**http://andaily.com/blog/?cat=19**

**Category Archives: Oauth**

[**Spring Boot中Spring Security OAuth2的变化(相比XML配置)**](http://andaily.com/blog/?p=1000)

By [andaily](http://andaily.com/blog/?author=2) in [Oauth](http://andaily.com/blog/?cat=19), [spring-security](http://andaily.com/blog/?cat=54), [原创](http://andaily.com/blog/?cat=77), [默认](http://andaily.com/blog/?cat=1)[2018-09-15](http://andaily.com/blog/?p=1000) [0 Comment](http://andaily.com/blog/?p=1000#respond) 399 views

在升级使用基于 Spring Boot 的 Spring Security OAuth2 后，相比于XML配置，除了所谓的“零配置”区别外，

在OAuth2的使用与流程中有以下几点变化

1. resource-id 只能配置一个（或不要）；在XML配置时，可设置多个resource-id（[详细](https://gitee.com/shengzhao/spring-oauth-server/blob/master/src/main/resources/spring/security.xml)请点此）;但在Spring Boot中，

一个程序中只能配置唯一的一个resource-id （[详细](https://gitee.com/shengzhao/spring-oauth-server/blob/config/src/main/java/com/monkeyk/sos/config/OAuth2ServerConfiguration.java)请点此）—— *个人觉得是因为一个微服务就是一资源(resource)吧。*

2.ClientDetails中的 client\_secret 值加密存储；使用的加密方式与Spring Security的密码加密方式一致；这是安全上

的一大提升，点赞。

[](https://gitee.com/shengzhao/spring-oauth-server)

<https://gitee.com/shengzhao/spring-oauth-server>

[**spring-oauth-server 2.0.0 发布,使用Spring-Boot2.0实现**](http://andaily.com/blog/?p=937)

By [andaily](http://andaily.com/blog/?author=2) in [access\_token](http://andaily.com/blog/?cat=118), [MyOIDC](http://andaily.com/blog/?cat=146), [Oauth](http://andaily.com/blog/?cat=19), [spring-mvc](http://andaily.com/blog/?cat=55), [spring-security](http://andaily.com/blog/?cat=54)[2018-04-21](http://andaily.com/blog/?p=937) [0 Comment](http://andaily.com/blog/?p=937#respond) 2,093 views

经过一年多的时间后,我们发布了 spring-oauth-server 2.0.0版本.

使用Spring-Boot 2.0版本重构, 进入模块化,零配置时代. 更符合技术发展与实际需要.

相比之前的版本(1.0及之前版本), 更新如下:

1. 全新的 Spring-Boot风格

2.ClientDetails中的 client\_secret 字段加密保存

3.密码加密方式由MD5变成 BCrypt

4.增加CSRF机制支持

5.resourceId变为可选(但建议使用固定的resourceId)

<https://gitee.com/shengzhao/spring-oauth-server/tree/2.0.0/>

[**[推荐]到2017年6月看到的最全面的OIDC介绍(中文)—-值得一读**](http://andaily.com/blog/?p=813)

By [andaily](http://andaily.com/blog/?author=2) in [MyOIDC](http://andaily.com/blog/?cat=146), [Oauth](http://andaily.com/blog/?cat=19), [OIDC](http://andaily.com/blog/?cat=111), [OpenId-Connect](http://andaily.com/blog/?cat=136)[2017-06-15](http://andaily.com/blog/?p=813) [0 Comment](http://andaily.com/blog/?p=813#respond) 1,569 views

感谢这篇文章, 对于帮助学习与理解 OIDC 协议与使用的 朋友来说, 最全面的一文章.

不多言, 详细请访问

<http://www.cnblogs.com/linianhui/p/openid-connect-core.html>

拥抱OIDC

[**4月,**](http://andaily.com/blog/?p=732)

By [andaily](http://andaily.com/blog/?author=2) in [memory](http://andaily.com/blog/?cat=5), [Oauth](http://andaily.com/blog/?cat=19), [think](http://andaily.com/blog/?cat=18), [原创](http://andaily.com/blog/?cat=77), [城市生活](http://andaily.com/blog/?cat=64), [杂想](http://andaily.com/blog/?cat=74), [生活](http://andaily.com/blog/?cat=20)[2017-04-17](http://andaily.com/blog/?p=732) [0 Comment](http://andaily.com/blog/?p=732#respond) 911 views

好几天时间想起来,我4月没有记录点什么,没发点什么文章,不管是技术的,开源的,还是生活的,所思所想的.

后来我还是决定写点记录点,连标题都不太清晰,名叫4月.

感觉对开源的热情在下降,是计划着继续更新完善OAUTH2相关的知识介绍与内容,还是没有去行动记录于纸上或文章里.

在码不完的代码中,继续着码代码—这类似生活.

我还记得租房一个人时,有了房子也继续是一个人在很多的时间中看着书,继续着前进.

技术,不能停止;脑子,不能静止,感觉于此.

每日来共享单车中徘徊与周旋, 感觉是既锻炼又费力.

感觉脑子进水去拍了CT,等等结果,等待报告出来…

脑子本来就有水,无需进水.

感觉看书中才能有所寄托与进步,就在床头,沙发上,背包中,到处着书的影子. 依旧感觉到上厕所时是最好的看书时光,简单快捷且两不误,只要起身时脚不发麻即可.

以往的总是一本书一本书地看完,而现在是转变为多本书多线程地并行着看.

多年以来,唯有书才感觉是系统学习的来源与未来的可支撑物.

在感觉快有4月18日这个有些特殊的日子来临前写下点思绪有些乱的现状.

等待着又一波的网络营销,电子商务平台的盛宴(感觉会是).

关于编程,更多的是叫编程思想, 思想才是目的,而编程只是过程; 编程是量变,思想是量变的升华,是质变;

在编程中思考,思索,思念,思乡,最后变成思想.

我不用更多的思想,行动才是王道,行动才能解决问题.

请别在思想.

不知啥的,一说到4月,就想到秋生.

[**spring-oauth-server 如何扩展使用其他数据存储方式**](http://andaily.com/blog/?p=617)

By [andaily](http://andaily.com/blog/?author=2) in [access\_token](http://andaily.com/blog/?cat=118), [Oauth](http://andaily.com/blog/?cat=19), [spring-security](http://andaily.com/blog/?cat=54), [原创](http://andaily.com/blog/?cat=77)[2016-11-22](http://andaily.com/blog/?p=617) [0 Comment](http://andaily.com/blog/?p=617#respond) 1,886 views

[spring-oauth-server](http://git.oschina.net/shengzhao/spring-oauth-server)项目默认提供了MYSQL数据库进行数据存储, 另一个branch: [mongodb](http://git.oschina.net/shengzhao/spring-oauth-server/tree/mongodb) 使用了MongoDB进行数据存储.

但如果你想在使用时用其他的数据存储方式(如ORACLE,SQL SERVER, REDIS),该如何办呢? 以下给出一些思路, 希望能帮上你.

**理论准备**:

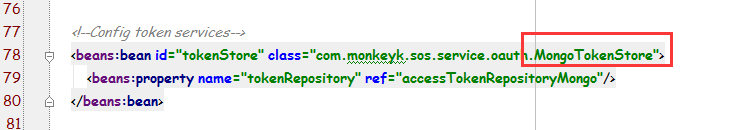
spring-oauth-server项目实际是一个基于Spring框架的, 而Spring的核心思想是什么——面向接口编程, 而面向接口编程带来的好处就是扩展性很强(不是一般的强), 以此为出发点来思考.

同时想想默认的spring-oauth-server使用MYSQL时有哪些数据进行了存储, 就寻找这些存储相关的类进行扩展.

**具体实现**:

(以mongodb 分支为例)

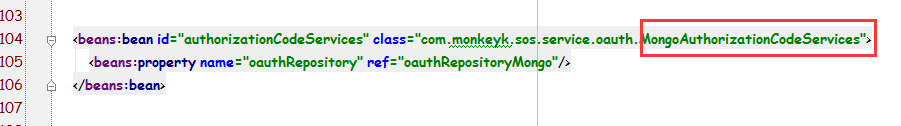
1.扩展接口TokenStore.java, 写自己的TokenStore实现类, 参见MongoTokenStore.java类. 同时在security.xml中配置tokenStore时使用自己扩展的类,如下图:

[](http://andaily.com/blog/wp-content/uploads/2016/11/tokenStore.png)

2.扩展接口ClientDetailsService.java,写自己的ClientDetailsService实现类, 参见MongoClientDetailsService.java类,同时在security.mxl中配置clientDetailsService时使用自己扩展的类,如下图:

[](http://andaily.com/blog/wp-content/uploads/2016/11/clientDetailsService.png)

3.扩展接口AuthorizationCodeServices.java,写自己的AuthorizationCodeServices实现类, 参见MongoAuthorizationCodeServices.java类,同时在security.xml中配置authorizationCodeServices时使用自己扩展的类,如下图:

[](http://andaily.com/blog/wp-content/uploads/2016/11/authCode.png)

OK, 在实现上面3个步骤后, 扩展实现数据存储方式就差不多了(当然,写一些辅助类是不错的,如MongoTokenStore实现时注入使用了AccessTokenRepository; 这能带来代码结构清晰,维护方便 ,扩展性强等好处), 以上的3个接口都有些抽象的或默认的实现类,如TokenStore默认时使用的实现类为InMemoryTokenStore.java, 借助强大的IDE工具去查看各个接口已有的实现类, 仿照或参考去扩展实现自己需要的实现类并配置即可, 比如可以继承一些已有的实现类进行覆盖扩展也是不错的主意.

编程之艺在于融会贯通,周而复始; 在于一种思想思考—面向期望编程!!!

[**spring-oauth-server 0.6 发布，OAuth2 与 Spring Security 安全应用整合**](http://andaily.com/blog/?p=602)

By [andaily](http://andaily.com/blog/?author=2) in [access\_token](http://andaily.com/blog/?cat=118), [Cache](http://andaily.com/blog/?cat=94), [EhCache](http://andaily.com/blog/?cat=93), [Oauth](http://andaily.com/blog/?cat=19), [OIDC](http://andaily.com/blog/?cat=111), [OpenId-Connect](http://andaily.com/blog/?cat=136), [spring-security](http://andaily.com/blog/?cat=54)[2016-10-13](http://andaily.com/blog/?p=602) [0 Comment](http://andaily.com/blog/?p=602#respond) 2,476 views

spring-oauth-server 0.6版本发布，该版本主要增加有缓存的支持, 具体的功能更新如下：

* (150) – 修改OAUTH错误时返回JSON数据
* (151) – 数据添加Ehcache缓存支持
* (158) – 对配置,代码必要的地方添加注释,方便理解
* 添加OIDC协议文档

OIDC是基于OAuth2的下一代身份认证授权协议,在0.6版本中添加了对OIDC1.0版本协议文档,同时在进行中文的翻译,具体请访问 [https://andaily.com/blog/?p=440](https://andaily.com/blog/?cat=111)

spring-oauth-server 是一个整合OAuth2与Spring Security的安全应用项目，是从<https://github.com/spring-projects/spring-security-oauth/tree/master/spring-security-oauth2>扩展的符合实际项目需要的开源项目，可扩展为移动SSO解决方案.

0.6版本访问地址: <http://git.oschina.net/shengzhao/spring-oauth-server/tree/0.6/>

[**OIDC(OpenID-Connect) 1.0 协议中文版**](http://andaily.com/blog/?p=582)

By [andaily](http://andaily.com/blog/?author=2) in [Oauth](http://andaily.com/blog/?cat=19), [OIDC](http://andaily.com/blog/?cat=111), [OpenId-Connect](http://andaily.com/blog/?cat=136), [翻译](http://andaily.com/blog/?cat=135)[2016-08-28](http://andaily.com/blog/?p=582) [0 Comment](http://andaily.com/blog/?p=582#respond) 2,649 views

OIDC(OpenID-Connect) 1.0协议的最终版本早已确认, 用自己的语言在进行翻译中(core部分),

OpenID Connect Core 1.0 incorporating errata set 1

截止2016-08-28的翻译进度(章节3.1.2.3)

中文版: <https://monkeyk.com/oidc/Final_OpenID-Connect-Core-1.0-incorporating-errata-set-1_CN.html>

英文原版: <http://openid.net/specs/openid-connect-core-1_0.html>

JSON Web Signature(JWS): <https://tools.ietf.org/html/rfc7515>

对应的开源项目地址: [http://git.oschina.net/shengzhao/spring-oauth-server/tree/master/others/oidc?dir=1&filepath=others%2Foidc&oid=ec2b907c58efcb56a8c774e8f7868ab80c6cb69a&sha=25a0e1f2357768eadf3f0e3b8035959e3bbd3d66http://git.oschina.net/shengzhao/spring-oauth-server/tree/master/others/oidc?dir=1&filepath=others%2Foidc&oid=ec2b907c58efcb56a8c774e8f7868ab80c6cb69a&sha=25a0e1f2357768eadf3f0e3b8035959e3bbd3d66](http://git.oschina.net/shengzhao/spring-oauth-server/tree/master/others/oidc?dir=1&filepath=others%2Foidc&oid=ec2b907c58efcb56a8c774e8f7868ab80c6cb69a&sha=25a0e1f2357768eadf3f0e3b8035959e3bbd3d66)

持续更新中…

也可访问: <http://www.cnblogs.com/linianhui/p/openid-connect-core.html>

与OIDC相关介绍请访问: <https://andaily.com/blog/?p=440>

[**oauth2-shiro 0.2版本发布**](http://andaily.com/blog/?p=578)

By [andaily](http://andaily.com/blog/?author=2) in [access\_token](http://andaily.com/blog/?cat=118), [Oauth](http://andaily.com/blog/?cat=19), [oauth2-shiro](http://andaily.com/blog/?cat=86), [Shiro](http://andaily.com/blog/?cat=80)[2016-08-17](http://andaily.com/blog/?p=578) [0 Comment](http://andaily.com/blog/?p=578#respond) 1,945 views

oauth2-shiro 0.2版本正式发布, 在7月完成开发工作,但由于当时发布 <http://git.oschina.net/mkk/oauth2-shiro-redis> 项目竟然给忘记了. 该版本主要更新如下

1.更新首页UI, 参照[spring-oauth-server](http://git.oschina.net/shengzhao/spring-oauth-server)

2.Add client details overview

3.Add client details testing

4.user add/edit, overview

5.添加API使用说明, 举例各个场景

6.发布到测试服务器

7.resources模块更新UI说明

从0.2版本开始可在线测试,测试地址分别为:

authz: <https://andaily.com/authz/>

resources:<https://andaily.com/rs/>

0.2版本访问地址: <http://git.oschina.net/mkk/oauth2-shiro/tree/0.2/>

oauth2-shiro 整合OLTU与SHIRO, 提供一个轻量的OAUTH2应用框架.

[**正确处理spring-oauth-server中在验证失败或错误时的方式**](http://andaily.com/blog/?p=528)

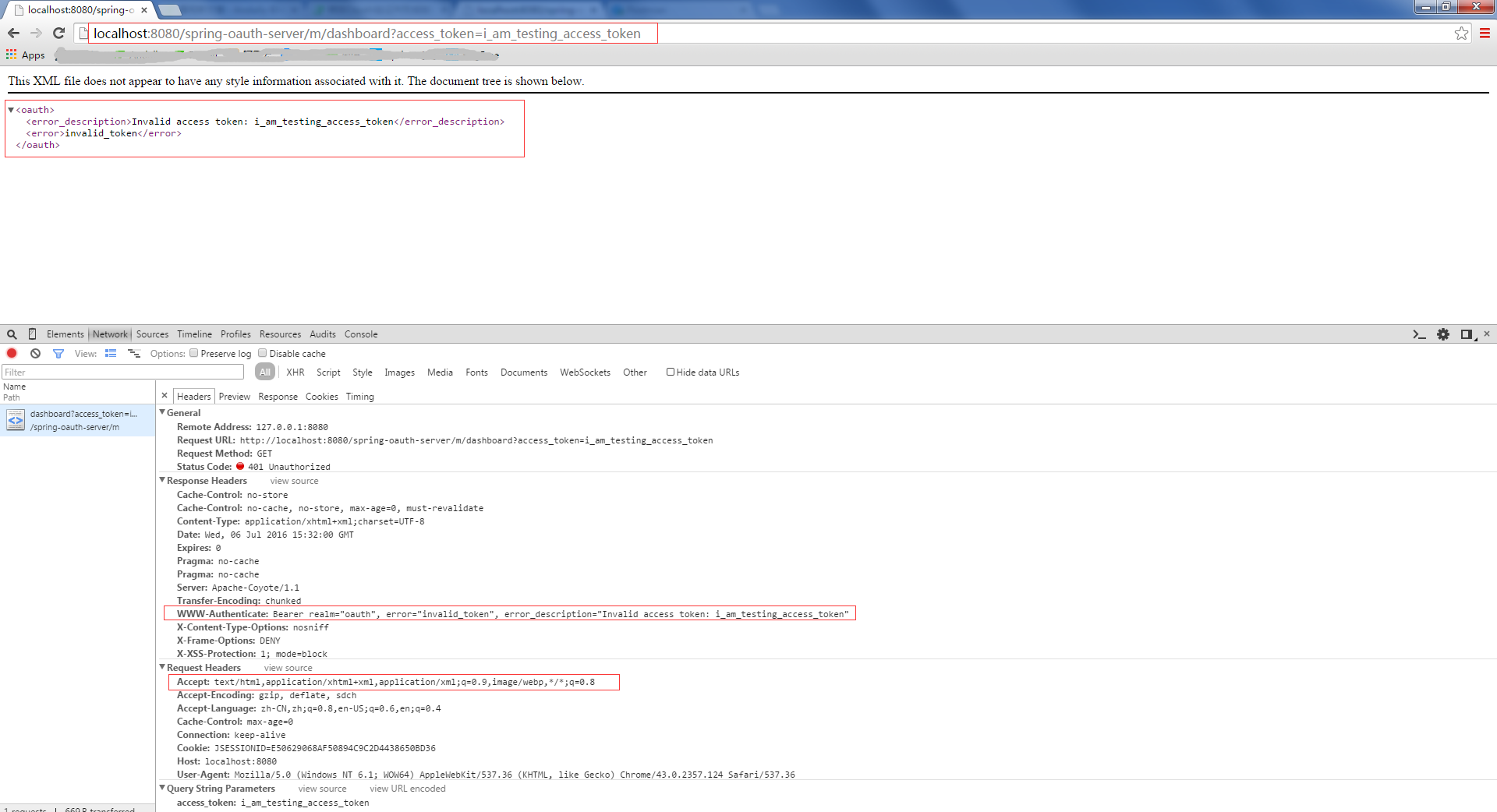
By [andaily](http://andaily.com/blog/?author=2) in [Oauth](http://andaily.com/blog/?cat=19), [spring-security](http://andaily.com/blog/?cat=54)[2016-07-07](http://andaily.com/blog/?p=528) [0 Comment](http://andaily.com/blog/?p=528#respond) 2,948 views

在文章 <https://andaily.com/blog/?p=97> 中提到了OAUTH验证失败或错误时的处理办法, 但该方式是不完美的.

通过剖析代码发现,为什么在浏览器中会返回XML格式的数据, 而在使用类似HttpClient的请求时返回的数据格式为JSON,

其实所有的根源都在于请求header中的Accept的值所决定的,如果Accept不支持application/json, 那服务端肯定不会返回JSON格式的数据,

通过浏览器的开发者工具很容易追踪,如下图:

[](http://andaily.com/blog/wp-content/uploads/2016/07/web_response_xml.png)

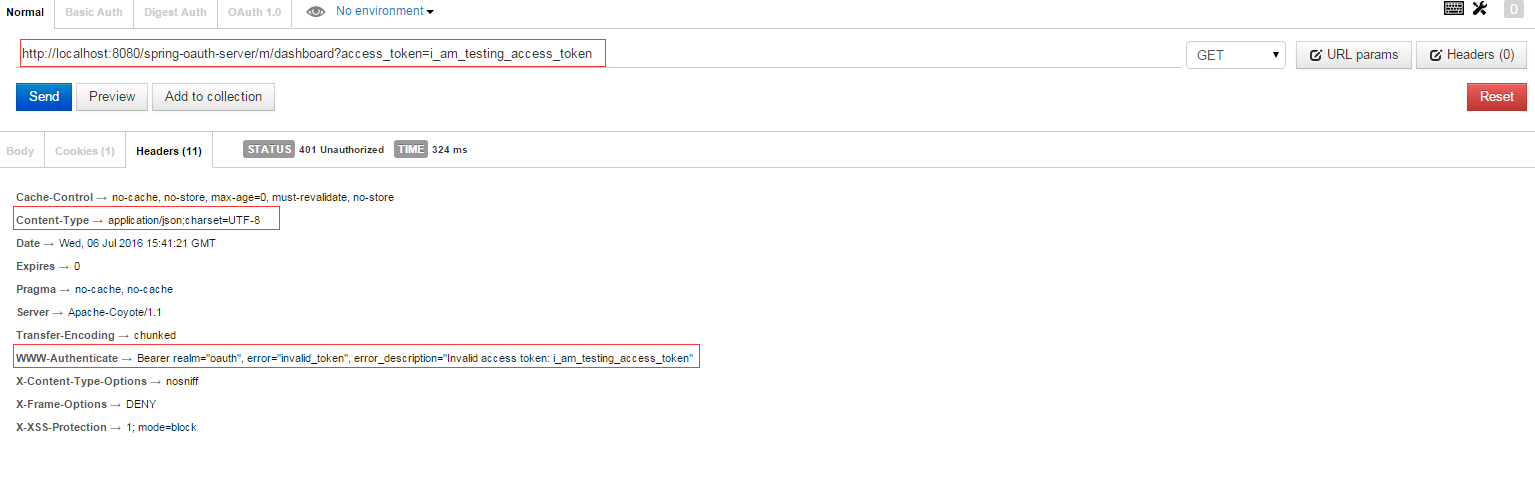
从图中可以看出我们在请求时header中Accept的值为:text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,\*/\*;q=0.8; 服务器响应401

没有json格式的 media type, 但有xml的, 所以返回的数据格式不会是JSON的在浏览器请求时, 同时请注意 response header中的WWW-Authenticate.

相同的请求,如果我们换一种方式, 如Postman, 其结果返回的却是JSON格式的.如下图:

[](http://andaily.com/blog/wp-content/uploads/2016/07/postman_response_json1.png)

这也是因为请求时的header中Accept导致的, 看下图:

[](http://andaily.com/blog/wp-content/uploads/2016/07/postman_response_json2.png)

Postman在请求时的Header中设置了Content-Type: application/json; 同时也注意WWW-Authenticate.

综上分析, 你应该知道如何在请求时设定响应时的数据格式了吧. 方法如下:

1. 如果能指定Header中的Accept或Content-Type, 则指定为具体的数据类型或 media type  (一般用在非浏览器环境)

2.不管返回的数据体(response body)是XML还是JSON, 通过响应的header来处理, 如果响应码不是200, 则获取Header中WWW-Authenticate的值.

通过解析WWW-Authenticate的值来获取错误信息 (适用几乎所有环境)

<http://git.oschina.net/shengzhao/spring-oauth-server>

[**grant\_type=authorization\_code 中 redirect\_uri的代码实现参考**](http://andaily.com/blog/?p=504)

By [andaily](http://andaily.com/blog/?author=2) in [access\_token](http://andaily.com/blog/?cat=118), [Oauth](http://andaily.com/blog/?cat=19), [oauth2-shiro](http://andaily.com/blog/?cat=86), [spring-security](http://andaily.com/blog/?cat=54), [SSO](http://andaily.com/blog/?cat=108)[2016-06-11](http://andaily.com/blog/?p=504) [0 Comment](http://andaily.com/blog/?p=504#respond) 2,077 views

OAuth2中grant\_type=authorization\_code时, 需要填写redirect\_uri, 而该uri将如何处理与实现呢, 在此提供具体的讲解与实现参考

redirect\_uri在注册ClientDetails时需要提供(当grant\_type包含authorization\_code时), 且在发起OAUTH2流程时需要在参数中传递redirec\_uri, 这两处的值必须一致.

**在业务中, redirect\_uri代码实现主要用于接收返回的code值, 校验state是否合法, 以及通过code换取accessToken操作.**

在 [spring-oauth-client](http://git.oschina.net/mkk/spring-oauth-client) 项目的 AuthorizationCodeController.java 类中, authorization\_code\_callback 方法实现了以上的逻辑, 在具体实现是可参考该方法的实现, 代码如下:

@RequestMapping(value = "authorization\_code\_callback")

public String authorizationCodeCallback(AuthCallbackDto callbackDto, HttpServletRequest request, Model model) throws Exception {

if (callbackDto.error()) {

//Server response error

model.addAttribute("message", callbackDto.getError\_description());

model.addAttribute("error", callbackDto.getError());

return "redirect:oauth\_error";

} else if (correctState(callbackDto, request)) {

//Go to retrieve access\_token form

AuthAccessTokenDto accessTokenDto = oauthService.createAuthAccessTokenDto(callbackDto);

model.addAttribute("accessTokenDto", accessTokenDto);

model.addAttribute("host", host);

return "code\_access\_token";

} else {

//illegal state

model.addAttribute("message", "Illegal \"state\": " + callbackDto.getState());

model.addAttribute("error", "Invalid state");

return "redirect:oauth\_error";

}

}

这里实现首先检查服务端是否返回error信息,

然后判断state是否正确, 若正确则显示一个页面, 并在code\_access\_token方法中(同一个类中)使用 httpclient 发起请求(grant\_type=authorization\_code)通过 code换取access\_token.

其它情况则显示error信息.

<http://git.oschina.net/mkk/spring-oauth-client>