**什么是 Cookie**

“cookie 是存储于访问者的计算机中的变量。每当同一台计算机通过浏览器请求某个页面时，就会发送这个 cookie。你可以使用 JavaScript 来创建和取回 cookie 的值（document.cookie）。” - w3school  
　　cookie 是访问过的网站创建的文件，用于存储浏览信息，例如个人信息。HTTP是无状态的，通过使用cookie技术来维护应用的状态。Web服务器接到请求时能读出cookie的值，根据cookie值的内容就可以判断和恢复一些用户的信息状态。

从JavaScript的角度看，cookie 就是一些字符串信息。这些信息存放在客户端的计算机中，用于客户端计算机与服务器之间传递信息。JavaScript / PHP（服务端的语言）都可以存取 cookie。

Cookie 与网站关联，而不是与特定的页面关联。因此，无论用户请求站点中的哪一个页面，浏览器和服务器都将交换 Cookie 信息。用户访问不同站点时，各个站点都可能会向用户的浏览器发送一个 Cookie；浏览器会分别存储所有 Cookie。

使用 Cookie 能够达到多种目的，所有这些目的都是为了帮助网站记住用户。例如，一个实施民意测验的站点可以简单地将 Cookie 作为一个 Boolean 值，用它来指示用户的浏览器是否已参与了投票，这样用户便无法进行第二次投票。要求用户登录的站点则可以通过 Cookie 来记录用户已经登录，这样用户就不必每次都输入凭据。

在请求一个页面的Http头中，会将属于此页面的本地Cookies信息加在Http头中。

**Cookie 基础知识**

cookie 是有大小限制的，每个 cookie 所存放的数据不能超过4kb，如果 cookie 字符串的长度超过4kb，则该属性将返回空字符串。浏览器还限制站点可以在用户计算机上存储的 Cookie 的数量。

由于 cookie 最终都是以文件形式存放在客户端计算机中，所以查看和修改 cookie 都是很方便的，所以cookie 不能存放重要信息。

每个 cookie 的格式都是这样的：<cookie名>=<值>；名称和值都必须是合法的标示符。alert(typeof document.cookie) 结果是 string。

cookie 有**域**domain**和路径**routing这个概念。因为浏览器是个注意安全的环境，所以不同的域之间是不能互相访问 cookie 的(当然可以通过特殊设置的达到 cookie 跨域访问)。路径：一个网页所创建的 cookie 只能被与这个网页在同一目录或子目录下的所有网页访问，而不能被其他目录下得的页访问。

其实创建cookie的方式和定义变量的方式有些相似，都需要使用 cookie 名称和 cookie 值。同个网站可以创建多个 cookie ，而多个 cookie 可以存放在同一个cookie 文件中。

**Cookie的有效期**

在asp.net中还可以通过设置HttpCookie对象的过期时间为DateTime.MinValue来指定此Cookies为跟随浏览器生效。

cookie 是存在有效期的。在默认情况下，一个 cookie 的生命周期就是在浏览器关闭的时候结束。如果想要 cookie 能在浏览器关掉之后还可以使用，就必须要为该 cookie 设置有效期，也就是 cookie 的失效日期。① 临时性质的cookie。当前使用的过程中网站会储存一些你的个人信息，当浏览器关闭后这些信息也会从计算机中删除；② 设置失效时间的cookie。就算浏览器关闭了，这些信息业依然会在计算机中。如登录名称和密码，这样无须在每次到特定站点时都进行登录。这种cookie 可在计算机中保留几天、几个月甚至几年。

**Cookie安全性**

通常 cookie 信息都是使用HTTP连接传递数据，这种传递方式很容易被查看，所以 cookie 存储的信息容易被窃取。假如 cookie 中所传递的内容比较重要，那么就要求使用加密的数据传输。

　　所以 cookie 的这个属性的名称是“secure”，默认的值为空。如果一个 cookie 的属性为secure，那么它与服务器之间就通过HTTPS或者其它安全协议传递数据。语法如下：

　　document.cookie = "username=Darren;secure"

　　把cookie设置为secure，只保证 cookie 与服务器之间的数据传输过程加密，而保存在本地的 cookie文件并不加密。如果想让本地cookie也加密，得自己加密数据。

**Cookie常见问题**

cookie 存在两种类型：① 你浏览的当前网站本身设置的 cookie； ② 来自在网页上嵌入广告或图片等其他域来源的第三方 cookie (网站可通过使用这些 cookie 跟踪你的使用信息)

* cookie 有两种清除方式：① 通过浏览器工具清除 cookie (有第三方的工具，浏览器自身也有这种功能)；② 通过设置 cookie 的有效期来清除 cookie。注：删除 cookie 有时可能导致某些网页无法正常运行。
* 浏览器可以通过设置来接受和拒绝访问 cookie。
* 出于功能和性能的原因考虑，建议尽量降低 cookie 的使用数量，并且要尽量使用小 cookie。
* 关于cookie编码的细节问题将会在cookie高级篇中单独介绍。
* 假如是本地磁盘中的页面，chrome的控制台是无法用JavaScript读写操作 cookie 的，解决办法...换一个浏览器^\_^。

**Cookie基础用法**

**一.简单的存取操作**

　　在使用JavaScript存取 cookie 时，必须要使用Document对象的 cookie 属性；一行代码介绍如何创建和修改一个 cookie ：

　　document.cookie = 'username=Darren'

　　以上代码中'username'表示 cookie 名称，'Darren'表示这个名称对应的值。假设 cookie 名称并不存在，那么就是创建一个新的 cookie；如果存在就是修改了这个 cookie 名称对应的值。如果要多次创建 cookie，重复使用这个方法即可。

**二. cookie的读取操作**

　　要精确的对 cookie 进行读取其实很简单，就是对字符串进行操作。从w3school上copy这段代码来做分析：

function getCookie(c\_name){  
 if (document.cookie.length>0){　　//先查询cookie是否为空，为空就return ""  
 c\_start=document.cookie.indexOf(c\_name + "=")　　//通过String对象的indexOf()来检查这个cookie是否存在，不存在就为 -1　　  
　　if (c\_start!=-1){   
 c\_start=c\_start + c\_name.length+1　　//最后这个+1其实就是表示"="号啦，这样就获取到了cookie值的开始位置  
　　　c\_end=document.cookie.indexOf(";",c\_start)　　//其实我刚看见indexOf()第二个参数的时候猛然有点晕，后来想起来表示指定的开始索引的位置...这句是为了得到值的结束位置。因为需要考虑是否是最后一项，所以通过";"号是否存在来判断  
　　　if (c\_end==-1) c\_end=document.cookie.length　　  
　 return unescape(document.cookie.substring(c\_start,c\_end))　　//通过substring()得到了值。想了解unescape()得先知道escape()是做什么的，都是很重要的基础，想了解的可以搜索下，在文章结尾处也会进行讲解cookie编码细节  
　　　　　　}   
　　　　}  
　　　　return ""  
　　}

　　当然想实现读取cookie的方法还有不少，比如数组，正则等，这里就不往细说了。

下面是封装了的专门用于操作Cookies的jQuery工具函数。

方法签名： jQuery.cookie(name, subName, value,  options)

方法说明：读取、写入、删除Cookies

方法参数：

|  |  |  |
| --- | --- | --- |
| 名称 | 说明 | 举例 |
| name | cookies的主键值 | 读取主键：  $.cookie("singleKey")  写入cookies，值为字符串：  $.cookie("singleKey", "", "singleKey-value", { expires: 1, path: "/", secure: false }) |
| subName | 子键名称。在写入时请传递空或者null | 读取子键：  $.cookie("multiKey", "subName1")  写入cookies，值为对象：  var subNameObj = { subName1: "aaa", subName2: "bbb", subName3: "ccc" };  $.cookie("multiKey", "", subNameObj, { expires: 1, path: "/", secure: false }); |
| value | Cookies值，可以是字符串或者对象。  如果是对象，则将对象的每个属性保存在Cookies子键。 | 参见上面实例。 |
| options | 参数：  expires：可以是数字或者Data类型的对象。  如果传入数字表示几天后过期。  path：路径，默认为域名根目录（“/”）。  secure：是否启用加密，默认为否。 | 指定过期时间：  var myDate = new Date();  myDate.setFullYear(2009, 10, 10);  $.cookie("singleKey", "", "singleKey-value", { expires: myDate, secure: false })   1天后过期：  var time = Date();  $.cookie("singleKey", "", "singleKey-value", { expires: 1, path: "/", secure: false }) |

**三.设置cookie的有效期**

　　文章中常常出现的 cookie 的生命周期也就是有效期和失效期，即 cookie 的存在时间。在默认的情况下，cookie 会在浏览器关闭的时候自动清除，但是我们可以通过expires来设置 cookie 的有效期。语法如下：

　　document.cookie = "name=value;expires=date"

　　上面代码中的date值为GMT(格林威治时间)格式的日期型字符串，生成方式如下：

　　var \_date = new Date();  
　　\_date.setDate(\_date.getDate()+30);  
　　\_date.toGMTString();

　　上面三行代码分解为几步来看：

* 通过new生成一个Date的实例，得到当前的时间；
* getDate()方法得到当前本地月份中的某一天，接着加上30就是我希望这个cookie能过在本地保存30天；
* 接着通过setDate()方法来设置时间；
* 最后用toGMTString()方法把Date对象转换为字符串，并返回结果

　　通过下面这个完整的函数来说明在创建 cookie 的过程中我们需要注意的地方 - 从w3school复制下来的。创建一个在 cookie 中存储信息的函数：

[IMG_258](http://www.cnblogs.com/Darren_code/archive/2011/11/24/javascript:void(0);" \o "复制代码)

1 　function setCookie(c\_name, value, expiredays){  
2 　　　　var exdate=new Date();  
3 　　　　exdate.setDate(exdate.getDate() + expiredays);  
4 　　　　document.cookie=c\_name+ "=" + escape(value) + ((expiredays==null) ? "" : ";expires="+exdate.toGMTString());  
5 　　}  
6 　　使用方法：setCookie('username','Darren',30)

[IMG_259](http://www.cnblogs.com/Darren_code/archive/2011/11/24/javascript:void(0);" \o "复制代码)

　　现在我们这个函数是按照天数来设置cookie的有效时间，如果想以其他单位（如：小时）来设置，那么改变第三行代码即可：

　　exdate.setHours(exdate.getHours() + expiredays);

　　这样设置以后的cookie有效期就是按照小时为单位的。

　　常见问题中有提到清除 cookie 的两种方法，现在要说的是使 cookie 失效，通过把有效期的时间设置为一个已过期的时间。既然已经有了设置有效期的方法，那么设置失效期的方法就请感兴趣的朋友自己动手了^\_^。下面继续比较深的cookie话题。

**Cookie 高级篇**

**一.cookie 路径概念**

　　在基础知识中有提到 cookie 有域和路径的概念，现在来介绍路径在 cookie 中的作用。

　　cookie 一般都是由于用户访问页面而被创建的，可是并不是只有在创建 cookie 的页面才可以访问这个 cookie。

　　默认情况下，只有与创建 cookie 的页面在同一个目录或子目录下的网页才可以访问，这个是因为安全方面的考虑，造成不是所有页面都可以随意访问其他页面创建的 cookie。举个例子：

　　在 "http://www.cnblogs.com/Darren\_code/" 这个页面创建一个cookie，那么在"/Darren\_code/"这个路径下的页面如： "http://www.cnblogs.com/Darren\_code/archive/2011/11/07/Cookie.html"这个页面默认就能取到cookie信息。

　　可在默认情况下， "http://www.cnblogs.com"或者 "http://www.cnblogs.com/xxxx/" 就不可以访问这个 cookie（光看没用，实践出真理^\_^）。

　　那么如何让这个 cookie 能被其他目录或者父级的目录访问类，通过设置 cookie 的路径就可以实现。例子如下：

　　document.cookie = "name=value;path=path"  
　　document.cookie = "name=value;expires=date;path=path"

 　　红色字体path就是 cookie 的路径，最常用的例子就是让 cookie 在跟目录下,这样不管是哪个子页面创建的 cookie，所有的页面都可以访问到了:

　　document.cookie = "name=Darren;path=/"

**二.cookie 域概念**

　　路径能解决在同一个域下访问 cookie 的问题，咱们接着说 cookie 实现**同域**之间访问的问题。语法如下：

　　document.cookie = "name=value;path=path;domain=domain"

　　红色的domain就是设置的 cookie 域的值。

　　例如 "www.qq.com" 与 "sports.qq.com" 公用一个关联的域名"qq.com"，我们如果想让 "sports.qq.com" 下的cookie被 "www.qq.com" 访问，我们就需要用到 cookie 的domain属性，并且需要把path属性设置为 "/"。例：

　　document.cookie = "username=Darren;path=/;domain=qq.com"

　　注：一定的是同域之间的访问，不能把domain的值设置成非主域的域名。

**三.cookie 安全性**

　　通常 cookie 信息都是使用HTTP连接传递数据，这种传递方式很容易被查看，所以 cookie 存储的信息容易被窃取。假如 cookie 中所传递的内容比较重要，那么就要求使用加密的数据传输。

　　所以 cookie 的这个属性的名称是“secure”，默认的值为空。如果一个 cookie 的属性为secure，那么它与服务器之间就通过HTTPS或者其它安全协议传递数据。语法如下：

　　document.cookie = "username=Darren;secure"

　　把cookie设置为secure，只保证 cookie 与服务器之间的数据传输过程加密，而保存在本地的 cookie文件并不加密。如果想让本地cookie也加密，得自己加密数据。

　　注：就算设置了secure 属性也并不代表他人不能看到你机器本地保存的 cookie 信息，所以说到底，别把重要信息放cookie就对了，囧...

**四.cookie 编码细节**

　　原本来想在常见问题那段介绍cookie编码的知识，因为如果对这个不了解的话编码问题确实是一个坑，所以还是详细说说。

　　在输入cookie信息时不能包含空格，分号，逗号等特殊符号，而在一般情况下，cookie 信息的存储都是采用未编码的方式。所以，在设置 cookie 信息以前要先使用escape()函数将 cookie 值信息进行编码，在获取到 cookie 值得时候再使用unescape()函数把值进行转换回来。如设置cookie时：

　　document.cookie = name + "="+ escape (value)

　　再看看基础用法时提到过的getCookie()内的一句：

　　return unescape(document.cookie.substring(c\_start,c\_end))

　　这样就不用担心因为在cookie值中出现了特殊符号而导致 cookie 信息出错了。