# 会话技术

## 会话技术概述

### 什么是会话

为了实现某一个功能(购物)，浏览器和服务器之间可能会产生多次的请求和响应。从打开浏览器访问服务器开始，到访问服务器结束关闭浏览器为止，这期间产生的多次请求和响应加在一起就称之为是浏览器和服务器之间的一次会话！

### 会话中的数据存储问题

在一次会话中往往会产生一些数据，而这些数据是需要保存起来的，如何在一次会话中保存所产生的数据呢?

如果需要保存会话中产生的数据，可以考虑我们之前学习过的request域和ServletContext域：

（1）如果使用request作用域来保存会话中产生的数据，由于会话是多次请求，多次响应，而request域的作用范围仅仅是一次请求，所以用request域保存会话中产生的数据，作用范围太小了!

（2）如果使用ServletContext作用域来保存会话中产生的数据，ServletContext的作用范围是整个WEB应用，如果将每一个客户端和服务器产生的会话数据都保存在ServletContext域中，很可能会发生混乱.

## Cookie

### Cookie的工作原理

Cookie是通过Set-Cookie响应头和Cookie请求头将会话中产生的数据保存在客户端，是客户端的 技术.

客户端向服务器发送请求，服务器获取需要保存的数据，并将需要保存的数据通过Set-Cookie响应头发送给浏览器，浏览器会以Cookie的形式保存在浏览器的内部.

当客户端再次发送请求访问服务器，服务器可以通过Cookie请求头获取上次发送给浏览器的Cookie信息，通过这种方式可以保存会话中产生的数据.

由于Cookie技术是将会话中产生的数据保存在客户端，每个客户端各自持有自己的数据，当需要时再带给服务器，因此不会发生混乱！

图示：



### Cookie的API介绍

**1、如何创建Cookie对象**

Cookie cookie = new Cookie(String name，String value);

**2、如何将Cookie对象添加到响应中**

response.addCookie(Cookie cookie) – 将指定的Cookie对象添加到响应中，最后随响应发送给浏览器保存(底层还是通过set-Cookie响应头将Cookie值发送给浏览器)

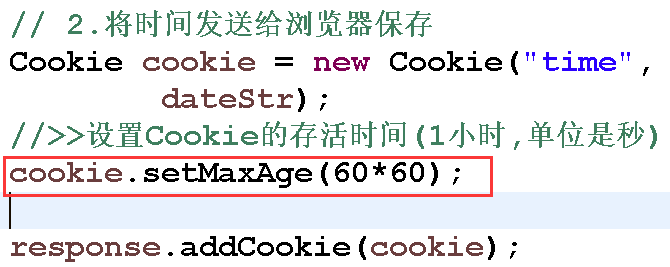
**3、如何获取请求中的Cookie信息**

request.getCookies() – 返回请求中所有的Cookie对象组成的数组. 如果请求中没有任何Cookie信息，该方法将会返回null。

**4、setMaxAge方法 – 设置Cookie的最大生存时间**

（1）如果不设置该方法，Cookie默认是会话级别的Cookie,是保存在浏览器的内存中. 随着浏览器的关闭，Cookie也会跟着销毁.

（2）如果设置了该方法，并且是一个有效的时间，Cookie将会以文件的形式保存在浏览器的临时文件夹中。



**5、setPath方法 – 设置Cookie被带给服务器的路径**

比如：发送Cookie的Servlet是CookieDemo2

CookieDemo2的访问路径为：[http：//localhost/day16/CookieDemo2](http://localhost/day16/CookieDemo2)

Cookie的path为：/day16/，这样浏览器在访问/day16/和其子孙路径时将会带着Cookie

如果Cookie的path不是当前web应用的根路径(比如：不是/day16/,而是/day16/servlet/)，这样浏览器在访问day16下某些资源时将不会带着Cookie，所以最好把Cookie的路径设置为当前WEB应用的根路径，代码如下：

//设置Cookie的路径(确保浏览器在访问当前应用下任何一个资源时都可以带着Cookie)

String cp = request.getContextPath();

cookie.setPath(cp+"/");

**6、如何删除Cookie**

在Cookie的API中没有提供直接删除Cookie的方法.

我们可以间接删除Cookie：可以向浏览器再发一个同名、同path、同domain(如果前后都不设置，则domain默认是相同的)的Cookie，由于浏览器是根据Cookie的名字+path+domain来区分一个Cookie的，如果两个Cookie的名字+path+domain完全一致，则浏览器会认为是同一个Cookie，这时后发的Cookie会覆盖之前发送的Cookie，而后发的Cookie只要设置setMaxAge方法值为0,浏览器收到之后也会将后发的Cookie立即删除。

示例：

//创建一个同名的Cookie

Cookie cookie = new Cookie("time","");

//指定相同的path

cookie.setPath(

request.getContextPath()+"/");

//指定Cookie的生存时间为0

cookie.setMaxAge(0);

//将Cookie添加到响应中发送给浏览器

response.addCookie(cookie);

response.getWriter().write("Cookie Delete Success!!");

**7、Cookie的常用方法**

（1）cookie.getName() – 获取Cookie的名字

（2）cookie.getValue() – 获取Cookie的值

（3）cookie.setValue() – 设置Cookie的值

（4）cookie.setMaxAge() – 设置Cookie的最大生存时间

（5）cookie.setPath() – 设置Cookie的路径

### 案例：在网页中显示上次访问时间

需求：当用户访问网站时，可以获取到上次访问的时间.

**1、代码实现\_1（传统方式实现）：**

// 1.获取本次访问的时间.

String dateStr = new Date().toLocaleString();

// 2.将时间通过Set-Cookie响应头响应给浏览器

response.setHeader("Set-Cookie"，"time=" + dateStr);

// 3.获取上次访问的时间

String dateStr2 = request.getHeader("Cookie");

// 4.将上次访问时间响应给浏览器并显示在窗口中

// 处理响应正文乱码

response.setContentType("text/html;charset=utf-8");

if(dateStr2 != null){

response.getWriter().write("您上一次访问的时间为："+ dateStr2);

}else{

response.getWriter().write("您是第一次访问本网站...");

}

**2、代码实现\_2（通过Cookie的API实现）：**

// 0.处理响应正文乱码

response.setContentType("text/html;charset=utf-8");

// 1.获取(记录)本次访问的时间

String dateStr = new Date().toLocaleString();

// 2.创建Cookie对象,将时间设置在Cookie对象中

Cookie cookie = new Cookie("time"，dateStr);

//设置Cookie最大存活时间

cookie.setMaxAge(3600\*24\*365);

//设置Cookie的路径(确保浏览器在访问当前应用下任何一个资源时都可以带着Cookie)

String cp = request.getContextPath();

cookie.setPath(cp+"/");

// 3.将Cookie对象添加到响应中(发送给浏览器保存)

response.addCookie(cookie);

// 4.通过请求获取所有的Cookie对象组成的数组

Cookie[] cs = request.getCookies();

boolean flag = false;// false表示没有名称为time的cookie

if(cs != null){

for (Cookie c ：cs) {

if ("time".equals(c.getName())) {

response.getWriter().write("您上次访问的时间是：" + c.getValue());

flag = true;

}

}

}

// 5.如果没有名称为time的Cookie，则提示是第一次访问

if (!flag) {

response.getWriter().write("您是第一次访问本网站...");

}

## Session

### Session的原理

Session是将会话中产生的数据保存在服务端，是服务端的技术.

浏览器第一次发送请求需要保存数据时,服务端获取到需要保存的数据，去服务器内部检查一下有没有为当前浏览器服务的session，如果有就直接拿过来用，如果没有session就创建一个新的session拿过来用. 接着将数据保存在Session中，做出响应

当浏览器再去访问服务器时，服务器可以从session中获取到之前为当前浏览器保存的数据，通过这种方式，也可以来保存会话中产生的数据！

图示：



### Session是一个域对象

1、域对象：如果一个对象具有可以被看见的范围，利用该对象上的map可以在整个范围内实现数据的共享.

提供的操作域中属性的方法：

setAttribute();

getAttribute();

removeAttribute();

getAttributeNames();

2、生命周期：

（1）创建：当第一次调用request.getSession方法时，创建session对象

request.getSession();

request.getSession(true);

在调用上述方法时，如果服务器内部有为当前客户端服务的session，就直接拿过来使用，如果没有对应的session，就创建一个新的session拿过来用.

request.getSession(false); 在调用该方法时，如果服务器内部有为当前浏览器服务的session就直接拿过来使用，如果没有对应的session，就返回null.

（2）销毁：

a）超时销毁：如果session 30分钟没有被使用，则会超时销毁. 可以在web应用的web.xml文件中修改session的超时时间.

b）自杀：当调用session.invalidate方法时,将会立即销毁session

c）意外身亡：当服务器非正常关闭时，session会销毁! 如果服务器正常关闭，session将会以文件的形式保存在服务器的work目录下，这个过程称之为session的钝化. 当服务器再次启动时，钝化着的session还可以再恢复回来，这个过程称之为session的活化!

3、作用范围：整个会话范围内

4、主要功能：在整个会话范围内共享数据

### 案例：实现购物车

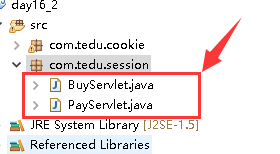
1、需求：实现用户浏览器商品列表，可以将商品加入购物车，并可以对购物车中的商品进行结算.

2、实现步骤：

（1）创建一个商品列表页面：prod\_list.jsp



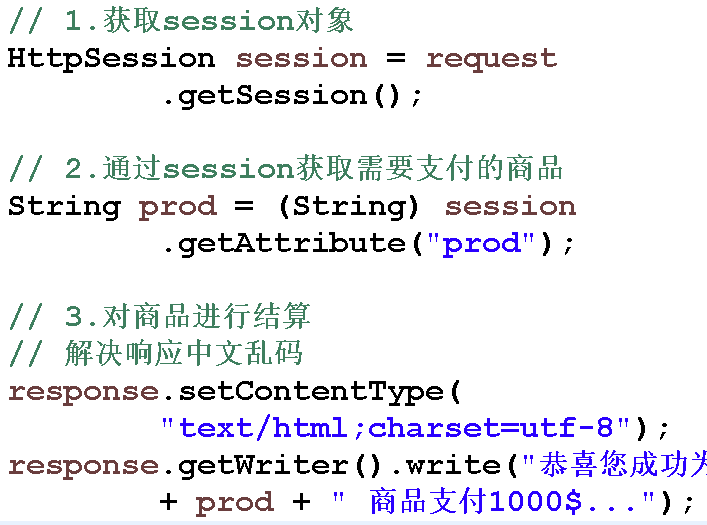
（2）创建两个Servlet：BuyServlet和PayServlet



（3）实现BuyServlet：将商品加入购车并提示.



（4）实现PayServlet：对购物车中的商品进行结算并提示



### 如何在浏览器关闭之后再打开时获取之前的session

1、**问题描述：**由于session创建之后，服务器会将Session的id作为Cookie发送给浏览器保存，方便浏览器下次来访问服务器时，可以通过Cookie将Session的id带给服务器，这样就可以找到之前的session. 如果浏览器中途关闭，Cookie会丢失(服务器默认发送的Cookie是会话级别的Cookie)，session的id也会丢失，所以再次访问服务器时，找不到之前的session.

2、**解决方案：**

在创建session后，服务器会向浏览器发送Cookie来保存session的id，我们也可以向浏览器发送Cookie，也保存session的id，区别是我们发送的Cookie，可以设置最大生存时间，这个Cookie将会保存在硬盘上，即使浏览器关闭，Cookie也不会丢失，下次来访问服务器仍然可以带着session的id，也可以获取之前的之前.

