为什么要有通知模块，而不是和新鲜事模块合并？

定义不同，应用场景不同。

不要自己新造轮子，通知模块是一个非常常用的模块，可以在github上去搜索。

站内通知：Twitter手机端，网站上有类似一个铃铛标志的通知提醒。

，你的tweet，comment被谁点赞了，你的tweet被谁评论了，这个会存储在数据库中。

django-notifications和django-push-notifications

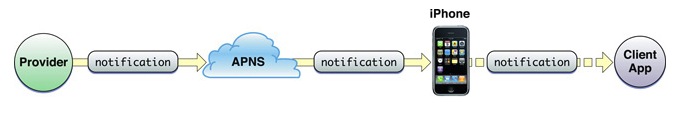
notifications主要是app内部的消息推送，比如twitter上有人点了你的赞，评论了你的tweet，这些消息通常要存储在数据表中；而push notifications则是针对设备的消息推送，例如吴某凡跪了，手机的下拉屏幕上会出现某个app或多个app相应的新闻，这些消息通常不会存储下来，关闭后就没有了。

push notifications的实现：

苹果的APNS推送：使用苹果的APNS，是苹果的推送服务器效率高，稳定性高，不要自己造轮子，增加自己的工作。

1. 装有XXX应用的设备首先需要向APNs服务器注册，注册成功后APNs服务器会返给我们一个device\_token，并维持一个和APNS的socket长链接，拿到这个token后我们将这个token发给我们自己的XXX应用服务器。
2. 当有消息需要被推送时，XXX应用服务器会将消息按指定的格式结合设备的deviceToken一并打包 然后发给APNs服务器。
3. APNs将新消息推送给我们的设备上，然后就在设备的屏幕上显示出来了 （因为我们的设备和APNs服务器二者之间维持了一个长连接）

deviceToken是一个APP装在一个设备上的唯一标识符。一个APP在不同设备上的deviceToken不同，同一个设备卸载后重新安装的APP，deviceToken也不同。



APNS推送消息的时长都是秒级，最慢的有几个小时，但推送消息通常不需要那么快，而且越快越好资源。

设备卸载了XXX应用后，苹果APNS上会有一个Feedback Service，当我们的Provider将信息发给APNs服务器，APNs推送信息到我们的设备时，如果这时设备无法将消息推送到指定的应用（应用已经删除），就会向APNs服务器发送一个反馈信息，而这个信息就记录在Feedback service中。按照这种方式，Provider应该定时的去检测Feedback service的列表，然后删除在自己数据库中记录的存在于反馈列表中的deviceToken，从而不再向这些设备发送推送信息。

即时通讯不会走上述的推送流程，而是APP自己的一套推送服务。

<https://blog.csdn.net/weixin_43217710/article/details/90319309>

django-notifications中的send方法

from notifications.signals import notify

notify.send(actor, recipient, verb, action\_object, target, level, description, public, timestamp, \*\*kwargs)

参数：

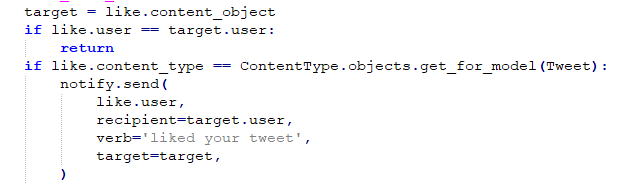
actor：发送通知的对象[必需]，注意：如果使用关键字参数，使用sender，即send(sender:xxx)，而不是actor

recipient：接收通知对象[必需]

verb：动词短语[必需]，告知recipient对象actor干了什么

action\_object：链接到动作的对象[可选]，相当于下图中的like对象

target：被发送通知的对象[可选]，例如当点赞了某条tweet时，actor就是点赞的user，recipient就是tweet的user，而target就是点赞的tweet。



level：通知等级[可选]，有4种(success', 'info', 'warning', 'error')，默认为info

description：描述字符串[可选]

public：是否公开[可选]，默认为True

timestamp：时间戳[可选]，默认为当前时间



verb动词短语有一些小技巧，例如：

给你的帖子点了赞，前端进行渲染时，希望把tweet对应的超链接渲染到“给你的帖子”后面，“点了赞”的前面，这时可以和前端约定：



前端拿到verb后，将其中的{target}替换成想渲染的超链接，并且能支持英文：



后端一般不处理具体渲染的对象，因为前端有可能是Web端，也有可能是手机端，跳转的方式不一致，因此需要足够的灵活性，后端之负责提供好数据。

工程中inbox是notifications的别名，notifications已经被第三方的库占用了。

django中的Signal机制和Qt的信号与槽机制类似，二者之间应该有区别，否则不会有在Python中调用Qt的信号与槽的方法。

django-notifications中自带的url不太符合restful规范，需要自己实现。

django中有自带的用户模块，但它有些属性没有包含，例如昵称，头像等，所以需要添加这些属性，有两种方式，一种方式是直接继承django中的AbstractUser类，然后再修改配置文件，定义AUTH\_USER\_MODEL字段，这种方式不建议，一是因为用户自定义的Model一般都直接继承models.Model就行了，避免过深的继承。二是之前使用了User类，如果重新自定义User类，有可能会产生数据不兼容的问题。三是根据“劝分不劝和”的原则，User类中原本的username，password，email通常是登陆时使用，而昵称，头像则是展示用户时，二者的应用场景不同，而且前者不太经常改，而后者则有可能经常改。另一种方式是增加一个UserProfile模型，存放新增的字段。这样的好处是二者可以分开存储，尤其是如果使用缓存cache时，二者放在不同的地方，由于UserProfile会经常更改，这样cache中会经常失效，但不会影响User的cache。

Web Server可能有很多台，所以上传的头像图片上不能传到Web Server服务器上，否则登陆到这台服务器，而头像上传到另一台服务器上，就会出现找不到的问题。

文件应该存储在一个中心服务器上，这台服务器就只存文件，类似一个云文件系统，常见的有Amazon S3和aliyun oss

yaml用来存配置，可以手工编辑，不容易出错，一行一个配置项，而且是key:value形式

json：web端数据传输，尤其是api response的标准格式，非常的轻量级。

xml用来存配置，很容易丢失tag，导致出错，逐渐被淘汰，它的体积比较大。

上传头像时，默认情况下会放到工程文件夹下，显然不符合我们的要求，需要进行设置上传文件的存放路径。

CDN(Content Delivery Network)：内容分发网络。Web系统的文件通常存储在Amason S3或aliyun oss，这是文件存储的大后方。但由于用户可能在不同的国家和地区，如果都之间访问文件系统，aliyun oss或Amason S3，那么会出现有些用户很快，有些用户很慢的现象，这时就需要CDN。它相当于一个cache，用户访问时，首先看看CDN上有没有相应的图片，如果没有就回源到云文件系统上，一个CDN的好坏就看其回源率的高低，越低越好。与cache不同的是，CDN并不全是懒惰加载，有时云文件系统会主动把一些文件通过脚本传输到各个CDN上。

发tweet成功，但fanout失败。如果粉丝很多，fanout时可以拆分成1000条这样分批执行，这样一组失败，不会所有都失败。另外可以加上log，然后根据脚本进行重试机制。

django的queryset以下三种写法是兼容的：增加灵活性

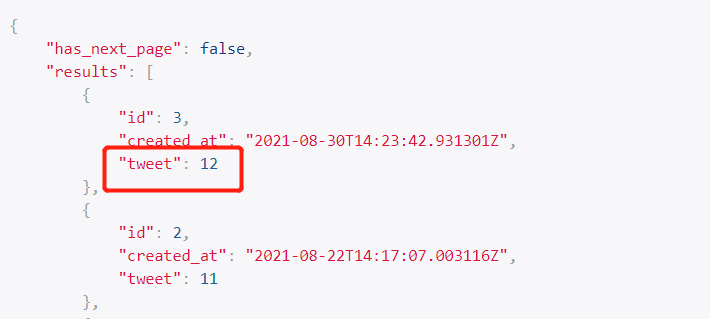
其中tweet是Coment的ForeignKey

Comment.objects.filter(tweet=1)

Comment.objects.filter(tweet\_id=1)

Comment.objects.filter(tweet=tweet\_object)

但在Drf中没有去兼容很好的\_id，在NewsfeedSerializer中，如果屏蔽掉tweet字段，http://localhost/api/newsfeeds/展示，可以看出展示的字段就是tweet，如果希望展示tweet\_id，就需要把fields中的字段修改为tweet\_id。



手机号码telephone\_number目前不存在于django的User类中，可以放在UserProfile中，但这样的设计并不好，因为UserProfile存储的是头像，昵称，以及类似朋友圈中添加的动态这样的信息，而手机号码是比较私密的信息，一般应该放在User类中。要实现这个功能：

1. 在accounts/models.py中定义一个User类，继承自AbstractUser

class User(AbstractUser):

phone\_number = model.CharField(max\_length=11,null=True, blank=True)

其中blank=True，默认可以是空字符串，主要是在admin/界面添加数据时，提交时如果不填phone\_number这一项就是空字符串，假如不设置blank=True，会提示你这一项要填。null=True更多是在数据库层面体现是否需要填写。

1. 在settings.py中添加AUTH\_USER\_MODEL = ‘accounts.models.User’

方法，属性的定义位置有两种选择，一个是serializer中，一个是model中，如果只用于渲染，就放在serializer中，否则就放在model中。

Model中定义一个方法，属性不需要migrate，只有field字段产生了更改才需要migrate。

Model中的内嵌的class Meta中定义的字段ordering可以指定排序规则，在使用ORM的方法时如果没有自己通过order\_by指定排序规则，就会按照ordering的规则来排序，最好是二者都有，即使存在ordering，也最好写一个order\_by，这样不需要再去找，这也是函数名尽可能的取的很明了的原因。

如果什么规则都没有指定，mysql会默认按id排序，PostgreSQL则不是，它是一段一段的是按照id排序，按页有序的。

在Web开发中，基本不用多线程，而是多进程。一个request过去，理想状况是200ms内结束，在处理request中需要非常非常快结束。request通常是分析它要什么数据，去数据库取数据，组织数据并返回response，整个流程中基本不需要批量处理数据，类似fanout操作需要大批量处理(例如有10万粉丝，要一次写10万次)，也不需要多线程，因为一个进程中的多线程不稳定，一个线程出问题，可能导致其他线程都挂掉，这种情况可以使用MessageQueue，把一个大批量的任务拆解成多个子任务，然后被多台机器分别领走，处理完后写入数据库。

web开发中request请求会分配(不会每个request都新开一个进程)到不同的进程中处理，而不是线程中。

一个request都在fanout，另一个request把tweet删掉了或者unfollow取关了，这种情况需要代码写的更为兼容：需要在fanout过程中进行检测，例如tweet\_id是否存在，进而报错。另一种可能是已经fanout成功，但之后unfollow了，这样就需要在渲染自己的新鲜事列表时，也要进行相应的filter，删掉自己已经取关的新鲜事。

django中的mixins类似java中的interface，定义能干什么，但只定义方法，不定义字段，而且提供了默认的实现方法。

localhost/默认的首页显示哪些链接，只有views提供了list方法才会显示，否则不会显示。