前端缓存机制

cookie session Web Stroage

cookie

Cookie 是服务器保存在浏览器的一小段文本信息。浏览器每次向服务器发出请求，就会自动附上这段信息，是前端的一种常见而又传统的缓存机制。

作用：
1、用于保存页面信息：如自动登录，记住用户名
2、 以域名为单位，数量大小有4k-10k，有过期时间
3、js中通过doucment.cookie调用

设置：doucument.cookie="name=value;max-age=seconds;Path=path;HttpOnly;"等；一次设置一个值，同名参数会进行覆盖

指定过期时间的cookie会存储在本地，没有指定时为session Cookie，会话结束后cookie删除，设置domain和paath可指定cookie的作用域。httponly指定cookie必须用http或https传输，该属性可限制js访问操作此cookie。secure可指定cookie只能使用https传输。

读取：document.cookie会返回cookie的名和值，其他参数不会返回，如：a=1;b=2;每个cookie的名值通过分号分割。

删除：将max-age设置为-1

4、cookie的传递
从服务器端，发送cookie给客户端，是对应的Set-Cookie头信息。包括了对应的cookie的名称，值，以及各个属性。
从客户端发送cookie给服务器，对应Cookie头信息，不发送cookie的各个属性，只发送对应的名称和值。

cookie优点

可配置性和可扩展性
1、可以在客户端上保存用户数据，起到简单的缓存和用户身份识别等作用
2、 保存用户的偏好，比如网页的字体大小、背景色等等。
3、记录用户的行为

cookie的缺陷

数据数量：每个cookie的大小限制在4k，不同的浏览器对同一个域下的cookie的数量有限制，IE6 20，IE7+ 50，Firefox 50，Safari ∞，Chrome 53。数量超出时， IE和Opera 会清理近期最少使用的cookie，Firefox会随机清理cookie。保险起见cookie的数量应控制在20个，单个cookie大小应小于4KB。

安全性问题：HTTP请求中的cookie是明文传递（HTTPS不是），故敏感信息不能使用cookie存储，如用户密码等。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。

网络负担：cookie会被附加在每个HTTP请求中，在请求和响应时都会被传输，所以增加了流量的损失。

seesion

存储会话机制，保存在服务器上。客户端访问服务器时，服务器把客户端信息以某种形式记录在服务器上。再次访问时只需要从该Session中查找该客户的状态就可以了。

标识用户身份： 用户与服务器建立连接的同时，服务器会自动为其分配一个SessionId，cookie把SessiondId自动带到服务器。

session创建：当程序需要为某个客户端的请求创建一个session时，服务器首先检查这个客户端的请求里是否已包含了sessionId，如果已包含则说明以前已经为此客户端创建过session，服务器就按照sessionId把这个session检索出来使用（检索不到，会新建一个）。如果客户端请求不包含sessionId，则为此客户端创建一个session并且生成一个与此session相关联的sessionId，sessionId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sessionId将被在本次响应中返回给客户端保存。

禁用cookie：如果客户端禁用了cookie，通常有两种方法实现session而不依赖cookie。
1）URL重写，就是把sessionId直接附加在URL路径的后面。
2）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

Session共享：对于多网站(同一父域不同子域)单服务器，我们需要解决的就是来自不同网站之间SessionId的共享。由于域名不同(aaa.test.com和bbb.test.com)，而SessionId又分别储存在各自的cookie中，因此服务器会认为对于两个子站的访问,是来自不同的会话。解决的方法是通过修改cookies的域名为父域名达到cookie共享的目的,从而实现SessionId的共享。带来的弊端就是，子站间的cookie信息也同时被共享了。

cookie与session的区别

1、cookie数据存放在客户的浏览器上，session数据放在服务器上。

2、cookie不是很安全，别人可以分析存放在本地的cookie并进行cookie欺骗，考虑到安全应当使用session。

3、session会在一定时间内保存在服务器上。当访问增多，会比较占用你服务器的性能，考虑到减轻服务器性能方面，应当使用cookie。

4、单个cookie保存的数据不能超过4K，很多浏览器都限制一个站点最多保存20个cookie。

5、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。

web Storage

web 本地存储对浏览器来说，使用 Web Storage 存储键值对比存储 Cookie 方式更直观，而且容量更大，它包含两种：localStorage 和 sessionStorage

sessionStorage（临时存储）：为每一个数据源维持一个存储区域，在浏览器打开期间存在，包括页面重新加载

localStorage（长期存储）： 一直存储在本地，数据存储是永久的，除非用户或程序对其进行删除操作；localStorage 对象存储的数据没有时间限制。第二天、第二周或下一年之后，数据依然可用。

sessionStorage 和 localStorage 的用法基本一致，引用类型的值要转换成JSON

特点：

域内安全、永久保存。即客户端或浏览器中来自同一域名的所有页面都可访问localStorage数据且数据除了删除否则永久保存，但客户端或浏览器之间的数据相互独立。

数据不会随着Http请求发送到后台服务器；

存储数据的大小至少4MB。

存储方式：以键值对(Key-Value)的方式存储字符串。
主要应用：购物车、客户登录、游戏存档。。。
可储存的数据类型：数组，图片，json，样式，脚本。。。（只要是能序列化成字符串的内容都可以存储）

JS API
localStorage.setItem(键名，键值)　　在本地客户端存储一个字符串类型的数据
localStorage.getItem(键名)　　读取已存储在本地的数据
localStorage.removeItem(键名)　移除已存储在本地的数据
localStorage.clear()　　移除本地存储所有数据

sessionStorage中的四个函数与以上localStorage类的函数用法基本一致

cookie与web Storage的区别
1、cookie数据始终在同源的http请求中携带
2、cookie可设置路径，限制其作用域
3、存储大小，cookie不能超过4k，web Storage可达5M
4、cookie有数据有效期，web Storage无
5、作用域不同，sessionStroage不在不同浏览器窗口共享，
　　localStorage与cookie在所有同源窗口共享

cookie与webStorage的使用方法
1、兼容性：IE9+支持web Storage，cookie作为远古时期产物，无兼容性问题
2、交互性：web Storage不与服务器交互，需要与服务器交互时使用cookie
3、数据大小：cookie不能超过4k，web Storage可达5M