## [springboot和springsecurity整合OAuth2](https://www.cnblogs.com/ifme/p/12188982.html)

### 1. OAuth2.0介绍

OAuth（开放授权）是一个开放标准，允许用户授权第三方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。很多大公司如Google，Yahoo，Microsoft等都提供了OAUTH认证服务，这些都足以说明OAUTH标准逐渐成为开放资源授权的标准。

OAuth协议目前发展到2.0版本，1.0版本过于复杂，2.0版本已得到广泛应用。

下边分析一个 OAuth2认证的例子，通过例子去理解OAuth2.0协议的认证流程，

这个过程的简要描述如下：

用户借助微信认证登录A网站，用户就不用单独在A网站注册用户，怎么样算认证成功吗？A网站需要成功从微信获取用户的身份信息则认为用户认证成功，那如何从微信获取用户的身份信息？

用户信息的拥有者是用户本人，微信需要经过用户的同意方可为A网站生成令牌，A网站拿此令牌方可从微信获取用户的信息。

1、 客户端请求第三方授权  
用户进入A网站的登录页面，点击微信的图标以微信账号登录系统，用户是自己在微信里信息的资源拥有者。  
点击“微信”出现一个二维码，此时用户扫描二维码，开始给A网站授权。

2、 资源拥有者同意给客户端授权  
资源拥有者扫描二维码表示资源拥有者同意给客户端授权，微信会对资源拥有者的身份进行验证， 验证通过后，微信会询问用户是否给授权A网站访问自己的微信数据，用户点击“确认登录”表示同意授权，微信认证服务器会颁发一个授权码，并重定向到A网站的网站。

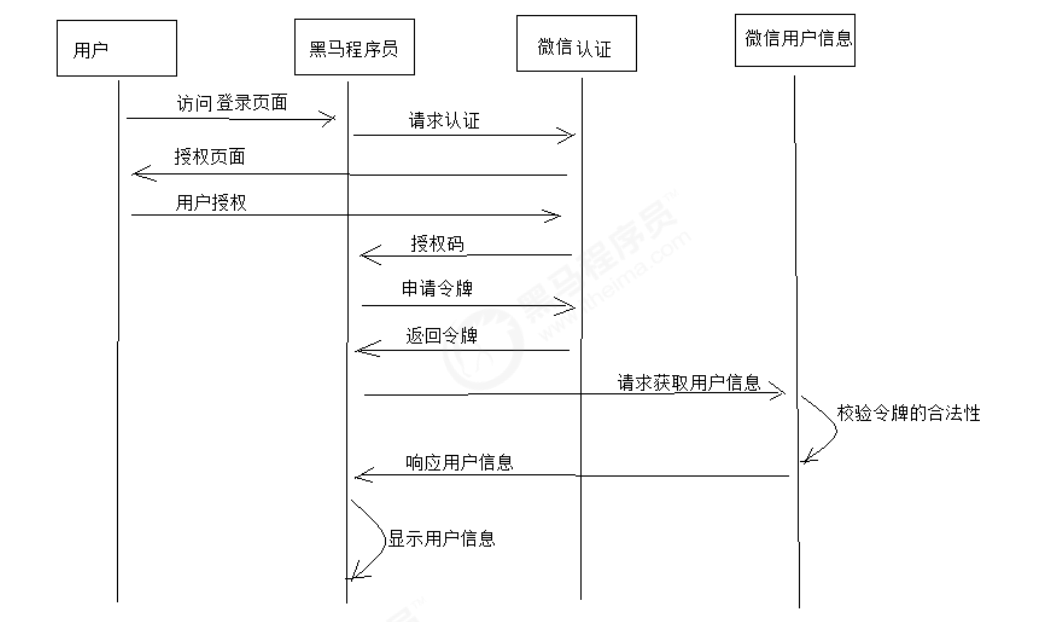
3、 客户端获取到授权码，请求认证服务器申请令牌  
此过程用户看不到，客户端应用程序请求认证服务器，请求携带授权码。

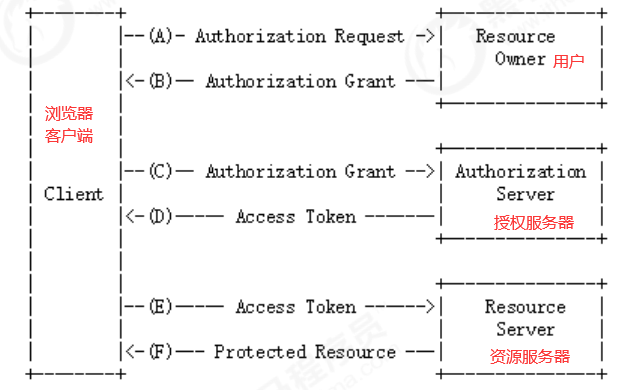
4、认证服务器向客户端响应令牌  
微信认证服务器验证了客户端请求的授权码，如果合法则给客户端颁发令牌，令牌是客户端访问资源的通行证。  
此交互过程用户看不到，当客户端拿到令牌后，用户在A网站看到已经登录成功。

5、客户端请求资源服务器的资源  
客户端携带令牌访问资源服务器的资源。  
A网站携带令牌请求访问微信服务器获取用户的基本信息。

6、资源服务器返回受保护资源  
资源服务器校验令牌的合法性，如果合法则向用户响应资源信息内容。

以上认证授权详细的执行流程如下：





OAauth2.0包括以下角色：

1、客户端

本身不存储资源，需要通过资源拥有者的授权去请求资源服务器的资源，比如：Android客户端、Web客户端（浏览器端）、微信客户端等。

2、资源拥有者

通常为用户，也可以是应用程序，即该资源的拥有者。

3、授权服务器（也称认证服务器）

用于服务提供商对资源拥有的身份进行认证、对访问资源进行授权，认证成功后会给客户端发放令牌（access\_token），作为客户端访问资源服务器的凭据。本例为微信的认证服务器。

4、资源服务器

存储资源的服务器，本例子为微信存储的用户信息。

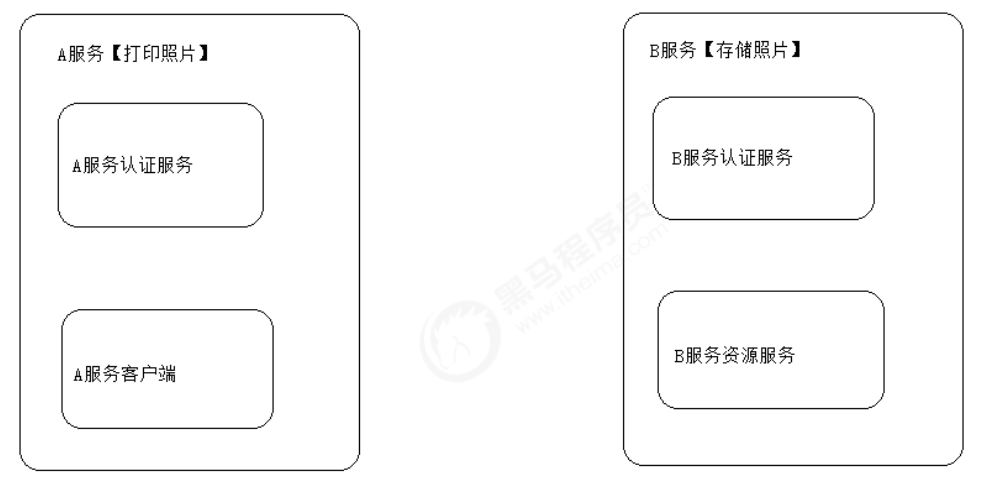
现在还有一个问题，服务提供商能允许随便一个客户端就接入到它的授权服务器吗？答案是否定的，服务提供商会给准入的接入方一个身份，用于接入时的凭据:

client\_id：客户端标识

client\_secret：客户端秘钥

因此，准确来说，授权服务器对两种OAuth2.0中的两个角色进行认证授权，分别是资源拥有者、客户端。

### 2. OAuth2.0 中四种授权方式



1. 授权码模式（authorization code）

流程

说明：【A服务客户端】需要用到【B服务资源服务】中的资源

第一步：【A服务客户端】将用户自动导航到【B服务认证服务】，这一步用户需要提供一个回调地址，以备【B服务认证服务】返回授权码使用。

第二步：用户点击授权按钮表示让【A服务客户端】使用【B服务资源服务】，这一步需要用户登录B服务，也就是说用户要事先具有B服务的使用权限。

第三步：【B服务认证服务】生成授权码，授权码将通过第一步提供的回调地址，返回给【A服务客户端】。

注意这个授权码并非通行【B服务资源服务】的通行凭证。

第四步：【A服务认证服务】携带上一步得到的授权码向【B服务认证服务】发送请求，获取通行凭证token。

第五步：【B服务认证服务】给【A服务认证服务】返回令牌token和更新令牌refresh token。

使用场景

授权码模式是OAuth2中最安全最完善的一种模式，应用场景最广泛，可以实现服务之间的调用，常见的微信，QQ等第三方登录也可采用这种方式实现。

2. 简化模式（implicit）

流程

说明：简化模式中没有【A服务认证服务】这一部分，全部有【A服务客户端】与B服务交互，整个过程不再有授权码，token直接暴露在浏览器。

第一步：【A服务客户端】将用户自动导航到【B服务认证服务】，这一步用户需要提供一个回调地址，以备【B服务认证服务】返回token使用，还会携带一个【A服务客户端】的状态标识state。

第二步：用户点击授权按钮表示让【A服务客户端】使用【B服务资源服务】，这一步需要用户登录B服务，也就是说用户要事先具有B服务的使用权限。

第三步：【 B服务认证服务】生成通行令牌token，token将通过第一步提供的回调地址，返回给【A服务客户端】。

使用场景

适用于A服务没有服务器的情况。比如：纯手机小程序，JavaScript语言实现的网页插件等。

3. 密码模式（resource owner password credentials）

流程

第一步：直接告诉【A服务客户端】自己的【B服务认证服务】的用户名和密码

第二步：【A服务客户端】携带【B服务认证服务】的用户名和密码向【B服务认证服务】发起请求获取

token。

第三步：【B服务认证服务】给【A服务客户端】颁发token。

使用场景

此种模式虽然简单，但是用户将B服务的用户名和密码暴露给了A服务，需要两个服务信任度非常高才能使用。

4. 客户端模式（client credentials）

流程

说明：这种模式其实已经不太属于OAuth2的范畴了。A服务完全脱离用户，以自己的身份去向B服务索取token。换言之，用户无需具备B服务的使用权也可以。完全是A服务与B服务内部的交互，与用户无关了。

第一步：A服务向B服务索取token。

第二步：B服务返回token给A服务。

使用场景

A服务本身需要B服务资源，与用户无关

### 3. OAuth2.0 sql语句

说明  
既可以写死在代码中，也可以写入到数据库中，通常写入到数据库  
create table oauth\_client\_details (

client\_id VARCHAR(256) PRIMARY KEY,

resource\_ids VARCHAR(256),

client\_secret VARCHAR(256),

scope VARCHAR(256),

authorized\_grant\_types VARCHAR(256),

web\_server\_redirect\_uri VARCHAR(256),

authorities VARCHAR(256),

access\_token\_validity INTEGER,

refresh\_token\_validity INTEGER,

additional\_information VARCHAR(4096),

autoapprove VARCHAR(256)

);

create table oauth\_client\_token (

token\_id VARCHAR(256),

token LONGVARBINARY,

authentication\_id VARCHAR(256) PRIMARY KEY,

user\_name VARCHAR(256),

client\_id VARCHAR(256)

);

create table oauth\_access\_token (

token\_id VARCHAR(256),

token LONGVARBINARY,

authentication\_id VARCHAR(256) PRIMARY KEY,

user\_name VARCHAR(256),

client\_id VARCHAR(256),

authentication LONGVARBINARY,

refresh\_token VARCHAR(256)

);

create table oauth\_refresh\_token (

token\_id VARCHAR(256),

token LONGVARBINARY,

authentication LONGVARBINARY

);

create table oauth\_code (

code VARCHAR(256), authentication LONGVARBINARY

);

create table oauth\_approvals (

userId VARCHAR(256),

clientId VARCHAR(256),

scope VARCHAR(256),

status VARCHAR(10),

expiresAt TIMESTAMP,

lastModifiedAt TIMESTAMP

);

-- customized oauth\_client\_details tablecreate table ClientDetails (

appId VARCHAR(256) PRIMARY KEY,

resourceIds VARCHAR(256),

appSecret VARCHAR(256),

scope VARCHAR(256),

grantTypes VARCHAR(256),

redirectUrl VARCHAR(256),

authorities VARCHAR(256),

access\_token\_validity INTEGER,

refresh\_token\_validity INTEGER,

additionalInformation VARCHAR(4096),

autoApproveScopes VARCHAR(256)

);