1.应用层协议：

1.1 http协议：

1.1.1 cookie和session的由来：

1.http协议是七层协议最上层的应用层的协议，本身是无状态。因此每次请求

（一次请求就是客户端向服务端请求资源的一个过程，request到response，在 struts中就是表示一个action）都不能记录用户的记录。

2.为了补充http协议的无状态，出现了cookie和session技术。

cookie： Cookies技术是服务端生成的存储在客户端的一些 键值对 数据，用 来记录用户数据的。

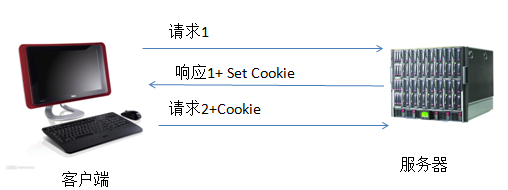
cookie的生成是通过服务器端生成的，具体代码如下：

|  |
| --- |
| **Cookie** c = **new** Cookie("JSESSIONID",request.getSession().getId()); c.setMaxAge(60);//表示3600秒 c.setPath("/");//路径,/表示根目录,默认是"/项目名" response.addCookie(c);//添加cookie到response中，存储到浏览器端 |

但是，值得注意的是，JSEESSIONID 这个cookie是服务器自动生成的。

当访问jsp和servlet的时候，会自动创建session，原因是在：

HttpSession session=request.getSession(true);这样就会自动在请求的时候创建一个session，同时，创建一个JSEEIONID的cookie

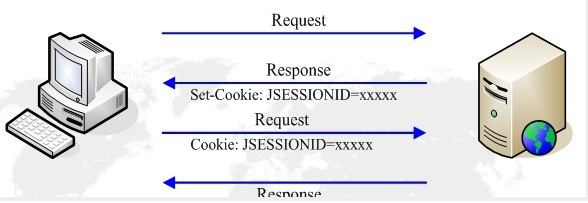


session:

***通过Session保存状态信息***

Session机制是一种服务器端的机制，服务器使用一种类似于散列表的结构（也可能就是使用散列表）来保存信息。  
当程序需要为某个客户端的请求创建一个session的时候(比如访问jsp或者servlet的时候)，服务器首先检查这个客户端的请求里是否已包含了一个session标识 - 称为 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（如果检索不到，可能会新建一个），如果客户端请求不包含session id，则为此客户端创建一个session并且生成一个与此session相关联的session id，session id的值应该是一个既不会重复，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保存（变成cookie形式）。

**Session的实现方式：**

1、使用Cookie来实现  
服务器给每个Session分配一个唯一的JSESSIONID，并通过Cookie发送给客户端。  
当客户端发起新的请求的时候，将在Cookie头中携带这个JSESSIONID。这样服务器能够找到这个客户端对应的Session。  


2、使用URL回写来实现  
URL回写是指服务器在发送给浏览器页面的所有链接中都携带JSESSIONID的参数，这样客户端点击任何一个链接都会把JSESSIONID带会服务器。如果直接在浏览器输入服务端资源的url来请求该资源，那么Session是匹配不到的。  
Tomcat对Session的实现，是一开始同时使用Cookie和URL回写机制，如果发现客户端支持Cookie，就继续使用Cookie，停止使用URL回写。如果发现Cookie被禁用，就一直使用URL回写。jsp开发处理到Session的时候，对页面中的链接记得使用response.encodeURL() 。

**Cookie和Session有以下明显的不同点：**  
1）Cookie将状态保存在客户端，Session将状态保存在服务器端；  
2）Cookies是服务器在本地机器上存储的小段文本并随每一个请求发送至同一个服务器。Cookie最早在RFC2109中实现，后续RFC2965做了增强。网络服务器用HTTP头向客户端发送cookies，在客户终端，浏览器解析这些cookies并将它们保存为一个本地文件，它会自动将同一服务器的任何请求缚上这些cookies。Session并没有在HTTP的协议中定义；  
3）Session是针对每一个用户的，变量的值保存在服务器上，用一个sessionID来区分是哪个用户session变量,这个值是通过用户的浏览器在访问的时候返回给服务器，当客户禁用cookie时，这个值也可能设置为由get来返回给服务器；  
4）就安全性来说：当你访问一个使用session 的站点，同时在自己机子上建立一个cookie，建议在服务器端的SESSION机制更安全些。因为它不会任意读取客户存储的信息。

1.1.2浏览器运行时的cookie和session：

问题1：cookie本质是什么？

一般来说一次http请求并不会自己创造一个cookie，并在回写的时候，回写到客户端。而是通过

**Cookie** c = **new** Cookie("JSESSIONID",request.getSession().getId());  
c.setMaxAge(60);//表示3600秒  
c.setPath("/shop");//路径,/表示根目录,默认是"/项目名"  
response.addCookie(c);//添加cookie到response中

这个方法，把自己创造的cookie写进set-cookie回应头里，那么在响应的时候就会带上相应的设置的cookie。

|  |
| --- |
| domain：表示的是cookie所在的域，默认为请求的地址。  path：表示cookie所在的域，默认为”/项目”  cookie的作用范围由 domain+path 来决定。 |

这个cookie如果没设置时间，那么在会话结束的时候（关闭浏览器），这个cookie就消失了，他是保存在相应的浏览器内存中。

如果设置了时间，那么会直到时间消失，这个cookie才会消失，因为它是保存在硬盘的。

再来看看cookie的作用域示例：

|  |
| --- |
| 我们在项目A(/shop)中：  **Cookie** c = **new Cookie**("JSESSIONID",request.getSession().getId()); c.setMaxAge(60);//表示3600秒 c.setPath("/shop");//路径,/表示根目录,默认是"/项目名" response.addCookie(c);//添加cookie到response中  **Cookie** c1= **new Cookie**("abc","zk"); c1.setMaxAge(600);//表示3600秒 c1.setPath("/");//路径,/表示根目录,默认是"/项目名" response.addCookie(c1);//添加cookie到response中  在项目A中，我们设置了两个cookie，并且设置了时间，我们对项目A中的资源请求的时候，浏览器中cookie如图：  屏幕剪辑  上面一个作用域是：localhost/shop  下面一个作用域是：localhost/  关闭项目A，然后访问项目B（/java），项目B中没有设置任何cookie，但是：  屏幕剪辑 |
| 屏幕剪辑  可以看到访问项目B的时候，因为项目A中的第二个cookie的设置域为  localhost:/ ,因此访问localhost:/java的时候，这个cookie也会起作用。 |

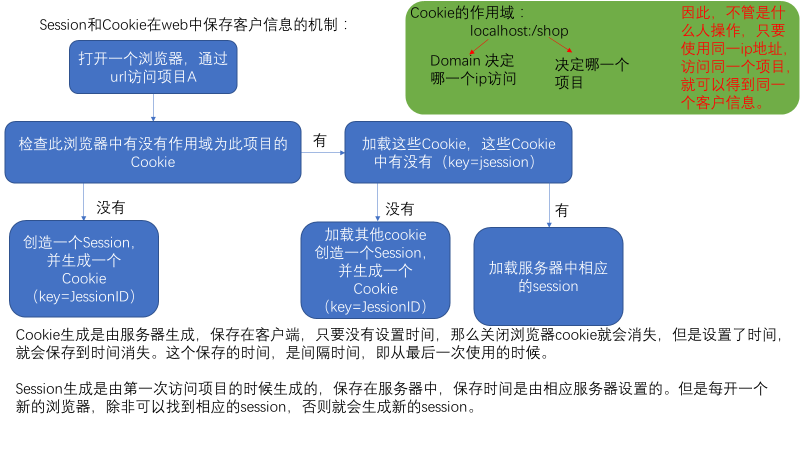
2.session是怎么通过cookie保持的呢？

在访问jsp或者servlet的时候，会自动执行

request.getSession(true);

因此，在第一次访问服务器的（jsp和servlet）资源的时候，服务器会创建一个相应的session（这个session存在的时间由服务器apache设置），存储在服务器的内存中。但是HTTP协议是无状态的，Session是存放在服务器中的，浏览器不能依据HTTP连接来判断是否为同一客户。所以，与此同时，服务器端会生成一个key为jesession的cookie。

**cookie和session一起保存信息的原理图**



1.1.3 利用cookie和session的实例：

需求：



一个网站登录后：

 在关闭浏览器后，再访问这个页面，依旧可以显示登录状态了。

解决方式：

**在访问主页的action前**

**HttpServletResponse** response=**ServletActionContext**.getResponse();  
HttpServletRequest request=**ServletActionContext**.getRequest();  
**Cookie** c = **new** Cookie("JSESSIONID",request.getSession().getId());  
c.setMaxAge(3600);//表示3600秒  
c.setPath("/shop");//路径,/表示根目录,默认是"/项目名"  
response.addCookie(c);//添加cookie到response中

2.详解http协议：

主要由请求和相应组成

2.1请求：

格式：

请求方式 请求URL 协议版本

请求头1：

请求头2：

（这里是一行空，代表请求头和请求体分离）

请求体

请求头一般有：Host：主机地址 User-Agent:浏览器信息 Cookie:cookie

请求方式有5大类：

put,get,post,delete,head

主要有get和post： 1. 当为get的时候，请求地址主要在地址栏里面附带，且请求头中不会出现Content\_type 和 Content\_length ,没有请求体。

2.当为post的时候，url中不带参数，而是把请求参数放到请求体中，且请求头中有Content\_Type 和 Content\_length。

上传一般信息的时候Content\_Type采用application/x-www-form-urlcoded，上传其他信息如文件和图片的时候采用的是application/octet-stream或者multipart/form-data

2.2 响应：

格式：

版本 相应状态

响应头1：

响应头2：

（一行空格）

响应体：

响应状态码：

2XX:表示成功

3XX:表示重定向或者转发

4XX:表示资源未找到

5XX:表示服务器有错误。

响应头：

Content-type: 设置响应体的格式 text/html;charset=utf-8;

Content\_length: 响应体长度

last-modified: 最后一次修改时间

Expires:过期时间

cach-control:缓存控制

响应体：

服务器传给客户端的东西，例如网页，字符串。