

Topic: 13. Nested Serializers for Related Models Part 2

Speaker: Personal / Notebook: API Development using Django Framework



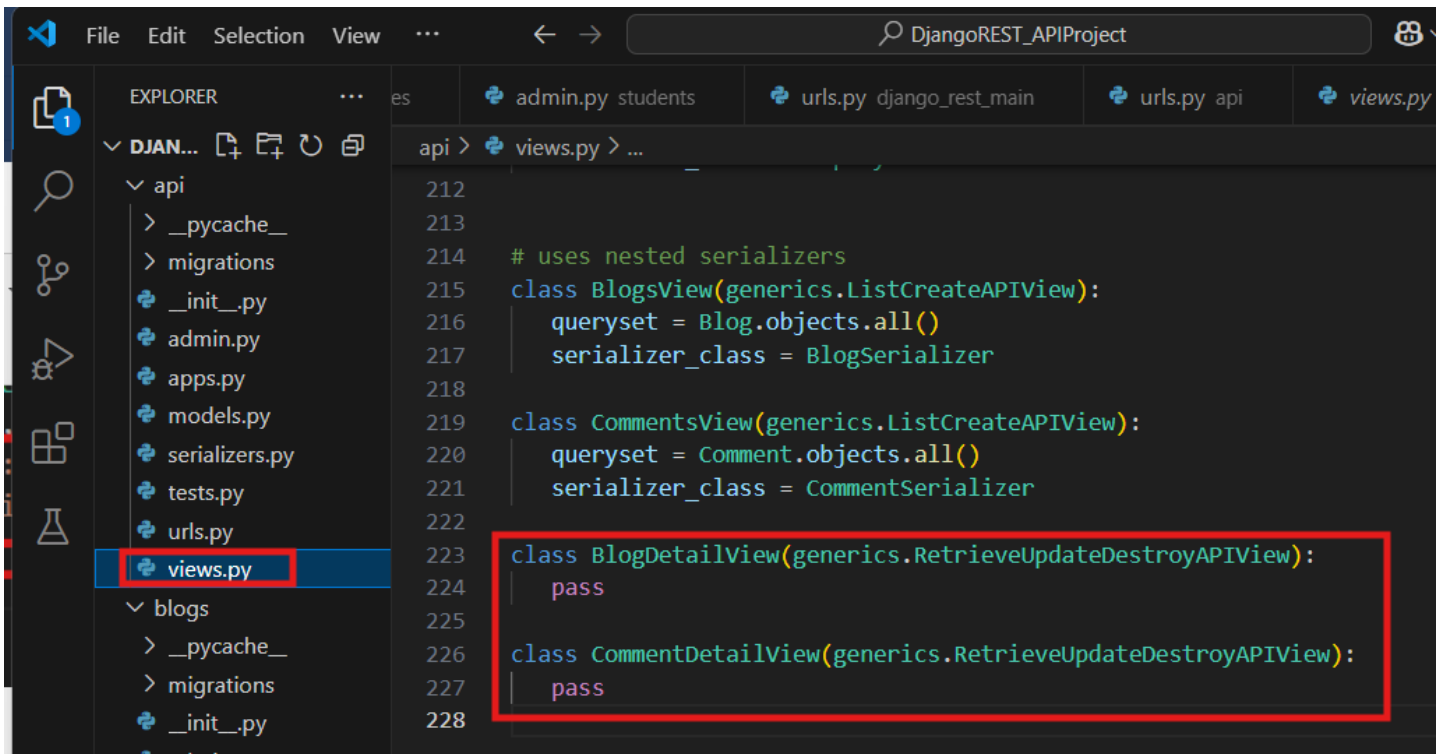
In this post, we use the primary key to allow other CRUD operations.

1. So we update the APIURLS.PY to include the PK-based operations.

```
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
    blogs
      > __pycache__
      > migrations
      > __init__.py
      > admin.py
      > apps.py
      > models.py
      > serializers.py
      > tests.py
      > views.py
  es
  admin.py students
  urls.py django_rest_main
  urls.py api X
  views.py

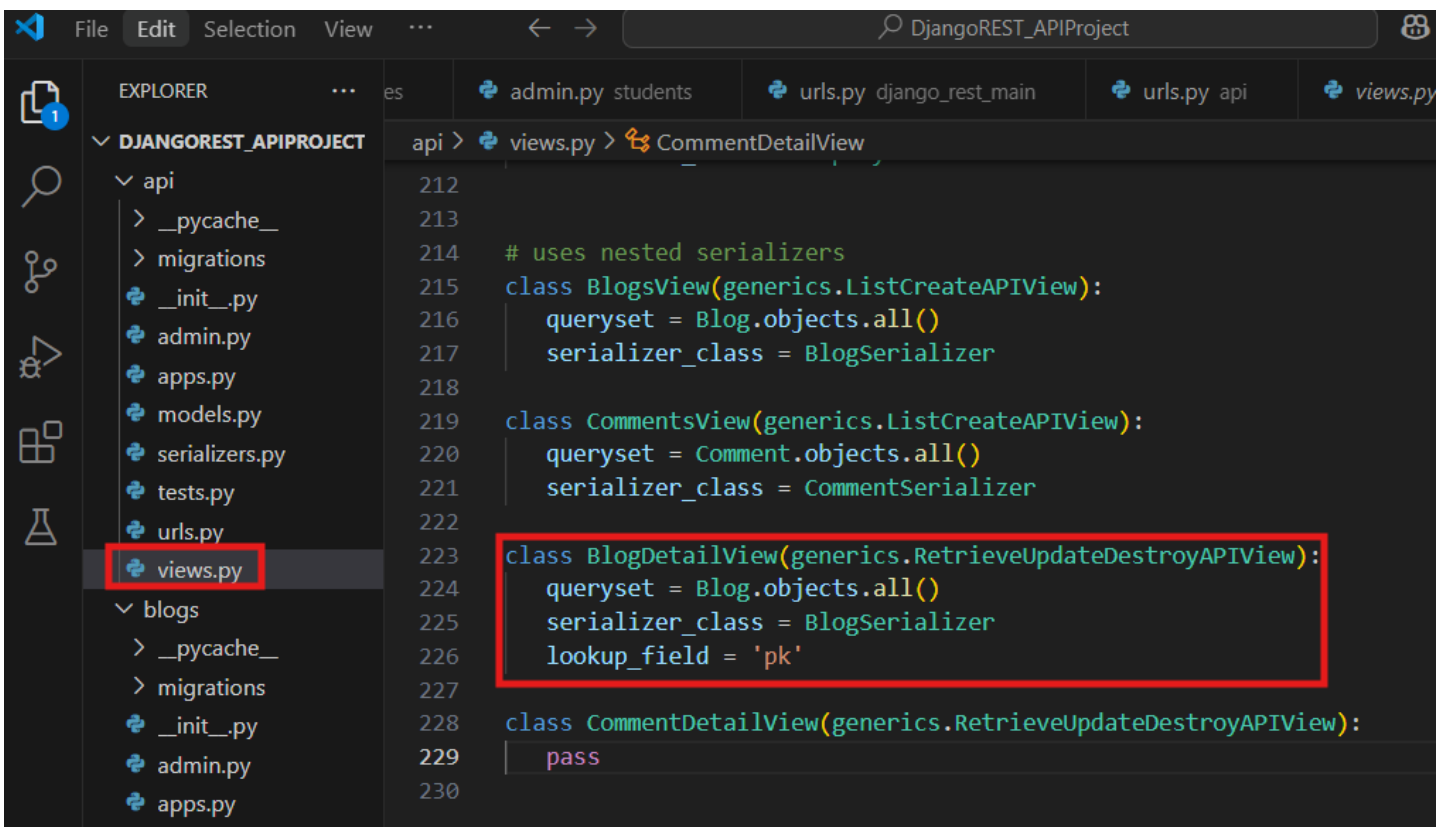
api > urls.py > ...
6 router = DefaultRouter()
7 router.register('employees', views.EmployeeViewSet, basename='employee')
8
9 urlpatterns = [
10     #uses the function-based views
11     path('students/', views.studentsView),
12     path('students/<int:pk>', views.studentDetailView),
13
14     #uses 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
21     # uses nested serializers
22     path('blogs/', views.BlogsView.as_view()),
23     path('comments/', views.CommentsView.as_view()),
24     path('blogs/<int:pk>', views BlogDetailView.as_view()),
25     path('comments/<int:pk>', views.CommentDetailView.as_view())
26 ]
27 ]
```

2. Update the API/VIEWS.PY



```
212
213
214 # uses nested serializers
215 class BlogView(generics.ListCreateAPIView):
216     queryset = Blog.objects.all()
217     serializer_class = BlogSerializer
218
219 class CommentsView(generics.ListCreateAPIView):
220     queryset = Comment.objects.all()
221     serializer_class = CommentSerializer
222
223 class BlogDetailView(generics.RetrieveUpdateDestroyAPIView):
224     pass
225
226 class CommentDetailView(generics.RetrieveUpdateDestroyAPIView):
227     pass
228
```

Then update as:



```
212
213
214 # uses nested serializers
215 class BlogView(generics.ListCreateAPIView):
216     queryset = Blog.objects.all()
217     serializer_class = BlogSerializer
218
219 class CommentsView(generics.ListCreateAPIView):
220     queryset = Comment.objects.all()
221     serializer_class = CommentSerializer
222
223 class BlogDetailView(generics.RetrieveUpdateDestroyAPIView):
224     queryset = Blog.objects.all()
225     serializer_class = BlogSerializer
226     lookup_field = 'pk'
227
228 class CommentDetailView(generics.RetrieveUpdateDestroyAPIView):
229     pass
230
```

3. Now view the specific path of blog post # 1:

→ ↺ ⓘ 127.0.0.1:8000/api/v1/blogs/1/ 🔍 ☆

Django REST framework api_djangoadmin

Api Root / Blogs / Blog Detail

Blog Detail

DELETE OPTIONS GET ▾

GET /api/v1/blogs/1/

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

```
{
  "id": 1,
  "comments": [
    {
      "id": 1,
      "comment": "this is comment #1",
      "blog": 1
    },
    {
      "id": 2,
      "comment": "this is comment #2",
      "blog": 1
    },
    {
      "id": 3,
      "comment": "this is comment #3",
      "blog": 1
    }
  ],
  "blog_title": "this is a blog 1",
  "blog_body": "this is a blog body 1"
}
```

Raw data HTML form

Blog title

this is a blog 1

Blog body

this is a blog body 1

PUT

Now, you can update and delete this record.

→

127.0.0.1:8000/api/v1/blogs/1/

🔍

☆

Q

🔗

📁

🌐

Django REST frameworkapi_djangoadmin

Api Root / Blogs / Blog Detail

Blog Detail

DELETEOPTIONSGET

DELETE /api/v1/blogs/1/

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

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

Raw dataHTML form

Blog title

Blog body

PUT

You can also add a new post here:

→ ↻ ⓘ 127.0.0.1:8000/api/v1/blogs/ 🔍 ☆ 📄 🗑️ 🎵

Django REST framework api_djangoadmin

Api Root / Blogs

Blogs

OPTIONS GET ▾

GET /api/v1/blogs/



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

```
[
  {
    "id": 2,
    "comments": [
      {
        "id": 4,
        "comment": "this is comment #1",
        "blog": 2
      },
      {
        "id": 5,
        "comment": "this is comment #2",
        "blog": 2
      }
    ],
    "blog_title": "this is a blog 2",
    "blog_body": "this is a blog body 2"
  }
]
```

Raw data HTML form

Blog title

Blog body

POST

Then you can reload your page:

Api Root / Blogs

Blogs

OPTIONS GET

GET /api/v1/blogs/

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

[
  {
    "id": 2,
    "comments": [
      {
        "id": 4,
        "comment": "this is comment #1",
        "blog": 2
      },
      {
        "id": 5,
        "comment": "this is comment #2",
        "blog": 2
      }
    ],
    "blog_title": "this is a blog 2",
    "blog_body": "this is a blog body 2"
  },
  {
    "id": 3,
    "comments": [],
    "blog_title": "this is a blog 3",
    "blog_body": "This is a blog description"
  }
]
```

Raw data HTML form

Blog title

Blog body

POST

4. To allow CRUD operations on Comment model, then we update the class CommentDetailView:

The screenshot shows a code editor with the following structure:

- EXPLORER**
 - DJANGOREST_APIPROJECT
 - api
 - views.py (highlighted)

The main editor displays the code for `views.py` with the following content:

```
213
214 # uses nested serializers
215 class BlogsView(generics.ListCreateAPIView):
216     queryset = Blog.objects.all()
217     serializer_class = BlogSerializer
218
219 class CommentsView(generics.ListCreateAPIView):
220     queryset = Comment.objects.all()
221     serializer_class = CommentSerializer
222
223 class BlogDetailView(generics.RetrieveUpdateDestroyAPIView):
224     queryset = Blog.objects.all()
225     serializer_class = BlogSerializer
226     lookup_field = 'pk'
227
228 class CommentDetailView(generics.RetrieveUpdateDestroyAPIView):
229     queryset = Comment.objects.all()
230     serializer_class = CommentSerializer
231     lookup_field = 'pk'
232
233
```

5. To view the COMMENTS path:

→ ↺ ⓘ 127.0.0.1:8000/api/v1/comments/ 🔍 ☆

🔍 📁 🎵 🌐

Django REST frameworkapi_djangoadmin

Api Root / Comments

Comments

OPTIONSGET

GET /api/v1/comments/

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

```
[
  {
    "id": 4,
    "comment": "this is comment #1",
    "blog": 2
  },
  {
    "id": 5,
    "comment": "this is comment #2",
    "blog": 2
  }
]
```

Raw dataHTML form

Comment

This is a comment #1

Blog

this is a blog 3

POST

6. To delete the specific comment:

→ ↻ ⓘ 127.0.0.1:8000/api/v1/comments/7/ 🔍 ☆

Django REST framework api_djangoadmin

Api Root / Comments / Comment Detail

Comment Detail

DELETE OPTIONS GET ▾

GET /api/v1/comments/7/

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

{
  "id": 7,
  "comment": "test comment",
  "blog": 3
}
```

Raw data HTML form

Comment test comment

Blog this is a blog 3 ▾

PUT

7. View the deleted record:

127.0.0.1:8000/api/v1/comments/7/

Django REST frameworkapi_djangoadmin

Api Root / Comments / Comment Detail

Comment Detail

DELETEOPTIONSGET

GET /api/v1/comments/7/

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

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

Raw dataHTML form

Comment

Blog

this is a blog 2

PUT

8.