OAuth 2.0学习

[OAuth](http://en.wikipedia.org/wiki/OAuth)是一个关于授权（authorization）的开放网络标准，在全世界得到广泛应用。

**名词解释：**

（1） **Third-party application**：第三方应用程序，本文中又称"客户端"（client），即上一节例子中的"云冲印"。

（2）**HTTP service**：HTTP服务提供商，本文中简称"服务提供商"，即上一节例子中的Google。

（3）**Resource Owner**：资源所有者，本文中又称"用户"（user）。

（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。

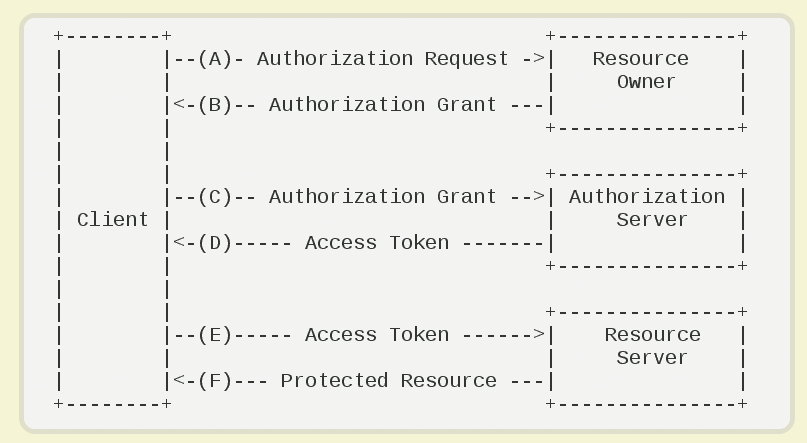
**OAuth的思路：**

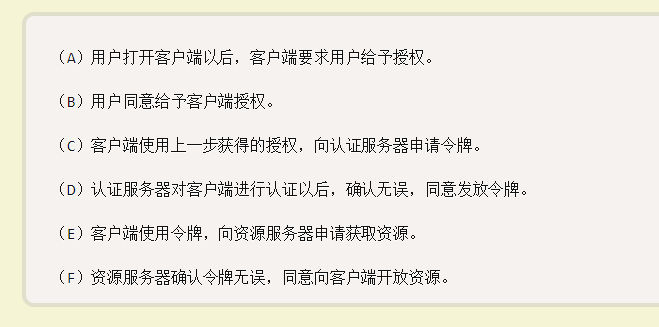
OAuth在"客户端"与"服务提供商"之间，设置了一个授权层（authorization layer）。

"客户端"不能直接登录"服务提供商"，只能登录授权层，以此将用户与客户端区分开来。"客户端"登录授权层所用的令牌（token），与用户的密码不同。用户可以在登录的时候，指定授权层令牌的权限范围和有效期。

"客户端"登录授权层以后，"服务提供商"根据令牌的权限范围和有效期，向"客户端"开放用户储存的资料。

**运行流程：**





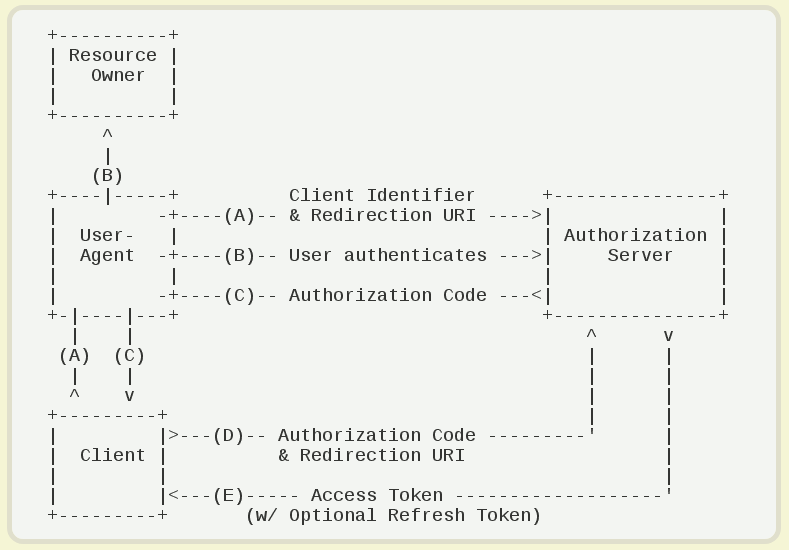
不难看出来，上面六个步骤之中，B是关键，即用户怎样才能给于客户端授权。有了这个授权以后，客户端就可以获取令牌，进而凭令牌获取资源。

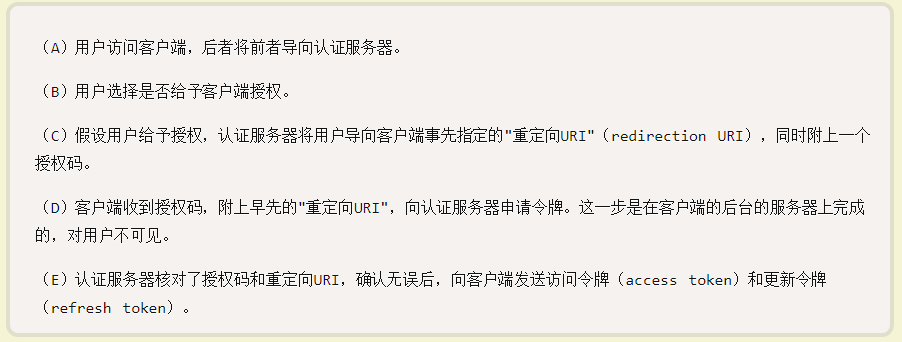
**客户端的授权模式：**

客户端必须得到用户的授权（authorization grant），才能获得令牌（access token）。OAuth 2.0定义了四种授权方式。

1. 授权码模式（authorization code）
2. 简化模式（implicit）
3. 密码模式（resource owner password credentials）
4. 客户端模式（client credentials）

（1）授权码模式（authorization code）是功能最完整、流程最严密的授权模式。它的特点就是通过客户端的后台服务器，与"服务提供商"的认证服务器进行互动。





下面是上面这些步骤所需要的参数。

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。

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。

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。

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。

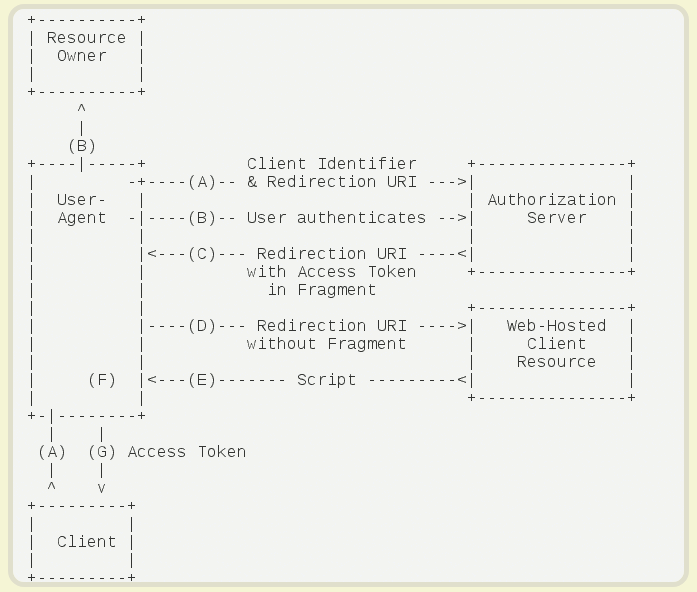
token\_type：表示令牌类型，该值大小写不敏感，必选项，可以是bearer类型或mac类型。

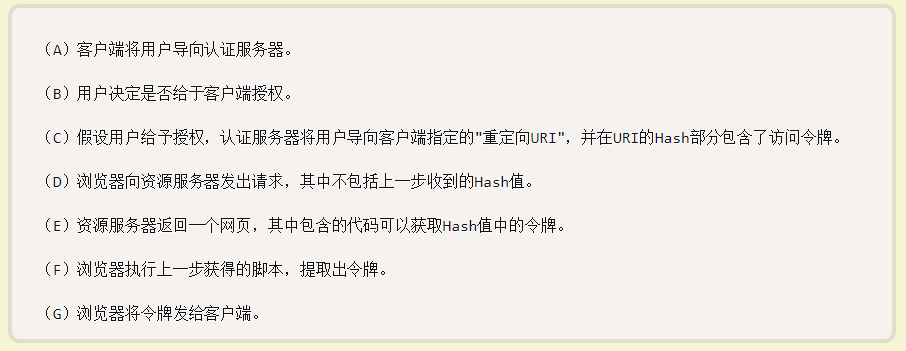
expires\_in：表示过期时间，单位为秒。如果省略该参数，必须其他方式设置过期时间。

refresh\_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。

（2）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。





下面是上面这些步骤所需要的参数。

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。

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。

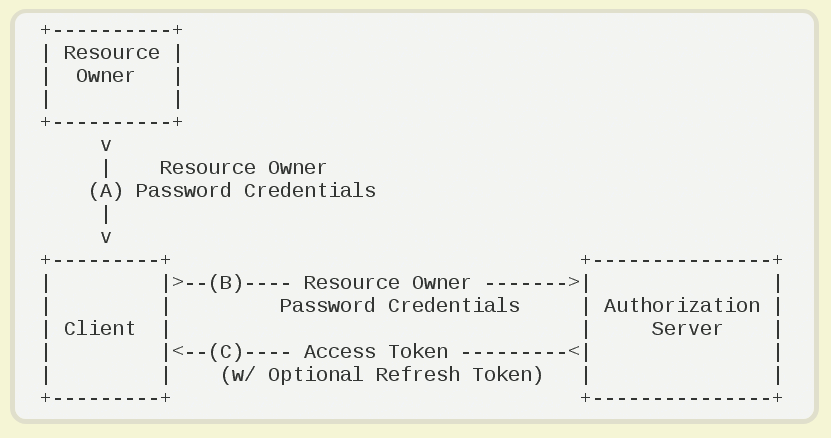
expires\_in：表示过期时间，单位为秒。如果省略该参数，必须其他方式设置过期时间。

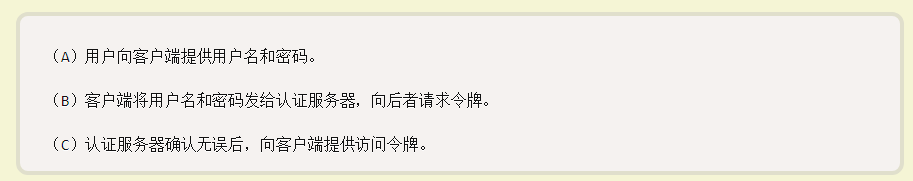
scope：表示权限范围，如果与客户端申请的范围一致，此项可省略。

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。

（3）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，比如客户端是操作系统的一部分，或者由一个著名公司出品。而认证服务器只有在其他授权模式无法执行的情况下，才能考虑使用这种模式。





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。

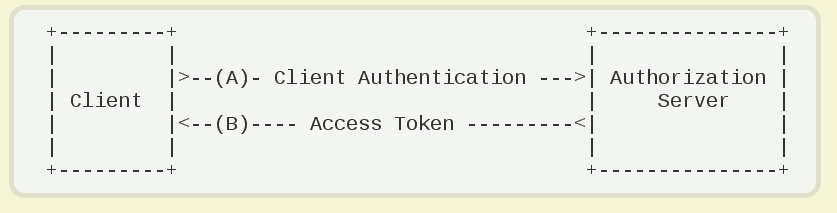
password：表示用户的密码，必选项。

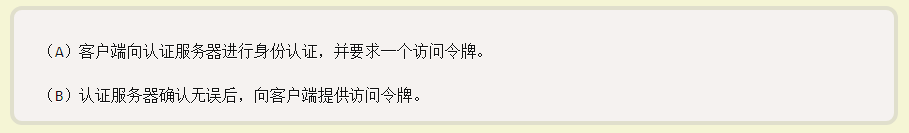
scope：表示权限范围，可选项。

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。

整个过程中，客户端不得保存用户的密码。

（3）客户端模式（Client Credentials Grant）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。

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

**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。