

## 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.

```
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 ]
```

2. Update the APIVIEWS.PY

The screenshot shows the Visual Studio Code editor interface for a Django REST API project. The Explorer sidebar on the left shows the project structure, with the `views.py` file selected. The main editor window displays the code for `views.py`. The code includes a comment `# uses nested serializers` and defines three classes: `BlogsView`, `CommentsView`, and `BlogDetailView`. The `BlogDetailView` class is highlighted with a red box. The `CommentDetailView` class is also visible below it.

```
212
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     pass
225
226 class CommentDetailView(generics.RetrieveUpdateDestroyAPIView):
227     pass
228
```

Then update as:

The screenshot shows the Visual Studio Code editor interface for the same Django REST API project. The Explorer sidebar on the left shows the project structure, with the `views.py` file selected. The main editor window displays the code for `views.py`. The code includes a comment `# uses nested serializers` and defines three classes: `BlogsView`, `CommentsView`, and `BlogDetailView`. The `BlogDetailView` class is highlighted with a red box. The `CommentDetailView` class is also visible below it.

```
212
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     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

Blog body

PUT

Now, you can update and delete this record.

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

Django REST framework api\_djangoadmin

Api Root / Blogs / Blog Detail

## Blog Detail

**DELETE** OPTIONS GET

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 data HTML form

Blog title

Blog body

PUT

You can also add a new post here:

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

this is a blog 3

Blog body

This is a blog description



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:

```
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:

Api Root / Comments

# Comments

OPTIONS

GET ▾

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 data

HTML 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

Blog

PUT

7. View the deleted record:

# Comment Detail

DELETE OPTIONS GET

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 data HTML form

Comment

Blog

PUT

8.