🡪**为什么需要Cookie和Session?**

由于Http协议是无状态的，导致服务器无法分辨是谁浏览了网页。为了维持用户在网站的状态，比如是否登陆、购物车信息等，先后出现了四种技术，分别是隐藏表单域、URL重写、cookie、session。并且在HTML5时候推出了新的WebStorage机：localStorage（本地存储）和sessionStorage（会话存储）.

例如我们想让登陆的用户在发表评论以后的一个小时内不允许再次发送评论。那么首先当用户访问的时候，我们后台知道这个用户没有Session，或者Session中没有登陆标记位，那么要求用户登录。后台返回给用户登录成功信息时候，便在这个SessionID所联系的Session中加入已经登陆的标记位。然后用户进行发表评论，发表评论成功时候，后台会为这个用户的Session添加标记，表明下次发送评价的时间。那么在一个小时内，我们可以做到在后台根据Session中的状态不显示提交评论按钮给用户，或者用户点击提交评论的按钮后，在后台提交的时候检查这个Session状态，返回用户信息提交太频繁不接受。

如果我们没有Session机制，那么只靠HTTP无状态特性，这个功能是无法完成的。

🡪

cookie机制采用的是在客户端保持状态的方案，而session机制采用的是在服务器端保持状态的方案。同时我们也看到，由于采用[服务器端](http://cpro.baidu.com/cpro/ui/uijs.php?adclass=0&app_id=0&c=news&cf=1001&ch=0&di=128&fv=11&is_app=0&jk=f27d78b457762e6d&k=%B7%FE%CE%F1%C6%F7%B6%CB&k0=%B7%FE%CE%F1%C6%F7%B6%CB&kdi0=0&luki=10&n=10&p=baidu&q=65035100_cpr&rb=0&rs=1&seller_id=1&sid=6d2e7657b4787df2&ssp2=1&stid=0&t=tpclicked3_hc&td=1836545&tu=u1836545&u=http:%2F%2Fwww.bubuko.com%2Finfodetail-388671.html&urlid=0" \t "_blank)保持状态的方案在客户端也需要保存一个Session ID[标识](http://cpro.baidu.com/cpro/ui/uijs.php?adclass=0&app_id=0&c=news&cf=1001&ch=0&di=128&fv=11&is_app=0&jk=f27d78b457762e6d&k=%B1%EA%CA%B6&k0=%B1%EA%CA%B6&kdi0=0&luki=8&n=10&p=baidu&q=65035100_cpr&rb=0&rs=1&seller_id=1&sid=6d2e7657b4787df2&ssp2=1&stid=0&t=tpclicked3_hc&td=1836545&tu=u1836545&u=http:%2F%2Fwww.bubuko.com%2Finfodetail-388671.html&urlid=0)，所以session机制可能需要借助于cookie机制，来达到保存标识的目的.如果用户在浏览器上禁用了Cookie,则我们还可以有其他选择去让用户浏览器发送Session ID给服务器，手段有URL重写（在URL末尾加入SessionID），表单隐藏字段。

🡪

cookie是浏览器保存在用户电脑上的一小段文本，用来保存用户在网站上的必要的信息。Web页面或服务器告诉浏览器按照一定的规范存储这些信息，并且在以后的所有请求中，这些信息就会自动加在http请求头中发送给服务器。

通常Cookie的格式是key=name的一串键值对字符串格式。

对于Cookie的过期时间：如果不设置cookie过期时间，cookie会在会话结束后销毁，称为会话cookie。如果想将会话cookie设置为持久cookie，只需设置一下cookie的过期时间。

🡪**Session的产生过程：**

当客户端请求创建一个session的时候，服务器首先检查客户端请求里是否已包含了一个session标识session id，如果已包含一个session id则说明以前已经为此客户端创建过session，服务器就按照session id把这个session检索出来使用。 如果客户端请求不包含session id，则为此服务器创建一个session并且生成一个与此session相关联的session id。session id的值是一个既不会重复，又不容易被找到规律以仿造的字符串，这个session id将被在本次响应中返回给客户端的浏览器保存。接下来在服务器端，Session会被设置一个失效时间，在失效时间之前这个Session会被保存在服务器上。除非手动进行代码删除。

🡪**localStorage以及sessionStorage**

localStorage 用于在客户端持久存储数据，除非主动删除数据，否则数据是永远不会过期的。

sessionStorage 用于在会话期间存储数据，数据存储得地方也在客户端。被存储的数据只有在同一个会话中的页面才能访问，并且当会话结束后数据也会随之销毁。

与Cookie相比，localStorage有以下优势：

1.每个Cookie最多只能存储4KB信息，而官方建议的localStorage的大小为5M。

2.每个域下可创建的Cookie数量有限制（IE6下至多存储20个Cookie），localStorage对数据条数无限制。

3.原生的cookie接口并不友好，程序员需重新封装，localStorage提供了方便的数据操作接口。

4.Cookie的内容会随页面请求在服务器和客户端之间来回传递，占用带宽资源，传输效率低，而localStorage内存储的数据不会被发送到服务器端。

5.Cookie存在安全性问题，如果cookie被人拦截了，那人就可以取得所有的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。

localStorage的局限：

1、各浏览器的localStorage大小不统一，并且IE只在IE8以上版本才支持localStorage。

2、localStorage只能存储字符串，任何格式的数据在存储的时候都会被自动转为字符串，所以读取的时候，需要进行类型的转换。

3、localStorage本质上是对字符串的存储和读取，存储数据会消耗浏览器的内存空间，存储数据多的情况下，会影响浏览器响应效率。

4、localStorage在浏览器的隐私模式下是不可读取的。

5、localStorage不能被爬虫抓取到。

🡪**localStorage以及sessionStorage的数据API接口**

window. sessionStorage和window.localStorage属性返回的数据类型都是Storage，支持HTML5的浏览器都实现了Storage接口，因此这两个属性都可以返回一个Storage对象。该Storage对象可以获取与**当前域**对应的本地存储区域，其中存储的数据可以在当前域的多个页面和会话之间传递。

|  |  |
| --- | --- |
| 方法或属性 | 功能 |
| Storage.length | 返回存储在Storage对象内的 key/value 对的个数 |
| Storage.key(n) | 返回存储在Storage对象内的第 n 条数据的 key 。 |
| Storage.getItem(key) | 返回指定key的value。 |
| Storage.setItem(key,value) | 向Storage对象内添加或更新 key/value。 |
| Storage.removeItem(key) | 从Storage对象内删除指定 key/value。 |
| Storage.clear() | 从Storage对象内删除所有 key/value。 |

P.s Storage对象不但可以用自身的setItem(), getItem()等方法方便存取，还可以像普通的JavaScript对象一样使用点 . 操作符，及以 [] 的方式进行数据存储

Storage对象事件：

当使用的存储区域内容发生变化时（Storage.setItem()、Storage.removeItem() 和Storage.clear()方法都可以触发该事件）还可能触发storage事件，使用window.onstorage可以捕获该事件。例如下面的代码：  
var storage=window.localStorage;

(function(){

window.addEventListener("storage",function storageHandler(event){

var myDiv=document.getElementById("myDiv");

myDiv.innerHTML="存储发生了变化：<br/><b>"+

event.key+"</b>键改变了<br/>旧值<b>"+

event.oldValue+"</b>被改变为新值<b>"+

event.newValue+"</b><br/>发生更改所在的网址：<b>"+

event.url+"</b>";

});

})()

Event事件对象可以获取的属性如下：

（1）属性key，表示被更改键。

（2）属性oldValue，表示被更改的键的旧值。

（3）属性newValue，表示被更改的键的新值。

（4）属性url，表示发生更改所在的网址。

（5）属性storageArea，表示发生更改所在的Storage对象