

Topic: 11. CRUD using Viewsets

Speaker: Personal / Notebook: API Development using Django Framework



According to Django Documentation:

Django REST framework allows you to combine the logic for a set of related views in a single class, called a `ViewSet`. In other frameworks, you may also find conceptually similar implementations named 'Resources' or 'Controllers'.

A `ViewSet` class is simply **a type of class-based View, that does not provide any method handlers** such as `.get()` or `.post()`, and instead provides actions such as `.list()` and `.create()`.

The method handlers for a `ViewSet` are only bound to the corresponding actions at the point of finalizing the view, using the `.as_view()` method.

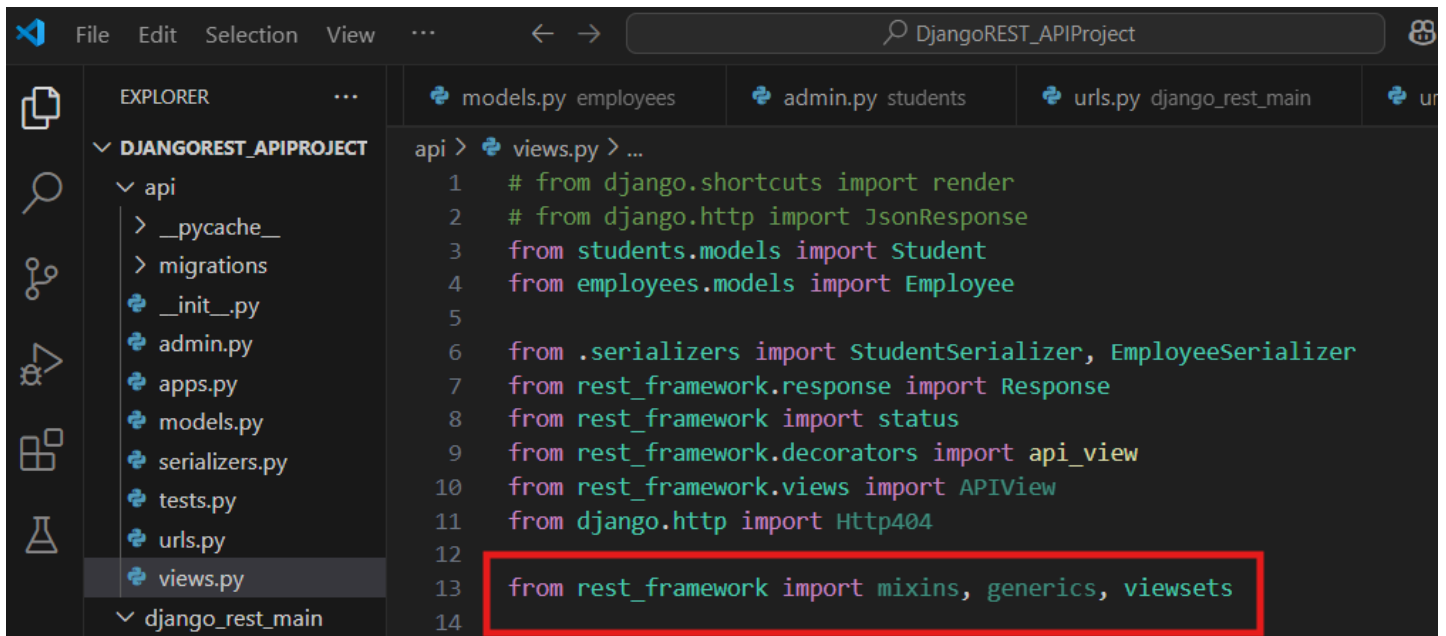
1. We use `viewsets.ViewSet` class for `list()`, `retrieve()`, `update()`, `create`, `delete()`. The `viewsets.ModelViewSet` uses only `queryset` and `serializer_class` and provides for `pk` and `non-pk`-based operations.

2. To use Viewsets, we use ROUTERS with the paths that we normally use in our Django project. So we comment on the paths we used with function-based, class-based, mixins, and generics in our `URLS.PY`.

```
File Edit Selection View Go ... 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
      + wsgi.py
      + employees
  settings.py
  models.py employees
  admin.py students
  urls.py django_rest_main

api > urls.py > ...
1  from django.urls import path, include
2  from . import views
3  from rest_framework.routers import DefaultRouter
4
5  # uses the Viewsets for CRUD
6  router = DefaultRouter()
7  router.register('employees', views.EmployeeViewSet)
8
9  urlpatterns = [
10     #uses the function-based views
11     path('students/', views.studentsView),
12     path('students/<int:pk>', views.studentDetailView),
13
14     #uses the class-based views, mixins, generics
15     # path('employees/', views.Employees.as_view()),
16     # path('employees/<int:pk>', views.EmployeeDetail.as_view()),
17
18     # uses viewsets
19     path('', include(router.urls))
20
21 ]
```

3. Update the `VIEWS.PY` as:



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

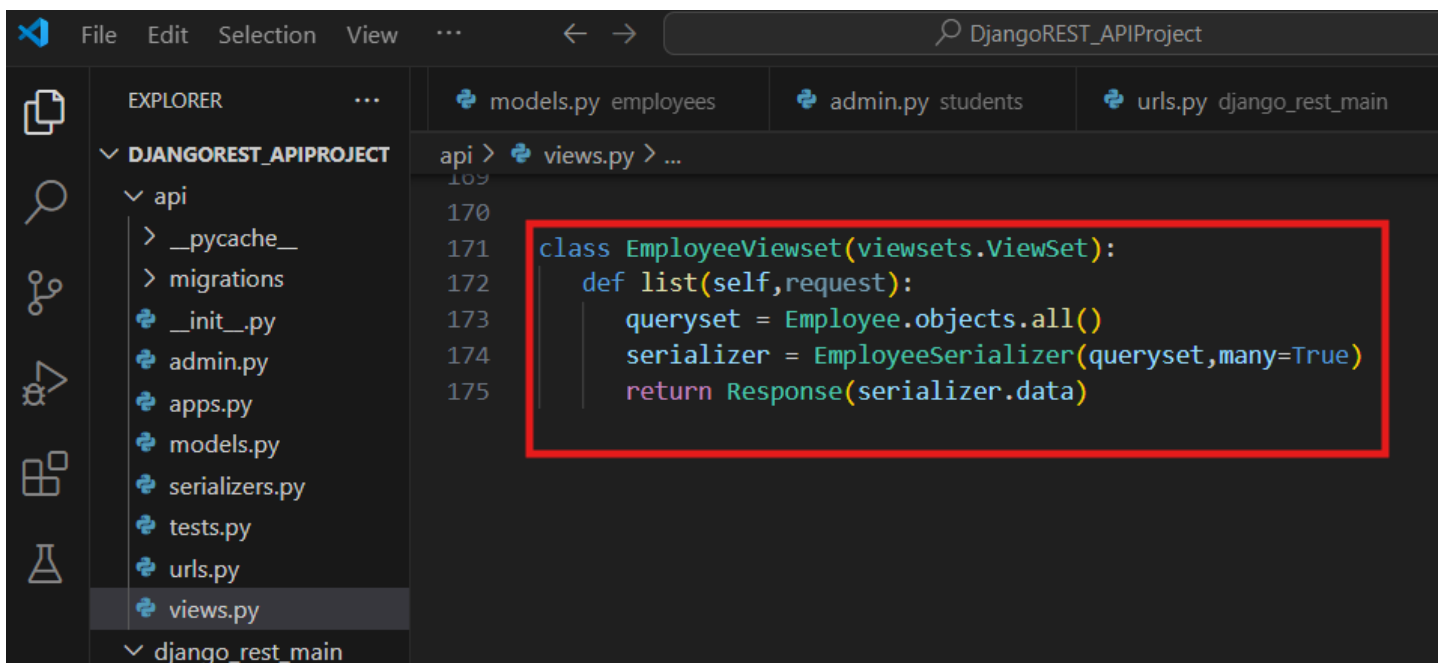
EXPLORER

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

models.py employees admin.py students urls.py django_rest_main

api > views.py > ...

```
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 from django.http import Http404
12
13 from rest_framework import mixins, generics, viewsets
14
```



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

EXPLORER

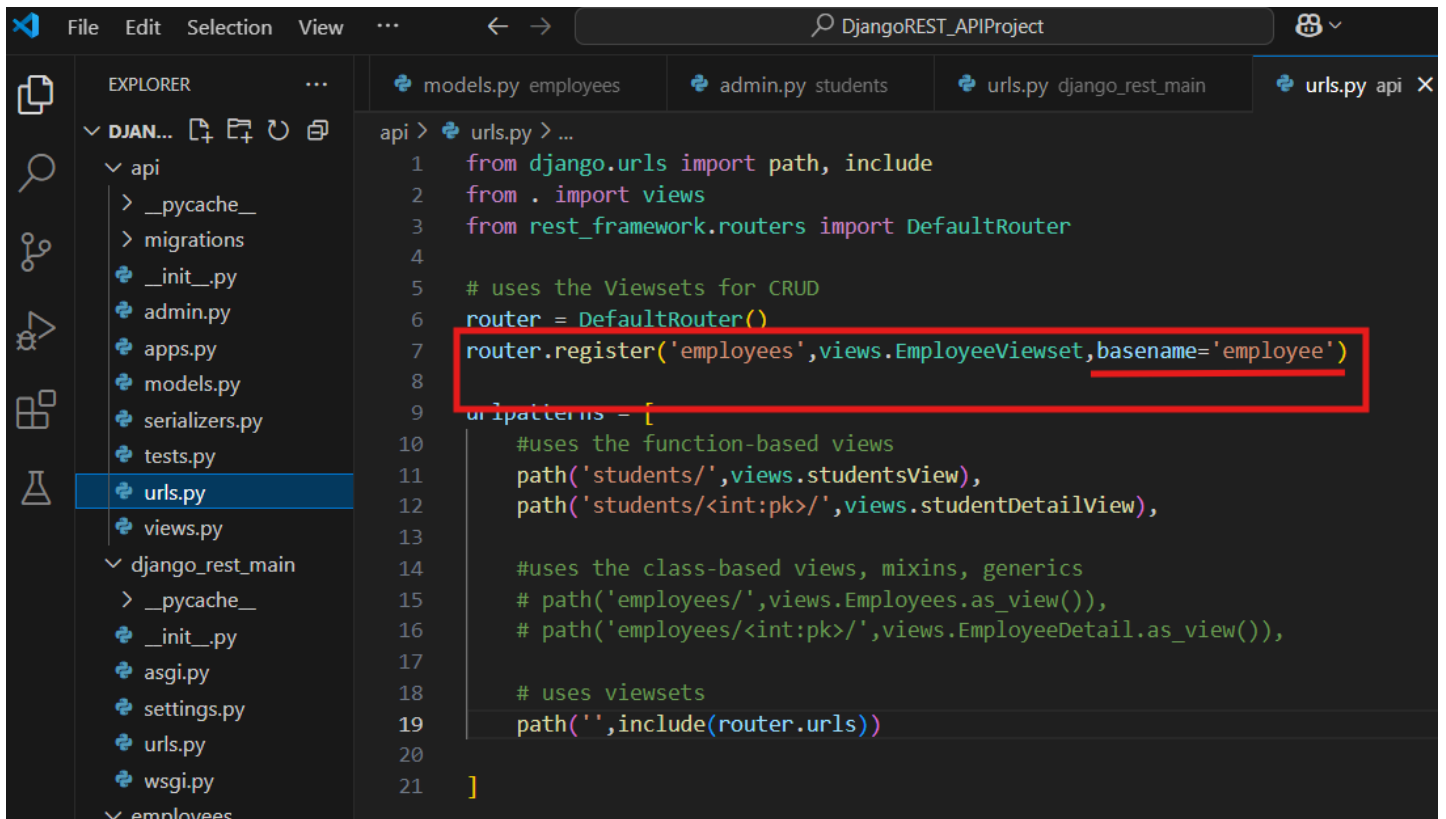
- DJANGOREST_APIPROJECT
 - api
 - __pycache__
 - migrations
 - __init__.py
 - admin.py
 - apps.py
 - models.py
 - serializers.py
 - tests.py
 - urls.py
 - views.py
 - django_rest_main

models.py employees admin.py students urls.py django_rest_main

api > views.py > ...

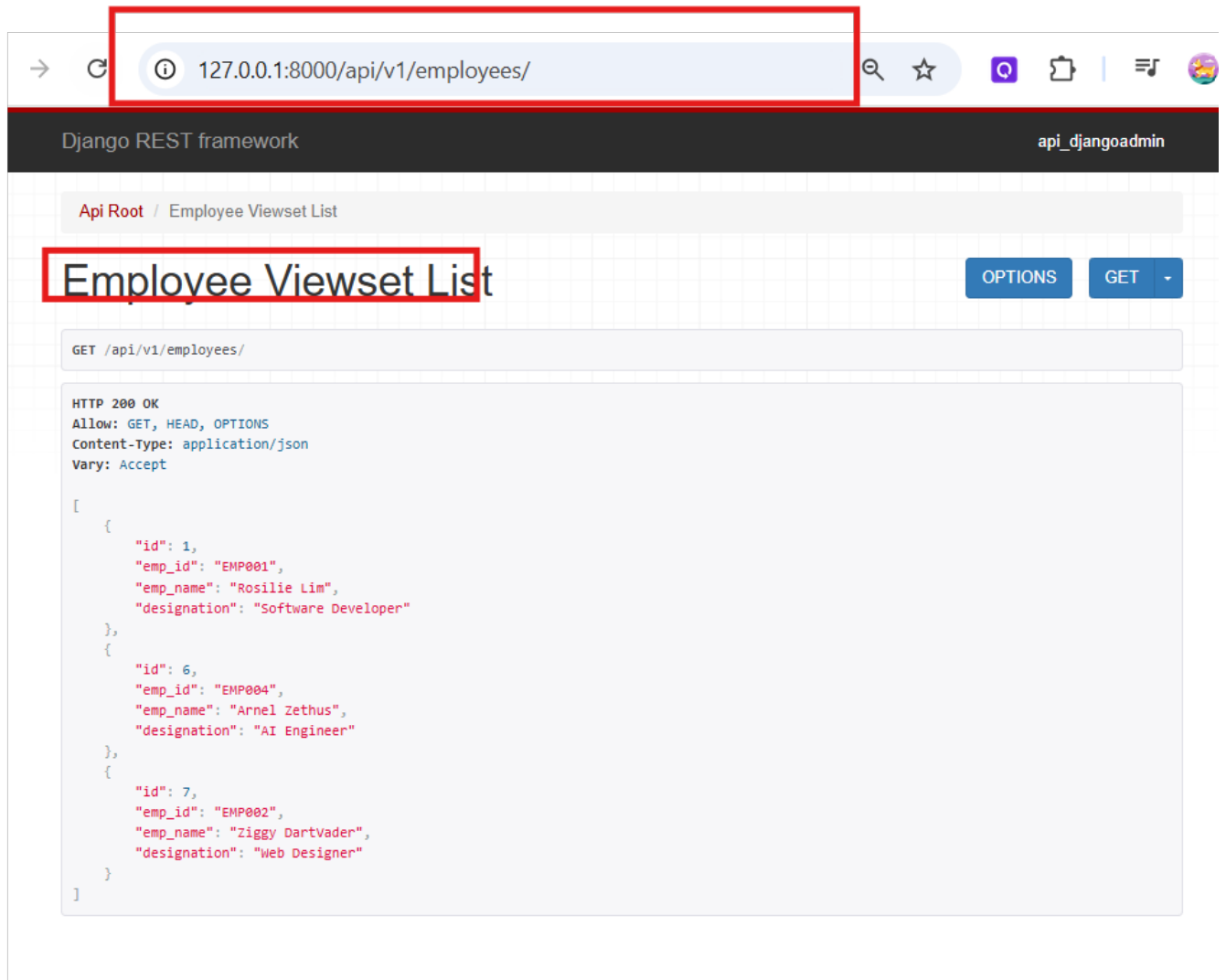
```
109
170
171 class EmployeeViewSet(viewsets.ViewSet):
172     def list(self, request):
173         queryset = Employee.objects.all()
174         serializer = EmployeeSerializer(queryset, many=True)
175         return Response(serializer.data)
```

We further update our URLS.PY to include the basename as the name of our model.



```
1 from django.urls import path, include
2 from . import views
3 from rest_framework.routers import DefaultRouter
4
5 # uses the Viewsets for CRUD
6 router = DefaultRouter()
7 router.register('employees', views.EmployeeViewSet, basename='employee')
8
9 urlpatterns = [
10     #uses the function-based views
11     path('students/', views.studentsView),
12     path('students/<int:pk>/', views.studentDetailView),
13
14     #uses the class-based views, mixins, generics
15     # path('employees/', views.Employees.as_view()),
16     # path('employees/<int:pk>/', views.EmployeeDetail.as_view()),
17
18     # uses viewsets
19     path('', include(router.urls))
20 ]
```

So, when we run our path for Employees:



127.0.0.1:8000/api/v1/employees/

Django REST framework api_djangoadmin

Api Root / Employee Viewset List

Employee Viewset List

OPTIONS GET

GET /api/v1/employees/

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

```
[
  {
    "id": 1,
    "emp_id": "EMP001",
    "emp_name": "Rosilie Lim",
    "designation": "Software Developer"
  },
  {
    "id": 6,
    "emp_id": "EMP004",
    "emp_name": "Arnel Zethus",
    "designation": "AI Engineer"
  },
  {
    "id": 7,
    "emp_id": "EMP002",
    "emp_name": "Ziggy DartVader",
    "designation": "Web Designer"
  }
]
```

4. For other CRUD operations, we create a new method, CREATE, to add a new record:

127.0.0.1:8000/api/v1/employees/

Django REST framework

Api Root / Employee Viewset List

Employee Viewset List

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 Lin",
  "designation": "Software Developer"
},
{
  "id": 6,
  "emp_id": "EMP004",
  "emp_name": "Arnel Zethus",
  "designation": "AI Engineer"
},
{
  "id": 7,
  "emp_id": "EMP002",
  "emp_name": "Ziggy Dertvader",
  "designation": "Web Designer"
}
```

Media type: application/json

Content:

```
{
  "emp_id": "EMP007",
  "emp_name": "Dylan Cheese",
  "designation": "Product Manager"
}
```

api_djangoadmin

EXPLORER

DJANGO REST API PROJECT

- 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
 - wsgi.py
- employees
 - > _pycache__
 - > migrations
 - > _init_.py
 - admin.py
 - apps.py
 - models.py
 - tests.py

models.py employees admin.py students urls.py django_rest_main urls.py api

api > views.py > EmployeeViewSet > create

```
167 lookup_field = 'pk'
168
169 """
170
171 # uses Viewsets
172 class EmployeeViewSet(viewsets.ViewSet):
173
174     def list(self, request):
175         queryset = Employee.objects.all()
176         serializer = EmployeeSerializer(queryset, many=True)
177         return Response(serializer.data)
178
179     def create(self, request):
180         serializer = EmployeeSerializer(data=request.data)
181         if serializer.is_valid():
182             serializer.save()
183             return Response(serializer.data, status=status.HTTP_201_CREATED)
184             return Response(serializer.errors)
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

Performing system checks...

System check identified no issues (0 silenced).

March 31, 2025 - 18:02:46

This results to:

127.0.0.1:8000/api/v1/employees/

Django REST framework

api_djangoadmin

Api Root / Employee Viewset List

Employee Viewset List

OPTIONS GET

POST /api/v1/employees/

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

```
{
  "id": 9,
  "emp_id": "EMP007",
  "emp_name": "Dylan Cheese",
  "designation": "Product Manager"
}
```

Media type: application/json

Content:

POST

5. To retrieve a single record, we don't have to create a separate path for our employee details, by using the retrieve method, we can get this record immediately.

The image shows a web browser window and a VS Code editor. The browser window displays the URL `127.0.0.1:8000/api/v1/employees/9/` and the response for the `GET /api/v1/employees/9/` endpoint. The response is a JSON object representing an employee:

```
{
  "id": 9,
  "emp_id": "EMP007",
  "emp_name": "Dylan Cheese",
  "designation": "Product Manager"
}
```

The VS Code editor shows the `views.py` file in the `api` directory. The code defines the `EmployeeViewSet` class, which inherits from `viewsets.ViewSet`. The `retrieve` method is highlighted with a red box, showing the implementation for retrieving a single employee record:

```
def retrieve(self, request, pk=None):
    employee = get_object_or_404(Employee, pk=pk)
    serializer = EmployeeSerializer(employee)
    return Response(serializer.data, status=status.HTTP_200_OK)
```

Our URLS.PY is still this:

```
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 > urls.py > ...
1 from django.urls import path, include
2 from . import views
3 from rest_framework.routers import DefaultRouter
4
5 # uses the Viewsets for CRUD
6 router = DefaultRouter()
7 router.register('employees', views.EmployeeViewSet, basename='employee')
8
9 urlpatterns = [
10     #uses the function-based views
11     path('students/', views.studentsView),
12     path('students/<int:pk>/', views.studentDetailView),
13
14     #uses the class-based views, mixins, generics
15     # path('employees/', views.Employees.as_view()),
16     # path('employees/<int:pk>/', views.EmployeeDetail.as_view()),
17
18     # uses viewsets
19     path('', include(router.urls))
20 ]
```

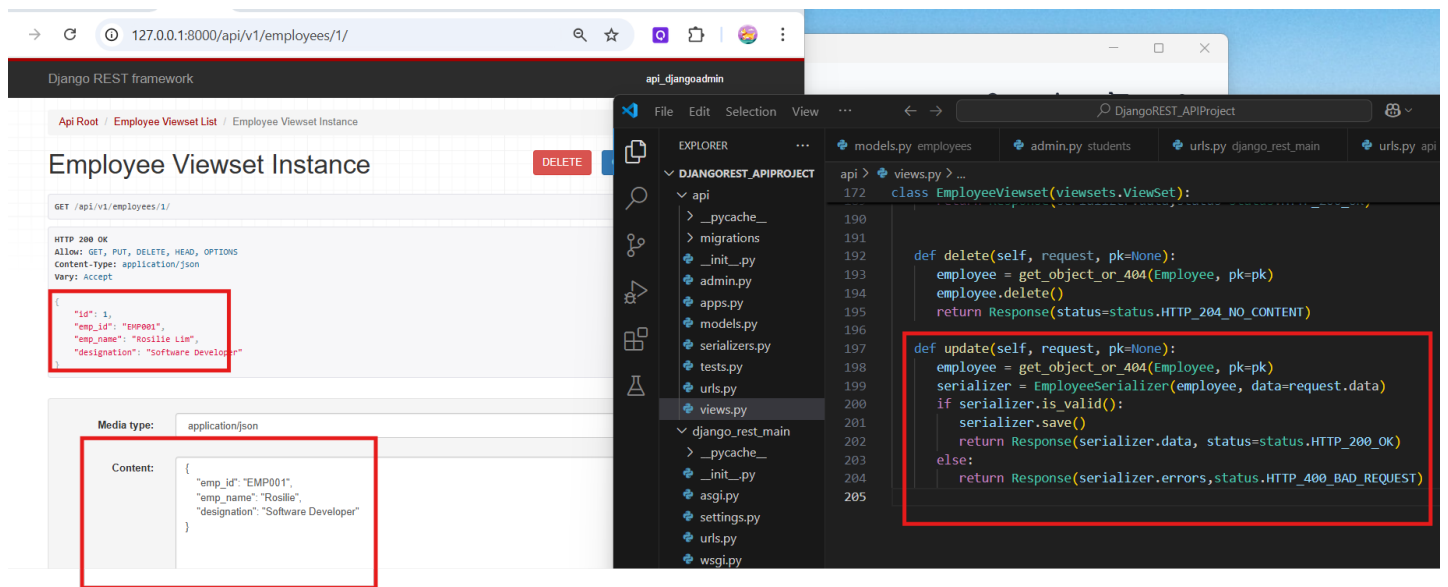
6. To delete a record, we create a new method:

```
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 > EmployeeViewSet > delete
172 class EmployeeViewSet(viewsets.ViewSet):
186     def retrieve(self, request, pk=None):
187         employee = get_object_or_404(Employee, pk=pk)
188         serializer = EmployeeSerializer(employee)
189         return Response(serializer.data, status=status.HTTP_200_OK)
190
191
192     def delete(self, request, pk=None):
193         employee = get_object_or_404(Employee, pk=pk)
194         employee.delete()
195         return Response(status=status.HTTP_204_NO_CONTENT)
```

7. To update the record,



8. Our updated Employee model shall look like this:

Employee Viewset List

DELETE

OPTIONS

GET ▾

GET /api/v1/employees/

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

```
[
  {
    "id": 1,
    "emp_id": "EMP001",
    "emp_name": "Rosilie",
    "designation": "Software Developer"
  },
  {
    "id": 6,
    "emp_id": "EMP004",
    "emp_name": "Arnel Zethus",
    "designation": "AI Engineer"
  },
  {
    "id": 7,
    "emp_id": "EMP002",
    "emp_name": "Ziggy Dartvader",
    "designation": "Web Designer"
  }
]
```

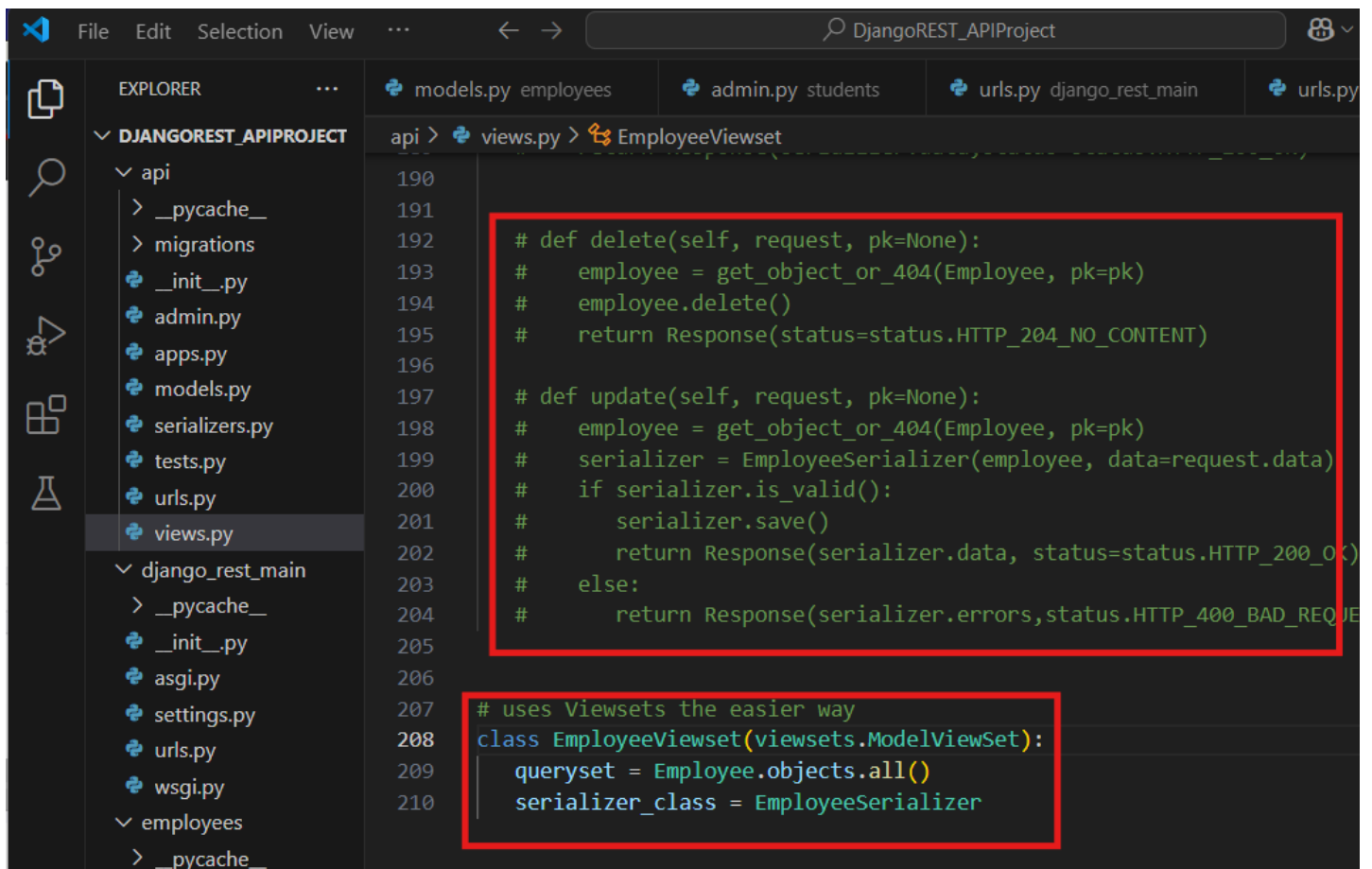
Media type:

application/json ▾

Content:

POST

Using the steps above, we were able to create one class with several methods to perform CRUD operations. However, there is an easier alternative, the use of `viewsets.ModelViewSet`, which will handle all the CRUD operations an easier and more efficient way.



The screenshot shows a code editor with the following structure:

- 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`
 - `wsgi.py`
 - employees**
 - `__pycache__`

The main editor shows the `EmployeeViewSet` class and its methods:

```
190  
191  
192 # def delete(self, request, pk=None):  
193 #     employee = get_object_or_404(Employee, pk=pk)  
194 #     employee.delete()  
195 #     return Response(status=status.HTTP_204_NO_CONTENT)  
196  
197 # def update(self, request, pk=None):  
198 #     employee = get_object_or_404(Employee, pk=pk)  
199 #     serializer = EmployeeSerializer(employee, data=request.data)  
200 #     if serializer.is_valid():  
201 #         serializer.save()  
202 #         return Response(serializer.data, status=status.HTTP_200_OK)  
203 #     else:  
204 #         return Response(serializer.errors, status.HTTP_400_BAD_REQUEST)  
205  
206  
207 # uses Viewsets the easier way  
208 class EmployeeViewSet(viewsets.ModelViewSet):  
209     queryset = Employee.objects.all()  
210     serializer_class = EmployeeSerializer
```

Two sections of code are highlighted with red boxes:

- The `delete` and `update` methods (lines 192-204).
- The `EmployeeViewSet` class definition (lines 208-210).

To add a new record:

→ ↺ ⓘ 127.0.0.1:8000/api/v1/employees/ 🔍 ☆ 📄 🗑️ 📑 🌐

Django REST framework api_django admin

Api Root / Employee Viewset List

Employee Viewset List

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",
    "designation": "Software Developer"
  },
  {
    "id": 6,
    "emp_id": "EMP004",
    "emp_name": "Arnel Zethus",
    "designation": "AI Engineer"
  },
  {
    "id": 7,
    "emp_id": "EMP002",
    "emp_name": "Ziggy DartVader",
    "designation": "Web Designer"
  }
]
```

Raw data

HTML form

Emp id

EMP003

Emp name

Russell

Designation

Security

POST

To update a record:

Api Root / Employee Viewset List / Employee Viewset Instance

Employee Viewset Instance

DELETE

OPTIONS

GET ▾

GET /api/v1/employees/10/

HTTP 200 OK

Allow: GET, PUT, PATCH, DELETE, HEAD, OPTIONS

Content-Type: application/json

Vary: Accept

```
{
  "id": 10,
  "emp_id": "EMP003",
  "emp_name": "Russell",
  "designation": "Security"
}
```

Raw data

HTML form

Emp id

EMP003

Emp name

Russell

Designation

Security Officer

PUT

To delete a record:

→ ↻ ⓘ 127.0.0.1:8000/api/v1/employees/10/ 🔍 ☆

Django REST framework api_djangoadmin

Api Root / Employee Vi

Employee

PUT /api/v1/employees/10/ OPTIONS GET ▾

Are you sure you want to delete this Employee Viewset Instance?
Cancel Delete

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

```
{  "id": 10,  "emp_id": "EMP003",  "emp_name": "Russell",  "designation": "Security Officer"}
```

Raw data HTML form

Emp id	EMP003
Emp name	Russell
Designation	Security Officer

PUT

Viewing our updated list:

Api Root / Employee Viewset List

Employee Viewset List

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",
    "designation": "Software Developer"
  },
  {
    "id": 6,
    "emp_id": "EMP004",
    "emp_name": "Arnel Zethus",
    "designation": "AI Engineer"
  },
  {
    "id": 7,
    "emp_id": "EMP002",
    "emp_name": "Ziggy Dartvader",
    "designation": "Web Designer"
  }
]
```

Raw data

HTML form

Emp id

Emp name

Designation

POST

To view a non-existent record, it also shows validations:

→ 127.0.0.1:8000/api/v1/employees/50/ 🔍 ☆ Q 📄 | 📁 🌐

Django REST framework api_djangoadmin

Api Root / Employee Viewset List / Employee Viewset Instance

Employee Viewset Instance

DELETE OPTIONS GET ▾

GET /api/v1/employees/50/

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

```
{  
  "detail": "No Employee matches the given query."  
}
```

Raw data HTML form

Emp id

Emp name

Designation

PUT

9. In summary, the `viewsets.ModelViewSet` is much faster than other methods.