***API NOTES***

***TABLE OF CONTENT***

1. ***Without Using API***
2. ***Without Using Serializer Function Based***
3. ***Without Using Serializer Class Based***
4. ***Using Serializer(serializer) Function Based***
5. ***Using Serializer(Modelserializer) Function Based***
6. ***Using serializer(serializer) Class Based***
7. ***Using serializer(serializer) Class Based with Validation(3 types)***
8. ***Using serializer(Modelserializer) Class Based***

|  |  |
| --- | --- |
| **No** | **Table of Content** |
| 1 | Without API |
| 2 |  |
| 3 |  |
| 4 | Function Based Views - With Serialization (Serializer Class) Without Validation |
| 5 | Class Based Views - With Serialization (Serializer Class) Without Validation |
| 6 | Class Based Views - With Serialization (Serializer Class) With Validation |
| 7 | Class Based Views - With Serialization (ModelSerializer Class) With Validation |
| 8 | Class Based Views - With Serialization (ModelSerializer Class) ForeignKey |
| 9 | Class Based Views - With Serialization (ModelSerializer Class) ForeignKey + Nested Serializer |
| 10 | Class Based Views - With Serialization (ModelSerializer Class) ForeignKey + Nested Serializer (Advanced) |
| 11 | Class Based Views - Generic View + Mixin View + API View – ModelSerializer + ForeignKey |
| 12 | Url Mapping with Diifernt Values |
| 13 | Class Based Generic Views + VIEWSET (Viewset,Modelviewset,ReadOnlyModelViewset) |
| 14 |  |

***Initial setting***

1.Create Virtual Environment.

2.Install Packages

pip install djangorestframework

pip install mysqlclient

3.create Project

4.Create New Application

5.Add New App and restful api in settings installed app.

INSTALLED\_APPS = [

    'django.contrib.admin',

    'django.contrib.auth',

    'django.contrib.contenttypes',

    'django.contrib.sessions',

    'django.contrib.messages',

    'django.contrib.staticfiles',

    'Movieapp',

    'rest\_framework'

]

6.change database

7

**TOPICS**

**1.Relationship**

* 1. **One to One**
  2. **Many to One**
  3. **Many to Many**

**2.Serializers Types**

1. **Serializer:** This is the base class for creating serializers. You define the fields explicitly, and it's the most flexible but requires more manual work.
2. **ModelSerializer**: This is a subclass of Serializer that automatically generates fields based on the model you provide. It simplifies the process of creating serializers for your models.
3. **HyperlinkedModelSerializer**: This is similar to ModelSerializer but includes hyperlinks in the serialized representation. Instead of just representing relationships with primary keys, it uses hyperlinks, which can be more RESTful and informative.

*## Serilaizer Example*

from **rest\_framework** import **serializers**

class **MyCustomSerializer**(**serializers**.**Serializer**):

    field1 = **serializers**.**CharField**(max\_length=100)

    field2 = **serializers**.**IntegerField**()

*## Model serializer*

from **rest\_framework** import **serializers**

from myapp.models import MyModel

class **MyModelSerializer**(**serializers**.**ModelSerializer**):

    class **Meta**:

        model = MyModel

        fields = ['field1', 'field2', 'field3']

from **rest\_framework** import **serializers**

from myapp.models import MyModel

Hyperlinked Example

class **MyHyperlinkedModelSerializer**(**serializers**.**HyperlinkedModelSerializer**):

    class **Meta**:

        model = MyModel

        fields = ['url', 'field1', 'field2', 'field3']

        extra\_kwargs = {

            'url': {'view\_name': 'mymodel-detail', 'lookup\_field': 'pk'}

        }

**3.Field Types in Foreign Key**

1. **StringRelatedField**: This represents the foreign key relationship using the \_\_str\_\_ method of the related model. It's useful if you want to display a human-readable representation of the related object.
2. **PrimaryKeyRelatedField**: This represents the foreign key relationship using the primary key (usually an integer) of the related model. It is straightforward and efficient if you just need the ID of the related object.
3. **HyperlinkedRelatedField**: This represents the foreign key relationship using a hyperlink to the related object's detail view. It is more RESTful and provides more information about the related object.