Day01. 会话技术

1 会话技术

1.1什么是会话

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

会话中往往会产生一些数据，而这些数据往往是需要我们保存起来的，如何保存会话中产生的数据呢?

这里可以使用会话技术(也就是Cookie和session)来保存会话中产生的数据

1.2 Cookie

1.2.1 Cookie保存数据的原理

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!

1.2.2 Cookie的API

1、创建Cookie对象

```
Cookie c = new Cookie(String name, String value);
```

2、将Cookie添加到response响应中

```
response.addCookie(Cookie c);//可以调用多次，表示将多个cookie添加到响应中
```

3、获取请求中的所有cookie对象组成的数组

```
Cookie[] cs = request.getCookies();//该方法会返回请求中的所有cookie组成的数组，如果请求中没有任何cookie，该方法会返回null.
```

4、删除Cookie

-- 没有直接删除cookie的方法

我们可以向浏览器再发送一个同名的cookie(比如名称为prod的cookie)，并设置cookie的存活时间为零，最后将cookie发送给浏览器。

由于浏览器是根据cookie的名字来区分cookie的，如果前后发送了两个名称一样的cookie，后发送cookie会覆盖之前发送的cookie，又由于后发送的cookie生存时间为零，浏览器收到后也会立即删除！！

```
//1.创建一个名称为prod的cookie
Cookie c = new Cookie("prod", "");
//2.设置cookie的生存时间为零
c.setMaxAge( 0 );
//3.将cookie发送给浏览器
response.addCookie( c );
//4.响应浏览器
response.getWriter().write("delete Cookie success!!");
```

5、Cookie的常用方法

```
Cookie.getName();//获取cookie的名字
Cookie.getValue();//获取cookie中保存的值
Cookie.setValue();//设置/修改cookie中的值
Cookie.setMaxAge();//设置cookie的最大生存时间.
```

6、setMaxAge方法

-- 设置cookie的最大生存时间

如果不设置该方法，cookie生存时间默认是一次会话。Cookie默认保存在浏览器的内存中。当浏览器关闭（会话结束），随着内存的释放，cookie也会跟着销毁。

如果设置了该方法，cookie就会以文件的形式保存在浏览器的临时文件夹中(在硬盘上)，此时如果再关闭浏览器，内存释放了，但是硬盘上的cookie文件还在，当再次打开浏览器，硬盘上的cookie文件还可以再获取到，再发送给浏览器。

Cookie的细节总结：Cookie是从服务器发送给浏览器，发送后也会保存在浏览器内部，浏览器不会无限(苹果浏览器除外)的接收服务器发送的cookie，每一个浏览器对cookie的个数限制可能是不相同的(有的浏览器限制不超过20、有的浏览器限制不超过30个，有的限制不超过50个等)，并且每一个浏览器对cookie的大小也会有限制，一般是不超过4kb，个别浏览器限制不超过1kb。

所以在发送cookie时，最好个数不要超过20个，大小不要超过1kb。

1.2.3 案例：实现购物车

```
1.prodlist.html
<body>
    <h3>点击商品将商品加入购物车</h3>
    <p>
       <a href="/day10/CartServlet?prod=iphonexs">iphonexs</a>
    </p>
    <p>
       <a href="/day10/CartServlet?prod=huaweip20">huaweip20</a>
    </p>
    <p>
       <a href="/day10/CartServlet?prod=oppor11">oppor11</a>
    </p>
    <p>
       <a href="/day10/CartServlet?prod=vivox21">vivox21</a>
    </p>

    <h3>点击支付为购物车中的商品进行结算</h3>
    <a href="/day10/PayServlet">支付</a>
</body>
```

```
2.CartServlet
protected void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
    //1.获取要保存的商品信息(GET提交)
    String prod = request.getParameter("prod");

    //2.将商品以Cookie发送给浏览器保存
    //2.1.创建一个cookie对象,并保存商品到cookie中
    Cookie cookie = new Cookie("prod", prod);

    //>>设置cookie的最大存活时间
    cookie.setMaxAge(60*60*24*30);

    //2.2.将cookie添加到response响应中
    response.addCookie(cookie);

    //3.响应浏览器
    response.setContentType(
            "text/html;charset=utf-8");
    PrintWriter pw = response.getWriter();
    pw.write("<h1>成功将"+prod+"加入了购物车...</h1>");

}
```

```
3.PayServlet
protected void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
    //1.获取要进行结算的商品
    //1.1.获取所有cookie组成的数组
    Cookie[] cs = request.getCookies();
    //1.2.获取名称为prod的cookie
    String prod = null;
    if(cs != null){
       for( Cookie c : cs ){
           if( "prod".equals(c.getName()) ){
              prod = c.getValue();//商品
           }
       }
    }

    //2.模拟支付并响应浏览器
    response.setContentType(
            "text/html;charset=utf-8");
    response.getWriter().write(
            "<h1>成功为"+prod
            +"商品支付了1000.0$...</h1>");
}
```

1.3 Session

1.3.1 Session的工作原理

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。

1.3.2 Session是一个域对象

域：如果一个对象具有可以被访问的范围，利用该对象上提供的map集合可以在整个范围内实现数据的共享
提供的存取数据的方法：

```
Session.setAttribute(String name, Object value);
-- 添加域属性
Session.getAttribute(String name);
-- 获取域中的属性值
Session.removeAttribute(String name);
-- 删除域中的属性
```

域对象的三大特征：
生命周期：
(1)创建session：第一次调用request.getSession方法时创建session对象
(2)销毁session：
a）超时销毁:超过30分钟没有访问session，session就会超时销毁。

```
<session-config>
    <session-timeout>1</session-timeout>
</session-config>
```

b）自杀销毁:调用session的invalidate方法会立即销毁session
c）意外身亡:当服务器非正常关闭(断电、宕机等)时，session会随着服务器的关闭而销毁。如果服务器是正常关闭，session不会销毁。Session会以文件的形式保存在服务器的work目录下，这个过程叫做session的钝化（序列化）。如果服务器再次启动，钝化着的session还会再恢复回来（变成服务器中的对象），这个过程叫做session的活化（反序列化）！

作用范围：
整个会话范围内
主要功能：
在整个会话范围内实现数据的共享

1.3.3 案例:实现购物车

```
1.prodlist.html
<body>
    <h3>点击商品将商品加入购物车</h3>
    <p>
        <a href="/day10_session/CartServlet?prod=iphonexs">iphonexs</a>
    </p>
    <p>
        <a href="/day10_session/CartServlet?prod=huaweip20">huaweip20</a>
    </p>
```

```
        <p>
            <a href="/day10_session/CartServlet?prod=oppor11">oppor11</a>
        </p>
        <p>
            <a href="/day10_session/CartServlet?prod=vivox21">vivox21</a>
        </p>

        <h3>点击支付为购物车中的商品进行结算</h3>
        <a href="/day10_session/PayServlet">支付</a>
    </body>
```

```
    2.CartServlet
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        //1.获取要加入购物车的商品信息
        String prod = request.getParameter("prod");

        //2.将商品信息加入到session对象中保存
        HttpSession session = request.getSession();
        session.setAttribute("prod", prod);

        /* 我们可以向浏览器再发送一个名称为JSESSIONID
         * 的cookie，cookie的值就是session的ID，同时
         * 设置cookie的存活时间为有效值(比如1h)，设置
         * 后保存sessionID的cookie就不会存到浏览器的内
         * 存中,而是以文件的形式保存到硬盘上，此时即使
         * 浏览器多次开关，内存释放了，cookie也不会销毁
         * session的id也不会丢失，再次打开浏览器，还可以
         * 将session的id带回服务器，根据id找到之前的session
         * 从session中获取数据!!! */
        Cookie c = new Cookie(
                "JSESSIONID", session.getId());
        c.setMaxAge( 60*60 );//1h
        response.addCookie(c);


        //3.响应浏览器
        response.setContentType(
                "text/html;charset=utf-8");
        response.getWriter().write(
                "成功将"+prod+"商品加入了购物车...");
    }
```

```
    3.PayServlet
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        //1.获取session对象
        HttpSession session = request.getSession();
        //2.从session中获取要支付的商品
        String prod = (String)session
                      .getAttribute("prod");

        //3.模拟支付，并响应浏览器
        response.setContentType(
                "text/html;charset=utf-8");
        response.getWriter().write(
                "成功为"+prod+"商品支付了2000.0$...");
```

```
    }
```

1.4总结：Cookie和session的区别

1.4.1 Cookie的特点

1、cookie是将会话中产生的数据保存在浏览器客户端，是客户端技术
2、cookie是将数据保存在客户端浏览器，容易随着用户的操作导致cookie丢失或者被窃取，因此cookie中保存的数据不太稳定，也不太安全。
但是cookie将数据保存在客户端，对服务器端没有太多影响，可以将数据保存很长时间。
因此cookie中适合保存对安全性要求不高，但是需要长时间保存的数据。
3、浏览器对cookie的大小和个数都有限制，一般推荐每一个站点给浏览器发送的cookie数量不超过20个，每一个cookie的大小不超过1kb。
Cookie的应用：实现购物车、记住用户名、显示上次访问的时间、地点

1.4.2 Session的特点

1、session是将会话中产生的数据保存在服务器端，是服务器端技术
2、session将数据存在服务器端的session对象中,相对更加的安全,而且更加稳定。不容易随着用户的操作而导致session中的数据丢失或者是被窃取。
因此session中适合存储对安全性要求较高，但是不需要长时间保存的数据。
Session的应用：保存登录状态