**Cookie**

JavaScript 可以使用 document.cookie 属性来创建 、读取、及删除 cookie。

JavaScript 中，创建 cookie 如下所示：

document.cookie="username=John Doe";

您还可以为 cookie 添加一个过期时间（以 UTC 或 GMT 时间）。

默认情况下，cookie 在浏览器关闭时删除：

document.cookie="username=John Doe; expires=Thu, 18 Dec 2043 12:00:00 GMT";

可以使用 path 参数告诉浏览器 cookie 的路径。默认情况下，cookie 属于当前页面。

document.cookie="username=John Doe; expires=Thu, 18 Dec 2043 12:00:00 GMT; path=/";

如果您设置了新的 cookie，旧的 cookie 不会被覆盖。 新 cookie 将添加到 document.cookie 中，所以如果您重新读取document.cookie，您将获得如下所示的数据：

cookie1=value; cookie2=value;

**优点：**

1.分担服务器存储的负担

2.通过加密和安全传输技术（SSL），减少cookie被破解的可能性。

3.只在cookie中存放不敏感数据，即使被盗也不会有重大损失。

4.控制cookie的生命期，使之不会永远有效。偷盗者很可能拿到一个过期的cookie。

**缺点：**

1. Cookie数量和长度的限制。每个特定的域名下最多生成20个cookie，每个cookie长度不能超过4KB，否则会被截掉.
2. 可能被删除，被禁用；
3. Cookie的大小是受限的，并且每次你请求一个新的页面的时候Cookie都会被发送过去，这样无形中浪费了带宽，**另外cookie还需要指定作用域，不可以跨域调用**。
4. 安全性问题。如果cookie被人拦截了，就暴露了所有的session信息。即使加密也与事无补，因为拦截者并不需要知道cookie的意义，他只要原样转发cookie就可以达到目的了。
5. 不同浏览器不相通，不同域中的cookie不共享；。

**如何删除一个cookie**

1.将时间设为当前时间往前一点。

var date = newDate();

date.setDate(date.getDate() - 1);//真正的删除

setDate()方法用于设置一个月的某一天。

2.expires的设置

document.cookie= 'user='+ encodeURIComponent('name') + ';expires = ' + newDate(0)

**会话cookie和持久cookie的区别:**

如果不设置过期时间，则表示这个cookie生命周期为浏览器会话期间，**只要关闭浏览器窗口，cookie就消失了。**这种生命期为浏览会话期的cookie被称为会话cookie。会话cookie一般不保存在硬盘上而是保存在内存里。

如果设置了过期时间，浏览器就会把cookie保存到硬盘上，这样的cookie称为持久cookie,关闭后再次打开浏览器，这些**(持久)cookie依然有效直到超过设定的过期时间**。

存储在硬盘上的cookie可以在不同的浏览器进程间共享，比如两个IE窗口。而对于保存在内存的cookie，不同的浏览器有不同的处理方式。

**httpOnly**

这个选项用来设置cookie是否能通过 js 去访问。

默认情况下，cookie不会带httpOnly选项(即为空)，**所以默认情况下，客户端是可以通过js代码去访问（包括读取、修改、删除等）这个cookie的。**

当cookie带httpOnly选项时，客户端则无法通过js代码去访问（包括读取、修改、删除等）这个cookie。在客户端是不能通过js代码去设置一个httpOnly类型的cookie的，这种类型的cookie**只能通过服务端来设置**。

之所以限制客户端去访问cookie，主要还是出于**安全的目的**。因为如果任何 cookie 都能被客户端通过**document.cookie**获取，那么假如合法用户的网页受到了XSS攻击，有一段恶意的script脚本插到了网页中，这个script脚本，通过document.cookie读取了用户身份验证相关的 cookie，那么只要原样转发cookie，就可以达到目的了。

**session机制：**

无需设置过期时间，服务器会根据判断是否有session创建新的session对象，并通常会用cookie存储session到客户端。当服务器收到请求需要创建session对象时，首先会检查客户端请求中是否包含sessionid。如果有sessionid，服务器将根据该id返回对应session对象。如果客户端请求中没有sessionid，服务器会创建新的session对象，并把sessionid在本次响应中返回给客户端。

通常使用cookie方式存储sessionid到客户端，在交互中浏览器按照规则将sessionid发送给服务器。如果用户禁用cookie，则要使用URL重写，可以通过response.encodeURL(url) 进行实现；API对encodeURL的结束为，当浏览器支持Cookie时，url不做任何处理；当浏览器不支持Cookie的时候，将会重写URL将SessionID拼接到访问地址后。

Session的缺点：

（1）Session保存的东西越多，就越占用服务器内存，对于用户在线人数较多的网站，服务器的内存压力会比较大；

（2）依赖于cookie(sessionID保存在cookie)，如果禁用cookie，则要使用URL重写，不安全；

（3）创建Session变量有很大的随意性，可随时调用，不需要开发者做精确地处理，所以，过度使用session变量将会导致代码不可读而且不好维护。

**session什么时候被创建**

一个常见的错误是以为session在有客户端访问时就被创建

事实是直到某server端程序(如Servlet)调用HttpServletRequest.getSession(true)这样的语句时才会被创建。

**使session失效的方法：**

1.服务器进程被停止

2.程序调用HttpSession.invalidate()

3.距离上一次收到客户端发送的session\_id时间间隔超过了session的最大有效时间

4.无效的session

**保存session id的几种方式**

A．保存session\_id的方式可以**采用cookie**，这样在交互过程中浏览器可以自动的按照规则把这个标识发送给服务器。

B．由于cookie可以被人为的禁止，必须有其它的机制以便在cookie被禁止时仍然能够把session\_id传递回服务器，经常采用的一种技术叫做**URL重写**，就是把session\_id附加在URL路径的后面，**附加的方式也有两种，一种是作为URL路径的附加信息，另一种是作为查询字符串附加在URL后面。**网络在整个交互过程中始终保持状态，就必须在每个客户端可能请求的路径后面都包含这个session\_id。

C．另一种技术叫做**表单隐藏字段**。就是服务器会自动修改表单，添加一个隐藏字段，以便在表单提交时能够把session\_id传递回服务器。

**URL重写有什么缺点**

①重写需要给很多超链接添加额外的信息,工程量大,可能导致每个页面都需要动态生成

②即使所有的页面都动态生成，如果用户离开会话并通过书签或链接再次回来，会话的信息都会丢失，因为存储下来的链接含有错误的标识信息－该URL后面的session\_id已经过期了。

**使用隐藏的表单域有什么缺点(???)**

仅当每个页面都是有表单提交而动态生成时，才能使用这种方法。单击常规的<a href..>超文本链接并不产生表单提交，因此隐藏的表单域不能支持通常的会话跟踪，只能用于一系列特定的操作中，比如在线商店的结账过程

**getSession()/getSession(true)、getSession(false)的区别**

getSession()/getSession(true)：当session存在时返回该session，否则新建一个session并返回该对象

getSession(false)：当session存在时返回该session，否则不会新建session，返回null

**cookie和session的区别：**

①存在的位置：

cookie 存在于客户端，临时文件夹中；

session存在于服务器的内存中

②安全性:

cookie是以明文的方式存放在客户端的，安全性低，可以通过一个加密算法进行加密后存放；

session存放于服务器的内存中，所以安全性好

③生命周期:

cookie的生命周期是累计的，从创建时，就开始计时，到规定时间后，cookie生命周期结束；

session的生命周期是间隔的，从创建时，开始计时，到规定时间之前如果没有访问session，那么session生命周期被销毁。但是，如果在规定时间内访问过session，那么，将重新计算session的生命周期。

④访问范围:

cookie为多个用户浏览器共享；

session为一个用户浏览器独享

⑤其他:

可以考虑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。

**为什么说session 比cookie更安全？**

真正的cookie存在于客户端硬盘上的一个文本文件，如果两者一样的话，只要cookie就好了，让客户端来分提服务器的负担，并且对于用户来说又是透明的。但实际上不是。

session的sessionID是放在cookie里，要想功破session的话，得分两步：

第一要得到sessionID。攻破cookie后，你要得到sessionID,sessionID是要有人登录，或者启动session\_start才会有，你不知道什么时候会有人登录。

第二取有效sessionID。sessionID是加密的，第二次session\_start的时候，前一次的sessionID就没有用了，session过期时sessionid也会失效，想在短时间内功破加了密的 sessionID很难。session是针对某一次通信而言，会话结束session也就随着消失了。

**简述:**

**①要先攻破cookie,但此时cookie里面不一定有session\_id(别人不一定登陆过)**

**②前一次的session\_id可能因而第二次的session\_start而失效;**

**session\_id可能因为到期而失效;**

**session\_id可能被加了密,很难破解;**