断点调试的方式适合初学者，高效率的方式是打印日志。

打印日志的代码不需要删除，不必使用assert，assert一般用于unittest。

Django rest framework，简称drf

将和drf API有关一般放到api文件夹下。

联合索引：index\_together

index会存在包含关系，例如建联合索引(‘a’, ‘b’, ‘c’)后，不需要再建(‘a’, ‘b’)。

Mysql，Primary key使用自增主键id，一般不是uuid。

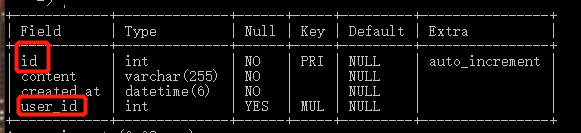
uuid比较适合分布式数据库，它是全球唯一的。

Django中创建了Tweet模型后，通过浏览器访问服务器时，发现其并不生效，这是因为没有修改数据库的原因。需要执行

python manage.py makemigrations

python mange.py migrate

执行后，查看mysql数据库：



即使是在一个app下，也可能存在多个tests.py用于测试，tweets下面有tests.py，它的api文件夹下也有tests.py。

Django中如果模型中创建了新的索引，也需要更新数据库，使用makemigrations和migrate

ModelViewSet尽可能不用，因为它默认提供了增删查改数据库的功能，但大多数情况下，设计的ViewSet其实并不具有这些权限。

设计Tweet相关的API

TweetViewSet，rest framework中ViewSet不再提供Django中View的get()和post()，取代的是list()和create()

接口list是获取tweet列表，访问时需要通过?操作符传入user\_id，因为我们不会要求列出所有tweet，而是某个用户的tweet

/api/tweets/user\_id=4

在浏览器执行：http://localhost/api/tweets/?user\_id=3

下图中加红框的部分，如果TweetSerializer中没有添加user = UserSerializer()，就不会出现，而是会变成”user”: 3，也就是只显示user\_id。



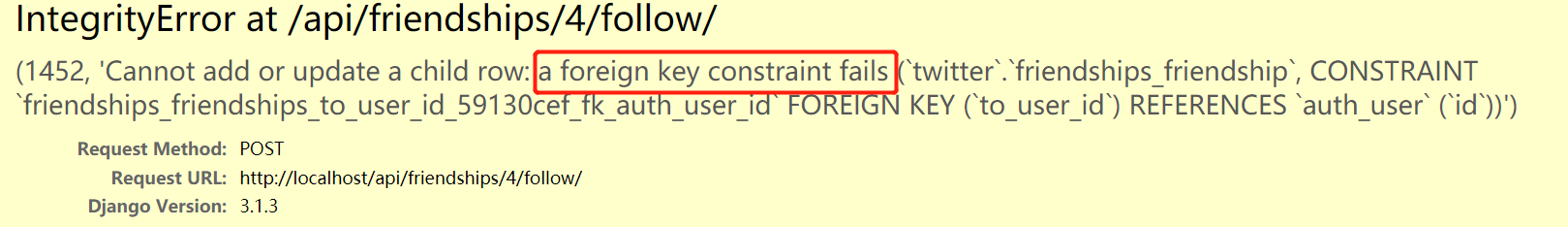
Django中unittest指定的测试文件：

python manage.py test tweets.api.tests -v2

更进一步只测试某个函数：

python manage.py test tweets.api.tests.TweetApiTests.test\_list\_api

Friendship模块中，在浏览器测试follow时，如果指定的to\_user\_id不存在，就会抛出外键失败的错误：



MySQL使用：

1. 不要使用JOIN
2. 不要用CASCADE
3. 数据量比较大的话，删除外键

Friendship模块中，可以把followers和followings都整合到list中

URL就是/api/friendships/?type=follower&to\_user\_id=1

FriendshipService中get\_followers()方法中，N+1次Queries产生的原因：

friendships = Friendship.objects.filter(to\_user=user)

return [friendship.from\_user for friendship in friendships]

queryset是懒惰加载的，执行第一行代码时不会查询数据库，第二行代码对friendships进行迭代时，就会查询数据库得到所有to\_user=user的friendships，这是1次Query，之所以后面还有N次Queries，是因为from\_user是friendship表单中的外键，数据库表单中存储的是from\_user\_id，所以迭代执行friendship.from\_user时还要根据from\_user\_id从User表单中查找User对象，因此产生了N次Queries。如果访问的字段是friendship表单中其它非外键的字段，其实这个for循环不会产生N次Queries。

NewsFeedService和FriendshipService中错误的写法可以之后验证下效率。