# 一、Cookie定义

起源

使用普通的可执行程序的时候，能够读、写、修改程序并运行计算机上的多种文件。这些可执行程序几乎能够以所有方式操作硬盘上任意位置的文件(当然在要求权限的操作系统中运行这些程序时需要合适的用户权限)。这种操作方式通常并不会出现什么问题，这是因为这些程序是由著名公司或用户开发的，它们都会以信誉保证这种访问方式不会出现不可预计的问题。但是在Web环境中上面说的这种保证就不复存在了。因为能够访问网络、会使用文本编辑器的任何人都可能编写出具有破坏性的危险程序来。比如在浏览网站的时候经常被挂马，从而中了马。

正是由于上述原因，在Web页面中使用的绝大多数编程语言都不允许编程者使用储存在客户端计算机上的文件。各种浏览器也都提供了一组规则，阻止发生上述情况。

浏览器是使用HTTP协议访问Web页面。HTTP是一种无状态协议，也就是说在浏览器和服务器之间完成一次会话后，就丢弃了这个连接。浏览器和服务器都不会保存这次会话之间的状态。但是随着互联网的发展，如今可以使用购物车购买各种物品、订购各种东西，国内较出名的有易趣等等。而对于像易趣这样的商家，记住客人的购买习惯、爱好、注册号、用户名就成为一种非常迫切的需要。

1994年，Netscape公司引入了cookie的概念。Cookie是保存在客户端计算机，也就是运行在浏览器的计算机中的一个短小文本文件，它用于Web站点在用户的计算机上保存信息并且以后再取回它，这些信息通常采用“名-值”对儿的方式存储。Cookie通常用来记录对一个Web站点的访问。比如，在易趣购买商品，为了方便网上采购，cookie就可能保存所选择的商品，以及你的姓名、电子邮件地址等个人识别信息。为了帮助Web站点跟踪每一个来访人员，cookie通常包含一个唯一的标识。当然为你创建cookie的Web站点有责任告诉你cookie中存储了什么信息，以及这些信息如何被站点所使用。

Cookie简介

我们可以在自己电脑的硬盘上找到存储的cookie文件，对于使用windows操作系统使用IE浏览器的朋友通常可以在C:\Documents and Settings\用户名\Cookies目录中找到。如果不是这个目录可以利用windows自带的搜索功能搜索关键字cookies也可找到。在这个文件夹里的每个cookie文件都是一个简单而又普通的文本文件。我们可以通过文件名，可以看到是哪个Web站点在你的计算机中放置了cookie文件。

例如：访问自己电脑的C:\Documents and Settings\Administrator\Cookies目录中的cookie文件，以百度搜索引擎的cookie文件来做说明，其文件名为administrator@baidu[1].txt，内容如下所示：

BAIDUID

9D9DE43F143ABD236DB39C258D84A211

baidu.com/

1536

3368053120

32039905

1482168528

29835532

\*

利用cookie，可以使Web网页更加人性化，更加友好，它能够存储用户使用的语言、阅读习惯、网站访问次数、跟踪购物车中的物品等等。但是cookie带来了一写安全的问题，Cookie是Web世界中的一部分，但不是一定需要。可以根据自己的需要控制是否使用cookie。如果不喜欢cookie，那么可以关闭浏览器的cookie功能，并删除浏览器中的所有cookie。

Web浏览器中的cookie只占据少量的存储空间，一般来说，浏览器中最多存储300个cookie。每个cookie占据的存储空间不超过4K字节，因此存储不了多少信息。Cookie文件的实际文件名和存储位置与操作系统和浏览器有关。Windows下的Netscape Navigator将cookie保持在Navigator系统目录下的cookies.txt文件中，IE将cookie存储在C:\Documents and Settings\用户名\Cookies目录中，cookie的文件名则与用户名、网站名有关，但都是文本文件。

作用原理

下面通过举例说明cookie数据的流动过程，即cookie的创建、改变等等。假设目前正在访问http://www.google.com 网站。

(1)、在浏览器的地址栏内输入网址http://www.google.com，浏览器从本地计算机上寻找google网站设置的cookie文件。如果找到了google的cookie文件，浏览器会把文件文件中的所有“名-值”对同先前的URL一同发给google服务器。如果没有找到，就不发送cookie数据，而只发送URL。

(2)、google服务器接受cookie数据和对网页的请求。如果存在“名-值”对，google将使用它。

(3)、如果服务器没有收到“名-值”对，google知道在此之前没有访问过它的站点，服务器会创建一个新ID并放进google的数据库中，然后把“名-值”对放在要传回网页的头信息里，并传递给你。这样你的浏览器将在硬盘上的cookie文件中保存“名-值”对。

(4)、以后每当再次访问网站时，网站服务器就会改变“名-值”对或增加新的“名-值”对。除此之外，服务器回随着“名-值”对发送一些其他的信息，主要包括生存期和路径，这样网站借此把不同的cookie值与不同的网站部位关联起来。

# 二、Cookie的组成

语法格式

Cookie通常由服务器端的程序通过HTTP请求和响应头发送给浏览器，但是利用javascript，可以在本地浏览器中设置Cookie好处是免除了服务器端程序处理Cookie的工作，从而降低服务器的负担。Cookie在默认的情况下其生命期是当前会话范围，当前会话结束后，相应的Cookie也就被删除了。

Cookie是由“名-值”对儿组成的文本。一个cookie字符串中最多能够存储20个“名-值”对儿。对于每一个HTML文档来说，浏览器只存储一个cookie。在构造cookie时，制定的“名-值”对儿的name=value称为属性。它必须使用分号(;)作为结束符。在组成cookie的字符串中，不允许使用分号、逗号以及空白符HTTP Set-Cookie头的格式为：

*Set-Cookie:name=value;[expires=date];[path=pathname];[domain=domainna*

*me];[secure];*

其中，方括号表示该属性可以省略，name=value属性可以包含多个。例如：

*Set-Cookie:id= "曾云好";expires=Wed,28-Fer-07 15:13:00 GMT;domain=*

*"google.com";path= "/";secure;*

属性

**(1)、name=value**

这个属性是指定cookie数据项的名称。一般比较常见的名称包括会话标识用户名称等，可以指定喜欢的任何标识符作为该名称。该属性的语法格式*nameofcookie=value;*

其中nameofcookie是cookie的数据项的名称，value是该名称对应的值，是一个字符串，它可以是由一系列字符组成，但不能含有分号(;)、逗号(,)和空白符。如果在nameofcookie或value中包含这些字符，那么需要使用URL风格的%XX格式进行编码。名称属性是Set-Cookie头要求的必选属性。

下面是名称属性的几个应用示例：

id=123456789

email=zengyunhao0@tom.com

name=demon

**(2)、expires=date**

这是一个可选属性，它指定cookie的过期日期。一般情况下，给date一个接近会话时间的日期时间值，这样，在浏览器结束当前会话时，cookie就会过期，从而删除cookie。但是可以通过给expires属性指定一个较长的时间值，从而长久地保留该cookie。其格式为：

*expires=weekday,DD-MON-YY HH:MM:SS GMT*

其中，weekday为星期几，比如Monday、Tuesday、Wednesday等，也可以用缩写的星期几，比如Mon、Tues、Wed等；DD-MON-YY指定日期，分别为日、月、年；HH:MM:SS指定时间，分别为时、分、秒；GMT表示该时间为格林威治标准时间，这里的日期时间总是使用GMT。开发人员通过指定expires属性控制cookie的过期日期，一旦cookie已经过期，就会被自动删除。

下面是expires属性的几个示例：

*expires=Friday,14-Jan-07 15:50:00 GMT*

*expires=Mon, 14-Jan-07 15:50:00 GMT*

**(3)、path=pathname**

这是一个可选属性，指定cookie可用于特定的服务器中的什么位置。在cookie中指定路径可以同一个域中的多个页面共享cookie。例如：

*path=\home*

*path=\aa\bb\int*

**(4)、domain=domainname**

这是一个可选属性，它指定是否所作用的域名后，可以在同一个域的多个服务器之间共享cookie，而无需为每一个服务器都指定cookie。Domainname为域名，可以使用完整的URL格式http://domain，也可以使用省略http://的简写方式。例如：

*domain=www.google.com*

*domain=http://www.google.com*

**(5)、secure**

这是一个可选属性，它指定是否通过安全通道传递cookie。如果设定了该属性，那么cookie必须使用HTTPS协议进行传递。其语法格式为：

*secure*

注意：HTTPS协议是由Netscape开发并内置于其浏览器中，用于对数据进行压缩和解压的操作，并返回网络上传送回的结果。HTTPS实际上应用了Netscape的安全套接字层（SSL）作为HTTP应用层的子层。HTTPS使用端口443，而不是象HTTP那样使用端口80来和TCP/IP进行通信。SSL使用40位关键字作为RC4流加密算法，这对于商业信息的加密是合适的。HTTPS和SSL支持使用X.509数字认证，如果需要的话用户可以确认发送者是谁。

# 三、Cookie的使用

下面使用javascript来创建、查看、删除Cookie。

不过需要注意的是，在设置Cookie时，它们首先保存在浏览器的内存中，退出浏览器时，才被写入到硬盘中。

Cookie对象

在处理Cookie时，JavaScript将它保存为document对象的一个属性，其属性名称是cookie。利用这个属性，可以创建和读取Cookie数据。在启动浏览器并打开某个页面后，如果存在相应的Cookie，那么它被装入到document对象的cookie属性中。Document.cookie属性采用name=value对方式保存各个cookie数据项值。在document.cookie属性中不能够看到其他cookie属性的设定值，包括过期日期、路径、安全性等。在程序中可以使用alert(document.cookie)来调用当前页面的Cookie数据项值。

创建Cookie

在JavaScript中创建Cookie的方法很简单：只要为document.cookie属性指定name=value对即可。不过需要注意的是，在为document.cookie属性赋值时，把一个字符串赋值给它，所以使用引号时需要特别注意，不要弄错了，赋值字符串的格式为：

*name=value;[expires=date];[path=path];[domain=somewhere.com];[secure]*

**例**：

*document.cookie="id="+form1.cookie.value+";expires="+expiration\_date+";path=/";*

再次值得注意的是，在设置name=value属性时，name和value中不能包含分号、逗号以及空白符。如果需要在这些元素中包含这些字符或者汉字，那么必须对他们进行编码。

JavaScript的escape()函数提供了编码字符串的功能，它将字符串中所有非字母字符转换为用字母数字表示的等价字符串。万花谷病毒就用到了escape()函数进行字符处理，把某些字符、汉字等变成乱码以达到迷惑人的目的。这种编码方式应用的非常广泛，比如当使用google搜索“黑客”时，浏览器向服务器发送字符串q=%E9%BB%91%E5%AE%A2，其中%E9%BB%91%E5%AE%A2就是汉字“黑客”经过编码后的字符串。这种编码称为URI编码。使用unescape()函数能够把URI编码的字符串转换为正常的字符串，万花谷病毒也是利用unescape()函数将程序解码后在本地机器上运行。

不过目前在javascript1.5标准中，提供了函数encodeURI()和decodeURI()来取代escape()和unescape()。这四个函数的功能、语法如下

|  |  |
| --- | --- |
| 函数(语法格式) | 功能说明 |
| escape(s) | 返回字符串S编码后的结果，编码时，将非字母函数以及空白字符使用十六进制转义序列来表示。该函数已经被encodeURI(uri)取代 |
| unescape() | 解码escape(s)函数编码的字符串，返回字符串原来的值。该函数已经被decodeURI(uri)取代 |
| encodeURI(uri) | 返回字符串uri编码的结果，将非字母函数以及空白字符使用十六进制转义序列来表示。但不编码字符#、？和@，这些字符用于分隔URL中各组成部分。 |
| decodeURI(uri) | 解码由encodeURI(uri)函数编码的字符，返回字符串原来的值。 |

对于网络上的一些黑客游戏，在过关中至少有一关用到了加密技术，对网页中的部分代码进行了加密，其实用这四个函数实现就可以对代码的加解密。下面通过示例来演示encodeURI()和decodeURI()的编码(加解密)功能。代码如下：

<html><head><title>encodeURI()和decodeURI()的编码(加解密)功能</title>

</head><center><h2>URL编码和解码 </h2>

<script language="JavaScript">

<!--

function urlEncoding(form){

var myString = form.input.value;

alert(encodeURI(myString)); //实现编码效果

}

function urlDecoding(form){

var myString = form.input.value;

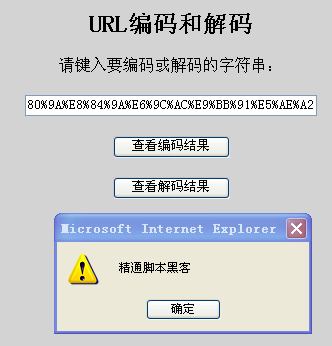
alert(decodeURI(myString)); //实现解码效果

}

// -->

</script>

<body bgcoloR="lightgrey" >

<form name="form1">

请键入要编码或解码的字符串：<p>

<input type="text" name="input" size=40><p>

<input type="button"

value="查看编码结果"

onClick="urlEncoding(this.form);"> <p>

<input type="button"

value="查看解码结果"

onClick="urlDecoding(this.form);"><p>

</form>

</center>

</body>

</html>

代码运行了之后，首先演示编码的效果，输入关键字“精通脚本黑客”后，点击按钮“查看编码结果”后就出现了如图1的效果，其编码后的字符是%E7%B2%BE%E9%80%9A%E8%84%9A%E6%9C%AC%E9%BB%91%E5%AE%A2。然后再把%E7%B2%BE%E9%80%9A%E8%84%9A%E6%9C%AC%E9%BB%91%E5%AE%A2放入输入框中，点击按钮“查看解码结果”就出现了如图2的效果。

接下来演示如何创建一个Cookie。代码如下：

<html><head><title>创建Cookie演示 </title>

<script language="JavaScript">

<!--

function makeCookie(form){

var when = new Date();

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

//getTime()函数是取得服务器的当前时间,24 \* 60 \* 60 \* 1000是取得一天的//毫秒数,这里是功能是时间增加24小时

when.setFullYear(when.getFullYear() + 1);

//getFullYear()返回本地时间的年份值,加1表示年度增加一年

yname=form.yourname.value; //取得输入的用户名

ypasswd=form.passwd.value; //取得输入的密码

//开始创建cookie

document.cookie=encodeURI("name")+"="+encodeURI(yname)+";expires="+when.toGMTString();

document.cookie=encodeURI("passwd")+"="+encodeURI(ypasswd)+";expires="+when.toGMTString();

//显示cookie

alert(document.cookie);

}

function welcome(myForm){

you=myForm.yourname.value;

var position=document.cookie.indexOf("name=");

//indexof()方法是返回"name="第一次出现子字符串的字符位置

if ( position != -1){

var begin = position + 5;

var end=document.cookie.indexOf(";", begin);

if(end == -1){ end=document.cookie.length;}

you= decodeURI(document.cookie.substring(begin, end));

//substring()方法是返回位于begin和end指定的字符串

str=you;

alert("欢迎你！" + you);

}

else{alert("嘿嘿，还没有Cookie");}

}

// -->

</script>

</head>

<body bgcoloR="lightgrey" onLoad="document.form1.reset()" >

<center>

<h2>用户登录</h2>

<form name="form1">

用户名：

<input type="text" name="yourname" >

<p>

密码：

<input type="text" name="passwd" >

<p>

<input type="button" value="创建Cookie"

onClick="makeCookie(this.form);">

<p>

<input type="button"

value="查看Cookie" onClick="welcome(this.form);">

<p>

</form>

</body>

</html>

上面的代码是在cookie中创建两个cookie数据项，一个名称为name，用于保存用户名；另一个名称为passwd，用于保存用户密码，并将他们的过期日期设置为1年加1天。在document.cookie属性中会看到这两个name=value数据对。

代码运行之后，在用户名中输入“曾云好”，在密码中输入“123456”，单击“创建cookie”按钮，程序创建cookie，并显示如图1所示的对话框；单击“查看cookie”按钮，显示如图2所示的欢迎对话框。

读取Cookie

记录cookie的目的就是为了使用cookie，了解用户访问网站或网页的情况。在读取cookie的时候，只能够读取网页所在服务器生成的cookie，而不能读取其他服务器生成的cookie。cookie的访问控制由浏览器决定。

在下面的示例中，读取上面示例所生成的cookie，并显示在一个单独网页中，代码如下：

<html><head><title>读取Cookie</title>

<script language="JavaScript">

<!--

function seeCookie(){

if(document.cookie == ""){

document.write("没有定义Cookie！");

}

else{

var myCookie = document.cookie.split("; ");

document.write("<body> <font face=宋体>");

document.write("<br><b>解码前的结果为：<br>");

for(var i=0;i<myCookie.length; i++){

document.write("<b>Cookie: " + myCookie[i] + "<br>");

}

document.write("<br><br><b>解码后的结果为：<br>");

for(var i=0;i<myCookie.length; i++){

document.write("<b>Cookie: " + decodeURI(myCookie[i]) + "<br>");

}

}

}

// -->

</script>

</head>

<body bgcoloR="lightgrey" onLoad="document.myForm.reset()" >

<center>

<h2> 读取Cookie</h2>

<form name="myForm">

请单击下面的按钮，了解当前Cookie的值。

<p>

<input type="button" value="读取Cookie" onClick="seeCookie();">

<p>

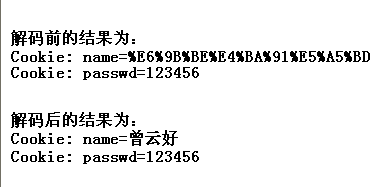
</form>

</center>

</body>

</html>

该代码执行后，显示如图1所示的页面，单击“读取cookie”按钮后，显示之前示例所定义的cookie值解码前和解码后的结果，如图2所示。



对于Cookie欺骗漏洞的缘由：

一是cookie文件保存在本地计算机上，造成了用户可以修改其内容，所以这样做就已经存在安全隐患了，

二是程序员在写程序的时候对情况考虑不周全，从而导致前面存在的安全隐患变成了安全问题了。