# Cookie&Session&Token理解

## 发展史

详情：https://www.jianshu.com/p/30bd9c229751

1、以前的web基本就是文档的浏览，不需要记录谁在某一段时间里都浏览了什么文档，每次请求都是一个新的http协议（请求+响应）。

2、随着交互式web应用兴起，如在线购物网站需要登录的网站，必须记住那些人登录系统，哪些人在购物车上放商品。http请求是无状态的，所以想到的方法就是给大家发一个会话标识（session id），每个人收到的都不一样，每次大家向我发起http请求时，把这个字符串一并捎过来，这样服务器就能区分谁谁了。

3、可是这样的问题就是：每个人仅保存自己的sessionId，而服务器保存的是所有人的sessionId！这样服务器存储的数据是一个巨大的开销，另一问题就是服务器集群时的各服务器存储sessionId会不一样。

服务器集群时的sessionId不同问题解决：使用memcached或redis。把sessionId集中存储到一个地方，所有机器都来访问这个地方的数据。这样也带来的另一问题：负责session的机器挂了怎么办？解决：负责session的机器也搞出集群，增加可靠性。

4、**如果服务器不保存sessionId，而是验证客户端发过来的sessionId即可**。这里的关键点就是验证！这样就能解除了sessionId的负担！

例如：小A已登录系统，系统给他发一个令牌（token），里面包含了小A的userId，下次小F再次通过http请求访问系统时，把这个token通过http header带过来。

问题：如何防止别人伪造。答：对数据做一个签名，使用hmac-sha256算法加上密钥，对这个签名和数据一起作为token，别人不知道密钥就无法伪造token了。

这个token 我不保存， 当小F把这个token 给我发过来的时候，我再用同样的HMAC-SHA256 算法和同样的密钥，对数据再计算一次签名， 和token 中的签名做个比较， 如果相同， 我就知道小F已经登录过了，并且可以直接取到小F的user id , 如果不相同， 数据部分肯定被人篡改过， 我就告诉发送者： 对不起，没有认证。

## Cookie介绍

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。

### 概念

基本通信流程：设置cookie-->cookie被自动添加到request的header中-->服务器接收cookie。

在console中输入document.cookie能查看到该页面相关的cookie信息

cookie属性：expires、domain、path、secure、HttpOnly

例如："key=name; expires=Thu, 25 Feb 2016 04:18:00 GMT; domain=ppsc.sankuai.com; path=/; secure; HttpOnly"

-1）expires属性：cookie的失效时间；

-2）domain属性：域名。在t.a.b.com下，指定domain为t.a.b.com、a.b.com、b.com都可行。domain=a.b.com和domain=.a.b.com是完全相同的；

-3）path属性：路径。注：domain和path一起来限制cookie能被哪些URL访问

### Cookie版单点登录

相关链接： https://segmentfault.com/a/1190000004556040

在不同的子域名之间实现单点登录的最佳途径是采用Cookie的Forms认证。他的原理简单的来说，就是让Forms Authentication生成的Cookie能够跨域名访问。

跨域访问：

-1）域A为t1.test.com，域B为t2.test.com

-2）在域A生产一个令域A和域B都能访问的cookie，则该cookie的domain设置为.test.com

-3）在域A生产一个令域A能访问而域B不能访问的cookie，则该cookie的domain设置为t1.test.com

### Cookie与Session区别

-1）Cookie是客户端技术，是由服务器将用户的数据写回给用户浏览器。

-2）Session是服务端技术，是把用户数据写到服务器后，为用户所创建的独享的Session对象中，但Session是基于Cookie的。

服务器在运行时可以为每一个用户的浏览器创建一个其独享的session对象，注意是默认情况下，一个浏览器独占一个session，用户把各自数据存放在各自的session中，当用户再去访问服务器的其他web资源时，其他web资源再从用户各自的session中取出数据为用户服务。

## Session介绍

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。

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。

问1：JESSIONID生成

当你在应用程序调用了Request对象的getSession(boolean create)方法，并且传入的参数是true时，Session才会被创建出来，这样Tomcat才会在Response中添加JESSIONID的这个cookie。

问2：如果我从别人的浏览器中拿到jsessionid，把它放在我自己的请求头中，是不是在服务端对应同一个session，那是不是就可以窃取人家信息了？

是的。虽然敏感的用户信息没有在网络上传输了，但是攻击者拿到sessionid也可以冒充受害者发送请求，这就是为什么我们需要https，加密后攻击者就拿不到sessionid了，另外CSRF也是一种防止session劫持的方式。

问3：spring session与servlet session区别：

spring通过Servlet规范中的Filter机制拦截了所有Servlet请求，偷梁换柱，将标准的Servlet请求对象包装了一下，换成它自己的Request包装类对象，这样当程序员通过包装后的Request对象的getSession方法拿Session时，是通过Spring拿Session，没Web容器什么事了。

## Token介绍

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。

特性会让你在程序中使用基于Token的身份验证：

无状态、可扩展

支持移动设备

跨程序调用

安全

未完成！！！！

https://www.jianshu.com/p/30bd9c229751