<https://www.cnblogs.com/giserliu/p/4372455.html>

# [whgiser](https://www.cnblogs.com/giserliu/)

随笔 - 36  文章 - 0  评论 - 25

# [OAuth2的学习小结](https://www.cnblogs.com/giserliu/p/4372455.html)

### [Spring](http://www.iteye.com/blogs/tag/Spring) [Security](http://www.iteye.com/blogs/tag/Security) [json](http://www.iteye.com/blogs/tag/json) [Access](http://www.iteye.com/blogs/tag/Access) [maven](http://www.iteye.com/blogs/tag/maven)

# OAUTH2核心参数说明

**grant\_type参数说明表格：**

|  |  |
| --- | --- |
| **grant\_type** | **说明** |
| authorization\_code | 标准的Server授权模式 |
| password | 基于用户密码的授权模式 |
| client\_credentials | 基于APP密钥的授权模式 |
| refresh\_token | 刷新accessToken |

**response\_type参数说明表格：**

|  |  |
| --- | --- |
| **response\_type** | **说明** |
| code | 标准的Server授权模式响应模式 |
| token | 脚本的授权响应模式，直接返回token，需要对回调进行校验 |

# OAUTH2各种请求流程

## Authorization Code（标准请求流程，必须实现）

标准的的Server授权模式，与目前开放平台的Session机制很像。

**APP首先发送获取code请求**

GET /authorize?response\_type=code&client\_id=s6BhdRkqt3&

         redirect\_uri=https%3A%2F%2Fclient%2Eexample%2Ecom%2Fcb HTTP/1.1

     Host: server.example.com

**容器返回code**

HTTP/1.1 302 Found

     Location: <https://client.example.com/cb?code=i1WsRn1uB1>

**APP根据code发送获取token请求**

POST /token HTTP/1.1

     Host: server.example.com

     Content-Type: application/x-www-form-urlencoded

     grant\_type=authorization\_code&client\_id=s6BhdRkqt3&

     client\_secret=gX1fBat3bV&code=i1WsRn1uB1&

     redirect\_uri=https%3A%2F%2Fclient%2Eexample%2Ecom%2Fcb

**容器直接返回token**

     HTTP/1.1 200 OK

     Content-Type: application/json

     Cache-Control: no-store

     {

       "access\_token":"SlAV32hkKG",

       "token\_type":"example",

       "expires\_in":3600,

       "refresh\_token":"8xLOxBtZp8",

       "example\_parameter":"example-value"

     }

## Implicit Grant（直接发放模式）

适用于运行于浏览器中的脚本应用，需要校验callback地址，而且只返回该应用注册的回调地址

**APP直接请求token**

GET /authorize?response\_type=token&client\_id=s6BhdRkqt3&

         redirect\_uri=https%3A%2F%2Fclient%2Eexample%2Ecom%2Fcb HTTP/1.1

     Host: server.example.com

**容器通过重定向返回token**

HTTP/1.1 302 Found

     Location: http://example.com/rd#access\_token=FJQbwq9&

               token\_type=example&expires\_in=3600

## Resource Owner Password Credentials （基于用户名与密码模式）

称之为用户名密码模式，需要提供终端用户的用户名和密码，适用于比如操作系统或者高权限的应用。

**APP直接带上用户名和密码请求**

POST /token HTTP/1.1

     Host: server.example.com

     Content-Type: application/x-www-form-urlencoded

     grant\_type=password&client\_id=s6BhdRkqt3&

     client\_secret=47HDu8s&username=johndoe&password=A3ddj3w

**容器直接返回token**

     HTTP/1.1 200 OK

     Content-Type: application/json

     Cache-Control: no-store

     {

       "access\_token":"SlAV32hkKG",

       "token\_type":"example",

       "expires\_in":3600,

       "refresh\_token":"8xLOxBtZp8",

       "example\_parameter":"example-value"

     }

## Client Credentials

基于APP的密钥直接进行授权，APP的权限非常大，慎用。这个模式可以考虑用于目前我们不需要弹出授权的特殊应用，如淘江湖，前端插件等。

**APP直接根据客户端的密码来请求**

POST /token HTTP/1.1

     Host: server.example.com

     Content-Type: application/x-www-form-urlencoded

     grant\_type=client\_credentials&client\_id=s6BhdRkqt3&

     client\_secret=47HDu8s

**容器直接返回token**

HTTP/1.1 200 OK

     Content-Type: application/json

     Cache-Control: no-store

     {

       "access\_token":"SlAV32hkKG",

       "token\_type":"example",

       "expires\_in":3600,

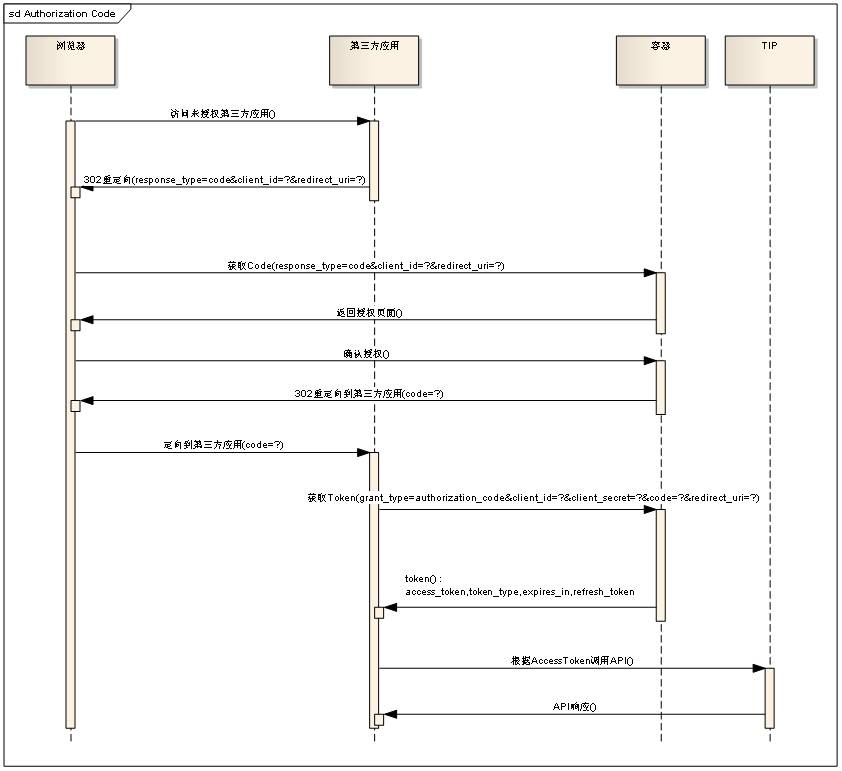
       "refresh\_token":"8xLOxBtZp8",

       "example\_parameter":"example-value"

     }

# 优先考虑实现的流程

Authorization Code为我们需要优先支持的流程，很多开源的OAUTH实现都是优先实现了该授权流程。ETAO的B2C网站会用这个流程与开放平台交互。



# 开源实现

目前OAUTH 2有比较多的开源实现，其中比较好的开源实现是OAuth for Spring Security，大家可以参考<http://static.springsource.org/spring-security/oauth/tutorial.html>这个网址去具体了解。有兴趣的同学可以去这个网址去下载其源代码看看<http://maven.springframework.org/milestone/org/springframework/security/oauth/spring-security-oauth/1.0.0.M2/spring-security-oauth-1.0.0.M2-sources.jar> ，容器主要关注下面几个类：org.springframework.security.oauth2.provider.OAuth2AuthorizationFilter

org.springframework.security.oauth2.provider. DefaultOAuth2GrantManager

org.springframework.security.oauth2.provider.verification.VerificationCodeFilter

第一个和第二个类为参数校验和参数解析，第三个类为响应生成的类。

TIP主要关注下面的类：

org.springframework.security.oauth2.provider.OAuth2ProtectedResourceFilter

这个类主要实现了对AccessToken的校验

详细的例子请访问:<http://git.oschina.net/shengzhao/spring-oauth-server>