OAuth2.0有五种授权模式。

（1）授权码模式（Authorization Code）   
（2）授权码简化模式（Implicit）   
（3）Pwd模式（Resource Owner Password Credentials）   
（4）Client模式（Client Credentials）   
（5）扩展模式（Extension）

***注：文中的client，可理解为浏览器或APP。***

但不论哪种模式，都是为了从认证服务器获取Access Token，用来访问资源服务器。   
而申请Access Token，需要提交相应信息。例如，client\_ID(我是谁)，response\_type或grant\_typt(申请哪种模式)，scope(申请哪些权限，由授权服务器定义)，redirect\_uri(申请结果跳转至哪儿)等。当然不同的模式，提交信息内容也不同。

我们先从简单的模式开始。因为简单模式，申请流程短，安全级别较低，个人感觉可用场景不多。

**（一）Client模式**

流程图



申请该模式时，需要在HTTP request entity-body中提交以下信息。

grant\_type：

REQUIRED. Value MUST be set to "client\_credentials".

scope：

OPTIONAL. The scope of the access request

当然，可以根据授权服务器的实现，提交其它必要信息。   
若申请成功，服务器将返回access token和token有效时间。

附规范中的例子。   
Request：

POST /token HTTP/1.1

Host: server.example.com

Authorization: Basic czZCaGRSa3F0MzpnWDFmQmF0M2JW

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

grant\_type=client\_credentials

Response：

HTTP/1.1 200 OK

Content-Type: application/json;charset=UTF-8

Cache-Control: no-store

Pragma: no-cache

{

"access\_token":"2YotnFZFEjr1zCsicMWpAA",

"token\_type":"example",

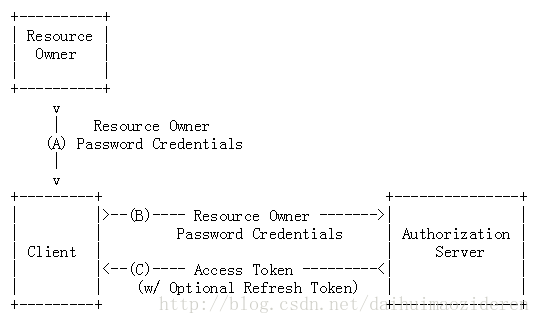
"expires\_in":3600,

"example\_parameter":"example\_value"

}

**注：expires\_in为token有效时长，单位为秒。**

## （二）Pwd模式



该模式下，需要用户将自身的account ID和password交由client，client将使用它们来申请access token，整个过程会将用户信息暴露。因此，除非client十分可靠（例如硬件设备，系统APP），否则，不建议使用该模式。

申请该模式时，需要在HTTP request entity-body中提交以下信息。

grant\_type：

REQUIRED. Value MUST be set to "password".

username：

REQUIRED. The resource owner username.

password：

REQUIRED. The resource owner password.

scope：

OPTIONAL. The scope of the access request

申请成功后，授权服务器将返回access token和token有效时间，以及可选的refresh token，用于在access token过期时进行token更新

附规范中的例子。   
Request：

POST /token HTTP/1.1

Host: server.example.com

Authorization: Basic czZCaGRSa3F0MzpnWDFmQmF0M2JW

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

grant\_type=password&username=johndoe&password=A3ddj3w

Response：

HTTP/1.1 200 OK

Content-Type: application/json;charset=UTF-8

Cache-Control: no-store

Pragma: no-cache

{

"access\_token":"2YotnFZFEjr1zCsicMWpAA",

"token\_type":"example",

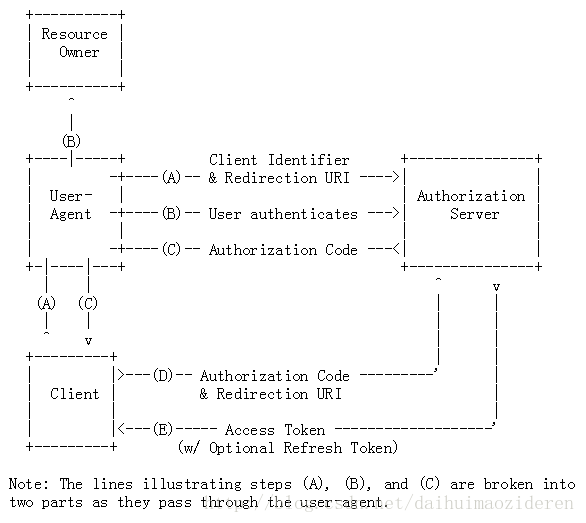
"expires\_in":3600,

"refresh\_token":"tGzv3JOkF0XG5Qx2TlKWIA",

"example\_parameter":"example\_value"

}

## （三）授权码模式



该模式是五种授权中最啰嗦的。从流程上看，申请分为两个阶段。首先，需要申请Authorization Code，之后，使用Authorization Code来申请Access Token。

我们以优酷为例，讲述流程。   
（A）优酷向用户展示，“我”可以支持QQ、微信、支付宝等第三方式登录。用户选择其中一种，例如QQ，则跳转至QQ界面（User-Agent），通常为WEB界面。此时，若用户未登录，则要求用户登录，若已登录，则询问是否授权，以及展示授权后会获得哪些权限。   
（B）用户点击授权，触发申请。   
（C）假设授权通过，QQ认证服务器将用户导向优酷事先指定的”重定向URI”（redirection URI），同时附上一个Authorization Code。   
（D）优酷收到授权码，附上早先的”重定向URI”，向认证服务器申请Access Token。这一步是在优酷的后台的服务器上完成的，对用户不可见。   
（E）认证服务器核对了授权码和重定向URI，确认无误后，向优酷发送访问令牌（access token）和更新令牌（refresh token）。

Authorization Code只能使用一次，且有时间限制，规范建议为10分钟。

申请Authorization Code时，需要在URI的query component中附加以下信息

response\_type：

REQUIRED. Value MUST be set to "code".

client\_id：

REQUIRED. The client identifier

redirect\_uri：

OPTIONAL.

scope:

OPTIONAL. The scope of the access request

state：

RECOMMENDED. An opaque value used by the client to

maintain state between the request and callback. The

redirecting the user-agent back to the client. The

parameter SHOULD be used for preventing cross-site

request forgery.

Authorization Code返回时，URI的query component中的附加信息如下

code：

REQUIRED. The authorization code generated by the

authorization server. The authorization code MUST of

expire shortly after it is issued to mitigate the risk

of leaks. A maximum authorization code lifetime of 10

minutes is RECOMMENDED. The client MUST NOT use the

authorization code more than once. If an authorization

code is used more than once, the authorization server

MUST deny the request and SHOULD revoke (when possible)

all tokens previously issued based on that authorization

code. The authorization code is bound to the client

identifier and redirection URI.

state：

REQUIRED if the "state" parameter was present in the

client authorization request. The exact value received

from the client.

使用Authorization Code申请Access Token时，需要在HTTP request entity-body中提交以下信息。

grant\_type：

REQUIRED. Value MUST be set to "authorization\_code".

code：

REQUIRED. The authorization code received from the

authorization server.

redirect\_uri：

REQUIRED, if the "redirect\_uri" parameter was included

in the authorization request, and their values MUST be

identical.

client\_id：

REQUIRED, if the client is not authenticating with the

authorization server.

申请成功后，授权服务器将返回access token和token有效时间，以及可选的refresh token。

附规范中的例子。   
Request Authorization Code：

GET /authorize?response\_type=code&client\_id=s6BhdRkqt3&state=xyz&redirect\_uri=https%3A%2F%2Fclient%2Eexample%2Ecom%2Fcb HTTP/1.1

Host: server.example.com

Authorization Code Response：

HTTP/1.1 302 Found

Location: https://client.example.com/cb?code=SplxlOBeZQQYbYS6WxSbIA&state=xyz

Request Access Token：

POST /token HTTP/1.1

Host: server.example.com

Authorization: Basic czZCaGRSa3F0MzpnWDFmQmF0M2JW

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

grant\_type=authorization\_code&code=SplxlOBeZQQYbYS6WxSbIA

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

Access Token Response：

HTTP/1.1 200 OK

Content-Type: application/json;charset=UTF-8

Cache-Control: no-store

Pragma: no-cache

{

"access\_token":"2YotnFZFEjr1zCsicMWpAA",

"token\_type":"example",

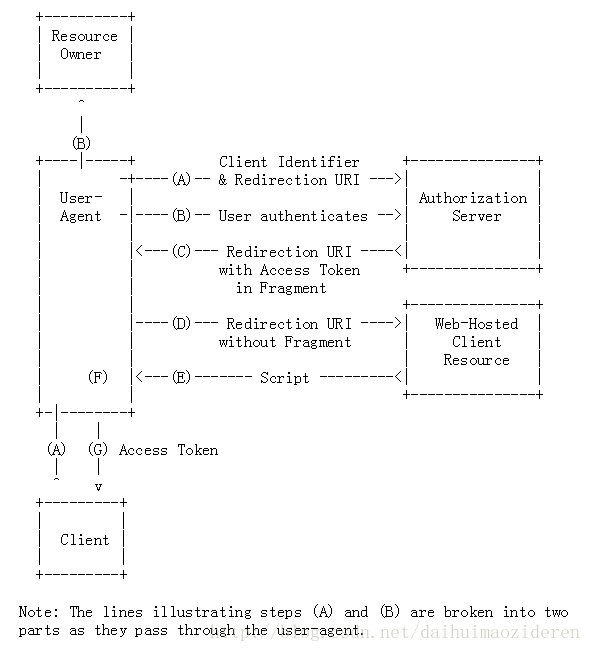
"expires\_in":3600,

"refresh\_token":"tGzv3JOkF0XG5Qx2TlKWIA",

"example\_parameter":"example\_value"

}

## （四）授权码简化模式



在授权码模式中，Authorization Code和Access Token都由授权服务器生成和验证，而最终只用到Access Token，这让Authorization Code显得无足轻重。因此，授权码简化模式，去掉了Authorization Code的申请流程，从而通过User-Agent（Browser）直接申请Access Token。

我们还以优酷为例，讲述流程。   
（A）优酷向用户展示，“我”可以支持QQ、微信、支付宝等第三方式登录。用户选择其中一种，例如QQ，则跳转至QQ界面（User-Agent），通常为WEB界面。此时，若用户未登录，则要求用户登录，若已登录，则询问是否授权，以及展示授权后会获得哪些权限。   
（B）用户点击授权，触发申请。   
（C）假设授权通过，QQ认证服务器将用户导向优酷事先指定的”重定向URI”（redirection URI），同时附上Access Token。   
（D）QQ界面（User-Agent）收到重定向响应后，向优酷服务器提出请求，表示想提取URI中的Access Token。   
（E）优酷服务器返回带有解析脚本的页面，用于解析重定向URI fragment中的Access Token。   
（F）User-Agent使用解析脚本，获取Access Token。   
（G）User-Agent将Access Token转交给优酷。

申请Access Token时，需要在URI的query component中附加以下信息

response\_type：

REQUIRED. Value MUST be set to "token".

client\_id：

REQUIRED. The client identifier.

redirect\_uri：

OPTIONAL.

scope：

OPTIONAL. The scope of the access request.

state：

RECOMMENDED. An opaque value used by the client to

maintain state between the request and callback. The

authorization server includes this value when

redirecting the user-agent back to the client. The

parameter SHOULD be used for preventing cross-site

request forgery.

Access Token返回时，URI的query component中的附加信息如下

access\_token：

REQUIRED. The access token issued by the authorization

server.

token\_type：

REQUIRED. The type of the token issued. Value is case

insensitive.

expires\_in:

RECOMMENDED. The lifetime in seconds of the access

token. For example, the value "3600" denotes that the

access token will expire in one hour from the time the

response was generated. If omitted, the authorization

server SHOULD provide the expiration time via other

means or document the default value.

scope：

OPTIONAL, if identical to the scope requested by the

client; otherwise, REQUIRED. The scope of the access

token.

state：

REQUIRED if the "state" parameter was present in the

client authorization request. The exact value received

from the client.

**注：授权码简化模式，不生成refresh token。**

附规范例子。   
Access Token Request:

GET /authorize?response\_type=token&client\_id=s6BhdRkqt3&state=xyz&redirect\_uri=https%3A%2F%2Fclient%2Eexample%2Ecom%2Fcb HTTP/1.1

Host: server.example.com

Access Token Response:

HTTP/1.1 302 Found

Location: http://example.com/cb#access\_token=2YotnFZFEjr1zCsicMWpAA

&state=xyz&token\_type=example&expires\_in=3600

**注：此处存在质疑。规范例子中URI不一致，个人认为Response中的URI应为：**[***https://client.example.com/cb***](https://client.example.com/cb)**。若同学有其他见解，望赐教。**

## （五）扩展模式

扩展模式，其实是一种自定义模式。规范中仅对“grant type”参数提出了须为URI的要求。对于其他申请数据，可以根据需求进行自定义。

附规范例子。

POST /token HTTP/1.1

Host: server.example.com

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

grant\_type=urn%3Aietf%3Aparams%3Aoauth%3Agrant-type%3Asaml2-

bearer&assertion=PEFzc2VydGlvbiBJc3N1ZUluc3RhbnQ9IjIwMTEtMDU

[...omitted for brevity...]aG5TdGF0ZW1lbnQ-PC9Bc3NlcnRpb24-

至此，授权篇完结。若有理解有误之处，还望同学们指教。

之后，将重心放在服务器及客户端的实现。使用框架为Spring Security OAuth。

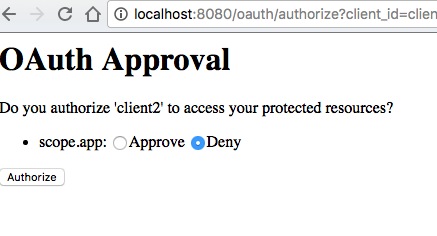
# 授权流程

访问授权页面：

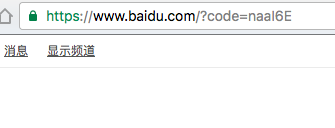
localhost:8081/oauth/authorize?client\_id=client&response\_type=code&redirect\_uri=http://www.baidu.com

* 1

此时浏览器会让你输入用户名密码，这是因为 Spring Security 在默认情况下会对所有URL添加Basic Auth认证。默认的用户名为user, 密码是随机生成的，在控制台日志中可以看到。

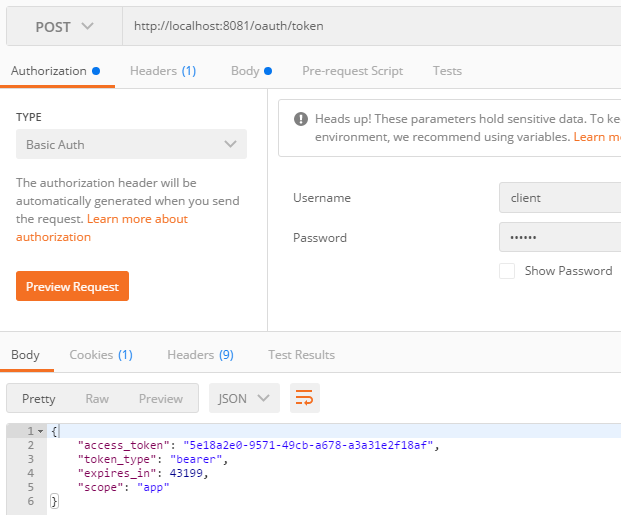


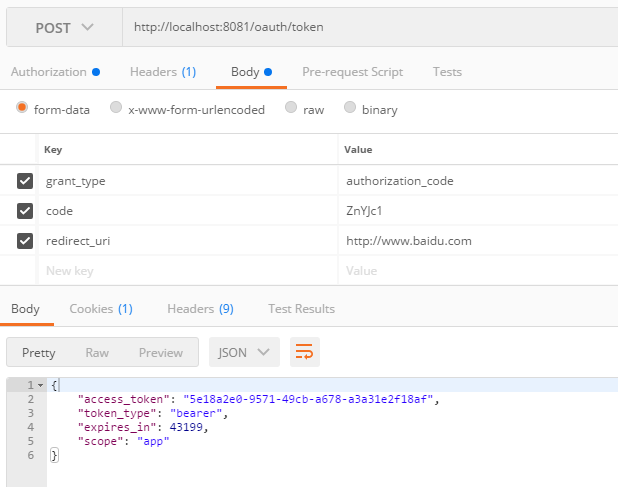
画风虽然很简陋，但是基本功能都具备了。点击Authorize后，浏览器就会重定向到百度，并带上code参数：



拿到code以后，就可以调用

POST/GET <http://client:secret@localhost:8080/oauth/token>





返回如下：

{

"access\_token": "32a1ca28-bc7a-4147-88a1-c95abcc30556", // 令牌

"token\_type": "bearer",

"expires\_in": 2591999,

"scope": "app"

}