阳光e购-[springsecurity oauth2]

设计方案

阳光易采科技有限公司

**版本历史**

|  |  |  |  |
| --- | --- | --- | --- |
| 文档版本 | 更新日期 | 说明 | 作者 |
| Version1 | 2019-09-26 | Springsecurity oauth2 | 侯泳 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

[1. 总述 4](#_Toc26007)

[1.1 Springsecurity概述 4](#_Toc1447)

[1.2 Springsecurity认证 & 授权 4](#_Toc10782)

[2.1 OAuth2介绍 4](#_Toc23476)

[2.2 oauth2 定义 4](#_Toc9005)

[2. Oauth2四种模式 5](#_Toc19586)

[2.1授权码模式(Authorization code Grant)： 5](#_Toc21488)

[2.1.1导入项目 5](#_Toc27471)

[2.1.2授权码授权流程 6](#_Toc13418)

[2.1.3.流程演示 6](#_Toc8932)

[1.请求认证服务获取授权码： 6](#_Toc21445)

[2.通过授权码，获取令牌 8](#_Toc32547)

[2.1.4 通过token,访问资源服务 10](#_Toc27192)

[1.没有携带token访问资源服务 10](#_Toc11227)

[2.没有携带正确得token访问 11](#_Toc16129)

[3.携带正确得token访问 11](#_Toc6994)

[3.1密码模式(Resource Owner Password Credentials Grant)： 12](#_Toc11075)

[3.1.1导入项目 12](#_Toc6707)

[3.1.2密码模式流程 13](#_Toc2251)

[3.1.3密码流程演示 13](#_Toc951)

[1.请求获取token 13](#_Toc5130)

[2.参数错误演示 15](#_Toc8255)

[3.正常返回值 16](#_Toc17090)

[3.1通过token,访问资源服务 16](#_Toc18888)

[1.令牌不对或者过期 17](#_Toc23863)

[2.权限不足演示 17](#_Toc3286)

[3.权限通过 19](#_Toc21995)

[4.1隐式授权模式（Implicit Grant） 20](#_Toc29121)

[4.1.1 隐式授权流程图 20](#_Toc31787)

[4.1.2 隐式授权模式说明 20](#_Toc14747)

[5.1客户端凭证模式（Client Credentials Grant） 22](#_Toc18414)

[5.1.1客户端凭证流程图 22](#_Toc29439)

[5.1.2客户端凭证模式说明 22](#_Toc12766)

[6.1刷新TOKEN 24](#_Toc11905)

[6.1.1流程图 24](#_Toc24707)

[6.1.2 刷新token说明 24](#_Toc6057)

# 总述

## 1.1 Springsecurity概述

一个能够为基于Spring的企业应用系统提供声明式的安全訪问控制解决方式的安全框架（简单说是对访问权限进行控制嘛），应用的安全性包括用户认证（Authentication）和用户授权（Authorization）两个部分。用户认证指的是验证某个用户是否为系统中的合法主体，也就是说用户能否访问该系统。用户认证一般要求用户提供用户名和密码。系统通过校验用户名和密码来完成认证过程。用户授权指的是验证某个用户是否有权限执行某个操作。在一个系统中，不同用户所具有的权限是不同的。比如对一个文件来说，有的用户只能进行读取，而有的用户可以进行修改。一般来说，系统会为不同的用户分配不同的角色，而每个角色则对应一系列的权限。   spring security的主要核心功能为 认证和授权，所有的架构也是基于这两个核心功能去实现的。

## 1.2 Springsecurity认证 & 授权

认证（Authentication）

用户认证指的是验证某个用户是否为系统中的合法主体，即用户能否访问该系统。用户认证一般要求用户提供账号和密码，系统通过校验账号和密码来完成认证过程。

授权（Authorization）

用户授权指的是验证某个用户是否有权限执行某个操作。在一个系统中，不同用户所具有的权限通常是不同的，以一个为例，有的用户只能读，有的用户能读能写。一般来说，系统会为不同的用户分配不同的角色，而每个角色则对应一系列的权限。

1.1.2

## 2.1 OAuth2介绍

OAUTH协议为用户资源的授权提供了一个安全的、开放而又简易的标准。与以往的授权方式不同之处是OAUTH的授权不会使第三方触及到用户的帐号信息（如用户名与密码），即第三方无需使用用户的用户名与密码就可以申请获得该用户资源的授权，因此OAUTH是安全的。oAuth是Open Authorization的简写。

## 2.2 oauth2 定义

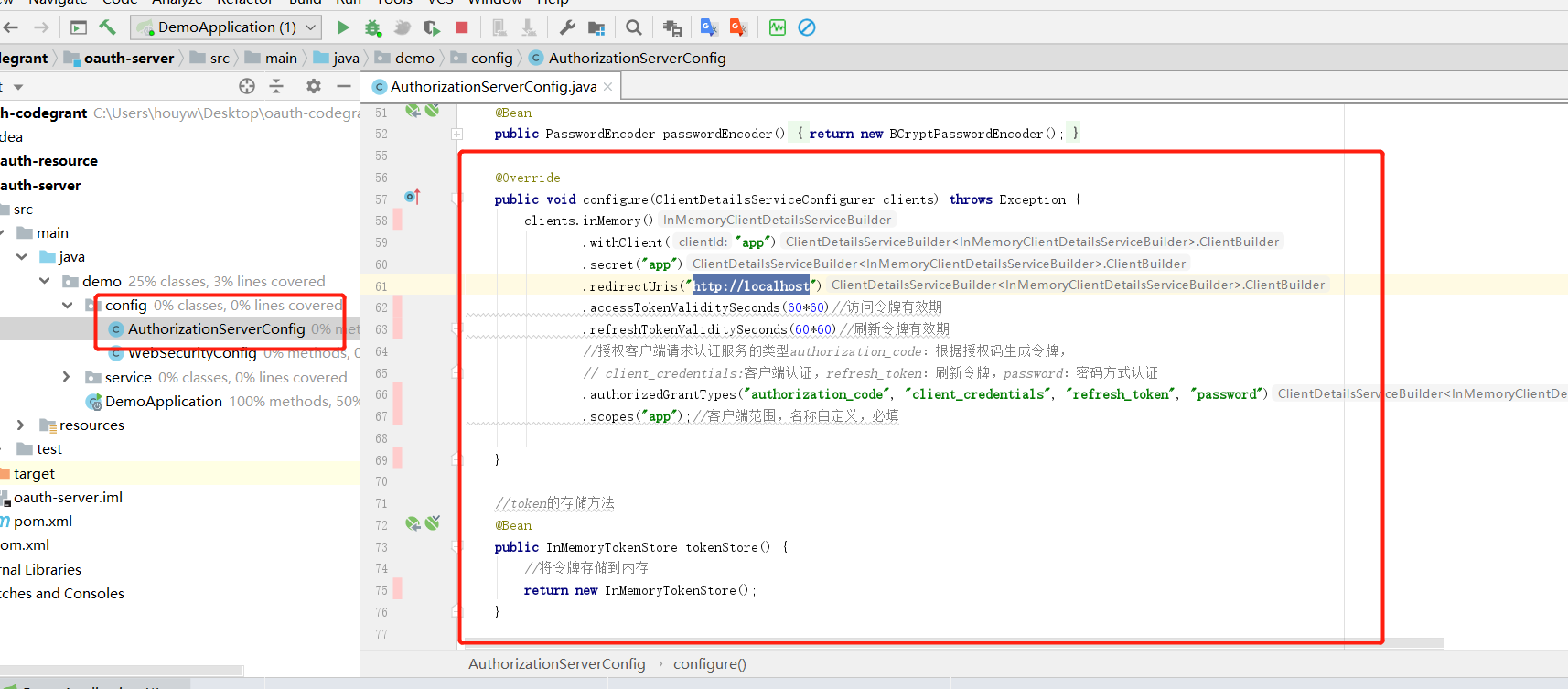
OAUTH协议为用户资源的授权提供了一个安全的、开放而又简易的标准。同时，任何第三方都可以使用OAUTH认证服务，任何服务提供商都可以实现自身的OAUTH认证服务，因而OAUTH是开放的。业界提供了OAUTH的多种实现如PHP、JavaScript，Java，Ruby等各种语言开发包，大大节约了程序员的时间，因而OAUTH是简易的。互联网很多服务如Open API，很多大公司如Google，Yahoo，Microsoft等都提供了OAUTH认证服务，这些都足以说明OAUTH标准逐渐成为开放资源授权的标准。

# Oauth2四种模式

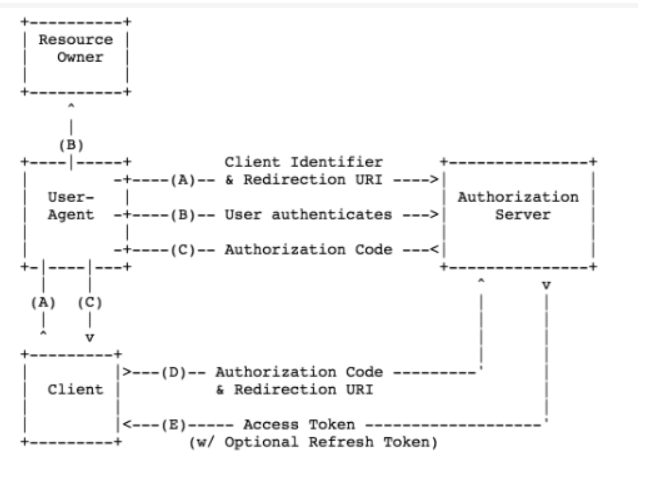
## 2.1授权码模式(Authorization code Grant)：

### 2.1.1导入项目

（授权码模式得client\_id,client\_secret保存在内存中,token保存在mysql数据库中）



### 2.1.2授权码授权流程



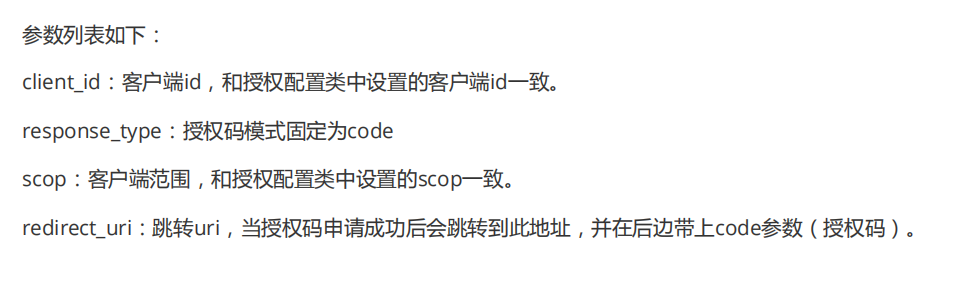
1、客户端请求第三方授权 2、用户(资源拥有者)同意给客户端授权 3、客户端获取到授权码，请求认证服务器申请令牌 4、认证服务器向客户端响应令牌 5、客户端请求资源服务器的资源 6、资源服务器返回受保护资源

### 2.1.3.流程演示

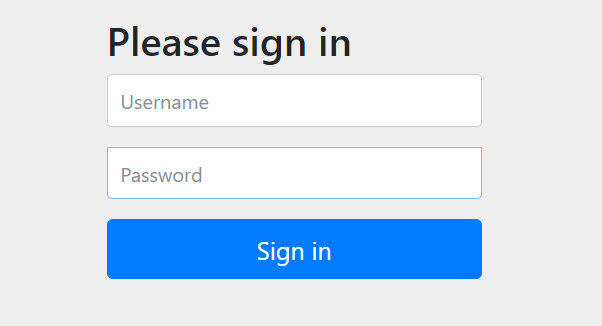
#### 1.请求认证服务获取授权码：

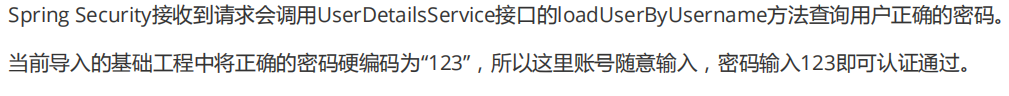
Get请求：

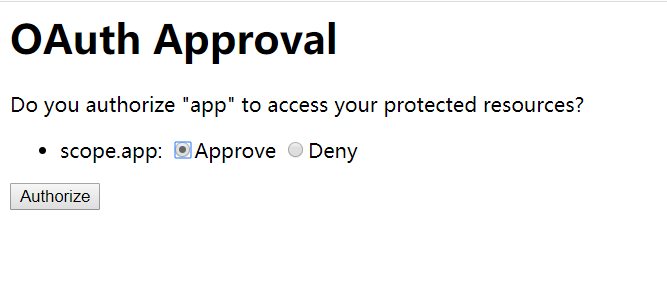
http://localhost:8888/oauth/authorize?client\_id=app&redirect\_uri=http://localhost&response\_type=code&scope=app



首先跳转到登陆页面





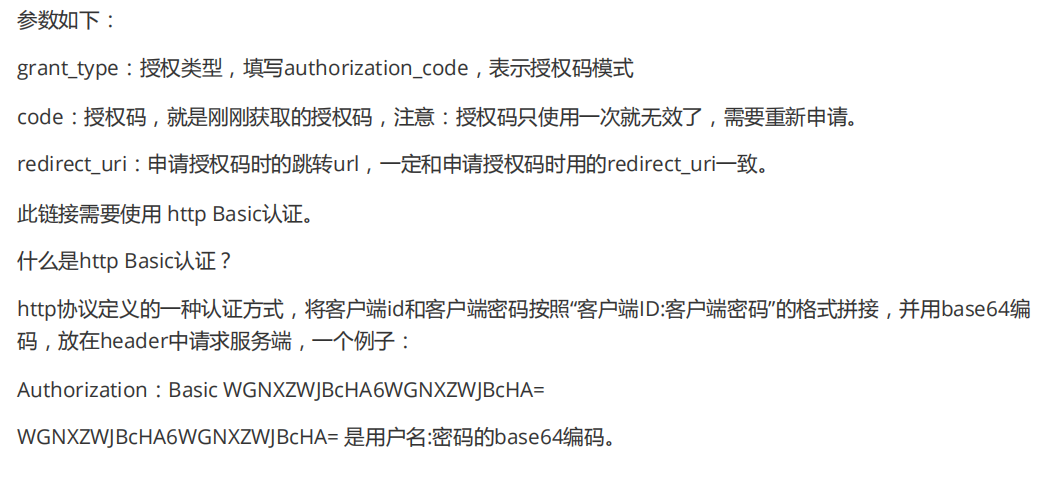


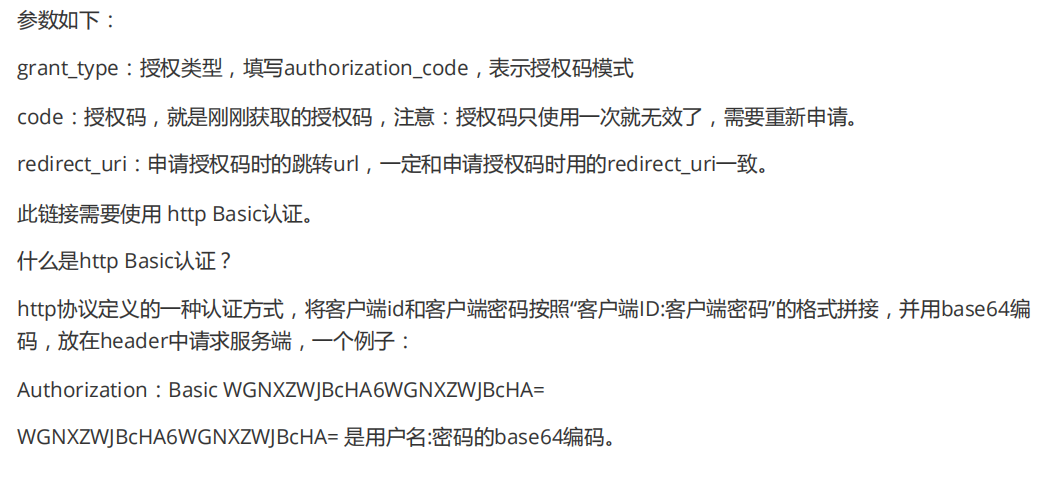
点击同意，在指定得redierct\_uri返回授权码

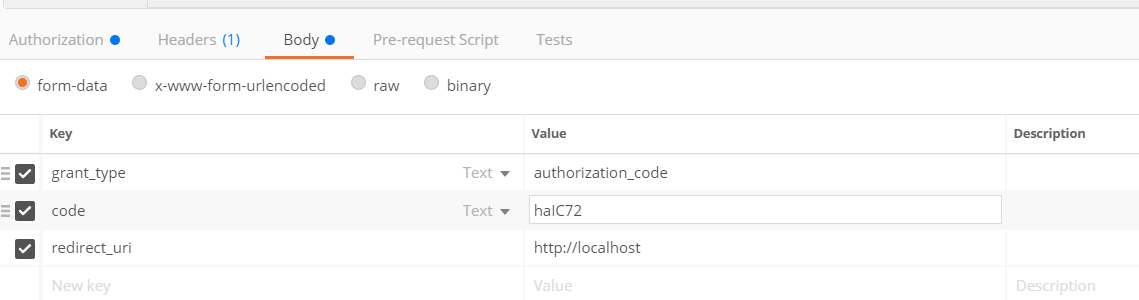


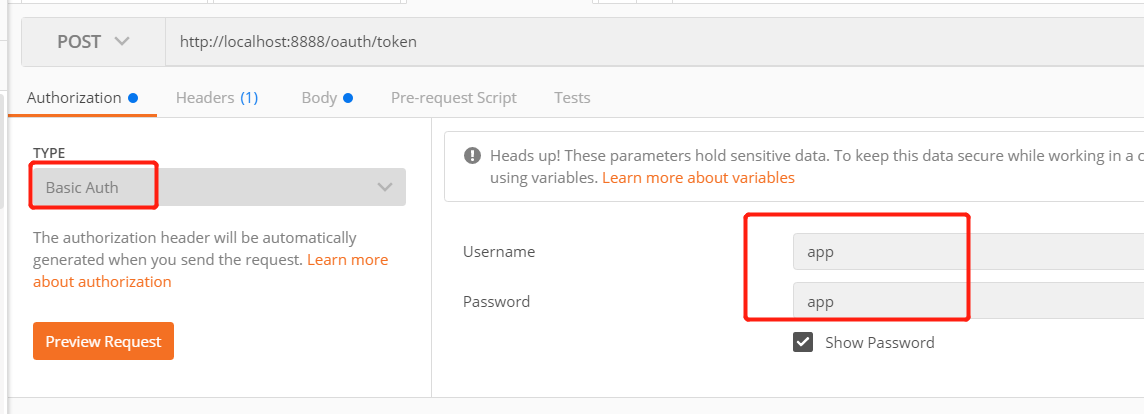
#### 2.通过授权码，获取令牌

Post请求，使用postman测试 : <http://localhost:8888/oauth/token>





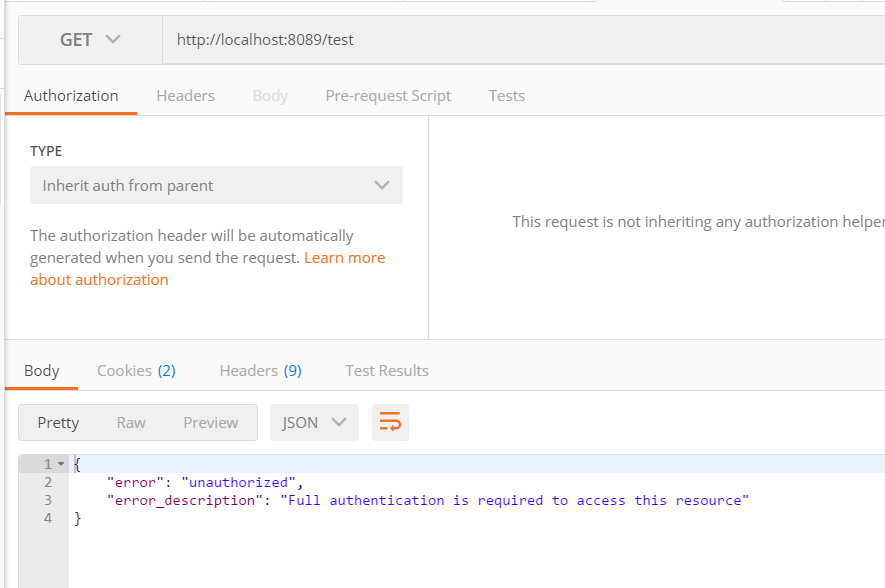




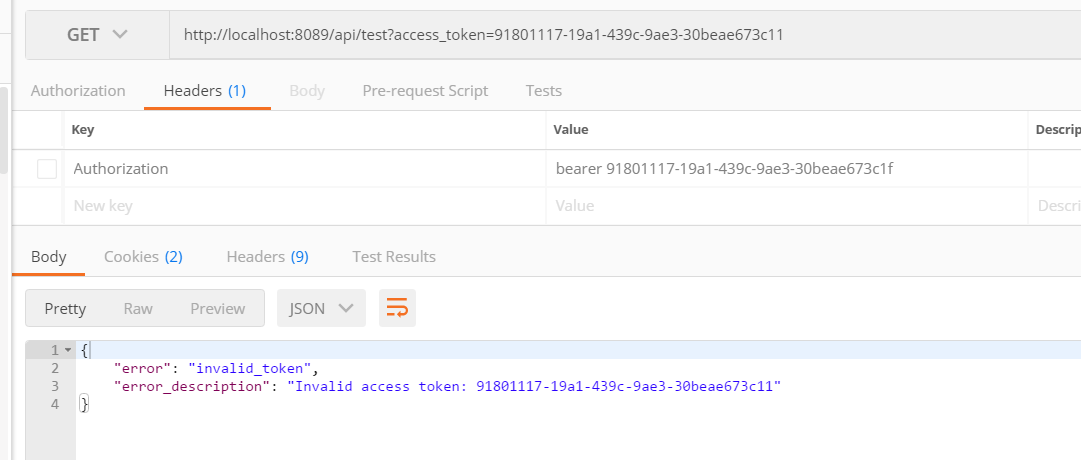


### 2.1.4 通过token,访问资源服务

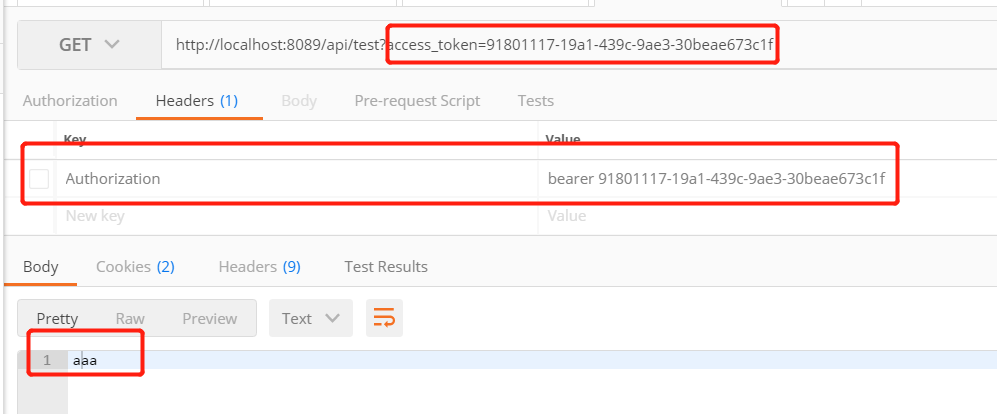
#### 1.没有携带token访问资源服务



#### 2.没有携带正确得token访问



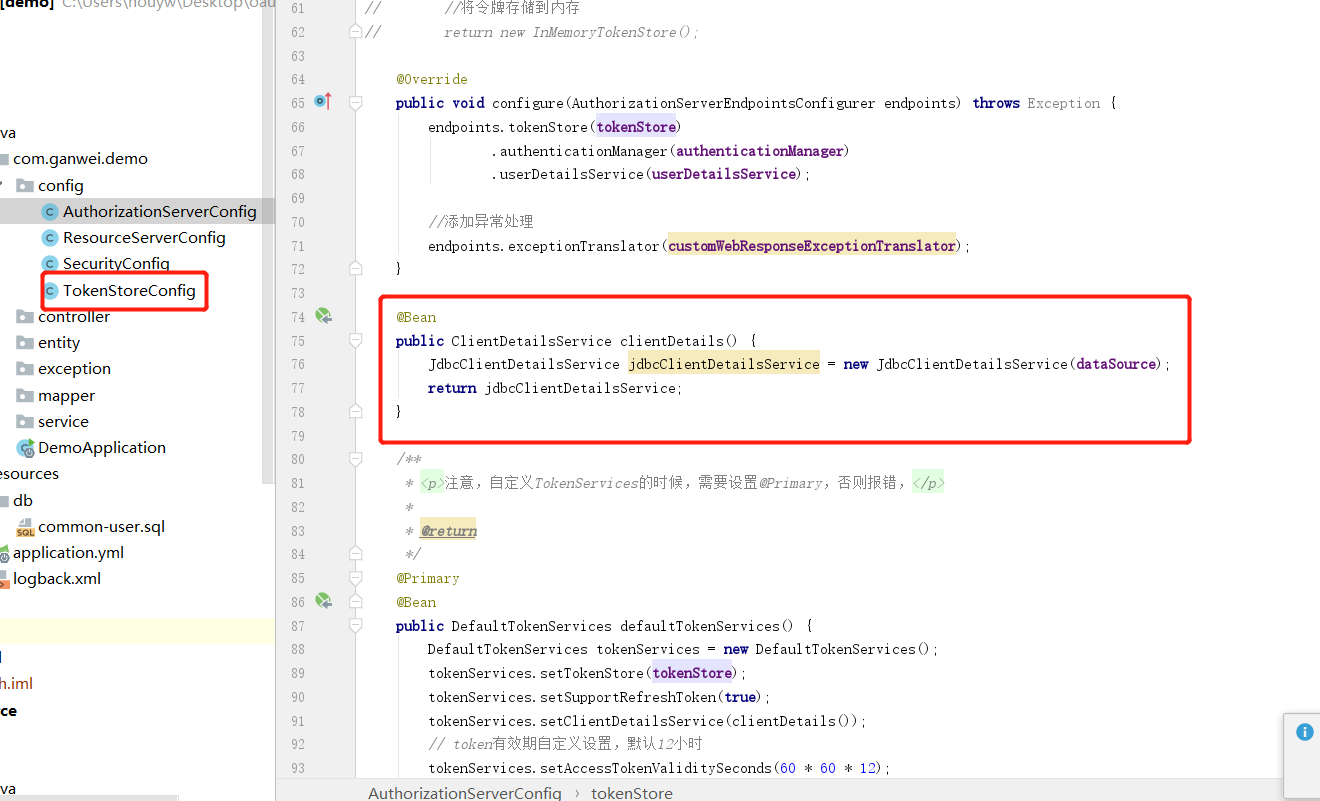
#### 3.携带正确得token访问



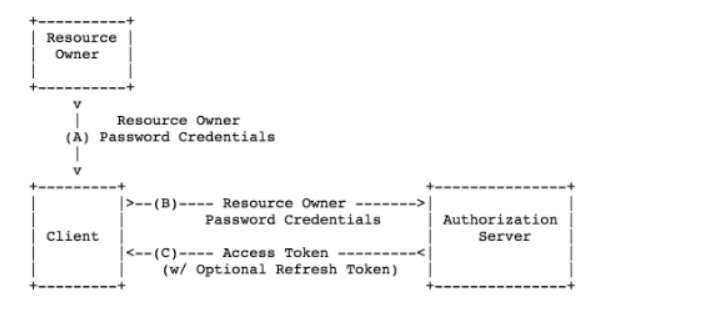
## 3.1密码模式(Resource Owner Password Credentials Grant)：

### 3.1.1导入项目

（密码模式得client\_id,client\_secret保存在mysql数据库中,token保存在redis中）



### 3.1.2密码模式流程

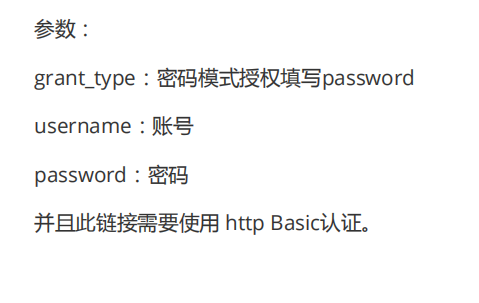


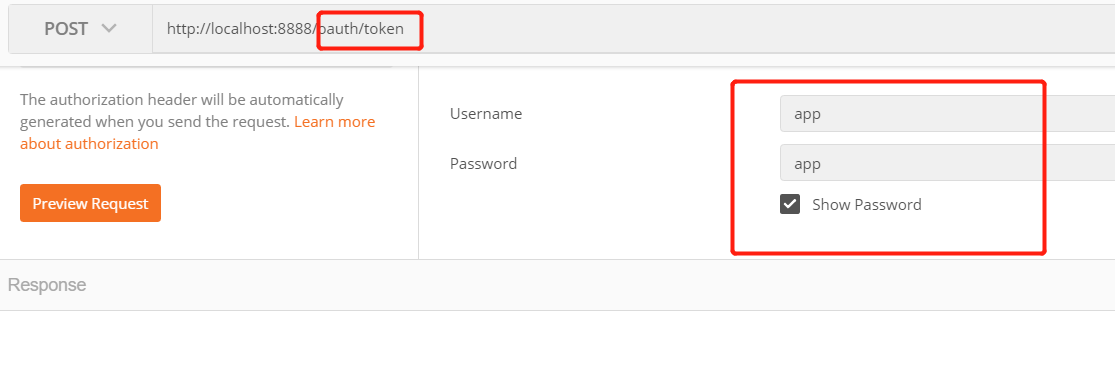
密码模式（Resource Owner Password Credentials）与授权码模式的区别是申请令牌不再使用授权码，而是直接 通过用户名和密码即可申请令牌。

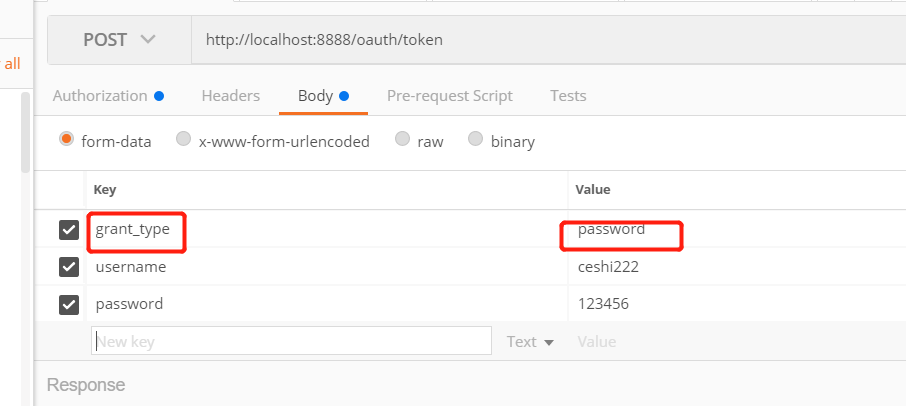
### 3.1.3密码流程演示

#### 1.请求获取token

Post请求：http://localhost:8888/oauth/token

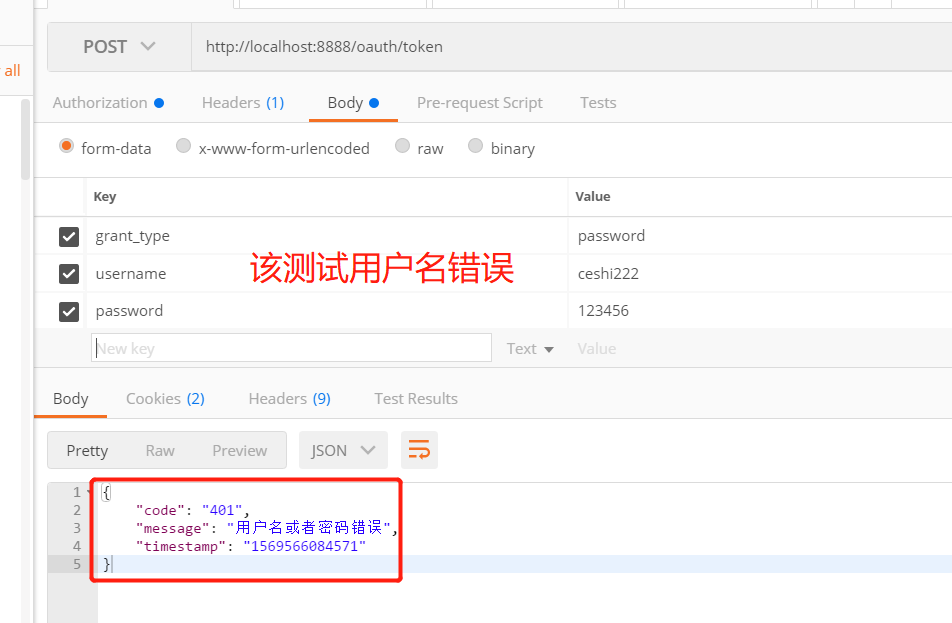




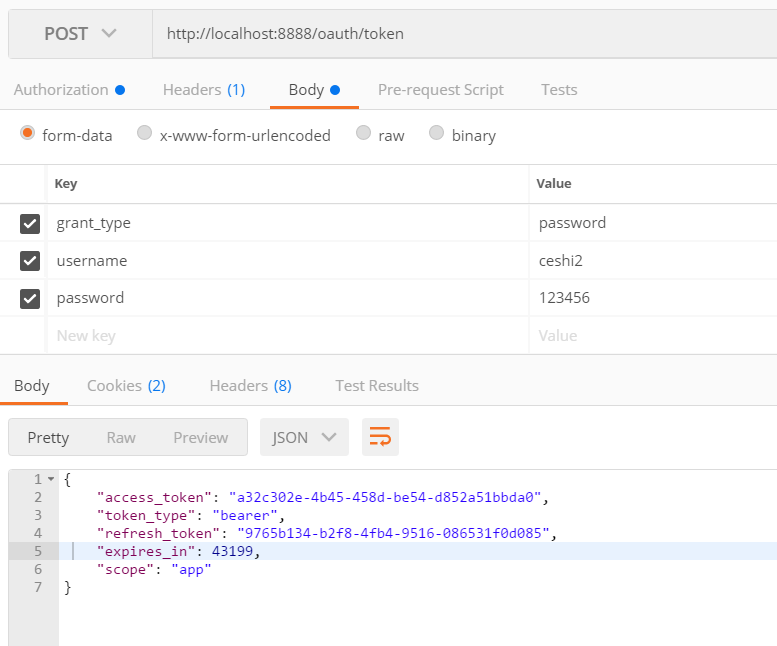


#### 2.参数错误演示

如果用户名，密码，client\_Id，client\_serect都会报错，本实例已经对异常进行处理

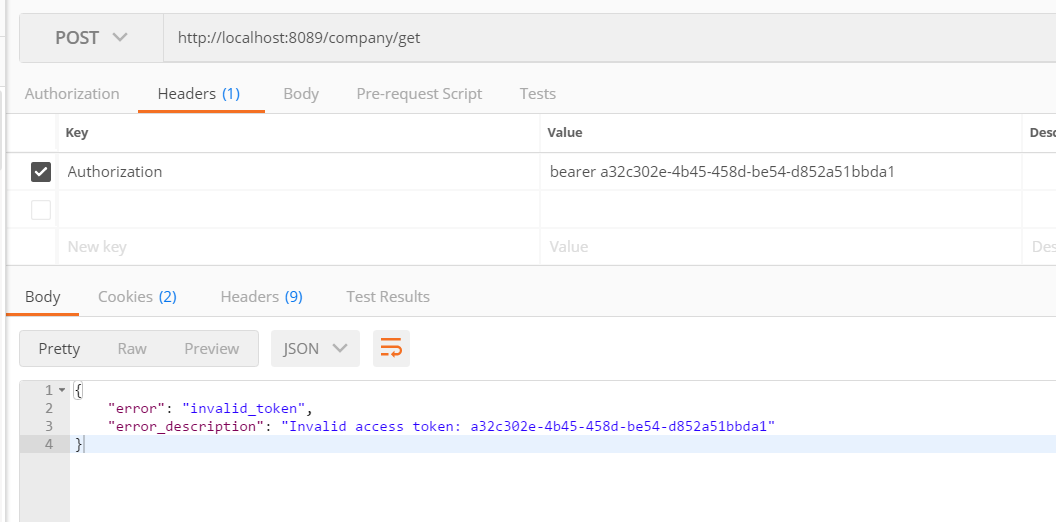


#### 3.正常返回值



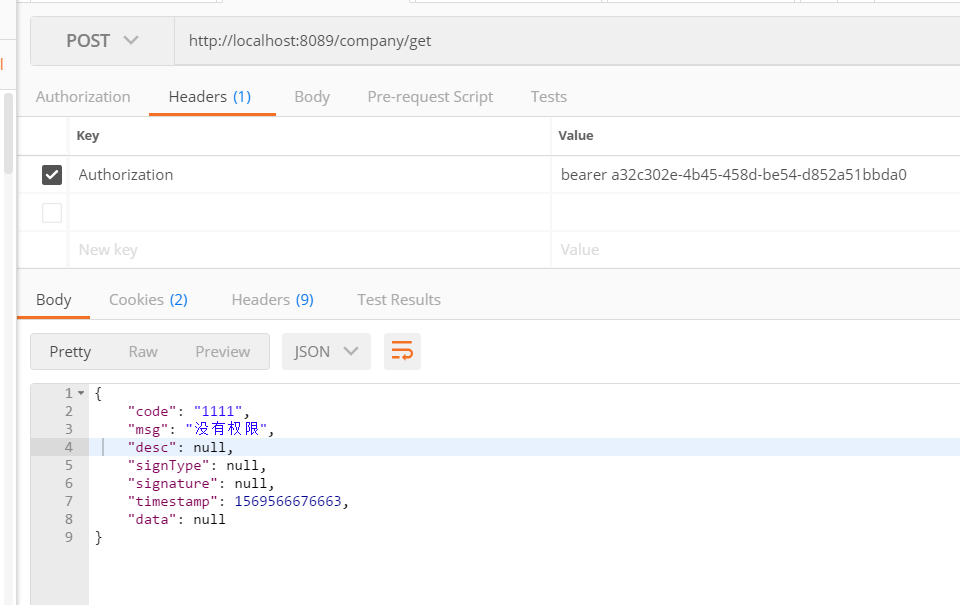
### 3.1通过token,访问资源服务

1.令牌不对或者过期

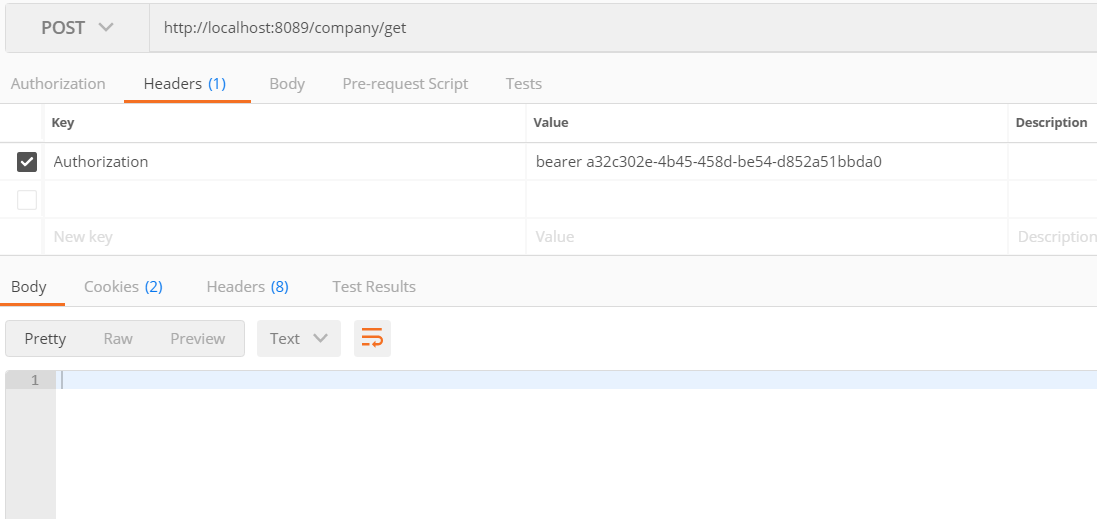


2.权限不足演示

在资源服务加了@PreAuthorize("hasAuthority('')")权限校验，如果令牌中不含有此权限，会抛出AccessDeniedException异常，该异常已经被统一处理

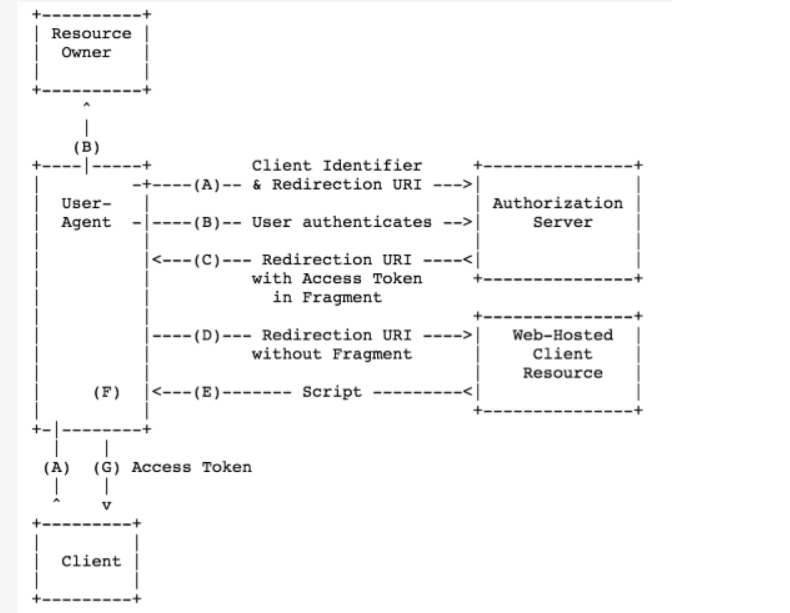


3.权限通过



## 4.1隐式授权模式（Implicit Grant）

### 4.1.1 隐式授权流程图



### 4.1.2 隐式授权模式说明

申请授权token，参数和申请授权码类似，client\_id，redirect\_uri回调地址，response\_type有变动，改为直接获取token，scope权限，state用于认证标记，传过去什么回调时传回来什么



Get请求，请求路径：

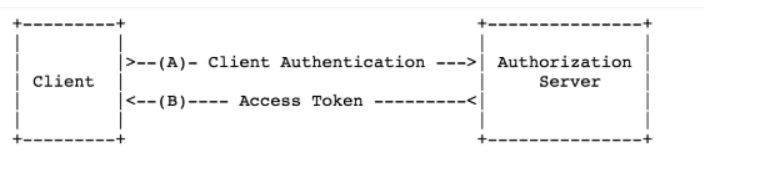
[http://localhost:8888/oauth/authorize?client\_id=app&redirect\_uri=http://localhost&response\_type=token&scope=app&state=abc](http://localhost:8888/oauth/authorize?client_id=app&redirect_uri=http:/localhost&response_type=token&scope=app&state=abc)

返回结果：access\_token会直接带回来，然后访问资源服务和授权码模式一样



## 5.1客户端凭证模式（Client Credentials Grant）

### 5.1.1客户端凭证流程图

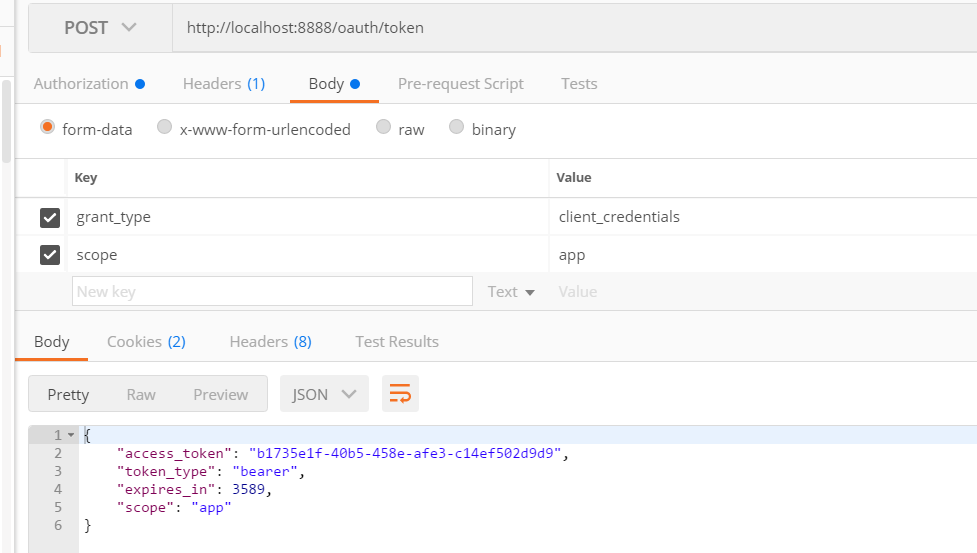


### 5.1.2客户端凭证模式说明

只需修改授权服务器，authorizedGrantTypes类型client\_credentials，客户端凭证模式也需要在header里添加认证账户密码

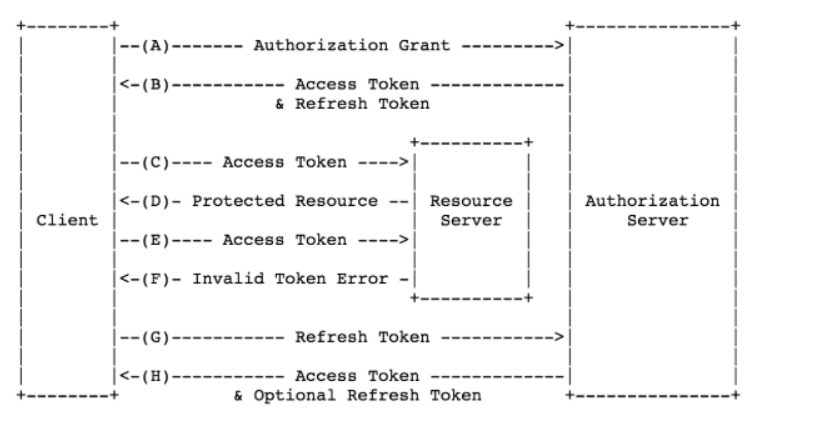
Post请求，请求地址：

<http://localhost:8888/oauth/token>



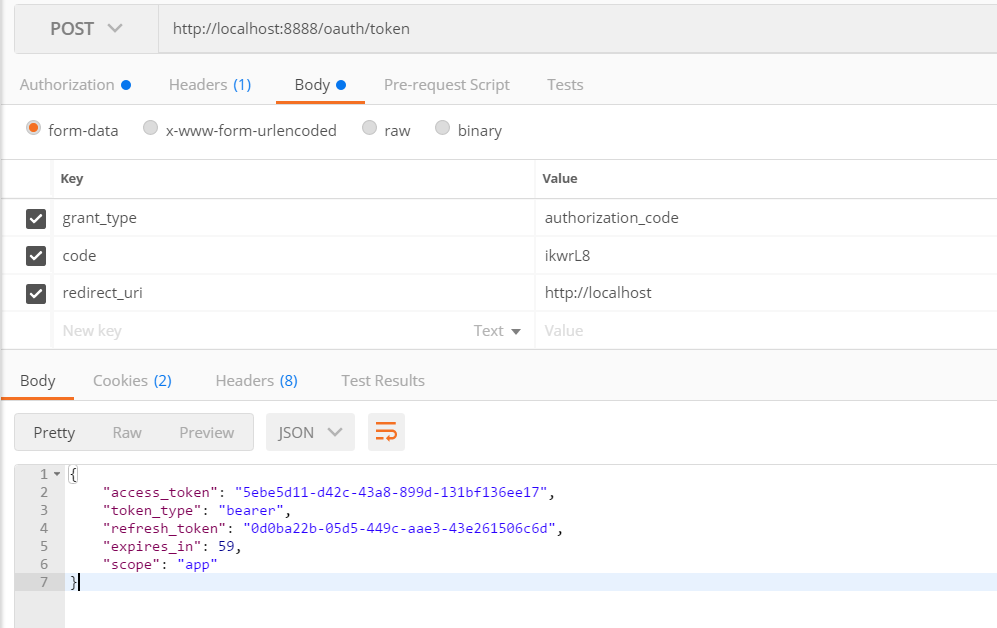
## 6.1刷新TOKEN

### 6.1.1流程图



### 6.1.2 刷新token说明

以授权码模式为例，步骤同授权码模式，取得授权码后，去取token时，返回



在token过期后，调用(注意刷新token时都要认证用户名和密码):

[http://localhost:8888/oauth/token?grant\_type=refresh\_token&refresh\_token=0d0ba22b-05d5-449c-aae3-43e261506c6d](http://localhost:8080/oauth/token?grant_type=refresh_token&refresh_token=ad3941d1-c6dd-4a2e-a9c8-eac6a9a59dd2)

