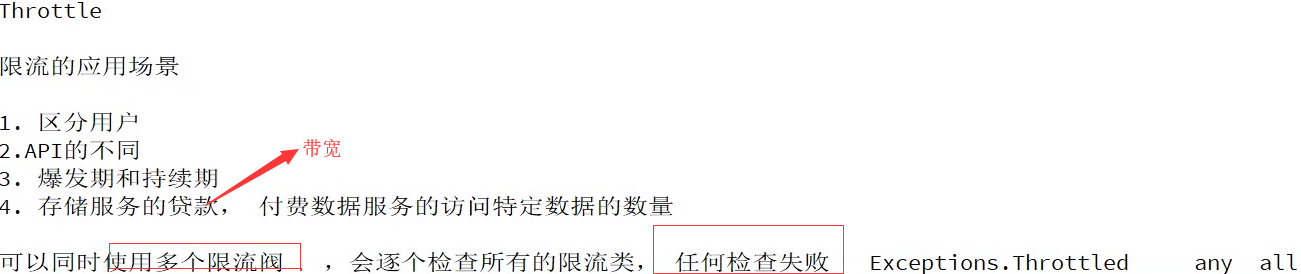
1. **限流Throttling**

限流类似于权限机制，因为它也决定是否接受当前请求。限流可以形象地比喻为节流阀，指示 一种临时状态，用于控制客户端在某段时间内允许向API发出请求的次数，也就是频率。



**配置限流机制**



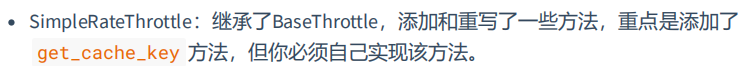
既然要限流，那么必须识别客户端！那么DRF是如何判断和区分当前客户端的身份的呢？

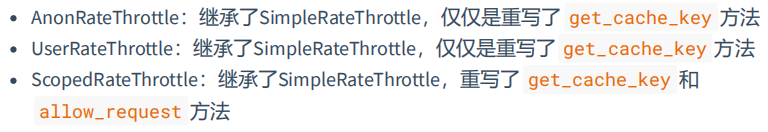
DRF利用**HTTP报头的 'x-forwarded-for'** 或**WSGI中的 'remote-addr'** 变量来唯一标识客户端的IP地址。

**如果存在 'x-forwarded-for' 头部属性，则使用它，否则将使用WSGI中 'remote- addr' 变量的值。**

get\_cache\_key()

**获取标识符的方法，即判断用户是否登录等等**



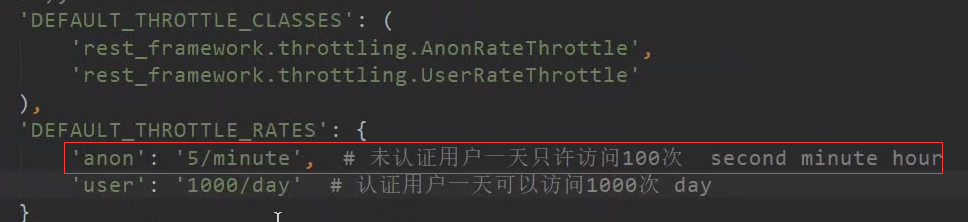


**AnonRateThrottle**

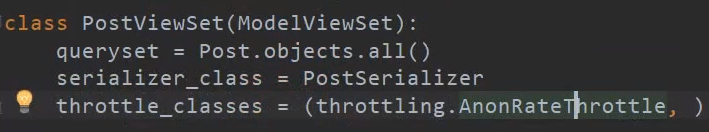
AnonRateThrottle 只会限制未经身份验证的用户。传入的请求的IP地址用于生成一个唯一的密钥。

AnonRateThrottle 测试

1. Settings



1. Views



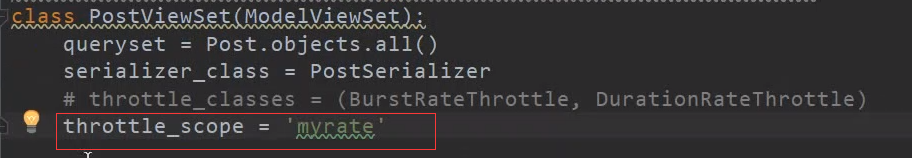
结果：未登录用户每分钟最多访问5次

**ScopedRateThrottle**

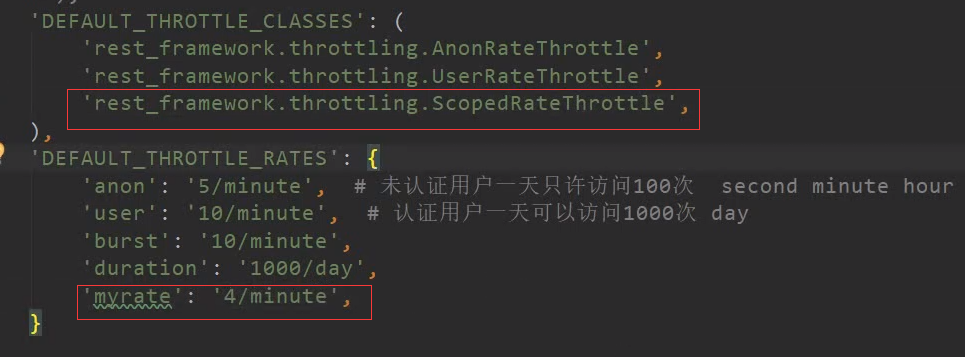
ScopedRateThrottle 类用于限制对APIs**特定部分的访问，也就是视图级别的限流，不是全局性的**。只有当正在访问的视图包含 **throttle\_scope 属性时，才会应用此限制**。然后，通过将 视图的“scope”属性值与唯一的用户ID或IP地址连接，生成唯一的密钥。

列：API试图级别限制

1. 编写视图

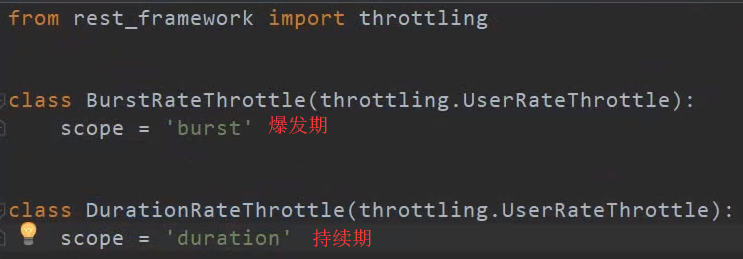
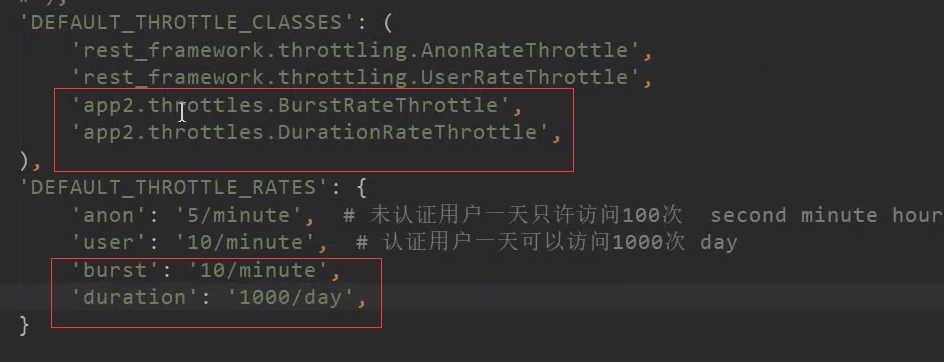


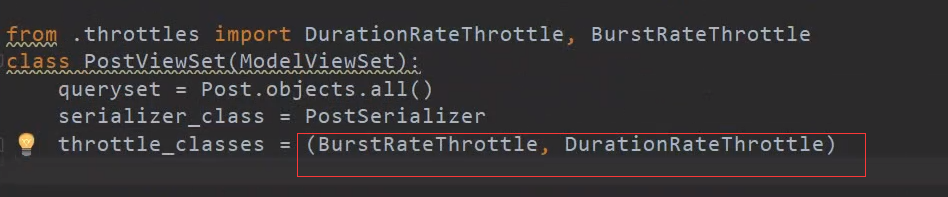
1. 配置settings



**自定义限流类：**

用户在高峰期访问限制为10次/minute，用户在持续期（一般）限制为1000次/day

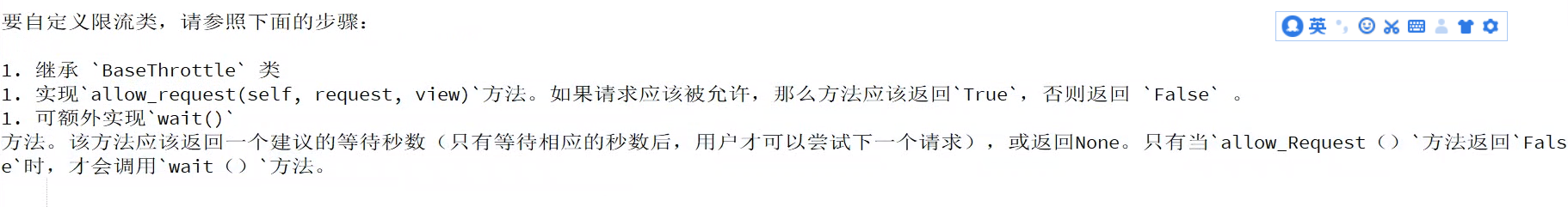
1. 编写throttles类
2. 配置settings
3. 编写视图



DRF限流需要使用DJango的缓存后端。默认的是基于内存的缓存

**在ScopedRateThrottle源码中。**

当执行完初始化方法后，是在APIView里面调用allow\_request方法，然后返回到ScopedRateThrottle中执行allow\_request方法



自定义一个随机限流类

