هنگامیکه دیتایی را میخواهید به کلاس مربوطه serializer بدهید باید آن دیتا را به متغیر data بدهید

my\_ser = {"phone\_number":"099085sdsd37607" , "first\_name":"ahamd" , "last\_name":"moshtaghi"}

my\_data = ProfileSerializers(data = my\_ser)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

هنگامیکه از serializers.Serializer در استفاده میکنید حتما باید تابع create ساخته شود به صورت زیر :

    def create(self , validated\_data):

        return Profiles.objects.create(\*\*validated\_data)

\*\*validated\_data -----> به صورت دیکشنری

هنگام update هم حتما باید تابع update را implement کنیم

Regexfield حتما مطالعه شود

اولین آرگومانی که متود init هر کلاسی در سریالازر دارد instance میباشد به این صورت که برای بروزرسانی دیتاهایی که در دیتابیس دخیره شده هست میتوانیم به instance بدهیم مثلا:

My\_profiel = Profiles.objects.get(id=3)

My\_ser = ProfilesSerializers(instance = my\_profile , data = my\_data)

هنگامیکه از متد is\_valid() استفاده میکنیم ممکن است خروجی false دریافت کنیم . برای اینکه متوجه شویم کدام فیلد مشکل دارد باید از متد errors() استفاده شود که کاملا توضیح میدهد

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

posts = PostsSerializers( many = True )

serializers = ProfileSerializers(profiles , many = True)

many=True به این معنی است که serializer میتواند چندین objects را دریافت و نمایش دهد

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

هنگامیکه از متد post استفاده میکنیم در veiws لازم است request را به json تبدیل کنیم

برای این کار کافیست از دستورات زیر استفاده کنیم

from rest\_framework.parsers import JSONParser

data = JSONParser().parse(request)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

در serializer اگر بخواهیم متوجه شویم مقدار فیلدی درست وارد شده است کافیست از روش زیر استفاده کنیم مثلا برای (phone\_number) :

import re

from rest\_framework.exceptions import ValidationError

def validate\_phone\_number(self , value):

phone\_number\_pattern = re.compile("09\d{9}$")

if phone\_number\_pattern.match(value):

return value

raise ValidationError("{phone\_number is not correct}")

اگر بخواهیم برای فیلد دیگری این کار را انجام دهیم باید از قانون زیر استفاده کنیم:

def validate\_”namefield”(self , value)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

هنگامیکه بخواهیم دو مدل را از طریق کلید خارجی به هم متصل کنیم لازم است و همچنین از serializer هم استفاده کنیم حتما باید از related\_name استفاده کنیم

class Posts(models.Model):

author = models.ForeignKey(Profiles , related\_name='posts' , on\_delete=models.CASCADE)

اکنون میتوانیم در serializer که مربوط به profiles است postهای مربوطه را اینچنین نمایش دهیم

posts = PostsSerializers( many = True , read\_only=True)

“posts----------->related\_name=”post

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

هنگامیکه بخواهیم از طریق کلید خارجی و متد POST مدل دیگری بسازیم (profiles , posts) بدیل اینکه profiles همان لحطه ساخته میشود لازم است برای post متد create را دوباره نویسی کنیم که میتوانیم از قانون زیر پیروی کنیم :

post = PostsSerializers(write\_only=True)

class ProfileSerializers(serializers.ModelSerializer):

    def create(self , validate\_data):

        post = validate\_data.pop("post")

        my\_author = Profiles.objects.create(\*\*validate\_data)

        my\_post = Posts.objects.create(\*\*post , author=my\_author)

        return my\_author

اکنون لازم است فیلد author که فیلد مربوط به مدل post میباشد و با استفاده از کلید خارجی به مدل profiles وصل شده است ویژگی دیگری به آن اضافه کنیم این ویژگی required میباشد که این گونه اضافه میشود :

class PostsSerializers(serializers.ModelSerializer):

    class Meta:

        model = Posts

        fields = '\_\_all\_\_'

        extra\_kwargs = {'author':{'required':False}}

در واقع این ویژگی میگوید لازم نیست این فیلد را داخل ورودی inputها داشته باشیم

گاهی اوقات لازم نیست مدلی که از طریق کلید خارجی به مدل اصلی ما متصل شده است تمام فیلدهایش نمایش داده شود و فقط لازم است مثلا بگوید کدام postها مربوط به کدام profile میباشد برای همین کار از primarykeyrelatedfield به صورت زیر استفاده کنید :

class ProfileSerializers(serializers.ModelSerializer):

    posts = serializers.PrimaryKeyRelatedField( many=True , read\_only=True)

دقت شود posts که در بالا تعریف شده است حتما باید هم نام با related\_name ی باشد که در مدل post تعریف شده است

class Posts(models.Model):

    author = models.ForeignKey(Profiles , related\_name='posts' , on\_delete=models.CASCADE)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_