## Cookie

cookie 是一个非常具体的东西，指的就是浏览器里面能永久存储的一种数据，仅仅是浏览器实现的一种数据存储功能。

cookie由服务器生成，发送给浏览器，浏览器把cookie以kv形式保存到某个目录下的文本文件内，下一次请求同一网站时会把该cookie发送给服务器。由于cookie是存在客户端上的，所以浏览器加入了一些限制确保cookie不会被恶意使用，同时不会占据太多磁盘空间，所以每个域的cookie数量是有限的。

## Session

session 从字面上讲，就是会话。这个就类似于你和一个人交谈，你怎么知道当前和你交谈的是张三而不是李四呢？对方肯定有某种特征（长相等）表明他就是张三。

session 也是类似的道理，服务器要知道当前发请求给自己的是谁。为了做这种区分，服务器就要给每个客户端分配不同的“身份标识”，然后客户端每次向服务器发请求的时候，都带上这个“身份标识”，服务器就知道这个请求来自于谁了。至于客户端怎么保存这个“身份标识”，可以有很多种方式，对于浏览器客户端，大家都默认采用 cookie 的方式。

服务器使用session把用户的信息临时保存在了服务器上，用户离开网站后session会被销毁。这种用户信息存储方式相对cookie来说更安全，可是session有一个缺陷：如果web服务器做了负载均衡，那么下一个操作请求到了另一台服务器的时候session会丢失。

## Token

在Web领域基于Token的身份验证随处可见。在大多数使用Web API的互联网公司中，tokens 是多用户下处理认证的最佳方式。

以下几点特性会让你在程序中使用基于Token的身份验证

1. 无状态、可扩展
2. 支持移动设备
3. 跨程序调用
4. 安全

那些使用基于Token的身份验证的大佬们

大部分你见到过的API和Web应用都使用tokens。例如Facebook, Twitter, Google+, GitHub等。

### Token的起源

在介绍基于Token的身份验证的原理与优势之前，不妨先看看之前的认证都是怎么做的。

### 基于服务器的验证

我们都是知道HTTP协议是无状态的，这种无状态意味着程序需要验证每一次请求，从而辨别客户端的身份。

在这之前，程序都是通过在服务端存储的登录信息来辨别请求的。这种方式一般都是通过存储Session来完成。

随着Web，应用程序，已经移动端的兴起，这种验证的方式逐渐暴露出了问题。尤其是在可扩展性方面。

### 基于服务器验证方式暴露的一些问题

1. **Seesion：**每次认证用户发起请求时，服务器需要去创建一个记录来存储信息。当越来越多的用户发请求时，内存的开销也会不断增加。
2. **可扩展性：**在服务端的内存中使用Seesion存储登录信息，伴随而来的是可扩展性问题。
3. **CORS(跨域资源共享)：**当我们需要让数据跨多台移动设备上使用时，跨域资源的共享会是一个让人头疼的问题。在使用Ajax抓取另一个域的资源，就可以会出现禁止请求的情况。
4. **CSRF(跨站请求伪造)：**用户在访问银行网站时，他们很容易受到跨站请求伪造的攻击，并且能够被利用其访问其他的网站。

在这些问题中，可扩展行是最突出的。因此我们有必要去寻求一种更有行之有效的方法。

### 基于Token的验证原理

基于Token的身份验证是无状态的，我们不将用户信息存在服务器或Session中。

这种概念解决了在服务端存储信息时的许多问题

NoSession意味着你的程序可以根据需要去增减机器，而不用去担心用户是否登录。

基于Token的身份验证的过程如下:

1. 用户通过用户名和密码发送请求。
2. 程序验证。
3. 程序返回一个签名的token 给客户端。
4. 客户端储存token,并且每次用于每次发送请求。
5. 服务端验证token并返回数据。

每一次请求都需要token。token应该在HTTP的头部发送从而保证了Http请求无状态。我们同样通过设置服务器属性Access-Control-Allow-Origin:\* ，让服务器能接受到来自所有域的请求。