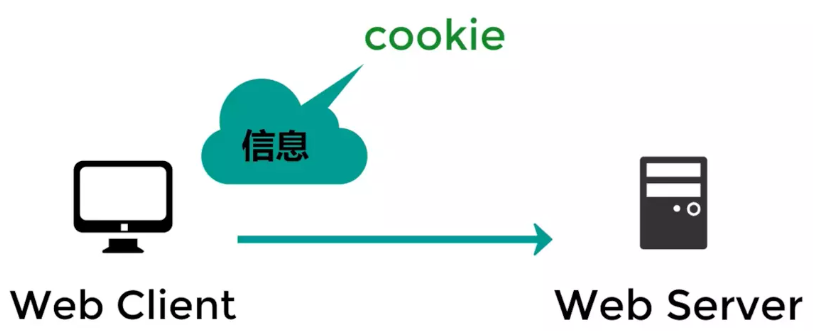
**Cookie和Session的使用和区别**



Cookie中译小甜饼，是网页浏览器用来保存用户信息的文件，可以保存比如用户是谁，购物车有哪些商品等。

Session会话，会话是指我们访问网站的一个周期。

**我们浏览网页使用的HTTP协议是无状态的协议，就是说网页一关闭，浏览器和服务端的连接就会断开，下次打开网页又要重新连接，服务器无法从你一打开的连接上恢复上一次的会话，服务器不知道是你又回来了。**



有了cookie情况就不同了，除非我们之前把你的信息记录在cookie里，在你打开网页和服务器建立连接的时候，把cookie记录的你的信息一起发送给服务器，这样服务器就能从cookie接收到的信息里识别你的身份，让页面为你提供特别属于你的内容。

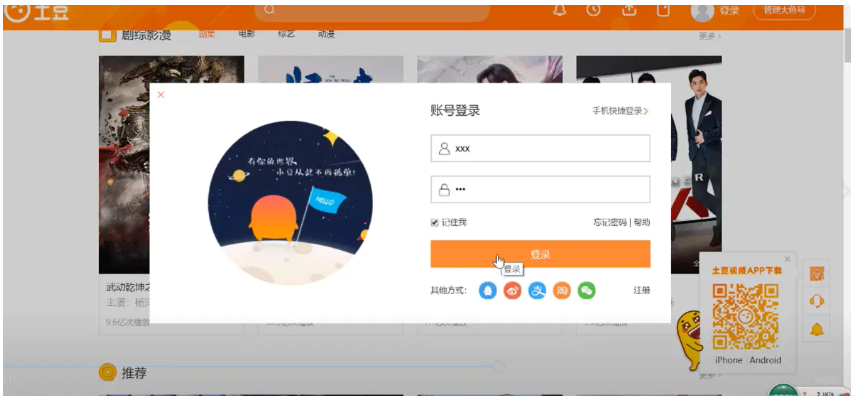


**我们访问浏览器的时候，浏览器会发送一个HTTP请求到服务器端；**

**服务器会发送一个HTTP响应到客户端，其中包括Sst-Cookie，意思就是浏览器建立一个cookie保存服务器指定的内容，比如用户信息和用户操作信息；**

**浏览器保存好信息之后，下次我们再次访问网站的时候，浏览器再发送HTTP请求到服务器端时都会携带之前保存的cookie；**

**服务器端会从收到的cookie中识别用户身份，就能让页面为你提供专门属于你的内容了**。



比如我们从网站的登陆界面中看到有记住我这个选项，你勾选了它以后，登陆成功，浏览器就会把你的信息放在cookie里，下次再访问这个网站的时候，服务器就能根据收到的cookie识别出是你，帮你自动登陆，显示专属于你的内容。

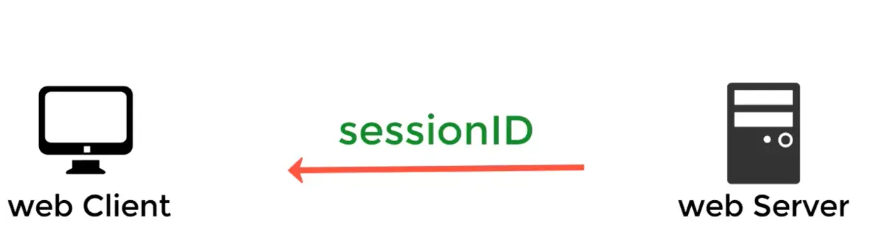
cookie存储的数量和字符数量都有限制，只能存储几十个，不超4096左右个字符。

cookie存储在用户浏览器里也不安全，任何人都能直接查看。

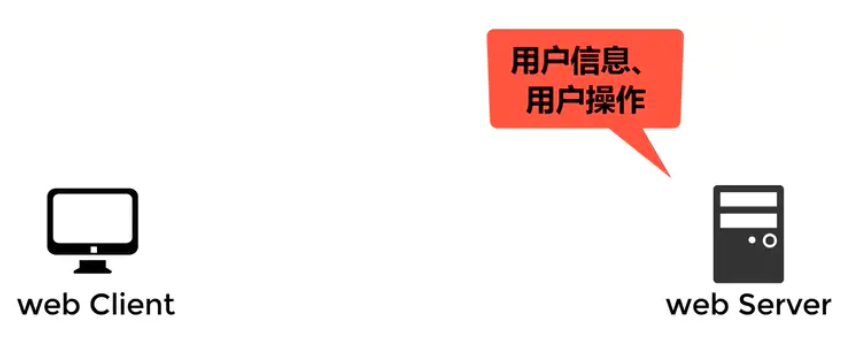
session在计算机网络应用中被称为“会话控制”。



客户端浏览器访问网站的时候



服务器会向客户浏览器发送一个每个用户特有的会话编号sessionID，让他进入到cookie里。



服务器同时也把sessionID和对应的用户信息、用户操作记录在服务器上，这些记录就是session。



客户端浏览器再次访问时，会发送cookie给服务器，其中就包含sessionID。



服务器从cookie里找到sessionID，再根据sessionID找到以前记录的用户信息就可以知道他之前操控些、访问过哪里。

  
session保存在服务器端比较安全，但是可能需要记录千百万用户的信息，对服务器的存储压力很大，所以我们应该有选择的合理使用cookie和session。

---cookie会根据从服务器端发送的响应报文内的一个叫做Set-cookie的首部字段信息，通知客户端保存cookie。当下次客户端再往该服务器发送请求时，客户端会自动在请求报文中加入cookie值后发送出去。服务器端发送客户端发送来的cookie后，会去检查究竟是从哪一个客户端发送来的连接请求，然后对比服务器上的记录，最后得到之前的状态信息。

Cookie: 当你在浏览网站的时候，WEB 服务器会先送一小小资料放在你的计算机上，Cookie 会帮你在网站上所打的文字或是一些选择，都纪录下来。当下次你再光临同一个网站，WEB 服务器会先看看有没有它上次留下的 Cookie 资料，有的话，就会依据 Cookie里的内容来判断使用者，送出特定的网页内容给你。 Cookie 的使用很普遍，许多有提供个人化服务的网站，都是利用 Cookie来辨认使用者，以方便送出使用者量身定做的内容，像是 Web 接口的免费 email 网站，都要用到 Cookie。

**具体来说cookie机制采用的是在客户端保持状态的方案，而session机制采用的是在服务器端保持状态的方案。**

1、什么是session

浏览器访问服务器时，服务器会创建一个 session 对象(该对象有一个唯一的 id, 一般称为 sessionId)。服务器在缺省情况下，会将 sessionId 以 cookie 机制发送给浏览器。当浏览器再次访问服务器时， 会将sessionId 发送给服务器。 服务器依据 sessionId 就可以找到对应的 session 对象。

**cookie 和session 的区别：**

1、cookie数据存放在客户的浏览器上，session数据放在服务器上。

2、cookie不是很安全，别人可以分析存放在本地的COOKIE并进行COOKIE欺骗

考虑到安全应当使用session。

3、session会在一定时间内保存在服务器上。当访问增多，会比较占用你服务器的性能

考虑到减轻服务器性能方面，应当使用COOKIE。

4、单个cookie保存的数据不能超过4K，很多浏览器都限制一个站点最多保存20个cookie。

5、所以个人建议：

将登陆信息等重要信息存放为SESSION

其他信息如果需要保留，可以放在COOKIE中

# web面试中，遇到无数次的问题“无状态的HTTP协议如何提供有状态的连接”？

HTTP是无状态链接，请求-回应完毕后客户端和服务器之间的链接就会断开。那么服务器是如何记住用户的，例如网上购物车，每次新添加购物项，上次添加的购物项还能存在，这是如何实现的？可以推测应该是客户端或服务器端将购物的信息记录下来，创建一个新的链接时候读取记录下来的数据，这样就实现无状态的HTTP连接提供有状态的连接的效果。

**服务器端：**

在网络服务器看来，所有的HTTP请求都独立于之前的请求，也就是说他只识别你这次请求头中包含的信息，然后做出对应的响应。状态管理机制克服了HTTP的一些限制并允许网络客户端和服务器端维护请求间的关系，这种关系维持的期间叫做Session（会话）。Session是服务器端的一种保存上下文信息的机制，它是针对每一个用户的，session的值保存在服务器端。当服务器端程序需要为客户端请求创建一个session的时候，会首先检查请求中是否包含了一个session标识—sessionID，如果包含sessionID，说明以前就已经为该用户创建过session，服务器会直接按照这个sessionID将对应的session检索出来使用（没有检索到可能会新建一个）；如果客户端请求没有包含sessionID，则为这个用户新建一个session并生成一个相关联的sessionID（sessionID的值应该是一个不会重复，也不容易找到规律的字符串），这个sessionID将在本次响应中返回给客户端保存。

购物车例子：

当你买第一个物品的时候点击“购买”，浏览器将请求发送到服务器（该请求中没有sessionID，因为这是你要买的第一个物品，之前没有和服务器交互），服务器发现没有sessionID，就会为客户端创建一个session和与之相关联的sessionID，并把刚创建的这个sessionID在本次响应中返回给客户端，客户端的浏览器接收到返回的响应把这个sessionID存储，当你再次向购物车添加物品时候，浏览器会在后台自动将你存储的sessionID发送给服务器，服务器按照这个sessionID检索到对应的session，然后进行处理返回给你页面，这样你才能够看到上次添加的购物项。

**客户端：**

在客户端用来存储服务器返回的sessionID和其他一些信息的称为cookie。正统的cookie分发是通过扩展HTTP协议来实现的，服务器通过在HTTP的响应头中加上一行特殊的指示以提示浏览器按照指示生成相应的cookie，在交互的时候浏览器检查所有存储的cookie，如果某个cookie所声明的作用范围大于等于将要请求的资源所在的位置，则把该cookie附在请求资源的HTTP请求头上发送给服务器。可以使用HEAD命令得到服务器的响应头，有的网站响应头中包含set-cookie项（例如hi.baidu.com），该项中包含一个名值对，过期时间，路径和域，这些都是由服务器指定。路径就是跟在域后面的url路径，路径和域组成了cookie的作用范围的大小；设置过期时间，就会将cookie保存到硬盘上，关闭浏览器再打开，cookie依然有效，直到请求的时间超过了指定的时间。如果不设置过期时间，则cookie一般不存储在硬盘上而是保存在浏览器的内存里，当然这种行为并不是规范规定的。所以cookie有两种，一种是存储到本地的文本文件之中的，叫做persistent （持久化的）cookie；另一种是并不存储在本地硬盘之中，而是放在浏览器的内存里面，我叫做session cookie。Persistent cookie可以在不同的浏览器进程间共享，比如两个IE窗口。而对于session cookie，不同的浏览器有不同的处理方式，有的浏览器可以跨窗口使用比如Firefox，有的不能比如IE但是IE允许一个窗口生成的窗口（子窗口）共享session cookie。

推测：如果客户端禁用cookie，那么将无法向购物车中添加物品。

禁用IE的cookie，访问当当网，选中一本书，点击购买，发现返回的页面为空白页面，并没有我购买的书；允许使用cookie，购买返回的页面中购买的书已被添加到购物车之中。说明推测是正确的。可是如果禁用浏览器的cookie就无法访问需要创建session的服务器了吗？由于cookie可以被人为的禁止，必须有其他机制以便在cookie被禁止时仍然能够把session id传递回服务器。经常被使用的一种技术叫做URL重写，就是把session id 直接附加在URL路径的后面，这样服务器就可以解析url地址后面的sessionID进行处理。发送SessionID有两种方式，一种就是将sessionID存放在cookie里面浏览器自动发送；另一种是当客户端禁用cookie时，通过URL重写将sessionID发送给服务器。

**Session的生命周期：**

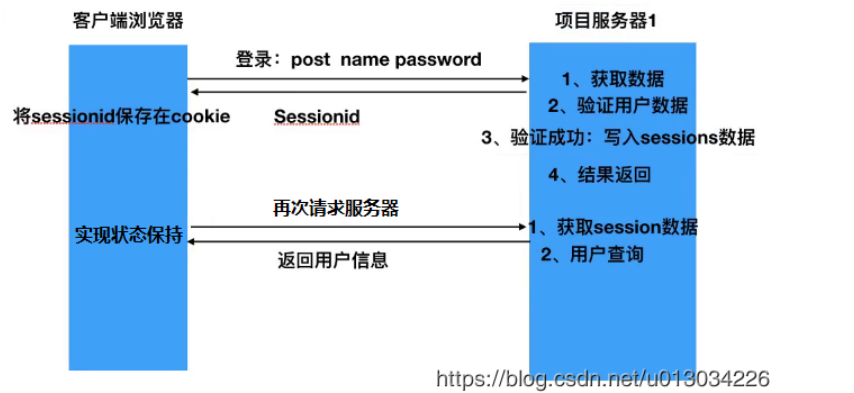
Session什么时候会被服务器删除，有人可能会认为当客户端的浏览器窗口关闭时候session就会被删除，其实不然。事实上除非程序通知服务器删除一个session，否则服务器会一直保留。浏览器关闭之前不会通知服务器，所以服务器不会知道浏览器什么时候关闭，之所以有“浏览器关闭，session就消失”这样的感觉，是因为大多数的session机制都是用会话 cookie（session cookie）来保存sessionID，这样浏览器关闭之后sessionID就消失了，再次连接服务器时候，分配了新的sessionID，也就不能找到以前那个session了，但是如果用persistent cookie来保存sessionID的话，再打开新的浏览器也能够找到那个sessionID加到请求之中，服务器就能够找到那个session。由于浏览器的关闭或者浏览器的进程被强行杀死或者断网，服务器都无法知道，所以服务器设置了一个失效时间，只要距离客户端上次访问的时间超过了失效时间，则认为客户端活动停止了，此时删除session节省存储空间。

例子：登录电子邮箱，长时间（超过失效时间）没有操作，再对邮箱操作的话页面刷新之后提示你页面已经过期，此时session已经被删除。

**总结：**HTTP是无状态链接，因为要实现有状态链接的效果，所以采用session和cookie两种机制。客户端请求需要包含服务器指定的唯一的sessionID，如果请求中没有sessionID，服务器将为该请求创建一个session和与之相关联的sessionID并把这个sessionID放入响应之中，并加上一行特殊的指示以提示浏览器按照指示生成相应的cookie，将响应返回给客户端，客户端的浏览器按照规则解析响应，如果浏览器支持cookie则将sessionID和其他信息存储起来（persistent cookie 或者session cookie），下次再次访问同一服务器时候浏览器将对应的cookie和请求头一起发送给服务器，服务器按照sessionID进行检索找到对应的session进行处理。

**一、session的状态保持及弊端**

当用户第一次通过浏览器使用用户名和密码访问服务器时，服务器会验证用户数据，验证成功后在服务器端写入session数据，向客户端浏览器返回sessionid，浏览器将sessionid保存在cookie中，当用户再次访问服务器时，会携带sessionid，服务器会拿着sessionid从数据库获取session数据，然后进行用户信息查询，查询到，就会将查询到的用户信息返回，从而实现状态保持。



弊端：

1、服务器压力增大

通常session是存储在内存中的，每个用户通过认证之后都会将session数据保存在服务器的内存中，而当用户量增大时，服务器的压力增大。、

2、CSRF跨站伪造请求攻击

session是基于cookie进行用户识别的, cookie如果被截获，用户就会很容易受到跨站请求伪造的攻击。

3、扩展性不强

如果将来搭建了多个服务器，虽然每个服务器都执行的是同样的业务逻辑，但是session数据是保存在内存中的（不是共享的），用户第一次访问的是服务器1，当用户再次请求时可能访问的是另外一台服务器2，服务器2获取不到session信息，就判定用户没有登陆过。

# 二、token认证机制

**token与session的不同主要在①认证成功后，会对当前用户数据进行加密，生成一个加密字符串token，返还给客户端（服务器端并不进行保存）**

②浏览器会将接收到的token值存储在Local Storage中，（通过js代码写入Local Storage，通过js获取，并不会像cookie一样自动携带）

③再次访问时服务器端对token值的处理：服务器对浏览器传来的token值进行解密，解密完成后进行用户数据的查询，如果查询成功，则通过认证，实现状态保持，所以，即时有了多台服务器，服务器也只是做了token的解密和用户数据的查询，它不需要在服务端去保留用户的认证信息或者会话信息，这就意味着基于token认证机制的应用不需要去考虑用户在哪一台服务器登录了，这就为应用的扩展提供了便利，解决了session扩展性的弊端。

