做之前先想想，不要直接看答案，那样没有自己的体会和思考，那样学到的知识不牢固

8-friendships-model-api-and-tests

1. friendships model create
    - follower 继承自User
    - following 继承者User
    - create_at
    - update_at
    - __str__方法 {follower} follow {following} at {update_time}
2. FriendShipViewSet
    - 继承谁？ModelViewSet,还是GenericViewSet+mixins?
    - 需要list吗 ，这是查询接口的话不需要展示一堆关注关系吧
    - 需要create吗，如果需要就要用SerializerForCreateFriendShip
3. FriendShipCreateSerializer
    - 继承什么？ModelSerializer?
    - 验证什么字段，follow是外键怎么验证
    - 需要如何处理validate(self, data)方法
    - 之后是不是要调用create方法创建关注关系？
4. 如何设计FriendshipViewSet测试用例
    - 创建2个用户，然后A关注B
    - 再创建2个用户，然后C,D关注A,查询A的followers,即following=A.id,查询C的following,即follower=C.id
    - A取消关注B
    - 关注需要登录，所以创建至少一个已登录用户，另创建5个用户，无需登录，只要id就行

对照老师的最佳实践:

我没考虑到	我也考虑到了
✘	✔

1. FriendShipViewSet 应该实现4个功能

FriendShipViewSet功能点	对照老师的最佳实践
查看user关注了哪些人	✘
查看哪些人关注了user	✘
关注user A到user B的好友关系	✔
取消 user A到 user B的好友关系	✘

2. 对FriendShip模型的理解之前有误区，FriendShipViewSet不只是查看某个用户的follower和following,也可以创建和取消friendship关系

3. 因为不是标准的RESTFul风格,所以要用action额外动作来定制 ❌

4. FriendShipSerializer共有3种

Serializer	**对照老师的最佳实践**
FollowerSerializer	❌
FollowingSerializer	❌
FriendShipForCreateSerializer	✔

5. 如果待创建的关注关系已存在，做忽略处理，不用报错 ❌

6. 模型中级联操作设置为SET_NULL防止删除时引发危险的操作 ❌

7. 测试的思路

tests	**老师设计的测试点**	**对照老师的最佳实践**
SetUp	创建匿名客户端	❌
	FriendShip 模型创建follower/following	❌
	创建两个认证客户端	✔
test_follow	请求方式get	❌
	不登录直接关注	❌
	自己关注自己	❌
	关注自己	❌
	正常关注	✔
	关注者数量+1	❌
test_follower	post方式	❌
	匿名用户	❌
	验证时间排序	❌
	验证帖子的内容排序	❌
test_following	post方式	❌
	匿名用户	❌
	验证时间排序	❌
	验证帖子的内容排序	❌

9-newsfeed-model-api-tests

预习不代表空想，预习就是看第一遍有什么感想，知道哪些不足

1. newsfeed是 给用户的追随者们发送tweet，发送的范围是用户的follower
2. bulk_creat 批量创建
3. services处理内部逻辑，让代码更整洁
4. prefetch_relate怎么用
5. NewsFeed(user=tweet.user)又要同时NewsFeed(user=follower),这两个为什么一起存储
6. NewsFeed视图的权限设计
7. newsfeed工作方式，follower用户查询自己的newsfeed, following的用户负责发推，在tweetViewSet的create视图中， 通过FriendShip找到所有的followers,并把tweet文存储到这些followers的NewsFeed里，最终每个用户就能看到自己的newsfeed
8. 实例级别的属性懒加载

```python
from rest_framework.test import APIClient
class A:
    @property
    def anonymous(self):
        if hasattr(self,'_anonymous'):
            return self.anonymous
        self._anonymous = APIClient()
        return self._anonymous
```

10-comments-model-admin

1. 一个评论里包含了哪些要素呢
    - user,评论者
    - tweet,评论的推文
    - created_at,评论发表的时间
    - update_at , 修改评论的时间
    - content,评论的内容

11-comment-create-api

1. list接口如何传递tweet参数呢，要不要传tweet？想不明白
2. request参数需要封装给serializer吗
3. get_permission方法里面if条件如何写，is_authenticated从何而来，request好像也没有
4. Q: 为什么不需要list方法，难道不是需要展示推文下所有的评论吗？
    - A: 母鸡
5. Q: 测试的时候如何给comment传tweet,先要查询吗?
    - A: 先创建一个tweet实例

12-comment-update-and-destroy-api

1. Q:自定义permission类如何具体发挥作用？
2. update和delete需要tweet的instance对象，通过self.get_object()方法获取. ❌
    - update和delete获取的是Comment对象. ☑
3. Q:如何验证update的内容? 🚫
    - 根本不需要验证🚫， 直接在update方法里判断并保存 ☑ 是不是有点神奇，侧面说明我忘了前天学习的内容了 👀

13-comments-list-api

1. 怎么导入local settings这段代码怎么理解？

```
#settings.py
try:
   form .local_settings import *
except:
   pass
```

2. django-filter的作用推测应该是根据字段过滤queryset

14-tweet-retrieve-api-with-comments

1. 这个分支重点是decorator,意图在tweet下带出所有的comment

15-likes-model-and-api

1. 为什么 Tweet 和 Comment 也是可以加like的，怎么加？通过ContentType黑科技关联
2. Q:ContentType是外键吗? 为什么可以关联任意的模型，即使like没有定义comment的外键？如何使用代码里已经说明。
    - A:根据chatGpt的回答，我又有了新的认识
    - A:ContentType关联的是model的名称和所属的app_label， object_id是关联的model主键id,联合起来就可以找到某个模型的某一条记录。
    - A:这是动态查找关联的，而不是数据库级别的约束，灵活关联任意的模型，无需硬编码模型的外键定义。
    - A&Q:这就解释了like_set定义到comment和tweet模型，用来关联用户的tweet/comment的like数量
      Q:新的疑问，用户的点赞api是如何实现的？
        - A:
3. 测试完全没有思维，如何测试我的model?如何创建一个like?
    - 需要tweet_id或者comment_id吧 ❌
    - content_type参数应该传递的是模型实例，而不是模型本身 ✔
    - like创建时传入任意实例，不管是tweet还是comment，或更多其他的，target就能解决一切 ✔

16-likes-create-api

1. 看了一下系统设计，里面第一章是关于的twitter的，要好好再复习一下，对项目架构有所把握，而不是只写代码，不见泰山

18-inject-like-infos-to-other-api

1. 改动了哪些Serializer
    - `CommentSerializer` 添加了方法字段
        - `like_count`
        - `has_liked`
    - `LikeService`
    - `newsfeedViewSet` 添加了 context={request:'request'},因为?
    - `TweetSerializer` 添加了方法字段

        - `comment_count` `comment_set`默认反向关联
        - `like_count`
        - `has_liked` 调用LikeService
        - 子类 `TweetSerializerForDetail`
            - `likes` 调用`source='like_set'`属性反向查询tweet的所有点赞
        - 删除了`CreateTweetSerializer`,替换为 `TweetSerializerForCreate`
2. 添加了哪些测试
    - tweet的view要测试`response.data`: comment_count,like_count,has_like
    - comment的view要测试`response.data`: like_count,has_liked