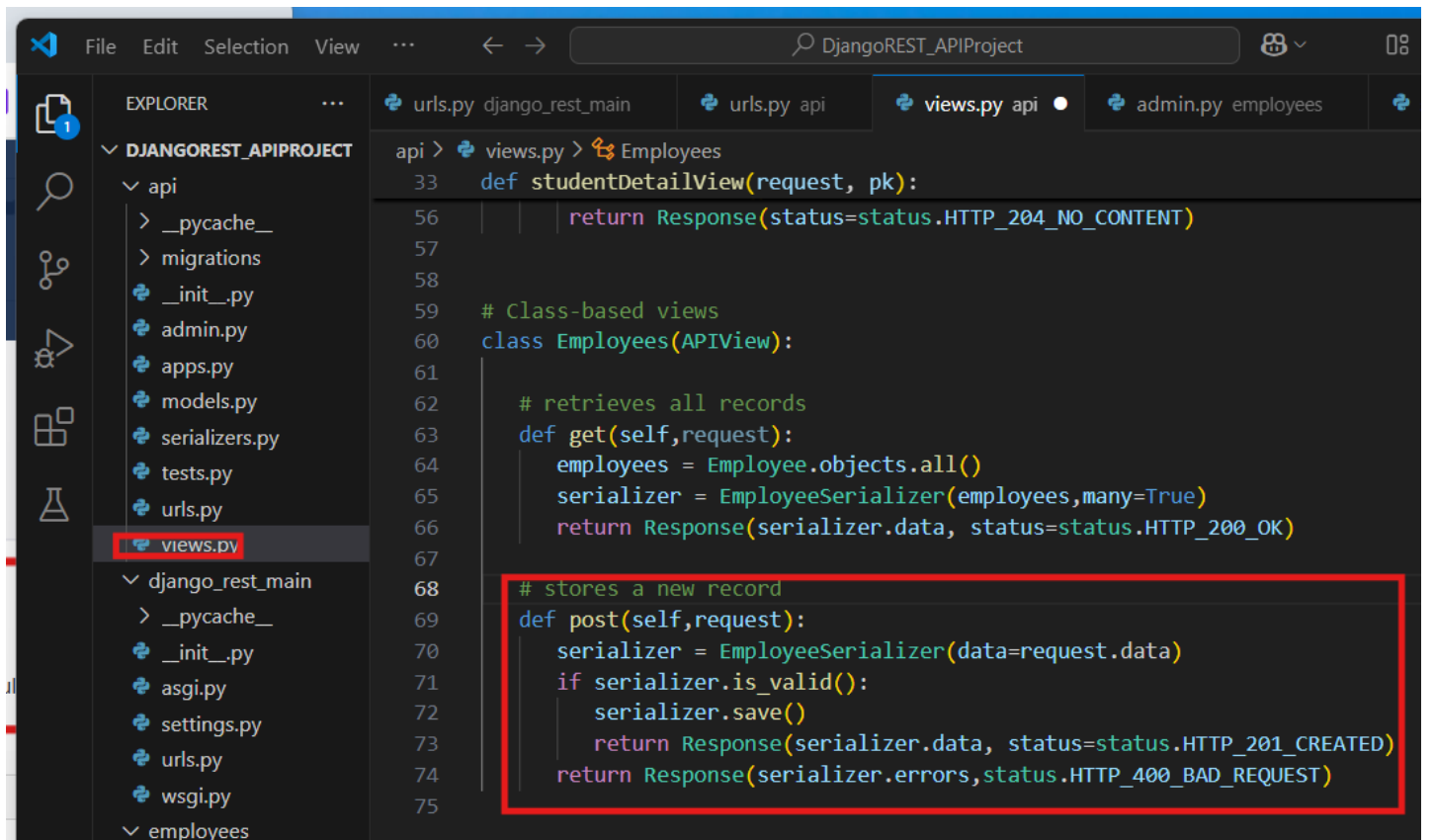
1. Previously, we updated the APIVIEWS.PY to create an EMPLOYEE class and its methods:

```
File Edit Selection View ... 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__  
    < .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
```

We run the new path:



2. To post or add a new record to our model:



Add the new record. Follow the correct format to be able to save successfully.

The screenshot shows a web browser at the URL `127.0.0.1:8000/api/v1/employees/`. The page title is "Django REST framework" and the user is logged in as "api\_djangoadmin". The main heading is "Employees".

On the right side, there are two buttons: "OPTIONS" and "GET". Below them, the request method is shown as "GET /api/v1/employees/".

The response is an HTTP 200 OK with the following headers:  
Allow: GET, POST, HEAD, OPTIONS  
Content-Type: application/json  
Vary: Accept

The response body is a JSON array of two employee objects:

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

Below the response, there is a form for adding a new record. The "Media type" is set to "application/json". The "Content" field contains the following JSON object:

```
{
  "emp_id": "EMP003",
  "emp_name": "Russell Lim",
  "designation": "Security"
}
```

A "POST" button is located at the bottom right of the form, highlighted with a red box.

This will result to:

Employees

# Employees

OPTIONS

GET ▾

POST /api/v1/employees/

HTTP 201 Created

Allow: GET, POST, HEAD, OPTIONS  
Content-Type: application/json  
Vary: Accept

```
{  
  "id": 3,  
  "emp_id": "EMP003",  
  "emp_name": "Russell Lim",  
  "designation": "Security"  
}
```

Media type:

application/json ▾

Content:

POST

3. To allow a single record operation like CRUD on a specific record, we update our APIURLS.PY as:

The screenshot shows the VS Code editor interface for a Django REST API project. The Explorer sidebar on the left shows the project structure under 'DJANGOREST\_APIPROJECT', with 'urls.py' highlighted. The main editor window displays the 'urls.py' file for the 'api' app. The code includes imports for 'path' and 'views', and a list of URL patterns. The following line is highlighted with a red box: `path('employees/<int:pk>', views.EmployeeDetail.as_view()),`

4. Update the APIVIEWS.PY to include EmployeeDetail class:

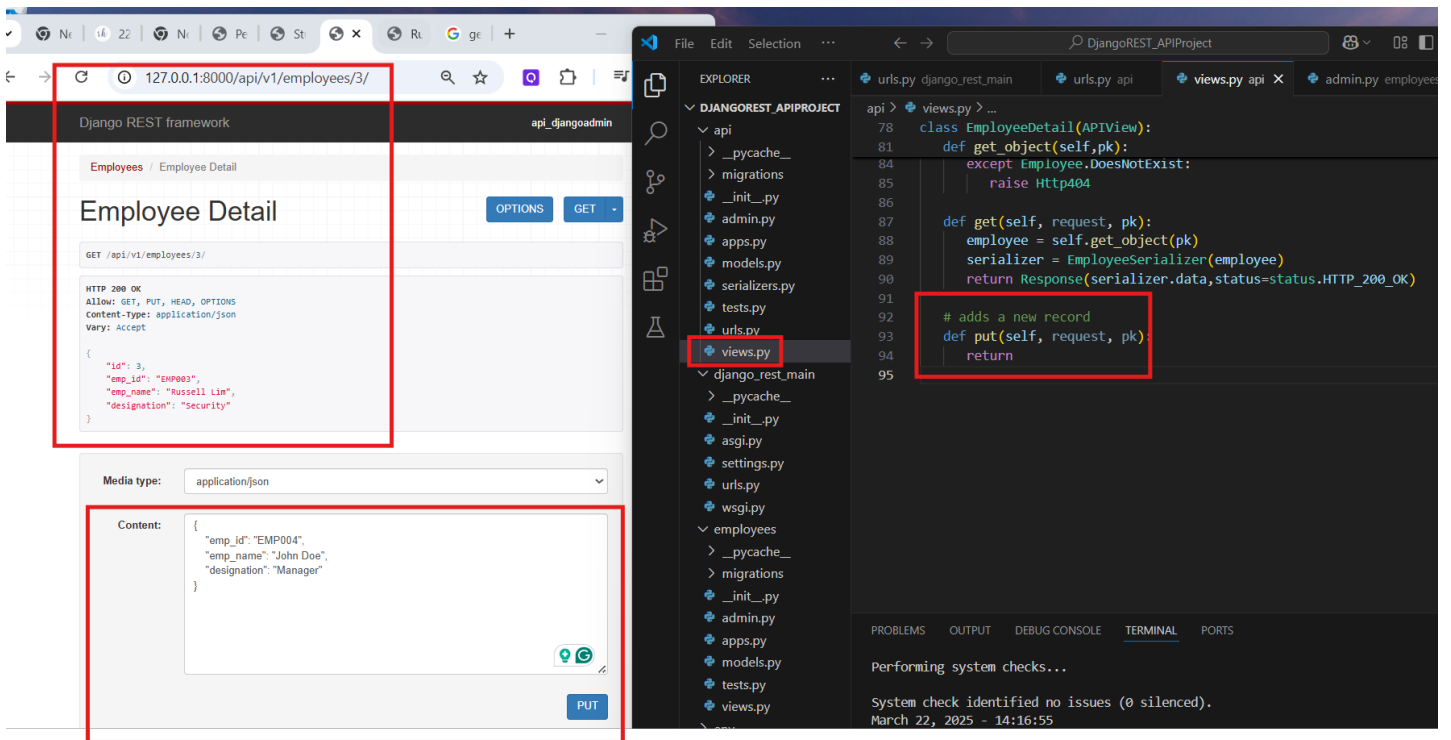
To test, add the path:

`http://127.0.0.1:8000/api/v1/employees/3/`

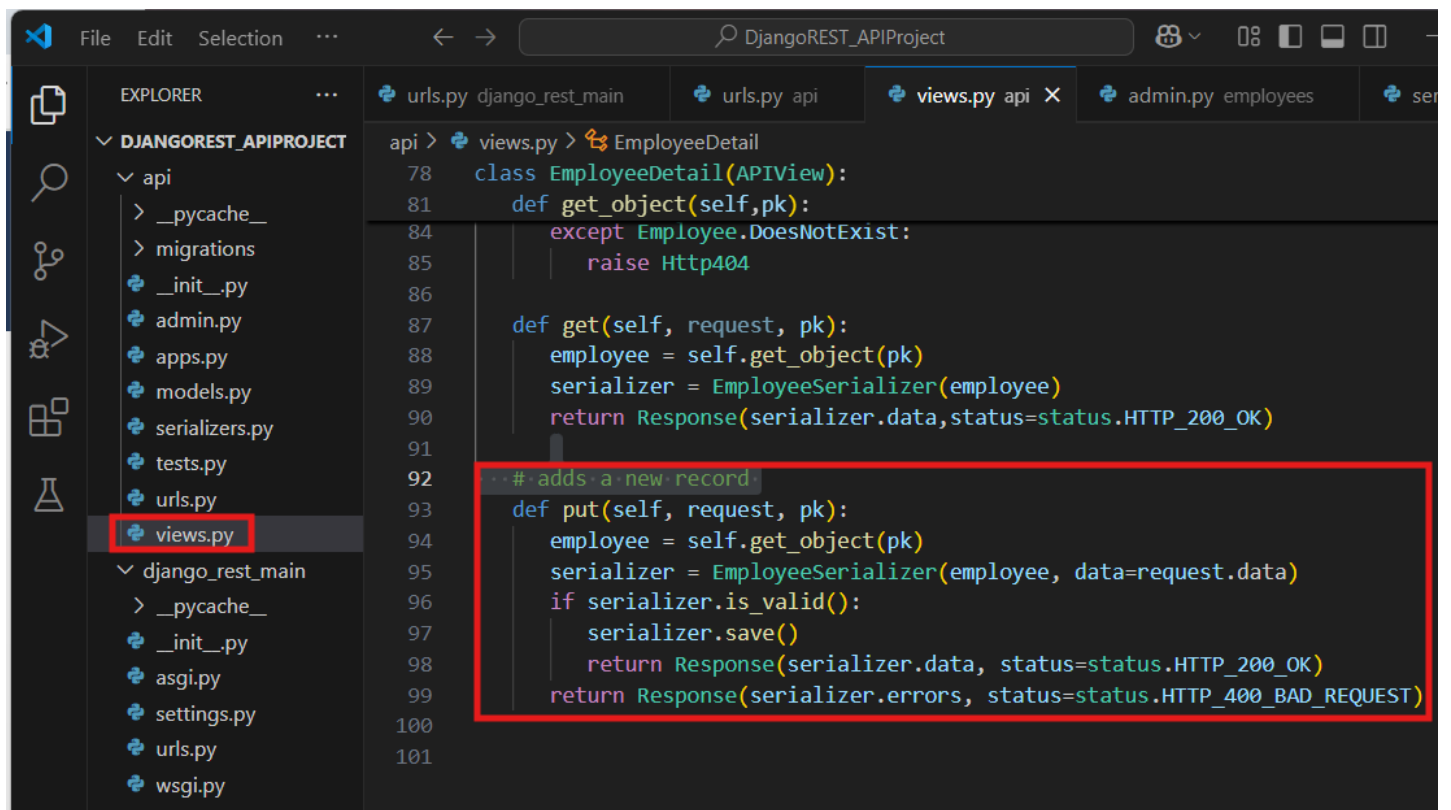
And if the record is not in the table model:

The screenshot shows a web browser window with the address bar containing `127.0.0.1:8000/api/v1/employees/30/`, where the path is highlighted with a red box. The page title is 'Django REST framework' and the user is logged in as 'api\_djangoadmin'. The breadcrumb shows 'Employees / Employee Detail'. The main heading is 'Employee Detail' with 'OPTIONS' and 'GET' buttons. Below, the request details for 'GET /api/v1/employees/30/' are shown. The response is highlighted with a red box and contains the following text: `HTTP 404 Not Found`, `Allow: GET, HEAD, OPTIONS`, `Content-Type: application/json`, `Vary: Accept`, and a JSON body: `{ "detail": "Not found." }`

5. To update a single record, update the APIVIEWS.PY as:



This creates the form immediately but this causes an error when you submit the PUT button. So update it as:



Add the new record:

Employees

# Employees

OPTIONS

GET

GET /api/v1/employees/

HTTP 200 OK

Allow: GET, POST, 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"
  },
  {
    "id": 3,
    "emp_id": "EMP003",
    "emp_name": "Russell Lim",
    "designation": "Security"
  }
]
```

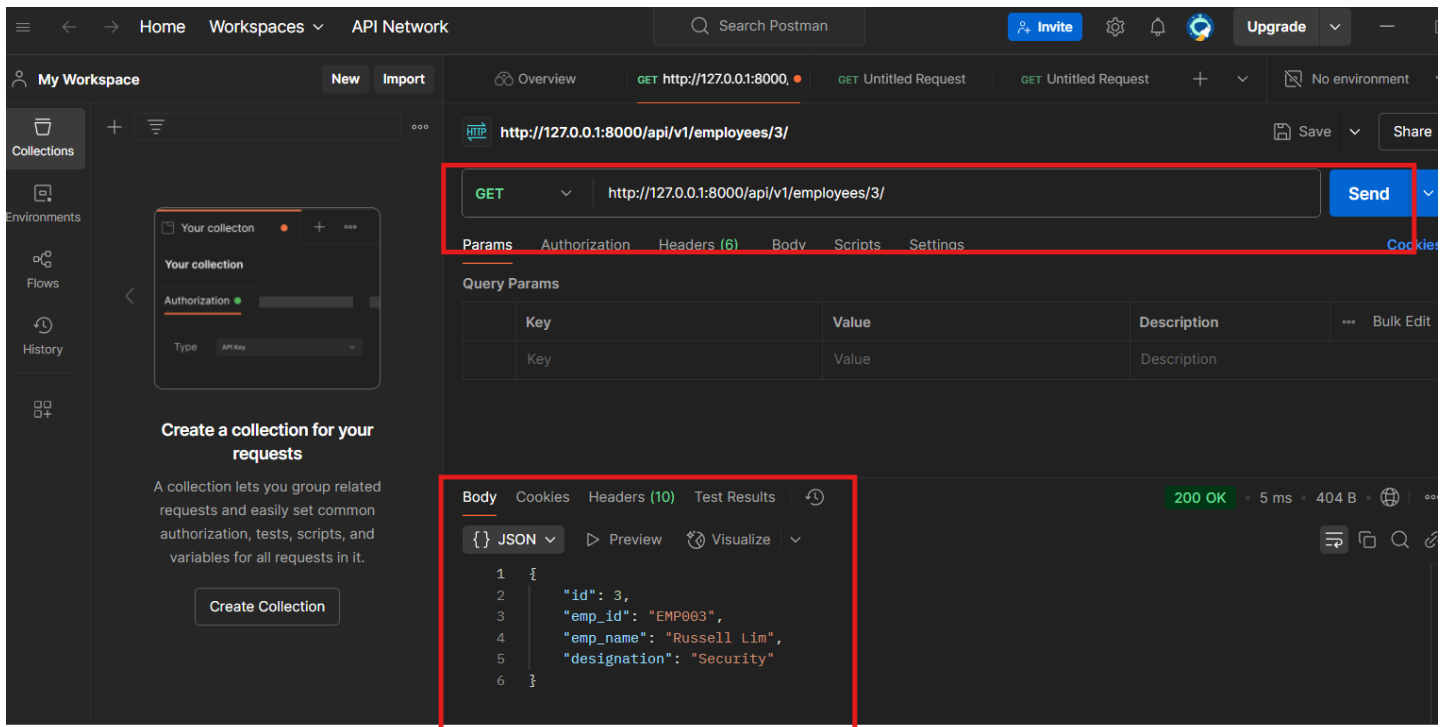
Media type:

application/json

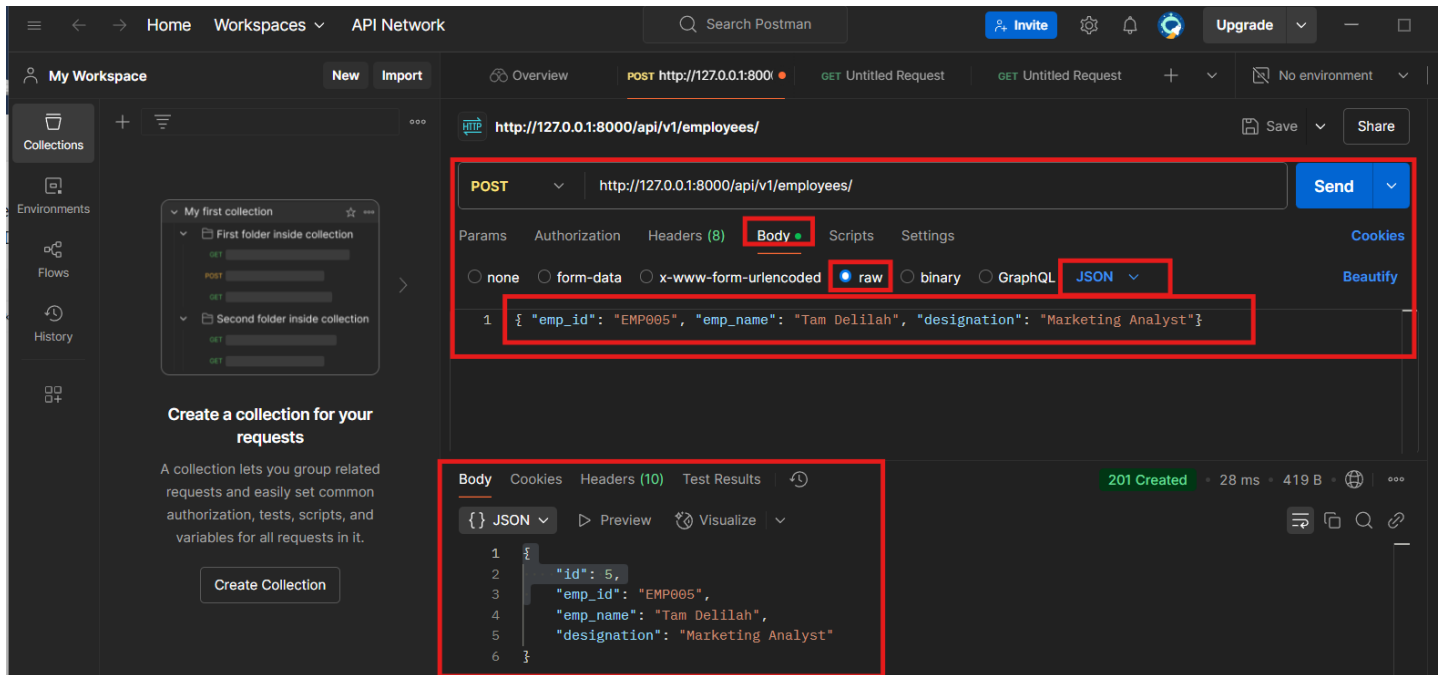
Content:

```
{
  "emp_id": "EMP004",
  "emp_name": "John Doe",
  "designation": "Ai Engineer"
}
```

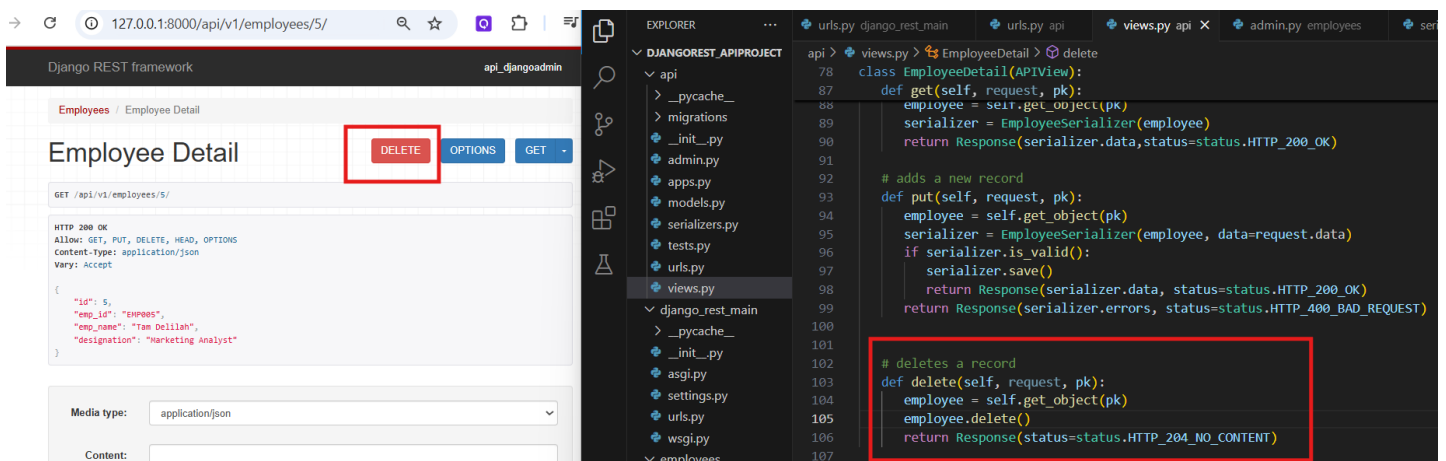
POST



Using POSTMAN to store, choose POST, then select BODY, then RAW, then JSON. Add your records then select the SEND method.



7. To delete a record, create a DELETE method and update as:



We deleted record ID = 5.

8. In POSTMAN, we can also issue a delete option:

