<http://andaily.com/spring-oauth-client/>

## Spring Security&Oauth2 Client is work!

操作说明:

1. spring-oauth-client 的实现没有使用开源项目 [spring-security-oauth2](https://github.com/spring-projects/spring-security-oauth/tree/master/spring-security-oauth2) 中提供的代码与配置, 如:<oauth:client id="oauth2ClientFilter" />
2. 按照Oauth2支持的grant\_type依次去实现. 共5类.
   * authorization\_code
   * password
   * client\_credentials
   * implicit
   * refresh\_token
3. 在开始使用之前, 请确保 [*spring-oauth-server*](http://git.oschina.net/shengzhao/spring-oauth-server) 项目已正确运行, 且 spring-oauth-client.properties (位于项目的 src/main/resources 目录) 配置正确

Δ 注意: 项目前端使用了 Angular-JS 来处理动态数据展现.

**菜单**

* [authorization\_code](http://andaily.com/spring-oauth-client/authorization_code)  
  授权码模式(即先登录获取code,再获取token) [最常用]
* [password](http://andaily.com/spring-oauth-client/password)   
  密码模式(将用户名,密码传过去,直接获取token) [适用于移动设备]
* [client\_credentials](http://andaily.com/spring-oauth-client/client_credentials)   
  客户端模式(无用户,用户向客户端注册,然后客户端以自己的名义向'服务端'获取资源)
* [implicit](http://andaily.com/spring-oauth-client/implicit)   
  简化模式(在redirect\_uri 的Hash传递token; Auth客户端运行在浏览器中,如JS,Flash)
* [refresh\_token](http://andaily.com/spring-oauth-client/refresh_token)   
  刷新access\_token

**注意**: 在测试时默认填写的数据有可能不正确, 建议先在 [spring-oauth-server](http://andaily.com/spring-oauth-server/) 添加 client\_details 后, 使用其client\_id, client\_secret来进行测试.

## authorization\_code 从 spring-oauth-server获取 'code'

grant\_type = 'authorization\_code' 模式是Oauth中最常用的, 一般是通过浏览器来完成. 整个流程分3步完成,依次为:

1. 从 spring-oauth-server获取 'code'   
   -- 该步骤将根据从 spring-oauth-server 中获取的client信息(如client\_id,client\_secret)将用户引导到server的登录页面.   
    在实际应用中, 表现为在登录页面中展示的 '通过第三方登录' 或 '通过其他账号登录'
2. 用 'code' 换取 'access\_token'   
   -- 在server端当用户登录成功并授权后将返回到spring-oauth-client的回调地址(即redirect\_uri),   
   当检查通过后(指检查返回是否有code与state值), 将根据server端获取access\_token的URL去换取access\_token值.   
    在实际应用中, 该步骤一般由client后端代码完成,前端不需要表现.
3. 用 'access\_token' 去访问 spring-oauth-server 的API   
   -- 在成功获取access\_token后,将根据 spring-oauth-server 中提供的API去获取 资源服务器 中的数据.   
   在 spring-oauth-server 中当前只有一个API可供调用(即http://andaily.com/spring-oauth-server/unity/user\_info), 用于获取当前登录用户的信息.   
    在实际应用中, 资源服务器都会提供许多API供调用

步骤1: 从 spring-oauth-server获取 'code'

窗体顶端

authorizationUri

http://andaily.com/spring-oauth-server/oauth/authorize  [测试连接](http://andaily.com/spring-oauth-server/oauth/authorize)

authorizationUri value from 'spring-oauth-client.properties'

[显示请求参数](javascript:void(0);)

response\_type



固定值 'code'

scope



client\_id



redirect\_uri



redirect\_uri 是在 'AuthorizationCodeController.java' 中定义的一个 URI, 用于检查server端返回的 'code'与'state',并发起对 access\_token 的调用

state



一个随机值, spring-oauth-server 将原样返回,用于检测是否为跨站请求(CSRF)等

最终发给 spring-oauth-server的 URL:

http://andaily.com/spring-oauth-server/oauth/authorize?response\_type=code&scope=read write&client\_id=unity-client&redirect\_uri=http://andaily.com/spring-oauth-client/authorization\_code\_callback&state=7a2a486b-8020-4344-a7f4-c62b3eb952d7

去登录 将重定向到 'spring-oauth-server' 的登录页面 **GET**

窗体底端

## password

grant\_type = 'password' 模式一般用在移动设备上(如IOS,Android), 适用于非浏览器的环境.   
在此提供两种方式来展示该模式.

方式1

Java代码, 详见项目中的PasswordOauthHandler.java文件.   
使用HttpClient来向服务器发送请求,处理数据, 获取access\_token并调用资源API **[推荐]**

方式2

在页面上点击链接 'Password grant\_type' 按钮, 将打开新窗口,展示服务器端响应的JSON数据.   
若是开发者关心请求的参数,可点击'显示请求参数' 展示请求的参数细节.

窗体顶端

accessTokenUri

http://andaily.com/spring-oauth-server/oauth/token  [测试连接](http://andaily.com/spring-oauth-server/oauth/token)

[显示请求参数](javascript:void(0);)

client\_id



客户端从 Oauth Server 申请的client\_id, 有的Oauth服务器中又叫 appKey

client\_secret



客户端从 Oauth Server 申请的client\_secret, 有的Oauth服务器中又叫 appSecret

grant\_type



固定值 'password'

scope



username



用户在 Oauth Server 中的账号名称

password



用户在 Oauth Server 中的账号密码

Password grant\_type **POST**

窗体底端

## client\_credentials

grant\_type = 'client\_credentials' 模式不需要用户去资源服务器登录并授权, 因为客户端(client)已经有了访问资源服务器的凭证(credentials).   
所以当用户访问时,由client直接向资源服务器获取access\_token并访问资源即可.

若客户端需要登录(或注册), 则用户仅需在客户端登录(或注册)即可,与资源服务器没有关系

 在实际应用中, client\_credentials一般都是由后台来完成的,前台没有任何表现, 常用于子应用中去访问主应用的资源或API.

在本操作中, 首先需要向 spring-oauth-server 数据库中添加client\_credentials的client信息(oauth\_client\_details表),SQL如下:

insert into oauth\_client\_details

(client\_id, resource\_ids, client\_secret, scope, authorized\_grant\_types,

web\_server\_redirect\_uri,authorities, access\_token\_validity,

refresh\_token\_validity, additional\_information, create\_time, archived, trusted)

values

('credentials-client','unity-resource', 'credentials-secret', 'read,write','client\_credentials',

null,'ROLE\_UNITY,ROLE\_USER',null,

null,null, now(), 0, 0);

(若已添加则忽略; 在实际应用中, 添加的数据是需要向服务端申请注册的)

第一步 获取access\_token

点击 '获取access\_token' 按钮, 将向spring-auth-server请求获取access\_token.   
若是开发者关心请求的参数,可点击'显示请求参数' 展示请求的参数细节.

窗体顶端

accessTokenUri

http://andaily.com/spring-oauth-server/oauth/token  [测试连接](http://andaily.com/spring-oauth-server/oauth/token)

[显示请求参数](javascript:void(0);)

client\_id



客户端从 Oauth Server 申请的client\_id, 有的Oauth服务器中又叫 appKey

client\_secret



客户端从 Oauth Server 申请的client\_secret, 有的Oauth服务器中又叫 appSecret

grant\_type



固定值 'client\_credentials'

scope



获取access\_token **POST**

窗体底端

第二步 访问资源服务器的API

请先获取access\_token

## implicit

grant\_type = 'implicit' 模式通过在redirect\_uri 的Hash传递token; Auth客户端运行在浏览器中,如JS,Flash(不通过client服务器).   
使用该模式时要求浏览器绝对可信，不然可能会将访问信息(client\_id,client\_secret等)泄露给恶意用户或应用程序. 一般使用在临时访问的场景.

在本操作中, 首先需要向 spring-oauth-server 数据库中添加implicit的client信息(oauth\_client\_details表),SQL如下:

insert into oauth\_client\_details

(client\_id, resource\_ids, client\_secret, scope, authorized\_grant\_types,

web\_server\_redirect\_uri,authorities, access\_token\_validity,

refresh\_token\_validity, additional\_information, create\_time, archived, trusted)

values

('implicit-client','unity-resource', 'implicit-secret', 'read','implicit',

'http://localhost:7777/spring-oauth-client/implicit','ROLE\_UNITY,ROLE\_USER',null,

null,null, now(), 0, 0);

注意检查SQL中的web\_server\_redirect\_uri字段的值,必须是正确能访问的.

(若已添加则忽略; 在实际应用中, 添加的数据是需要向服务端申请注册的)

第一步 去spring-oauth-server登录并授权

点击 '登录并授权' 按钮, 将跳转到spring-auth-server的登录页面.   
若是开发者关心请求的参数,可点击'显示请求参数' 展示请求的参数细节.

窗体顶端

userAuthorizationUri

http://andaily.com/spring-oauth-server/oauth/authorize  [测试连接](http://andaily.com/spring-oauth-server/oauth/authorize)

[显示请求参数](javascript:void(0);)

client\_id



客户端从 Oauth Server 申请的client\_id, 有的Oauth服务器中又叫 appKey

client\_secret



客户端从 Oauth Server 申请的client\_secret, 有的Oauth服务器中又叫 appSecret

response\_type



固定值 'token'

scope



必须是client\_details中支持的scope; 当前的只支持read

redirect\_uri



必须与client\_details中的web\_server\_redirect\_uri一致

登录并授权 **GET**

窗体底端

第二步 访问资源服务器的API

请先 登录并授权

© 201

## refresh\_token

grant\_type = 'refresh\_token' 模式用在当获取的access\_token未过期之前向服务端换取新的access\_token.   
在获取access\_token时返回的数据如下

{"access\_token":"3420d0e0-ed77-45e1-8370-2b55af0a62e8","token\_type":"bearer","refresh\_token":"b36f4978-a172-4aa8-af89-60f58abe3ba1","expires\_in":43199,"scope":"read write"}

数据中的expires\_in即access\_token的有效时间(单位:秒), 默认的有效时间为43199(约12小时), 在服务端可配置默认的有效时间.

若在一个时间段内(即在43199秒之内)多次去获取access\_token, 将返回相同的access\_token值, 但expires\_in的值将会减少.   
(比如: 相隔10秒去获取access\_token, 第一次返回的expires\_in=43199,第二次返回的expires\_in=43189)

数据中refresh\_token的值将用于换取新的access\_token.   
在此将首先通过 grant\_type='password' 去获取access\_token, 然后调用 grant\_type='refresh\_token' 去换取新的access\_token.   
换取成功后将得到新的access\_token,并会看到expires\_in值又还原成43199.   
**注意** refresh\_token 成功后旧的access\_token将不能再使用.

 在实际应用中, refresh\_token一般都是由后台来完成的,前台没有任何表现.

1. 通过 grant\_type='password' 去获取access\_token

点击 '获取access\_token' 按钮, 从spring-oauth-server取到access\_token数据.   
若是开发者关心请求的参数,可点击'显示请求参数' 展示请求的参数细节.

窗体顶端

accessTokenUri

http://andaily.com/spring-oauth-server/oauth/token  [测试连接](http://andaily.com/spring-oauth-server/oauth/token)

[显示请求参数](javascript:void(0);)

client\_id



客户端从 Oauth Server 申请的client\_id, 有的Oauth服务器中又叫 appKey

client\_secret



客户端从 Oauth Server 申请的client\_secret, 有的Oauth服务器中又叫 appSecret

grant\_type



固定值 'password'

scope



username



用户在 Oauth Server 中的账号名称

password



用户在 Oauth Server 中的账号密码

获取access\_token

窗体底端

2. 调用 grant\_type='refresh\_token' 去换取新的access\_token

请先获取access\_token