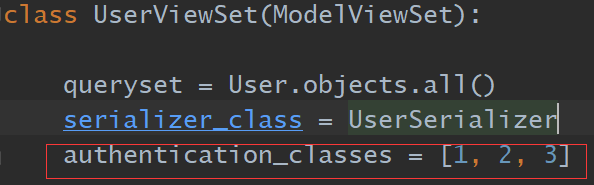
**一、认证**

**在权限和限流检查发生之前，以及在执行任何其他代码之前，始终在视图的最开始处运行身份验证。也就是说，认证过程优先级最高，最先被执行。**

**认证的机制**

允许使用的认证模式一般以一个类的列表的配置形式存在。

DRF会尝试使用列表中的每个类进行认证，并使用第一个成功通过的验证类的返回值设置 request.user 和 request.auth。如果所有认证类尝试了一遍，但还是没有通过验证。 request.user 将被设置为django.contrib.auth.models.AnonymousUser 的实例，也就是匿名用户。request.auth 也将被设置为 None 。可见，DRF**的认证机制依赖**Django**的**auth**框架！**

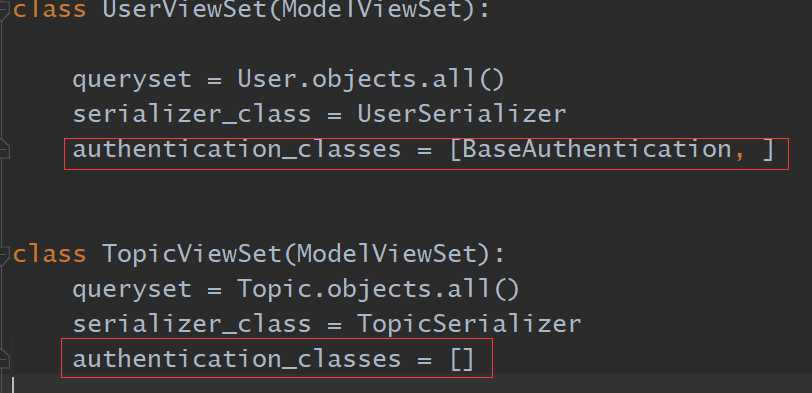
其中可在settings上为所有视图做**全局配置**如：



也可以在是图中单独配置，且权限高于全局配置。即

第一个视图表示用一个**BaseAuthentication**（覆盖了全局配置）

第二个视图表示**不用认证**,任何人都可以访问（覆盖了全局配置）



认证功能的核心源代码在这里：



当未经身份验证的请求被拒绝时，可能有两个不同的错误代码。

HTTP 401 **Unauthorized**

HTTP 403 **Permission Denied**

HTTP 401响应一般会包含一个 WWW-Authenticate 头部属性，用于指引用户如何认证。而 HTTP 403响应则不会包含这个 WWW-Authenticate 属性。

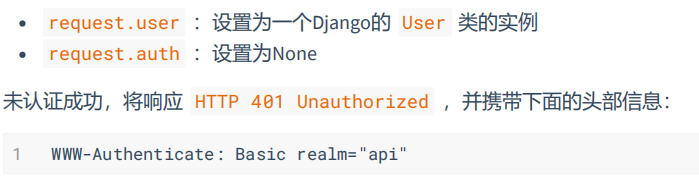
**BasicAuthentication：**

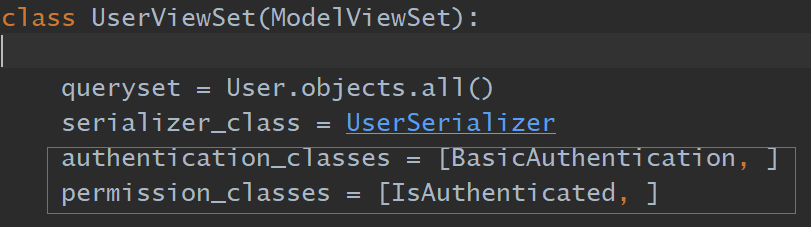
BasicAuthentication使用HTTP基本的认证机制，通过用户名/密码的方式验证。

用户名和密码必须在HTTP报文头部，为 Authorization 属性提供值为

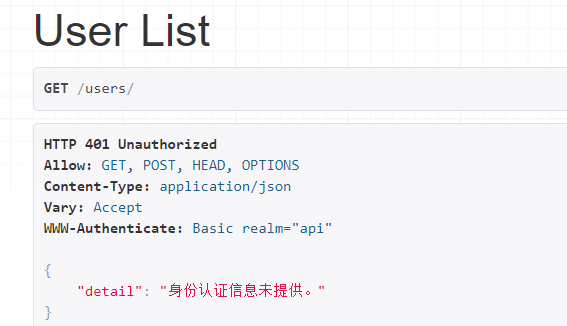
Basic amFjazpmZWl4dWVsb3ZlMTAw 的方式提供。其中 Basic 字符串是键，后面的一串乱码是通过base64 库使用明文的用户名和密码计算出的密文，这一部分工作可以通过postman工具进行。

如果认证成功， BasicAuthentication 提供下面的属性：



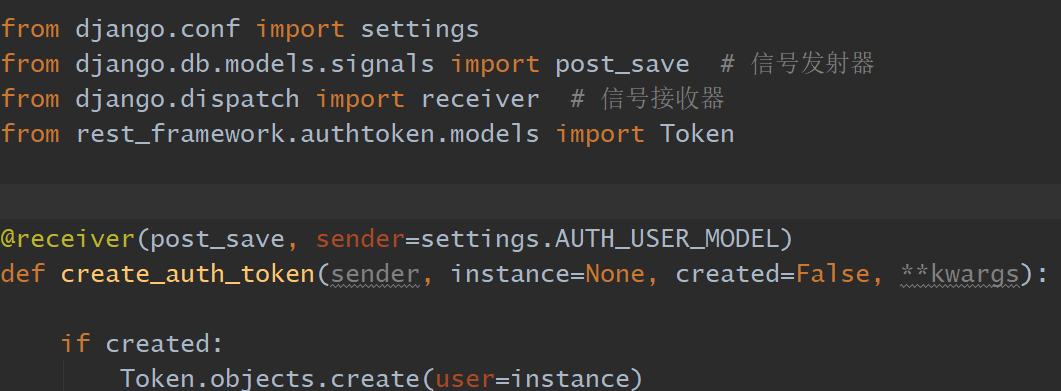
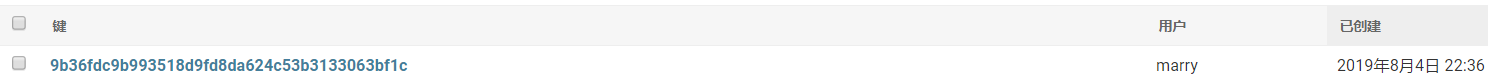
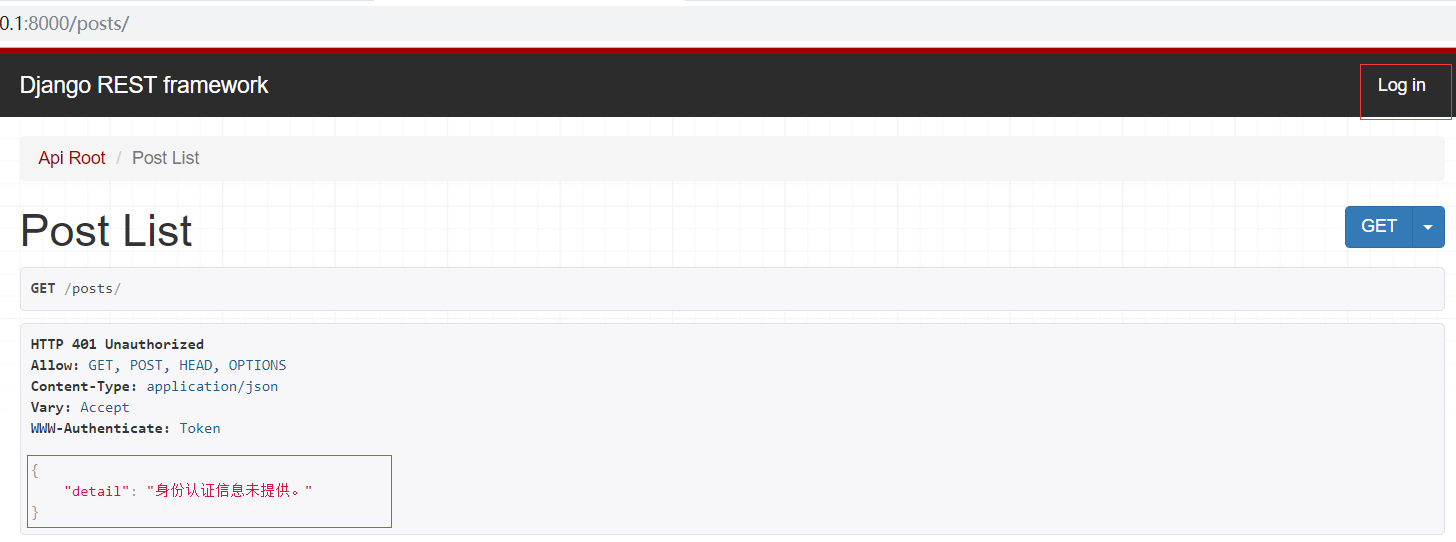
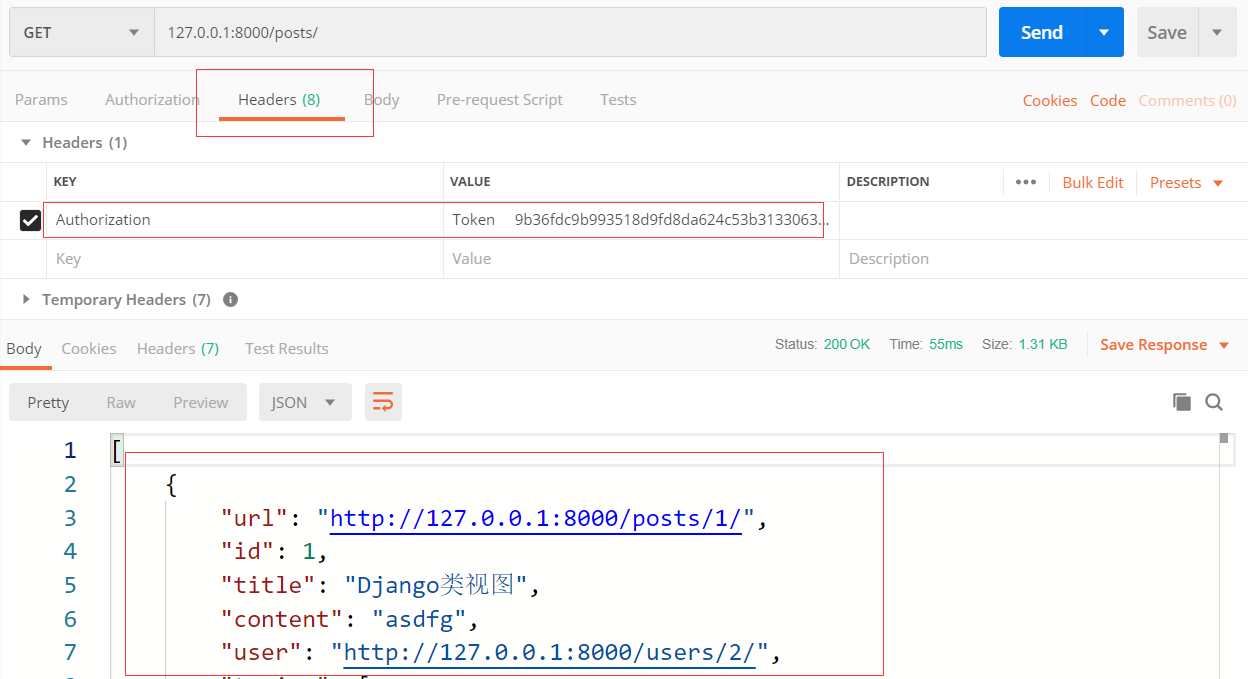


**搭配权限管理**：即未认证的不允许访问UserViewSet视图内容即



**TokenAuthentication**

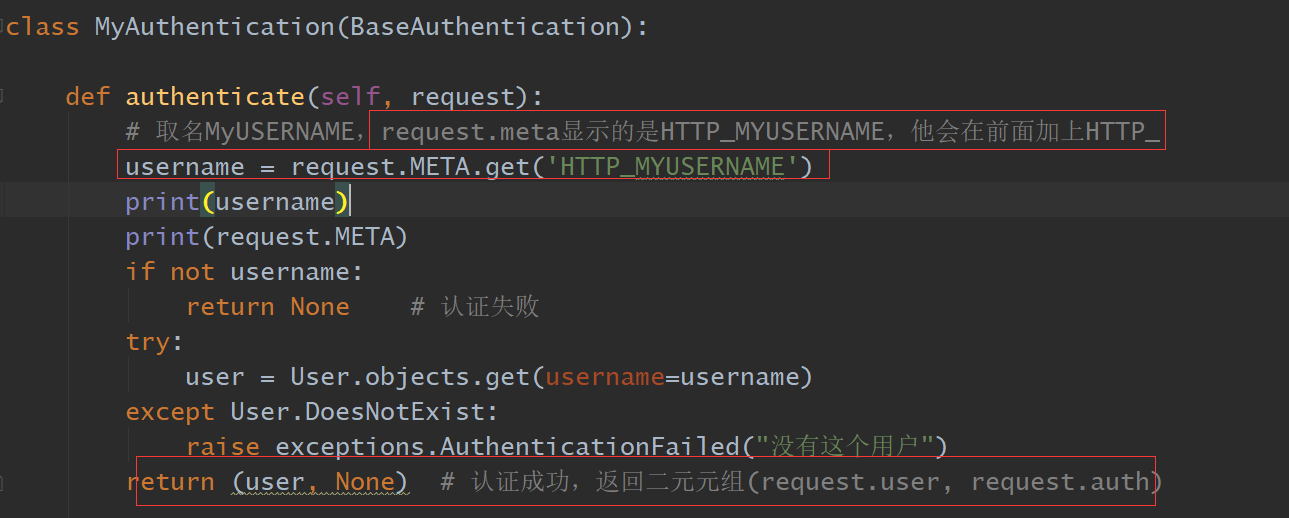
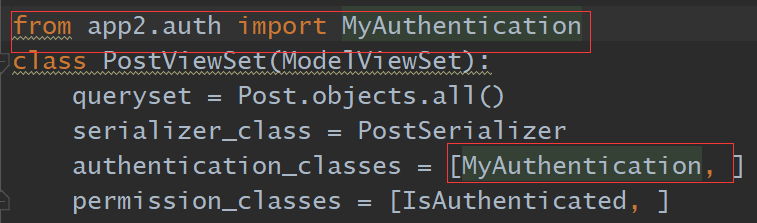
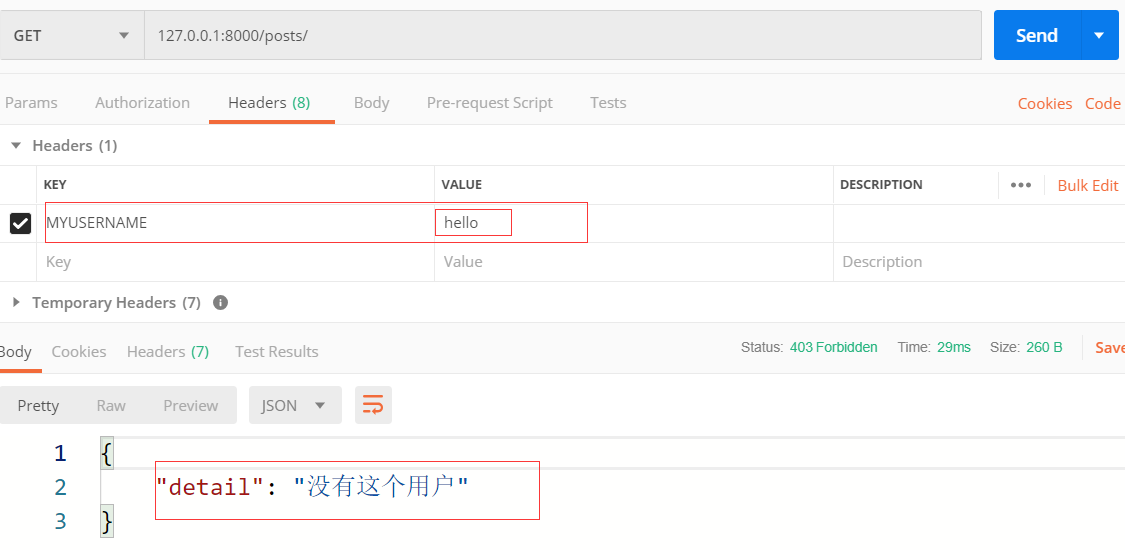
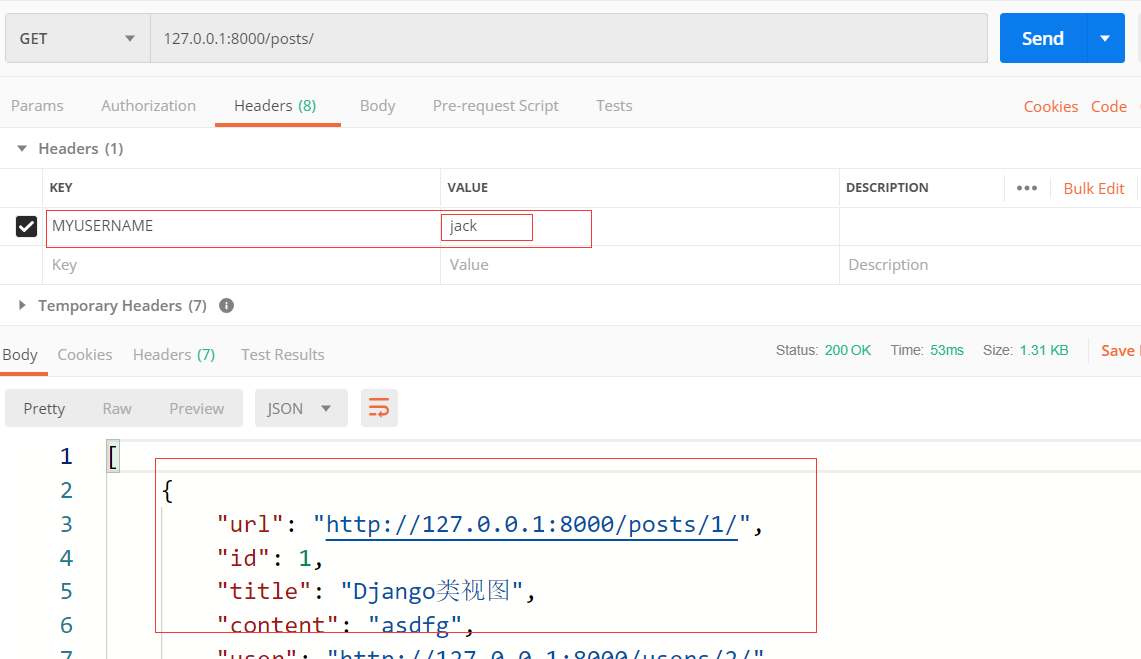
TokenAuthentication是一种简单的基于令牌的HTTP认证。它适用于CS架构，例如普通的桌面应用程序或移动客户端。使用步骤如下：

1. 需要在INSTALLED\_APPS 里面注册‘rest\_framework.authtoken’因为这是一个第三方模块
2. 
3. 需要运行python manage.py migrate 生成对应的表
4. 利用**信号机制**为你的用户创建令牌，在views里面新建如下：
5. 
6. 在admin下新建用户marry，查看如下
7. 
8. AUTH\_USER\_MODEL默认为django.contrib.auth.models.User即Django默认的User模型，若想指定修改为自己的User模型，需要在settings**里面重新配置如**：AUTH\_USER\_MODEL = ‘app.models.User’
9. 使用默认的User访问结果如下
10. 
11. 跟登录无关，需要发送Token值，此时我们要postman发送，结果如下：
12. 

**SessionAuthentication**

显示结果和**BasicAuthentication**差不多，但是**SessionAuthentication**基于会话的，**BasicAuthentication** 是在http头部设置用户名和密码。

**自定义认证：**

1. 写一个认证类
2. 
3. 
4. 通过postman在头部提供认证
5. 测试不存在用户
6. 
7. 测试存在用户
8. 

**第三方模块**



......