OAuth（开放授权）是一个开放标准，允许用户授权第三方移动应用访问他们存储在另外的服务提供者上的信息，而不需要将用户名和密码提供给第三方移动应用或分享他们数据的所有内容，OAuth2.0是OAuth协议的延续版本，但不向后兼容OAuth 1.0即完全废止了OAuth1.0。

应用场景

第三方应用授权登录：在APP或者网页接入一些第三方应用时，时长会需要用户登录另一个合作平台，比如QQ，微博，微信的授权登录。

原生app授权：app登录请求后台接口，为了安全认证，所有请求都带token信息，如果登录验证、请求后台数据。

前后端分离单页面应用（spa）：前后端分离框架，前端请求后台数据，需要进行oauth2安全认证，比如使用vue、react后者h5开发的app

名词说明

（1） Third-party application：第三方应用程序，本文中又称"客户端"（client），比如打开知乎，使用第三方登录，选择qq登录，这时候知乎就是客户端。  
（2）HTTP service：HTTP服务提供商，本文中简称"服务提供商"，即上例的qq。  
（3）Resource Owner：资源所有者，本文中又称"用户"（user）,即登录用户。  
（4）User Agent：用户代理，本文中就是指浏览器。  
（5）Authorization server：认证服务器，即服务提供商专门用来处理认证的服务器。  
（6）Resource server：资源服务器，即服务提供商存放用户生成的资源的服务器。它与认证服务器，可以是同一台服务器，也可以是不同的服务器。

运行流程

（A）用户打开客户端以后，客户端要求用户给予授权。  
（B）用户同意给予客户端授权。  
（C）客户端使用上一步获得的授权，向认证服务器申请令牌。  
（D）认证服务器对客户端进行认证以后，确认无误，同意发放令牌。  
（E）客户端使用令牌，向资源服务器申请获取资源。  
（F）资源服务器确认令牌无误，同意向客户端开放资  
源。

授权模式

授权码模式

授权码模式（authorization code）是功能最完整、流程最严密的授权模式。

（A）用户访问客户端，后者将前者导向认证服务器。

（B）用户选择是否给予客户端授权。

（C）假设用户给予授权，认证服务器将用户导向客户端事先指定的"重定向URI"（redirection URI），同时附上一个授权码。

（D）客户端收到授权码，附上早先的"重定向URI"，向认证服务器申请令牌。这一步是在客户端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，向客户端发送访问令牌（access token）和更新令牌（refresh token）。

A步骤中，客户端申请认证的URI，包含以下参数：

response\_type：表示授权类型，必选项，此处的值固定为"code"

client\_id：表示客户端的ID，必选项

redirect\_uri：表示重定向URI，可选项

scope：表示申请的权限范围，可选项

state：表示客户端的当前状态，可以指定任意值，认证服务器会原封不动地返回这个值。

例子：

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

C步骤中，服务器回应客户端的URI，包含以下参数：

code：表示授权码，必选项。该码的有效期应该很短，通常设为10分钟，客户端只能使用该码一次，否则会被授权服务器拒绝。该码与客户端ID和重定向URI，是一一对应关系。

state：如果客户端的请求中包含这个参数，认证服务器的回应也必须一模一样包含这个参数。

例子：

HTTP/1.1 302 Found

Location:

<a href="https://client.example.com/cb">[https://client.example.com/cb](https://links.jianshu.com/go?to=https%3A%2F%2Fclient.example.com%2Fcb)

?code=SplxlOBeZQQYbYS6WxSbIA&state=xyz

D步骤中，客户端向认证服务器申请令牌的HTTP请求，包含以下参数：

grant\_type：表示使用的授权模式，必选项，此处的值固定为"authorization\_code"。

code：表示上一步获得的授权码，必选项。

redirect\_uri：表示重定向URI，必选项，且必须与A步骤中的该参数值保持一致。

client\_id：表示客户端ID，必选项。

下面是一个例子：

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

E步骤中，认证服务器发送的HTTP回复，包含以下参数：

access\_token：表示访问令牌，必选项。

token\_type：表示令牌类型，该值大小写不敏感，必选项，可以是bearer类型或mac类型。

expires\_in：表示过期时间，单位为秒。如果省略该参数，必须其他方式设置过期时间。

efresh\_token：表示更新令牌，用来获取下一次的访问令牌，可选项。

scope：表示权限范围，如果与客户端申请的范围一致，此项可省略。

下面是一个例子：

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"

}

从上面代码可以看到，相关参数使用JSON格式发送（Content-Type: application/json）。此外，HTTP头信息中明确指定不得缓存。

Bearer类型和MAC类型Token 区别

Bearer 介绍

优点：  
　　调用简单，不需要对请求进行签名。  
缺点：  
　　请求API需要使用https协议保证信息传输安全。  
　　Access Token有效期一个月，过期后需要使用Refresh Token进行刷新。

MAC 介绍

优点：  
　　不依赖https协议，无协议加密带来的性能开销。  
　　Access Token长期有效，无需使用Refresh Token刷新。  
缺点：  
　　需要进行MAC计算。

Bearer类型token定义了三种token传递策略，客户端在传递token时必须使用其中的一种，且最多一种。

1. 放在Authorization请求首部
2. 放在请求实体中
3. 放在URI请求参数中

（1）用户访问客户端，后者将前者导向认证服务器，假设用户给予授权，认证服务器将用户导向客户端事先指定的"重定向URI"（redirection URI），同时附上一个授权码。

（2）客户端收到授权码，附上早先的"重定向URI"，向认证服务器申请令牌：GET /oauth/token?response\_type=code&client\_id=test&redirect\_uri=重定向页面链接。请求成功返回code授权码，一般有效时间是10分钟。

（3）认证服务器核对了授权码和重定向URI，确认无误后，向客户端发送访问令牌（access token）和更新令牌（refresh token）。POST /oauth/token?response\_type=authorization\_code&code=SplxlOBeZQQYbYS6WxSbIA&redirect\_uri=重定向页面链接。

简化模式

简化模式（implicit grant type）不通过第三方应用程序的服务器，直接在浏览器中向认证服务器申请令牌，跳过了"授权码"这个步骤，因此得名。所有步骤在浏览器中完成，令牌对访问者是可见的，且客户端不需要认证。

流程步骤：

（A）客户端将用户导向认证服务器。  
（B）用户决定是否给于客户端授权。  
（C）假设用户给予授权，认证服务器将用户导向客户端指定的"重定向URI"，并在URI的Hash部分包含了访问令牌。  
（D）浏览器向资源服务器发出请求，其中不包括上一步收到的Hash值。  
（E）资源服务器返回一个网页，其中包含的代码可以获取Hash值中的令牌。  
（F）浏览器执行上一步获得的脚本，提取出令牌。  
（G）浏览器将令牌发给客户端。

A步骤中，客户端发出的HTTP请求，包含以下参数：

response\_type：表示授权类型，此处的值固定为"token"，必选项。

client\_id：表示客户端的ID，必选项。

redirect\_uri：表示重定向的URI，可选项。

scope：表示权限范围，可选项。

state：表示客户端的当前状态，可以指定任意值，认证服务器会原封不动地返回这个值。

下面是一个例子:

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

C步骤中，认证服务器回应客户端的URI，包含以下参数：

access\_token：表示访问令牌，必选项。

token\_type：表示令牌类型，该值大小写不敏感，必选项。

expires\_in：表示过期时间，单位为秒。如果省略该参数，必须其他方式设置过期时间。

scope：表示权限范围，如果与客户端申请的范围一致，此项可省略。

state：如果客户端的请求中包含这个参数，认证服务器的回应也必须一模一样包含这个参数。

下面是一个例子:

HTTP/1.1 302 Found

Location:[http://example.com/cb](https://links.jianshu.com/go?to=http%3A%2F%2Fexample.com%2Fcb)#access\_token=2YotnFZFEjr1zCsicMWpAA

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

    在上面的例子中，认证服务器用HTTP头信息的Location栏，指定浏览器重定向的网址。注意，在这个网址的Hash部分包含了令牌。

    根据上面的D步骤，下一步浏览器会访问Location指定的网址，但是Hash部分不会发送。接下来的E步骤，服务提供商的资源服务器发送过来的代码，会提取出Hash中的令牌。

密码模式

密码模式（Resource Owner Password Credentials Grant）中，用户向客户端提供自己的用户名和密码。客户端使用这些信息，向"服务商提供商"索要授权。在这种模式中，用户必须把自己的密码给客户端，但是客户端不得储存密码。这通常用在用户对客户端高度信任的情况下。一般不支持refresh token。

步骤说明：  
（A）用户向客户端提供用户名和密码。  
（B）客户端将用户名和密码发给认证服务器，向后者请求令牌。  
（C）认证服务器确认无误后，向客户端提供访问令牌。

B步骤中，客户端发出的HTTP请求，包含以下参数：

grant\_type：表示授权类型，此处的值固定为"password"，必选项。

username：表示用户名，必选项。

password：表示用户的密码，必选项。

scope：表示权限范围，可选项。

下面是一个例子:

POST /tokenHTTP/1.1

Host:server.example.com

Authorization:Basic czZCaGRSa3F0MzpnWDFmQmF0M2JW

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

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

C步骤中，认证服务器向客户端发送访问令牌，下面是一个例子:

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"

}

    上面代码中，各个参数的含义参见5.1授权码模式一节。整个过程中，客户端不得保存用户的密码。

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

客户端模式

指客户端以自己的名义，而不是以用户的名义，向"服务提供商"进行认证。严格地说，客户端模式并不属于OAuth框架所要解决的问题。在这种模式中，用户直接向客户端注册，客户端以自己的名义要求"服务提供商"提供服务，其实不存在授权问题。

 A步骤中，客户端发出的HTTP请求，包含以下参数：

granttype：表示授权类型，此处的值固定为"clientcredentials"，必选项。

scope：表示权限范围，可选项。

POST /token HTTP/1.1

Host: server.example.com

Authorization: Basic

czZCaGRSa3F0MzpnWDFmQmF0M2JW

Content-Type:

application/x-www-form-urlencoded

grant\_type=client\_credentials

认证服务器必须以某种方式，验证客户端身份。

B步骤中，认证服务器向客户端发送访问令牌，下面是一个例子。

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"

}

上面代码中，各个参数的含义参见5.1 授权码模式一节。

POST /token HTTP/1.1

Host: server.example.com

Authorization: Basic czZCaGRSa3F0MzpnWDFmQmF0M2JW

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

grant\_type=client\_credentials

更新令牌

    如果用户访问的时候，客户端的"访问令牌"已经过期，则需要使用"更新令牌"申请一个新的访问令牌。

    客户端发出更新令牌的HTTP请求，包含以下参数：

    granttype：表示使用的授权模式，此处的值固定为"refreshtoken"，必选项。

    refresh\_token：表示早前收到的更新令牌，必选项。

    scope：表示申请的授权范围，不可以超出上一次申请的范围，如果省略该参数，则表示与上一次一致。

下面是一个例子。

Host: server.example.com

Authorization: Basic

czZCaGRSa3F0MzpnWDFmQmF0M2JW

Content-Type:

application/x-www-form-urlencoded

grant\_type=refresh\_token&refresh\_token=tGzv3JOkF0XG5Qx2TlKWIA