

## 16 - Shruti Gauchandra

### Assignment - 04

Develop a RESTful API for a Library Management System using Django REST Framework (DRF). The system should manage books and authors with basic CRUD functionality.

Code:

models.py

```
1  from django.db import models
2
3  class Author(models.Model):
4      name = models.CharField(max_length=100)
5
6      def __str__(self):
7          return self.name
8
9
10 class Book(models.Model):
11     title = models.CharField(max_length=200)
12     isbn = models.CharField(max_length=13, unique=True)
13     author = models.ForeignKey(Author, on_delete=models.CASCADE)
14
15     def __str__(self):
16         return self.title
```

serializers.py

```
1  from rest_framework import serializers
2  from .models import Author, Book
3
4  class AuthorSerializer(serializers.ModelSerializer):
5      class Meta:
6          model = Author
7          fields = '__all__'
8
9
10 class BookSerializer(serializers.ModelSerializer):
11     class Meta:
12         model = Book
13         fields = '__all__'
```

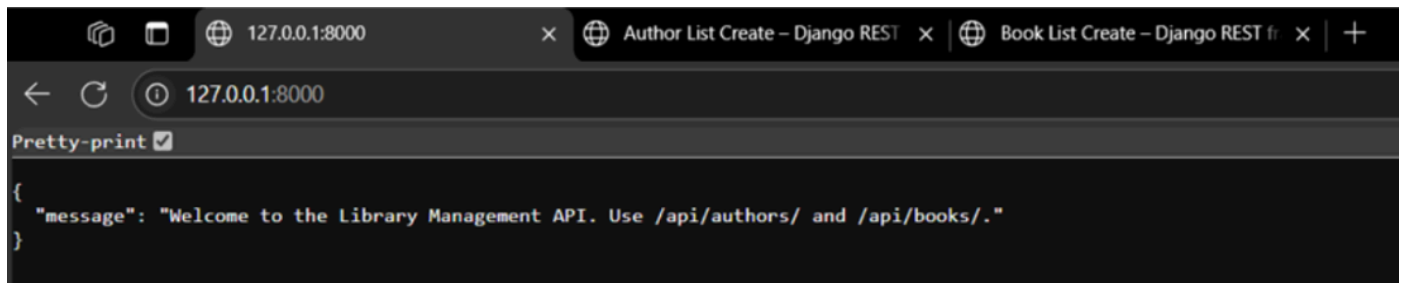
urls.py

```
1 from django.urls import path
2 from .views import AuthorListView, AuthorDetailView, BookListView, BookDetailView
3
4 urlpatterns = [
5     path('authors/', AuthorListView.as_view(), name='author-list-create'),
6     path('authors/<int:pk>', AuthorDetailView.as_view(), name='author-detail'),
7     path('books/', BookListView.as_view(), name='book-list-create'),
8     path('books/<int:pk>', BookDetailView.as_view(), name='book-detail'),
9 ]
10
```

views.py

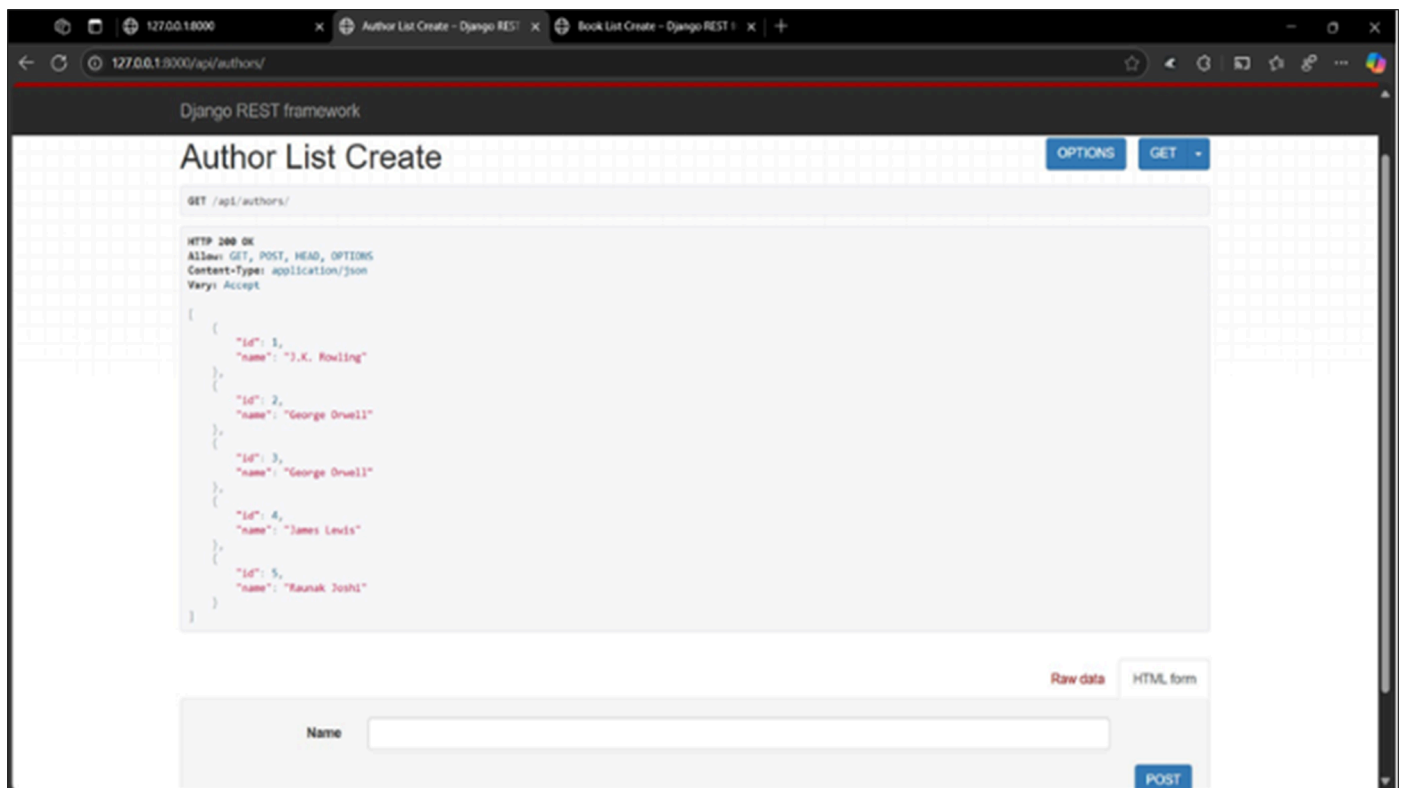
```
1 from django.shortcuts import render
2 from rest_framework import generics
3 from .models import Author, Book
4 from .serializers import AuthorSerializer, BookSerializer
5 from django.http import JsonResponse
6
7 # Author Views
8 class AuthorListView(generics.ListCreateAPIView):
9     queryset = Author.objects.all()
10    serializer_class = AuthorSerializer
11
12
13 class AuthorDetailView(generics.RetrieveUpdateDestroyAPIView):
14     queryset = Author.objects.all()
15     serializer_class = AuthorSerializer
16
17
18 # Book Views
19 class BookListView(generics.ListCreateAPIView):
20     queryset = Book.objects.all()
21     serializer_class = BookSerializer
22
23
24 class BookDetailView(generics.RetrieveUpdateDestroyAPIView):
25     queryset = Book.objects.all()
26     serializer_class = BookSerializer
27
28 def home_view(request):
29     return JsonResponse({"message": "Welcome to the Library Management API. Use /api/authors/ and /api/books/."})
```

Output:



A screenshot of a web browser window. The address bar shows the URL `127.0.0.1:8000`. The page content displays a JSON object with a single key-value pair: `{ "message": "Welcome to the Library Management API. Use /api/authors/ and /api/books/." }`. A "Pretty-print" checkbox is checked in the top left corner of the response area.

```
{
  "message": "Welcome to the Library Management API. Use /api/authors/ and /api/books/."
}
```



A screenshot of the Django REST framework API explorer. The browser address bar shows `127.0.0.1:8000/api/authors/`. The page title is "Author List Create". The method selected is "GET". The response is displayed in "Raw data" format, showing a JSON array of five author objects. At the bottom, there is a "Name" input field and a "POST" button. The "HTML form" tab is also visible.

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

```
[
  {
    "id": 1,
    "name": "J.K. Rowling"
  },
  {
    "id": 2,
    "name": "George Orwell"
  },
  {
    "id": 3,
    "name": "George Orwell"
  },
  {
    "id": 4,
    "name": "James Lewis"
  },
  {
    "id": 5,
    "name": "Ranak Joshi"
  }
]
```

Raw data HTML form

Name

POST

127.0.0.1:8000 x Author List Create - Django REST x Book List Create - Django REST x

127.0.0.1:8000/api/books/

Django REST framework

## Book List Create

OPTIONS GET

GET /api/books/

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

```
[
  {
    "id": 1,
    "title": "Harry Potter",
    "isbn": "1234567890123",
    "author": 1
  },
  {
    "id": 2,
    "title": "Harry Potter and the Sorcerer's Stone",
    "isbn": "1234567890123",
    "author": 1
  }
]
```

Raw data HTML form

Title

Isbn

Author

POST