Cookie与Session

0. 会话技术简介

在日常生活中，从拨通电话到挂断电话之间的一连串的你问我答的过程就是一个会话。Web应用中的会话过程类似于生活中的打电话过程，它指的是一个客户端(浏览器)与Web服务器之间连续发生的一系列请求和响应过程，例如，一个用户在某网站上的整个购物过程就是一个会话。在打电话过程中，通话双方会有通话内容，同样，在客户端与服务器端交互的过程中，也会产生一些数据。例如，用户甲和乙分别登录了购物站，甲购买了一个Nokia手机，乙购买了一个Ipad，当这两个用户结账时，Web服务器需要对用户甲和乙的信息分别进行保存。在前面章节讲解的对象中，HttpServletRequest对象和ServletContext对象都可以对数据进行保存，但是这两个对象都不可行，具体原因如下:

1. 客户端请求Web服务器时，针对每次HTTP请求，Web服务器都会创建一个HttpServletRequest对象，该对象只能保存本次请求所传递的数据。由于购买和结账是两个不同的请求，因此，在发送结账请求时，之前购买请求中的数据将会丢失。
2. 使用ServletContext对象保存数据时，由于同一个Web应用共享的是同一个ServletContext对象，因此，当用户在发送结账请求时，由于无法区分哪些商品是哪个用户所购买的，而会将该购物网站中所有用户购买的商品进行结算，这显然也是不可行的。
3. 为了保存会话过程中产生的数据，在Servlet技术中，提供了两个用于保存会话数据的对象，分别是Cookie和Session。
4. 

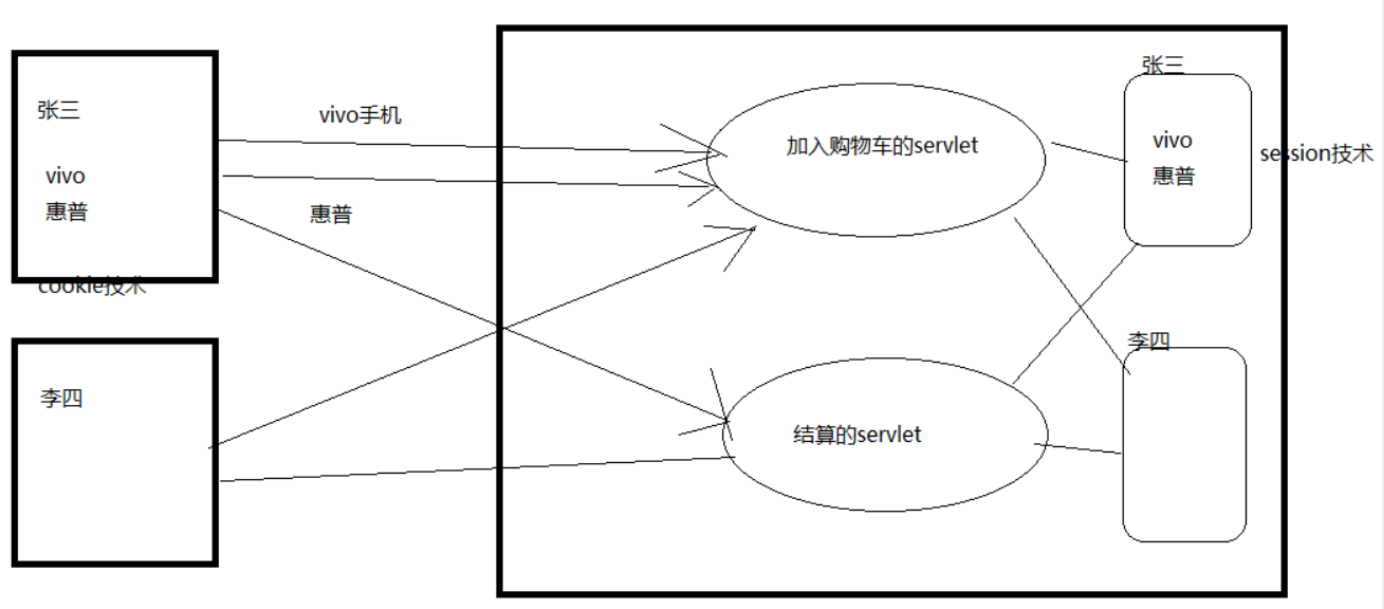
0.1 存储客户端的状态

由一个问题引出今天的内容，例如网站的购物系统，用户将购买的商品信息存储到哪里？因为Http协议是无状态的，也就是说每个客户访问服务器端资源时，服务器并不知道该客户端是谁，所以需要会话技术识别客户端的状态。会话技术是帮助服务器记住客户端状态（区分客户端）。

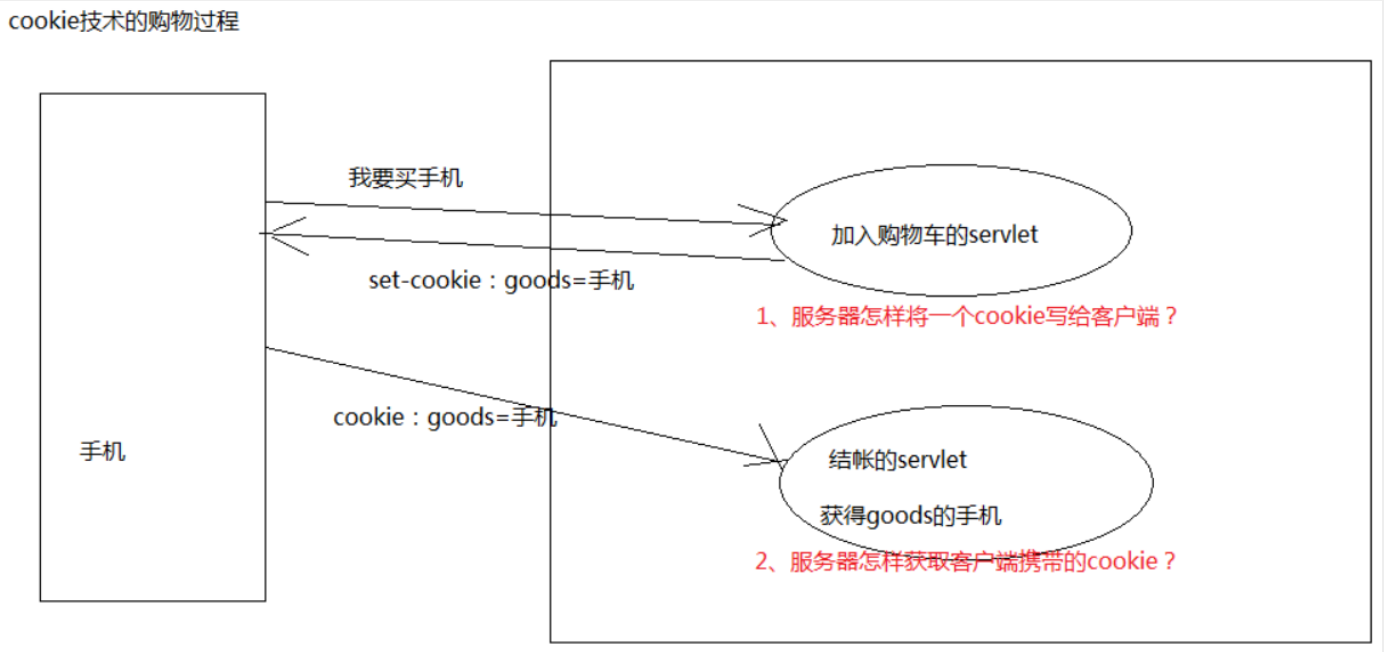
0.2 会话技术

从打开一个浏览器访问某个站点，到关闭这个浏览器的整个过程，成为一次会话。会话技术就是记录这次会话中客户端的状态与数据的。
会话技术分为Cookie和Session：

Cookie：数据存储在客户端本地，减少服务器端的存储的压力，安全性不好，客户端可以清除cookie。
Session：将数据存储到服务器端，安全性相对好，增加服务器的压力。



1. Cookie概念



在现实生活中，当顾客在购物时，商城经常会赠送顾客一张会员卡，卡上记录用户的个人信息(姓名，手机号等)、消费额度和积分额度等。顾客一旦接受了会员卡，以后每次光临该商场时，都可以使用这张会员卡，商场也将根据会员卡上的消费记录计算会员的优惠额度和累加积分。在Web应用中，Cookie的功能类似于这张会员卡，当用户通过浏览器访问、web服务器时，服务器会给客户端发送一些信息，这些信息都保存在Cookie中。这样，当该浏览器再次访问服务器时，都会在请求头中将Cookie发送给服务器，方便服务器对浏览器做出正确的响应。

2. Cookie的使用

2.1 设置Cookie

2.1.1 创建并添加Cookie

```
16          //1、创建cookie对象
17          Cookie cookie = new Cookie("name","zhangsan");
```

```
24          //2、将cookie中存储的信息发送到客户端---头，不要用response.setHeader
25          response.addCookie(cookie);
```

每个客户端访问我的servlet资源时，都会存储一个Cookie信息，Set-Cookie：name=zhangsan，在http响应中存在，若是第二次访问，那么在http请求中也存在。

2.1.2 设置Cookie的持久化时间

```
18          //1.1 为cookie设置持久化时间 ---- cookie信息在硬盘上保存的时间
19          //cookie.setMaxAge(10*60);//10分钟 ---- 时间设置为0代表删除该cookie
```

注意：如果不设置持久化时间，cookie会存储在浏览器的内存中，浏览器关闭cookie信息销毁（会话级别的cookie），如果设置持久化时间，cookie信息会被持久化到浏览器的磁盘文件里。

2.1.3 设置Cookie的携带路径

```
20          //1.2 为cookie设置携带的路径，以避免拖慢传输速度
21          cookie.setPath("/WEB16/sendCookie");//访问sendCookie资源时才携带这个cookie
22          //cookie.setPath("/WEB16");//访问WEB16下的任何资源时都携带这个cookie
23          //cookie.setPath("/");//访问服务器下的所有的资源都携带这个cookie
24          //2、将cookie中存储的信息发送到客户端---头，不要用response.setHeader
```

如果将该资源的配置路径变复杂一点，变成/demo/sendCookie,如果不配置携带路径，那么默认的访问路径是/demo下的路径及其任何子路径。
所以说，如果不配置携带路径，虚拟路径为一层，那么默认的访问路径就是该项目路径及其子路径，等同于这里的/WEB16。

2.2 获取Cookie

```java
        //获得客户端携带的cookie的数据
        Cookie[] cookies = request.getCookies();

        //通过cookie名称获得想要的cookie
        if(cookies!=null){
            for(Cookie cookie : cookies){
                //获得cookie的名称
                String cookieName = cookie.getName();
                if(cookieName.equals("name")){
                    //获得该cookie的值
                    String cookieValue = cookie.getValue();
                    System.out.println(cookieValue);
                }
            }
        }
```

2.3 移除Cookie

```java
        //删除客户端保存 name=zhangsan的cookie信息
        Cookie cookie = new Cookie("name","");
        //将path设置成与要删除cookie的path一致
        cookie.setPath("/WEB16");
        //设置时间是0
        cookie.setMaxAge(0);
```

将客户端的Cookie创建一个同名对象，直接将其持久化时间设置为0就行，注意要将path设置成与要删除的Cookie的path相同。

2.4 获取上次访问时间

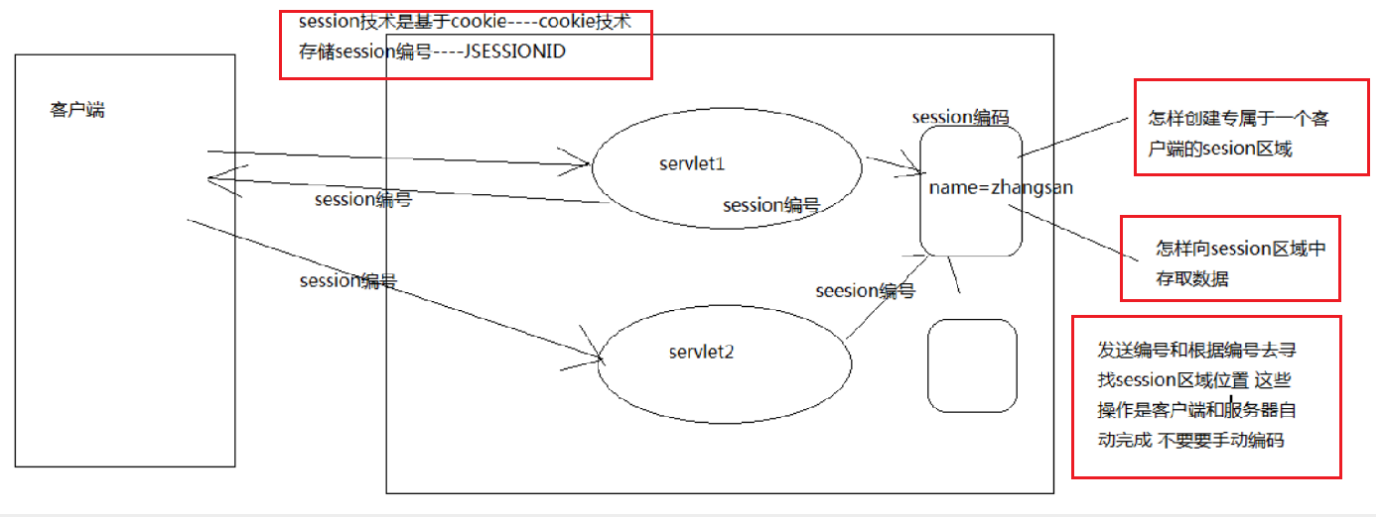
```java
        //获得当前时间
        Date date = new Date();
        SimpleDateFormat format = new SimpleDateFormat("yyyy-MM-dd hh:mm:ss");
        String currentTime = format.format(date);

        //1、创建Cookie 记录当前的最新的访问时间
        Cookie cookie = new Cookie("lastAccessTime",currentTime);
        cookie.setMaxAge(60*10*500);
        response.addCookie(cookie);

        //2、获得客户端携带cookie ---- lastAccessTime
        String lastAccessTime = null;
        Cookie[] cookies = request.getCookies();
        if(cookies!=null){
            for(Cookie coo : cookies){
                if("lastAccessTime".equals(coo.getName())){
                    lastAccessTime = coo.getValue();
                }
            }
        }

        response.setContentType("text/html;charset=UTF-8");
        if(lastAccessTime==null){
            response.getWriter().write("您是第一次访问");
        }else{
            response.getWriter().write("您上次的访问的时间是："+lastAccessTime);
        }
    }
```

3. Session的概念



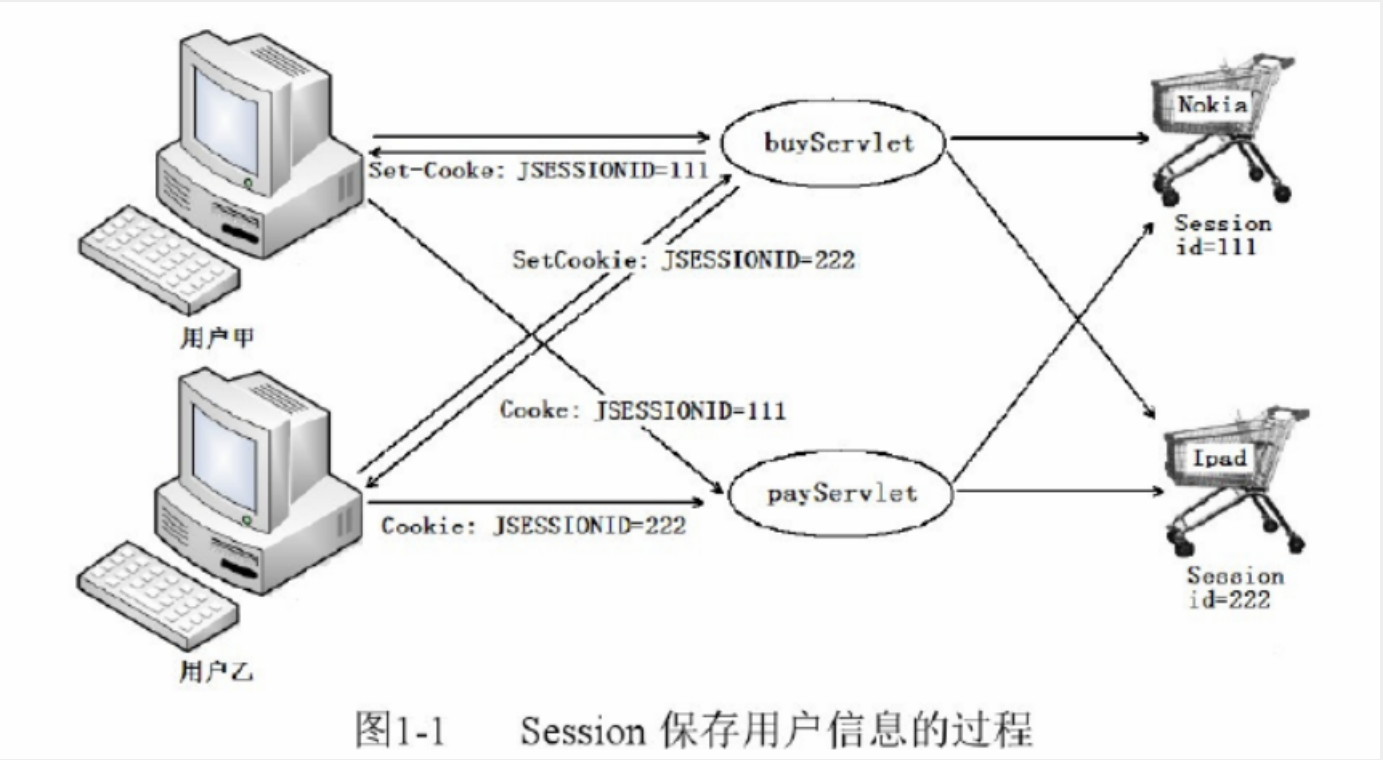


图1-1　Session 保存用户信息的过程

当人们去医院就诊时，就诊病人需要办理住院的就诊卡，该卡上只有卡号，而没有其它信息。但病人每次去该医院就诊时，只要出示就诊卡，医务人员便可根据卡号查询到病人的就诊信息。Session技术就好比医院发放给病人的就医卡和医院为每个病人保留病例档案的过程。当浏览器访问Web服务器时，Servlet容器就会创建一个Session对象和ID属性，其中，Session对象就相当于病历档案，ID就相当于就诊卡号。当客户端后续访问服务器时，只要将标识号传递给服务器，服务器就能判断出该请求是哪个客户端发送的，从而选择与之对应的Session对象为其服务。 需要注意的是，由于客户端需要接收、记录和回送Session对象的ID，因此，通常情况下，Session是借助Cookie技术来传递ID属性的。

5. Session的使用

5.1 如何创建Session对象

```
HttpSession session = request.getSession();
```

此方法会获得专属于当前会话的Session对象，如果服务器端没有该会话的Session对象会创建一个

新的Session返回，如果已经有了属于该会话的Session直接将已有的Session返回（实质就是根据JSESSIONID判断该客户端是否在服务器上已经存在session了）

5.2 怎样向session中存取数据（session也是一个域对象）

Session也是存储数据的区域对象，所以session对象也具有如下三个方法：

```
session.setAttribute(String name,Object obj);
session.getAttribute(String name);
session.removeAttribute(String name);
```

5.3 Session对象的生命周期（面试题/笔试题）

创建：第一次执行request.getSession()时创建
销毁：
1）服务器（非正常）关闭时
2）session过期/失效（默认30分钟）

问题：时间的起算点 从何时开始计算30分钟？
从不操作服务器端的资源开始计时

可以在工程的web.xml中进行配置

```
<session-config>
        <session-timeout>30</session-timeout>
</session-config>
```

3）手动销毁session
session.invalidate();

作用范围：
默认在一次会话中，也就是说在，一次会话中任何资源公用一个session对象

面试题：浏览器关闭，session就销毁了？ 不对

5.4 注意

Session是基于Cookie的，如果不设置Cookie的持久化时间，则关闭客户端再打开并访问Session，取不到Session域对象内的内容，因为钥匙丢了。

```java
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        //创建属于该客户端(会话)的私有的session区域
        /* request.getSession()方法内部会判断 该客户端是否在服务器端已经存在session
         * 如果该客户端在此服务器不存在session 那么就会创建一个新的session对象
         * 如果该客户端在此服务器已经存在session 获得已经存在的该session返回
         */
        HttpSession session = request.getSession();

        session.setAttribute("name", "jerry");

        String id = session.getId();//该session对象的编号id

        //手动创建一个存储JSESSIONID的Cookie 为该cookie设置持久化时间
        Cookie cookie = new Cookie("JSESSIONID",id);
        cookie.setPath("/WEB16/");
        cookie.setMaxAge(60*10);

        response.addCookie(cookie);


        response.getWriter().write("JSESSIONID:"+id);

    }
```