

邹邹很busy。

被测试耽误的打工人。少年不努力，长大搞 IT。

博客园 首页 新随笔 联系 订阅 管理

DRF--ModelSerializer和时间格式化

前戏

在之前我们写序列化器的时候，写的很low，遇到反序列化的有时候还需要重写该字段，用post请求的时候，还要重写create方法，用put请求的时候，还需要重写update方法。总而言之，写起来很麻烦。来看看之前的是怎么写的

```
class BookSerializer(serializers.Serializer):
    id = serializers.IntegerField(required=False) # 只序列化，不走校验
    title = serializers.CharField(max_length=32, validators=[my_validate])
    pub_time = serializers.DateField()
    category = serializers.CharField(source="get_category_display", read_only=True) # 只序列化
    # 因为前端传的是数字，所以需要重写
    post_category = serializers.IntegerField(write_only=True) # 只反序列化用

    publisher = PublisherSerializer(read_only=True) # 一对多的表 只序列化用
    authors = AuthorSerializer(many=True, read_only=True) # 多对多的表需要指定many=True 只序列化

    publisher_id = serializers.IntegerField(write_only=True) # 只反序列化用
    author_list = serializers.ListField(write_only=True) # 只反序列化用

    def create(self, validated_data):
        # validated_data校验通过的数据
        # 通过ORM操作给book表增加数据
        book_obj = Book.objects.create(title=validated_data['title'],
                                       pub_time=validated_data['pub_time'],
                                       category=validated_data['post_category'],
                                       publisher_id=validated_data['publisher_id'])
        book_obj.authors.add(*validated_data['author_list']) # 这个参数可能是一个列表
        return book_obj

    def update(self, instance, validated_data):
        # instance 更新的book_obj对象
        # validated_data 校验通过的数据
        instance.title = validated_data.get("title",instance.title)
        instance.pub_time = validated_data.get("pub_time",instance.pub_time)
        instance.category = validated_data.get("post_category",instance.category)
        instance.publisher_id = validated_data.get("publisher_id",instance.publisher_id)
        if validated_data.get("author_list"): # 可能有多个值
            instance.author.set(validated_data["author_list"])
        instance.save() # 保存
        return instance

    def validate_title(self, value): # 对单一段校验
        if "BDYJY" not in value.upper():
            return value
        raise serializers.ValidationError('标题里含有非法字符') # 抛出错误

    def validate(self, attrs): # 对多个字段校验
        # attrs是一个字典，里面是传过来的所有字段
        if 'python' in attrs['title'].lower() and attrs['post_category']==1:
            return attrs
```

公告



昵称： 邹邹很busy。
园龄： 1年11个月
粉丝： 79
关注： 1
+加关注

2021年5月						
<	日	一	二	三	四	五
	25	26	27	28	29	30
	2	3	4	5	6	7
	9	10	11	12	13	14
	16	17	18	19	20	21
	23	24	25	26	27	28
	30	31	1	2	3	4

搜索

找找看

谷歌搜索

常用链接

- 我的随笔
- 我的评论
- 我的参与
- 最新评论
- 我的标签

随笔分类

- adb&monkey(4)
- appium(16)
- celery(4)
- CSS(7)
- Django(22)
- docker(14)
- DRF(15)
- git(7)
- HTML(5)
- javascript(7)
- JMeter(7)
- Linux(13)
- mysql(3)

else:
 raise serializers.ValidationError('传的参数有误, 请重新上传')

from django.db import models

Create your models here.

__all__ = ["Book", "Publisher", "Author"]

class Book(models.Model):
 title = models.CharField(max_length=32)
 CHOICES = ((1, "python"), (2, "Liinux"), (3, "Go"))
 category = models.IntegerField(choices=CHOICES)
 pub_time = models.DateField()
 publisher = models.ForeignKey(to="Publisher")
 authors = models.ManyToManyField(to="Author")

class Publisher(models.Model):
 title = models.CharField(max_length=32)

 def __str__(self):
 return self.title

class Author(models.Model):
 name = models.CharField(max_length=32)

 def __str__(self):
 return self.name

ModelSerializer

既然上面的写法很low，DRF提供给了我们ModelSerializer，之前序列化器继承的是serializers.Serializer，现在要继承serializers.ModelSerializer

把之前的BookSerializer类里的东西删掉重写

class BookSerializer(serializers.ModelSerializer):

 class Meta:
 model = Book # 对应的表名
 fields = "__all__" # 显示所有的字段
 # fields = ['id','title'] # 显示指定的字段
 # exclude = ['id', 'title'] # 不显示指定的字段

这样我们访问这个接口查看数据

nginx(5)
postman(9)
更多

随笔档案

2021年5月(4)

2021年4月(7)

2021年3月(6)

2021年2月(3)

2021年1月(3)

2020年12月(6)

2020年11月(11)

2020年10月(6)

2020年9月(3)

2020年8月(9)

2020年7月(11)

2020年6月(7)

2020年5月(5)

2020年4月(8)

2020年3月(10)

更多

最新评论

1. Re:docker--部署vue项目
讲的太清楚了!!! 谢谢!!!
--勿忘归来

2. Re:docker--部署vue项目
想请教下, 怎么修改配置文件呢, 比如要隐藏版本号
dockerfile里要如何操作呢
--wufanqie

3. Re:Vue--ElementUI实现退出功能
@BlackCatFish 已经部署到服务器: ...
--邹邹很busy。

4. Re:Vue--ElementUI实现退出功能
有效果图更好
--BlackCatFish

5. Re:Vue--ElementUI实现退出功能
有源码就好了~
--心素如简

阅读排行榜

1. postman使用--添加headers、授权、cookies(24198)

2. Vue--登录页面(17078)

3. docker--部署vue项目(12991)

4. selenium--更改标签的属性值(7627)

5. selenium--浏览器滚动条操作(6276)

评论排行榜

1. python--装饰器(4)

2. Vue--ElementUI实现退出功能(3)

3. docker--部署vue项目(3)

4. Vue--ElementUI实现主页面横向导航(1)

5. Vue--ElementUI实现头部组件和左侧组件效果(1)

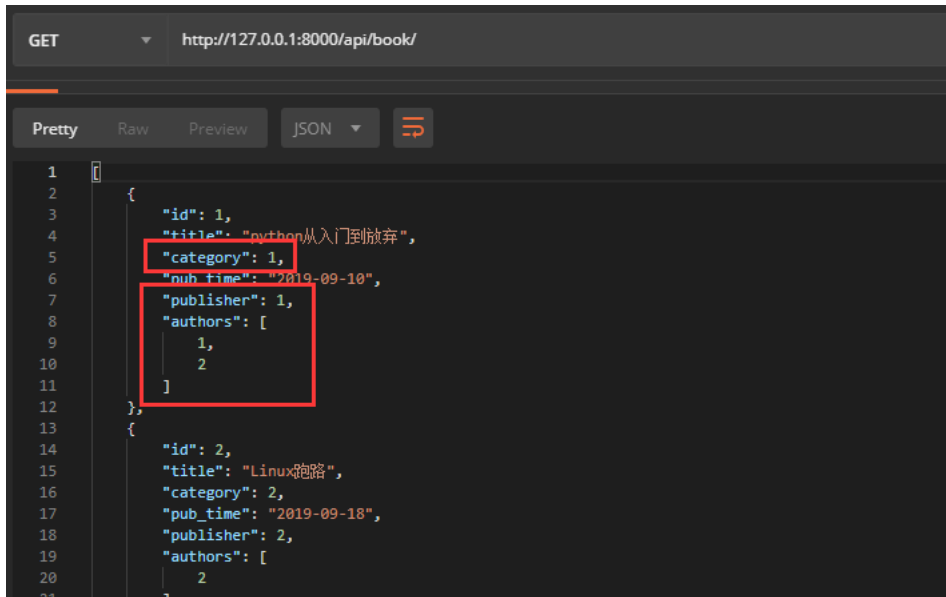
推荐排行榜

1. selenium--Xpath定位(2)

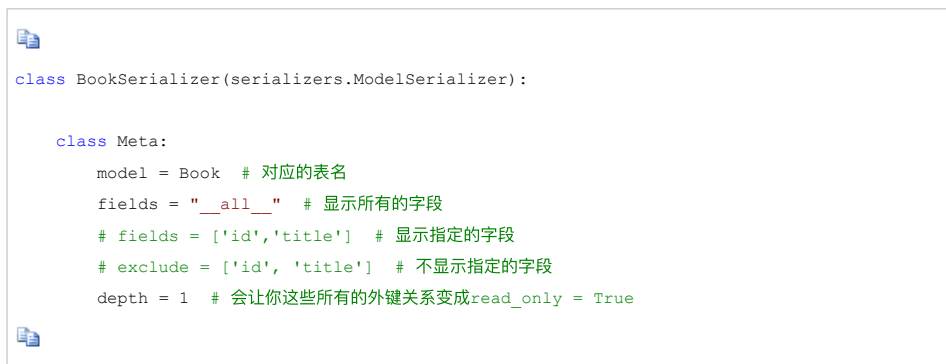
2. Vue--ElementUI实现主页面横向导航(1)

3. docker--部署vue项目(1)

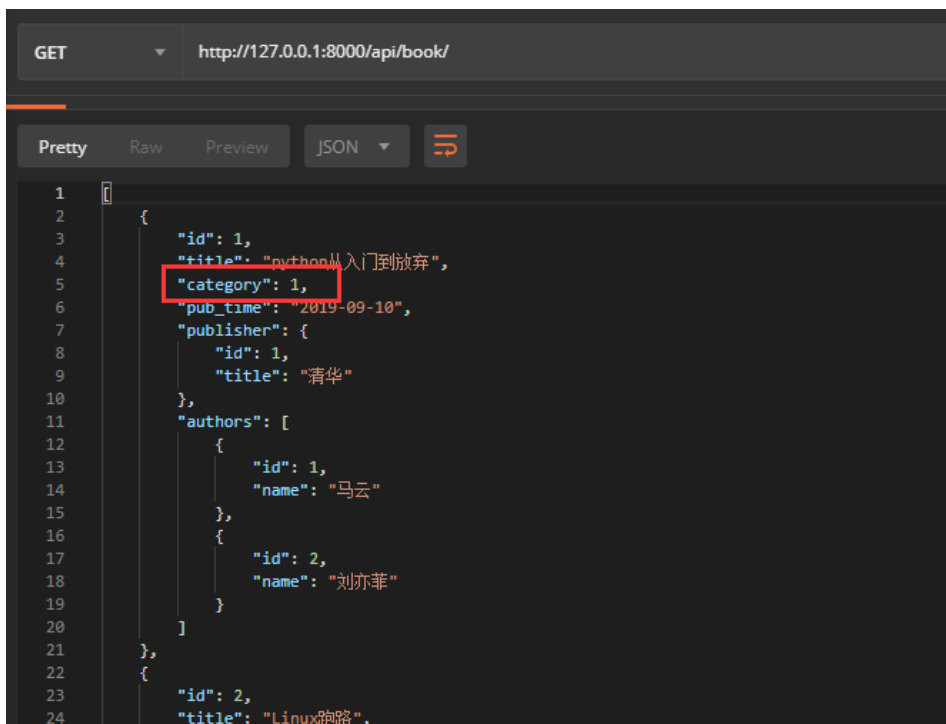
4. Vue--登录页面(1)
5. Vue--封装axios、跨域(1)



会发现Chiose字段和关联表的字段不是我们想要的，我们可以加上depth来让它显示成为我们想要的值，后面的值是表示几层，因为book表和author, publisher表都是一层关联，另外两者表没有一对多和多对一的关系，所以我们可以写成depth=1



再来请求下这个接口



这样我们虽然解决了一对一和多对多的字段，但是Chiose字段还不是显示我们想要的数，而且depth会让你这些所有的外键关系变成read_only = True，所以实际开发中都不用，都是重写

重写字段

```
class BookSerializer(serializers.ModelSerializer):
    # 重写publisher字段
    publisher_info = serializers.SerializerMethodField()
    author_info = serializers.SerializerMethodField()

    def get_publisher_info(self, obj):
        # 函数名为 get_自定义的字段名
        # return了一个'ok',表示上面的publisher_info='ok', 会返回给前端
        return 'ok'

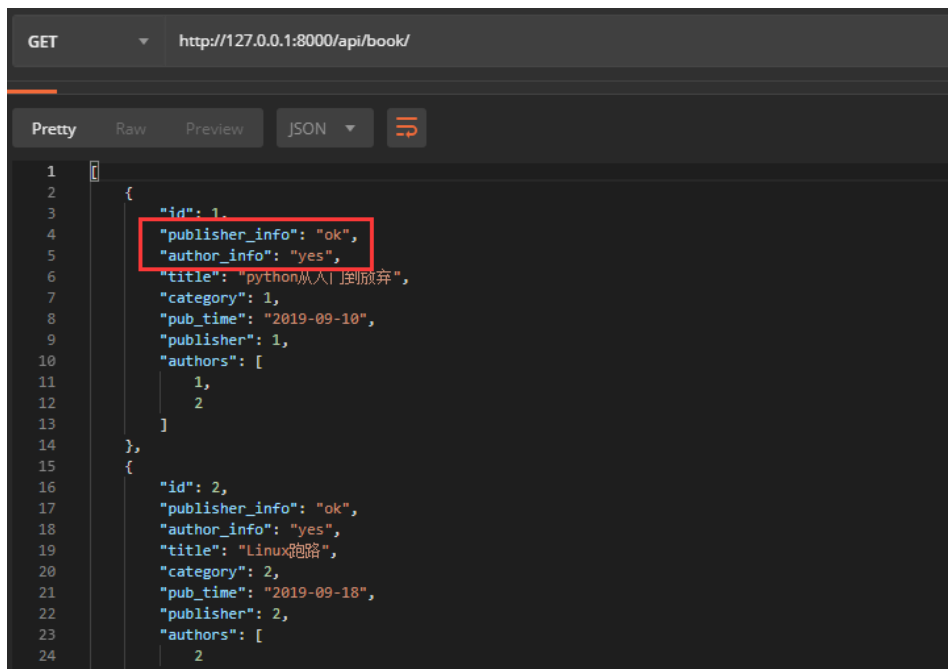
    def get_author_info(self, obj):
        return 'yes'

    class Meta:
        model = Book # 对应的表名
        fields = "__all__" # 显示所有的字段
        # fields = ['id','title'] # 显示指定的字段
        # exclude = ['id', 'title'] # 不显示指定的字段
```

其中的obj就是我们在views.py里序列化的每个对象，也就是下面的book_queryset

```
ser_obj = BookSerializer(book_queryset, many=True)
```

请求接口



```
GET http://127.0.0.1:8000/api/book/

Pretty Raw Preview JSON

1 {
2   {
3     "id": 1,
4     "publisher_info": "ok",
5     "author_info": "yes",
6     "title": "python从入门到放弃",
7     "category": 1,
8     "pub_time": "2019-09-10",
9     "publisher": 1,
10    "authors": [
11      1,
12      2
13    ]
14  },
15  {
16    "id": 2,
17    "publisher_info": "ok",
18    "author_info": "yes",
19    "title": "Linux跑路",
20    "category": 2,
21    "pub_time": "2019-09-18",
22    "publisher": 2,
23    "authors": [
24      2
25    ]
26  }
27 }
```

会发现我们重写数据都返回给了调用这个接口的，所以我们可以自定义需要返回哪些字段。

```
class BookSerializer(serializers.ModelSerializer):
    # 重写publisher字段
    publisher_info = serializers.SerializerMethodField()
    author_info = serializers.SerializerMethodField()
    category_display = serializers.SerializerMethodField()

    def get_publisher_info(self, obj):
        # 函数名为 get_自定义的字段名
        publisher_obj = obj.publisher # ForeignKey, 拿到的是publisher表的对象
        return {"id": publisher_obj.id, 'title': publisher_obj.title}

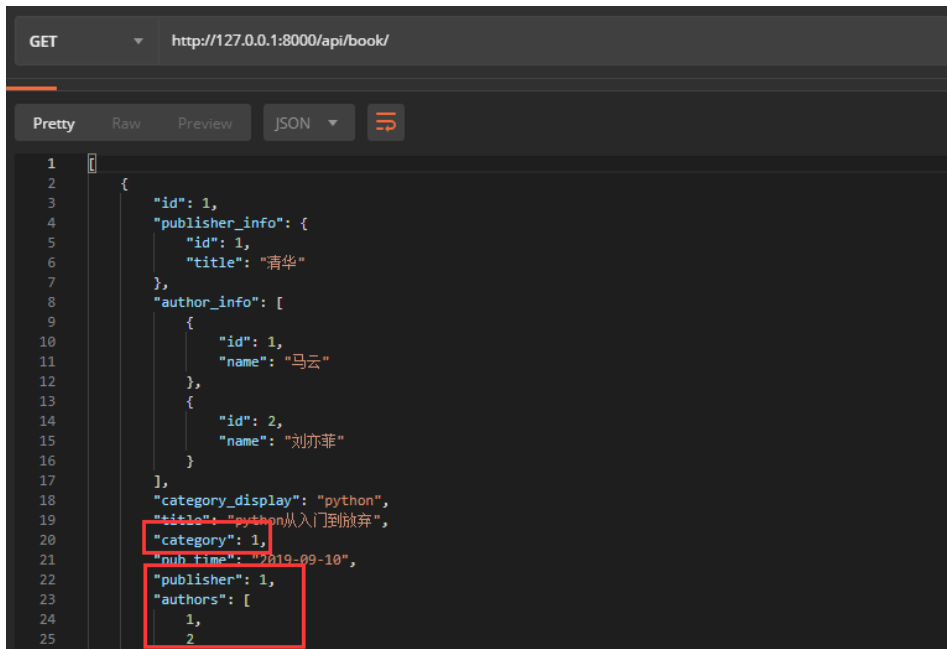
    def get_author_info(self, obj):
        authors_queryset = obj.authors.all() # ManyToMany, .all拿到的是queryset里所有的对象
        return [{"id": author.id, 'name': author.name} for author in authors_queryset]
```

```
def get_category_display(self, obj):
    return obj.get_category_display() # 调用这个方法, 会返回汉字

class Meta:
    model = Book # 对应的表名
    fields = "__all__" # 显示所有的字段
    # fields = ['id', 'title'] # 显示指定的字段
    # exclude = ['id', 'title'] # 不显示指定的字段
```



`publisher_info = serializers.SerializerMethodField()` 叫做方法字段, 需要一个方法, 把方法里的返回值赋值给该字段



上面我们自定义的字段已经显示了, 然而之前的字段我们可以让它不显示, 在Meta加个 `extra_kwargs`, 里面的是个字典, key为不显示的字段名, value如果是`write_only=True`, 就不显示了



```
class BookSerializer(serializers.ModelSerializer):
    # 重写publisher字段
    publisher_info = serializers.SerializerMethodField(read_only=True)
    author_info = serializers.SerializerMethodField(read_only=True)
    category_display = serializers.SerializerMethodField(read_only=True) # Choise字段, 后面

    def get_publisher_info(self, obj):
        # 函数名为 get_自定义的字段名
        publisher_obj = obj.publisher
        return {"id": publisher_obj.id, "title": publisher_obj.title}

    def get_author_info(self, obj):
        authors_queryset = obj.authors.all()
        return [{"id": author.id, "name": author.name} for author in authors_queryset]

    def get_category_display(self, obj):
        return obj.get_category_display()

    class Meta:
        model = Book # 对应的表名
        fields = "__all__" # 显示所有的字段
        # fields = ['id', 'title'] # 显示指定的字段
        # exclude = ['id', 'title'] # 不显示指定的字段
        extra_kwargs = {'category': {"write_only": True}, 'publisher': {"write_only": True},
                        'authors': {"write_only": True}}
```



这样，当我们使用post或者put方法时，就不需要再重写create方法和update方法了

在函数里重写方法字段时，里面写了read_only=True，表示序列化时使用，而在Meta类里，默认的字
段category，publisher，author为write_only，所以序列化的时候，这些字段不显示，显示的是重写的
那些字段。反序列化时用的是extra_kwargs里的字段

时间格式化

如果我们在创建时间的时候，使用的是DateTimeField，类似于下面这样

```
update_time = models.DateTimeField(auto_now=True, verbose_name="更新时间", help_text="更新时间")
```

如果想返回的时候，直接格式化好，我们就可以在序列化的时候加上这句就可以了

```
update_time = serializers.DateTimeField(format="%Y-%m-%d %H:%M:%S", required=False, read_on
```

```
class ProjectSerializer(serializers.ModelSerializer):  
    """  
    项目信息序列化  
    """  
    update_time = serializers.DateTimeField(format="%Y-%m-%d %H:%M:%S", required=False, read_only=True)  
  
    class Meta:  
        model = Projects  
        fields = '__all__'
```

但是只这样修改，返回的时间是UTC的时间，我们还需要在settings.py里更改，将原来的UTC时间改
为国内的时间，找到LANGUAGE_CODE和TIME_ZONE，改为中国上海的时间，如下

```
LANGUAGE_CODE = 'zh-hans'  
  
TIME_ZONE = 'Asia/Shanghai'
```

总结

ModelSerializer

默认生成关联的模型表里的所有字段

配置

```
class Meta:  
    model=表名  
    fields="__all__"["字段名",]  
    exclude=["字段名",]  
    depth=1 外键关系找一层 会让外键关系字段变成read_only=True  
    extra_kwargs={"字段名": {配置的属性}}
```

SerializerMethodField()

方法字段 会调用它自己的钩子方法，把方法的返回值给字段

```
def get_字段名(self, obj):  
    循环序列化的每个模型对象就是obj  
    对obj进行ORM操作  
    return 自定义的数据结构
```

分类: [DRF](#)

好文要顶

关注我

收藏该文



邹邹很busy_

关注 - 1

粉丝 - 79

+加关注

0

0





« 上一篇: [DRF--验证器](#)

» 下一篇: [DRF--重写views](#)


刷新评论 刷新页面 返回顶部

发表评论

编辑 预览

B    

支持 Markdown

 自动补全

提交评论 退出 订阅评论 我的博客

[Ctrl+Enter快捷键提交]

- 【推荐】云上创新 | 2021阿里云峰会28-29日直播预告，敬请期待！
- 【推荐】大型组态、工控、仿真、CAD\GIS 50万行VC++源码免费下载！
- 【推荐】100个HarmonyOS 2.0开发者Beta公测名额，限时认领！
- 【推荐】阿里云爆品销量榜单出炉，精选爆款产品低至0.55折
- 【推荐】限时秒杀！国云大数据魔镜，企业级云分析平台

园子动态：

- 致园友们的一封检讨书：都是我们的错
- 数据库实例 CPU 100% 引发全站故障
- 发起一个开源项目：博客引擎 fluss

最新新闻：

- 华为：首台HI版极狐阿尔法S生产线验证样车下线
 - 史上最强Note！Redmi Note 10 Pro开箱图赏：1499元用上67W快充
 - 网易云音乐将赴港上市：去年收入49亿元 月活跃用户数1.805亿人
 - 跟紧苹果脚步！谷歌或通过三种方案改进Android隐私设置
 - 吉利投资成立新能源汽车科技公司，注册资本4亿
- » 更多新闻...