Cookie通过在**客户端**记录信息确定用户身份，Session通过在**服务器端**记录信息确定用户身份

**1.Cookie机制**

理论上，**一个用户的所有请求操作都应该属于同一个会话**，而**另一个用户的所有请求操作则应该属于另一个会话**，二者不能混淆。

HTTP协议是无状态的协议，当重新建立连接的时候，无法从连接上分辨出该连接属于哪个用户。即用户A购买了一件商品放入购物车，当再次购买商品时服务器无法判断该购买行为是属于用户A还是其他用户。

在Session出现之前，基本上所有的网站都采用Cookie跟踪。

**什么是cookie？**

Cookie相当于一个通行证，每个用于一个，无论谁访问服务器都必须携带通行证，这样服务器就能从通行证上确认客户身份了。

客户端请求服务器，如果服务器**需要记录该用户状态**，就**使用response向客户端浏览器颁发一个Cookie**，**客户端浏览器会把Cookie保存起来**。当浏览器再请求该网站时，**浏览器把请求的网址连同该Cookie一同提交给服务器**。服务器检查该Cookie，以此来辨认用户状态。服务器还可以**根据需要修改Cookie**的内容。

浏览器地址栏输入**javascript:alert (document. cookie**)就可以查看网站颁发的Cookie

**记录用户访问次数**

Java中把Cookie封装成了javax.servlet.http.Cookie类。**每个Cookie都是该Cookie类的对象。**服务器通过操作**Cookie类对象**对客户端Cookie进行操作。通过**request.getCookie()**获取客户端提交的所有Cookie（**以Cookie[]数组形式返回**）**，通过response.addCookie (Cookiecookie)向客户端设置Cookie。**

**Cookie的不可跨域名性**：根据Cookie规范，浏览器访问Google只会携带Google的Cookie，而不会携带Baidu的Cookie。Google也只能操作Google的Cookie，而不能操作Baidu的Cookie。

Cookie在客户端是**由浏览器来管理的**。浏览器能够**保证Google只会操作Google的Cookie**而**不会操作Baidu的Cookie**，从而保证用户的隐私安全。浏览器判断一个网站是否能操作另一个网站Cookie的依据是**域名**。Google与Baidu的域名不一样，因此**Google不能操作Baidu的Cookie。**

需要注意的是，**虽然网站images.google.com与网站www.google.com同属于Google**，**但是域名不一样，**二者同样不能互相操作彼此的Cookie。

**设置cookie的所有属性**

每个属性对应一个getter方法和setter方法

|  |  |
| --- | --- |
| 属性名 | 描述 |
| String name | 该cookie的名称。Cookie一旦创建，名称便不可更改 |
| Object value | 该cookie的值。如果为unicode字符，需要编码，如果二进制，使用BASE64编码 |
| Int maxAge | Cookie的实效时间，单位秒。如果正数，maxAge后失效，如果负数仅在本浏览器窗口以及本窗口打开的子窗口内有效，关闭窗口后该Cookie即失效，默认-1 |
| Boolean secure | 该cookie是否仅被使用安全协议传输。安全协议有HTTPS, SSL，默认为false |
| String path | 使用路径，如果“/sessionWeb/”，则只有contextPath为“/sessionWeb”的程序可以访问该cookie；如果设置为”/”，则contextPath都可以访问该cookie。注意最后一个字符必须为“/” |
| String domain | 可访问的域名，如果为.google.com,则所有以”google.com”结尾的域名都可以访问该cookie，注意第一个字符必须为”.” |
| String comment | 用处说明 |
| Int version | Cookie使用的版本号 |

**Cookie的修改。删除**

Cookie并不提供修改删除功能，如果需要修改某个cookie，只需要新建一个同名的cookie，添加到response中覆盖原来的cookie，如果需要删除某个cookie，只需要新建一个同名cookie，然后设置maxAge为0；

注意：修改、删除Cookie时，新建的Cookie除value、maxAge之外的所有属性，例如name、path、domain等，都要与原Cookie完全一样。否则，浏览器将视为两个不同的Cookie不予覆盖，导致修改、删除失败。

**Cookie的域名**

正常情况下，同一个一级域名下的两个域名如[www.baidu.com和images.baidu.com](http://www.baidu.com和images.baidu.com)也不能交互使用cookie，因为二者的域名并不严格相同。如果想所有baidu.com名下的二级域名都可以使用该Cookie，需要设置Cookie的domain参数：

|  |
| --- |
| Cookie cookie = new Cookie(“time”, “20181213”);  Cookie.setDomain(“.baidu.com”);  Cookie.setPath(“/”);  Cookie.setMaxAge(Integer.MAX\_VALUE);  Response.addCookie(cookie); |

domain参数必须以点(".")开始。另外，name相同但domain不同的两个Cookie是两个不同的Cookie。如果想要两个域名完全不同的网站共有Cookie，可以生成两个Cookie，domain属性分别为两个域名，输出到客户端。

**Cookie的路径**

Domain决定运行访问cookie的域名，而path属性决定允许访问cookie的路径（contextpath），如果只允许/sessionWeb/下的程序使用Cookie,可以：

|  |
| --- |
| Cookie cookie = new Cookie(“time”, “20181213”);  Cookie.setPath(“/session/);  Response.addCookie(cookie); |

设置为“/”时允许所有路径使用cookie，path属性需要使用符号“/”结尾。Name相同但domain相同的两个cookie也是不同的cookie

注意：页面只能获取它属于的Path的Cookie。例如/session/test/a.jsp不能获取到路径为/session/abc/的Cookie。使用时一定要注意。

**2.Session机制**

**什么是session？**

服务端把客户端信息以某种形式记录在服务器上，等到客户端再次访问的时候，只需要从session中查找客户的状态即可。

Session相当于程序在服务器上建立的一份客户档案。

**Session的生命周期？**

**为了获得更高的存取速度，服务器一般把Session放在内存里。每个用户都会有一个独立的Session。如果Session内容过于复杂，当大量客户访问服务器时可能会导致内存溢出。**

**Session在用户第一次访问服务器的时候自动创建**。**Session生成后，只要用户继续访问，服务器就会更新Session的最后访问时间，并维护该Session**。

**Session在用户第一次访问服务器的时候自动创建**。需要注意只有访问JSP、Servlet等程序时才会创建Session，只访问HTML、IMAGE等静态资源并不会创建Session。如果尚未生成Session，也可以使用request.getSession(true)强制生成Session。

**Session的常用方法：**

|  |  |
| --- | --- |
| 方法名 | 描述 |
| Void setAttribute(String attribute, Ojbect value) | 设置Session属性。Value参数可以为任何**Java Object**。通常java bean。 Value信息不宜过大。 |
| String getAttribute(String attribute) | 返回session属性 |
| Enumeration getAttributeNames() | 返回Session中存在的属性名 |
| Void removeAttribute(String attribute) | 移除Session属性 |
| String getId() | 返回Session的id，由服务器自动创建，不会重复 |