

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

Speaker: / Notebook: API Development using Django Framework

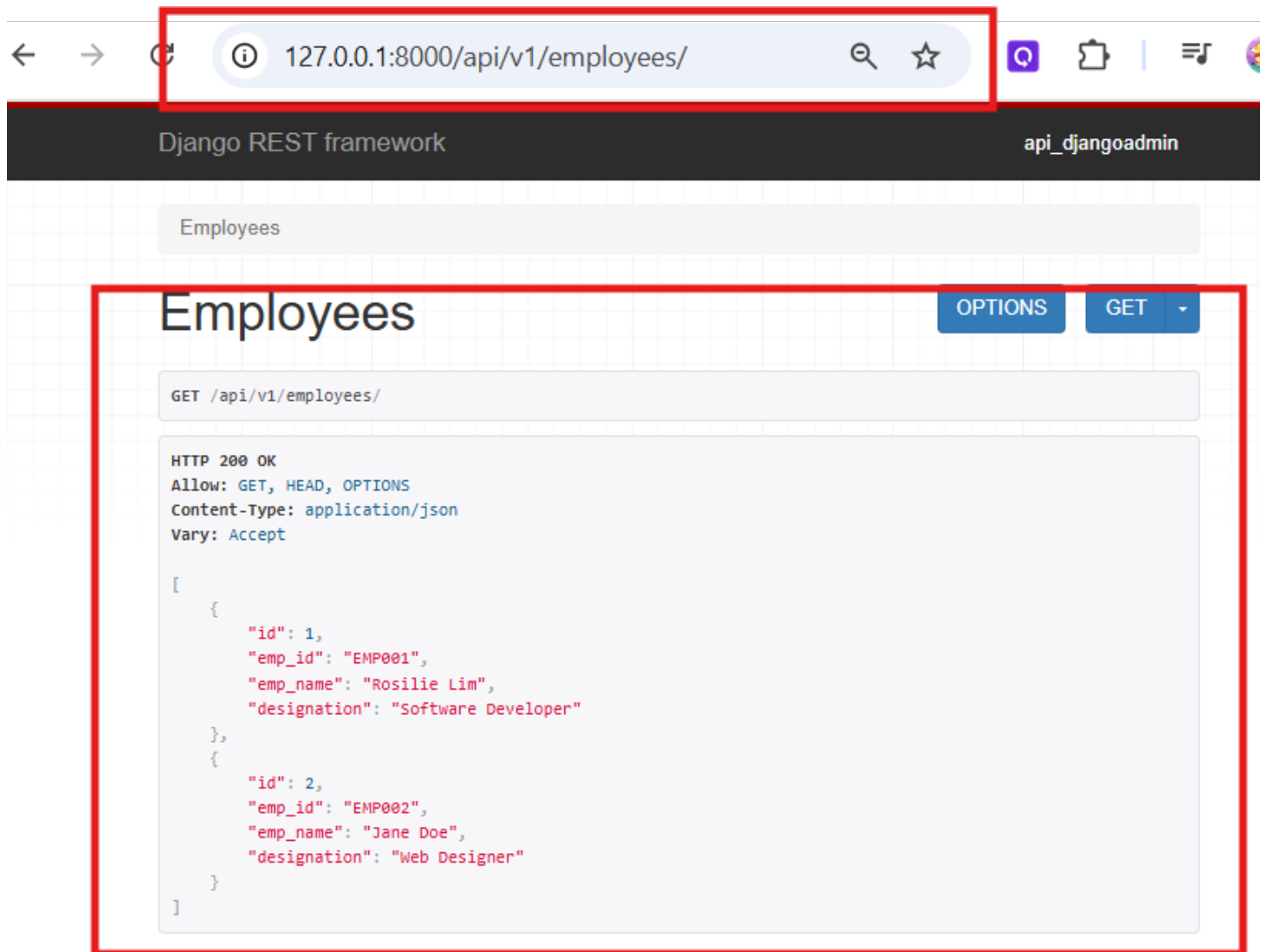


We previously created a new app called EMPLOYEES and a new Employee serializer for our Employee class.

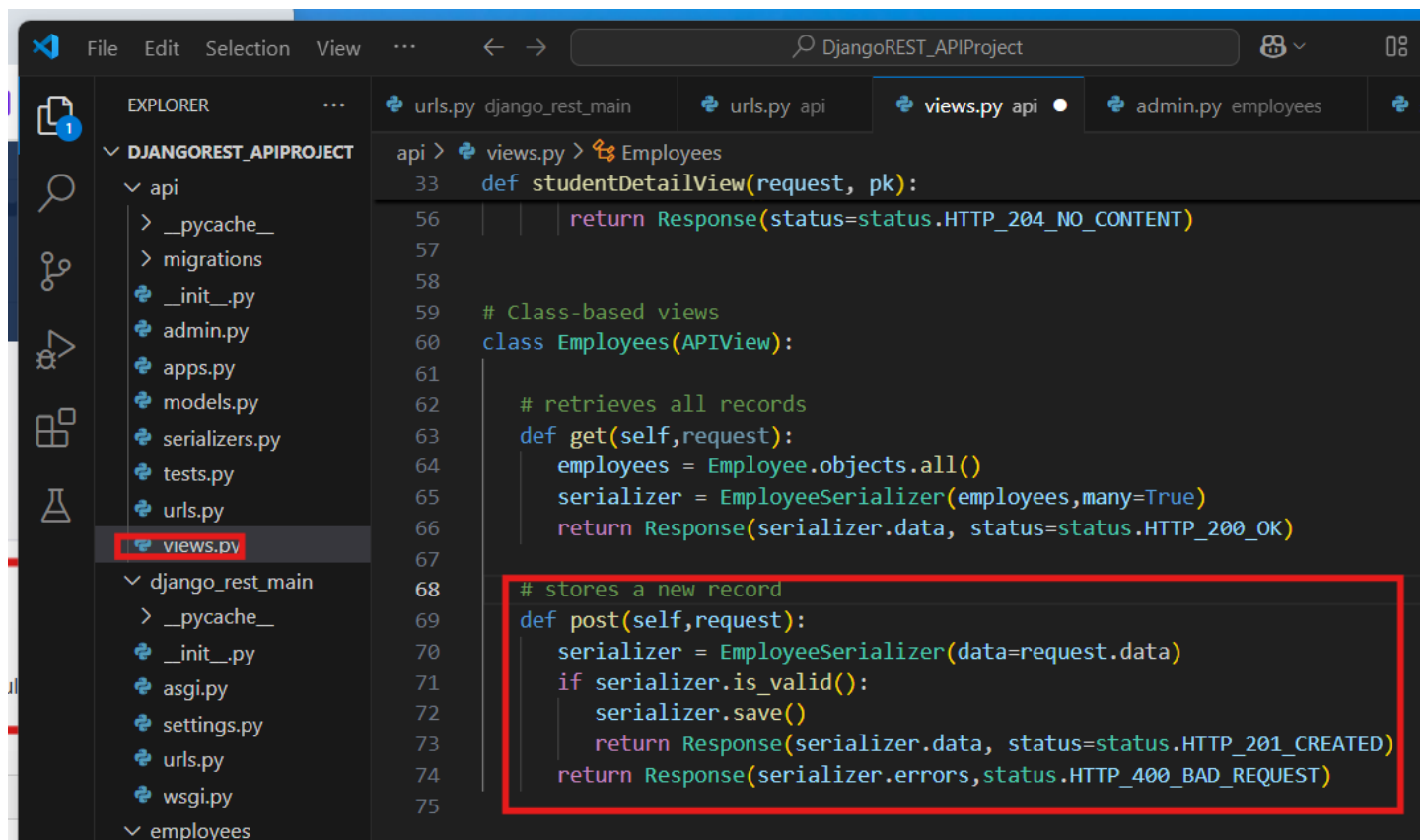
1. Previously, we updated the APIVIEWS.PY to create an EMPLOYEE class and its methods:

```
File Edit Selection View ... DjangoREST_APIProject
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.

127.0.0.1:8000/api/v1/employees/

Django REST framework

api_djangoadmin

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"
  }
]
```

Media type: application/json

Content:

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

POST

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"  
}
```

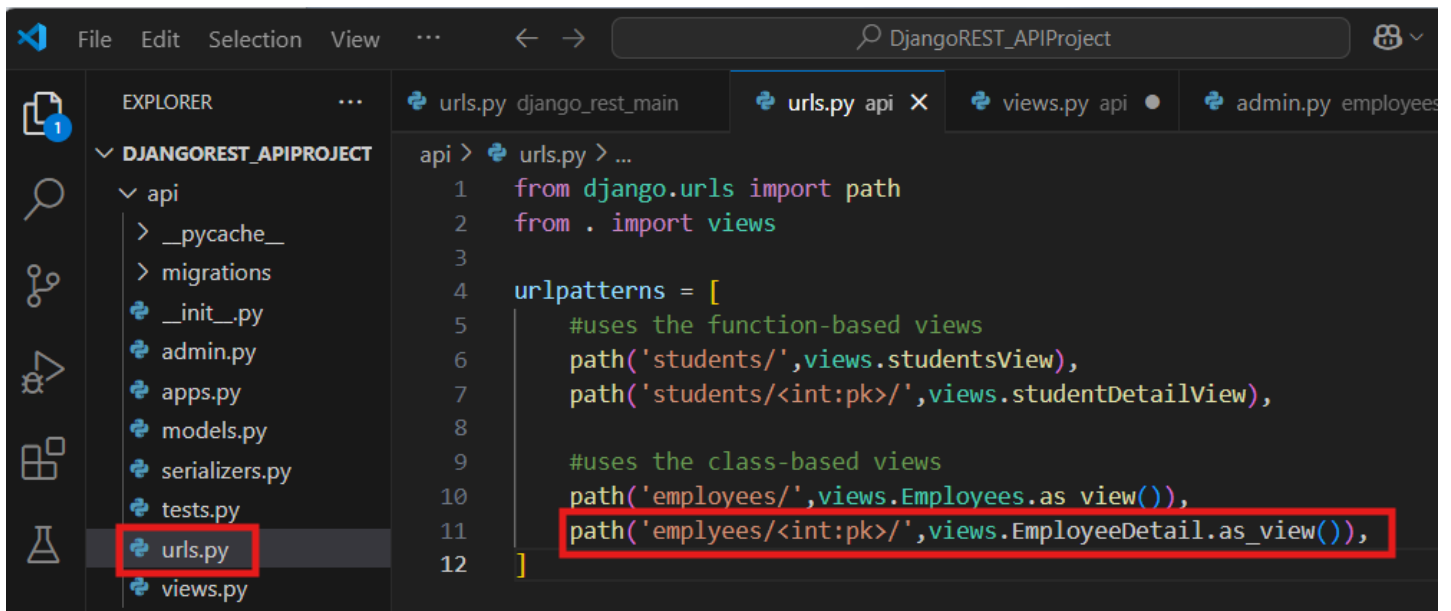
Media type:

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. On the left, the Explorer sidebar displays the project structure for 'DJANGOREST_APIPROJECT'. The 'api' directory is expanded, showing files like '__pycache__', 'migrations', '__init__.py', 'admin.py', 'apps.py', 'models.py', 'serializers.py', 'tests.py', 'urls.py', and 'views.py'. The 'urls.py' file is selected and highlighted with a red box. The main editor window shows the content of 'urls.py' for the 'api' app. The code includes imports for 'path' from 'django.urls' and 'views' from the local app. The 'urlpatterns' list contains two entries: one for 'students/' using 'views.studentsView' and another for 'employees/' using 'views.Employees.as_view()'. A third entry, 'path('employees/<int:pk>/', views.EmployeeDetail.as_view())', is highlighted with a red box. The top of the editor shows the project name 'DjangoREST_APIProject' and several open tabs: 'urls.py django_rest_main', 'urls.py api', 'views.py api', and 'admin.py employees'.

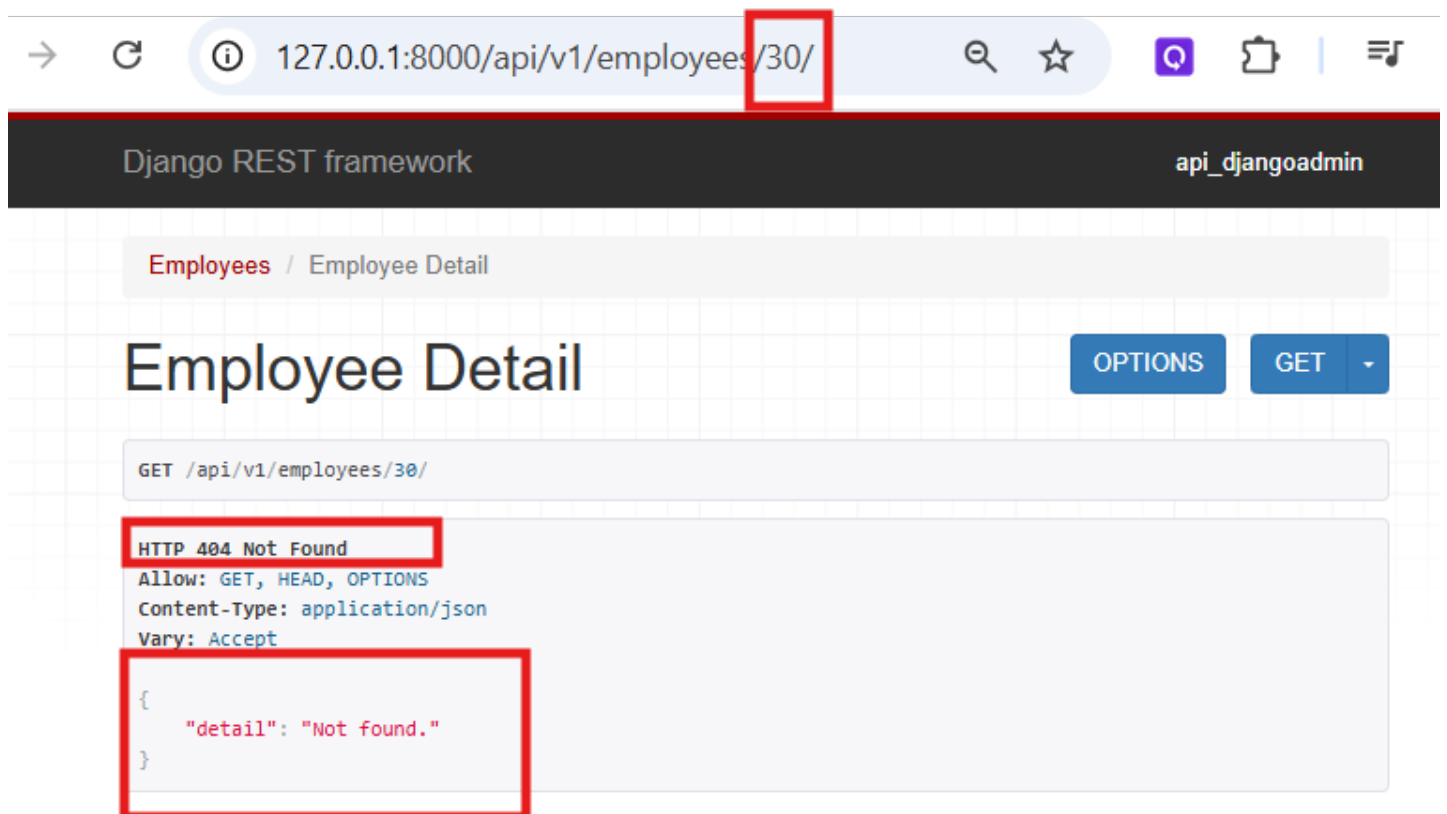
```
1 from django.urls import path
2 from . import views
3
4 urlpatterns = [
5     #uses the function-based views
6     path('students/', views.studentsView),
7     path('students/<int:pk>', views.studentDetailView),
8
9     #uses the class-based views
10    path('employees/', views.Employees.as_view()),
11    path('employees/<int:pk>', views.EmployeeDetail.as_view()),
12 ]
```

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

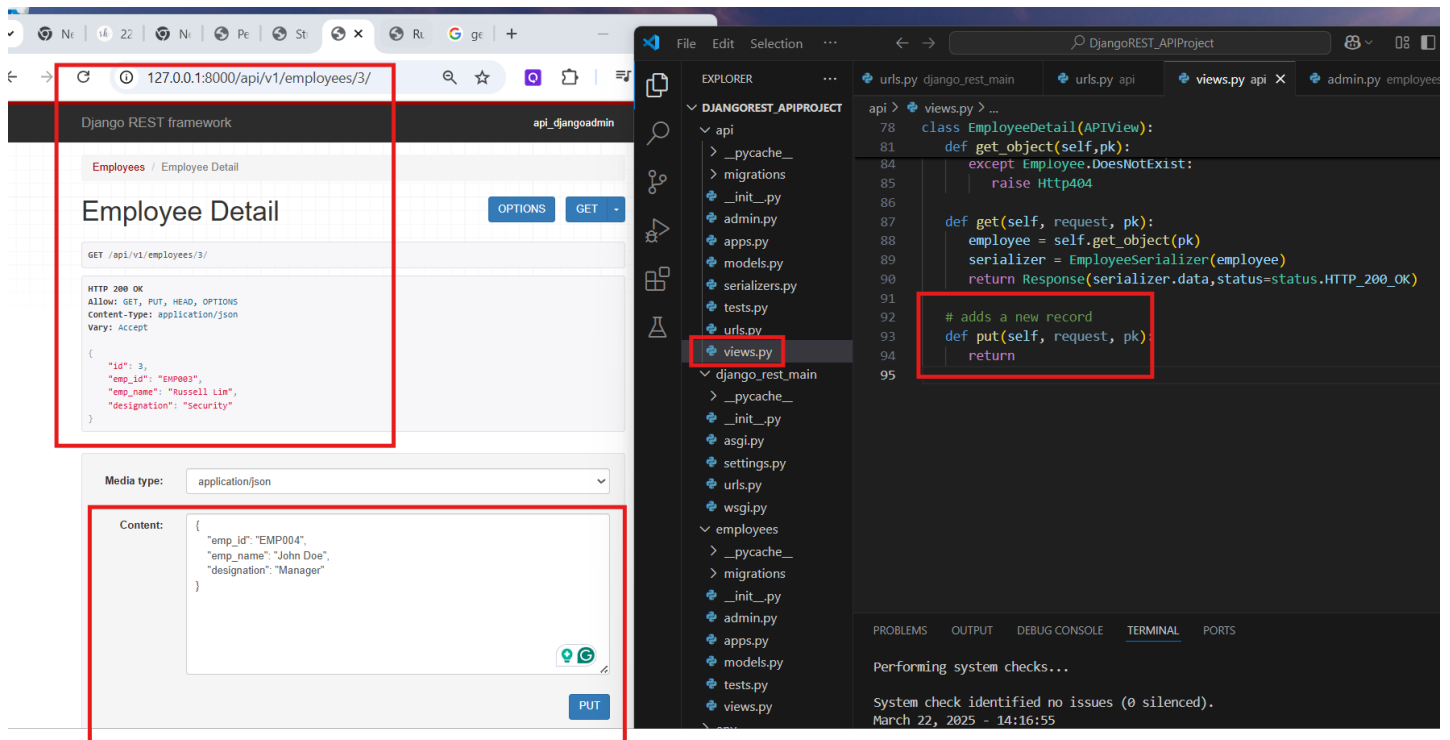
To test, add the path:

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

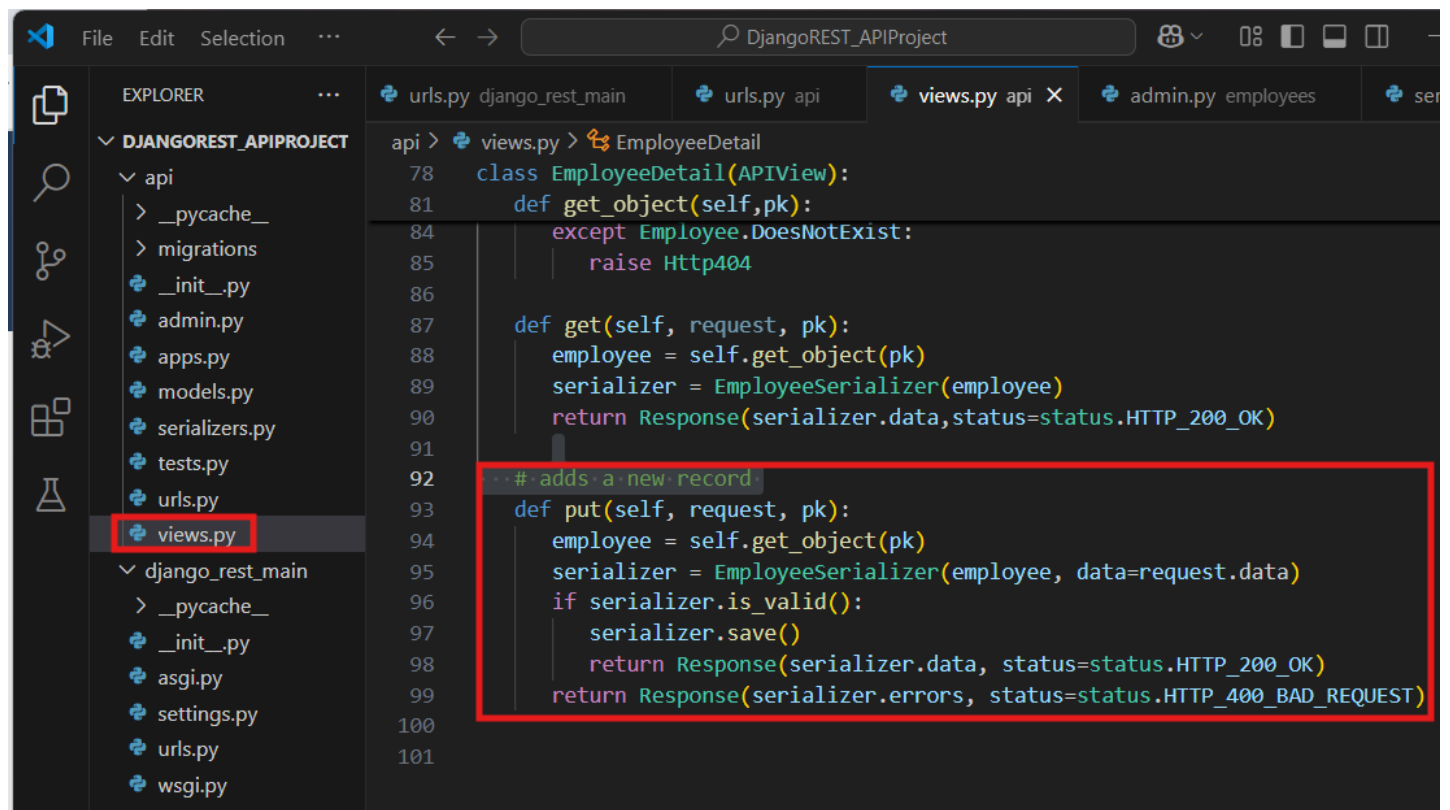
And if the record is not in the table model:



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

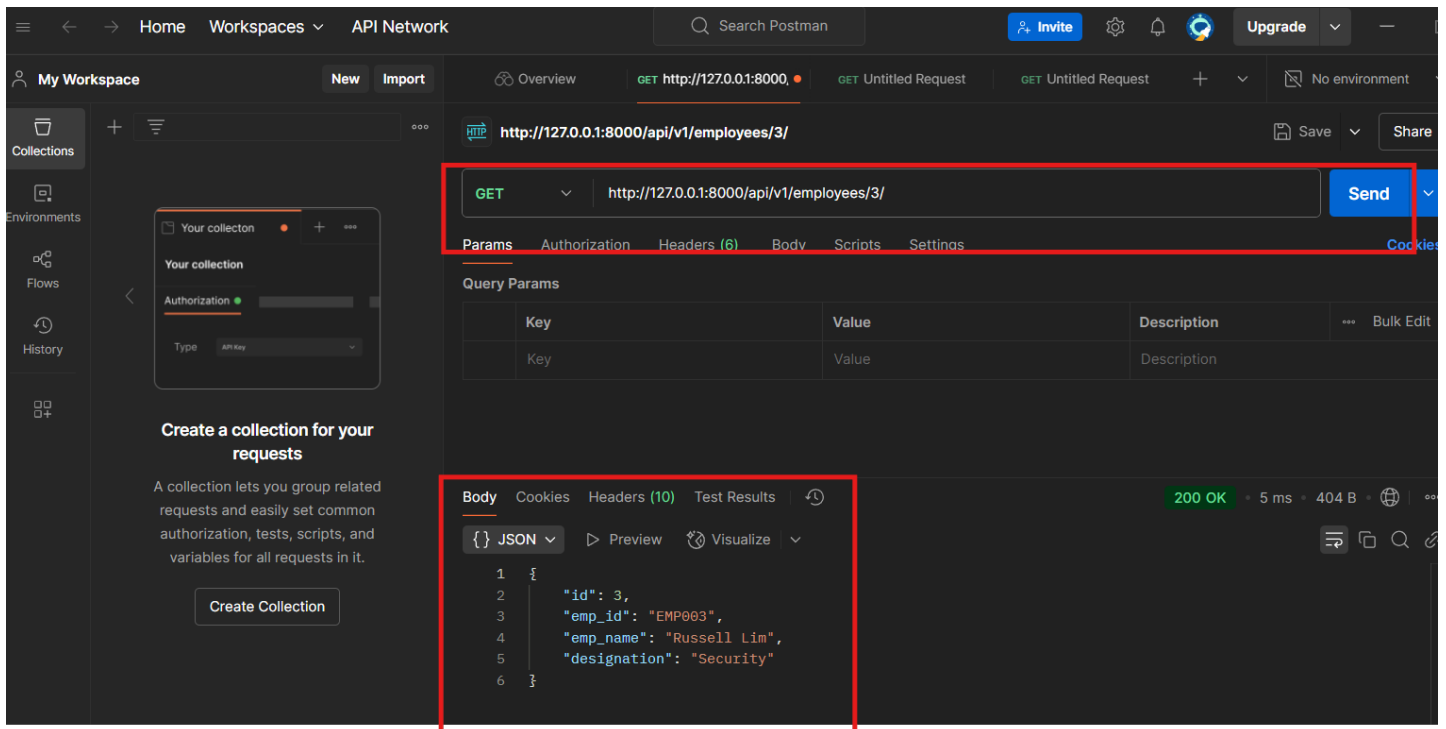
[
  {
    "id": 1,
    "emp_id": "EMP001",
    "emp_name": "Rosilie Lim",
    "designation": "Software Developer"
  },
  {
    "id": 2,
    "emp_id": "EMP002",
    "emp_name": "Jane Doe",
    "designation": "Web Designer"
  },
  {
    "id": 3,
    "emp_id": "EMP003",
    "emp_name": "Russell Lim",
    "designation": "Security"
  }
]
```

Media type: application/json

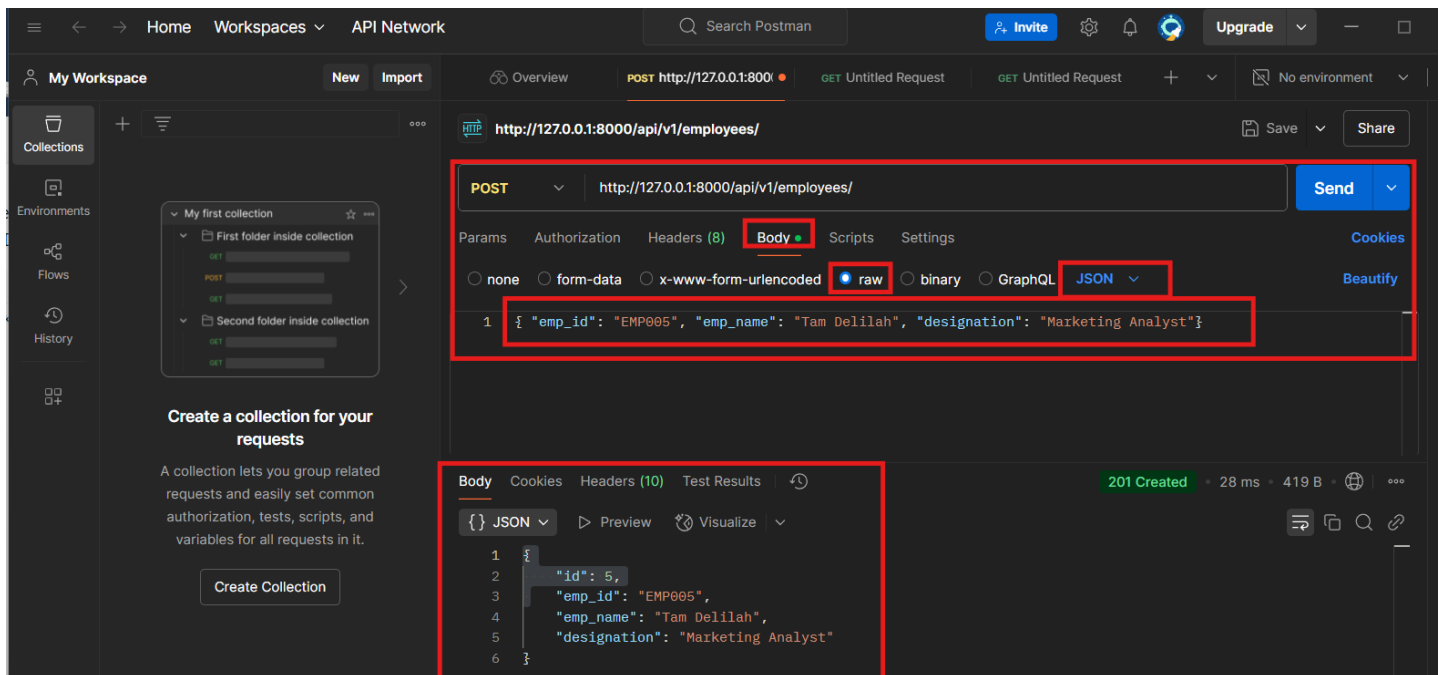
Content:

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

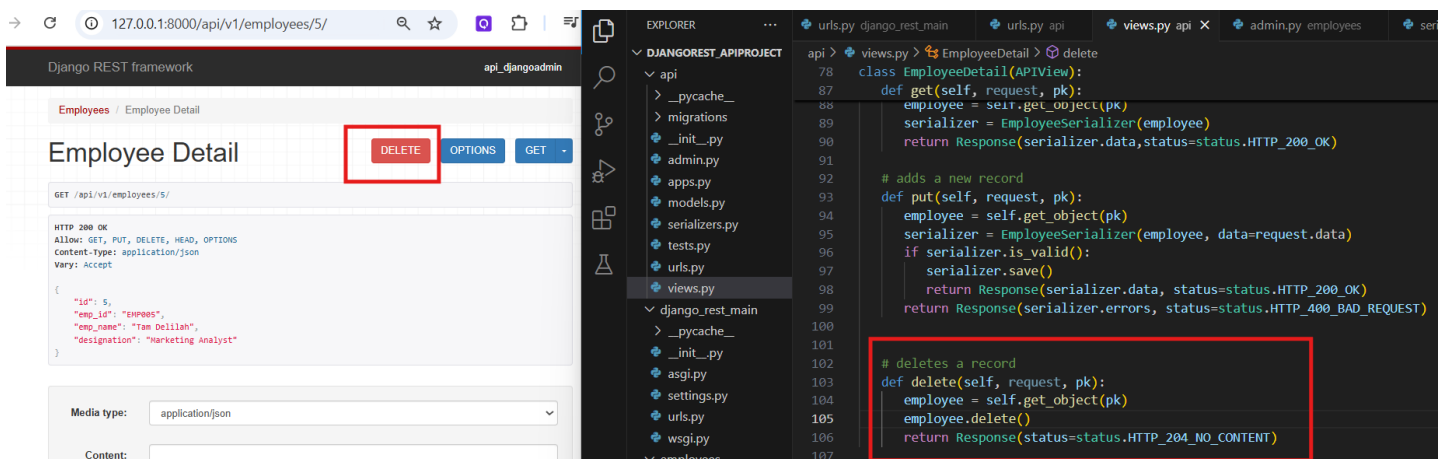
POST



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:

