第5章 JavaBean技术

本章介绍JSP程序开发中的JavaBean技术，JavaBean是一种可重用的组件技术，它可以将内部动作封装起来，用户不需要了解其如何运行，只需要知道如何调用及处理对应的结果即可。在动态网站开发应用中，使用JavaBean可以简化JSP页面的设计与开发，提高代码可读性，从而提高网站应用的可靠性和可维护性。

本章主要包括JavaBean的相关概念、JavaBean的属性、JavaBean的创建以及JavaBean的应用。通过本章的学习，读者应该了解JavaBean的基本概念，掌握JavaBean中各种属性的应用，掌握创建JavaBean的方法以及如何在JSP页面中应用JavaBean，从而能够应用JavaBean开发程序。

5.1 JavaBean的基本概念

JavaBean是用Java语言描述的、易用的、与平台无关的软件组件模型，用于设计可重用的组件，类似于Microsoft的COM组件概念。在Java模型中，通过JavaBean可以无限扩充Java程序的功能，通过JavaBean的组合可以快速地生成新的应用程序。对于程序员来说，最好的一点就是JavaBean可以实现代码的重用，另外对于程序的易维护性等也有重大意义。

JavaBean是使用一种符合某些命名方法和设计规范的Java类。创建JavaBean并不是一件困难的事情，要注意的一点就是在非可视化JavaBean中，常用getXXX()和setXXX()这样的成员方法来处理JavaBean的属性。

JavaBean具有以下特性。

(1) 可以实现代码的重复利用。

(2) 易维护性、易使用性、易编写性。

(3) 可以在支持Java的任何平台上工作，而不需要重新编译。

(4) 可以在内部、网内或者是网络之间进行传输。

(5) 可以以其他部件的模式进行工作。

下面是关于用户登录窗口的JavaBean，这是一个典型的JavaBean。

LoginUser.java程序代码如下：

```
public class LoginUser{
    public LoginUser(){}
```

```
    private String user = "";
    private String password = "";
    public String getPassword(){
    return password;
  }
  public void setPassword(String password){
    this.password = password;
  }
  public String getUser(){
    return user;
  }
  public void setUser(String user){
    this.user = user;
  }
}
```

5.1.1 JavaBean 的属性

JavaBean 的属性与一般 Java 程序中所指的属性,或者说与所有面向对象的程序设计语言中对象的属性是一个概念,在程序中的具体体现就是类中的变量。

属性是 Bean 组件内部状态的抽象表示。JavaBean 的属性可以分为以下 4 类。

(1) 简单属性(Simple)。

(2) 索引属性(Indexed)。

(3) 绑定属性(Bound)。

(4) 约束属性(Constrained)。

简单属性依赖于标准命名约定来定义 getXXX()方法和 setXXX()方法。索引属性则允许读取和设置整个数组,也允许使用数组索引单独地读取和设置数组元素。绑定属性则是其值发生变化时要广播给属性变化监听器的属性。约束属性则是那些值发生改变及起作用之前,必须由约束属性变化给监听器生效的属性。

绑定属性和约束属性通常在 JavaBean 的图形编程中使用,所以在这里不进行介绍,下面主要介绍 JavaBean 中的简单属性和索引属性。

1. 简单属性

简单属性就是在 JavaBean 中对应了简单的 setXXX()和 getXXX()方法的变量。在创建 JavaBean 时,简单属性最为常用。

在 JavaBean 中,简单属性的 getXXX()与 setXXX()方法如下:

```
public void setXXX(type value);
public type getXXX();
```

其中,type 表示属性的数据类型,若属性为布尔类型,则可使用 isXXX()方法代替 getXXX()方法。

例 5-1 简单属性

定义 formInput.java 文件,其代码如下:

```
public class formInput{
    //类型为 String,属性名为 sr
    String str = new String("NewYork  Trade  Center");
    public formInput (){}
    //set 属性
      public void setInput(String str){
          this.str = str;
    }
    //get 属性
      public String getInput(){
          return str ;
    }
}
```

2. 索引属性

需要通过索引访问的属性通常称为索引属性。如存在一个大小为 3 的字符串数组,若要获取该字符串数组中指定位置中的元素,需要得知该元素的索引,则该字符串数组就被称为索引属性。一个索引属性表示一个数组值。同上所述的简单属性一样,可以使用 getXXX()与 setXXX()方法取得数组中的值。

在 JavaBean 中,索引属性的 getXXX()与 setXXX()方法如下:

```
public void setXXX(type[ ] value);
public type[ ] getXXX();
public void setXXX(int index,type value);
public type getXXX(int index);
```

其中,type 表示属性类型,第一个 setXXX()方法为简单的 setXXX()方法,用来为类型为数组的属性赋值,第二个 setXXX()方法增加了一个表示索引的参数,用来为数组中索引为 index 的元素赋值为 value 指定的值;第一个 getXXX()方法为简单 getXXX()方法,用来返回一个数组,第二个 getXXX()方法增加了一个表示索引的参数,用来返回数组中索引为 index 的元素值。

例 5-2 索引属性

定义 formInputl.java 文件,其代码如下:

```
public class formInputl{
    //b 是一个索引属性
    String b[ ] = new String[ ]{"5","2","3","4"};
    public formInputl (){ }
```

```
    //取得整个数组的值
    public String[ ] getInput(){
      return b;
    }
    //设置整个数组
    public void setInput(String [ ]b){
      this.b = b;
    }
}
```

5.1.2 JavaBean 的方法

方法是处理事件的手段,而事件处理则是 JavaBean 体系结构的核心之一。

JavaBean 容器环境可以接收 JavaBean 组件的事件通知,并且,如果 JavaBeans 组件符合一些简单规则,就可以在设计时选择 JavaBean 组件可以响应的事件。在 JavaBean 组件上加入和删除方法必须以标准方式定义,以便分别加入和删除事件监听器。JavaBean 容器能够加入或删除对事件监听器的引用,它使用允许容器与组件事件交互的 JavaBean 组件。

JavaBean 组件上的事件可以用 Bean 进行注册——如果它实现了一个 addXXXListener(XXXListener)形式的方法,其中 XXX 是事件类型的名字。同样,Bean 如果实现了一个 removeXXXListener(XXXListene)形式的方法,事件就可以被注销。

最后说明一点,如果 JavaBean 组件在一个时刻只允许一个监听器,addXXXListener(XXXListener)方法应声明其产生 java.util.TooManyListenersException。

实训 16 创建简单属性的 JavaBean

【实训目的】

(1) 熟练掌握 JavaBean 的用法。

(2) 学会创建简单属性的 JavaBean。

【实训要求】

定义一个具有简单属性的 JavaBean,并定义相应的 setXXX()和 getXXX()方法进行属性的访问。

【实训步骤】

具体代码如下:

```
public class WordSingle {
    private String author;                    //存储留言者
    private String title;                     //存储留言标题
    private String content;                   //存储留言内容
    public String getAuthor() {
        return author;
```

```
    }
    public void setAuthor(String author) {
        this.author = author;
    }
    public String getContent() {
        return content;
    }
    public void setContent(String content) {
        this.content = content;
    }
    public String getTitle() {
        return title;
    }
    public void setTitle(String title) {
        this.title = title;
    }
}
```

5.2 在 JSP 中使用 JavaBean

5.2.1 创建 JavaBean

JavaBean 是 Java 程序的一种，所使用的语法与其他类似的 Java 程序一致。JavaBean 代码可以被其他程序引用。当一个项目很大的时候，可以建立没有用户界面的程序时，如计算、数据库引用等，就可以建立 JavaBean 了。

下面介绍如何在 MyEclipse 中创建一个 JavaBean。

例 5-3 在 MyEclipse 下创建 JavaBean

(1) 新建一个名为 SimpleBean 的 Web 项目。

(2) 右击项目中的 src 目录，并依次选择 New→Class 选项，在 Package 文本框中输入 com.ycl.bean，在 Name 文本框中输入要创建的 JavaBean 名，如 Hello，其他保持默认值，如图 5-1 所示。

(3) 单击 Finish 按钮，完成 JavaBean 的初步创建。

(4) MyEclipse 会自动以默认的与 Java 文件关联的编辑器打开创建的 Hello.java 文件，如图 5-2 所示。

(5) 在源代码中定义变量 hello，代码为：

```
String hello = "";
```

(6) 在图 5-2 所示的代码位置后面右击，并依次选择快捷菜单中的 Source→Goenerate Getters and Setters 选项。

(7) 在弹出的 Generate Getters and Setters 对话框中，单击 Select All 按钮，并保留其他选项的默认值，如图 5-3 所示。

图 5-1　创建 JavaBean

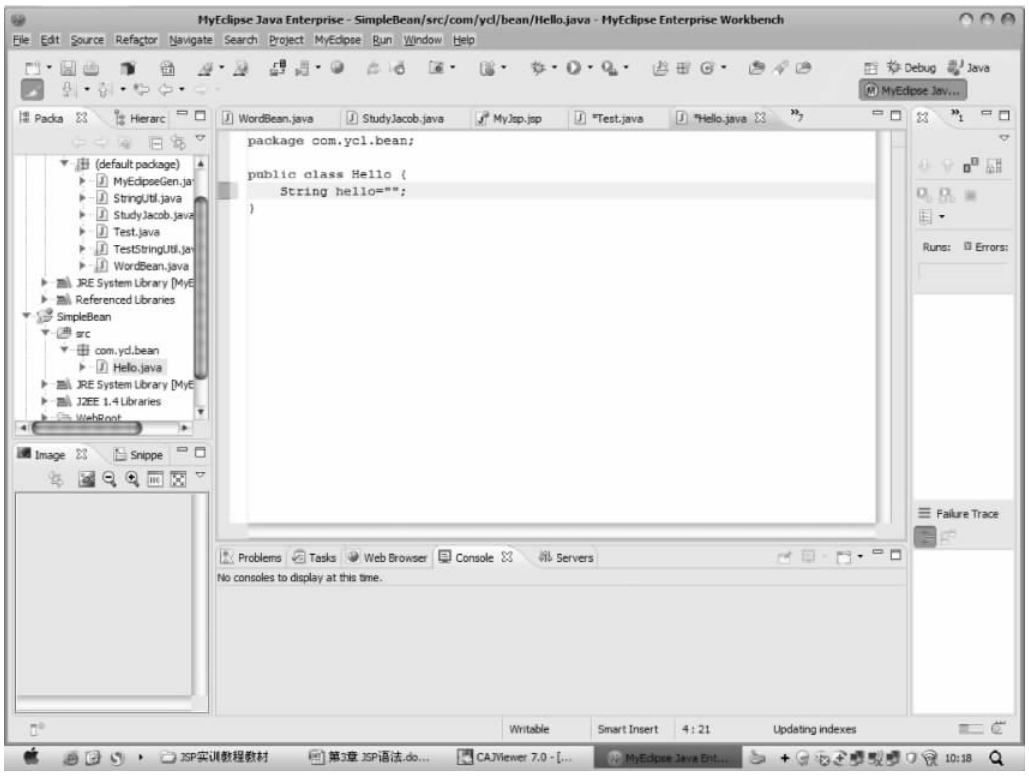

图 5-2　JavaBean 开发界面

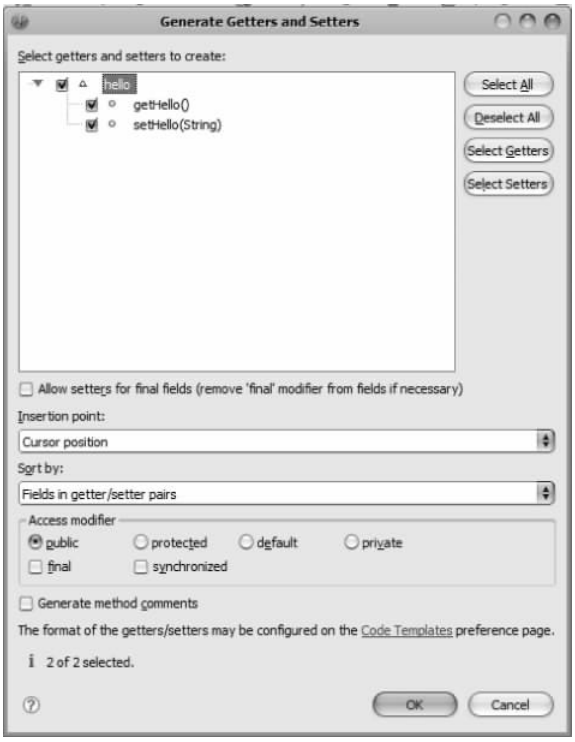

图 5-3 Generate Getters and Setters 对话框

(8) 设置完成后,生成代码如下:

```
package com.ycl.bean;
public class Hello {
    String hello = "";
    public String getHello() {
        return hello;
    }
    public void setHello(String hello) {
        this.hello = hello;
    }
}
```

5.2.2 在 JSP 页面中应用 JavaBean

JavaBean 能被应用在众多场合,特别是在 Servlet 和 JSP 这样的 Web 服务器端程序上。在 JSP 上使用 JavaBeans 不仅可以实现前台程序和业务逻辑的分离,还可以提高 JSP 程序的运行效率和代码重用的程度,并且可以实现并行开发。在 JSP 中可以通过动作标识<jsp:usebean>、<jsp:setproperty>、<jsp:getproperty>来应用 JavaBean。

无论哪一种 JavaBean,当它们被编译成 class 文件后,都需要放在项目中的 WEB-INF\classes 目录下,才可以在 JSP 页面中被调用。

JavaBean 作为信息的容器,通常用来封装表单数据,也就是将用户向表单字段中输入的数据存储到 JavaBean 对应的属性中。调用值 JavaBean 可以减少在 JSP 页面中嵌入大量的 Java 代码。

例 5-4　在 JSP 页面中调用值 JavaBean

例如,存在一个登录页面,如图 5-4 所示。当用户输入用户名和密码进行登录后,要求在另一个页面中输出用户输入的用户名和密码,如图 5-5 所示。

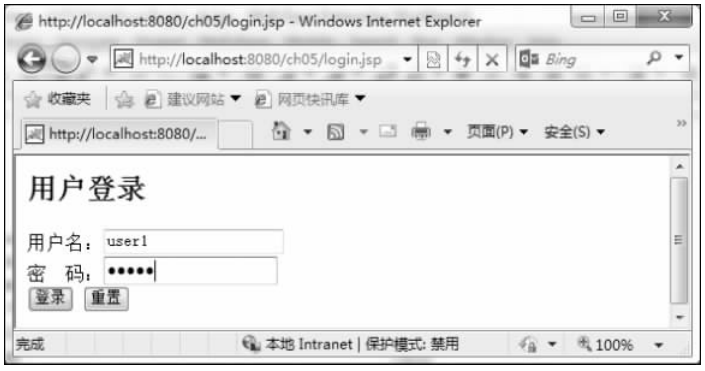

图 5-4　登录页面

图 5-5　JavaBean 运行结果

(1) 先创建登录页面 login.jsp,代码如下:

```
<%@ page contentType="text/html;charset=gb2312" %>
<form action="login_ok.jsp" >
    <h2>用户登录</h2>
        用户名:<input type="text" name="userName">
        <br>
        密   码:<input type="password" name="userPass">
        <br>
        <input type="submit" value="登录">
        <input type="reset" value="重置">
</form>
```

(2) 创建名为 Users 的值 JavaBean,该 Bean 中的属性要与登录页面 login.jsp 中表单的字段一一对应。Users 类的代码如下:

```
package com.ycl.bean;
public class Users {
    private String userName;                          //对应表单中的 userName 字段
    private String userPass;                          //对应表单中的 userPass 字段
    public String getUserName() {
        return userName;
    }
    public void setUserName(String userName) {
        this.userName = userName;
    }
    public String getUserPass() {
        return userPass;
    }
    public void setUserPass(String userPass) {
        this.userPass = userPass;
    }
}
```

(3) 创建表单处理页面 login_ok.jsp,在该页面中通过调用值 JavaBean 来获取表单数据。login_ok.jsp 页面的代码如下:

```
<%@ page contentType="text/html;charset=gb2312" %>
<jsp:useBean id="user" class="com.ycl.bean.Users">
    <jsp:setProperty name="user" property="*"/>
</jsp:useBean>
<center>
    <b>用户名:</b><jsp:getProperty name="user" property="userName"/>
    <b>密码:</b><jsp:getProperty name="user" property="userPass"/>
</center>
```

(4) 访问 login.jsp 页面,输入用户和密码后,单击“登录”按钮,将出现如图 5-5 所示的运行结果。

读者可以在代码中看到 JavaBean 中应用到了<jsp:useBean>、<jsp:setProperty>和<jsp:getProperty>动作标识。关于<jsp:useBean>、<jsp:setProperty>和<jsp:getProperty>动作标识的详细介绍,可查看本书 3.5 节中的内容。

JavaBean 也可以用于封装业务逻辑、数据操作等,例如连接数据库,对数据库进行增、删、改、查和解决中文乱码等操作。使用 JavaBean 可以实现业务逻辑与前台程序的分离,提高了代码的可读性与易维护性。

例 5-5 在 JSP 页面中应用 JavaBean 实现字符转换

例如,在实现用户留言功能时,要将用户输入的留言标题和留言内容输出到页面中,如图 5-6 所示。若用户输入的信息中存在诸如“<”和“>”HTML 标识,如输入<input type="text">,则将该内容输出到页面后,会显示一个文本框,如图 5-7 所示,而不是所输入的文

本。解决该问题的方法是在输出内容之前,将内容中的"<"和">"等 HTML 中的特殊字符进行转换,如将"<"转换为"<",将">"转换为">",这样当浏览器遇到"<"时,就会输出"<"字符,如图 5-8 所示。

(1) 先创建填写留言信息的 leave_words.jsp 页面,代码如下:

```
<%@ page contentType = "text/html;charset = gb2312" %>
<form action = "doWord.jsp" method = "post">
    <h2>用户留言</h2>
    标题:<input type = "text" name = "title" size = "26">
    <br>
    内容:<textarea name = "content" rows = "5" cols = "25"></textarea>
    <br><br>
    <input type = "submit" value = "留言">
    <input type = "reset" value = "重置">
</form>
```

(2) 创建名为 Convert 的 JavaBean,在该 JavaBean 中创建一个方法,该方法存在一个 String 型参数,在方法体内编码实现对该参数进行字符转换的操作。Convert 类的代码如下:

```
package com.ycl.bean;
public class Convert {
    public static String change(String str){
        str = str.replace("<","&lt;");
        str = str.replace(">","&gt;");
        return str;
    }
}
```

(3) 创建表单处理页 doWord.jsp,在该页面中首先通过 page 指令导入 Convert 类,然后获取表单数据,接着调用 Convert 类中的 change()方法转换表单数据。doWord.jsp 页面的代码如下:

```
<%@ page contentType = "text/html;charset = gb2312" %>
<%@ page import = "com.ycl.bean.Convert" %>
<%
    String title = request.getParameter("title");       //获取留言标题
    String content = request.getParameter("content");   //获取留言内容
    if(title == null) title = "";
    if(content == null) content = "";
    title = Convert.change (title);              //调用 change()方法转换标题中的"<"和">"字符
    content = Convert.change (content);          //调用 change()方法转换内容中的"<"和">"字符
%>
标题: <% = title %>
<br>
内容: <% = content %>
```

（4）访问 leave_words.jsp 页面，如图 5-6 所示，输入标题，如“< input type＝"text">”，单击“留言”按钮，将出现如图 5-8 所示的运行结果。如果在 doWord.jsp 页面中没有使用 Convert 类中的 change()方法转换表单数据，将会出现如图 5-7 所示的运行结果。

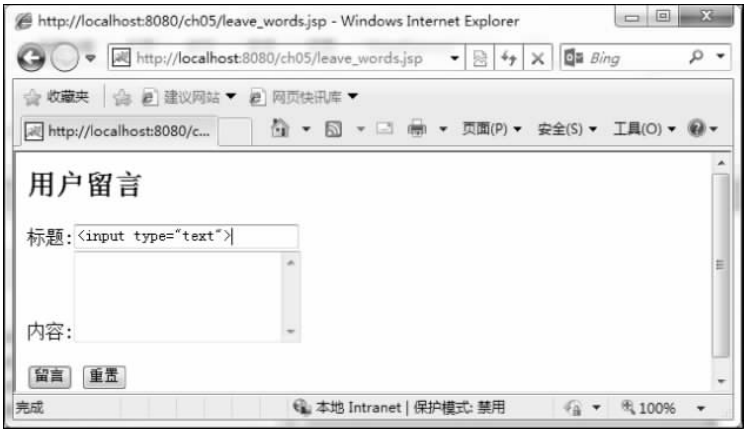

图 5-6 用户留言界面

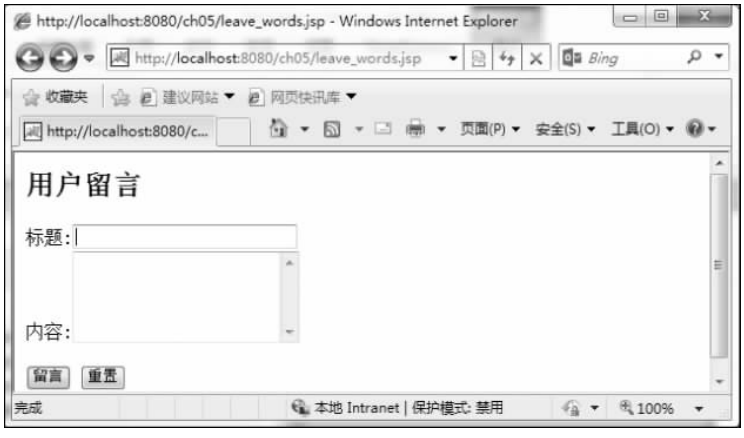

图 5-7 没有字符转换的结果

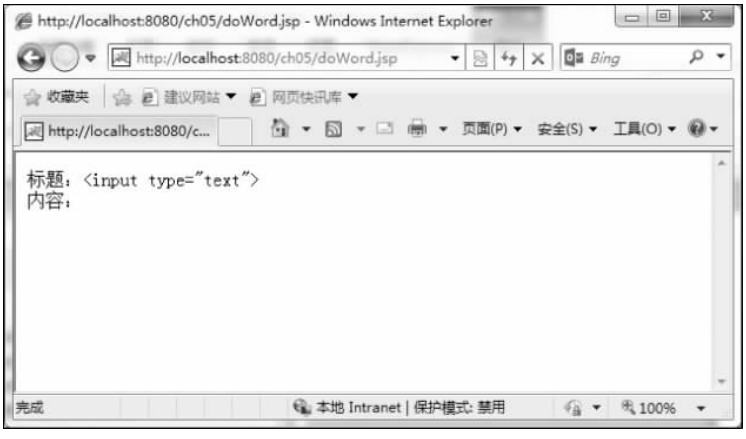

图 5-8 字符转换后的结果

实训 17　应用 JavaBean 封装数据库访问操作(需配置数据库)

(1) 创建封装数据库访问操作的 JavaBean——ConnDB。在该 JavaBean 中定义了用来连接数据库的属性和方法。ConnDB 类的关键代码如下：

```
package com.user.bean;
import java.sql.*;
import java.io.*;
import java.util.*;
public class ConnDB
{
    public Connection conn = null;
    public Statement stmt = null;
    public ResultSet rs = null;
    private static String dbDriver = "sun.jdbc.odbc.JdbcOdbcDriver";
    private static String dbUrl = "jdbc:odbc:shopData";
    private static String dbUser = "sa";
    private static String dbPwd = "";
    //打开数据库连接
    public static Connection getConnection()
    {
        Connection conn = null;
        try
        {
            Class.forName(dbDriver);
            conn = DriverManager.getConnection(dbUrl,dbUser,dbPwd);
        }
        catch(Exception e)
        {
            e.printStackTrace();
        }
    if(conn == null)
    {
          System.err.println("警告:数据库连接失败!");
    }
        return conn;
    }
    //读取结果集
    public ResultSet doQuery(String sql)
    {
        try
        {
            conn = ConnDB.getConnection();
            stmt = conn.createStatement(ResultSet.TYPE_SCROLL_INSENSITIVE, ResultSet.
CONCUR_READ_ONLY);
            rs = stmt.executeQuery(sql);
        }
        catch(SQLException e)
        {
```

```
            e.printStackTrace();
        }
        return rs;
    }
    //更新数据
    public int doUpdate(String sql)
    {
        int result = 0;
        try
        {
            conn = ConnDB.getConnection();
            stmt = conn.createStatement(ResultSet.TYPE_SCROLL_INSENSITIVE, ResultSet.
CONCUR_READ_ONLY);
            result = stmt.executeUpdate(sql);
        }
        catch(SQLException e)
        {
            result = 0;
        }
        return result;
    }
    //关闭数据库连接
    public void closeConnection()
    {
        try
        {
            if(rs!= null)
              rs.close();
        }
        catch(Exception e)
        {
            e.printStackTrace();
        }
        try
        {
            if(stmt!= null)
              stmt.close();
        }
        catch(Exception e)
        {
            e.printStackTrace();
        }
        try
        {
            if(conn!= null)
              conn.close();
        }
        catch(Exception e)
        {
            e.printStackTrace();
        }
```

```
    }
}
```

(2) 编写实现用户登录信息验证的 JSP 文件 login_success.jsp,具体代码如下:

```
<%@ page contentType = "text/html;charset = gb2312" %>
<%@ page import = "com.user.bean.ConnDB" %>
<%@ page import = "java.sql.*" %>
<%
    String c_name = (String)request.getParameter("c_name");
    String c_pass = (String)request.getParameter("c_pass");
    String cname = (String) session.getAttribute("c_name");
    String header = "";
    String name = "",pass = "";
    ConnDB conn = new ConnDB();
    if(c_name!= null || c_name!= "")
    {
        try
        {
            String strSql = "select c_name,c_pass,c_header from customer where c_name = '" + c_
name + "' and c_pass = '" + c_pass + "'";
            ResultSet rsLogin = conn.doQuery(strSql);
            while(rsLogin.next())
            {
                name = rsLogin.getString("c_name");
                pass = rsLogin.getString("c_pass");
                header = rsLogin.getString("c_header");
            }
        }
        catch(Exception e)    {}
    if(name.equals(c_name) && pass.equals(c_pass))
    {
        session.setAttribute("c_name",c_name);
        session.setAttribute("c_header",header);
            %>
                <jsp:forward page = "login.jsp"/>
            <%
    }
    else
    {
        out.println( "<script language = 'JavaScript'>alert('用户名或者密码错误,请重新登录');
window.location.href = 'login.jsp';</script>");
    }
}
 %>
```

login_success.jsp 文件通过调用 ConnDB 的 doQuery 方法实现数据库的连接,根据所输入的用户名和密码执行查询,以实现用户名和密码的验证。

(3) 编写登录页面 login.jsp,即可完成用户登录的数据库验证。

5.3 小结

本章介绍了 JavaBean 的相关概念,以及 JavaBean 在 JSP 中的应用。在介绍 JavaBean 的应用时,首先介绍了如何在 MyEclipse 中创建 JavaBean,然后介绍了如何在 JSP 页面中应用 JavaBean,最后本章给出了应用 JavaBean 开发的一个综合实例。通过学习本章,读者可以熟悉 JavaBean 的基本概念,并且掌握 JavaBean 的实际应用,为以后更深入地学习打好基础。

习题

5-1 一个标准的 JavaBean 具有哪些特征?

5-2 怎样实现 JavaBean 的一个属性与输入参数关联?怎样实现 JavaBean 中的所有属性与请求参数关联?