会话（Session）跟踪是Web程序中常用的技术，用来**跟踪用户的整个会话**。常用的会话跟踪技术是Cookie与Session。**Cookie通过在客户端记录信息确定用户身份**，**Session通过在服务器端记录信息确定用户身份**。

**1.1  Cookie机制**

在程序中，会话跟踪是很重要的事情。理论上，**一个用户的所有请求操作都应该属于同一个会话**，而另一个用户的所有请求操作则应该属于另一个会话，二者不能混淆。例如，用户A在超市购买的任何商品都应该放在A的购物车内，不论是用户A什么时间购买的，这都是属于同一个会话的，不能放入用户B或用户C的购物车内，这不属于同一个会话。

而Web应用程序是使用HTTP协议传输数据的。**HTTP协议是无状态的协议。一旦数据交换完毕，客户端与服务器端的连接就会关闭，再次交换数据需要建立新的连接。这就意味着服务器无法从连接上跟踪会话**。即用户A购买了一件商品放入购物车内，当再次购买商品时服务器已经无法判断该购买行为是属于用户A的会话还是用户B的会话了。要跟踪该会话，必须引入一种机制。

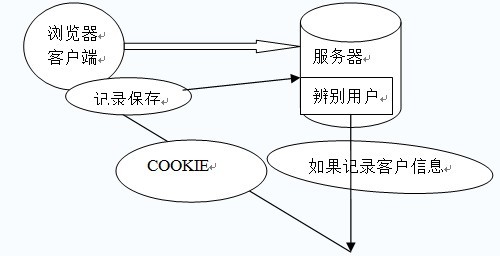
Cookie就是这样的一种机制。它可以弥补HTTP协议无状态的不足。在Session出现之前，基本上所有的网站都采用Cookie来跟踪会话。

**1.1.1  什么是Cookie**

Cookie意为“甜饼”，是**由W3C组织提出**，最早由Netscape社区发展的一种机制。目前Cookie已经成为标准，所有的主流浏览器如IE、Netscape、Firefox、Opera等都支持Cookie。

由于HTTP是一种无状态的协议，服务器单从网络连接上无从知道客户身份。怎么办呢？就**给客户端们颁发一个通行证吧，每人一个，无论谁访问都必须携带自己通行证。这样服务器就能从通行证上确认客户身份了。这就是Cookie的工作原理**。

Cookie实际上是一小段的文本信息。客户端请求服务器，如果服务器需要记录该用户状态，就使用response向客 户端浏览器颁发一个Cookie。客户端浏览器会把Cookie保存起来。当浏览器再请求该网站时，浏览器把请求的网址连同该Cookie一同提交给服务 器。服务器检查该Cookie，以此来辨认用户状态。服务器还可以根据需要修改Cookie的内容。



**1.2 Cookie 属性**

Cookie常用属性

|  |  |  |
| --- | --- | --- |
| 属 性 名 | | 描  述 |
| String name | | 该Cookie的名称。Cookie一旦创建，名称便不可更改 |
| Object value | | 该Cookie的值。如果值为Unicode字符，需要为字符编码。如果值为二进制数据，则需要使用BASE64编码 |
| **int maxAge** | | **该Cookie失效的时间，单位秒。如果为正数，则该Cookie在maxAge秒之后失效。如果为负数，该Cookie为临时Cookie，关闭浏览器即失效，浏览器也不会以任何形式保存该Cookie。如果为0，表示删除该Cookie。默认为–1** |
| boolean secure | | 该Cookie是否仅被使用安全协议传输。安全协议。安全协议有HTTPS，SSL等，在网络上传输数据之前先将数据加密。默认为false |
| String path | | 该Cookie的使用路径。如果设置为“/sessionWeb/”，则只有contextPath为“/sessionWeb”的程序可以访问该Cookie。如果设置为“/”，则本域名下contextPath都可以访问该Cookie。注意最后一个字符必须为“/” |
| String domain | 可以访问该Cookie的域名。如果设置为“.google.com”，则所有以“ggoogle.com”结尾的域名都可以访问该Cookie。注意第一个字符必须为 | |
| String comment | 该Cookie的用处说明。浏览器显示Cookie信息的时候显示该说明 | |
| int version | | 该Cookie使用的版本号。0表示遵循Netscape的Cookie规范，1表示遵循W3C的RFC 2109规范 |

对于cookie有的说明是这样的

除了基础的键值对外，cookie还有下面的属性：

Expires ：cookie最长有效期

Max-Age：在 cookie 失效之前需要经过的秒数。（当Expires和Max-Age同时存在时，文档中给出的是已Max-Age为准，可是我自己用Chrome实验的结果是取二者中最长有效期的值）

Domain：指定 cookie 可以送达的主机名。

Path：指定一个 URL 路径，这个路径必须出现在要请求的资源的路径中才可以发送 Cookie 首部

Secure：一个带有安全属性的 cookie 只有在请求使用SSL和HTTPS协议的时候才会被发送到服务器。

HttpOnly:设置了 HttpOnly 属性的 cookie 不能使用 JavaScript 经由 Document.cookie 属性、XMLHttpRequest 和 Request APIs 进行访问，以防范跨站脚本攻击（XSS）。

secure保证cookie必须安全加密全输，httponly保证js脚本不能访问。

**1.2.1  Cookie的不可跨域名性**

很多网站都会使用Cookie。例如，Google会向客户端颁发Cookie，Baidu也会向客户端颁发Cookie。那浏览器访问Google会不会也携带上Baidu颁发的Cookie呢？或者Google能不能修改Baidu颁发的Cookie呢？

答案是否定的。**Cookie具有不可跨域名性**。根据Cookie规范，浏览器访问Google只会携带Google的Cookie，而不会携带Baidu的Cookie。Google也只能操作Google的Cookie，而不能操作Baidu的Cookie。

Cookie在客户端是由浏览器来管理的。浏览器能够保证Google只会操作Google的Cookie而不会操作 Baidu的Cookie，从而保证用户的隐私安全。浏览器判断一个网站是否能操作另一个网站Cookie的依据是域名。Google与Baidu的域名 不一样，因此Google不能操作Baidu的Cookie。

需要注意的是，虽然网站images.google.com与网站www.google.com同属于Google，但是域名不一样，二者同样不能互相操作彼此的Cookie。

**1.1.4  Unicode编码：保存中文**

中文与英文字符不同，**中文属于Unicode字符，在内存中占4个字符，而英文属于ASCII字符，内存中只占2个字节**。Cookie中使用Unicode字符时需要对Unicode字符进行编码，否则会乱码。

提示：Cookie中保存中文只能编码。一般使用UTF-8编码即可。不推荐使用GBK等中文编码，因为浏览器不一定支持，而且JavaScript也不支持GBK编码。

**1.1.5  BASE64编码：保存二进制图片**

Cookie不仅可以使用ASCII字符与Unicode字符，还可以使用二进制数据。例如在Cookie中使用数字证书，提供安全度。使用二进制数据时也需要进行编码。

%注意：本程序仅用于展示Cookie中可以存储二进制内容，并不实用。由于浏览器每次请求服务器都会携带Cookie，因此Cookie内容不宜过多，否则影响速度。Cookie的内容应该少而精。

**1.1.7  Cookie的有效期**

Cookie的maxAge决定着Cookie的有效期，单位为秒（Second）。Cookie中通过getMaxAge()方法与setMaxAge(int maxAge)方法来读写maxAge属性。

如果maxAge属性为正数，则表示该Cookie会在maxAge秒之后自动失效。浏览器会将maxAge为正数的 Cookie持久化，即写到对应的Cookie文件中。无论客户关闭了浏览器还是电脑，只要还在maxAge秒之前，登录网站时该Cookie仍然有效。

如果maxAge为负数，则表示该Cookie仅在本浏览器窗口以及本窗口打开的子窗口内有效，关闭窗口后该 Cookie即失效。maxAge为负数的Cookie，为临时性Cookie，不会被持久化，不会被写到Cookie文件中。Cookie信息保存在浏 览器内存中，因此关闭浏览器该Cookie就消失了。Cookie默认的maxAge值为–1。

如果maxAge为0，则表示删除该Cookie。Cookie机制没有提供删除Cookie的方法，因此通过设置该Cookie即时失效实现删除Cookie的效果。失效的Cookie会被浏览器从Cookie文件或者内存中删除，

要想修改Cookie只能使用一个同名的Cookie来覆盖原来的Cookie，达到修改的目的。删除时只需要把maxAge修改为0即可。

注意：从客户端读取Cookie时，包括maxAge在内的其他属性都是不可读的，也不会被提交。浏览器提交Cookie时只会提交name与value属性。maxAge属性只被浏览器用来判断Cookie是否过期。

**1.1.8  Cookie的修改、删除**

Cookie并不提供修改、删除操作。如果要修改某个Cookie，只需要新建一个同名的Cookie，添加到response中覆盖原来的Cookie。

如果要删除某个Cookie，只需要新建一个同名的Cookie，并将maxAge设置为0，并添加到response中覆盖原来的Cookie。注意是0而不是负数。负数代表其他的意义。读者可以通过上例的程序进行验证，设置不同的属性。

注意：修改、删除Cookie时，新建的Cookie除value、maxAge之外的所有属性，例如name、path、domain等，都要与原Cookie完全一样。否则，浏览器将视为两个不同的Cookie不予覆盖，导致修改、删除失败。

**1.1.9  Cookie的域名**

Cookie是不可跨域名的。域名www.google.com颁发的Cookie不会被提交到域名www.baidu.com去。这是由Cookie的隐私安全机制决定的。隐私安全机制能够禁止网站非法获取其他网站的Cookie。

正常情况下，同一个一级域名下的两个二级域名如www.helloweenvsfei.com和 images.helloweenvsfei.com也不能交互使用Cookie，因为二者的域名并不严格相同。如果想所有 helloweenvsfei.com名下的二级域名都可以使用该Cookie，需要设置Cookie的domain参数.

注意：domain参数必须以点(".")开始。另外，name相同但domain不同的两个Cookie是两个不同的Cookie。如果想要两个域名完全不同的网站共有Cookie，可以生成两个Cookie，domain属性分别为两个域名，输出到客户端。

**1.1.10  Cookie的路径**

domain属性决定运行访问Cookie的域名，而path属性决定允许访问Cookie的路径（ContextPath）。

设置为“/”时允许所有路径使用Cookie。path属性需要使用符号“/”结尾。name相同但domain相同的两个Cookie也是两个不同的Cookie。

注意：页面只能获取它属于的Path的Cookie。例如/session/test/a.jsp不能获取到路径为/session/abc/的Cookie。使用时一定要注意。

**1.1.11  Cookie的安全属性**

HTTP协议不仅是无状态的，而且是不安全的。使用HTTP协议的数据不经过任何加密就直接在网络上传播，有被截获的可 能。使用HTTP协议传输很机密的内容是一种隐患。如果不希望Cookie在HTTP等非安全协议中传输，可以设置Cookie的secure属性为 true。浏览器只会在HTTPS和SSL等安全协议中传输此类Cookie。

提示：secure属性并不能对Cookie内容加密，因而不能保证绝对的安全性。如果需要高安全性，**需要在程序中对Cookie内容加密、解密，以防泄密。**

Cookie的写法，每个属性以 ; 隔开：

window.document.cookie = name + '=' + encodeURI(value) + '; ' + (hour ? ('expires=' + expire.toGMTString() + '; ') : '') + ('path=' + path);