《JavaEE 实验手册》

Java 教研室

版 本 1.3

文档提供：Java 教研室 孙丽萍

修 改 记 录

修改时间	修改人	修改内容
2009.8.25	刘战洪	文档创建
2012.8.27	张立飞	文档修改
2016.3.7	孙丽萍	文档修改

目录

一、内容概述.....4
二、实验一 掌握 JavaWeb 应用中对 Cookie 的操作.....4
2.1 实验目的.....4
2.2 准备.....4
2.3 实验步骤.....4
2.4 实验结论.....8
2.5 扩展.....8
三、实验二 JavaWeb 应用中 HttpSession 对象的使用.....8
3.1 实验目的.....8
3.2 功能描述.....9
3.3 实验步骤.....9
3.4 实验结论.....11
3.5 扩展.....11
四、实验三 使用 Cookie 技术实现用户登录功能.....11
4.1 实验目的.....11
4.2 功能描述.....11
4.4 编码实现.....11
4.4 实验结论.....14
4.5 扩展.....14
五、实验四 使用 HttpSession 实现购物车功能.....14
5.1 实验目的.....14
5.2 功能描述.....14
5.3 概要设计.....15
5.4 编码实现.....15
5.5 实验结论.....18
5.6 扩展.....18

第四章 会话(Session & Cookie)

一、内容概述

本章的教学内容是理解会话、会话跟踪技术概念的产生，以及会话跟踪技术的产生背景和实现思路。熟悉在 JavaWeb 应用程序中实现会话跟踪的技术解决方案都有哪些，并且能够熟练的将会话跟踪技术应用在实际项目中。

二、实验一 掌握 JavaWeb 应用中对 Cookie 的操作

2.1 实验目的

掌握 JavaWeb 应用中对 Cookie 的操作。

2.2 准备

在进行开发之前，要保证开发工具的正确安装与配置。

2.3 实验步骤

步骤一：

创建“Dynamic Web Project”，项目名称为 02_session。

步骤二：

由 Eclipse 创建 Servlet 及其配置，Servlet 类的名称为 CookieDemo，该类的主要功能是判断客户端有没有使用 Cookie 保存用户名，如果有就显示欢迎信息，如果没有就提示用户输入用户名并将用户名以 Cookie 的形式保存到访问者的客户端电脑中(保存时间为一周)。关键部分的代码如下所示

```
protected void doPost(HttpServletRequest request
        , HttpServletResponse response)
        throws ServletException, IOException {
    PrintWriter writer = response.getWriter();
    Cookie[] cookies = request.getCookies();

    String visitorName = null;
    if (cookies != null) {
        for (Cookie cookie : cookies) {
            if ("vName".equals(cookie.getName())) {
                visitorName = cookie.getValue();
                break;
            }
        }
        if (visitorName != null) {
            writer.write("hi <b>" + visitorName + "</b>, welcome come
back!");
        } else {
            writer.write("<form method='post'
action='CookieDemoAdd'>please input your name:<input type='text'
name='name' /><input type='submit' /></form>");
        }
    } else {
        writer.write("<form method='post'
action='CookieDemoAdd'>please input your name:<input type='text'
name='name' /><input type='submit' /></form>");
    }
}
```

在上面的代码中，需要注意的几点：

- 通过 request.getCookies 方法得到的是 Cookie 数组，有可能为 null
- 通过 getName 方法找到 Cookie 对象，再用 getValue 方法获得其值
- 进行 Cookie 验证时，如果没有 Cookie，向客户端输出提供输入用户名的<HTML>文档，其中表单(form)的 action 属性为 CookieDemoAdd。

步骤三：

在上一步骤中，如果没有检测到客户端的 Cookie 值，就提示用户输入名字并提交给 CookieDemoAdd(也是一个 Servlet 类)，该 Servlet 的主要功能是取得用户输入的名字并将名字保存到客户端 Cookie 中，保存结束后页面跳转到 CookieDemo 页面(Servlet)，关键代码如下：

```
protected void doPost(HttpServletRequest request
        , HttpServletResponse response)
        throws ServletException, IOException {
    String vname = request.getParameter("name");
    Cookie cookie = new Cookie("vName", vname);
    cookie.setMaxAge(60*60*24*7); //在客户端保存时间
    response.addCookie(cookie);
    response.sendRedirect("CookieDemo");
}
```

上面的代码中，有几点需要注意：

- 使用 Cookie 的构造器创建 Cookie 对象并传入 Key-Value
- 使用 setMaxAge 方法设置保存时间，单位为秒
- 需要调用 addCookie 方法才能将 Cookie 输出到客户端
- Cookie 发送结束后，将页面跳转到 CookieDemo

步骤四：

运行该项目，在浏览器中输入访问地址：http://localhost:8080/02_session /CookieDemo ，页面输出如*图 2-1*

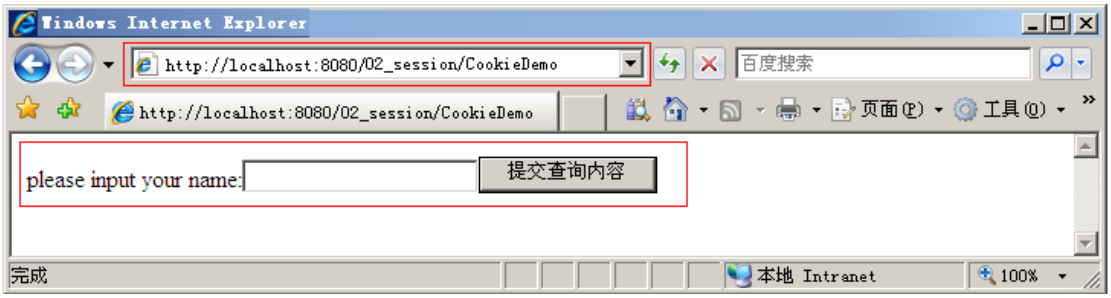

图 2-1

因为第一次访问，客户端没有 Cookie 的值，所以显示的页面信息是要求用户输入用户名。

步骤五：

这时观察 C:\Documents and Settings\Administrator\Cookies 目录下的最新文件(如果找不到该目录，将“隐藏受保护的操作系统文件”选项去掉，操作步骤是在资源管理器的“工具”->“文件夹选项...”->“查看”这个选项卡页中)，如*图 2-2*

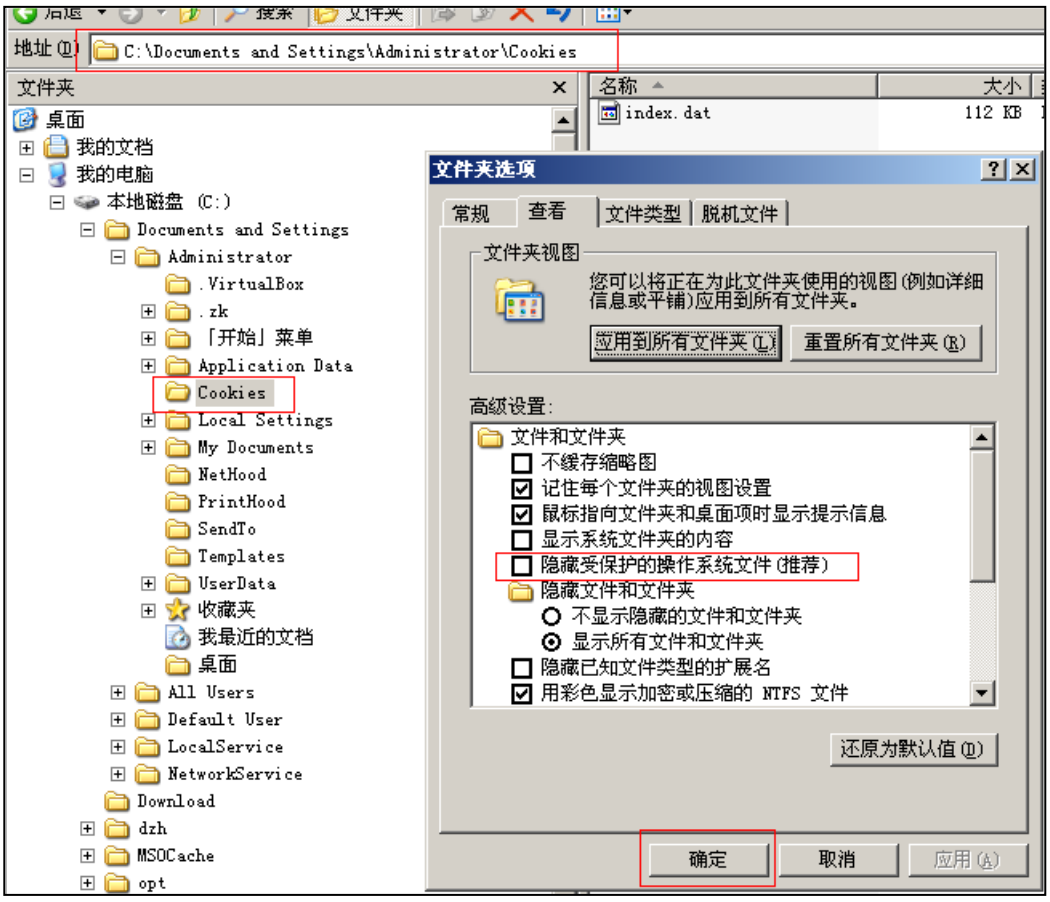

图 2-2

为了方便观察变化，可以将该目录下的所有文本文件删除掉。

步骤六：

转到浏览器中，输入你的名称，并点击提交按钮，如*图 2-3*

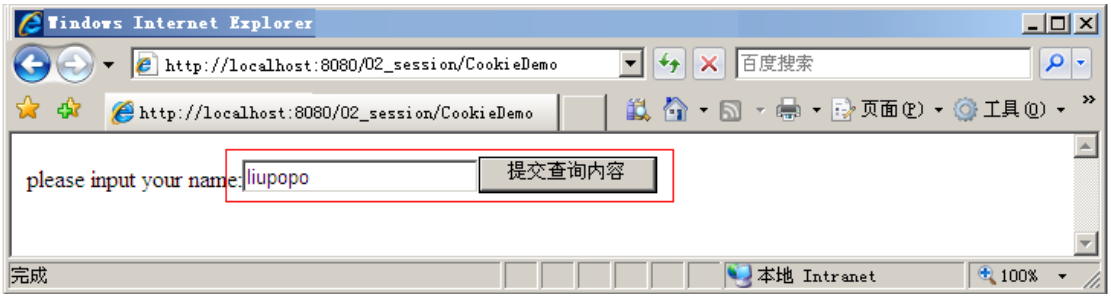

图 2-3

步骤七：

页面(CookieDemo)被提交到 CookieDemoAdd 页面(Servlet)，在 CookieDemoAdd 中，将用户名以 Cookie 的形式保存到客户端，再将页面跳转回 CookieDemo 页面。如*图 2-4*

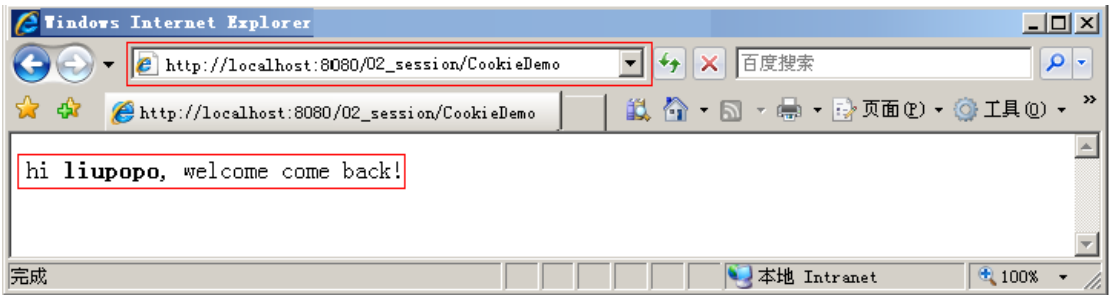

图 2-4

这时，客户端已经有 Cookie 的设置了，所以该页面显示欢迎信息，将用户名(该用户名称保存于 Cookie 中)输出到页面。

步骤八：

再次观察 C:\Documents and Settings\Administrator\Cookies 目录，会新增一个文本文件(administrator@localhost[1].txt，文件名可能不同，但形式同样是@localhost 这样的，表示是 localhost 站点发来的 Cookie 信息)，用记事本打开该文本文件，可以找到你输入的用户名。*图 2-5*

图 2-5

2.4 实验结论

通过试验得出结论：Servlet 程序可以使用 Cookie 类将一些信息保存在客户端，然后通过 request.getCookies 方法取出客户端保存的这些 Cookie 信息。

2.5 扩展

尝试使用 Cookie 技术对客户端的会话进行跟踪，也就是说，使用 Cookie 来识别与保存不同客户端以及该客户端的各种相关数据。

三、实验二 JavaWeb 应用中 HttpSession 对象的使用

3.1 实验目的

掌握 JavaWeb 应用中 HttpSession 对象的使用。

3.2 功能描述

访问页面(Servlet)时，读取 Session 对象“userName”属性的值，如果有值，将值信息输出到页面。如果没有值(为空 null)，提示用户输入名称并提交，提交后将用户输入的名称保存到 Session 对象中。

3.3 实验步骤

步骤一：

该功能在实验一的基础上完成，新建 Servlet 类：SessionDemo，该类的关键代码如下：

```
protected void doPost(HttpServletRequest request
        , HttpServletResponse response)
       throws ServletException, IOException {
    PrintWriter writer = response.getWriter();
    HttpSession session = request.getSession();
    Object attribute = session.getAttribute("userName");
    if (attribute == null) {
        writer.write("<form method='post' action=''>please input your
name:<input type='text' name='vName' /><input type='submit'
/></form>");
        String name = request.getParameter("vName");
        session.setAttribute("userName", name);
    } else {
        writer.write("hi <b>" + attribute.toString() + "</b>, welcome
come back!");
    }
}
```

在 doPost 方法中，使用 request.getSession 方法取得该客户端相关的会话对象(Session)，并判断 Session 对象“userName”属性的值(取得属性值使用方法 session.getAttribute)，如果值为 null，提示用户输入名称，并判断用户是否输入过用户名，如果能得到用户输入的值(request.getParameter 方法)，将该值保存到 Session 对象中(session.setAttribute 方法)。

注意：提示用户输入信息的表单的 action 为空，表示提交到自身 URL。

步骤二：

运行项目，在 URL 中输入 http://localhost:8080/02_session/SessionDemo，因为是第一次访问，没有 Session 的“userName”属性，所以会提示用户输入名称，如*图 3-1* 所示

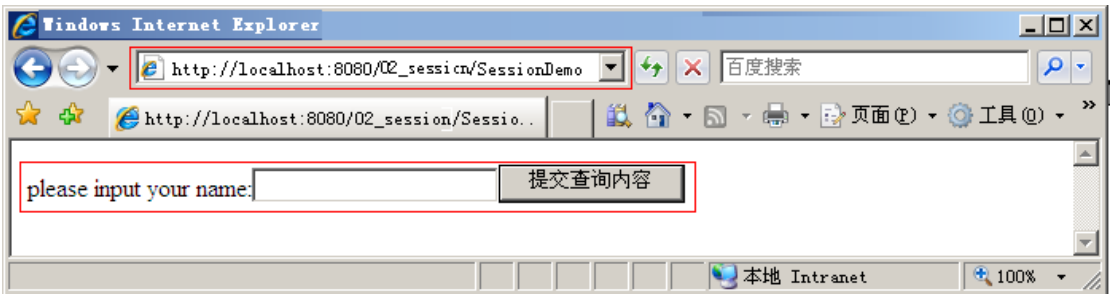

图 3-1

注意：这是第一次访问该页面，用户还没有提交过数据，所以此时的代码中使用的 request.getParameter 方法没有得到任何数据(为 null)，因而此时向 Session 对象中的“username”属性设置的值也为 null。

步骤三：

输入名称，如*图 3-2*

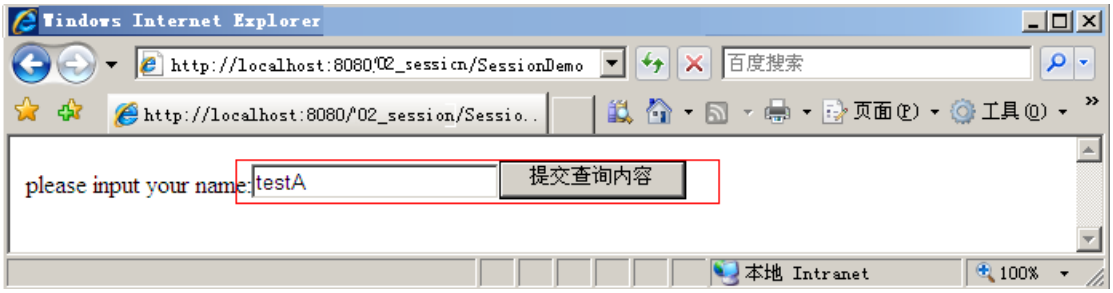

图 3-2

步骤四：

点击“提交查询内容”按钮，页面会跳转至如*图 3-3* 页面

图 3-3

注意：此时并没有显示用户名，而是又回到要求输入用户名的界面，不要怀疑，因为这是第一次提交，Session 属性中的值为 null，当该 Servlet 执行完的时候，SessionDemo 得到用户输入的名并设置成为 Session 对象的一个属性。

步骤五：

再输入另一个名称，如 testB，点击提交按钮，页面显示如*图 3-4*

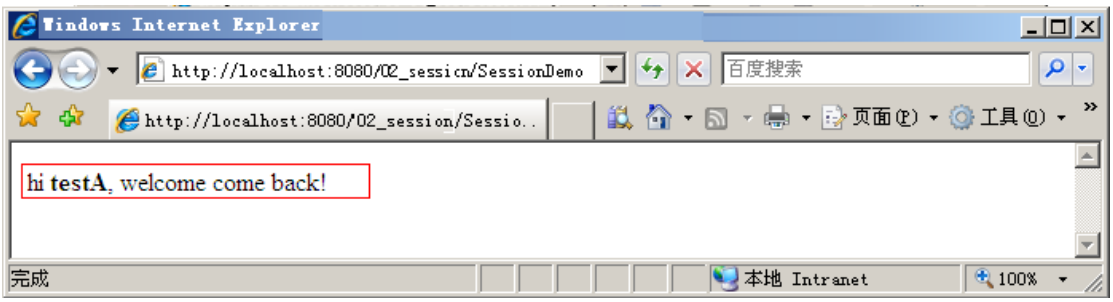

图 3-4

注意：输出的用户名是第一次输入的值(testA)，想一想为什么？

3.4 实验结论

可以使用 HttpSession 对象保存客户端的信息，相比起 Cookie 技术，HttpSession 的使用更方便。

3.5 扩展

怎样修改代码，当用户提交后显示的是当时输入的名称(Session 中保存的已经是最新的用户输入值)。

举例说明一下 Session 可以应用在什么场合，Session 与 Cookie 的区别，分别在什么应用场景下使用这两种技术。

四、实验三 使用 Cookie 技术实现用户登录功能

4.1 实验目的

使用 Cookie 技术实现用户登录功能。

4.2 功能描述

用户进行登录操作，如果在 Cookie 中保存有用户名信息，在页面上直接将用户名显示出来，否则转到用户输入的页面，要求用户输入用户名和密码，并且可以选择是否保存用户信息到客户端。用户输入用户名后，可成功登录。

4.3 概要设计

首先需要一个 Servlet 类(LoginServlet)作为登录功能模块的入口，当 Cookie 中保存有用户名信息输出用户名，否则将页面跳转到用户输入页面(login.html)。用户输入信息并提交到 LoginCheckServlet 类,在这个 Servlet 里根据用户输入的信息进行相应的对 Cookie 信息的处理。

4.4 编码实现

步骤一：

LoginServlet 类的关键代码如下：

```
protected void doPost(HttpServletRequest request
        , HttpServletResponse response)
        throws ServletException, IOException {
    Cookie[] cookies = request.getCookies();
    if (cookies == null) {
        response.sendRedirect("login.html");
    } else {
        String cookieValue = null;
        for (Cookie cookie : cookies) {
            if ("userName".equals(cookie.getName())) {
                cookieValue = cookie.getValue();
                break;
            }
        }
        if (cookieValue == null)
            response.sendRedirect("login.html");
        else
            response.getWriter().write("hello:" +
cookieValue.toString());
    }
}
```

该类主要用来判断 Cookie 中是否有用户信息，并分别处理。

步骤二：

login.html 的内容如下

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Insert title here</title>
</head>
<body>
<form action="loginCheckServlet">
please input you name:<input type="text" name="username" /><br />
input you password:<input type="text" name="password" /><br />
<input type="checkbox" name="isSave" />保存一周<br />
<input type="submit" />
</form>
</body>
</html>
```

注意：用户信息会提交到 loginCheckServlet，使用 checkbox 来让用户决定是否保存 Cookie 信息到客户端。

步骤三：

loginCheckServlet 的关键代码如下：

```
protected void doPost(HttpServletRequest request
        , HttpServletResponse response)
        throws ServletException, IOException {
    String parameter = request.getParameter("isSave");
    String username = request.getParameter("username");
    Cookie cookie = new Cookie("userName", username);
    response.addCookie(cookie);
    if ("on".equals(parameter))
        cookie.setMaxAge(60 * 60 * 24 * 7);
    response.sendRedirect("LoginServlet");
}
```

该 Servlet 接收用户输入的信息，注意客户输入的 checkbox 值，根据用户输入将信息保存到 Cookie 中，并再次转向 LoginServlet。

步骤四：

运行项目，验证执行结果，思考该实验的整个执行流程。整个执行过程中的页面应该如*图 4-1*、*图 4-2*、*图 4-3* 所示

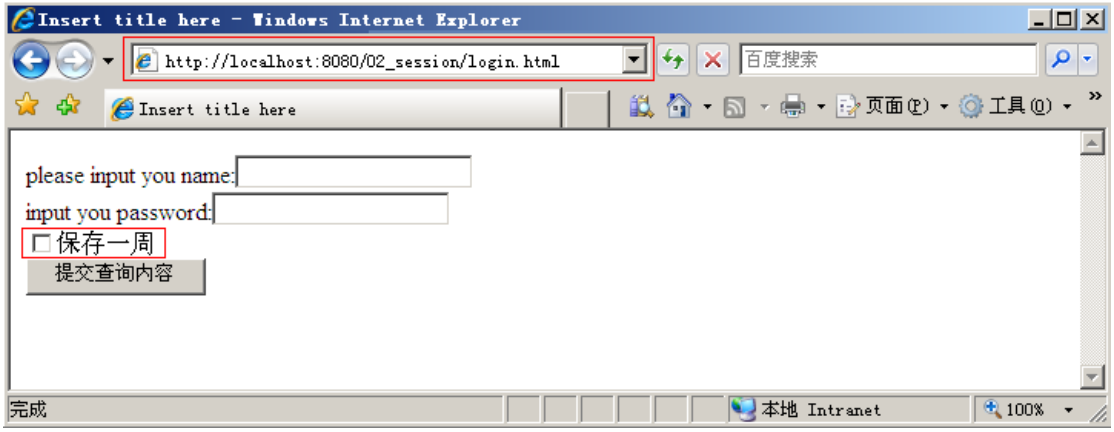

图 4-1

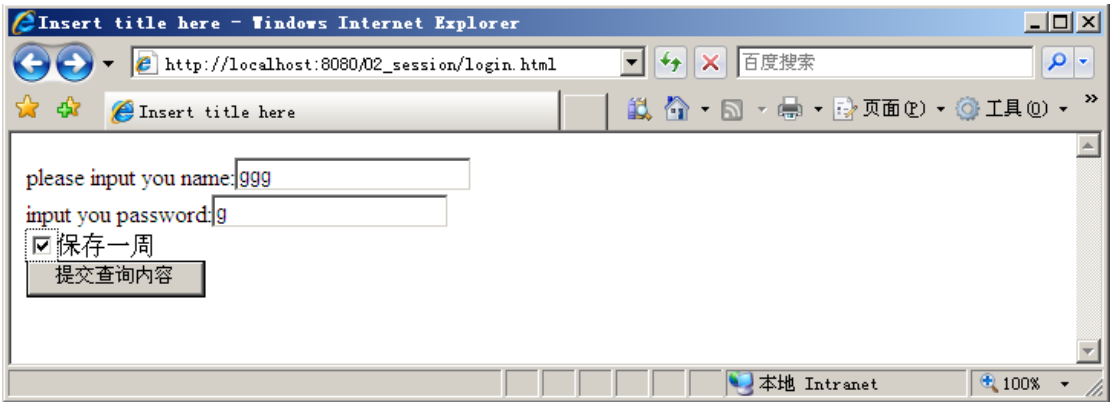

图 4-2

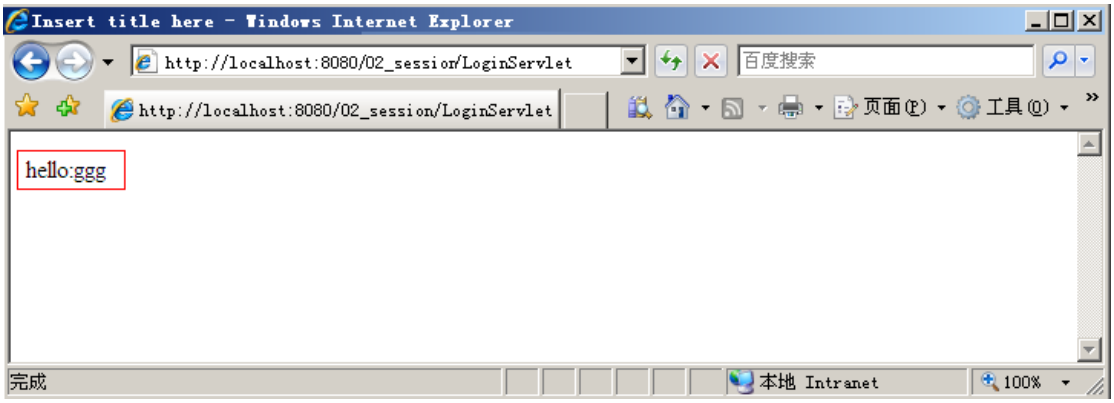

图 4-3

4.4 实验结论

可以使用 Cookie 技术保存用户登录信息到客户端。

4.5 扩展

测试用户不选择“保存一周”的情况，启动多个 IE 浏览器查看运行结果，应该得出结论：如果不使用 cookie.setMaxAge 方法设置保存时间，Cookie 的存活期在同一个浏览窗口中，关闭该窗口 Cookie 就失效；如果使用了 cookie.setMaxAge 方法，会将 Cookie 信息保存到客户端。

从安全角度考虑，密码输入框应该改为 password，并且提交方式应该选择 POST 方式提交(解释为什么？)。

用户输入数据的有效性应该进行客户端与服务器端的验证，尝试使用前面学过的 JavaScript 编写客户端验证，要求用户名和密码不可以为空。

五、实验四 使用 HttpSession 实现购物车功能

5.1 实验目的

使用 HttpSession 实现购物车功能。

5.2 功能描述

向用户展示一个图书信息的列表，用户可以将某本图书保存到自己的购物车中并继续购物，用户可以随时查看购物车中已经购买的置身图书信息。

5.3 概要设计

用户可以访问 productList.html 页面，该页面向用户展示所有图书信息的列表。在该页面中同时可以由用户选中某本图书放入购物车（由 AddShoppingCartServlet 实现），并可以通过连接显示购物车中的信息（由 ShowShoppingCartServlet 实现）。

5.4 编码实现

步骤一：

productList.html 页面代码如下

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Insert title here</title>
</head>
<body>
<table border="1">
    <tr>
        <th>No.</th>
        <th>Name</th>
        <th>Price</th>
        <th>...</th>
    </tr>
    <tr>
        <td>1</td>
        <td>C</td>
        <td>60</td>
        <td><a href="AddShoppingCartServlet?id=1&name=C">Add to
ShoppingCart</a></td>
    </tr> //省略两行记录...
    <tr>
        <td>4</td>
        <td>VB</td>
        <td>20</td>
        <td><a href="AddShoppingCartServlet?id=4&name=VB">Add to
ShoppingCart</a></td>
    </tr>
</table>
<a href="ShowShoppingCartServlet">show ShoppingCart</a>
</body>
</html>
```

注意向购物车添加图书是通过一个链接实现，该链接会调用AddShoppingCartServlet，并且同时将图书的ID和名称传递到Servlet中进行处理。显示购物车信息也是通过链接实现，直接转向ShowShoppingCartServlet。

浏览productList.html页面应该如*图5-1*

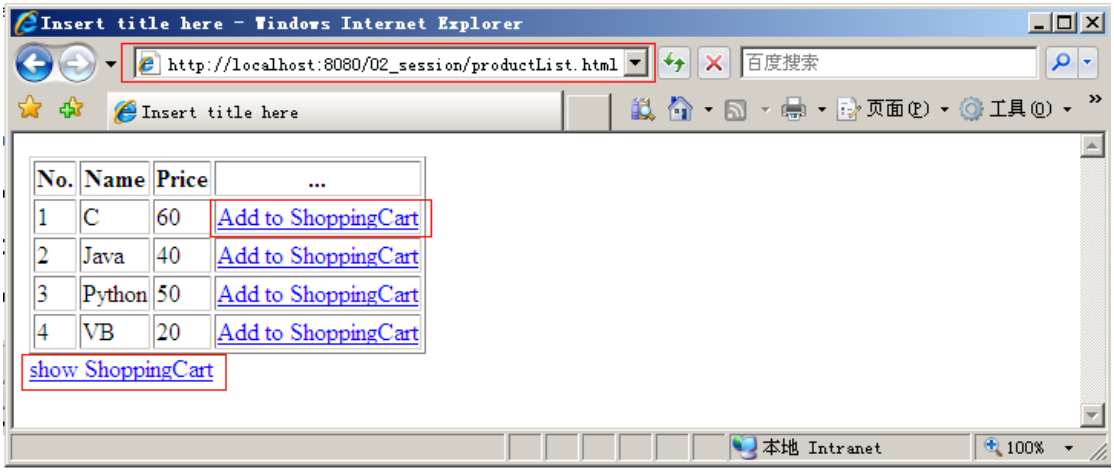

图5-1

步骤二：

AddShoppingCartServlet的关键代码如下：

```
protected void doPost(HttpServletRequest request
        , HttpServletResponse response)
        throws ServletException, IOException {
    String bookName = request.getParameter("name");
    List<String> booklist = null;
    HttpSession session = request.getSession();
    Object shopingcart = session.getAttribute("shoppingcart");
    if (shopingcart == null) {
        booklist = new ArrayList<String>();
        booklist.add(bookName);
        session.setAttribute("shoppingcart", booklist);
    } else {
        booklist = (List<String>) shopingcart;
        booklist.add(bookName);
        session.setAttribute("shoppingcart", booklist);
    }
```

当添加一个图书信息到购物车时，实际是将该图书信息保存到会话(Session)的“shoppingcart”属性中，“shoppingcart”属性以列表的形式保存了所有该会话购买的图书信息(booklist)。

步骤三：

ShowShoppingCartServlet的关键代码如下：

```
protected void doPost(HttpServletRequest request
        , HttpServletResponse response)
        throws ServletException, IOException {
    PrintWriter writer = response.getWriter();
    HttpSession session = request.getSession();
    Object shoppingcart = session.getAttribute("shoppingcart");
    if (shoppingcart == null) {
        writer.write("No Selected......");
    } else {
        writer.write("your Selected:<br />");
        List<String> booklist = (List<String>) shoppingcart;
        for (int i = 0; i < booklist.size(); i++) {
            writer.write(i + ": " + booklist.get(i).toString() + "<br
/>");
        }
    }
}
```

显示购物车信息的实质就是将会话中“shoppingcart”属性所保存的图书列表(booklist)显示到页面上。

步骤四：

运行该项目，选择相应操作进行测试，如*图 5-2*、*图 5-3*、*图 5-4* 所示

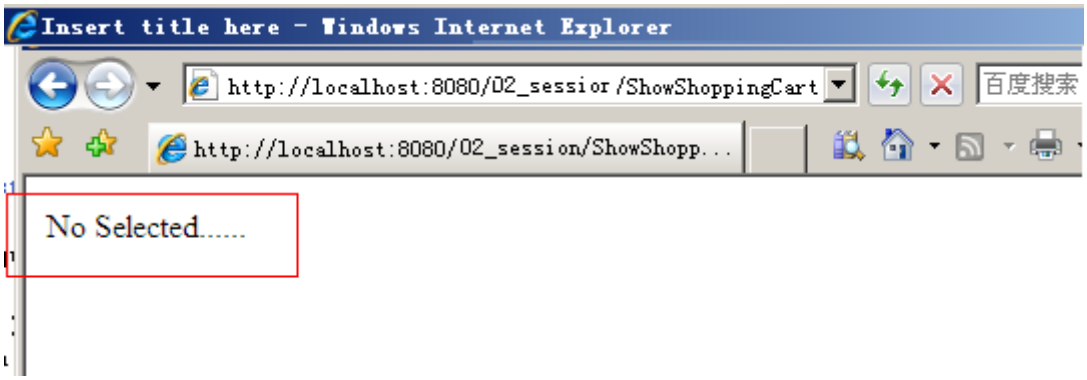

图 5-2

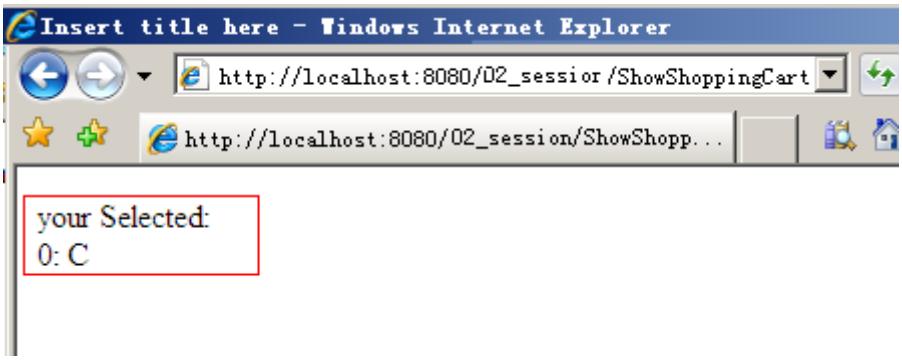

图 5-3

图 5-4

5.5 实验结论

掌握使用 HttpSession 保存客户端的信息。

5.6 扩展

在实际项目中，展示给用户的数据以及购物车中保存的数据信息要复杂的多，大多是从数据库中获取的。该购物车的实现实在太简陋，根本不能用于实际的项目中，但通过该实验同学们应该掌握 HttpSession 的应用以及实现购物车功能的思路。