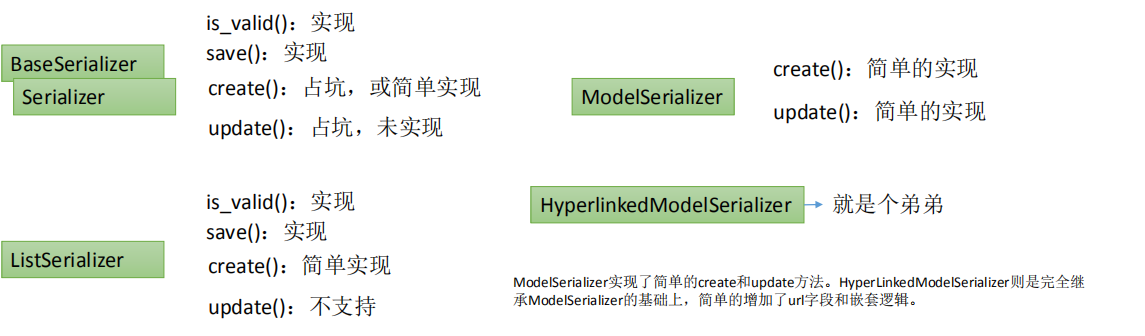
**反序列化：**

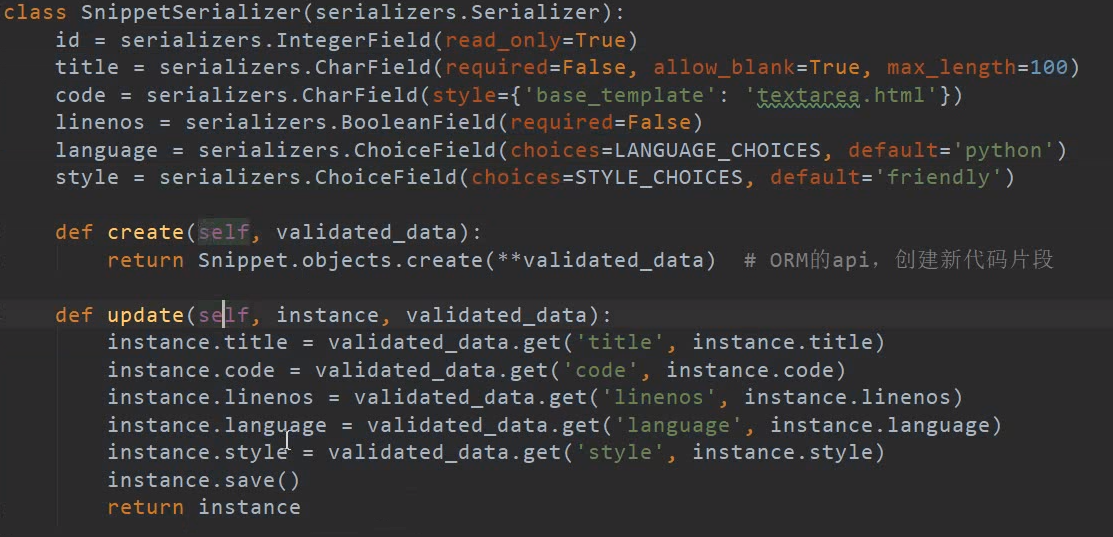
比如：前端通过post提交数据然后保存到后端数据库里面



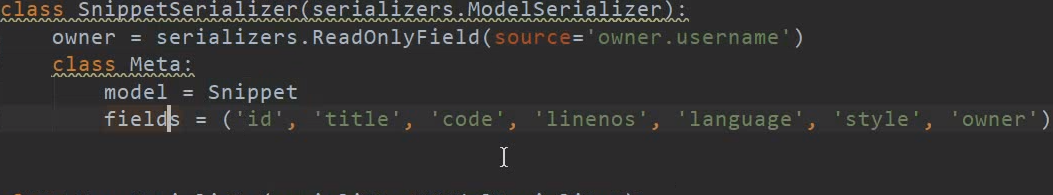
（POST）创建实例保存到数据库就需要实现create()方法：

（Put,Patch）创建实例保存到数据库就需要实现update()方法：

1. 继承Serializer类。需要写序列化字段，update(),create()



1. 继承ModelSerializer。不需要手写序列化model，直接指定需要序列化的模型(model)和字段,可以不写update(),create()。这里的owner不管



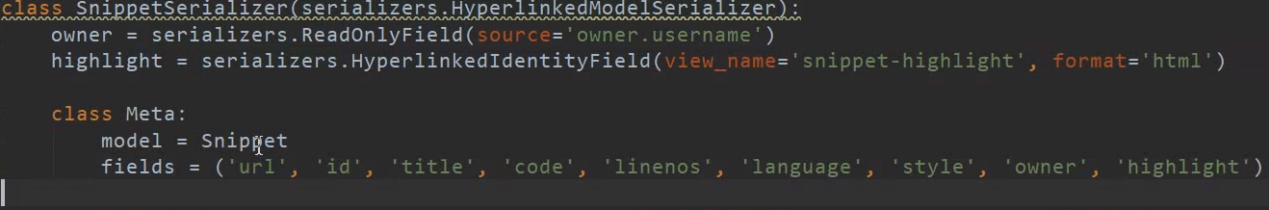
1. 继承HyperlinkedModelSerializer。

HyperlinkedModelSerializer 与 ModelSerializer 有以下区别：

默认情况下不包括 id 字段。

自带一个 url 字段，使用 HyperlinkedIdentityField 。

关联关系使用 HyperlinkedRelatedField 字段类型，而不是PrimaryKeyRelatedField 字段类型。



正向的序列化（从数据库取出数据返回给前端（如：get））时，不用再序列化类中写对应的方法（安全方法）因为不用对数据进行破坏性的修改

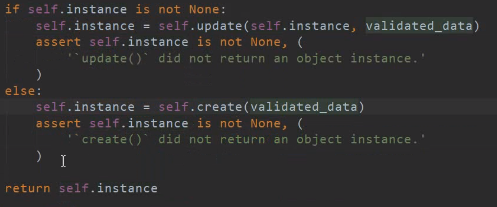
反向的序列化（如Post方法），需要进行如下标注的操作

在views中，如果继承了稍微基本点的视图如APIView就需要手写如下：

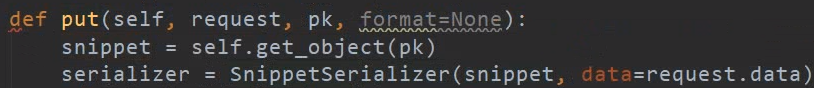


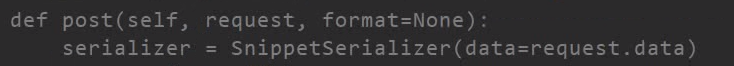
SnippetSerializer为自己写的序列化类，执行完is\_valid()方法之后会生成validated\_dat(合法的数据),在调用save()，通过ORM保存到数据库中

Serializer里面的save()方法的部分源码如下：

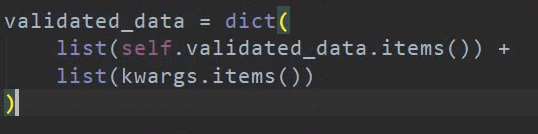


意思就是提供instance就是更新，没有就是创建



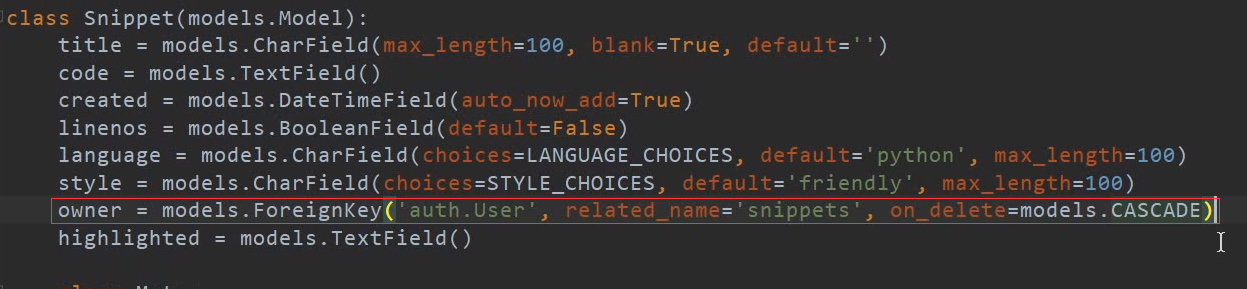


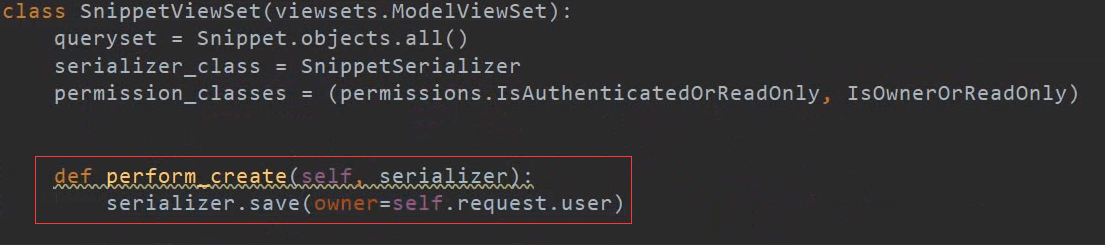
Serializer里面的save()方法的部分源码如下：

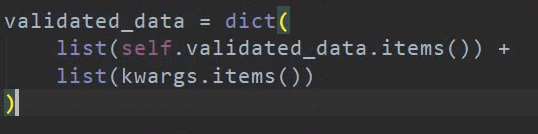


作用是：

我们在保存一个snippet对象的时候，我们对于snippet这个模型（如一图），owner是一个外键，我们在前端发送过来的数据的时候，并没有提供snippet所属片段人是谁（用的系统自带的User模型），但是我们在保存ORM外键时候的时候，必须提供owner字段的值，我们将在下图二位置提供，通过图三，把图二红色部分和合法验证的数据重新组成一个字典，然后通过save()方法保存到ORM里面，所以说save()方法可带参数

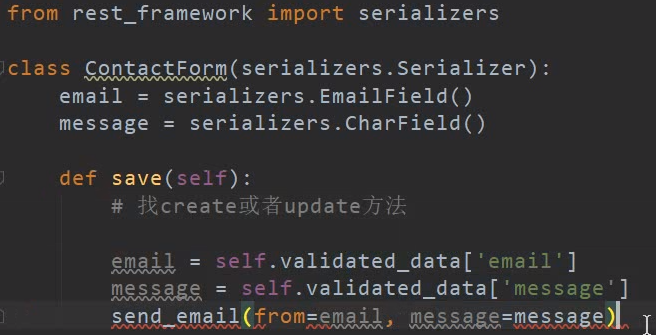


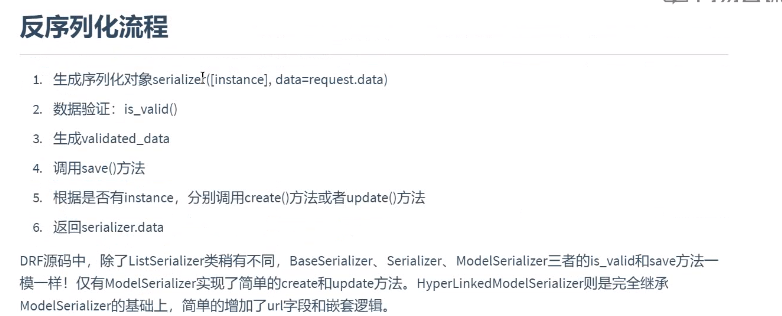




可以重写save()方法

如下是联系人重写save()实现简单的发送邮件



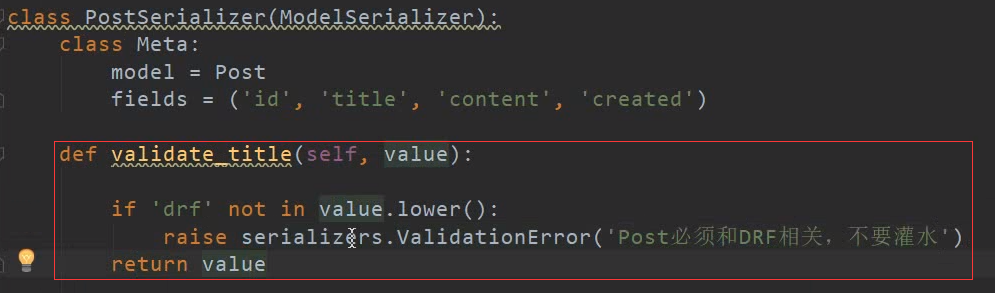


**自定义验证器：**

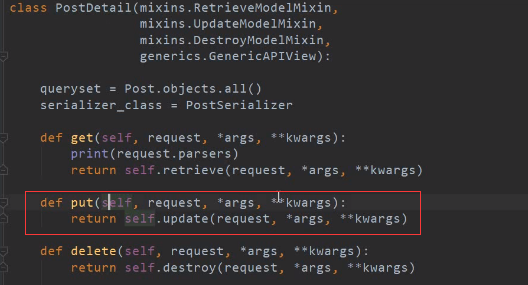
一：字段级别的验证

DRF定义了，如果你需要为某个字段单独的验证器需要这样命名：validate\_名字

1.在serializer类里面编辑：（value表示针对title里面的数据）

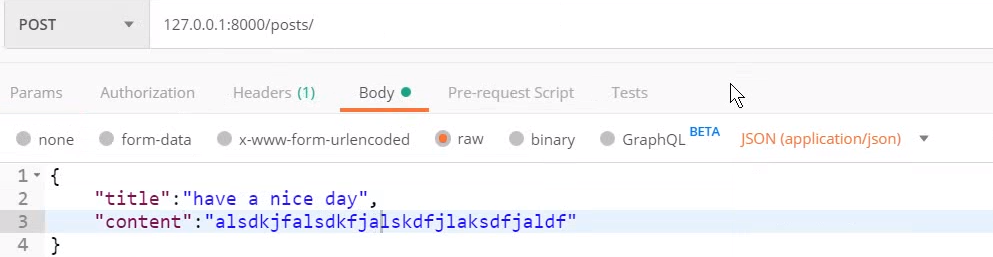


1. Views里面有put方法

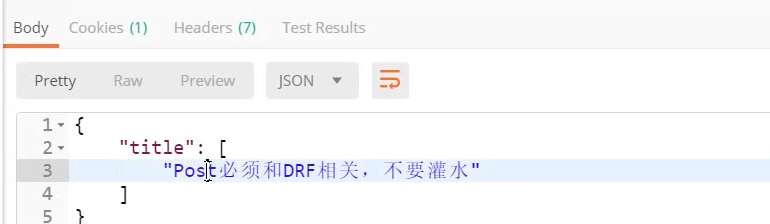


1. 测试

Postman

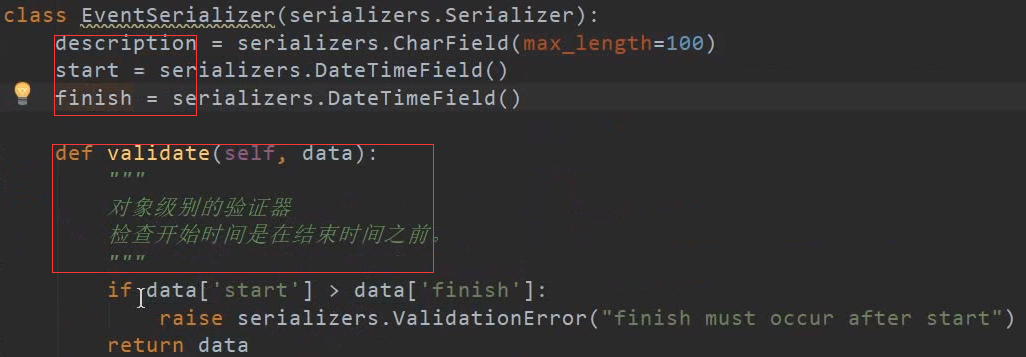


结果（因为title里面没有包含drf,所以未能正确发送）



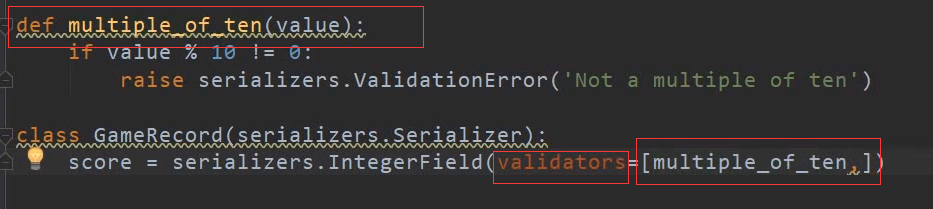
二：对象级别的验证（针对2个以上的字段）

1.在serializer类里面编辑：（data表示针对对象里面的所有数据）



三：以字段参数级别的验证器

1.在serializer类里面编辑：



1.参数名固定 2. 以列表的方式提供（可以有多个）

四：序列化里面的Meta类里面定义验证器（如下：具体后面介绍）

