# SpringSecurity+oauth2+jwt

本例说明：SpringSecurity+oauth2+jwt进行整合；

直接在前面写的SpringSecurity-oauth2里边进行修改，不在重新创建项目代码；

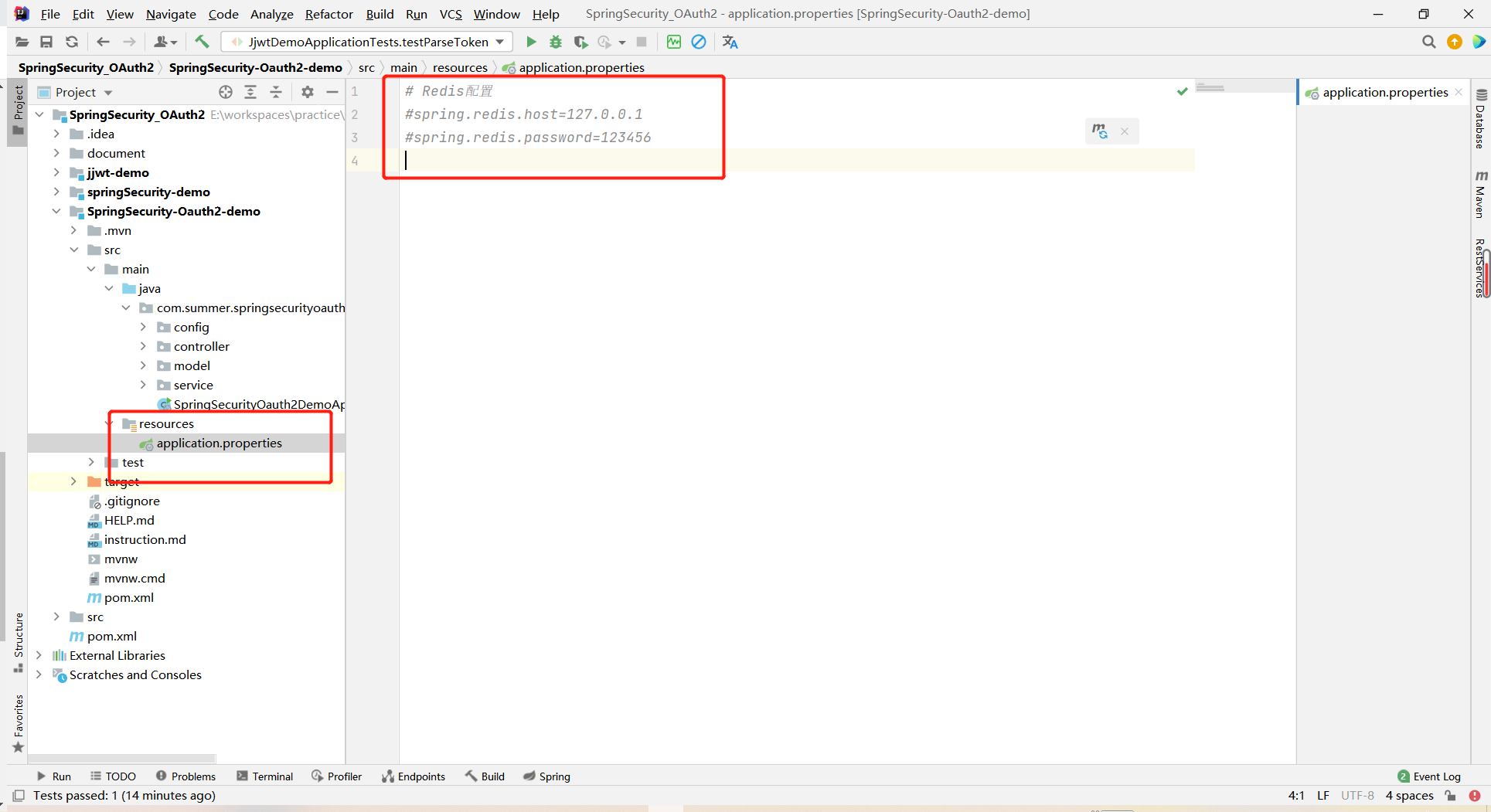
**实现原理：**

***我们需要吧oauth2自己生成的token，转成更加安全的引入的jwt的token；***

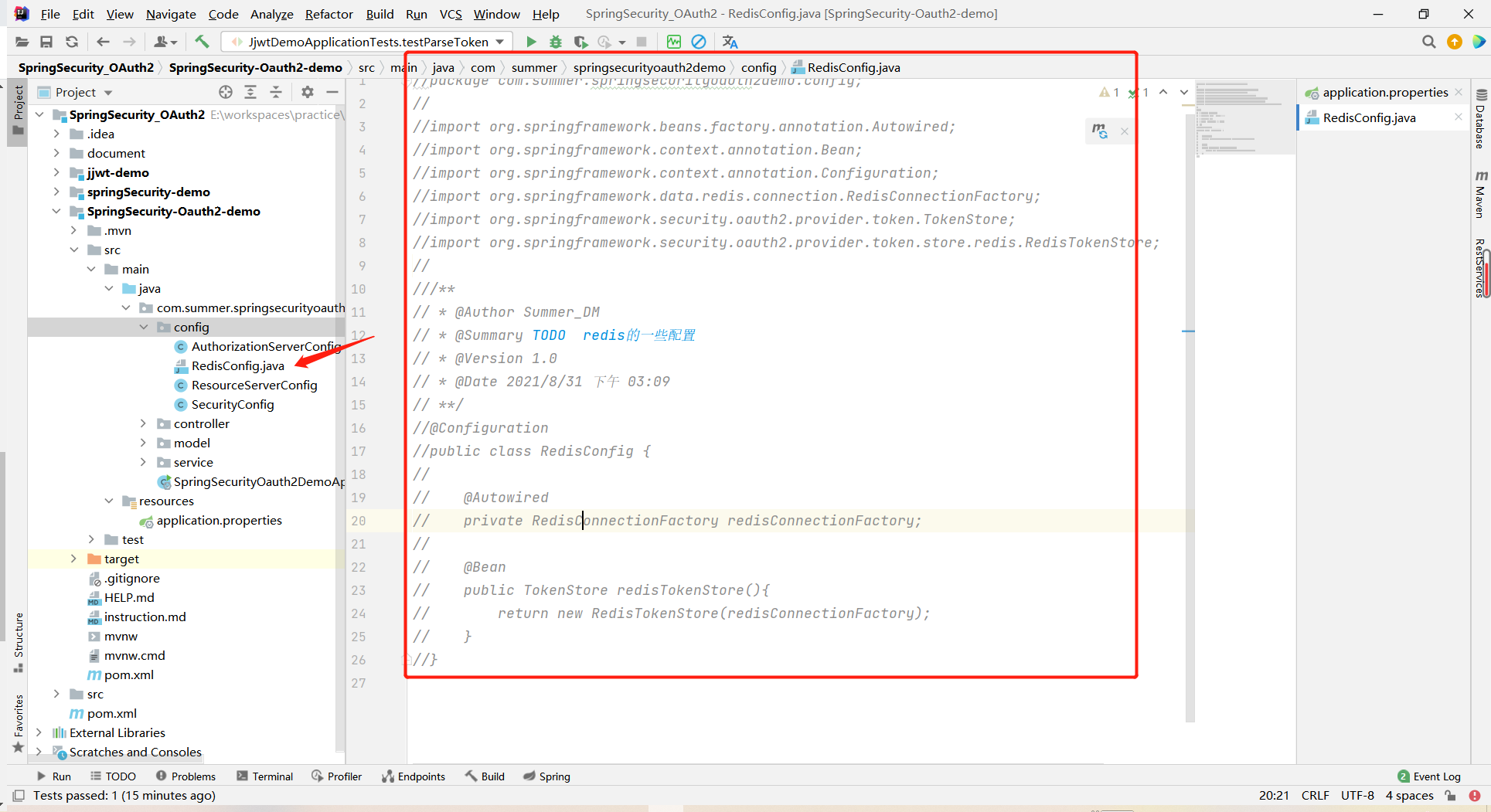
这里我们需要先将原来设置的使用redis存储token给干掉；

**一、干掉redis**

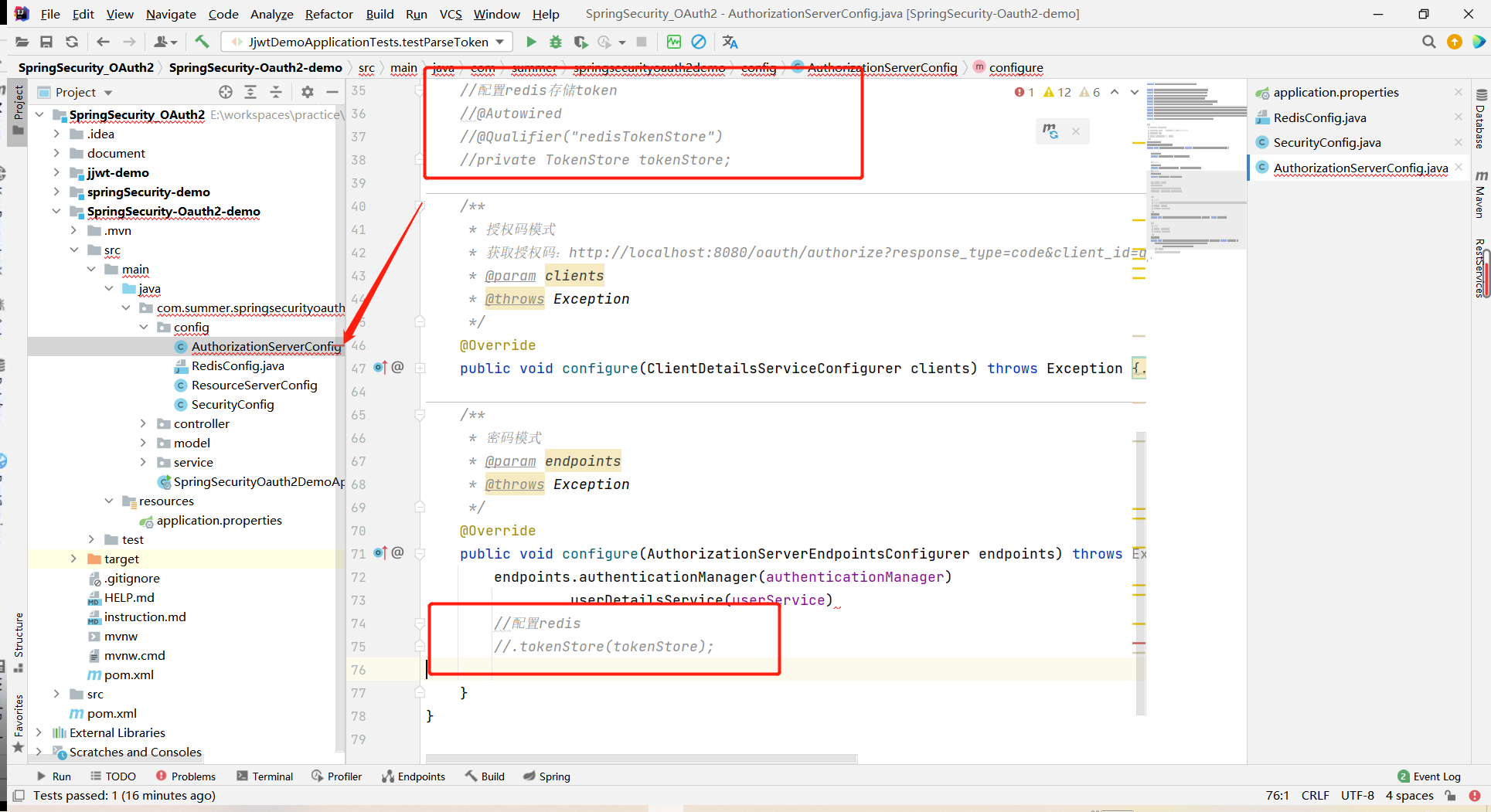
首先干掉配置文件：



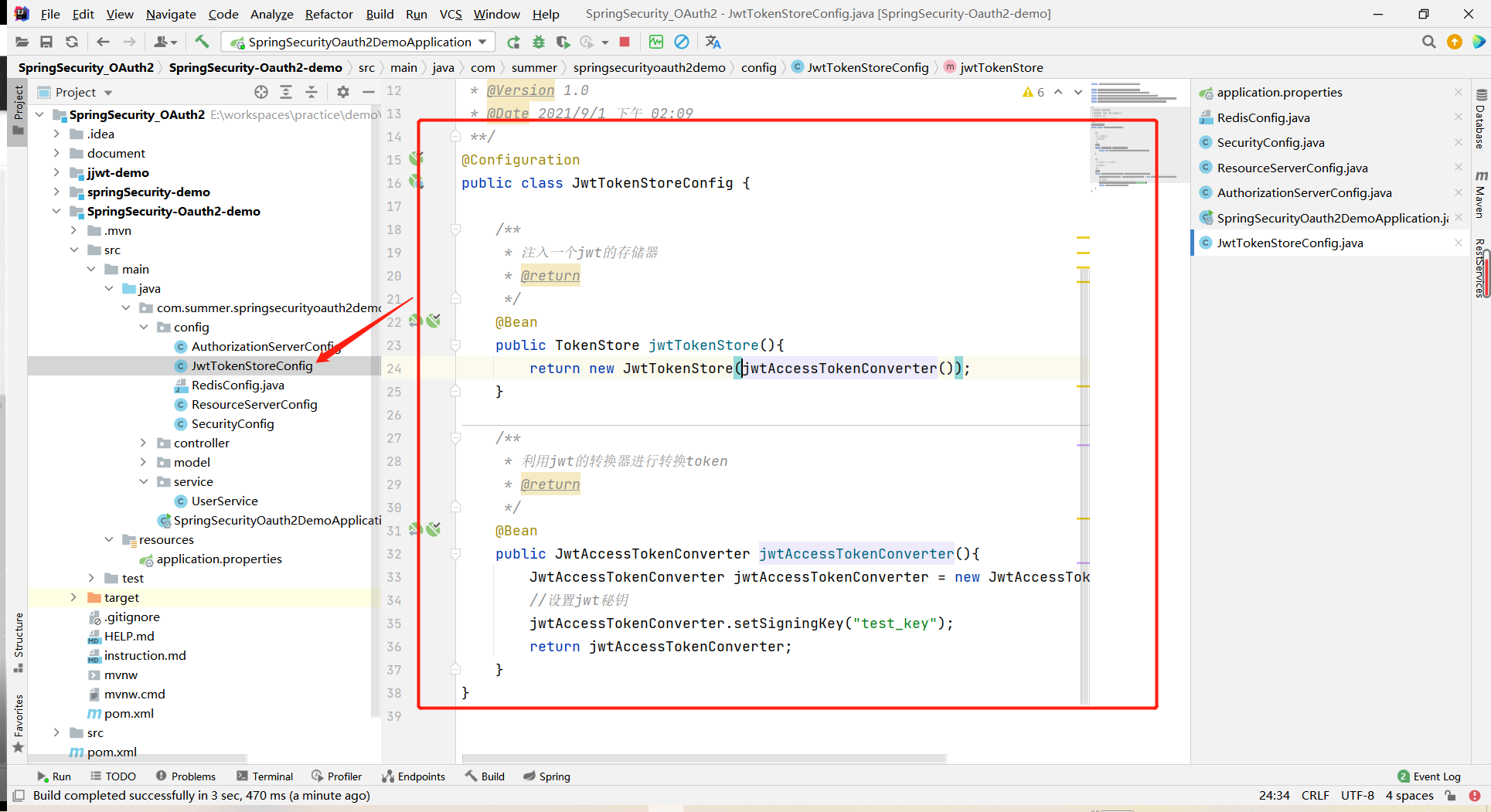
接着干掉RedisConfig配置类：



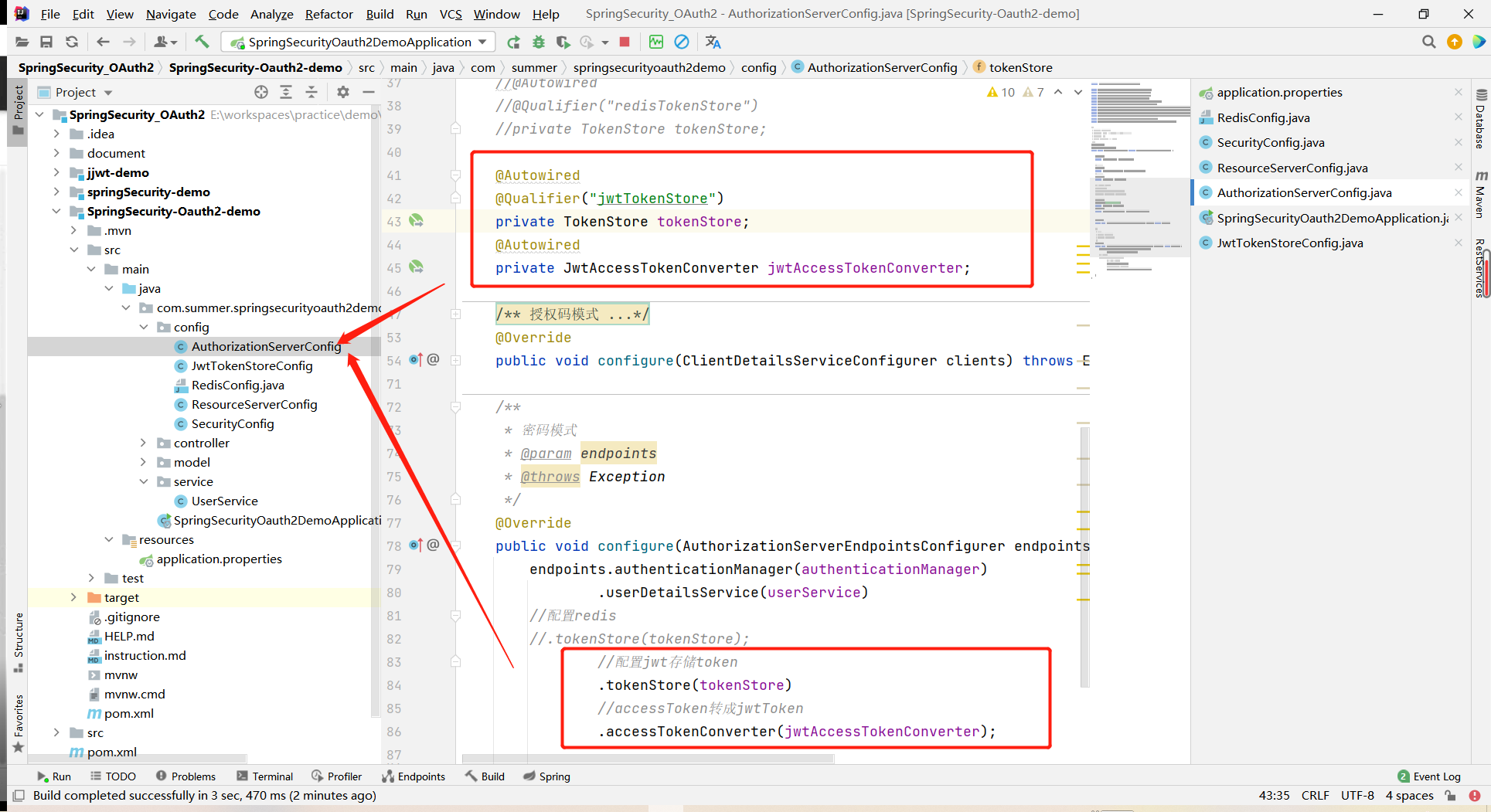
然后干掉授权服务器的关于redis的配置：



1. **开始创建jwt的token**
2. 创建一个jwt的jwtTokenStore配置类；

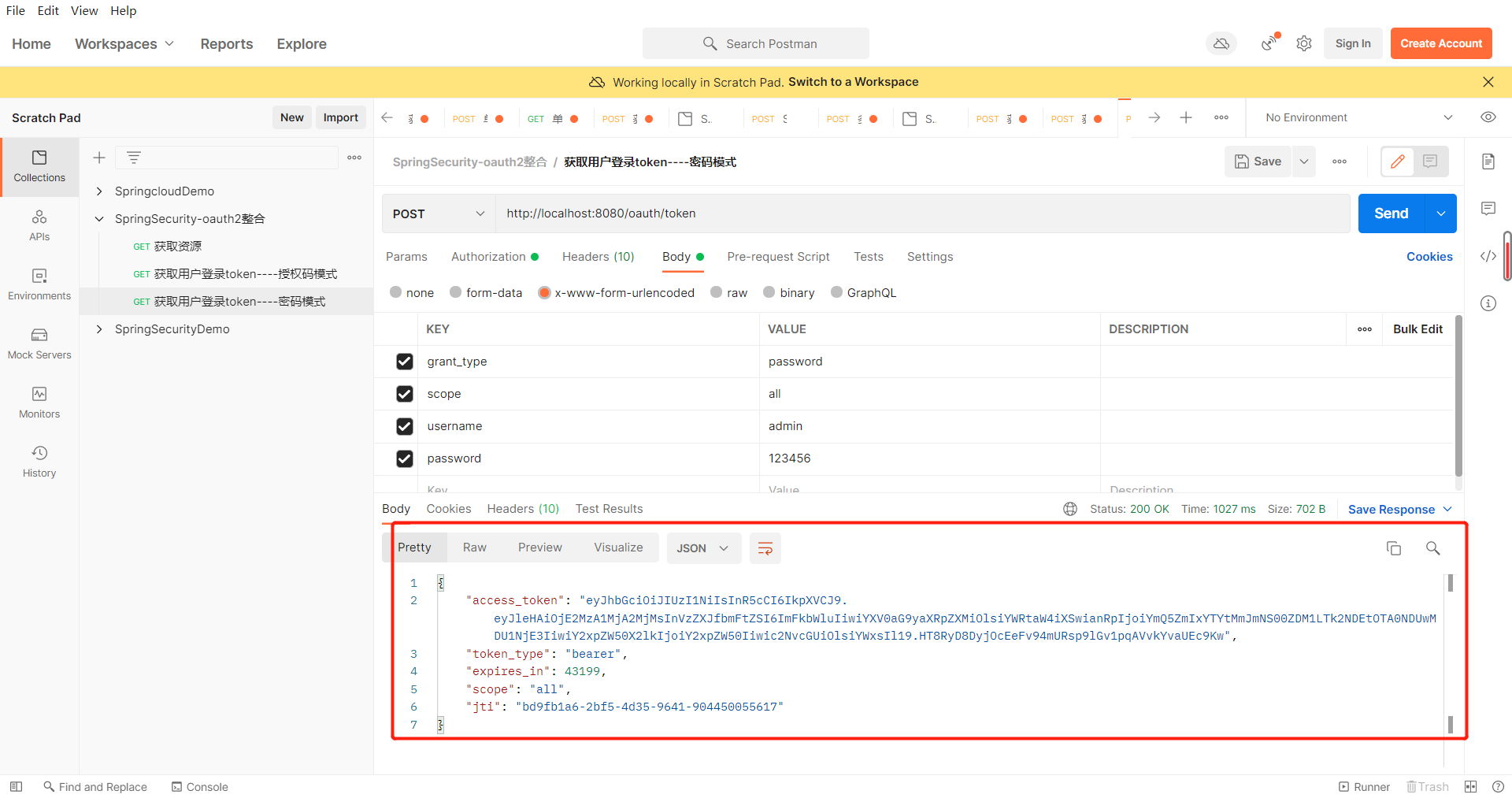


1. 在授权服务器引入jwt；

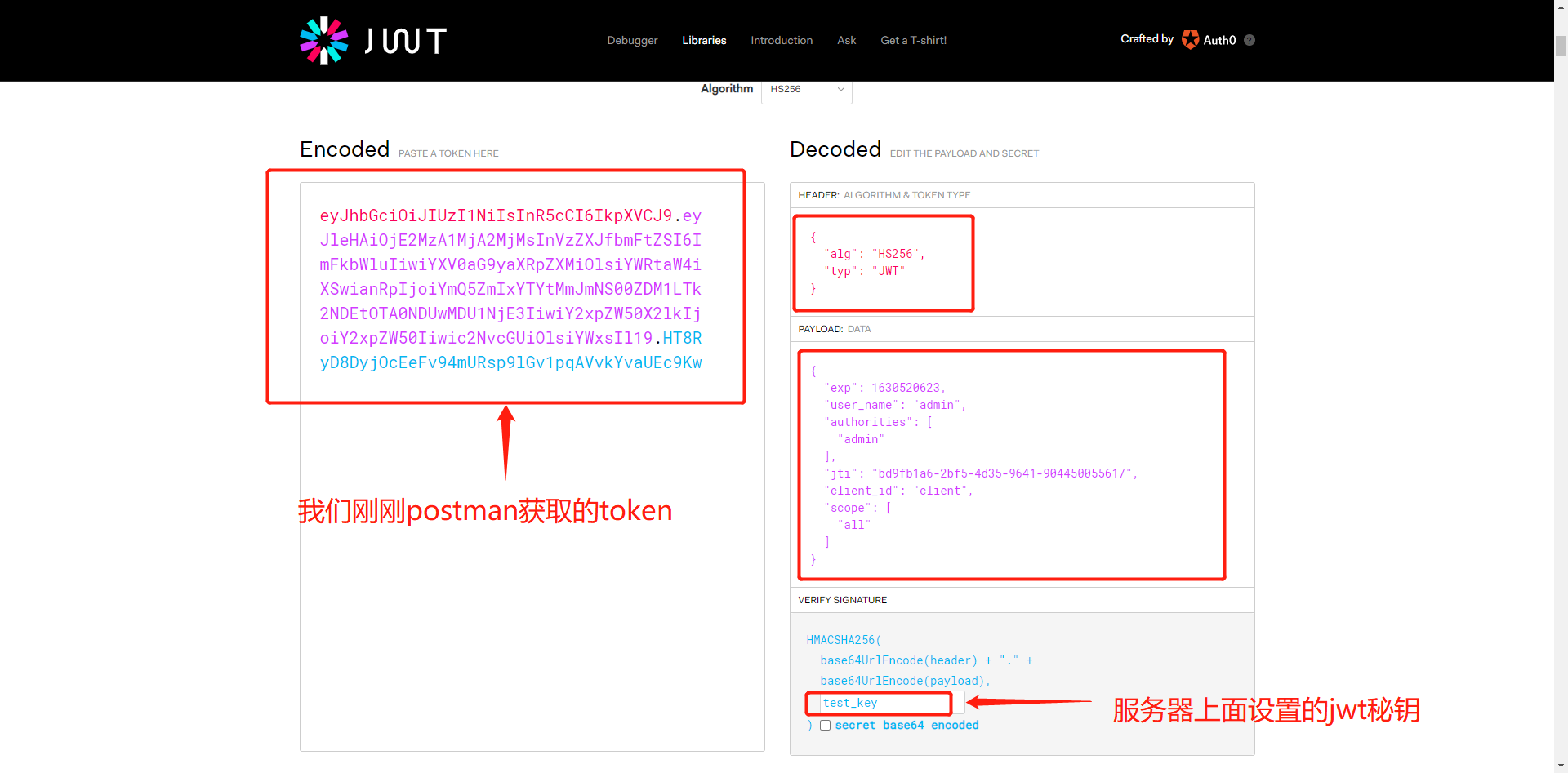


1. 然后重启服务，获取token；

就可以看到，现在的jwt生成的token，比较长。



1. 接下来，我们可以复制一下token，然后去官网或者咱们前面写的那个解析token的方法，去看一下jwt的token的组成；



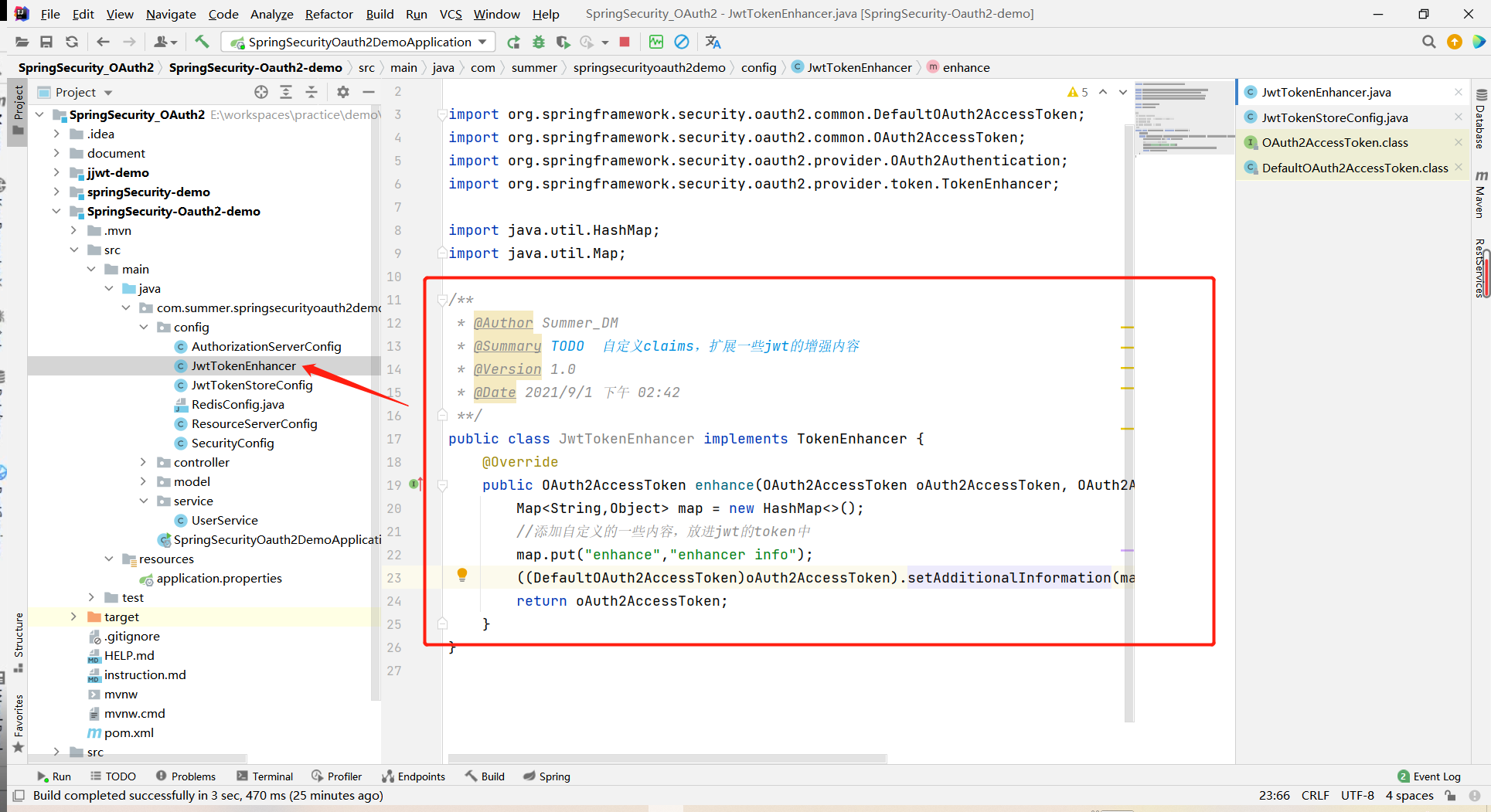
通过这里解析，可以看出来我们获取token的入参和自定义的一些东西；

***通过到这里，可以发现，jwtTokenStore和redisTokenStore特别类似，唯一的区别就是jwt多了一个转换器。***

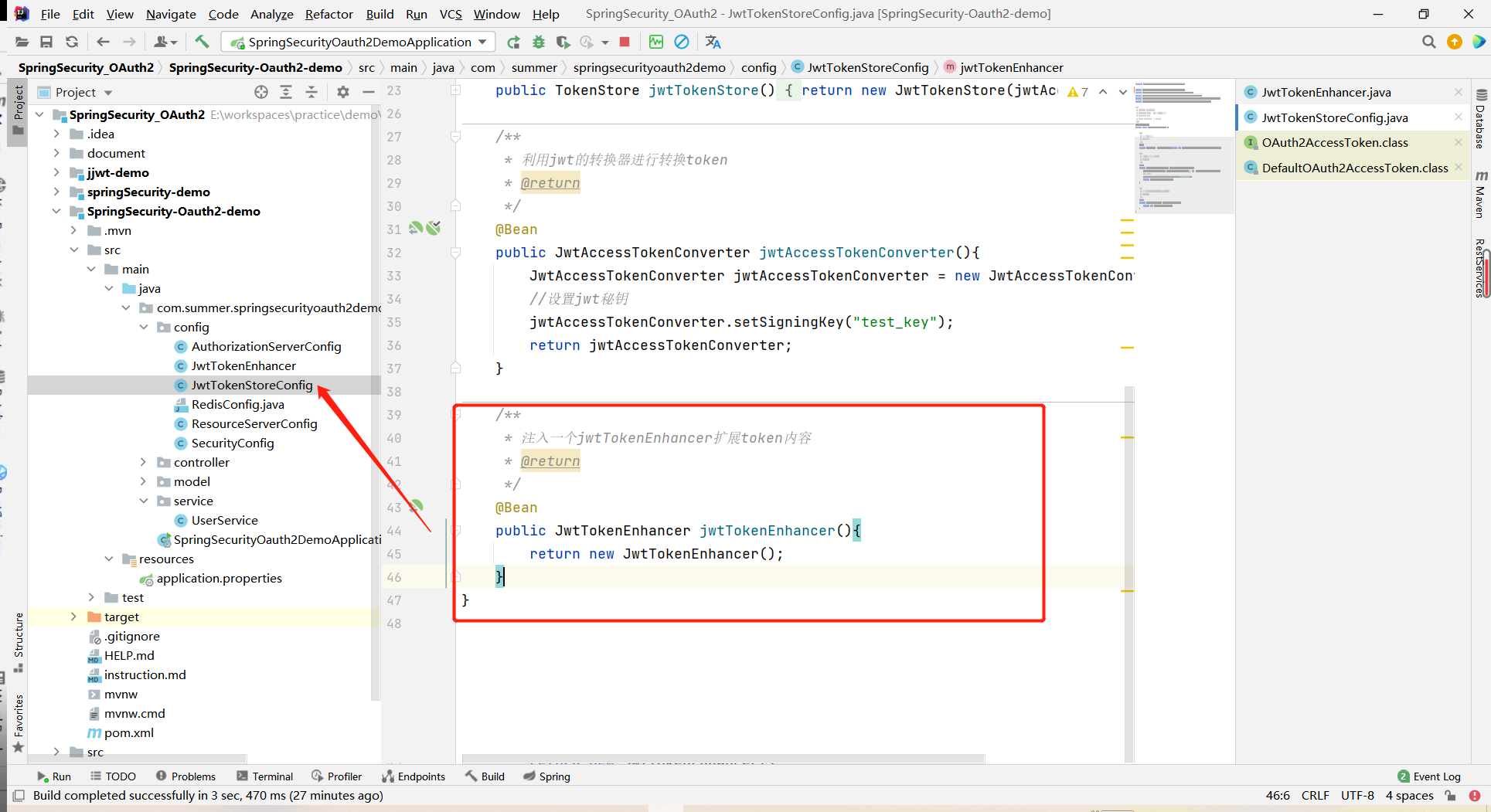
1. **接下来，我们去扩展一下jwt的存储内容**

拓展，其实就是自定义我们的claims，只不过因为要结合SpringSecurity和oauth2，所以和之前的不一样，难度会增加。

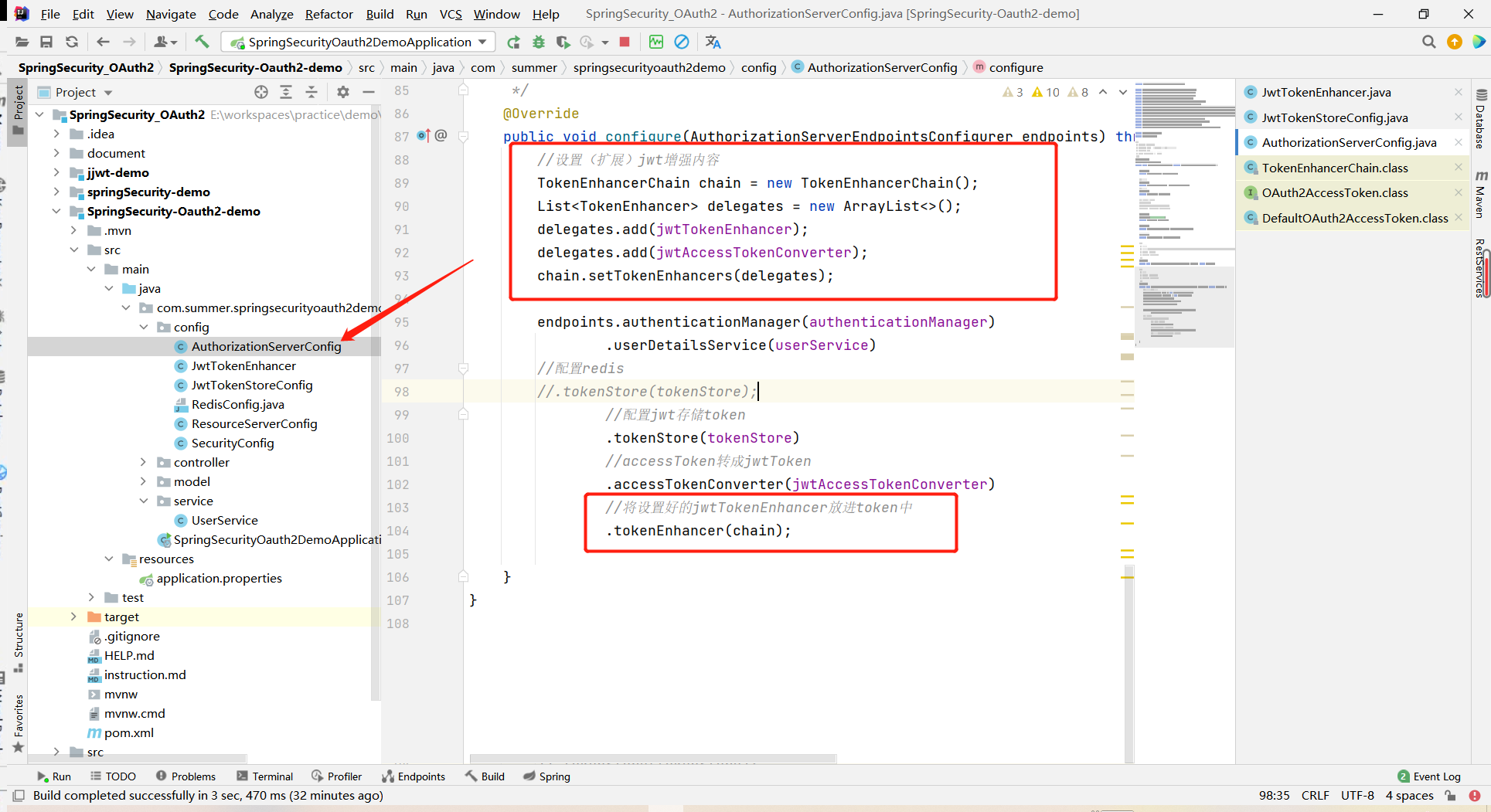
1. 首先我们需要定义一个JwtTokenEnhancer去实现TokenEnhancer；



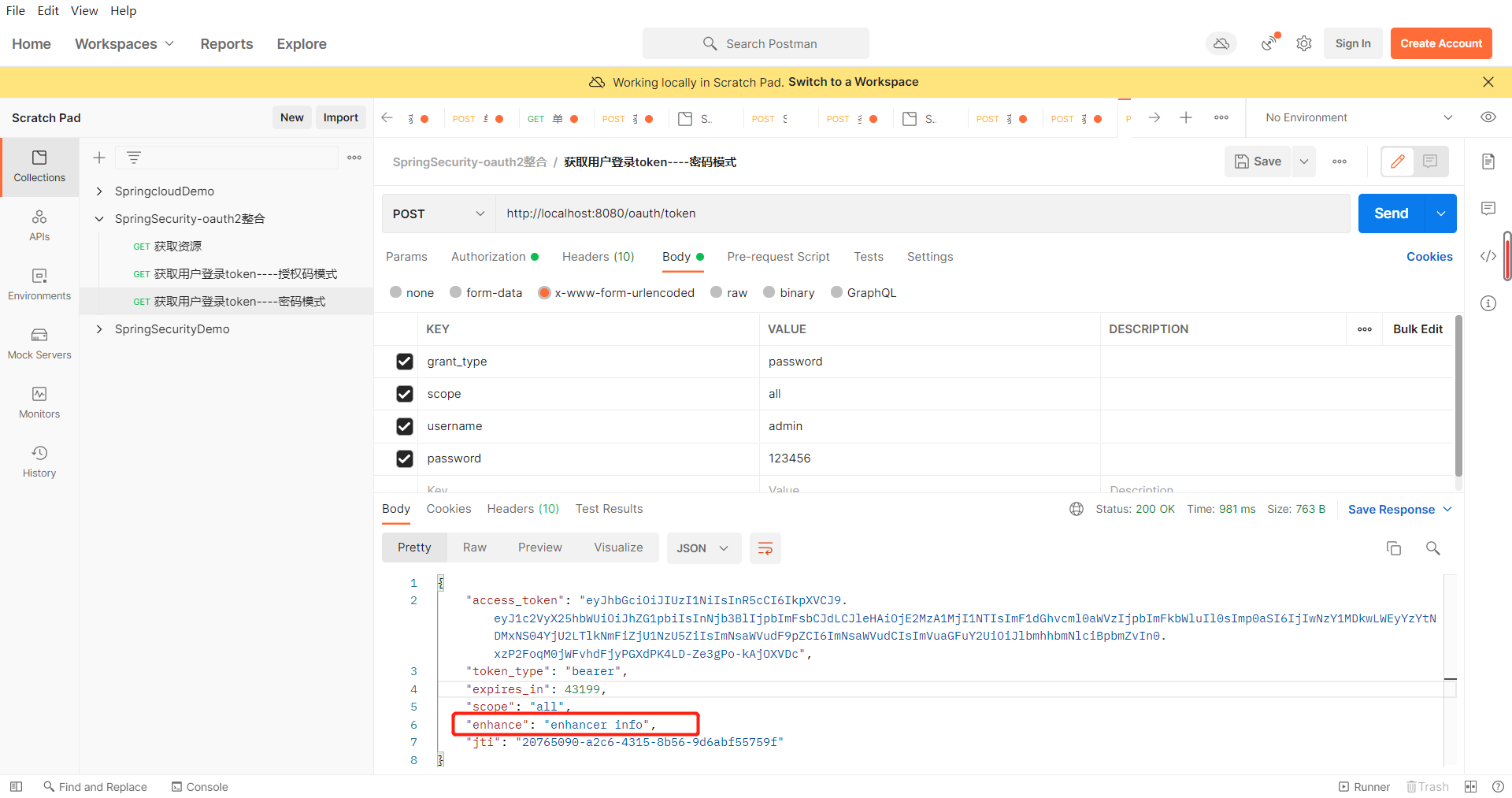
1. 然后需要去JwtTokenStoreConfig配置类中去注入一个JwtTokenEnhancer，交给spring去管理，一会使用；

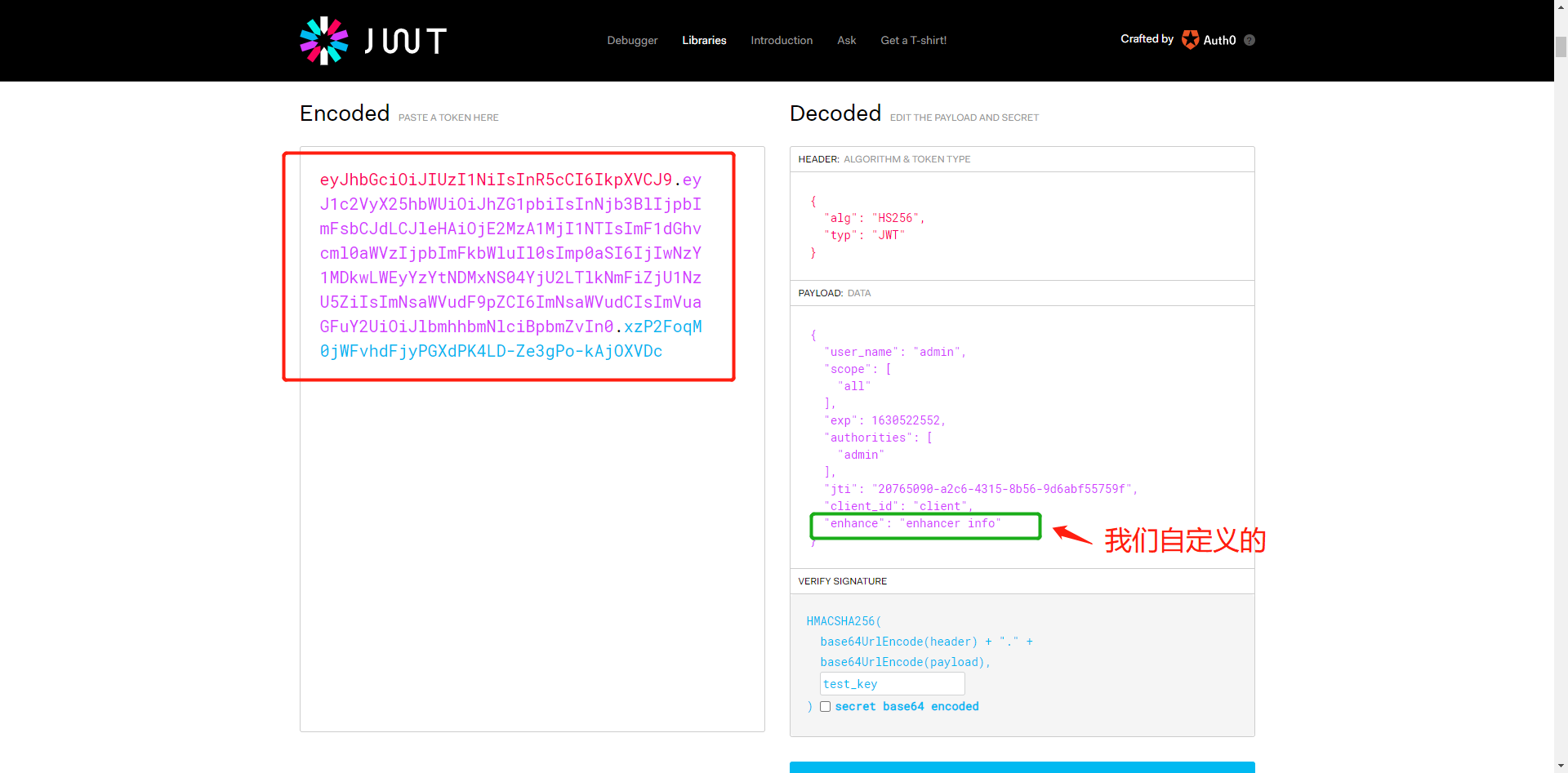


1. 最后我们需要去授权服务器的配置类进行配置；



1. 启动服务，测试，然后复制token，去解析确认结果；



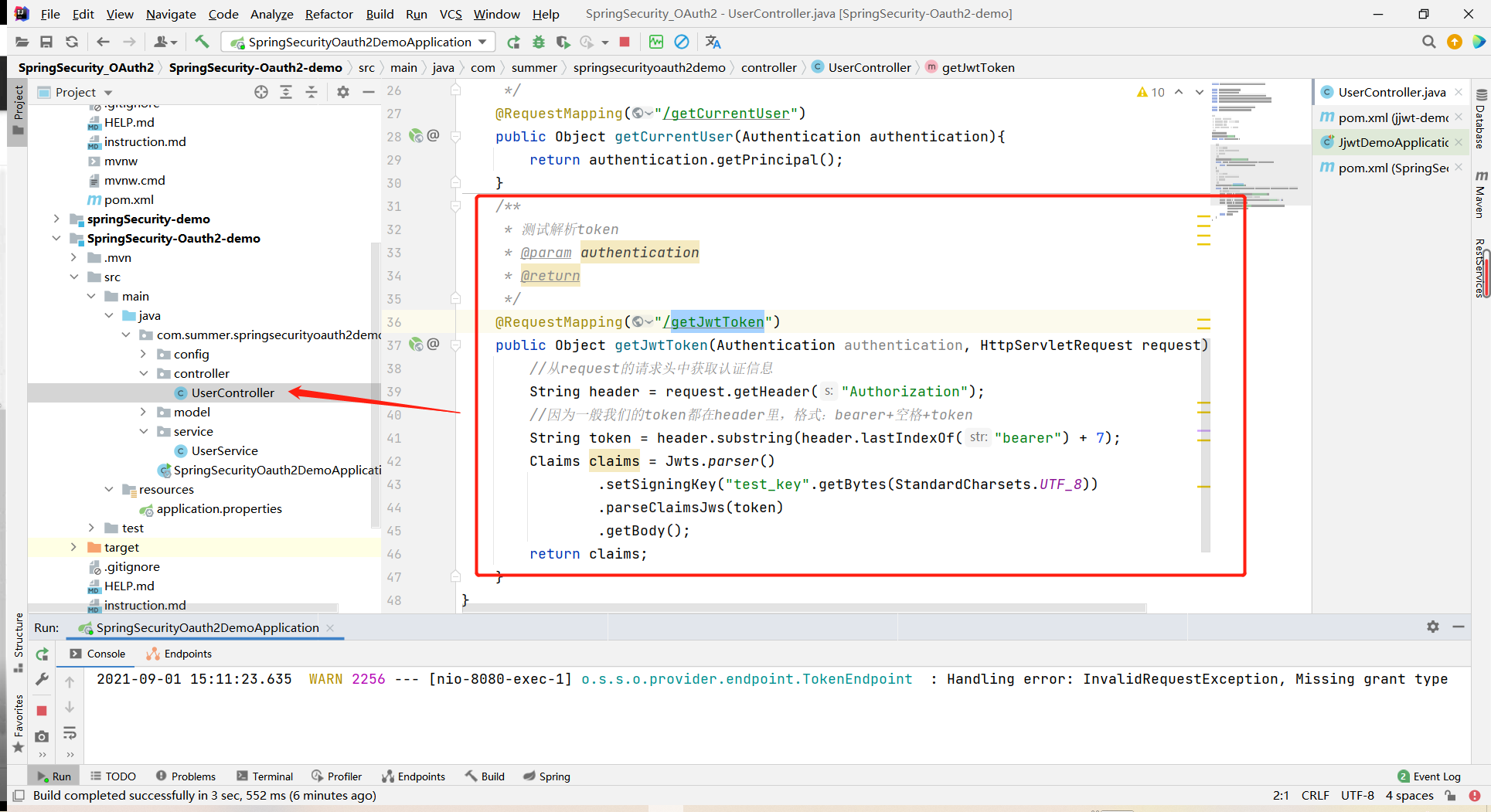


这样，我们就是实现了，可以自定义token里边添加内容；

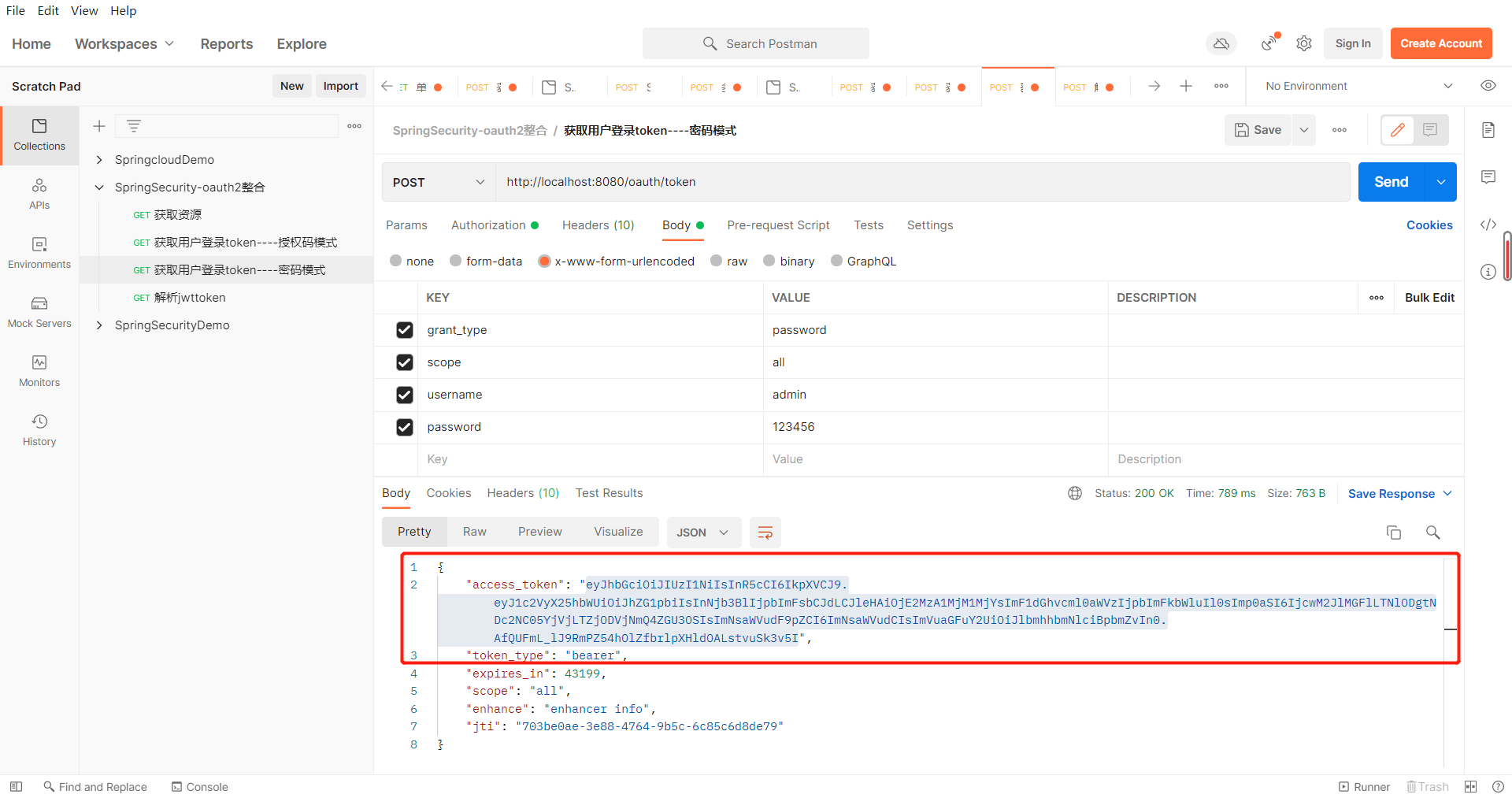
1. **紧接着我们来自己写代码，重新解析一下jwt的token**

这里主要实现一下，实际开发中，我们服务器的解析过程。

1. 我们需要去在controller中写一个获取解析token的接口；



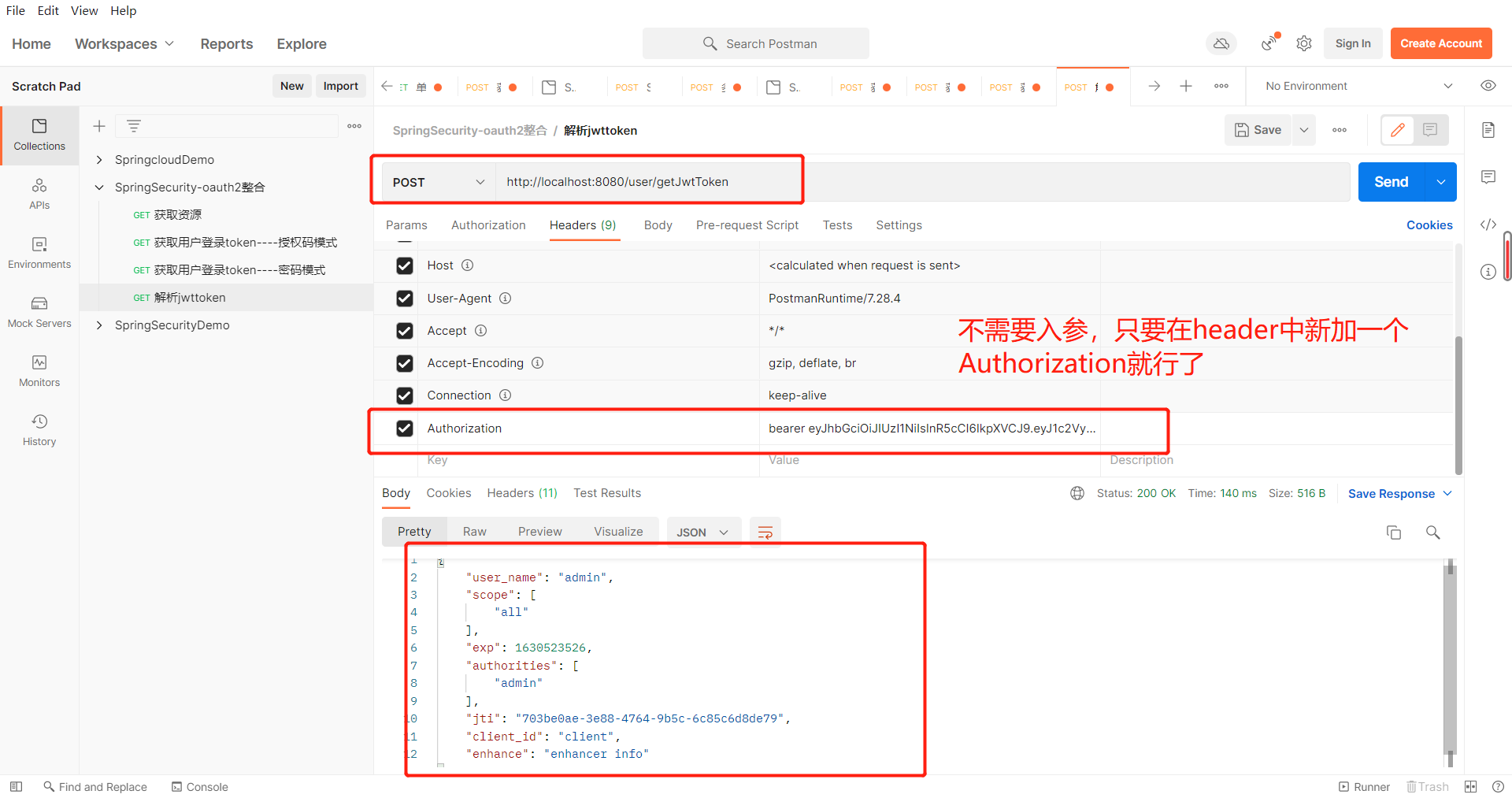
1. Postman先去请求获取一个token；



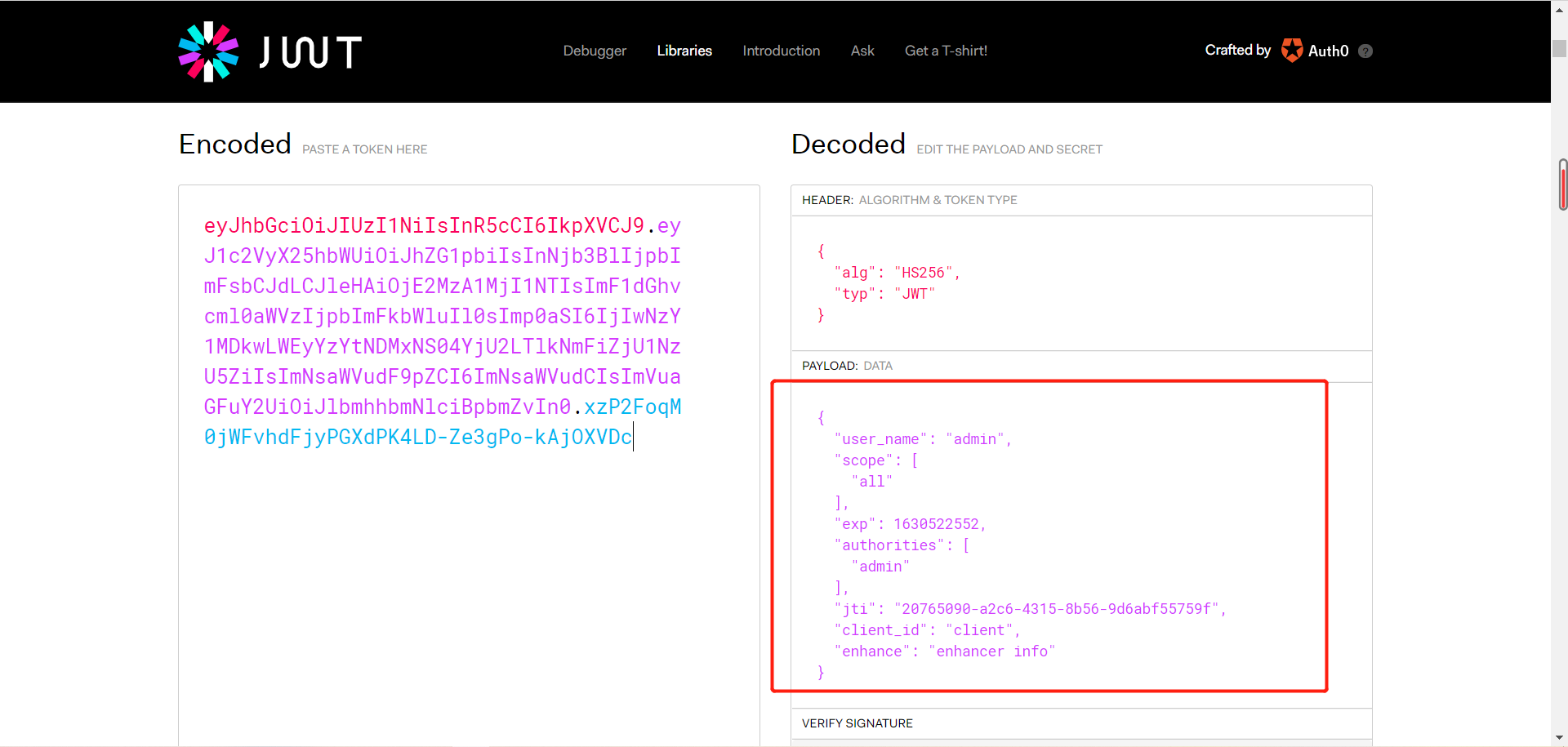
1. 再用postman请求刚才写的接口：

这里不需要入参，但需要在header中添加一个参数：

Authorization:bearer+空格+token

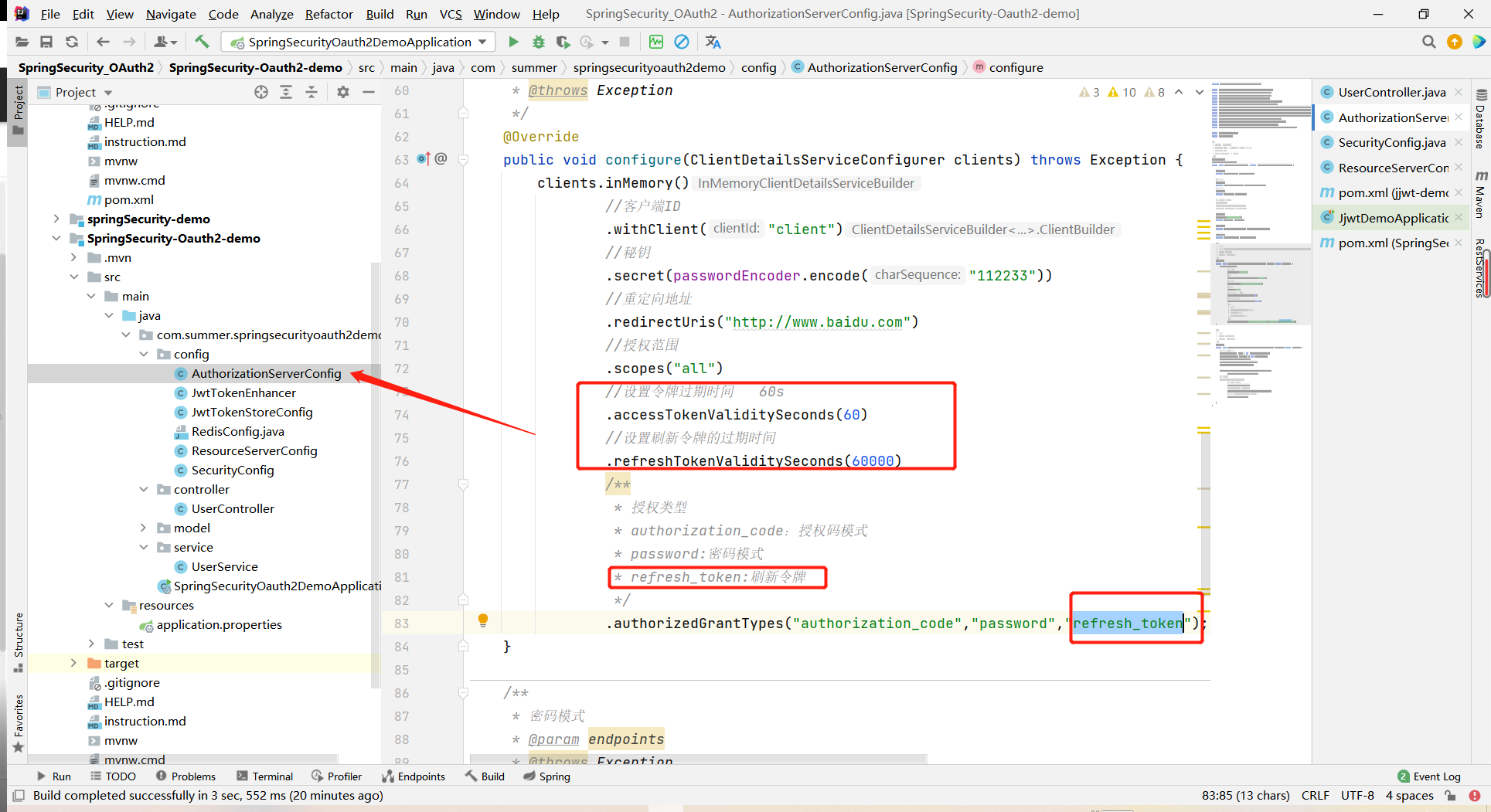


可以看到，和jwt官网的解析对比，结果是一样的。

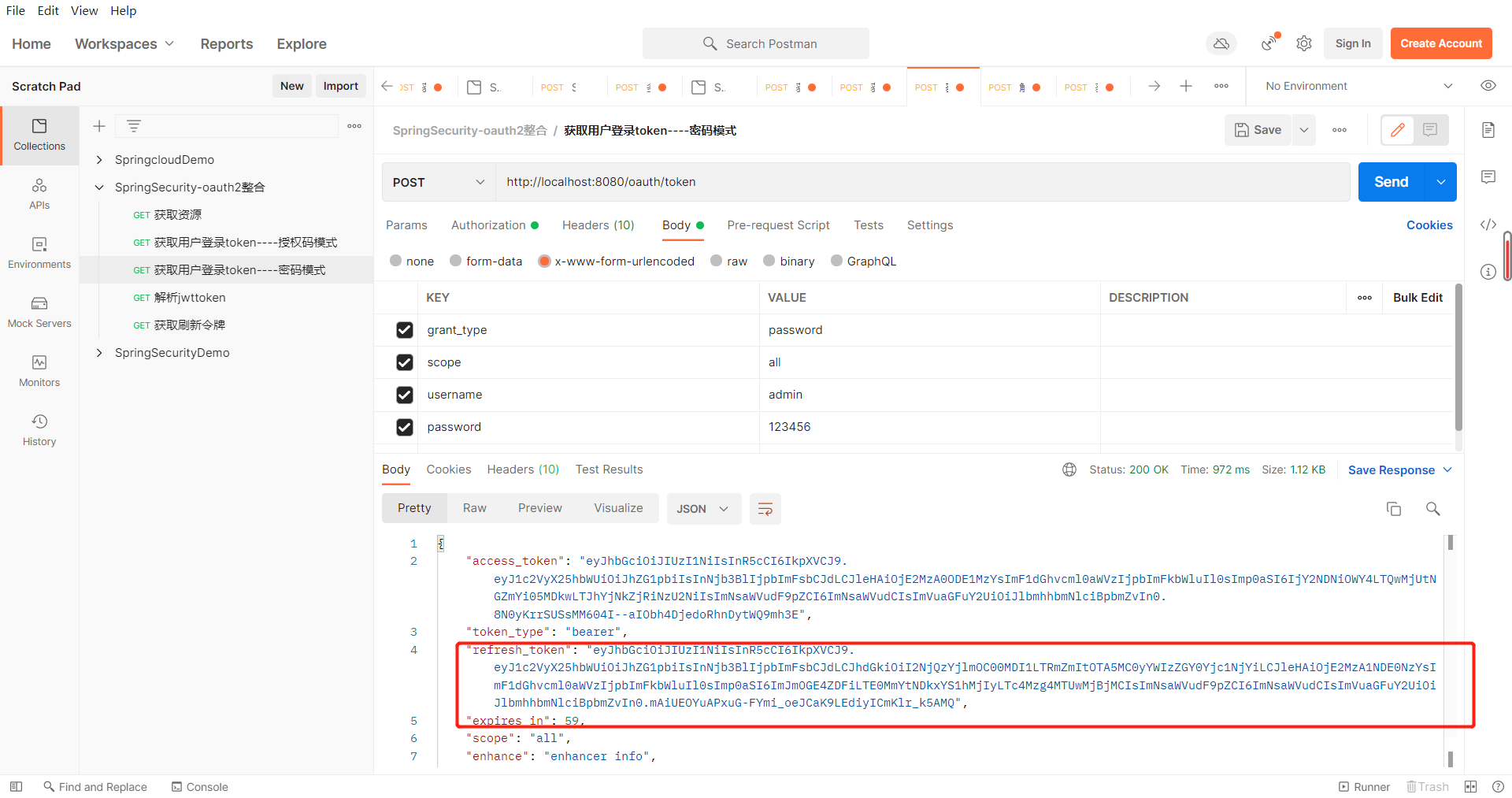


从postman测试也可以发现，我们在header里边配了一个参数，二服务器的解析代码也是一样的。

1. **最后我们在看一下如何刷新令牌**
2. 我们只需要去授权服务器的配置类里边，配置一下令牌过期时间、刷新令牌的过期时间和添加一个刷新令牌；



1. 接下来取postman获取令牌，看看结果

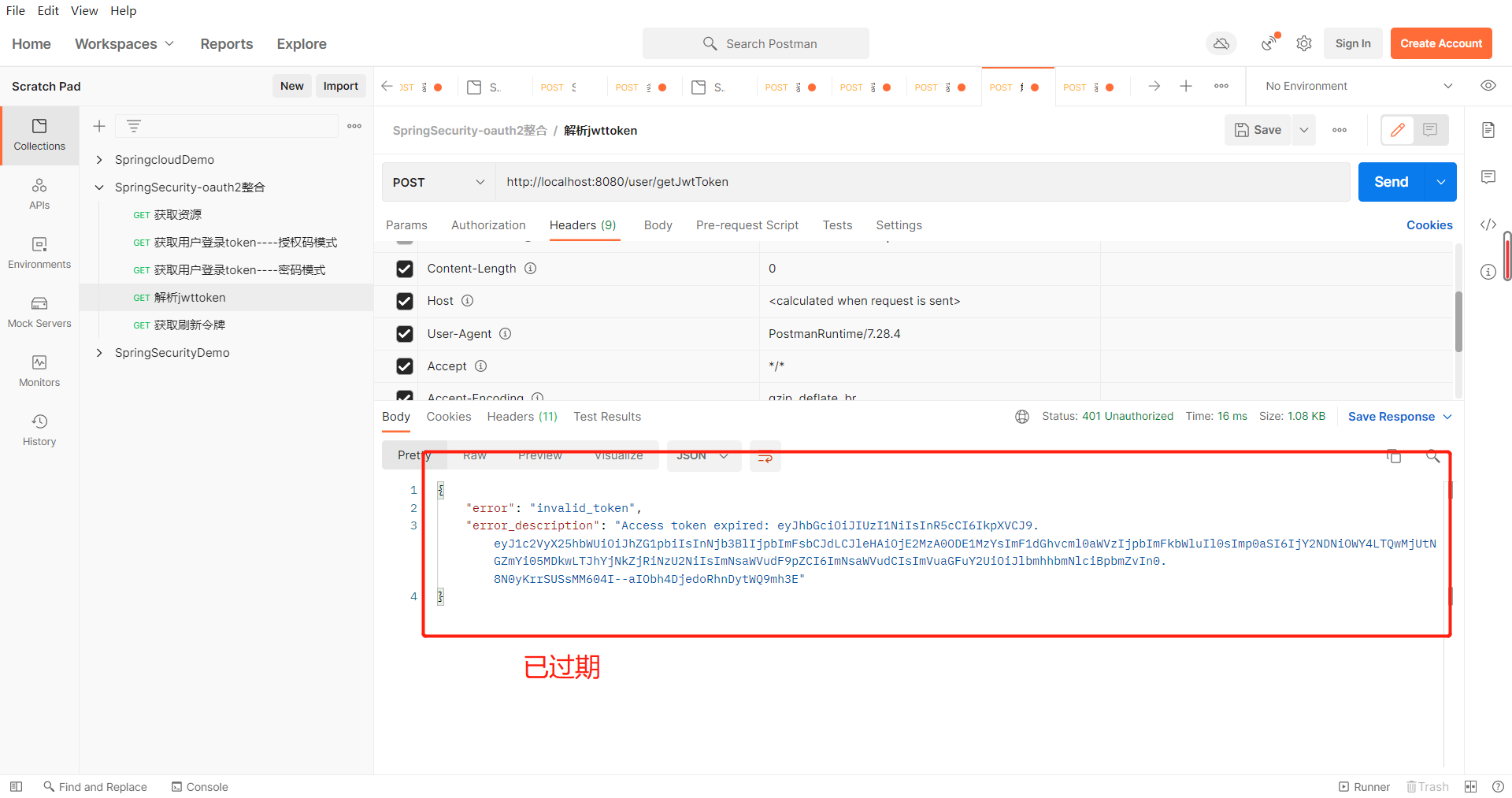


可以看出，已经多了一个refresh\_token。

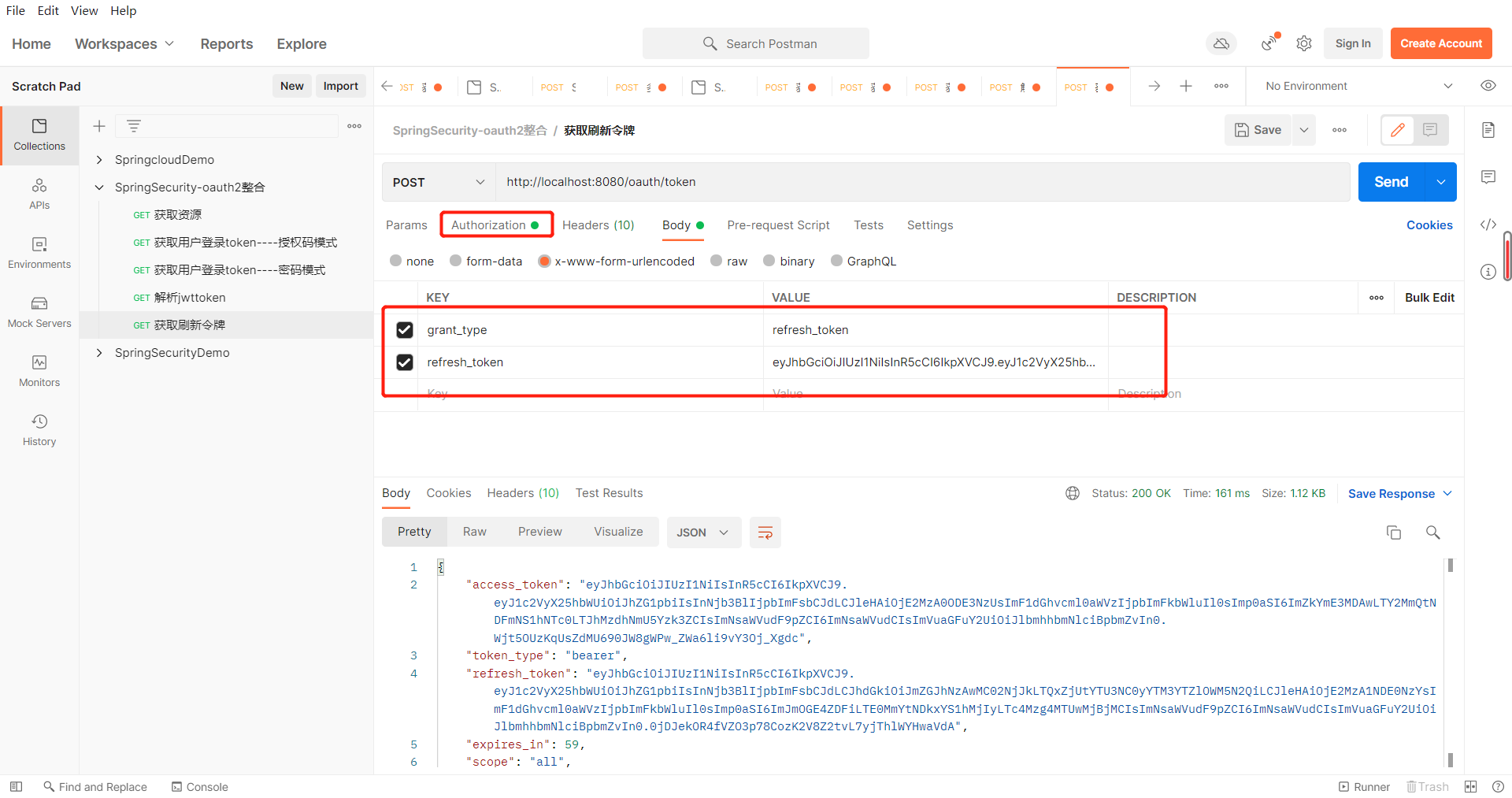
1. 复制refresh\_token去重新获取新的token

因为我们设置的令牌过期时间是60s，所以等过期了，我们再去复制；

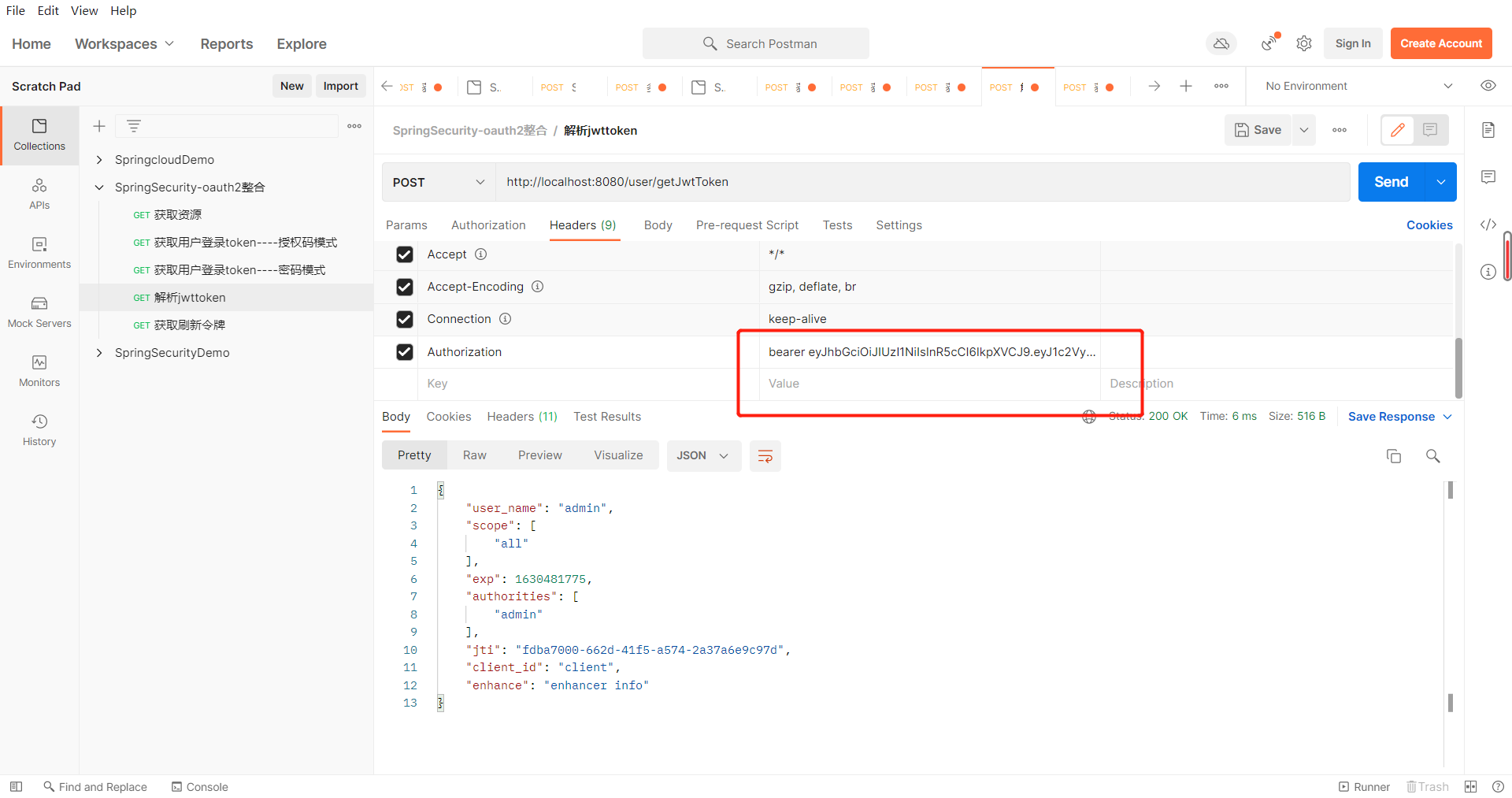
然后这里获取刷新令牌，只需要2个参数就行了：



获取到了新的令牌



在拿着新的令牌去访问资源：



完全没有问题

现在我们就写完了，SpringSecurity+oauth2+jwt的整合；