LAB REPORT

Topic: Developing a RESTful API Using Django REST-Framework

Name: Dipto Barua

Email:diptobarua18@gmail.com

Batch: CBI 033 Python-Django

Submitted To: Shaikat Sharma

Aim: To Create a basic Django project, build a simple Book model, and develop a RESTful API using Django REST Framework.

Objectives: To design and develop a basic Django project that includes a simple Book model and a RESTful API using Django REST Framework. The objective is to implement CRUD operations for the Book model, enabling seamless interaction with the database through well-structured API endpoints.

Resources:

Software/Tools:

- **Python**: The programming language required to build the Django Project
- **Django**: The web framework to create the project structure and handle server-side logic.
- **Django REST Framework (DRF)**: An extension to Django for building APIs.
- **Database**: A database system like SQLite (default with Django)
- **IDE/Text Editor**: Visual Studio Code to write and edit code.

Hardware/Tools

- A computer with at least 4 GB of RAM and a stable internet connection to download libraries and set up the environment.

Code:

Activate virtual Environment >> myenv\Scripts\activate

Installation of REST Framework >> pip install djangorestframework

Creating project and app

(myenv) PS C:\Users\dipto\Desktop\Django> django-admin startproject labtest

(myenv) PS C:\Users\dipto\Desktop\Django> cd labtest

(myenv) PS C:\Users\dipto\Desktop\Django\labtest> python manage.py startapp books

Adding the books app to the INSTALLED_APPS in project's settings

```
INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'rest_framework',
    'books',

]
```

Creating model

```
labtest > books > models.py > Book > title
from django.db import models

class Book(models.Model):
    title = models.CharField(max_length=255)
    author = models.CharField(max_length=255)
    description = models.TextField()
    published_date = models.DateField()

    def __str__(self):
        return self.title

```

Migration

```
(myenv) PS C:\Users\dipto\Desktop\Django\labtest> python manage.py makemigrations
Migrations for 'books':
  books\migrations\0001_initial.py
    + Create model Book
(myenv) PS C:\Users\dipto\Desktop\Django\labtest> python manage.py migrate
Operations to perform:
  Apply all migrations: admin, auth, books, contenttypes, sessions
Running migrations:
  Applying contenttypes.0001_initial... OK
```

Creating serializer

```
labtest > books > serializers.py > BookSerializer > Meta
from rest_framework import serializers
from .models import Book

class BookSerializer(serializers.ModelSerializer):
    class Meta:
        model = Book
        fields = '__all__'
```

Creating Views

```
labtest > books > views.py > BookDetail
from django.shortcuts import render
from rest_framework import generics
from .models import Book
from .serializers import BookSerializer

class BookList(generics.ListCreateAPIView):
    queryset = Book.objects.all()
    serializer_class = BookSerializer

class BookDetail(generics.RetrieveUpdateDestroyAPIView):
    queryset = Book.objects.all()
    serializer_class = BookSerializer
```

Urls Configuration

```
labtest > books > urls.py > ...
from django.urls import path
from . import views

urlpatterns = [
    path('books/', views.BookList.as_view(), name='book-list'),
    path('books/<int:pk>/', views.BookDetail.as_view(), name='book-detail'),
]
```

Including the app's URLs in the project

```
labtest > labtest > urls.py > ...
    1. Import the include() function: from django.urls import include, path
    2. Add a URL to urlpatterns:  path('blog/', include('blog.urls'))
"""
from django.contrib import admin
from django.urls import path, include

urlpatterns = [
    path('admin/', admin.site.urls),
    path('api/', include('books.urls')),
]
```

Output:

For testing the API >> python manage.py runserver

To post a new book >> http://127.0.0.1:8000/api/books/

Raw data | HTML form

Title

Author

Description

Published date dd/mm/yyyy

POST

To view all the Books>> http://127.0.0.1:8000/api/books/


```
[
    {
        "id": 1,
        "title": "Aspect Physics",
        "author": "Hassan",
        "description": "an approch to learn physics",
        "published_date": "2001-02-22"
    },
    {
        "id": 2,
        "title": "Onnorokom",
        "author": "MD Abdullah",
        "description": "drama",
        "published_date": "2007-05-23"
    },
    {
        "id": 3,
        "title": "basic computer",
        "author": "a hasan",
        "description": "an approch to learn computer fundamental",
        "published_date": "2015-02-22"
    }
]
```


To get a single Book >> http://127.0.0.1:8000/api/books/<id>/

Book Detail

```
GET /api/books/2/
```


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

{
    "id": 2,
    "title": "Onnorokom",
    "author": "MD Abdullah",
    "description": "drama",
    "published_date": "2007-05-23"
}
```

Conclusion:

In conclusion, developing a basic Django project with a Book model and a RESTful API using Django REST Framework provides a solid foundation for understanding web application development. By utilizing the appropriate software tools, libraries, and development best practices, this project demonstrates how to efficiently create and manage APIs with CRUD operations. With the skills and resources outlined, developers can extend this knowledge to build more complex and scalable web applications in the future.