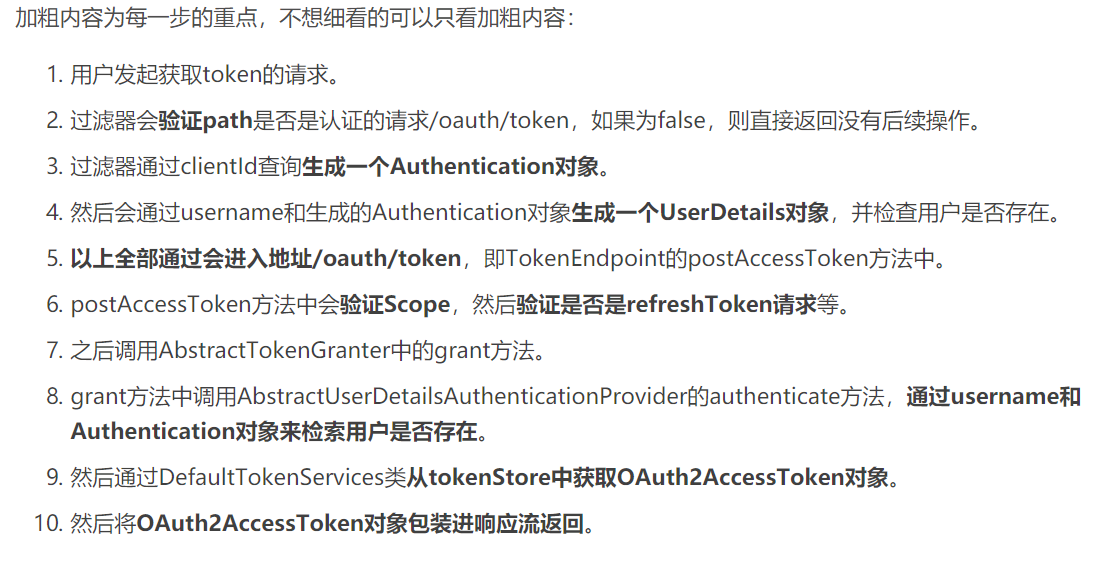
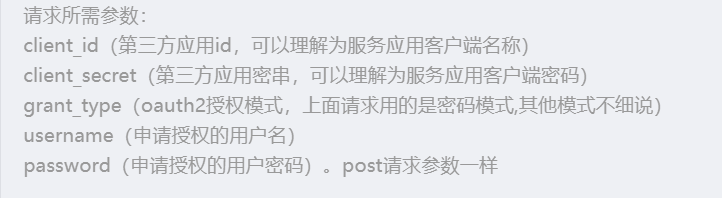
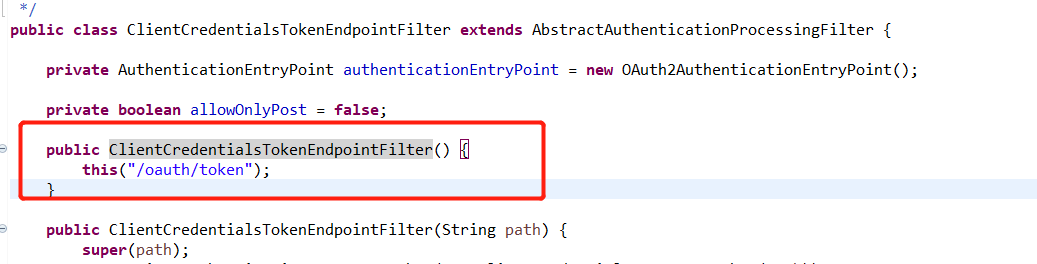
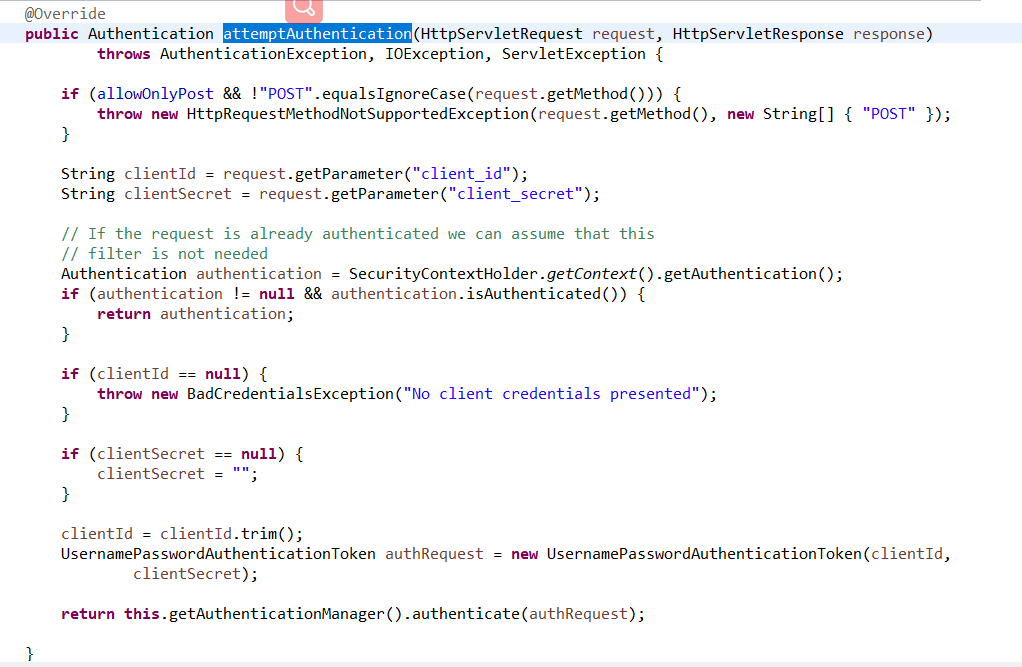
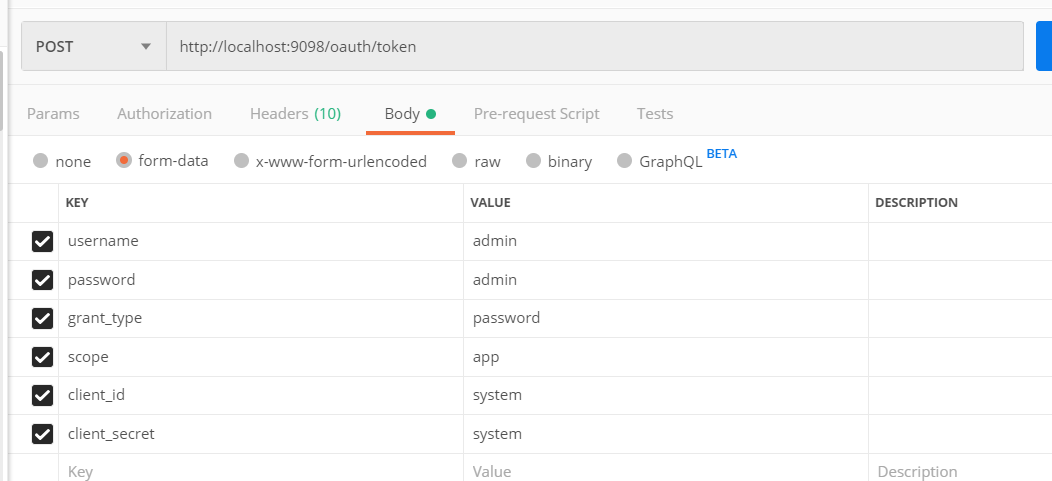
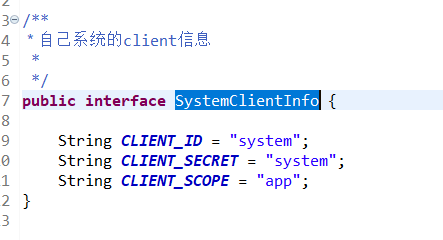
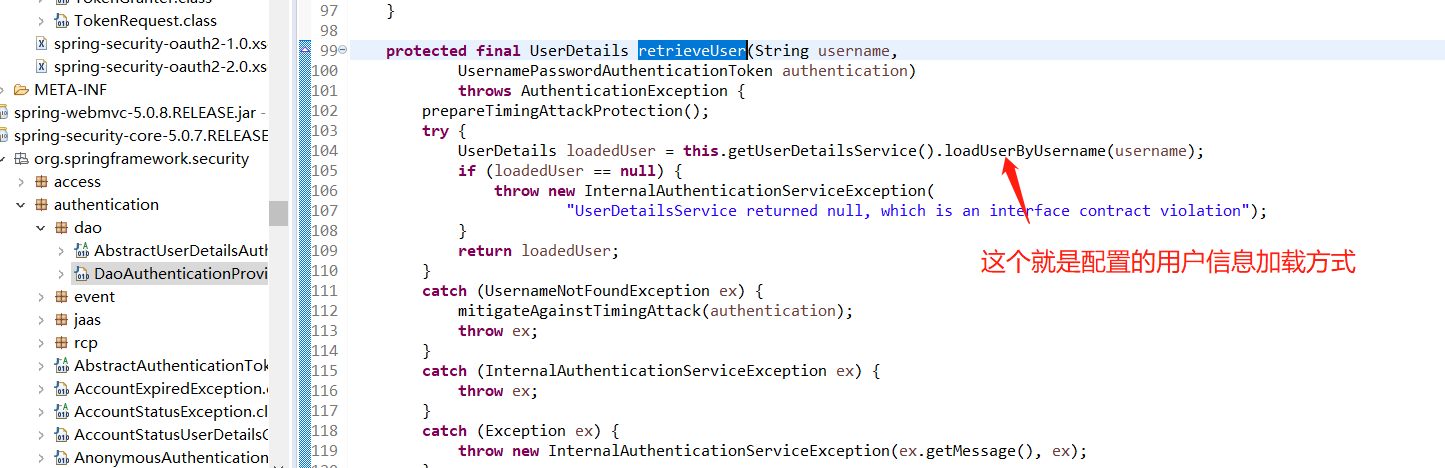
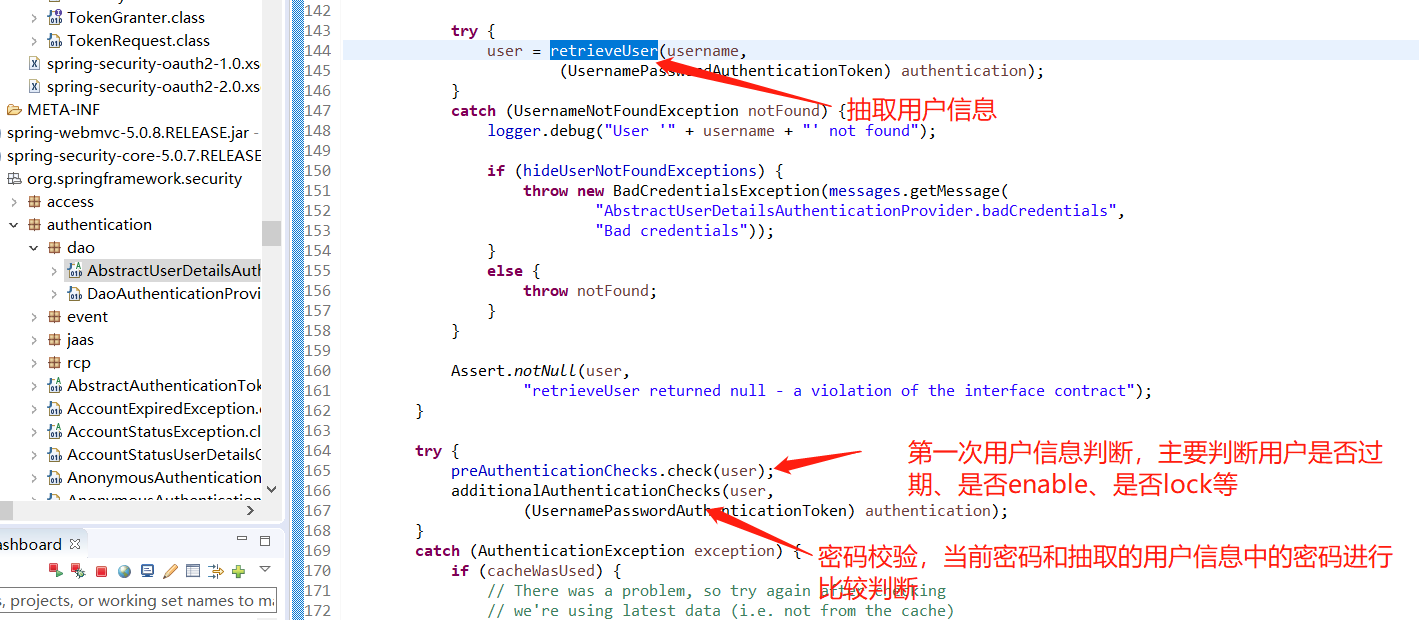
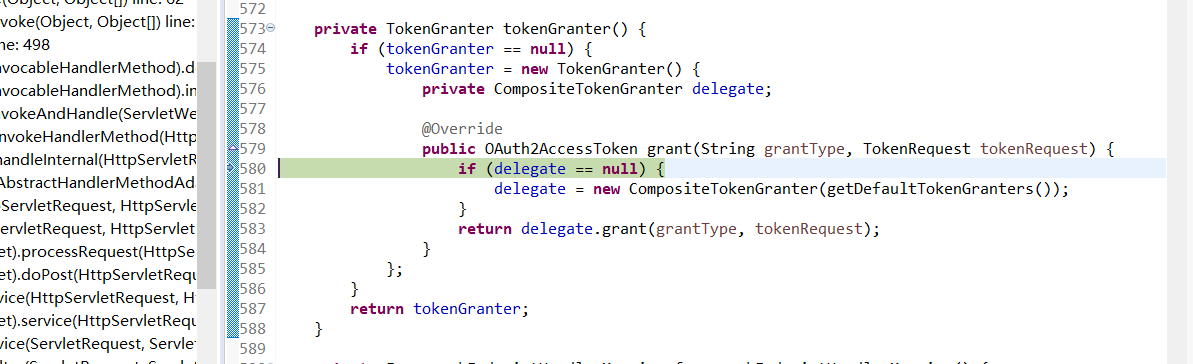
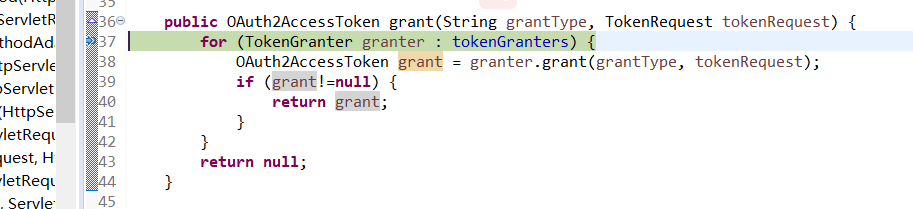
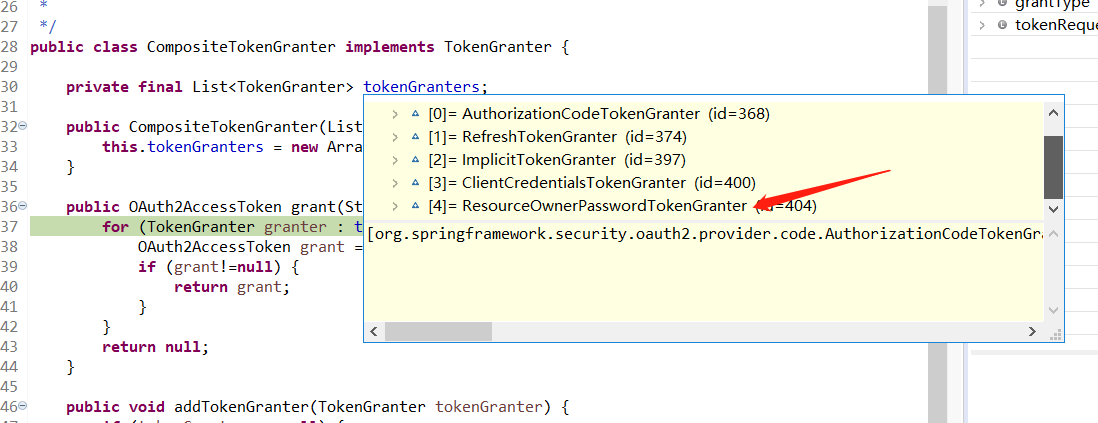
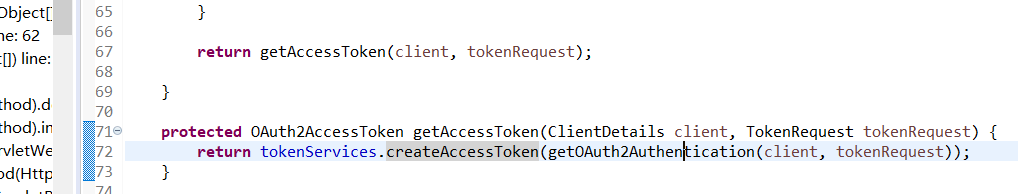
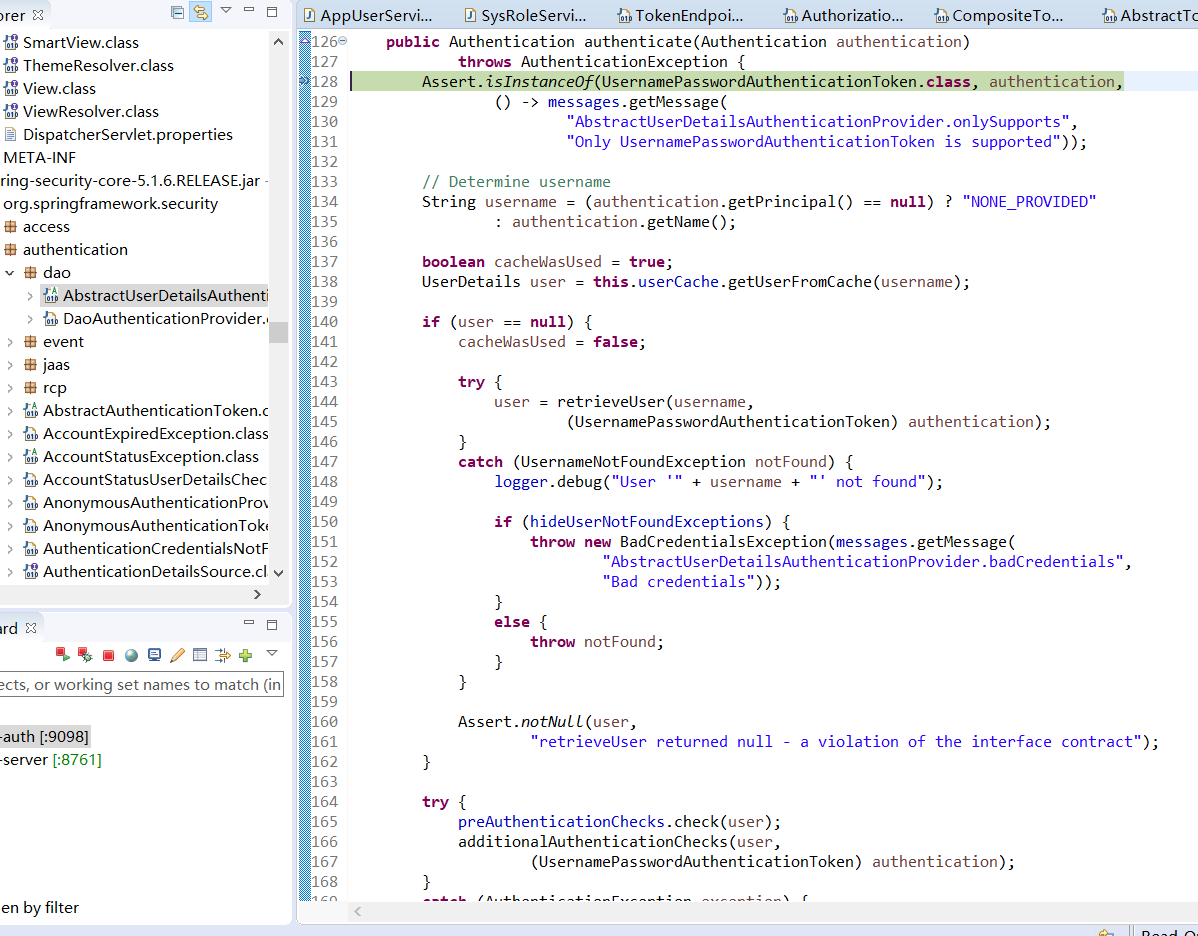
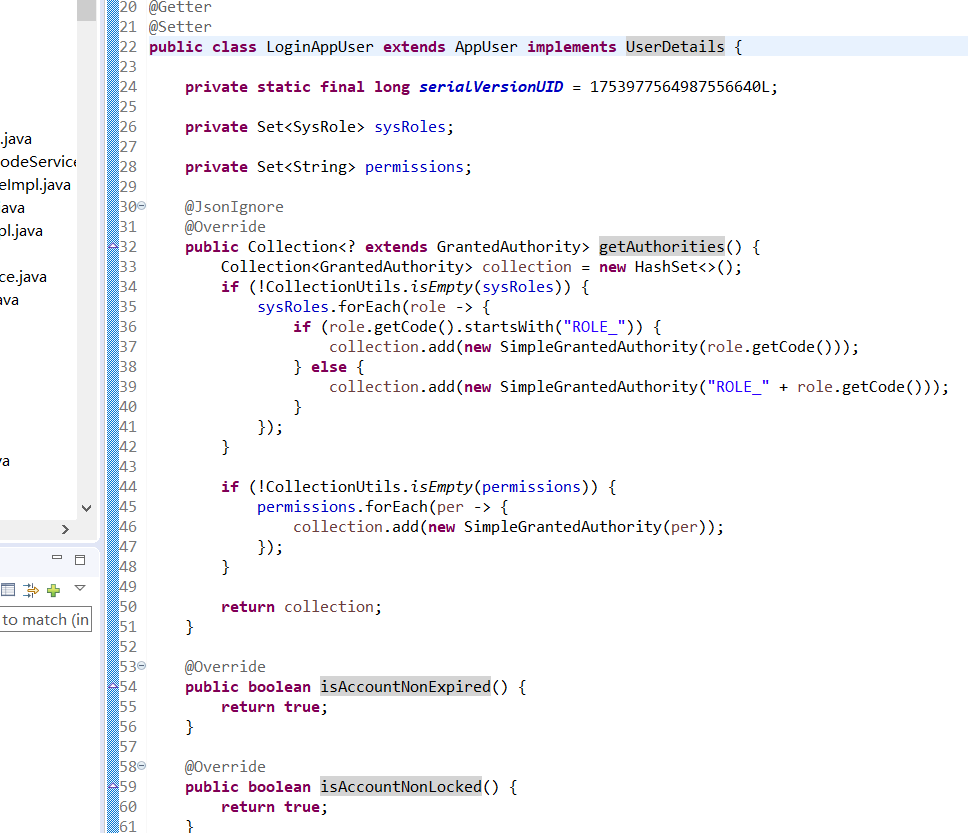
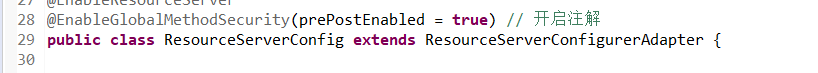
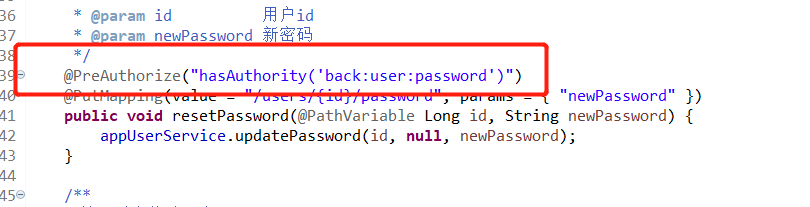
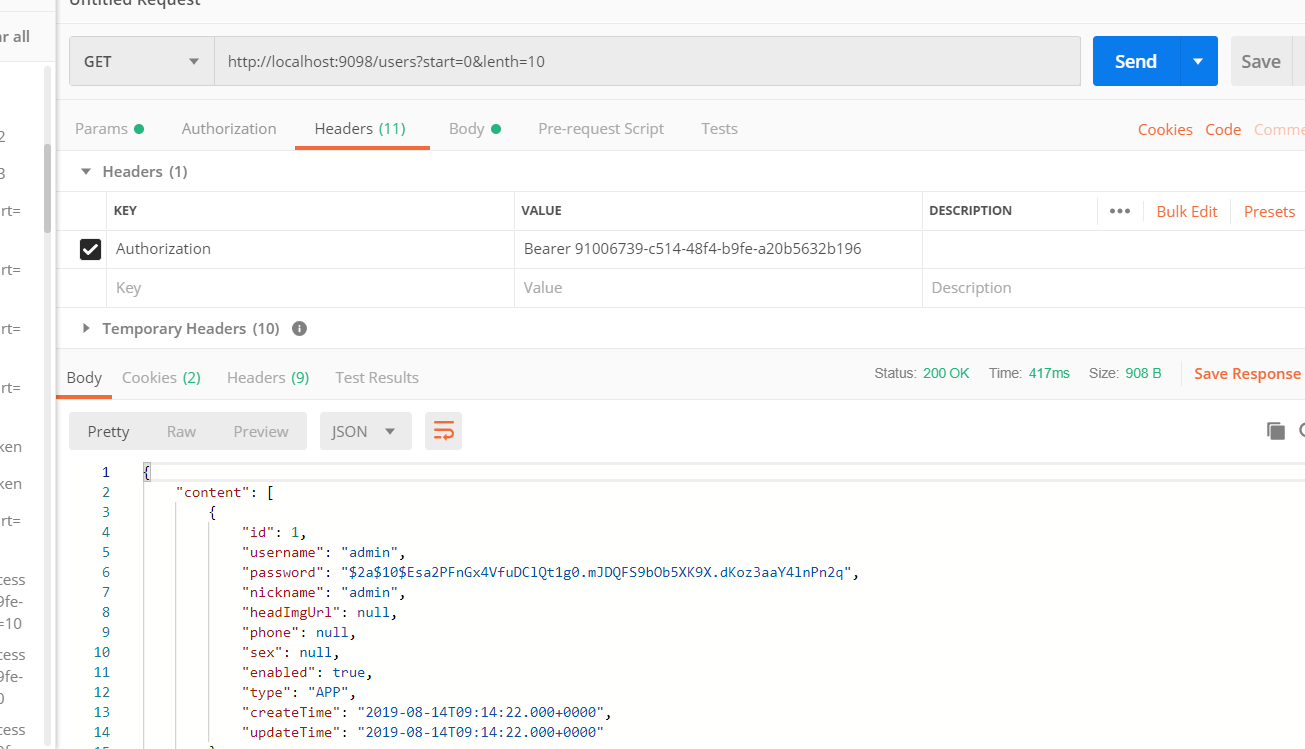
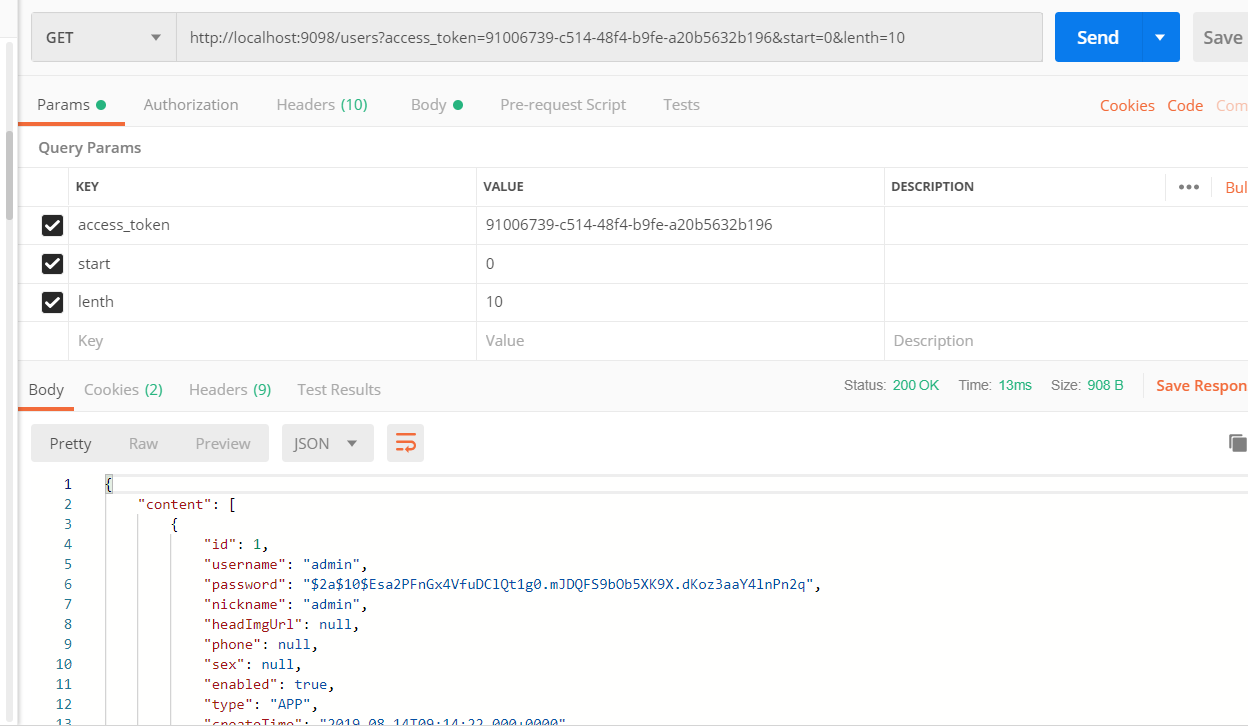
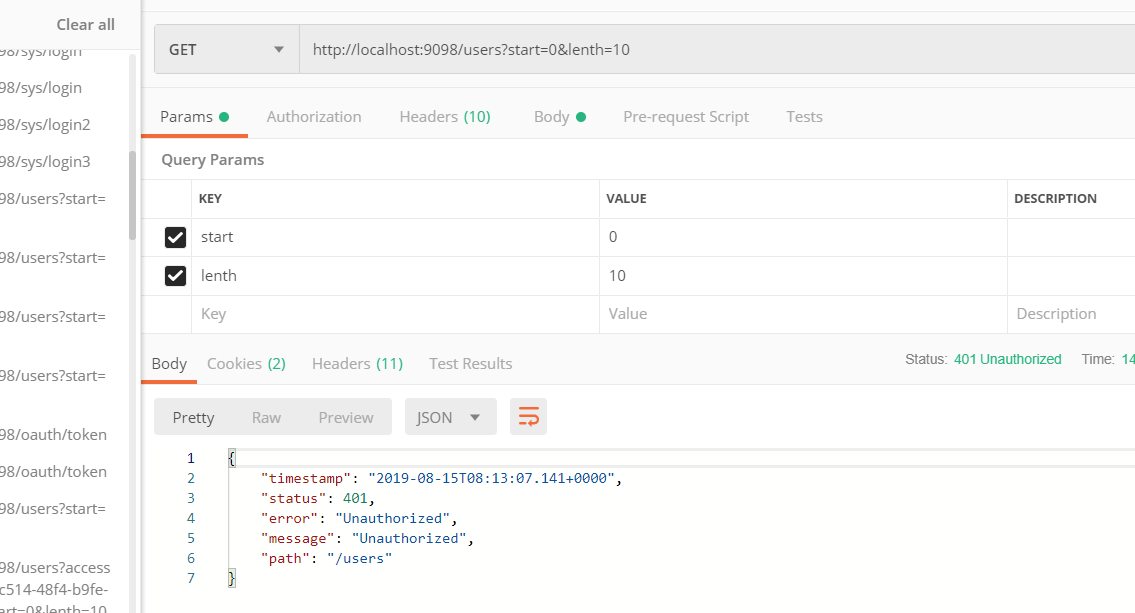
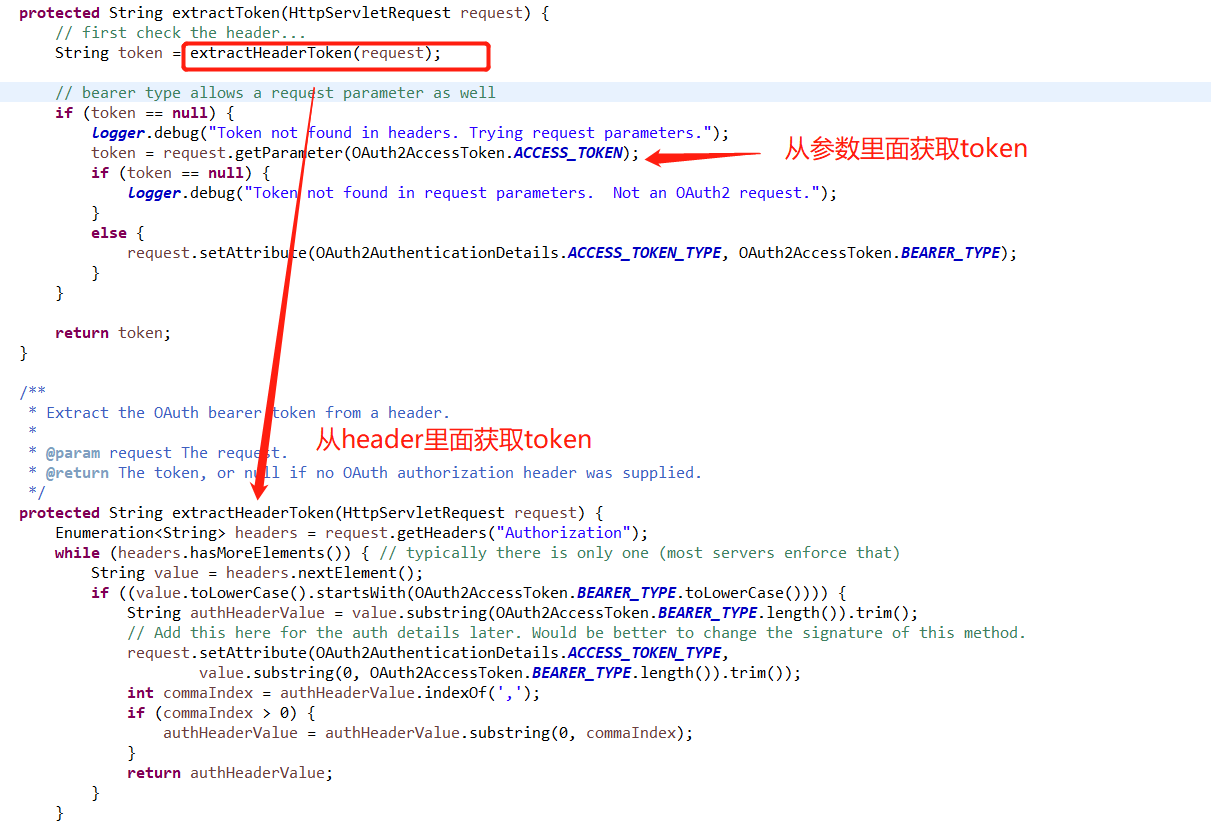
Spring cloud oauth2 登陆、密码校验、权限原理

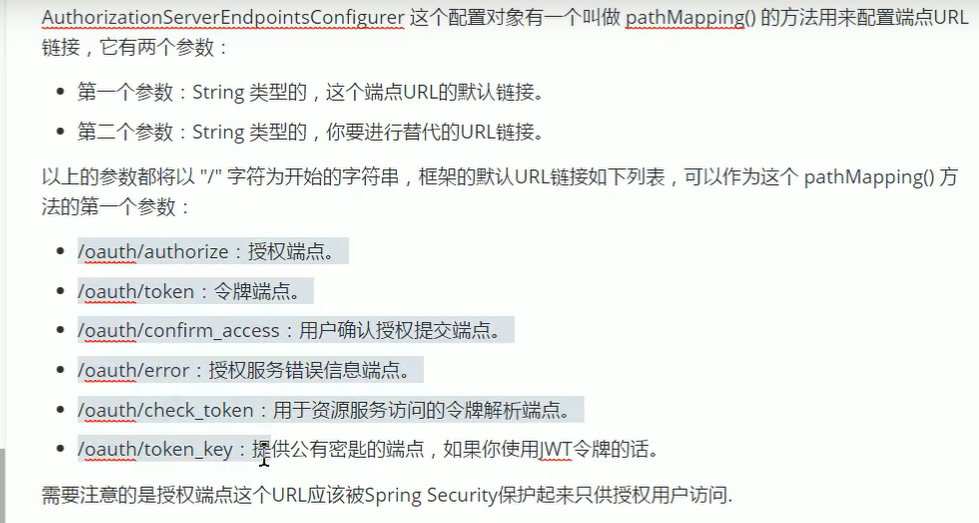
参考<https://blog.csdn.net/bluuusea/article/details/80284458>

<http://blog.didispace.com/spring-security-oauth2-xjf-2/>

获取token的主要流程



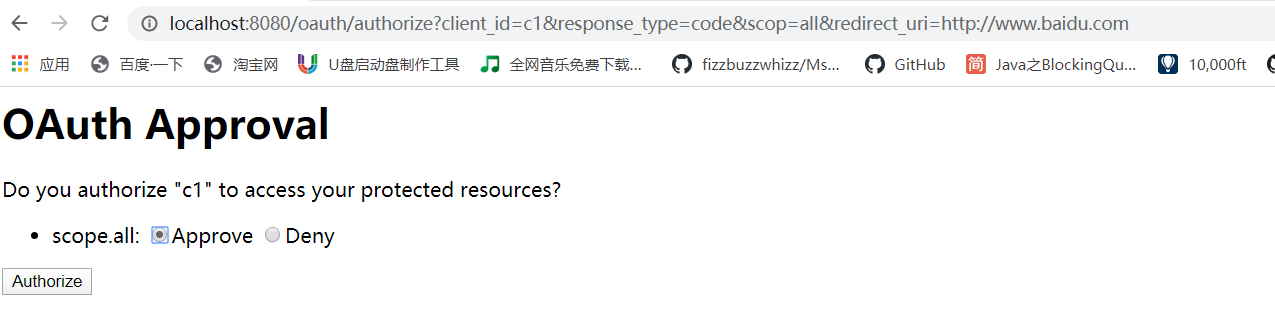
1. 底层登陆接口地址（登陆、获取access\_token）：spring-security-oauth2 包下的TokenEndpoint.class 中的 “ /oauth/token” 接口  
     
      
   一直走到下面的getTokenGgranter().grant 方法，遍历List<TokenGranter> 通过传入的GrantType，找到定位要使用的TokenGranter ，tokenServices生成accessToken，其中调用了在资源配置中的redisTokenStore，来判断该用户是否存在token，存在直接取出返回，不存在则生成个保存到redis里面并返回  
     
   其中创建RefreshToken和AccessToken的源码为  
   
2. 在请求到达/oauth/token之前经过了ClientCredentialsTokenEndpointFilter这个过滤器  
     
     
     
   认证之前执行了其中的attemptAuthentication方法，用来从请求中获取  
   client\_id,client\_secret，组装成一个UsernamePasswordAuthenticationToken作为身份标识，使用容器中的顶级身份管理器AuthenticationManager去进行身份认证（AuthenticationManager的实现类一般是ProviderManager。而ProviderManager内部维护了一个List,真正的身份认证是由一系列AuthenticationProvider去完成。而AuthenticationProvider的常用实现类则是DaoAuthenticationProvider，DaoAuthenticationProvider内部又聚合了一个UserDetailsService接口，UserDetailsService才是获取用户详细信息的最终接口
3. 接口传入参数，和调用方法 ，传入的参数必须为以下6个
   1. 微服务中通过feignClient  
        
       
   2. 本地直接post方法调用/oauth/token  
      
   3. 其中SystemClientInfo是自定义的  
      
4. 从数据库获取用户信息，与密码校验，和全局配置
   1. 全局配置  
      
   2. 获取用户信息：继承UserDetailsService接口，实现里面的通过用户名获取用户信息的方法，这里就可以通过我们自己的用户表，去调用用户服务接口获取用户信息
      1. 配置和实现用户信息获取的接口  
         
      2. 底层实现，主要是spring-security-core包下  
         AbstractUserDetailsAuthenticationProvider和DaoAuthenticationProvider两个类  
           
           
         
   3. 密码校验  
        
      密码校验的实现方法  
      
   4. grant方法调用先后顺序
      1. 1 、/oauth/token中的grant方法先调用了TokenGranter中的grant方法
      2. 2、1中的grant调用这个grant方法，其中tokenGranterss含5种TokenGranter  
         
      3. 3、方法2遍历List<TokenGranter> 调用3中的granter中的grant方法，比较granter中gantType与传入的参数的grant\_type=password比较，判断然后进行下一步的accesstoken生成  
         其中ClientCredentialsTokenGranter先调用的是下面的grant方法，在调用上面的grant  
           
         
      4. 最后调用了生成accessToken的方法  
         
5. 生成了accessToken后没有立即返回，而是调用了authenticate认证方法，进行用户密码认证  
     
   认证成功之后才返回token
6. 用户权限的设置
   1. 4中配置了的获取用户信息的本地实现接口，调用的是自己写的AppUser类封装的AppUserServer中的findByUserName()接口，返回的是自定义的LoginAppUser类，LoginAppUser继承AppUse并实现了Oauth2底层的UserDetails用户类，实现了其中的getAuthorities()方法，即获取权限集合的方法  
      
   2. 然后通过@EnableGlobalMethodSecurity(prePostEnabled = true) 的注解来开启注解  
      
   3. 最后通过在controller方法上面使用@PreAuthorize注解，并可以自动进行权限验证  
      
   4. 请求地址时可以在header\params存入access\_token  
        
        
      如果没传access\_token，则取不到用户，并被拦截  
        
        
      其中获取token的底层方法为BearerTokenExtractor.class中，主要是OAuth2AuthenticationProcessingFilter过滤器调用BearerTokenExtractor的extract()方法取出Access\_token并将其转化为authentication  
        
        
        
         
        
        
        
      



**测试**

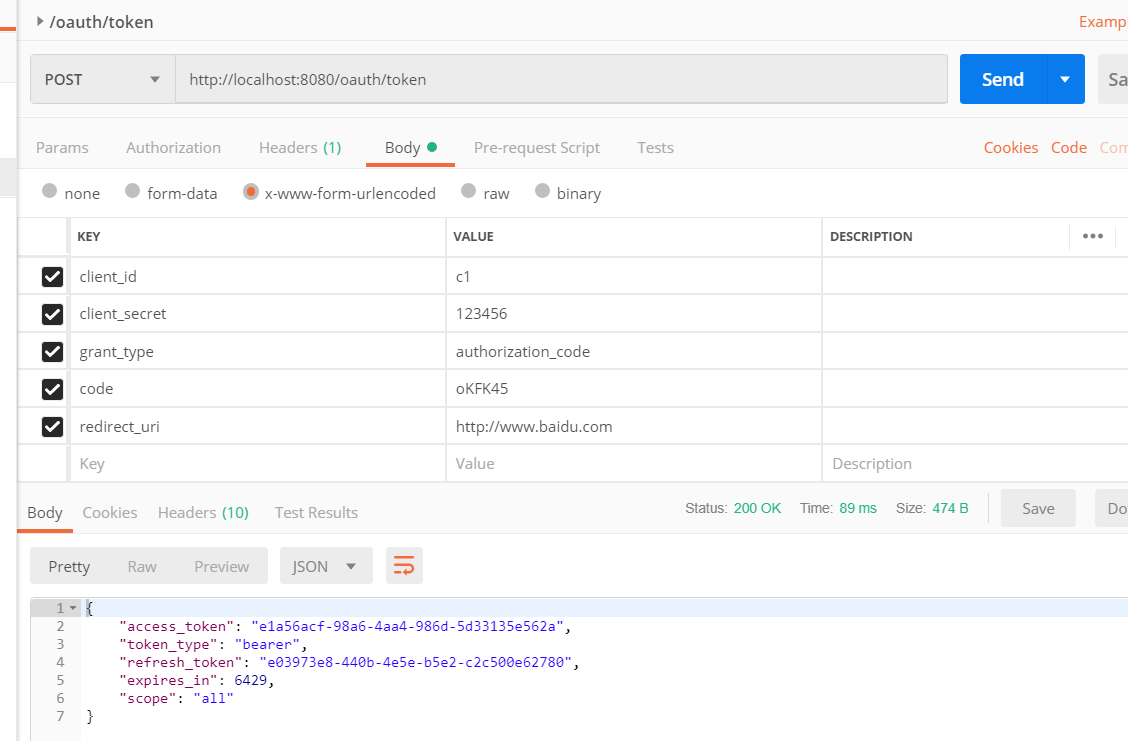
**授权码模式**，返回授权码code，然后通过code再次调用/oauth/token 获取token

<http://localhost:8080/oauth/authorize?client_id=c1&response_type=code&scop=all&redirect_uri=http://www.baidu.com>



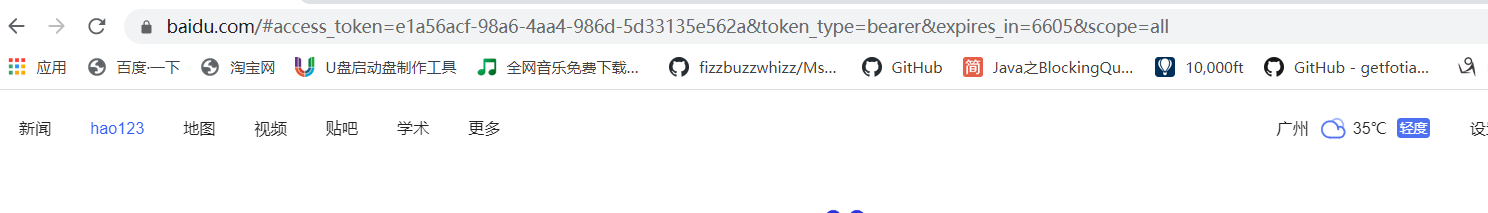


通过code去获取token

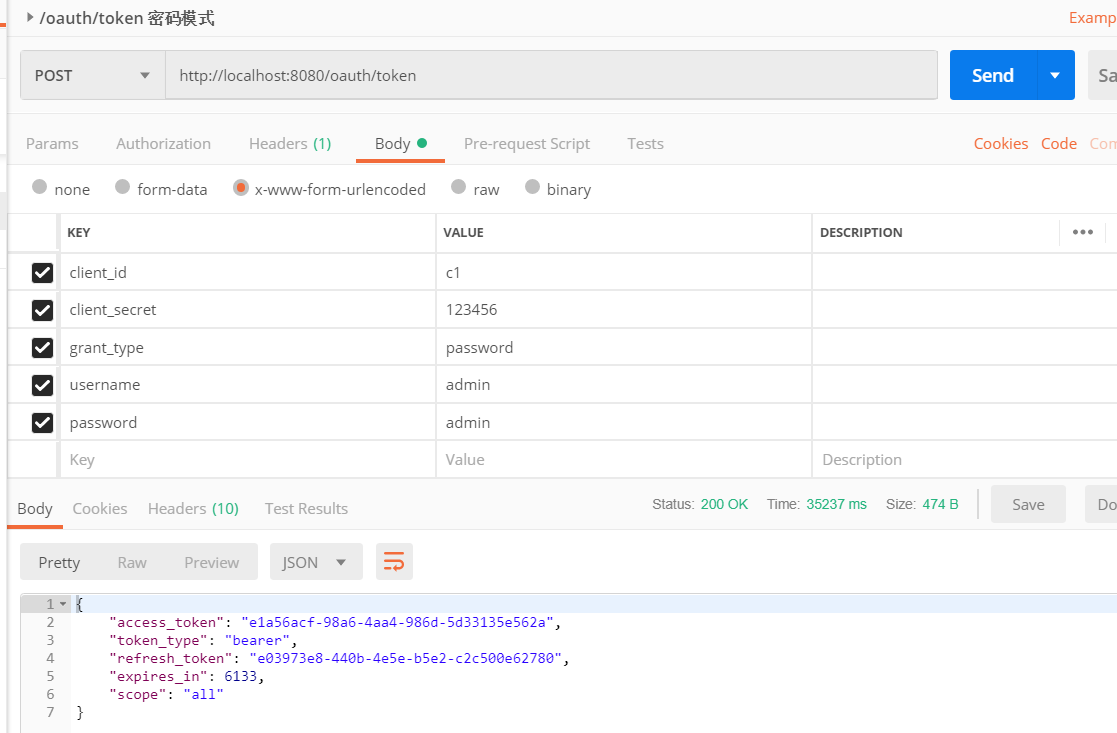


**授权码简化模式**，直接返回token

[http://localhost:8080/oauth/authorize?client\_id=c1&response\_type=token&scop=all&redirect\_uri=http://www.baidu.com](http://localhost:8080/oauth/authorize?client_id=c1&response_type=code&scop=all&redirect_uri=http://www.baidu.com)



**密码模式**，传入client id secret 和username password 知道grant\_type为password直接返回token



**客户端模式**，最简单，直接传入client id secret，直接返回token

