5 会话跟踪

实际应用中，Web App 都是由多个页面构成的，并由多名用户访问。Web App 中部分页面负责获取用户请求的数据，由于 Web Server 的“健忘性”，当用户浏览器发出一个对某种 Web 资源的请求后，Web Server 会处理并返回一个响应，但之后 Web Server 会忘记这个曾与其交互的客户端浏览器，即便同一个浏览器很快又再次发送一个新的请求，服务器也不会知道前后两个请求之间的联系。在多用户并行访问的情况下，这个问题尤为突出，怎样使 Web Server 记住客户的信息，判断某个请求是否属于同一用户浏览器提交，这便是会话跟踪技术需要解决的问题。

学习完本章，您能够:

- 了解会话跟踪的 4 种常用技术。
- 掌握 Cookie 技术。
- 全面掌握 session 技术。
- 熟悉 session 的生命周期。

5.1 实例引入

CHERRYONE 的测试员指出，登录系统应该能提供一个便捷的途径，能够让用户在登录时做出选择，后续登录如果在同一个客户端浏览器上，系统应能让客户直接访问相关网页，免除再次登录的操作，同时对之前用户注册信息中选择的国家、语言习惯进行记录，在用户的访问页面上得到相关显示。

新原型在原有功能的前提下，需要改进的功能如下：

- 通过一种方式记录用户信息。
- 让同一个客户端访问的用户再次登录时直接进入网站。
- 记录用户的语言习惯，在用户浏览的页面中显示。

5.2 会话跟踪简介

从客户端浏览器发出请求并得到服务器响应，到服务器中断与该浏览器的连接或浏览器关闭的通讯全过程称为会话。会话的实现必须建立在服务器保存浏览器的“记忆”的基础之上，这需要在浏览器和服务器之间来回发送状态信息；为了让服务器能够追踪用户浏览器的状态，提高请求及响应的效率，把信息保留在服务器端，只在浏览器和服务器之间传递标识符，这就是会话跟踪的实现方案。换句话说，会话跟踪就是当一个客户在 Web App 的多个页面间切换时，Web Server 会保存该客户的唯一信息。

5.2.1 有状态和无状态

有状态（Stateful）和无状态（Stateless）是众多 Internet 协议中对服务器端和客户端状态的两种不同表现类型。

有状态会话表示在客户端和服务器端连接后，会保持一种相同且持续性的联机状态，客户端和服务器在这种互联下完成各项操作，待本次会话的操作结束，服务器关闭与客户端的连接。在有状态的会话协议支持下，若多个客户端和服务器进行通讯，服务器会记住每个客户端的信息，并为不同的客户保持这些持续性的连接状态直到会话结束。常用的 Telnet 协议和 FTP 协议均属于有状态协议。

无状态会话则表示客户端提交请求时，服务器端才建立连接并根据请求内容响应，但是服务器端并不会维持和客户端的联机状态，一次请求和一次响应构成一个独立的事务[1]，一次事务结束之后，服务器便抹去客户端的信息，使得即使同一个客户端，不同事务之间没有状态联系。JSP 所依赖的 HTTP 协议便是无状态协议。当然，对于服务器而言，无状态协议大大减轻了服务器的负载压力，是实际可行的，所以就需要 JSP 这种服务器端脚本通过某种机制来维持服务器端对客户端浏览器的“记忆”，这就是会话跟踪。

下面通过一个较为形象的例子来阐述会话跟踪的重要性。某公司的秘书每天要接听公司各级部门的电话，每个电话的内容都很重要，但由于电话繁多，这名秘书的记性欠佳，出现了放下电话就忘记之前电话内容的情况，这给公司造成了非常大的损失，第一名秘书被解雇。第二名秘书在工作的时候准备了很多卡片，每张卡片按各部门编号。每接到一个电话，第二名秘书就把打电话的部门按照编号选出对应的卡片，并把内容记录在卡片上。由此一来，公司每天的电话内容都记录了下来，这名秘书能够准确地向公司领导汇报各部门的信息。

希望这个例子对读者有一定的启示。

注释：[1]事务：执行过程中的一个逻辑单位，由一个有限的操作序列构成。事务处理被广泛地应用于数据库和操作系统领域。

5.2.2 4 种会话跟踪的方式

服务器端技术一般通过两种方式来实现会话跟踪，一种是既让服务器在每个响应中返回与当前用户相关的所有客户端状态信息，又让浏览器将接收到的信息作为下一次请求中的一部分内容再次提交给服务器；另一种是在服务器端中保存客户端状态，仅仅把一个唯一的标识符附加在响应中发给客户端，浏览器在下一次请求中将继续提交该标识符，服务器通过接收该标识符来定位保存在服务器端的用户状态信息。

由于在浏览器和服务器之间来回传递所有的状态信息效率非常低下，所以如今大部分服务器端技术都采用将信息保存在服务器上，只在浏览器和服务器之间传递标识符的第二种方法。来自一个客户端浏览器的所有请求均包含被服务器端赋予并属于同一个会话的唯一标识符，又称为会话 ID，服务器根据会话 ID 来跟踪与对应会话相关联的所有信息。

会话跟踪技术，一般用下面 4 种方式实现：

- URL 重写（URL Rewriting）
- 隐藏表单域（Hidden Form Field）
- 持久性 Cookies（Persistent Cookies）
- Servlet 的 HttpSession 接口或 JSP 的 session 对象

URL 重写是把会话 ID 附加在 JSP 页所创建的 URL 中，即把会话 ID 作为额外的请求或响应参数添加到 URL 尾部。

一个附加了会话 ID 的 URL 形式如下：

```
http://localhost:8084/Chapter5d/lesson5d1.jsp;jsessionid=E44F6CAE1097C822580B5F0DAD4CF9D1
```

“:”后的 jsessionid 便是会话 ID 的参数名，而“=”后面的值则是服务器分配的会话 ID，jsessionid 是 Apache Tomcat 服务器对会话 ID 的写法，不同的服务器对会话 ID 有不同的写法，它们统称 sessionid 或会话 ID。

服务器接收到使用 URL 重写方式进行封装的请求信息时，会从 URI 中提取出会话 ID，并把该请求和相应的会话关联起来。JSP 技术中通过使用 response 对象中的 encodeURL()方法或 encodeRedirectURL()方法来实现 URL 重写。

例 5-1 通过 URL 重写保存用户输入。

lesson5d1.jsp 实现代码：

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>lesson5d1</title>
    </head>
    <body>
```

```
<h2>URL Rewriting</h2>
<form>
    <table border="1">
        <thead>
            <tr>
    <th colspan="2"><label for="encoded">encoded link</label></th>
            </tr>
        </thead>
        <tbody>
            <tr>
                <td>User Name:&amp;nbsp;&amp;nbsp;&amp;nbsp;</td>
                <td><input type="text" name="username" value="" style="width:200px"/></td>
            </tr>
            <tr>
                <td>Password:</td>
                <td><input type="password" name="password" value="" style="width:200px"/></td>
            </tr>
        </tbody>
    </table>
    <input type="submit" value="SUBMIT" name="submit" />
</form>
<%
    String action = request.getParameter("submit");
    String username = request.getParameter("username");
    String password = request.getParameter("password");
    String url = "lesson5d1.jsp?username=" + username + "&password=" + password;
    if (action != null) {
        url = response.encodeRedirectURL(url);
        out.println(url);
        response.sendRedirect(url);
    }
%>
<hr/>
<table border="1">
    <thead>
        <tr>
            <th style="width:300px" align="left">
                SessionId is: <%=request.getRequestedSessionId()%>
            </th>
        </tr>
    </thead>
    <tbody>
```

```
                    <tr>
                        <td>User Name is: <%=request.getParameter("username")%></td>
                    </tr>
                    <tr>
                        <td>Password is: <%=request.getParameter("password")%></td>
                    </tr>
                </tbody>
            </table>
        </body>
    </html>
```

页面执行结果如图 5-1 所示。

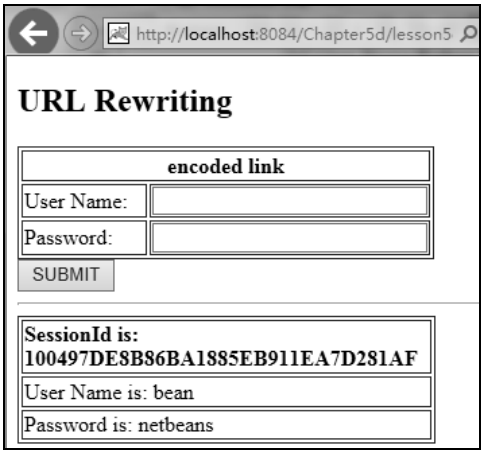

图 5-1 带有会话 ID 的 URL Rewriting 执行结果

例 5-1 中，使用 response 对象的 encodeRedirectURL()方法实现对跳转页面 URL 的封装，同时把传递的参数 username 和 password 通过 GET 方式附加在页面 URL 之后；由于通过服务器响应对象对 URL 进行重写，生成了会话 ID，在页面跳转之后，这个会话 ID 会发送到客户端浏览器上，并在下一次的客户端请求信息中被附加，由服务器接收并识别，由此来保持服务器和对应客户端之间的状态。

但是 URL Rewriting 会把数据暴露在浏览器的地址栏上，使得网页的安全性存在较大的漏洞。

隐藏表单域通过<input>标签的 hidden 属性把需要传送到服务器的数据在用户无法觉察的情况下存入 request 请求对象中一并提交。由于 hidden 属性让保存其中的数据不在页面上显示，并通过 request 提交，不会像 URL 重写那样把会话数据全暴露在 URL 上，具有一定的安全性；隐藏表单域传输数据的方式和<input>标签的 text 文本基本一致，使用方便。但是隐藏表单域属于 HTML 元素，在 JSP 中被识别为模板文本，在源码中并不会被隐藏，所以只要用户查看表单页的源文件，就能准确地找到保存在 hidden 属性中的数据，从而造成安全隐患。

例 5-2　用隐藏表单域传递用户 id。

简要分析：通过 NetBeans 向表单中插入隐藏表单域非常方便，只需要在屏幕右边的“组件”面板中选择“HTML 窗体”中的“文本输入”，“组件”面板可以通过“窗口”菜单打开，或者使用快捷键 Ctrl+Shift+8 打开。隐藏表单域的使用方式如图 5-2 所示。

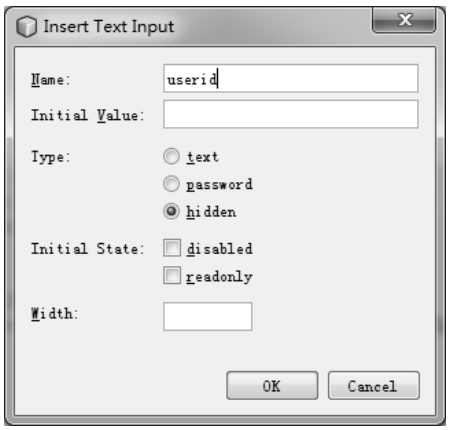

图 5-2　通过 NetBeans 插入隐藏表单域

下面给出实现代码。

lesson5d2.jsp 用户输入界面代码：

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>JSP Page</title>
    </head>
    <%!String uid;%>
    <%!
        public void genId(){
            int tmpid=new java.util.Random().nextInt();
            uid="u" + tmpid;
        }
    %>
    <body>
        <form action="lesson5d2_show.jsp" method="POST">
            <table border="1">
                <tbody>
                    <tr>
                        <%genId();%>
                        <input type="hidden" name="userid" value="<%=uid%>" />
```

```
                    <td>User Name: </td>
                    <td><input type="text" name="username" value="" /></td>
                </tr>
            <tr>
                <td>Password: </td>
                <td><input type="password" name="password" value="" /></td>
            </tr>
            <tr>
                <td colspan="2">
                    <input type="submit" value="SUBMIT" name="submit" /><input type="reset"
                    value="RESET" name="reset" />
                </td>
            </tr>
            </tbody>
        </table>
    </form>
</body>
</html>
```

lesson5d2_show.jsp 显示用户信息页面代码：

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>JSP Page</title>
    </head>
    <body>
        <h1>Hello <%=request.getParameter("username")%>...</h1>
        Ur user id is: <%String uid=request.getParameter("userid");%><%=uid%><br/>
        Ur password is:<%=request.getParameter("password")%><br/>
    </body>
</html>
```

例 5-2 的运行结果如图 5-3 所示；输入用户名和密码并提交后，结果如图 5-4 所示。

图 5-3　带有隐藏表单域的用户输入界面

图 5-4　显示隐藏表单域提交的用户 id

在图 5-3 中，并没有看到通过<input type="hidden" name="userid" value="<%=uid%>"/>在用户输入界面创建的隐藏域，证明隐藏表单域相较 URL 重写完全暴露请求信息安全性得到了一定的改善。但客户端浏览器能够把用户输入界面的 HTML 源码显示出来，造成了隐藏域数据的暴露，如图 5-5 所示。

```
<body>
    <form action="lesson5d2_show.jsp" method="POST">
        <table border="1">
            <tbody>
                <tr>
                    <input type="hidden" name="userid" value="n1697921495" />
                    <td>User Name: </td>
                    <td><input type="text" name="username" value="" /></td>
                </tr>
                <tr>
                    <td>Password: </td>
                    <td><input type="password" name="password" value="" /></td>
                </tr>
                <tr>
                    <td colspan="2">
                        <input type="submit" value="SUBMIT" name="submit" /><input type
                        ="reset" value="RESET" name="reset" />
                    </td>
                </tr>
                </tbody>
        </table>
    </form>
</body>
```

图 5-5　用户输入界面的 HTML 源码暴露了隐藏域的数据

持久性 Cookies 的运行方式与 URL 重写和隐藏表单域不尽相同。Cookie 最先是由网景（Netscape）公司开发，通过“名称”、“值”对称的形式将会话数据保存在客户端目录下一个很小的文本文件中。Cookie 被 Web Server 作为 HTTP 响应头标的一小部分发送给正访问它的客户端浏览器，然后由客户端浏览器将头标中有关 Cookie 的信息生成一个文本文件，并保存起来；当浏览器再次访问该服务器时，对应的 Cookie 便会随着请求原样提交给服务器，让服务器读取 Cookie 中的“名称”、“值”信息来确认用户。

持久性 Cookies 的最大优势是这些 Cookie 保存在客户端的硬盘上，会话结束后，甚至客户端计算机关闭或重启之后会话数据仍然保留。这使得 Cookie 在服务器网站和客户端浏览器之间搭建了一条快捷通道，在安全要求不高的场合，Cookie 能实现诸如避免用户再次输入用户名和密码的情况下进入曾经浏览过的站点，提高用户和服务器网站之间的访问效率。

Cookie 一般通过名字、值、过期时间、路径和域等属性来保存用户信息。路径与域一起构成 Cookie 的作用范围；过期时间默认为-1（s），表示该 Cookie 的生存期为浏览器与服务器的会话期间，关闭浏览器窗口，Cookie 就消失。这种生命周期为浏览器会话期的 Cookie 被称为临时 Cookies 或会话 Cookies。

持久性 Cookies 的优势也造成了它的安全隐患，由于持久性 Cookies 保存在客户端硬盘上，如果在一台公共场合的计算机上使用 Cookies，很大几率会出现不同时间段的不同用户访问同一站点的情况，而这导致后来的访问者可以不用键入用户名和密码便可以在这个站点上享用和之前的访问者同样的权利和个人设置，因为 Cookie 并不会区分用户，它只会根据要访问的服

务器站点选择对应的 Cookie 并发送请求，而服务器站点也只通过接收到的 Cookie 来确认访问者的信息，这便使得之前用户的个人信息完全曝光。所以，特定场合的计算机可以通过浏览器的设置关闭 Cookie 来保证个人隐私的安全性。

下面通过一个形象的例子来阐述 Cookie 实现会话跟踪的优势和可能出现的问题。

一家咖啡店有喝 5 杯咖啡免费赠送一杯咖啡的优惠，然而一次性消费 5 杯咖啡的机会微乎其微，这时就需要某种方式来记录某位顾客的消费数量。该咖啡店使用会员卡的方式来记录顾客所消费的咖啡数量，在发给顾客的卡片上记录着顾客的姓名、消费的数量及卡片的有效期限。每次消费时，如果顾客出示这张卡片，咖啡店的工作人员会把此次消费与以前或以后的消费相联系起来，达到 5 杯，则免费赠送一杯。这种把卡片留在顾客身上的做法就是在客户端保存 Cookie，借此保持状态。

上面这种做法的优势和弊端一目了然，优势是只要顾客出示该咖啡店的会员卡，店员根据卡片上记录的数量来判断是否免费赠送一杯咖啡；弊端是由于在客户身上的卡片容易遗失，并且会被别有用心的顾客修改，会造成双方的损失。

下面来研究一下 Web Server 实现 Cookie 的整个步骤。

第一步：创建 Cookie。

Servlet 提供对 Cookie 的支持，通过导入 javax.servlet.http.Cookie 类，并对其构造方法 Cookie(String name,String value)进行初始化。语法如下：

```
Cookie acookie=new Cookie("Key","Value");
```

注意：在 Cookie 的构造方法中，最初网景公司的 Cookie 版本不支持在两个参数均包含“@”、“:”、“;”、“?”、“"”、“/”、“[”、“]”、“(”、“)” 和 “=” 等特殊字符，但在较新的 RFC 2109 文档制定的版本中放宽了限制，为保险起见，读者还是尽量避免使用这些特殊字符，同时 Cookie 的名字在创建之后是不能修改的，即 Cookie 不支持重命名。

第二步：设置 Cookie。

Cookie API 通过类似 JavaBean 中 setter 的方式对各种属性进行赋值，Cookie 常用设置属性的方法如表 5-1 所示。

表 5-1 设置 Cookie 属性的方法

方法名	描述
setComment(String purpose)	设置 Cookie 的注释，注释表示 Cookie 的设计意图
setDomain(String pattern)	设置 Cookie 的域名，需要跨域共享 Cookie 时使用
setMaxAge(int expiry)	以秒为单位，设置 Cookie 的过期时间
setPath(String uri)	指定客户端应返回的 Cookie 路径
setSecure(boolean flag)	指出浏览器应使用的安全协议，如 HTTPS 或 SSL
setValue(String newValue)	Cookie 创建后为其设置一个新的值
setVersion(int v)	设置 Cookie 所遵从的协议版本

第三步：发送 Cookie。

JSP 通过 response 隐式对象调用 addCookie(Cookie acookie)方法将创建好的 Cookie 对象插入到 HTTP 响应的 Set-Cookie 头标中发送给客户端浏览器。假设 Cookie 对象名为 aCo，发送 Cookie 的代码如下：

```
response.addCookei(aCo);
```

第四步：读取 Cookie。

客户端浏览器再次访问服务器时会把 Cookie 对象附加在请求中提交到服务器，服务器使用 request 隐式对象调用 getCookies()方法读入 Cookie 并获取其相关属性。Cookie 常用获取属性的方法和 JavaBean 的 getter 方法类似，如表 5-2 所示。

表 5-2　获取 Cookie 属性的方法

返回类型	方法名	描述
String	getComment()	返回 Cookie 中的注释，如果没有注释的话将返回空值
String	getDomain()	返回当前 Cookie 的域名
int	getMaxAge()	返回 Cookie 的使用期限，以秒为单位
String	getName()	返回 Cookie 的名字
String	getPath()	返回 Cookie 的路径，如果不指定路径，Cookie 将返回给当前页面所在目录及其子目录下的所有页面
boolean	getSecure()	如果浏览器通过安全协议发送 Cookie 将返回 true 值，如果浏览器使用标准协议则返回 false 值
String	getValue()	返回 Cookie 的值
int	getVersion()	返回 Cookie 所遵从的协议版本

第五步：删除 Cookie。

Cookie 类中并无专门删除 Cookie 对象的 API，删除不再使用的 Cookie 只需要将新建的 Cookie 对象的 name 属性设置和需要删除的 Cookie 对象同名，value 属性设为 null，并将其存在时间设为“0”即可，即 setMaxAge(0)。假设需要删除的原 Cookie 对象名为 user，删除 Cookie 对象的代码如下：

```
Cookie delCo=new Cookie("user",null);
delCo.setMaxAge(0);
response.addCookie(delCo);
```

并不是所有的 Cookie 类的对象都是持久性地保存在客户端的硬盘目录中，还有一类 Cookie 对象称为临时 Cookie。临时 Cookie 在浏览器和服务器交互时保存在浏览器的缓存中，当浏览器关闭时，临时 Cookie 便随之消失。

例 5-3　利用 Cookie 保存用户对某站点提交的用户名和密码。

简要分析：

（1）用户在登录页面（lesson5d3.jsp）输入用户名和密码后单击“提交”按钮进入站点的确认页面（lesson5d3_recCo.jsp）。

（2）在确认页面中需要判断用户登录时是否选择了“Cookie 保存 10 天”复选框，如果选择了，则需要创建 Cookie 对象，Cookie 对象中保存用户信息，并设置 Cookie 对象的过期时间为 3600*240 秒，设置 Cookie 对象的路径为“\”，表示当前 Web 应用的根目录；如果未选择，则设置 Cookie 对象的过期时间为 0 秒，删除 Cookie 对象。

（3）在确认页面上设置一个跳转链接，返回登录界面。此时需要在登录界面上实现 Cookie 的读取，然后把 Cookie 中的用户信息取出，重新填入用户名和密码的输入框中。

下面给出实现代码。

lesson5d3.jsp 用户登录界面代码：

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<%!boolean blNew = true;%>
<%
    Cookie[] cookies = request.getCookies();
    if (cookies != null) {
        for (Cookie aCo : cookies) {
            if (aCo.getName().equals("user")) {
                String usernameByCo = aCo.getValue().split("-")[0];
                String passwordByCo = aCo.getValue().split("-")[1];
                request.setAttribute("username", usernameByCo);
                request.setAttribute("password", passwordByCo);
                blNew = false;
                break;
            }
        }
    }
%>
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>JSP Page</title>
    </head>
    <body>
        <form action="lesson5d3_recCok.jsp" method="POST">
            <table border="1">
                <tbody>
                    <tr>
                        <td>User Name: </td>
```

```
                    <td><input type="text" name="username" value="${username}" /></td>
                    </tr>
                    <tr>
                        <td>Password: </td>
                        <td><input type="password" name="password" value="${password}" /></td>
                    </tr>
                    <tr>
                        <td colspan="2">
                        Cookie is Remembered 10 days<input type="checkbox" name="isMem" value="ON" />
                        </td>
                    </tr>
                    <tr>
                    <td><input type="reset" value="RESET" name="reset" /></td>
                    <td><input type="submit" value="SUBMIT" name="submit" /></td>
                    </tr>
                </tbody>
            </table>
        </form>
    </body>
    <%
        String username = "";
        if(blNew==false){
            username = request.getAttribute("username").toString();
            out.println("<h1>Welcome Back " + username + "</h1>");
        }
    %>
</html>
```

用户信息确认页面 lesson5d3_recCo.jsp 代码：

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<%
    String ischeck = request.getParameter("isMem");
    String username = request.getParameter("username");
    String password = request.getParameter("password");
    if (ischeck != null && ischeck.equals("ON")) {
        //表示用户单击了 remember 按钮
        //创建 Cookie
        Cookie userCo = new Cookie("user", username + "-" + password);
        userCo.setMaxAge(60 * 60 * 24 * 10);
        userCo.setPath("/");
        response.addCookie(userCo);
    } else {
        Cookie deleteNewCookie = new Cookie("user", null);
```

```
            deleteNewCookie.setMaxAge(0);       //删除该 Cookie
            deleteNewCookie.setPath("/");
            response.addCookie(deleteNewCookie);
        }
    %>
    <!DOCTYPE html>
    <html>
        <head>
            <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
            <title>JSP Page</title>
        </head>
        <body>
            <h1>Hello <%=username%></h1>
            <h2>Welcome to out page,thank u !</h2>
            <a href="lesson5d3.jsp">Login Again...</a>
        </body>
    </html>
```

例 5-3 用户登录界面的运行结果如图 5-6 所示；提交之后，用户信息确认页面如图 5-7 所示。

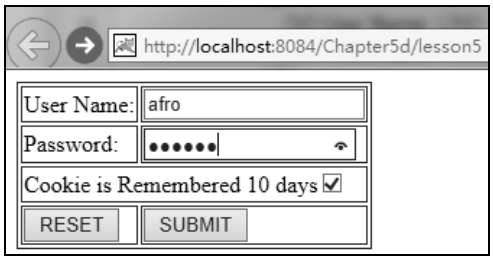

图 5-6 用户第一次输入

图 5-7 用户确认页面

点击链接返回登录页面，或再次运行用户登录界面，结果如图 5-8 所示。

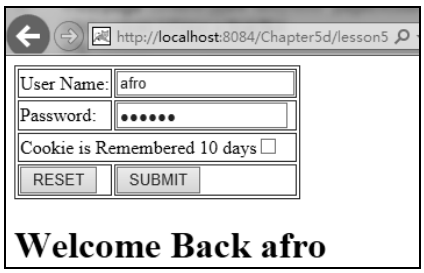

图 5-8 通过 Cookie 检索，用户信息直接显示

HttpSession 会话机制融合了 Cookie 和 URL 重写技术。当客户端允许使用 Cookie 时，HttpSession 会使用 Cookie 进行会话跟踪。通过 request 对象调用 getCookies()方法会返回一个 Cookie 数组，Cookie[0]中的 name 属性值即为"JSESSIONID"，由此来判断来自客户端浏览器的请求是否为同一个人。当客户端不支持 Cookie 时，HttpSession 会利用 response 对象的 encodeURL()或 encodeRedirectURL()方法，即 URL 重写来附加"JSESSIONID"实现会话跟踪。值得庆幸的是，HttpSession 接口封装了处理 Cookie 或

URL 重写的细节问题，并为开发者提供保存会话信息的区域。

关于 Session 的详细内容请参见 5.3 节。

5.3 session

JSP 封装了 Servlet 的 HttpSession 接口，通过隐式对象 session 实现该接口的所有功能，这使得在 JSP 中操作 session 变得更为简捷。

当浏览器第一次请求 JSP 页的时候，Web Server 会自动为该浏览器创建一个 session，并赋予一个 sessionID 发送给客户端浏览器。当客户端再次请求 Web Server 中应用程序的其他资源的时候，会自动在 HTTP 请求头标中添加从服务器得到的 session ID；当服务器端继续接到客户端请求时，就会一并接收 sessionID，并根据 sessionID 在内存中找到之前创建的 session 对象的引用，提供给请求使用。

在 JSP 中，session 几乎成了会话的代名词，5.2.2 节提到 session 是 Cookie 和 URL 重写两种技术的融合体，同时它也是一个容器，用来存储会话过程中需要保留的对象。

5.3.1 创建 session

在 Servlet 中，HttpSession 接口通过 HttpServletRequest 的对象 request 调用 getSession()或 getSession(boolean create)方法来创建会话。无参的 getSession()是调用 getSessionj(true)的一种简便写法；参数 create 如果为 false，Servlet 引擎不会创建新的会话，只对存在的会话验证其会话 ID，并执行相关操作，而为 true 时，则创建一个新的会话。

在 JSP 页中，session 作为一个内置对象，会自动创建会话，除非通过设置 page 指令中的 session 属性为 false 来禁止当前 JSP 页的会话。

session 对象中封装的 Servlet 代码段如下：

```
final javax.servlet.jsp.PageContext pageContext;
javax.servlet.http.HttpSession session = null;
pageContext = _jspxFactory.getPageContext(this, request, response,
                        null, true, 8192, true);
_jspx_page_context = pageContext;
      session = pageContext.getSession();
```

5.3.2 使用 session

session 的创建和使用均在 Web Server 上，客户端浏览器从未获取过 session 对象，浏览器获取的仅仅是会话 ID，所以会话对于用户而言往往是不可见的，当然用户也无需关心自己是否处于会话中或者处于哪一个会话中。

session 对象的常用方法如表 5-3 所示。

表 5-3 session 的常用方法

返回类型	方法名	描述
Object	getAttribute(String name)	返回 session 中指定 name 的属性值
Enumeration	getAttributeNames()	返回 session 中绑定的所有属性名称
long	getCreationTime()	返回 session 的创建时间
String	getId()	返回 sessionID
long	getLastAccessedTime()	返回客户端最后请求的时间，单位为毫秒
int	getMaxInactiveInterval()	获取 session 最大无响应时间，单位为秒
ServletContext	getServeltContext()	返回 session 所属的 Servlet 上下文
void	invalidate()	强制会话无效
boolean	isNew()	判断服务器是否第一次为客户端创建 session，是则返回 true
void	removeAttribute(String name)	移除指定 name 的对象
void	setAttribute(String name,Object value)	通过指定 name 把对象值保持在 session 中
void	setMaxInactiveInterval(int interval)	指定客户端请求和 JSP 容器间的最大无交互时间，以秒为单位，到指定时间使 session 无效

会话跟踪技术中，session 内置对象在内存中有独立的存储空间用来存储相关对象，在会话中保存对象的方法为 setAttribute()，通过提供对象在会话作用域 sessionScope 的名称来保存对应的对象内容。保存在 session 作用域中的对象名可以是任意字符串，在相同的作用域范围内对象名必须唯一，而在不同的作用域中则不存在名称的限制，即可以出现同名的对象。保存在会话中的对象可以是任意类型的对象，也可以是基础类型，如 int、char 等，因为在 JDK1.5 中便提出了基础类和包装类的自动转化，即基础类作为对象值保存的时候会被自动转化为对应的包装类。而在 session 的作用域范围检索对象则通过 getAttribute()方法实现，给出通过 setAttribute()指定的对象名，如果对象存在，即可获取对象值。

例 5-4 在 session 作用域范围存储和检索对象。

简要分析：在 request 和 session 的作用域范围内为各种保存的对象创建相同的名字，通过<jsp:forward>指令跳转到另外一页进行检索，把获取的值分别显示并比较。

lesson5d4.jsp 对象保存在会话作用域中。

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
    <head>
```

```
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>JSP Page</title>
    </head>
    <body>
        <h1>Save Objects in [session...request]</h1>
        <%
            session.setAttribute("afro", 55);
            request.setAttribute("afro", 22);
        %>
        <jsp:forward page="lesson5d4_retrieval.jsp"/>
    </body>
</html>
```

lesson5d4_retrieval.jsp 检索保存在 requestScope 和 sessionScope 中的同名对象。

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>JSP Page</title>
    </head>
    <body>
        <%
            Integer afro_fromReq=(Integer)request.getAttribute("afro");
            Integer afro_fromSes=(Integer)session.getAttribute("afro");
            %>
            <h1>Object is saved in Request Scope is:<font style="color: red"> <%=afro_fromReq%>
            </font></h1>
        <h1>Object is saved in Session Scope is: <font style="color: blueviolet"><%= afro_fromSes%>
        </font></h1>
    </body>
</html>
```

例 5-4 的运行结果如图 5-9 所示。

图 5-9　检索保存在 session 中的对象

再来一个稍微复杂点的例子，很多教材都以购物车作为会话跟踪的例子来强调 session 的作用，本书也撰写了一个简单的购物车应用，借此来阐述 session 在其作用域中对对象进行临时保存，实现会话跟踪的过程。

例 5-5 用 JSP 和 JavaBean 实现简单购物车。

简要分析：电子商务平台的购物车是从超市的购物篮抽象出来的，通过购物篮临时存储购物时在不同时间（购物时间段内）、不同位置（超市的各购物区）买到的商品，结账时取出购物篮中的商品，在收银台统一付款。JSP 中的购物车便通过 session 提供的会话跟踪技术实现了类似超市购物篮的功能。

整个 Web 应用共 5 个文件、3 个 JSP 页面（分别实现商品展示、购物车管理和商品移除）和 2 个 JavaBean（一个实现购买商品名称和数量的统计，另一个实现字符编码的转化）。

文件中有较为详细的注释来帮助读者掌握该 Web 应用的知识点。

下面给出实现代码。

商品展示页面 lesson5d5_productSelect.jsp 代码：

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>欢迎光临</title>
    </head>

    <body>
        <div align="center">
            <h1>欢迎光临购物车</h1>
            <h2>现在时间是：<font style="color:blueviolet"><%=new java.util.Date()%></h2>
            <form method="post" action="lesson5d5_cart.jsp" target="" >
                <table width="80%" border="1">
                    <tr>
                        <td width="50%" height="30" align="right">请选择您要购买的商品：</td>
                        <td width="50%" height="30" align="left"> 
                            <select name="goodsName">
                                <option value="笔记本" selected>笔记本</option>
                                <option value="冰箱">冰箱</option>
                                <option value="洗衣机">洗衣机</option>
                                <option value="电视机">电视机</option>
                                <option value="自行车">自行车</option>
                                <option value="打印机">打印机</option>
                                <option value="空调">空调</option>
                                <option value="音响">音响</option>
```

```
                                    </select></td>
                            </tr>
                            <tr>
                                <td width="50%" height="30" align="right">购买数量：</td>
                                <td width="50%" height="30" align="left"> 
                                <input type="text" name="goodsNumber" value="1" size="5"></td>
                            </tr>
                        </table>
                        <p><input type="submit" name="sub" value="提交">  
                            <input type="reset" name="res" value="重购"></p>
                    </form>
                </div>
            </body>
        </html>
```

购物车管理页 lesson5d5_cart.jsp 代码：

```
<%@page import="java.util.Enumeration"%>
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<%@page import="java.util.Hashtable"%>
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>欢迎光临购物车</title>
        <jsp:useBean id="prodbean" scope="session" class="org.me.afro.Product"/>
        <jsp:useBean id="transbean" class="org.me.afro.Translate"/>
    </head>
    <jsp:setProperty name="transbean" property="goodsName"/>
    <%
        //获取所要添加到购物车的商品名称和数量并转化编码
        String sGoodsName = transbean.getGoodsName();
    %>
    <%
        String sGoodsNumber = request.getParameter("goodsNumber");
        //根据商品名称是否为空判断是否需要保存商品信息
        if (sGoodsName != null && sGoodsNumber !=null) {
            int iGoodsNumber = Integer.parseInt(sGoodsNumber);
            prodbean.add(sGoodsName, iGoodsNumber);
        }
        //获取购物车对象信息
        Hashtable h = prodbean.show();
        //获取购物车中所有的商品名称
        Enumeration e = h.keys();
```

```
            //keys()，返回此哈希表中的键的枚举
    %>
        <body>
            <div align="center">
                <h1>欢迎光临购物车</h1>
                <p>您的购物信息如下：</p>
                <table width="80%" border="1">
                    <%
                        //循环显示购物车中的商品信息
                        while (e.hasMoreElements())    //hasMoreElements()，测试此枚举是否包含更多的元素
                        {
                            //根据商品名称获得相应商品数量
                            String sTemp = e.nextElement().toString();
                            int iTemp = ((Integer) h.get(sTemp)).intValue();
                    %>
                    <tr>
                        <td width="50%" height="25" align="right"><font color="#0000FF"><%=sTemp%>:
                        </font></td>
                        <td width="20%" height="25" align="left"> <font color="#FF0000"><%=
                        iTemp%></font></td>
                        <td width="30%" height="25" align="left"> 
                            <input type="button" name="GoodsName" value="删除"
                                onClick="javascript:window.location = 'lesson5d5_removeProduct.jsp?
                                goodsName=<%=sTemp%>';"></td>
                    </tr>
                    <%
                        }
                    %>
                </table>
                <p><input type="button" name="goon" value="继续购物" onClick="javascript:
                    window.location = 'lesson5d5_productSelect.jsp';"></p>
            </div>
        </body>
    </html>
```

商品移除页 lesson5d5_removeProduct.jsp 代码：

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>欢迎光临</title>
        <jsp:useBean id="prodbean" scope="session" class="org.me.afro.Product"/>
```

```
    </head>
    <body>
        <div align="center">
            <h1>欢迎光临购物车</h1>
            <%
                System.out.println(request.getParameter("goodsName"));
                //获取所要删除的商品名称并转化编码
                String sGoodsName = request.getParameter("goodsName");
                System.out.println(sGoodsName);
                //删除对应的商品信息
                prodbean.delete(sGoodsName);
                //跳转当前页面
                response.sendRedirect("lesson5d5_cart.jsp");
            %>
        </div>
    </body>
</html>
```

实现购买商品名称和数量统计的 JavaBean——Product 代码：

```
package org.me.afro;
import java.util.Hashtable;
import java.io.*;
public class Product implements Serializable{
 public Hashtable product = new Hashtable();
    //构造函数
    public Product() {
    }
    //将某个商品信息加入购物车
    public void add(String productName, int productNumber) {
        if (product.containsKey(productName))      //containsKey，测试指定对象是否为此哈希表中的键
        {    //购物车中存在此商品
            int iTemp = ((Integer) product.get(productName)).intValue();
            //intValue()，以 int 类型返回该 Integer 的值
            iTemp = iTemp + productNumber;
            product.put(productName, new Integer(iTemp));
        } else {//购物车中不存在此商品
            product.put(productName, new Integer(productNumber));
        }
    }
    //获取购物车中所有的商品信息
    public Hashtable show() {
        return product;
    }
```

```
        //从购物车中删除一件商品信息
        public void delete(String productName) {
            product.remove(productName);
        }
}
```

字符集转化的 JavaBean——Translate 代码：

```
package org.me.afro;
import java.io.UnsupportedEncodingException;
public class Translate {
        public Translate() {
        }
        private String goodsName;

        /**
         * @return the GoodsName
         */
        public String getGoodsName() {
            this.goodsName = transCoding();
            return goodsName;
        }

        private String transCoding() {
            String encodeStream = null;
            try {
                if (this.goodsName != null) {
                encodeStream = new String(this.goodsName.getBytes("ISO8859-1"), "UTF-8");
                }
            } catch (UnsupportedEncodingException uee) {
                uee.printStackTrace();
            } catch (Exception e) {
                e.printStackTrace();
            }
            return encodeStream;
        }

        /**
         * @param GoodsName the GoodsName to set
         */
        public void setGoodsName(String goodsName) {
            this.goodsName = goodsName;
        }
}
```

由于字符串对象从客户端传递到服务器端，接收时默认的编码形式是 ISO8859-1，这是 8 位字符集，不支持亚洲字符，所以需要在接收时对其进行重新编码，即把原有 ISO8859-1 格式的字符串转化为字节数组，再通过中文字符集进行编码即可实现中文或者亚洲字符在 JSP 不同页面中传输。

商品浏览及选择界面的运行结果如图 5-10 所示。

图 5-10　商品浏览及选择界面

用户通过购物车进行商品管理页面的运行结果如图 5-11 所示。

图 5-11　购物车商品管理

选择删除某项商品后的界面如图 5-12 所示。

图 5-12　删除某项商品后的购物车

例 5-5 作为一个相对简单的购物车模型，很多功能均未实现，有兴趣的读者不妨按照自己对购物车的理解来修改上面的程序代码，实现更完善的功能。

5.3.3 销毁 session

Servlet 采用了轻量级线程机制，所谓轻量级即表示构造和析构所需的资源很少，可以快速地创建并使用，但并不是轻量级的对象就可以无限构造并不用析构，虽然 Java 虚拟机提供了垃圾回收机制，但作为一种好的习惯，在对象不使用时应该关闭或者销毁。

session 对象在创建之后会消耗内存资源，所以如果在 Web 应用中不打算使用 session，应该在 page 指令中关闭，如下：

```
<%page contentType="text/html" pageEncoding="UTF-8" session=”false”%>
```

多数情况下都要使用会话跟踪技术，所以在某些情况下需要对不再使用的 session 对象进行销毁。一般存在以下 3 种情况：

- 服务器进程被停止，由于服务器停止，临时存储的 session 对象会被销毁。
- 距离前一次收到客户端发送请求并由服务器检索会话 ID 的时间间隔超过了通过 setMaxInactiveInterval(int interval)参数 interval 设置的值。
- 在 Web 应用中调用 invalidate()方法，强制会话无效，即销毁会话。

Tomcat 服务器对 session 的会话时间有具体的限制，默认为 30 分钟，在 web.xml 文件中可以进行配置，以设置当前 Web 应用中的所有会话，除非这些会话中通过第二种方法来覆盖这个默认值，或者通过修改 web.xml 文件的内容来修改整个 Web 应用的 session 最大无响应时间。web.xml 文件中关于 session 超时关闭的节点代码段如下：

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

临时 Cookies 会在关闭浏览器时结束与 Web Server 的会话，由于 session 在浏览器支持 Cookie 的情况下会优先使用临时 Cookies 来实现会话跟踪，所以一般认为的关闭浏览器即结束当前会话即是这种情况。

5.3.4 session 的生命周期

通过 5.3.3 节中的介绍，session 在 Tomcat 服务器中默认的最大响应时间是 30 分钟，这表示 session 的存在像人的生命一样有时间限制，拥有特殊的生命周期。

session 由于在 JSP 容器中会自动创建，可以通过调用 session 对象的 isNew()方法来判断当前 session 是否为新建。所谓新建 session 就是 Web Server 第一次响应客户端，给浏览器返回

sessionid 的时刻；随着浏览器继续提交请求，session 便不再是新建的了。

session 创建后，可以把在整个会话过程中所需的对象保存在 session 缓存中，并对其进行检索，但是由于 Web Server 限定了 session 的时限，当然不同的服务器默认的时限不尽相同，并且时限可以在配置文件中特定的节点或标签处修改。除此之外，session 的开发人员可以作为所建 session 的上帝，通过 setMaxAge(interval)强制修改 session 的生存时间，或者通过 invalidate()直接销毁 session，令 session 死亡。

掌握 session 的生命周期，能准确地把握保存在 session 缓存中对象的访问时间，这对 session 乃至整个 Web 应用的操作是非常有益的。

例 5-6 一个跟踪 session 的生命周期的例子。

简要分析：

例 5-6 需要在表单中输入用户名和密码，并通过验证页检查，如果用户名是 afro，密码是 1a2b3C；或者用户名是 bee，密码是 A1b2c3，则验证通过，进入页面 A，页面 A 会提示当前 session 是否为新建，同时会给出 4 个链接，分别是返回登录页面、页面 A、页面 B 和关闭 session 页。通过链接进入页面 B，则会提示用户的密码；进入关闭 session 页，会选择是立即关闭还是等待 5 秒，当然最终的结果均是销毁 session，结束其生命周期。当 session 被销毁后，返回登录页面，会给出提示，要求重新输入用户名和密码。

下面给出实现代码。

index.jsp 登录页面代码：

```
<%@page contentType="text/html" pageEncoding="GB2312"%>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
    "http://www.w3.org/TR/html4/loose.dtd">
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=GB2312">
        <title>JSP Page</title>
    </head>
    <%
                String error = (String) request.getAttribute("error");
                if (error == null) {
                    error = "";
                }
    %>
    <body>
        <jsp:include page="inc_AllPages.jsp"/>
        <font color="red"> <%=error%> </font>
        <form action="lesson5d6_valid.jsp" method="post">
            <table>
```

```
                    <tr>
                        <td>用户名：</td>
                        <td><input type="text" name="username"></td>
                    </tr>
                    <tr>
                        <td>密码：</td>
                        <td><input type="password" name="password"></td>
                    </tr>
                    <tr>
                        <td colspan="2">
                            <input type="submit" value="SUBMIT">
                        </td>
                    </tr>
                </table>
            </form>
        </body>
</html>
```

lesson5d6_valid.jsp 验证页面代码：

```
<%@page contentType="text/html" pageEncoding="GB2312"%>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
    "http://www.w3.org/TR/html4/loose.dtd">
<jsp:useBean id="validbean" class="org.me.afro.ValidateUP"/>
<jsp:setProperty name="validbean" property="*"/>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=GB2312">
        <title>Valid Page</title>
    </head>
    <body>
        <%
                    //也可以通过 bean 进行取值
                    String username = request.getParameter("username");
                    String password = request.getParameter("password");
                    if (validbean.isBlValid()) {
                        //创建 session 并在其缓存中保存对象
                        session.setAttribute("username", username);
                        session.setAttribute("password", password);
                        response.sendRedirect("lesson5d6a.jsp");
                    } else {
                        request.setAttribute("error", "用户名或密码无效，请重新输入！");
```

```
                    //下面使用 servlet 中的请求转发对象实现 forward 跳转，和<jsp:forward>功能一致
                    request.getRequestDispatcher("index.jsp").forward(request, response);
                }
            %>
        </body>
    </html>
```

验证页面通过调用验证 bean——validateUP 来实现验证功能，被许可的用户名是 afro，密码是 1a2b3C；或者用户名是 bee，密码是 A1b2c3，其他输入均不被认可。

```
package org.me.afro;
public class ValidateUP {
    private String username;
    private String password;
    private boolean blValid;
    public ValidateUP(){
        blValid=false;
    }
    /**
     * @return the username
     */
    public String getUsername() {
        return username;
    }

    /**
     * @param username the username to set
     */
    public void setUsername(String username) {
        this.username = username;
    }

    /**
     * @return the password
     */
    public String getPassword() {
        return password;
    }

    /**
     * @param password the password to set
     */
```

```
    public void setPassword(String password) {
        this.password = password;
    }

    private boolean validate() {
        if (password != null && username != null) {
            if (username.equalsIgnoreCase("afro") && password.equals("1a2b3C")
                    || username.equalsIgnoreCase("bee") && password.equals("A1b2c3")) {
                this.blValid = true;
            } else {
                this.blValid = false;
            }

        }
        return blValid;
    }

    /**
     * @return the blValid
     */
    public boolean isBlValid() {
        return validate();
    }
}
```

lesson5d6a.jsp 页面 A 代码：

```
<%@page contentType="text/html" pageEncoding="GB2312"%>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
    "http://www.w3.org/TR/html4/loose.dtd">
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=GB2312">
        <title>A Page</title>
    </head>
    <%
                String aName = (String) session.getAttribute("username");
                if (aName == null) {
                    request.setAttribute("error", "请您先登录再访问！ ");
                    request.getRequestDispatcher("index.jsp").forward(request, response);
                }
    %>
```

```
    <body>
        <jsp:include page="inc_AllPages.jsp"/>
        <font style="color:red">Welcome <%=aName%> to A!</font>
        <%
                    if (session.isNew()) {
                        out.println("this session is new,sessinid=" + session.getId());
                    } else {
                        out.println("the session has existed...");
                    }
        %>
    </body>
</html>
```

lesson5d6b.jsp 页面 B 代码：

```
<%@page contentType="text/html" pageEncoding="GB2312"%>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
    "http://www.w3.org/TR/html4/loose.dtd">

<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=GB2312">
        <title>B Page</title>
    </head>
    <%
                String bName = (String) session.getAttribute("username");
                if (bName == null) {
                    request.setAttribute("error", "请您先登录再访问!");
                    request.getRequestDispatcher("index.jsp").forward(request,
                            response);
                }

    %>
    <body>
        <jsp:include page="inc_AllPages.jsp"/>
        <form>
            <font style="color:green">Welcome <%=bName%> to B!</font>
        Do u remember ur password?
            <input type="submit" value="Yes" name="viewPass" />
        </form>
        <%
                    if (request.getParameter("viewPass") != null&&request.getParameter
```

```
                    ("viewPass").equals("Yes")) {
            %>
            <i><%=session.getAttribute("password").toString()%></i>
            <%
                    }
            %>
        </body>
</html>
```

lesson5d6_closeSession 关闭 session 页面代码：

```
<%@page contentType="text/html" pageEncoding="GB2312"%>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
    "http://www.w3.org/TR/html4/loose.dtd">

<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=GB2312">
        <title>inc_AllPage</title>
    </head>
    <body>
        <h1>
            <table border="1">
                <tr>
                    <td><a href="index.jsp">返回登录页面</a></td>
                </tr>
                <tr>
                    <td><a href="lesson5d6a.jsp">A 页面</a></td>
                </tr>
                <tr>
                    <td><a href="lesson5d6b.jsp">B 页面</a></td>
                </tr>
                <tr>
                    <td><a href="lesson5d6_closeSession.jsp">Close_Session</a></td>
                </tr>
            </table>
        </h1>
    </body>
</html>
```

inc_AllPages.jsp 为以上所有页面提供 table 表格和链接支持。

```
<%@page contentType="text/html" pageEncoding="GB2312"%>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
    "http://www.w3.org/TR/html4/loose.dtd">
```

```
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=GB2312">
        <title>inc_AllPage</title>
    </head>
    <body>
        <h1>
            <table border="1">
                <tr>
                    <td><a href="index.jsp">返回登录页面</a></td>
                </tr>
                <tr>
                    <td><a href="lesson5d6a.jsp">A 页面</a></td>
                </tr>
                <tr>
                    <td><a href="lesson5d6b.jsp">B 页面</a></td>
                </tr>
                <tr>
                    <td><a href="lesson5d6_closeSession.jsp">Close_Session</a></td>
                </tr>
            </table>
        </h1>
    </body>
</html>
```

例 5-6index.jsp 登录页面的运行结果如图 5-13 所示，lesson5d6a.jsp 页面的运行结果如图 5-14 所示。

图 5-13　登录页面

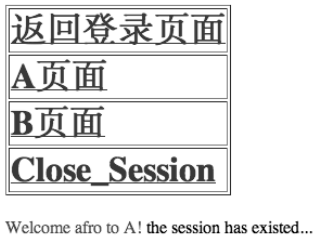

图 5-14　页面 A

lesson5d6b.jsp 页面的运行结果如图 5-15 所示，lesson5d6_closeSession.jsp 关闭会话页的运行结果如图 5-16 所示。

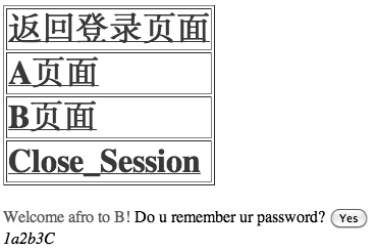

图 5-15　页面 B

图 5-16　关闭 session 页

关闭 session 之后，再跳转到页面 A 或 B，显示结果如图 5-17 所示。

图 5-17　session 被销毁，记录的对象一并解除

5.3.5　会话绑定监听器

由于 session 具有生存时限，保存在 session 缓存中的对象数据将是临时性的，在某些情况下，需要把临时存储在 session 中的对象在 session 结束时持久化地存储到文件系统或数据库中；另一种情况是，对象和 session 关系紧密，对象的每一次改变都需要通知 session 及时记录保存，这就需要在那些需要得到 session 即时状态的对象上绑定一个监听器，在以上两种情况发生时，产生一个事件 Event 来监视会话。

会话绑定监听器是 Servlet 提供的一个接口，接口名为 HttpSessionBindingListener，该接口包含以下两个抽象方法：

- public void valueBound(HttpSessionBindingEvent event);
- public void valueUnbound(HttpSessionBindingEvent event);

当对象需要存入 session 时，会自动调用 valueBound()方法，若 session 被销毁，会自动调用 valueUnbound()方法。

会话监听的主要优势在于不管客户端是主动关闭 Web 应用还是会话超时，会话监听器接口会迅速做出反应，同时也会回收 session 在存在时占用的系统资源。

5.4 实例实现

通过本章的学习，5.1 节引入的 CHERRYONE 公司需要改良的前代原型，读者是否能够根据 5.1 节给出的功能需求自行实现呢？下面来看一下 Zac 开发团队是如何实现该原型的。

提示：本章实例主要是在用户登录界面添加 Cookie 的接收方法，并按照例 5-3 的实现方式，用 EL 作为表单中控件 value 属性的值，同时提供用户需要保存 Cookie 的时间。在服务器端通过一个 Servlet 对用户的 Cookie 请求进行处理，然后通过 response.sendRedirect()方法跳转到相应的具有用户语言环境的页面中。

5.5 习题

1．解释 Java Web 的会话管理机制。
2．Cookies 技术和 JSP 中 session 的区别是什么？
3．JSP 的 4 种会话跟踪技术是哪些？
4．简述 JSP 中 session 的生命周期。
5．JSP 网页中为什么需要引入会话？