**Cookie存储**

HTTP无状态协议是指协议对于事务处理没有记忆能力。

**会话跟踪**

协议的状态是指下一次传输可以“记住”这次传输信息的能力，无状态是指同一个会话（注意什么叫同一个会话）的连续两个请求互相不了解，当浏览器发送请求给服务器的时候，服务器响应，但是同一个浏览器再发送请求给服务器的时候，他会响应，但是他不知道你就是刚才那个浏览器，每一次请求和响应都是相对独立的。

Tip:如果浏览器完全禁止Cookie功能的网站上有很多功能将不能被执行

**Cookie由来**

Cookie是由[服务器端](http://baike.baidu.com/view/7072073.htm)生成，发送给User-Agent（一般是浏览器），浏览器会将Cookie的key/value保存到某个目录下的[文本文件](http://baike.baidu.com/view/40622.htm)内，下次请求同一网站时就发送该Cookie给[服务器](http://baike.baidu.com/view/899.htm)（前提是浏览器设置为启用cookie）。Cookie名称和值可以由服务器端开发自己定义，对于JSP而言也可以直接写入jsessionid，这样服务器可以知道该用户是否是合法用户以及是否需要重新登录等，服务器可以设置或读取Cookies中包含信息，借此维护用户跟服务器会话中的状态。

**主要用途**

服务器可以利用Cookies包含信息的任意性来筛选并经常性维护这些信息，以判断在HTTP传输中的状态。Cookies最典型的应用是判定注册用户是否已经登录网站，用户可能会得到提示，是否在下一次进入此网站时保留用户信息以便简化登录手续，这些都是Cookies的功用。另一个重要应用场合是“购物车”之类处理。用户可能会在一段时间内在同一家网站的不同页面中选择不同的商品，这些信息都会写入Cookies，以便在最后付款时提取信息。

**使用和禁用Cookie**

用户可以改变浏览器的设置，以使用或者[禁用](http://baike.baidu.com/view/657791.htm)Cookies。

[**微软**](http://baike.baidu.com/view/2353.htm)[Internet Explorer](http://baike.baidu.com/view/85144.htm)

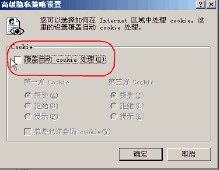
工具 > Internet选项 > 隐私页

调节滑块或者点击“高级”，进行设置.

[**Mozilla Firefox**](http://baike.baidu.com/view/193667.htm)

工具>[选项](http://baike.baidu.com/view/740366.htm)>[隐私](http://baike.baidu.com/view/350273.htm)

（注： 在[Linux版本](http://baike.baidu.com/view/897468.htm)中，是如下操作：编辑 > 首选项 > 隐私 ， 而Mac则是：Firefox > 属性 > 隐私）

[](http://baike.baidu.com/pic/cookie/1119/0/7ac8805184ae496042a75bc6?fr=lemma&ct=single)查看源网页

设置Cookies选项

设定阻止/允许的各个域内Cookie

查看Cookies管理窗口，检查现存Cookie信息，选择删除或者阻止它们

**苹果计算机 Safari**

Safari > 预置 > 安全标签

选择以下的选项

总是 接受 Cookies

永不 接受 Cookies

接受 Cookies 仅从您浏览的站点 （例如，不接受来自其它站点的广告) 预设的选项.

什么是Cookies（“小甜饼”）呢？简单来说，Cookies就是服务器暂时存放在你的电脑里的资料（.txt格式的文本文件），好让服务器用来辨认你的计算机。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，Cookies 会把你在网站上所打的文字或是一些选择都记录下来。当下次你再访问同一个网站，Web服务器会先看看有没有它上次留下的Cookies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。

cookies有什么作用呢？许多网站上都有新用户注册这一项，有时注册了一下，等到下次再访问该站点时，会[自动识别](http://baike.baidu.com/view/1070528.htm)到你，并且向你问好，是不是觉得很亲切？当然这种作用只是表面现象，更重要的是，网站可以利用cookies跟踪统计用户访问该网站的习惯，比如什么时间访问，访问了哪些页面，在每个网页的停留时间等。利用这些信息，一方面是可以为用户提供个性化的服务，另一方面，也可以作为了解所有用户行为的工具，对于网站经营策略的改进有一定参考价值。例如，你在某家航空公司站点查阅航班时刻表，该网站可能就创建了包含你旅行计划的Cookies，也可能它只记录了你在该站点上曾经访问过的Web页，在你下次访问时，网站根据你的情况对显示的内容进行调整，将你所感兴趣的内容放在前列。这是高级的Cookie应用。目前Cookies 最广泛的是记录用户登录信息，这样下次访问时可以不需要输入自己的用户名、密码了——当然这种方便也存在用户信息泄密的问题，尤其在多个用户共用一台电脑时很容易出现这样的问题。

另外，有人认为网站利用cookies可能存在侵犯用户隐私的问题，但由于大多用户对此了解不多，而且这种对用户个人信息的利用多数作为统计数据之用，不一定造成用户的直接损失，因此对于cookies与用户隐私权的问题并没有相关法律约束，很多网站仍然在利用cookie跟踪用户行为，有些程序要求用户必须开启cookie才能正常应用。IE浏览器用户可以通过“隐私”选项中的隐私设置的高低来决定是否允许网站利用cookie跟踪自己的信息，从全部限制到全部允许，或者限制部分网站，也可以通过手动方式对具体的网站设置允许或者禁止使用cookies进行编辑。IE浏览器的默认设置是 “中级”－对部分网站利用cookie有限制。个人电脑的cookies设置（对IE浏览器而言）可通过菜单“工具－Internet选项－隐私”来查看和修改。

脚本攻击

尽管cookie没有病毒那么危险，但它仍包含了一些敏感信息：用户名，计算机名，使用的浏览器和曾经访问的网站。用户不希望这些内容泄漏出去，尤其是当其中还包含有私人信息的时候。

这并非危言耸听，一种名为跨站点脚本攻击（Cross site scripting）可以达到此目的。通常跨站点脚本攻击往往利用网站漏洞在网站页面中植入脚本代码或网站页面引用第三方脚本代码，均存在跨站点脚本攻击的可能，在受到跨站点脚本攻击时，脚本指令将会读取当前站点的所有 Cookie 内容（已不存在 Cookie 作用域限制），然后通过某种方式将 Cookie 内容提交到指定的服务器（如：AJAX）。一旦 Cookie 落入攻击者手中，它将会重现其价值。

建议开发人员在向客户端 Cookie 输出敏感的内容时（譬如：该内容能识别用户身份）：

1）设置该 Cookie 不能被脚本读取，这样在一定程度上解决上述问题。  
　　2）对 Cookie 内容进行加密，在加密前嵌入时间戳，保证每次加密后的密文都不一样（并且可以防止消息重放）。  
　　3）客户端请求时，每次或定时更新 Cookie 内容（即：基于第2小条，重新加密）  
　　4）每次向 Cookie 写入时间戳，数据库需要记录最后一次时间戳（防止 Cookie 篡改，或重放攻击）。  
　　5）客户端提交 Cookie 时，先解密然后校验时间戳，时间戳若小于数据数据库中记录，即意味发生攻击。

基于上述建议，即使 Cookie 被窃取，却因 Cookie 被随机更新，且内容无规律性，攻击者无法加以利用。另外利用了时间戳另一大好处就是防止 Cookie 篡改或重放。

Cookie 窃取：搜集用户cookie并发给攻击者的黑客。攻击者将利用cookie信息通过合法手段进入用户帐户。

Cookie 篡改：利用安全机制，攻击者加入代码从而改写 Cookie 内容，以便持续攻击。

**Javascript使用cookie**

假设有这样一种情况，在某个用例流程中，由A页面跳至B页面，若在A页面中采用JS用变量temp保存了某一变量的值，在B页面的时候，同样需要使用JS来引用temp的变量值，对于JS中的全局变量或者静态变量的生命周期是有限的，当发生页面跳转或者页面关闭的时候，这些变量的值会重新载入，即没有达到保存的效果。解决这个问题的最好的方案是采用cookie来保存该变量的值，那么如何来设置和读取cookie呢？

首先需要稍微了解一下cookie的结构，简单地说：cookie是以键值对的形式保存的，即key=value的格式。各个cookie之间一般是以“;”分隔

**JS读取cookie:**

假设cookie中存储的内容为：name=jack;password=123

则在B页面中获取变量username的值的JS代码如下：

var username=document.cookie.split(";")[0].split("=")[1];

//JS操作cookies方法!

//写cookies

function setCookie(name,value)

{

var Days = 30;

var exp = new Date();

exp.setTime(exp.getTime() + Days\*24\*60\*60\*1000);

document.cookie = name + "="+ escape (value) + ";expires=" + exp.toGMTString();

}

**读取cookies**

function getCookie(name)

{

var arr,reg=new RegExp("(^| )"+name+"=([^;]\*)(;|$)");

if(arr=document.cookie.match(reg))

return unescape(arr[2]);

else

return null;

}

**删除cookies**

function delCookie(name)

{

var exp = new Date();

exp.setTime(exp.getTime() - 1);

var cval=getCookie(name);

if(cval!=null)

document.cookie= name + "="+cval+";expires="+exp.toGMTString();

}

//使用示例

setCookie("name","hayden");

alert(getCookie("name"));

//如果需要设定自定义过期时间

//那么把上面的setCookie　函数换成下面两个函数就ok;

//程序代码

function setCookie(name,value,time)

{

var strsec = getsec(time);

var exp = new Date();

exp.setTime(exp.getTime() + strsec\*1);

document.cookie = name + "="+ escape (value) + ";expires=" + exp.toGMTString();

}

function getsec(str)

{

alert(str);

var str1=str.substring(1,str.length)\*1;

var str2=str.substring(0,1);

if (str2=="s")

{

return str1\*1000;

}

else if (str2=="h")

{

return str1\*60\*60\*1000;

}

else if (str2=="d")

{

return str1\*24\*60\*60\*1000;

}

}

//这是有设定过期时间的使用示例：

//s20是代表20秒

//h是指小时，如12小时则是：h12

//d是天数，30天则：d30

setCookie("name","hayden","s20");

* 添加一个cookie

|  |
| --- |
| document.cookie = "username=123"; //一次写入一个键值对  一次写入多个，只有第一个生效，或者采用&符号链接  user=尹涛&password=123&name=xxxx; |

* + 保存形式：key=value
  + 取出显示cookie的类型，取出cookie的值

|  |
| --- |
| alert(typeof document.cookie);  alert(document.cookie); |

* + 中文编码问题，使用encodeURIComponent('xxxxx')编码，再使用decodeURIComponent(document.cookie)解码，能解决中文乱码问题

Var \_chinese=window.encodeURIComponent("中文"));  
alert(\_chinese)；alert(window.decodeURIComponent(\_chinese));

* cookie失效时间
  + 如果不加这个参数，默认情况下为临时cookie，也就是关闭浏览器则cookie失效
  + expires=date，指定在是时间，cookie被自动清理
  + 如果手动调整系统的时间，让cookie消失，可以进行恢复，此时为假删除
  + 当覆盖某个键值对，并设置失效时间过期时，则cookie被删除，不能恢复
* cookie的完整存储形式
  + 键=值;expires=失效时间;domain=域名访问;
* cookie的路径问题
  + 在同一路径下的网页可以共享cookie，路径不同时，不能访问
* cookie的域名问题
  + 必须在绑定域名的服务器上才可以设置域名
  + 并且只能设置绑定的域名，也就是说，不同服务器间的cookie文件不共享
* cookie的限制
  + Chrome和Safari没有对cookie的个数做限制，一般浏览器限制同一域名数量为50个
  + cookie文件的总大小一般为4KB（同一个域名）
* 封装cookie的增删改查
  + 添加一个cookie
  + 获取一个cookie
  + 删除cookie
* 案例：
  + 拖拽的效果保存
  + 用户密码的保存
  + 购物车