

Topic: 12. Nested Serializers for Related Models Part 1

Speaker: Personal / Notebook: API Development using Django Framework



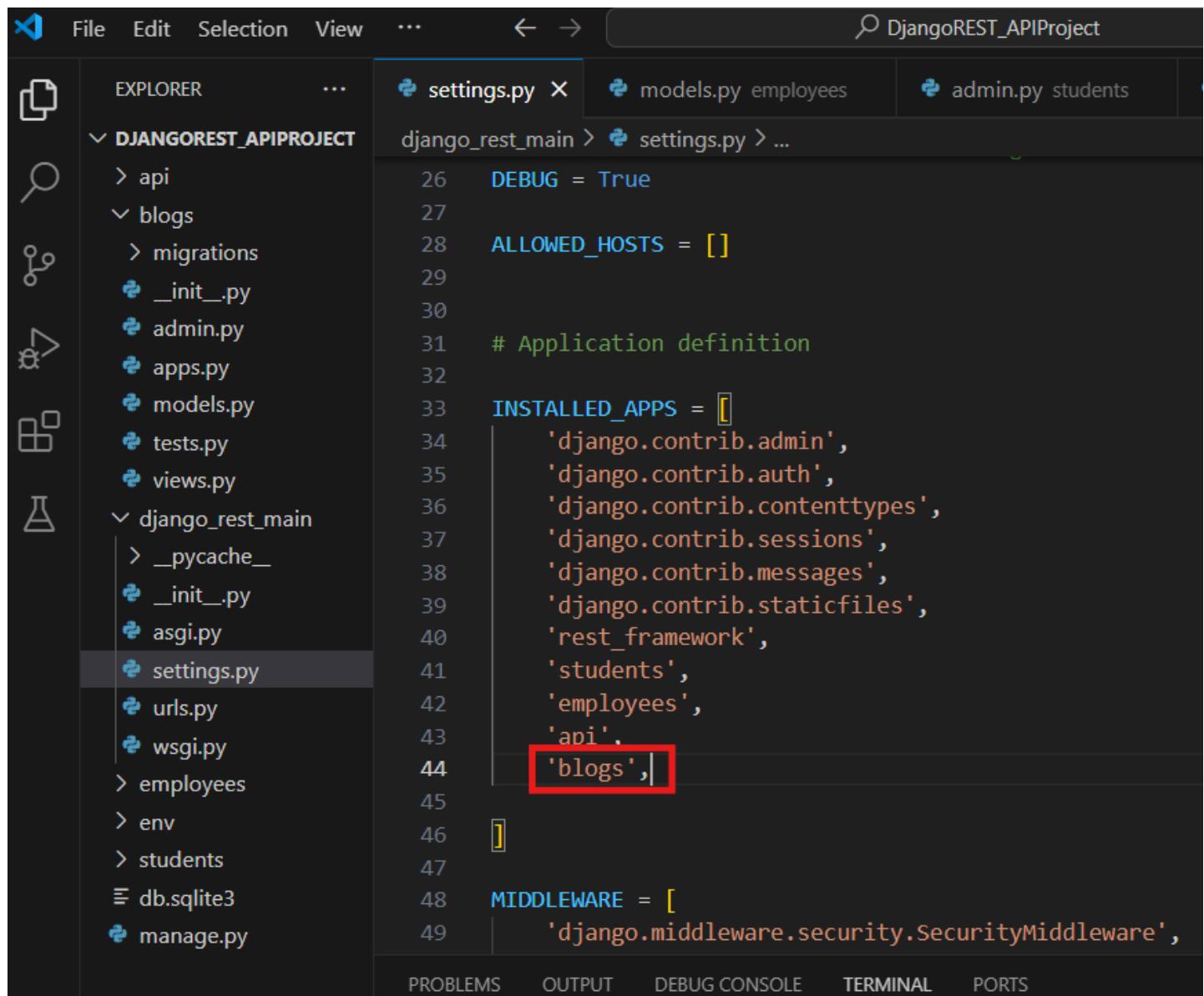
For additional reference, use [RestAPI Documentation](#) on nested serializer.

You can use the [Online JSON Viewer](#) to see/view the code or text.

1. Create the new app BLOGS where each blog post may have several comments. In the terminal, use the create startapp command:

```
(env)
rosil@LearnCodeRepeat MINGW64 /c/Users/rosil/OneDrive/Documents/MyCodingCareer/Django Projects/API De
REST_APIProject
$ python manage.py startapp blogs
```

2. Register this new app in the SETTINGS.PY:

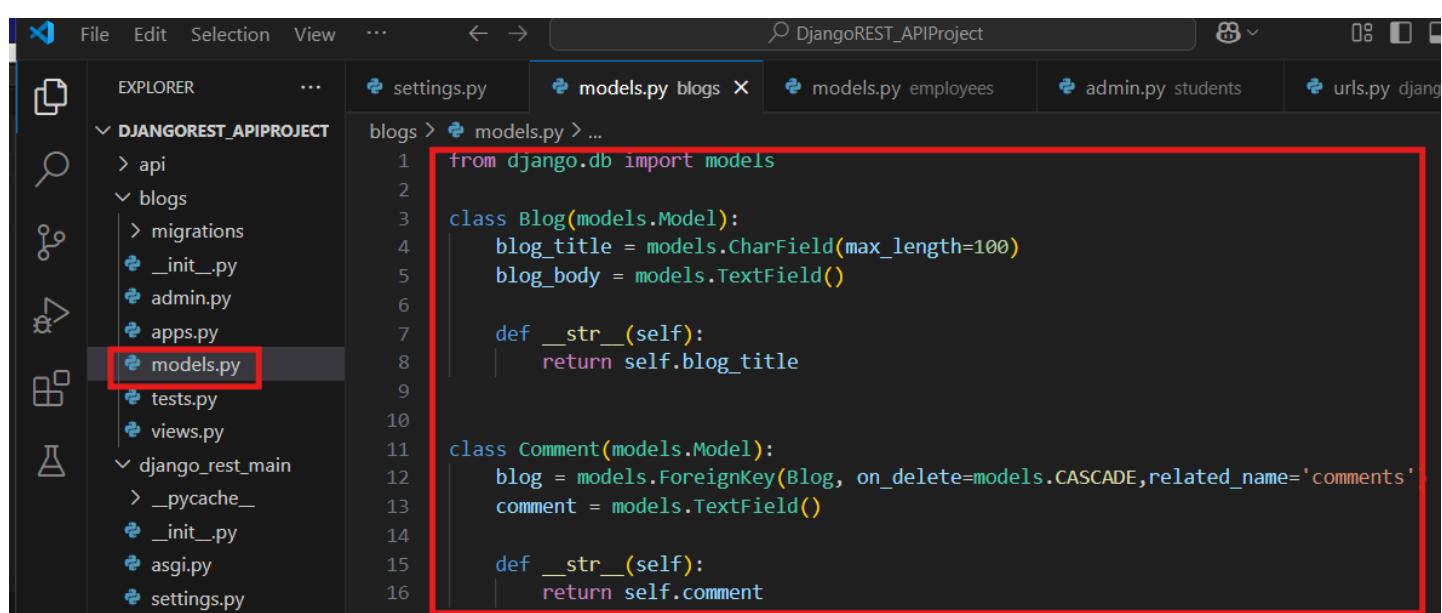


The screenshot shows a code editor with the title "DjangoREST_APIProject". The left sidebar displays the project structure under "DJANGOREST_APIPROJECT". The "models.py" file in the "blogs" directory is currently selected and highlighted with a red box around the line "from django.db import models". The code editor shows the following content for "settings.py":

```
26 DEBUG = True
27
28 ALLOWED_HOSTS = []
29
30 # Application definition
31
32 INSTALLED_APPS = [
33     'django.contrib.admin',
34     'django.contrib.auth',
35     'django.contrib.contenttypes',
36     'django.contrib.sessions',
37     'django.contrib.messages',
38     'django.contrib.staticfiles',
39     'rest_framework',
40     'students',
41     'employees',
42     'api',
43     'blogs',
44 ]
45
46 ]
47
48 MIDDLEWARE = [
49     'django.middleware.security.SecurityMiddleware',
```

The bottom navigation bar includes tabs for PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL, and PORTS.

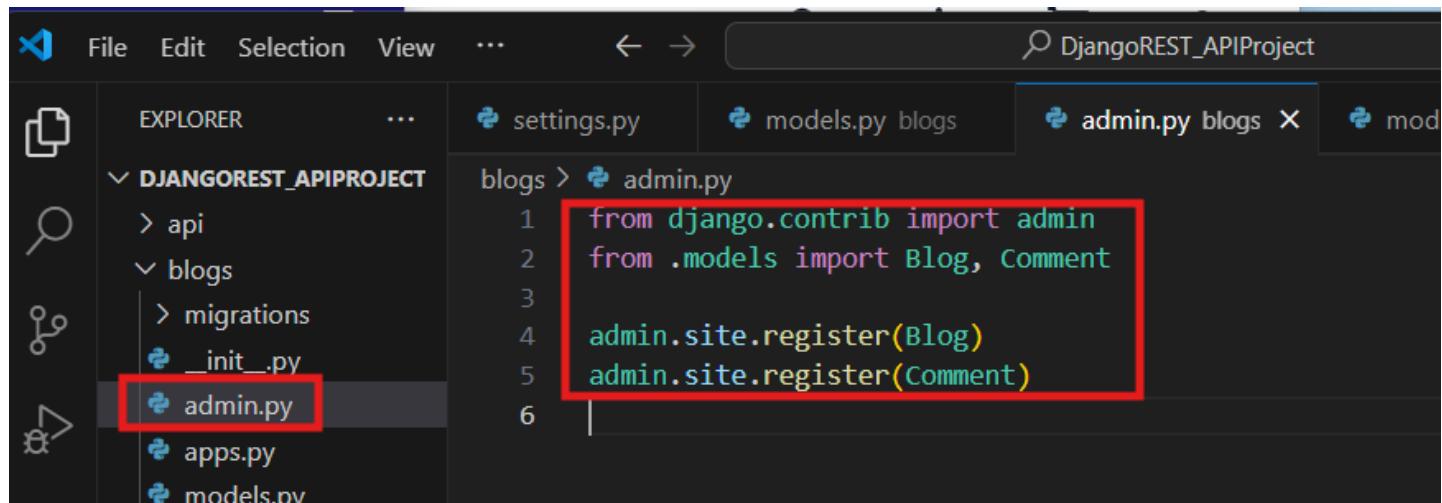
3. Create the models BLOG POST AND COMMENTS:



The screenshot shows a code editor with the title "DjangoREST_APIProject". The left sidebar displays the project structure under "DJANGOREST_APIPROJECT". The "models.py" file in the "blogs" directory is currently selected and highlighted with a red box around the code for the "Blog" and "Comment" models. The code editor shows the following content for "models.py":

```
1 from django.db import models
2
3 class Blog(models.Model):
4     blog_title = models.CharField(max_length=100)
5     blog_body = models.TextField()
6
7     def __str__(self):
8         return self.blog_title
9
10
11 class Comment(models.Model):
12     blog = models.ForeignKey(Blog, on_delete=models.CASCADE, related_name='comments')
13     comment = models.TextField()
14
15     def __str__(self):
16         return self.comment
```

4. Register the models in ADMIN.PY:



```
from django.contrib import admin
from .models import Blog, Comment

admin.site.register(Blog)
admin.site.register(Comment)
```

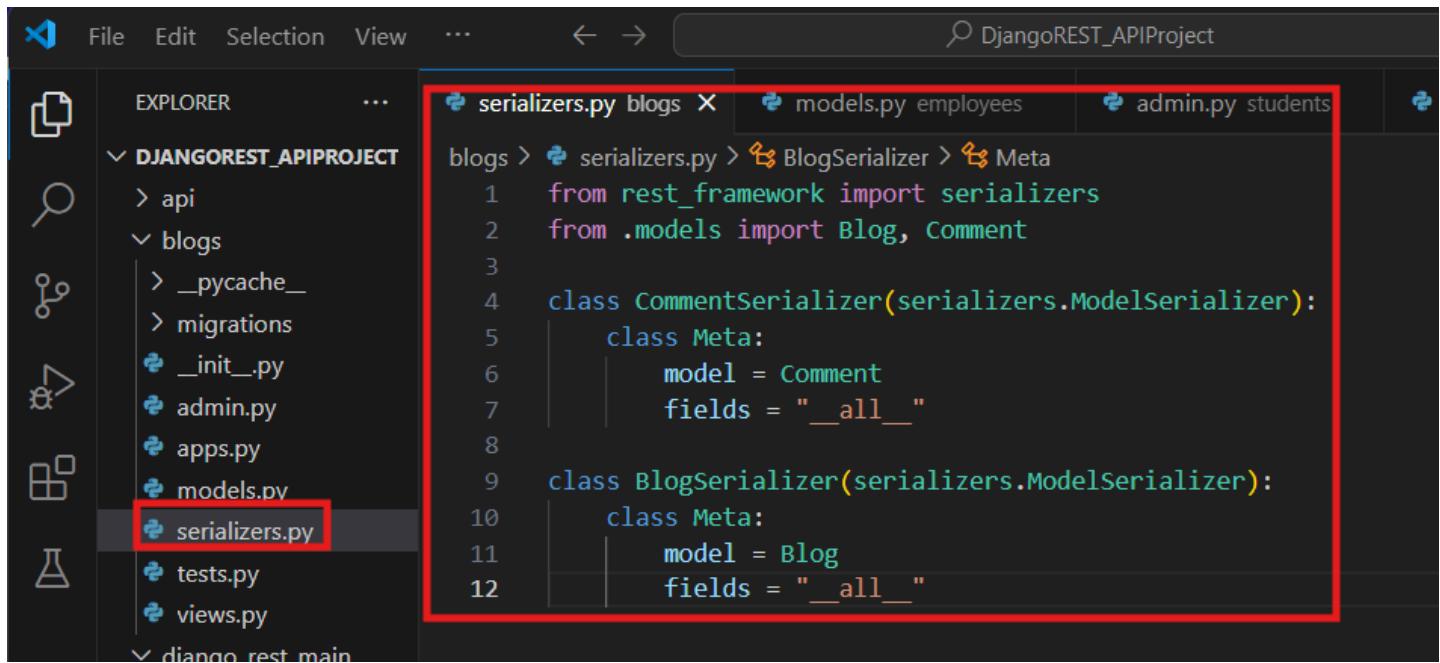
5. Make the migrations.

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL bash +
```

- \$ python manage.py startapp blogs
(env)
rosil@LearnCodeRepeat MINGW64 /c/Users/rosil/OneDrive/Documents/MyCodingCareer/Django Projects/API De
REST_APProject
- \$ python manage.py makemigrations
.Migrations for 'blogs':
 blogs\migrations\0001_initial.py
 + Create model Blog
 + Create model Comment
(env)
rosil@LearnCodeRepeat MINGW64 /c/Users/rosil/OneDrive/Documents/MyCodingCareer/Django Projects/API De
REST_APProject
- \$ python manage.py migrate
Operations to perform:
 Apply all migrations: admin, auth, blogs, contenttypes, employees, sessions, students
Running migrations:
 Applying blogs.0001_initial... **OK**
(env)
rosil@LearnCodeRepeat MINGW64 /c/Users/rosil/OneDrive/Documents/MyCodingCareer/Django Projects/API De
REST_APProject

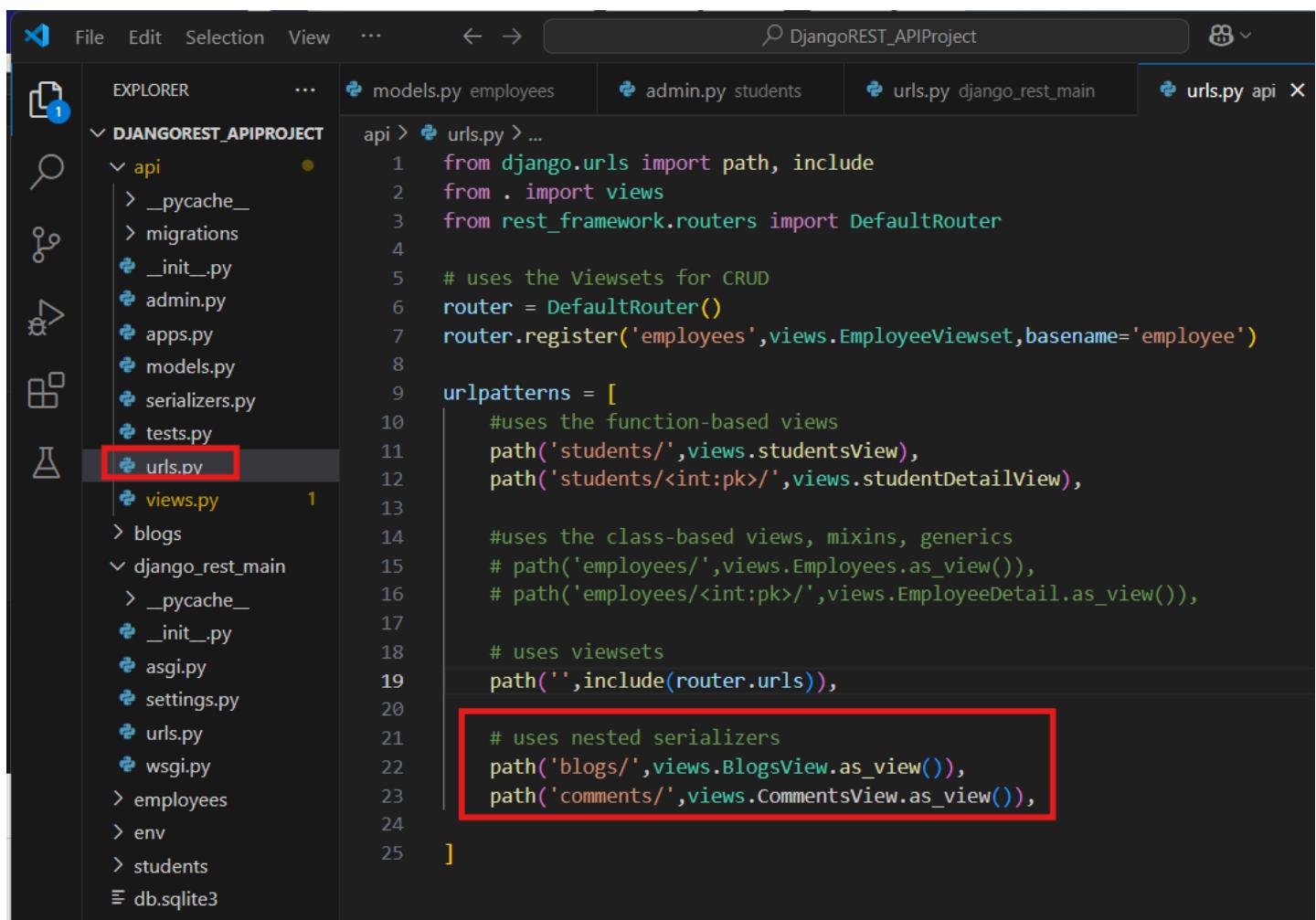
6. Use the admin dashboard to add a new record.

7. Create a new file **SERIALIZER** in the **BLOG FOLDER**. You may also use the **serializer.py** in the **API** folder. This may work so add the **serializers.py** where you want it. For organization, have it in the app you want to use the **serializer** with.



```
serializers.py blogs X models.py employees admin.py students
blogs > serializers.py > BlogSerializer > Meta
1  from rest_framework import serializers
2  from .models import Blog, Comment
3
4  class CommentSerializer(serializers.ModelSerializer):
5      class Meta:
6          model = Comment
7          fields = "__all__"
8
9  class BlogSerializer(serializers.ModelSerializer):
10     class Meta:
11         model = Blog
12         fields = "__all__"
```

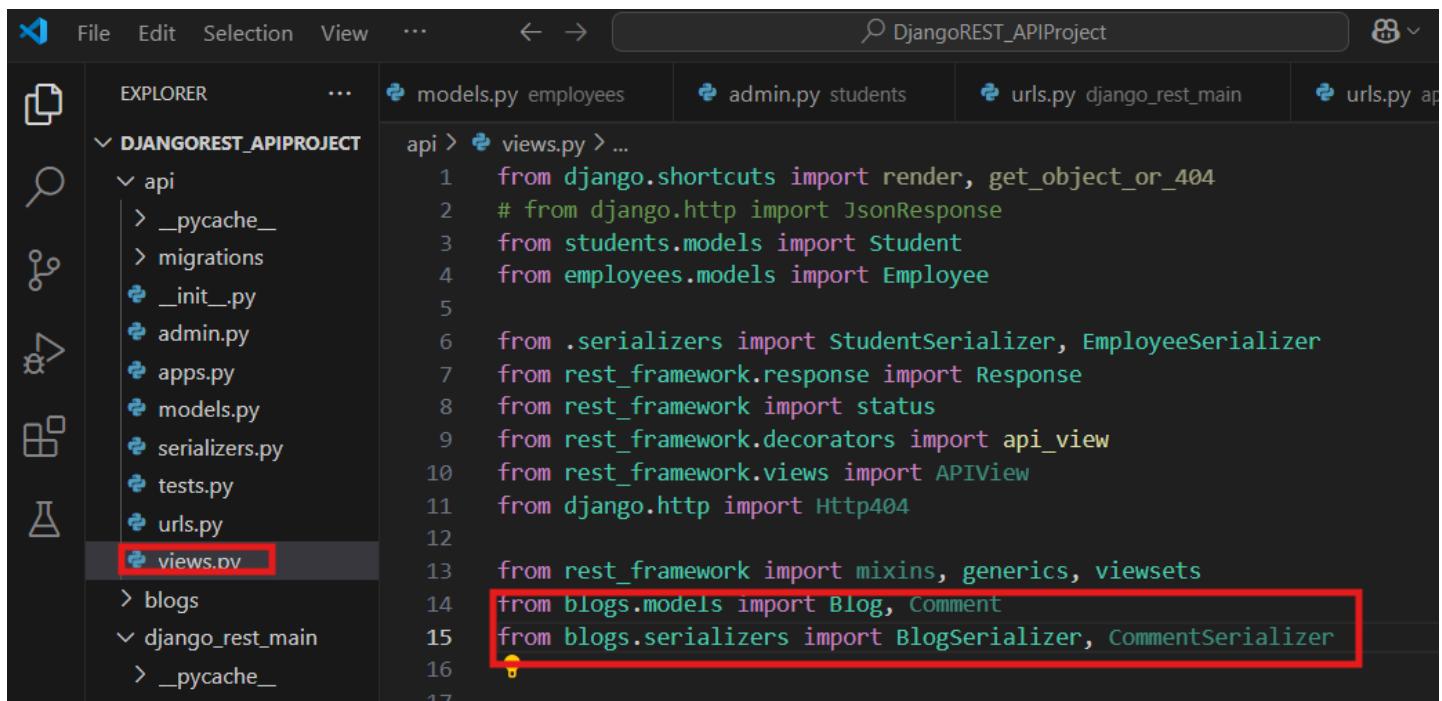
8. Create the path in the API\URLS.PY:



```
models.py employees admin.py students urls.py django_rest_main urls.py api X
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
21     # uses nested serializers
22     path('blogs/',views.BlogsView.as_view()),
23     path('comments/',views.CommentsView.as_view()),
24 ]
25 ]
```

9. Update the VIEWS.PY.

Add the models BLOG & COMMENT:



File Edit Selection View ... ← → 🔍 DjangoREST_APIProject 🌐

EXPLORER

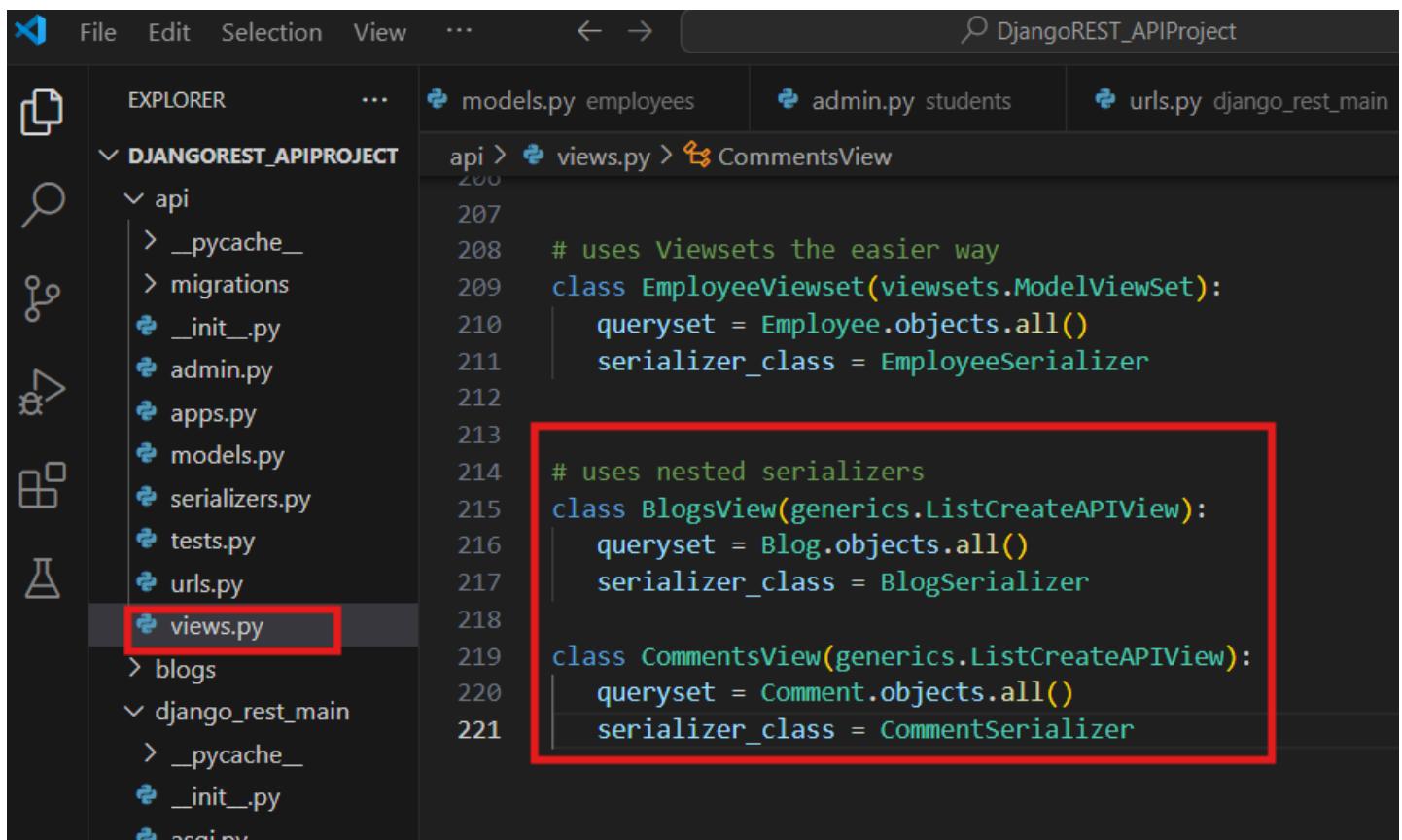
DJANGOREST_APIPROJECT

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

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

```
api > views.py > ...
1  from django.shortcuts import render, get_object_or_404
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 from blogs.models import Blog, Comment
15 from blogs.serializers import BlogSerializer, CommentSerializer
16
17
```

Create the classes in the VIEWS.PY:



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
- django_rest_main
 - __pycache__

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

```
api > views.py > CommentsView
207
208 # uses Viewsets the easier way
209 class EmployeeViewSet(viewsets.ModelViewSet):
210     queryset = Employee.objects.all()
211     serializer_class = EmployeeSerializer
212
213
214 # uses nested serializers
215 class BlogsView(generics.ListCreateAPIView):
216     queryset = Blog.objects.all()
217     serializer_class = BlogSerializer
218
219
220 class CommentsView(generics.ListCreateAPIView):
221     queryset = Comment.objects.all()
222     serializer_class = CommentSerializer
```

10. View the BLOG and COMMENT models:

→ G 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

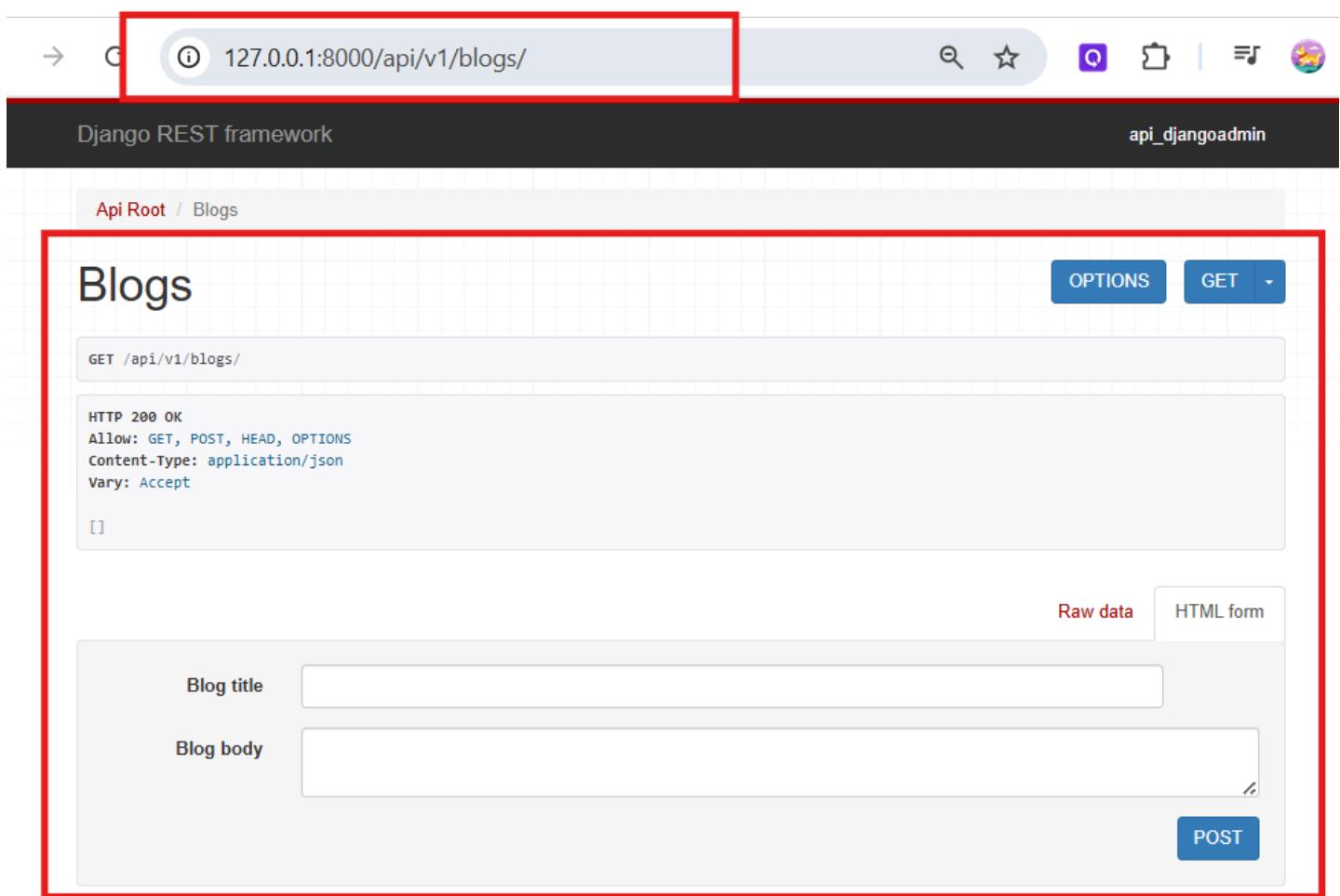
[]

Raw data HTML form

Blog title

Blog body

POST



→ G 127.0.0.1:8000/api/v1/comments/

Django REST framework api_djangoadmin

Api Root / Comments

Comments

OPTIONS GET

GET /api/v1/comments/

HTTP 200 OK

Allow: GET, POST, HEAD, OPTIONS

Content-Type: application/json

Vary: Accept

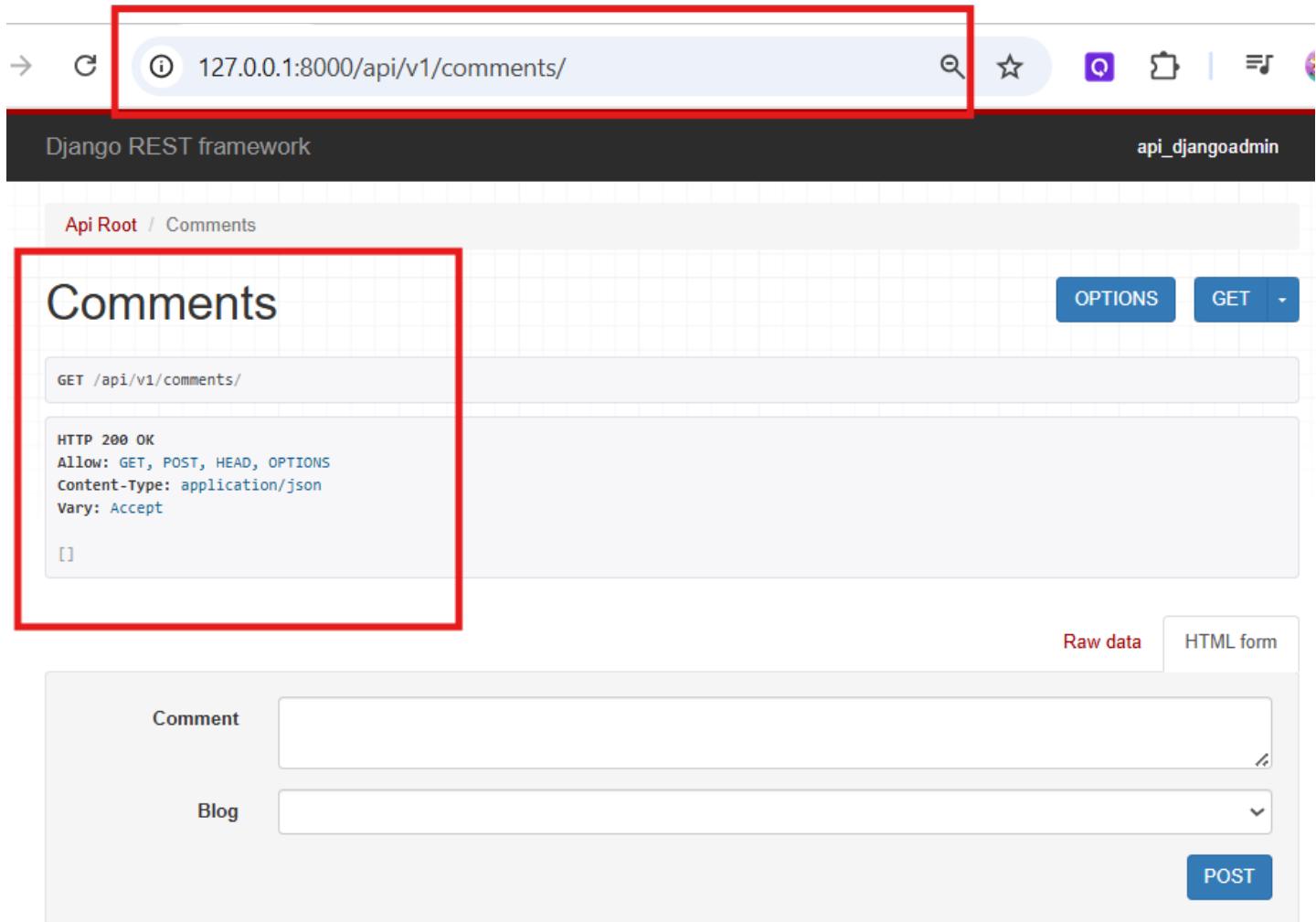
[]

Raw data HTML form

Comment

Blog

POST



11. Add new records.

The screenshot shows a browser window with the URL `127.0.0.1:8000/api/v1/blogs/`. The page title is "Django REST framework" and the sub-page title is "api_djangoadmin". The main content area is titled "Blogs". On the right, there are buttons for "OPTIONS", "GET", and "POST". Below the title, a "POST /api/v1/blogs/" button is visible. The response body shows a successful `HTTP 201 Created` response with the following headers and data:

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

{
    "id": 1,
    "blog_title": "this is a blog 1",
    "blog_body": "this is a blog body 1"
}
```

Below the response, there are two input fields: "Blog title" with the value "this is a blog 2" and "Blog body" with the value "this is a blog body 2". To the right of these fields are "Raw data" and "HTML form" buttons. At the bottom right is a "POST" button.

→ C ⓘ 127.0.0.1:8000/api/v1/comments/ ⚙ ⚙ ⚙ ⚙ ⚙ ⚙ ⚙

Django REST framework

api_djangoadmin

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

Raw data HTML form

Comment	<input type="text" value="this is comment #3"/>	
Blog	<input type="text" value="this is a blog 1"/>	

POST

12. Our COMMENTS model shall be:

Comments

[OPTIONS](#)[GET](#)[▼](#)[GET /api/v1/comments/](#)

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

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

[Raw data](#)[HTML form](#)

Comment

Blog

[POST](#)

Our BLOGS shall be:

Django REST framework

Api Root / Blogs

Blogs

GET /api/v1/blogs/

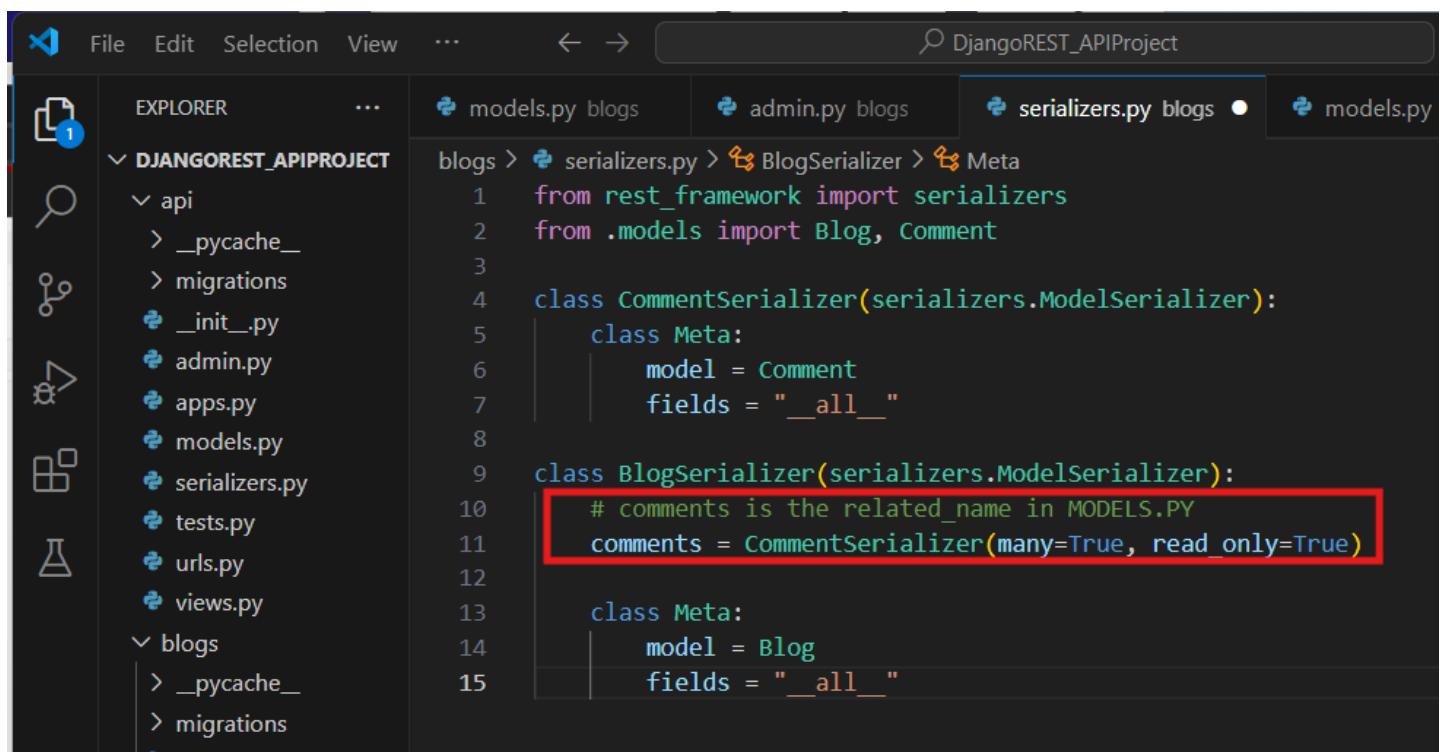
```
HTTP 200 OK
Allow: GET, POST, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept

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

Blog title

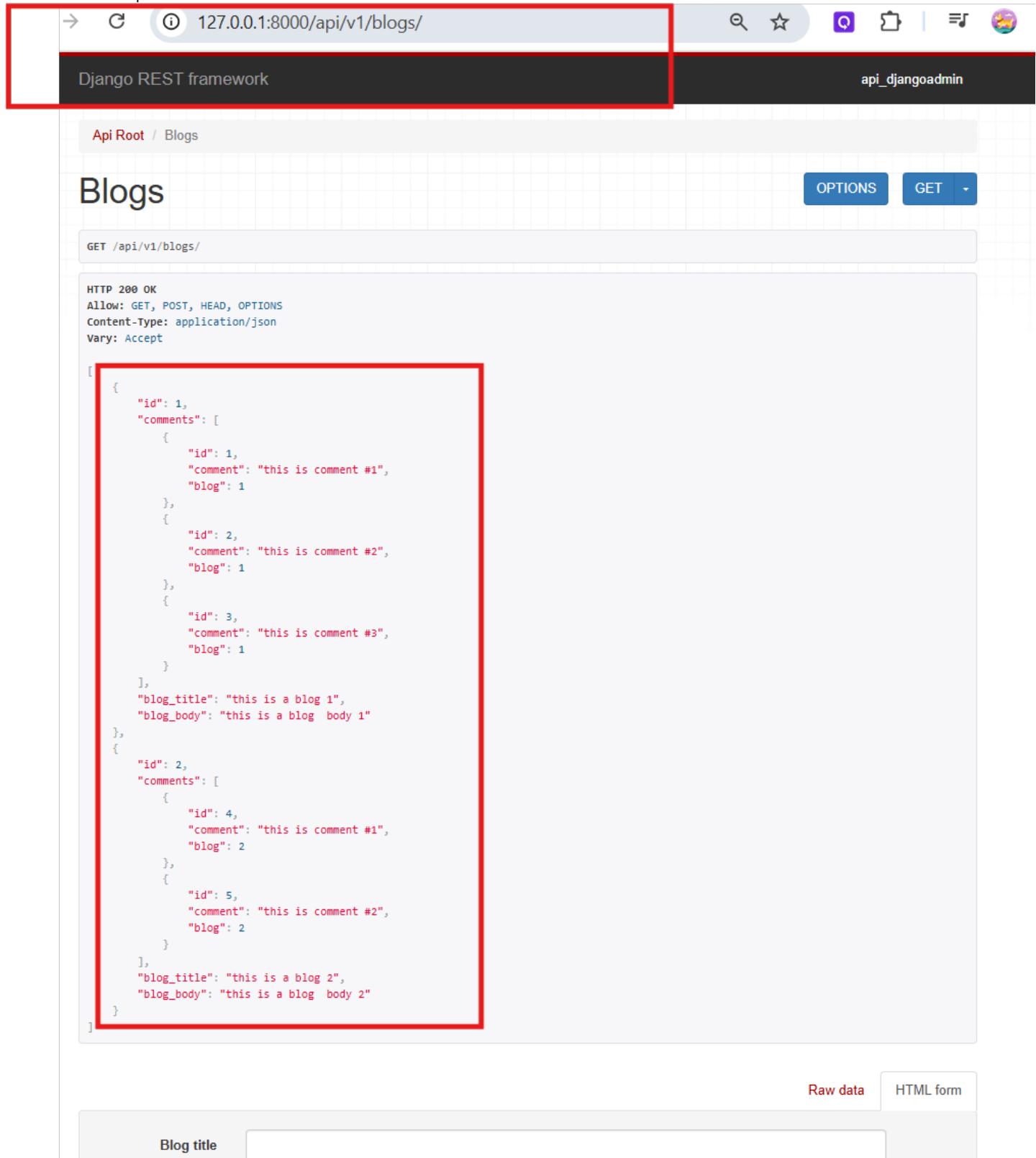
Blog body

13. To also show the comments under BLOGS model, update the SERIALIZERS.PY:



```
File Edit Selection View ... ← → 🔍 DjangoREST_APIProject
EXPLORER ... models.py blogs admin.py blogs serializers.py blogs ● models.py
DJANGOREST_APIPROJECT
  api
    __pycache__
    migrations
    __init__.py
    admin.py
    apps.py
    models.py
    serializers.py
    tests.py
    urls.py
    views.py
  blogs
    __pycache__
    migrations
    ...
serializers.py
blogs > serializers.py > BlogSerializer > Meta
1  from rest_framework import serializers
2  from .models import Blog, Comment
3
4  class CommentSerializer(serializers.ModelSerializer):
5      class Meta:
6          model = Comment
7          fields = "__all__"
8
9  class BlogSerializer(serializers.ModelSerializer):
10     # comments is the related_name in MODELS.PY
11     comments = CommentSerializer(many=True, read_only=True)
12
13     class Meta:
14         model = Blog
15         fields = "__all__"
```

To view the BLOGS pa



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": 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"}, {"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

14. So now, copy and paste the sample records to ONLINE JSON VIEWER:

TEXT MODE:



jsonviewer.stack.hu

Viewer

Text

Paste Copy | Format Remove white space | Clear | Load JSON data

```
[  
  {  
    "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"  
  },  
  {  
    "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"  
  }  
]
```

VIEWER MODE:

The screenshot shows a JSON viewer interface with a red box highlighting the URL bar and the main content area. The URL bar contains the text "jsonviewer.stack.hu". The main content area is titled "Viewer" and shows a hierarchical tree structure of a JSON document. The root node is "JSON". It contains a node "0" with an "id" of 1. This node has a "comments" array, which contains a node "0" with an "id" of 1, a "comment" of "this is comment #1", and a "blog" of 1. Below this are two more nodes, "1" and "2", each with a "blog_title" of "this is a blog 1" and a "blog_body" of "this is a blog body 1". At the bottom of the tree is another node "1".

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

15.