OAuth2.0 知多少

2017-03-19 圣杰 [dotNET跨平台](http://mp.weixin.qq.com/s?__biz=MzAwNTMxMzg1MA==&mid=2654068570&idx=2&sn=7d576e8a6a610bc1796cbe2498216c8d&chksm=80dbfb0fb7ac7219d61745943179be06f62872a609d98997ccaf1028c331e1fff1cd9fbcea2e&mpshare=1&scene=5&srcid=0327U5LfdkoEXuvuGljZc5Rf##)

**1. 引言**

周末逛简书，看了一篇写的极好的文章，点击大红心点赞，就直接给我跳转到登录界面了，原来点赞是需要登录的。

可是没有我并没有简书账号，一直使用的QQ的集成登录。下面有一排社交登录按钮，我们可以用第三方社交账号登陆即可。点击QQ图标，就给我跳转到了QQ登录授权页面，如下图：

从图片上我们可以看到主要包括两个部分，一个是左边的用户登录，一个是右边告知简书将获取哪些权限。输入QQ账号和密码，点击授权并登录，就成功登录到简书了，并成功获取到了我QQ的账号和昵称，如下图：

简书集成的社交登录，大大简化了我们的注册登录流程，真是一号在手上网无忧啊。  
这看似简单的集成，但背后的技术原理『OAuth2.0』可没那么简单，那我们废话不多说，一探究竟吧。

**2. OAuth 2.0**

OAuth 2.0是用于授权的行业标准协议。OAuth 2.0取代了在2006年创建的原始OAuth协议上所做的工作。OAuth 2.0专注于客户端开发人员的简单性，同时为Web应用程序，桌面应用程序，手机和客厅设备提供特定的授权流程。

在传统的client-server认证模型中，客户端请求访问服务器上受限的资源（protected resource），需要通过使用资源所有者（resource owner）的凭证在服务器上进行认证。为了支持第三方应用程序访问受限资源，资源所有者需要向第三方应用共享其凭证。这就会造成以下问题：

1. 第三方应用为了后续使用，会存储资源所有者的凭证主要是密码。
2. 服务端需要支持密码认证，尽管密码认证不安全。
3. 第三方应用获得对资源的过度访问而不仅局限于受限资源，且资源所有者没有办法对其进行限制。
4. 资源所有者无法收回权限，除非修改密码。
5. 如果第三方应用的密码被破解，就会导致所有被该密码保护的数据被泄露。

想一想这样一个场景，如果简书是直接使用QQ用户名密码登录，简书就很有可能会为了后续业务的需要而擅自保存QQ用户名及密码，简书只要拿到了QQ用户名密码就可以访问不仅仅QQ昵称、头像等信息，甚至可以获取到QQ用户的所有通讯录列表。如果简书的账号密码泄露，就会直接影响到QQ数据的安全。这是一个可怕的问题。

所以OAuth应运而生，来解决这一问题。

**3. OAuth 2.0授权流程**

下面我们就以简书使用QQ授权登录为例，来捋一捋OAuth 2.0的流程。  
先来看看OAuth 2.0的流程，如下图所示：

这里面主要包含四个角色：

1. Client：需要授权的客户端，本文中就是【简书】。
2. Resource Owner：资源所有者，在本文中你可能会以为是 QQ，但要想清楚，QQ是属于个人的，所以在本文中资源所有者是指【QQ用户】。
3. Authorization Server：认证服务器，本文中特指【QQ互联平台】。
4. Resource Server：资源服务器，顾名思义，用来专门保存资源的服务器，接受通过访问令牌进行访问。本文特指【QQ用户信息中心】。

**3.1. 第一步：引导用户到认证服务器**

圣杰打开简书网页，简书跳转到登录界面，要求用户登录。可是圣杰未在简书注册帐号，所以就点击了QQ图标，使用QQ帐号进行集成登录。跳转到QQ登录界面后，QQ要求用户授权。  
这一步中简书主要做了这样一件事就是引导用户到认证服务器。  
很显然【QQ互联平台】就是认证服务器。

如何引导？当然是页面跳转。  
那认证服务器如何知道是简书过来的认证请求？  
当然是传参。  
那需要传递哪些参数呢？

* response\_type：表示响应类型，必选项，此处的值固定为"code"；
* client\_id：表示客户端的ID，用来标志授权请求的来源，必选项；
* redirect\_uri：成功授权后的回调地址；
* 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。

咱们看看简书实际发送的授权请求Url是：  
https://graph.qq.com/oauth2.0/authorize?client\_id=100410602 &redirect\_uri=http://www.jianshu.com/users/auth/qq\_connect/callback &response\_type=code &state=bb38108d1aaf567c72da0f1167e87142d0e20cb2bb24ec5a

无图无真相，咱们看看控制台的网络监控：

如图所示，除了scope参数外，其他四个参数均有传参。  
此时你可能唯一对state参数比较迷惑，传递一个state参数，认证服务器会原封不动返回，那还干嘛要传递state参数呢？

我的理解是，简书用一个guid加长版字符串来唯一标识一个授权请求。这样才会正确获取授权服务器返回的授权码。

这里你可能会问了，既然我知道了这些参数，我岂不是可以伪造简书认证请求，修改redirect\_uri参数跳转到个人的网站，然后不就可以获取QQ授权？

跟我一样太傻太天真，简书在QQ互联平台申请时肯定已经预留备案了要跳转返回的URL。QQ互联平台在收到简书的授权请求时肯定会验证回调Url的。

**3.2. 第二步：用户同意为第三方客户端授权**

这一步，对于用户来说，只需要使用资源所有者（QQ）的用户名密码登录，并同意授权即可。点击授权并登录后，授权服务器首先会post一个请求回服务器进行用户认证，认证通过后授权服务器会生成一个授权码，然后服务器根据授权请求的redirect\_uri进行跳转，并返回授权码code和授权请求中传递的state。  
这里要注意的是：**授权码有一个短暂的时效**

无图无真相，咱们还是看一下控制台网络监控：

从图中即可验证我们上面所说，最终跳转回简书的Url为：  
http://www.jianshu.com/users/auth/qq\_connect/callback?code=093B9307E38DC5A2C3AD147B150F2AB3 &state=bb38108d1aaf567c72da0f1167e87142d0e20cb2bb24ec5a  
和之前的授权请求URL进行对比，可以发现redirect\_uri、state完全一致。  
而code=093B9307E38DC5A2C3AD147B150F2AB3就是返回的授权码。

**3.3. 第三步：使用授权码向认证服务器申请令牌**

从这一步开始，对于用户来说是察觉不到的。简书后台默默的在做后续的工作。

简书拿到QQ互联平台返回的授权码后，需要根据授权码再次向认证服务器申请令牌（access token）。  
到这里有必要再理清两个概念：

* 授权码（Authorization Code）：相当于授权服务器口头告诉简书，用户同意授权使用他的QQ登录简书了。
* 令牌（Access Token）：相当于临时身份证。

那如何申请令牌呢？  
简书需要后台发送一个get请求到认证服务器（QQ互联平台）。  
那要携带哪些必要信息呢？  
是的，要携带以下参数：

* grant\_type：表示授权类型，此处的值固定为"authorization\_code"，必选项；
* client\_id：表示从QQ互联平台申请到的客户端ID，用来标志请求的来源，必选项；
* client\_secret：这个是从QQ互联平台申请到的客户端认证密钥，机密信息十分重要，必选项；
* redirect\_uri：成功申请到令牌后的回调地址；
* code：上一步申请到的授权码。

根据以上信息我们可以模拟一个申请AccessToken的请求：  
https://graph.qq.com/oauth2.0/token?client\_id=100410602 &client\_secret=123456jianshu &redirect\_uri=http://www.jianshu.com/users/auth/qq\_connect/callback &grant\_type=authorization\_code &code=093B9307E38DC5A2C3AD147B150F2AB3

发送完该请求后，认证服务器验证通过后就会发放令牌，并跳转会简书，其中应该包含以下信息：

* access\_token：令牌
* expires\_in：access token的有效期，单位为秒。
* refresh\_token：在授权自动续期步骤中，获取新的Access\_Token时需要提供的参数。

同样，我们可以模拟出一个返回的token：  
http://www.jianshu.com/users/auth/qq\_connect/callback?access\_token=548ADF2D5E1C5E88H4JH15FKUN51F &expires\_in=36000&refresh\_token=53AD68JH834HHJF9J349FJADF3

这个时候简书还有一件事情要做，就是把用户token写到cookie里，进行用户登录状态的维持。咱们还是打开控制器验证一下。

从图中可以看出简书把用户token保存在名为remember\_user\_token的cookie里。  
不用打cookie的歪主意了，肯定是加密了的。  
可以尝试下手动把remember\_user\_token这条cookie删除，保证刷新界面后需要你重新登录简书。

**3.4. 第四步：向资源服务器申请资源**

有了token，向资源服务器提供的资源接口发送一个get请求不就行了，资源服务器校验令牌无误，就会向简书返回资源（QQ用户信息）。

同样咱们也来模拟一个使用token请求QQ用户基本信息资源的URL：  
https://graph.qq.com/user/get\_user\_info?client\_id=100410602 &qq=2098769873 &access\_token=548ADF2D5E1C5E88H4JH15FKUN51F

到这一步OAuth2.0的流程可以说是结束了，但是对于简书来说还有重要的事情要做。那就是：  
**拿到token、reresh\_token和用户数据这么重要的东西不存数据库傻呀？**

**3.5. 第五步：令牌延期（刷新）**

你肯定对第四步返回的refresh\_token比较好奇。  
它是用来对令牌进行延期（刷新）的。为什么会有两种说法呢，因为可能认证服务器会重新生成一个令牌，也有可能  
对过期的令牌进行延期。

比如说，QQ互联平台为了安全性考虑，返回的access\_token是有时间限制的，假如用户某天不想授权了呢，总不能给了个access\_token你几年后还能用吧。我们上面模拟返回的令牌有效期为10小时。10小时后，用户打开浏览器逛简书，浏览器中用户的token对应的cookie已过期。简书发现浏览器没有携带token信息过来，就明白token失效了，需要重新向认证平台申请授权。如果让用户再点击QQ进行登录授权，这明显用户体验不好。咋搞呢？refresh\_token就派上了用场，可以直接跳过前面申请授权码的步骤，当发现token失效了，简书从浏览器携带的cookie中的sessionid找到存储在数据库中的refresh\_token，然后再使用refresh\_token进行token续期（刷新）。

那用refresh\_token进行token续期需要怎么做呢？  
同样需要向认证服务器发送一个get请求。  
需要哪些参数？

* grant\_type：表示授权类型，此处的值固定为"refresh\_token"，必选项；
* client\_id：表示从QQ互联平台申请到的客户端ID，用来标志请求的来源，必选项；
* client\_secret：这个是从QQ互联平台申请到的客户端认证密钥，机密信息十分重要，必选项；
* refresh\_token：即申请令牌返回的refresh\_token。

根据上述信息，我们又可以模拟一个令牌刷新的URL：  
https://graph.qq.com/oauth2.0/token?client\_id=100410602 &client\_secret=123456jianshu &redirect\_uri=http://www.jianshu.com/users/auth/qq\_connect/callback &grant\_type=refresh\_token &refresh\_token=53AD68JH834HHJF9J349FJADF3  
那返回的结果呢?  
和第四步返回的结果一样。

这里你可能又有疑问了，那既然每次进行令牌延期后都会重新返回一个refresh\_token，那岂不是我可以使用refresh\_token无限延期？  
天真如我啊，refresh\_token也是有过期时间的。而这个过期时间具体是由认证服务器决定的。  
一般来说refresh\_token的过期时间要大于access\_token的过期时间。只有这样，access\_token过期时，才可以使用refresh\_token进行令牌延期（刷新）。

举个简单例子：  
假设简书从QQ互联平台默认获取到的access\_token的有效期是1天，refresh\_token的有效期为一周。

用户今天使用QQ登录授权后，过了两天再去逛简书，简书发现token失效，立马用refresh\_token刷新令牌，默默的完成了授权的延期。  
假如用户隔了两周再去逛简书，简书一核对，access\_token、refresh\_token全都失效，就只能乖乖引导用户到授权页面重新授权，也就是回到OAuth2.0的第一步。

**4.0 总结**

本文以简书通过QQ进行授权登录为例，对OAuth2.0 的授权流程进行了梳理，希望通读此文，对你有所帮助。

如果对OAuth2.0有所了解的话，你应该明白本文其实是对OAuth2.0中**授权码模式**授权方式的讲解。

如果想了解OAuth2.0其他几种授权方式，建议参考理解*OAuth 2.0* - 阮一峰的网络日志。

原文地址：http://www.cnblogs.com/sheng-jie/p/6564520.html

**.NET社区新闻，深度好文，微信中搜索dotNET跨平台或扫描二维码关注**