第 24 章　JSP＋Struts开发实例

本章将使用 Struts 实现一个用户登录系统实例。通过该实例的讲解，可以让读者更加轻松地掌握 Struts 知识。

本章要点包括以下内容：

- 创建一个 Struts 项目
- Struts 项目的具体开发过程
- Struts-config.xml 详细配置方法
- 资源文件的使用

24.1　创建Struts项目

本章节实现的用户登录系统比较简单，但是从该实例可以体会和领悟到 Struts 的原理和使用过程。选择用户登录应用作为实例有如下几个理由：

- 用户登录基本上是所有系统所必需的。登录系统的好处直接决定着整个系统的安全性能。通过登录系统可以区分用户权利等级，所以登录系统对整个系统来说是非常重要的。
- 用户登录应用对大部分读者来说更加常见和熟悉，容易被读者理解。
- 用户登录应用很有代表性。对 JSP 和数据库的操作和应用都比较全面，所以可以通过这个例子很好的对 JSP 学习。

24.1.1　构建一个J2EE项目

首先使用 Eclipse＋Lomboz 构架一个名为 MyLogin 的 J2EE 项目，Web 模块名为 Login。然后按照第二十三章中所介绍的步骤来安装和配置 Struts。该实例需要创建的 JSP 文件如下所示：

- index.jsp：该系统的主页面，提供“登录”以及“注销”链接，并提示用户是否处于登录状态的信息。
- logon.jsp：用户登录页面，填写用户名、密码和验证码后进行身份验证。

另外，需要创建的 Form 和 Action 类文件如下：

- 一个名为 LoginForm 的 ActionForm 类：接收客户输入的数据，并进行初步的数据合理性校验。
- 一个名为 LoginAction 的 Action 类：进行用户身份的校验，根据结果返回 ActionForward 类。
- 一个名为 LogoutAction 的 Action 类：进行用户注销操作。

还需要创建一个名为 Constants.java 类的静态类，用来保存系统中一些常量。资源文件 MessageResources.properties 用来保存一些提示信息。

最后一步是进行 Struts-config.xml 文件的配置，该配置文件放置在 Web 模块的 WEB-INF 目录下。接下来一步步实现该用户登录系统。

24.1.2　引用JavaBean类

本章直接将第 14 章创建的 User.java 接口类、AbstractUser.java 抽象类以及 SqlUser.java 类复制到本章创建的 MyLogin 项目的 cn.login.model 包中；将 Factory.java 接口类、SqlFactory.java 类复制到 MyLogin 项目的 cn.login.model 包中；将 DBConnect.java 数据库操作类复制到 MyLogin 项目的 cn.login.db 包中；另外，将 MD5.java 类也复制到 cn.login.model 包中。

注意：复制完这些类之后，由于这些类所在包名都变了，所以一定要修改各个类的包名。

24.1.3　Struts项目开发步骤

由于 Struts 是建立在 MVC 设计模式上的框架，所以可以遵从标准的开发步骤来开发这里的用户注册系统实例，具体的步骤如下：

（1）定义并生成所有代表应用程序的用户接口 Views，同时生成这些 Views 所用到的所有 ActionForms，并将它们添加到 struts-config.xml 文件中。

（2）在 ApplicationResource.properties 文件中添加必要的 MessageResource 项目。

（3）生成应用程序的控制器。

（4）在 struts-config.xml 文件中定义 View 与 Controller 的关系。

（5）生成应用程序所需要的 model 组件。

（6）编译、运行应用程序。

24.2　Struts开发

接下来创建该实例中用于显示的 JSP 文件和进行逻辑处理的 ActionForm 和 Action 类文件。

24.2.1　创建登录首页index.jsp页面

在 Login 模块的根目录下创建一个 index.jsp 文件，该页面主要提供“登录”和“注销”链接，并提示用户登录状态。读者从该页面中主要学习如何使用 Struts 提供的标签。在下面源代码中，Struts 标签的部分已经使用黑体形式显示出来：

```
<%@ page contentType="text/html; charset=GBK" %>
<%@ taglib uri="/tags/struts-bean" prefix="bean" %>
<%@ taglib uri="/tags/struts-html" prefix="html" %>
<%@ taglib uri="/tags/struts-logic" prefix="logic" %>
<html>
<head>
<html:base/>
<link rel="stylesheet" href="include/CSS.CSS" type="text/css">
<title>首页</title>
</head>
<body background="images/bg.gif">
<logic:present name="user">
   <h4>欢迎 <bean:write name="user"/>!</h4>
</logic:present>
```

```
<logic:notPresent scope="session" name="user">
      <h4>您还未登录</h4>
</logic:notPresent>
<html:errors/>
 <table width="70%"   border="0" cellspacing="0" cellpadding="0" align="center">
  <tr>
    <td colspan="2" align="center"> <h3>Welcome to E-Shopping!</h3></td>
  </tr>
  <tr>
    <td width="40%"> </td>
    <td> <html:link forward="login">登录</html:link></td>
  </tr>
  <tr>
    <td> </td>
    <td> 
     <logic:present name="user">
     <html:link forward="logout">注销</html:link>
       </logic:present>
    </td>
  </tr>
  <tr>
    <td> </td>
    <td> </td>
  </tr>
 </table>
</body>
</html>
```

程序说明：

（1）首先来看<%@ taglib uri=”/tags/struts-html” prefix=”html” %>，它是对使用到的 Struts 标签进行声明。Struts 共有五类标签（上一章中已经介绍，在 MyLogin 项目中安装 Struts 时，需要将这五个 tld 文件复制到 Login/WEB-INF/tlds 目录中，每一个 tld 文件对应一个标签库），本页总共使用到了三种标签。看这个声明中的各个属性意思：

- uir 属性值对应于在 web.xml 文件中标签配置的<taglib-uri>子项。
- prefix 属性值定义了此类标签的一个标记名，例如<html:text property=”username” />中的 html 就是使用的这个标记名，而且必须相同于 prefix 指定的标记名，不然容器无法识别此类标签。

（2）<html:base/>产生一个标准的 HTML base 标记，以便对外在文件引用要相对于这个 JSP 页面存放的目录位置。该标签是一个没有属性和体的单独标签。

例如引用外在文件<link rel="stylesheet" href="include/CSS.CSS" type="text/css">语句，如果存在一个名为 admin.jsp 文件是创建在 admin 目录下，而 CSS.CSS 文件是在上一目录下的 include 文件夹中。根据以往 JSP 开发经验，这里引用地址应该是“../include/CSS.CSS”，其实不然，这是由于 struts 在对所有文件引用时都是从根目录开始，除非在头部加上了<html:base/>标签，它表示所有文件引用从本页面文件的目录开始，这时引用地址才是“../include/CSS.CSS”。这一点读者要非常主要，不然系统会找不到文件。

（3）<logic:present>…</logic:present>标签的使用，使得开发人员感觉到会话好像是内建在 HTTP 里面一样，Struts 标签和 Servlet 容器一起自动维护了会话。

在本页面中，如果用户 user 登录，将使用一个专门的欢迎信息，并且提供“注销”链接。

<logic:notPresent>…</logic:notPresent>标签使用方法类似，它表示用户 user 未登录时，所要做的任务。

所有的 Struts 逻辑（logic）标签都使用了“this”和“notThis”的格式。

（4）<html:errors/>标签用来显示出所有可能发生的错误信息，如果没有错误信息，标签则什么都不输出，就像从页面消失一样。例如，用户名不存在或者密码输入错误的时候，Action 类会把相应的所有错误通过 ActionServlet 返回给这个页面，并在<html:errors/>处显示出来。

在这个标签中还可以指定一个属性，例如，<html:errors property=”username”/>和前面<html:errors/>一样，都是用来显示错误信息的，但是它只是显示一个关键字为 username 的这条错误信息。这是因为 Struts 的错误信息是一条条地被保存起来的，每一条错误信息都对应一个关键字，请看 LoginAction 类代码。

（5）<html:link forward=”login”>登录</html:link>给“登录”提供一个链接，login 作为关键字来查询 struts-config.xml 配置文件，并递交给下一步处理。这里不一定直接转到下一个页面，也有可能通过 struts-config.xml 配置文件而转移到一个 Action 类进行处理，ActionServlet 控制器再通过 Action 返回的 ActionForward 转移到某个页面。

如果想改变这个链接，只需要在配置文件里面进行修改就行了。这样使得页面转换可以集中在 struts-config.xml 配置文件进行集中管理。

当和<logic:present>结合使用时，表示仅当该用户已经登录时才显示这个链接。

24.2.2　创建用户登录login.jsp页面

创建的 logon.jsp 页面负责向用户提供登录窗口，用户输入用户名、密码以及验证码后进行表单提交，然后由 ActionMapping 转交给后台 Form 和 Action 进行相应处理。该创建的 JSP 文件也放置在 Login 模块的根目录下，具体代码如下：

```
<%@ page contentType="text/html; charset=GBK" %>
<%@ taglib uri="/tags/struts-html" prefix="html" %>
<html>
<head>
<META HTTP-EQUIV="Pragma" CONTENT="no-cache">
<META HTTP-EQUIV="Cache-Control" CONTENT="no-cache">
<META HTTP-EQUIV="Expires" CONTENT="0">
<link rel="stylesheet" href="include/CSS.CSS" type="text/css">
<title>用户登录</title>
</head>
<font color="#FF0000">
<html:errors property="username"/><br>
<html:errors property="password"/><br>
<html:errors property="chkma"/><br>
</font>
<center>
<body background="images/bg.gif">
  <html:form action="/loginAction" focus="username">
    <table width="170" border="0" cellspacing="0" cellpadding="0">
      <tr>
        <td><img src="images/userlogin.gif" border="0"></td>
      </tr>
    </table>
```

```
    <table width="170" bgcolor="D2E8FF" style="border-bottom:1px solid #0079CE; border-left:1px solid #0079CE; border-right:1px solid #0079CE ">
        <tr><td height="5"></td></tr>
        <tr><td><img src="images/username2.gif"></td><td>
                <html:text property="username" size="15"/>
        </td></tr>
        <tr><td width="56"><img src="images/userps2.gif"></td><td width="113">
                <html:password property="password" size="15"/>
        </td></tr>
        <tr><td width="56"><img src="images/userym2.gif"></td><td width="113">
                <html:text property="chkma" size="5"/> 
                <img border=1 src="createMa.jsp">
        </td></tr>
    <tr>
          <td align="center" colspan="2" height="30">
           <html:submit value="登录"></html:submit> 
           <html:reset value="重置"></html:reset>
    </td></tr>
    <tr><td colspan="2" align="right"></td></tr>
    </table>
  </html:form>
</center>
</body>
</html>
```

程序说明：

（1）此处的<html:errors/>标签中添加了 property 属性，表示不是所有的错误都显示，只是显示关键字指定的错误，请查看 LoginForm 或者 LoginAction 类中对错误信息的存储方式。如图 24.1 所示显示的错误提示。

图 24.1　错误提示

（2）<html:form action=”/loginAction” focuse=”username”>标签生成一个 HTML 表单供数据输入，并提交。它也产生了一个简单的 JavaScript 来将光标停止在“username”输入域上。

- Action 属性是对配置文件中的 ActionMapping 的引用。它告诉表单哪个 JavaBean helper 类将用来组装这个 HTML 控件。JavaBean helper 是基于 Struts 框架类 ActionForm。

- Focuse 属性定义的是表单默认的焦点位置，其中 username 属性值对应用户名文本矿的 property 属性值。

（3）<html:text>标签定义创建一个 HTML 输入控件来供文本域输入。<html:password>用于输入密码，类似于<html:text>标签，只是它显示文本为*而不是输入的字符。这里也是使用相应的 ActionForm 中的 username、password 和 chkma 属性来组装这个页面中的输入域的，见 LoginForm 类源代码。

如果表单被返回校验，缺省的情况下，username、password 和 chkma 输入域会重显示先前的值，而不用重新输入。除非你把 redisplay 属性设置为 false。

（4）<html:submit>和<html:reset>标签创建了 HTML 标准的 submit 和 reset 按钮。并可以通过 value 属性设置，指定按钮上显示的名称，例如，<html:submit value=”登录”/>。

当表单提交时，将涉及到两个框架对象：ActionForm（描述 HTML form 所需的属性，例如，username、password 和 chkma）和 Action（处理提交的表单），见下面的 LoginForm 和 LoginAction 的源代码。这两个对象必须由开发人员创建并包含应用细节。然后 ActionServlet 并使用 struts-config.xml 配置文件来决定使用哪个 ActionForm 和 Action 类，见配置的 struts-config.xml 文件。

24.2.3　创建LoginForm.java类

当在 login.jsp 页面提交登录信息之后，ActionServlet 控制器会自动根据 struts-config.xml 文件中的配置信息，将 login.jsp 表单中的属性值提交给一个 ActionForm 类进行数据的封装。下面即创建这个名为 LoginForm 的 ActionForm 类，代码如下：

```
package cn.login;
import javax.servlet.http.HttpServletRequest;
import org.apache.struts.action.ActionError;
import org.apache.struts.action.ActionErrors;
import org.apache.struts.action.ActionForm;
import org.apache.struts.action.ActionMapping;
public class LoginForm extends ActionForm{
    //由于继承了序列化接口，所以存在一个静态的常量
     private static final long serialVersionUID = 1L;
     private String username = null;                    //用户名
     private String password = null;                    //用户密码
     private String chkma = null;                       //验证码
     public void setUsername(String username){          //对应属性的 set 方法
          this.username = username;
     }
     public String getUsername(){                       //对应属性的 get 方法
          return username;
     }

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

     public void setChkma(String chkma)
```

```
    {
        this.chkma = chkma;
    }
    public String getChkma()
    {
        return chkma;
    }
                                                    //对即将封装的数据进行初步的检验
    public ActionErrors validate(ActionMapping mapping,HttpServletRequest request){
        ActionErrors errors = new ActionErrors();
        if((username==null)||(username.length()<1))
        {                                           //用户名不能为空或者 NULL
            ActionError error= new ActionError("error.username.required");
            errors.add(Constants.USERNAME,error);
        }
        if((password==null)||(password.length()<6))
        {                                           //密码不能为 NULL 或者小余六位
            ActionError error= new ActionError("error.password.required");
            errors.add(Constants.PASSWORD,error);
        }
        if((chkma==null)||(chkma.length()!=4))
        {                                           //验证不能为空，而且必须是 4 位
            ActionError error= new ActionError("error.chkma.required");
            errors.add(Constants.CHKMA,error);
        }
        return errors;
    }
}
```

程序说明：

（1）这个 LoginForm 类必须继承 Struts 框架中的 ActionForm 类。

（2）这里的 serialVersionUID 变量定义了一个 long 型常量。当不加这一行代码，Eclipse 编译器会出现一个警告信息，这是因为父类 ActionForm 实现了 Java 的序列化接口，序列化类要求应该有这么一个常量。

（3）在这个类中并定义了三个属性变量：username、password 和 chkma，这对应着 login.jsp 页面的中的 HTML form 表单中的属性。对应着这三个属性，并定义了各自的 setXxxx()和 geXxxx()方，这里的 Xxxx 代表类中定义的属性变量，但是首字母必须大写。

（4）loginForm 重实现了 ActionForm 类中的 validate()方法，ActinForm 类还包含另外一个标准方法：reset()方法。当使用 ActionForm 作为向导工作流的一部分时，reset 方法非常有用。如果 mapping 设置为请求范围，这个方法就没有必要实现。

validate 方法经常只是用于主要的外观校验，即它仅仅检查数据“看起来”正确，返回一个 ActionErrors 对象。

（5）ActionError 对象保存单个错误信息，ActionErrors 对象保存一系列 ActionErrror 对象，并且给每个 ActionError 对象指定一个关键字。Struts 框架会自动把返回的 ActionErrors 对象中的错误信息显示在页面中<html:errors/>标注的地方。

（6）本类中的 validate()方法定义了用户名不能为空、密码不能少于六位以及验证码必须是四位。

（7）这里的“error.username.required”语句是资源文件 MessageResources.properties 中相对应的字

符串“用户名不能为空”的一个键值，“error.password.required”和“error.chkma.required”相类似。

把页面上的所有提示语句都写入到资源文件中，这样有利于编码的国际化，当需要汉化一个英文版应用系统时，只需要把对应的资源文件翻译成中文即可。后面对资源文件 MessageResources.properties 的使用和编写作了详细介绍（在前面的第十五章和第十七章标准标签库对国际化问题有重点讲解）。

有读者可能会疑惑，资源文件是放置在 Login\WEB-INF\classes\cn\login 目录下，而这个类仅仅通过“error.username.required”键值怎么能找到相对应的“用户名不能为空”字符串。这是因为在 struts-config.xml 文件（详细请看后面小节对配置文件的介绍）中已经配置了代码：<message-resources parameter="cn.login.MessageResources" />。

（8）Constants.USERNAME 其实就是调用了 Constants 类中的静态属性 USERNAME，它对应属性值为“username”，其他 Constants.PASSWORD 和 Constants.CHKMA 相类似。详细请看 Constants 类的讲解。

（7）LoginForm 类和之前章节中创建的 User 以及实现类 SqlUser 存在很多相似，但是并不应该使用 LoginForm 类取代 User 等类，因为它们使用的范围是不一样的，User 类是依照数据库表创建的，特别是在使用持久层技术时。

24.3.4　创建LoginAction.java类

LoginAction.java 类是使用来对用户名和密码进行验证，并且进行响应操作。LoginAction.java 类用来处理 LoginForm.java 类所封装的数据。这里的 LoginAction.java 类必须要继承 Struts 框架中的 Action 类，详细代码如下：

```
package cn.login;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import org.apache.struts.action.Action;
import org.apache.struts.action.ActionForm;
import org.apache.struts.action.ActionForward;
import org.apache.struts.action.ActionMapping;
import org.apache.struts.action.ActionMessage;
import org.apache.struts.action.ActionMessages;
import cn.login.model.*;
public class LoginAction extends Action{
    //唯一自动执行的方法
    public ActionForward execute(ActionMapping mapping,
                                 ActionForm form,
                                 HttpServletRequest request,
                                 HttpServletResponse response) throws Exception
  {
        LogonForm actionForm = (LogonForm)form;          //获取到封装了页面数据的 LogonForm 类
        String username = actionForm.getUsername();
        String password = actionForm.getPassword();
        String chkma = actionForm.getChkma();
        HttpSession session = request.getSession();
        String rand = (String)session.getAttribute("rand");    //取出保存在 session 中的验证码
        ActionMessages messages = new ActionMessages();
                                                         //下面是对用户输入信息进行验证
        if(chkma.equals(rand))
```

```
        {                                                    //如果输入验证码正确
            SqlUser sqluser = Factory.getInstance().InitSqlUser();
             //调用 userFactory 类中的 ChkUser()方法验证用户身份
             String info = sqluser. checkUser (username,password);
             if(info.equals("1")){                           //用户名不存在
                        ActionMessage message
                         = new ActionMessage("logon.fail.username",username);
                        messages.add(Constants.USERNAME,message);
                        saveErrors(request,messages);
                        return (mapping.findForward(Constants.FAIL));
                    }
             else if(info.equals("2")){                      //用户名密码错误
                        ActionMessage message = new ActionMessage("logon.fail.password");
                        messages.add(Constants.PASSWORD,message);
                        saveErrors(request,messages);
                        return (mapping.findForward());
                    }
             else{                                           //登录身份正确
                       session.setAttribute(Constants.CURRENT_USER,username);
                        return (mapping.findForward(Constants.SUCCESS));
                   }
         }
        else
        {                                                    //验证码不正确
            ActionMessage message = new ActionMessage("logon.fail.chkma");
            messages.add(Constants.CHKMA,message);
            saveErrors(request,messages);
            return (mapping.findForward(Constants.FAIL));
        }
    }
}
```

程序说明：

（1）这个类中最为重要的方法就是 execute()，它是重实现了父类 Action 中的 execute()方法，这个方法并提供了四个参数，详细讲解如下：

- ActionMapping 对象用于查找 struts-config.xml 配置文件的 LoginAction 配置信息。详细请看类中对 ActiomMapping 对象的调用
- ActionForm 对象是页面中 HTML form 表单数据的封装类，它对这些数据进行了描述。由 struts-config.xml 配置文件来指定哪个 ActionForm 类来对表单数据进行封装并传递到相应的 Action 类中。这里传递是 LogonForm 类。
- HttpServletRequest 和 HttpServletResponse 对象和 JSP 页面中使用的内置对象 request 和 response 相对应，同样是负责输入请求和输出请求。

另外还有一个 perform()，作用相类似，但是现在已经不推荐使用 perform()方法，因为 execute()方法提供了更好的例外处理。

（2）execute()方法体中的前四行是提取出封装在 LoginForm 对象中的表单数据。这里包括用户名 username、密码 password 以及验证码 chkma。

（3）第五行的 HttpSession session = request.getSession()语句获得一个 HttpSession 对象（相当于 JSP 中的内置对象 session，负责会话），然后通过这个对象获取到保存在 JSP 的 session 中的验证码信息（详

细见第十六章中创建的 createMa.jsp 文件）。

下面的程序判断输入的验证码 chkma 和页面生成的验证码 rand 是否符合，并对用户身份进行验证，根据不同结果返回不同的 ActionForward 对象，交给 ActionServlet 控制器作不同的处理。

（4）SqlUser sqluser = Factory.getInstance().InitSqlUser()代码是使用工厂类 Factory 类初始化一个 SqlUser 类；SqlUser 类中定义了 checkUser ()方法，来对用户的身份进行验证。不同 info 返回值代码不同的处理结果。SqlUser、Factory 类都是在第十四章中创建的。

（5）ActionMessages 类是用来存在多条信息的，它保存的是 ActionMessage 对象，并给予一个关键字。下面介绍 ActionMessage 类的两种常用方法：

- new ActionMessage("logon.fail.password")：同样这里的"logon.fail.password"键值对应资源文件中的字符串“密码错误”。并把这个字符串信息以 ActionMessage 对象保存起来。
- new ActionMessage("logon.fail.username",username)："logon.fail.username"键值对象资源文件中的“用户名{0}不存在”，然后 username 字符串会嵌入到{0}的位置上。

（6）messages.add(Constants.USERNAME,message)语句是把名为 message 的 ActionMessage 对象（其中保存了字符串信息）保存到 ActionMessages 对象中，并给予一个关键字 Constants.USERNAME（属性值为“username”）。在<html:errors property=”username” />中，属性 property 指定的值就对应着 ActionMessages 保存时所指定的关键字。

其实 ActionMessages 就是一个 Java 的 HashMap，了解 Java 的读者应该对这个很熟悉。

（7）这里 saveErrors(request,messages)是调用的父类 Action 中的方法，它将所有添加到 ActionMessages 中的消息保存起来。只有这样，在页面的<html:errors/>位置才能显示出这些信息。

（8）return (mapping.findForward(Constants.SUCCESS))：Constants.SUCCESS 对应字符串“success”，其中名为 mapping 的 ActionMapping 对象包含了 struts-config.xml 文件中关于 LogonAction 的配置信息，从配置信息中可以看到 success 字符串对应/index.jsp 页面。

（9）如果验证的用户身份合理，则把用户名信息保存在 session 当中。代码为 session.setAttribute(Constants.CURRENT_USER,username)，其中 Constants.CURRENT_USER 对应字符串“user”。

24.2.5　创建LogoutAction.java类

在 index.jsp 页面文件中已经介绍，<logic:present name=”user”>标签首先会检查会话上下文中是否被 LogonAction 类放置一个关键字为 user 的用户信息到 session 中。

如果会话中有 user 用户信息，则<html:link>标记会引用 forward：

```
<html:link forward=”logout”>注销</html:link>
```

另外在 struts-config.xml 中关于 logout 的配置信息如下：

```
<forward
    name="logout"
    path="/logout.do"/>
```

这里的路径引用到了.do 动作，在 struts-config.xml 配置定义为：

```
<action
    path="/logout"
    type="org.apache.struts.actions.LogoutAction"
    scope="request">
        <forward name="return_index" path="/index.jsp"/>
</action>
```

这段配置告诉 struts 把控制传递给 LogoutAction 类（在包 org.apache.struts.actions 中），这里没有任何的参数传递（不像前面传递一个封装了数据的 loginForm 类给 loginAction 类）。以下的 logoutAction 类的工作也非常的简单，它只是把用户信息从会话（session）上下文中移除掉。

详细的 logoutAction.java 代码如下：

```
package cn.login;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;
import org.apache.struts.action.Action;
import org.apache.struts.action.ActionForm;
import org.apache.struts.action.ActionForward;
import org.apache.struts.action.ActionMapping;
import cn.login.Constants;
public class LogoutAction extends Action{
      public ActionForward execute(ActionMapping mapping,
                  ActionForm form,
                  HttpServletRequest request,
                  HttpServletResponse response) throws Exception
     {
            HttpSession session = request.getSession();
            String username = (String)session.getAttribute(Constants.CURRENT_USER);
            if(username!=null)                  //如果用户处于登录状态，则把用户信息从 session 中除去
                  session.removeAttribute(Constants.CURRENT_USER);
            return (mapping.findForward(Constants. RETURN_INDEX));
     }
}
```

程序说明：

在 execute()方法体的第一、二行中，调用 HttpSession 对象中的 getAttribute()方法获取到关键字为 Constants.CURRENT_USER（对应字符串“user”）的用户信息。如果会话上下文中不存在这个用户，则直接返回带有 Constants.RETURN_INDEX（对应字符串“return_index”）信息的 AcitonForward 对象给 ActionServlet 控制器。如果此用户存在，则调用 removeAttribute()方法把用户信息从会话上下文中移除再返回。

24.2.6　构造Constants类

这个类不是严格需要的，但是像这样将 ActionForward 名称和其他标记记录在文档中则是强烈推荐的。这样使用整个类来记录应用系统中所要用到的常数，可以便于应用代码的管理和后期维护，这样的思想还可以在 struts-config.xml 配置文件以及资源文件 MessageResources.properties 中得到体验。当系统需要适当修改时（变量名，页面跳转等），只要集中在个别文件就可以了。

Constants.java 类的详细代码如下：

```
package cn.login;
public class Constants {
      public static final String CURRENT_USER = "user";
      public static final String USERNAME = "username";
      public static final String PASSWORD = "password";
      public static final String CHKMA = "chkma";
```

```
    public static final String NOTICE = "notice";
    public static final String SUCCESS = "success";
    public static final String FAIL = "fail";
    public static final String RETURN_INDEX = "return_index";
}
```

程序说明：

（1）这里的代码就相当于一个文档，记录所有常数的一个文档。

（2）常数属性变量一般使用全大写字母表示。

（3）这里的所有属性变量都定义为 static 和 final 类型，便于 Constants 类直接调用，以及不允许在外部对这些变量进行修改。

24.2.7　struts-config.xml文件配置

如果不介绍这个 struts-config.xml 配置文件，读者一定会对前面的内容感到很迷惑，怎么知道把 login.jsp 页面中 form 表单中的数据封装到 loginForm 类中，loginForm 类又是怎样对应到 loginAction 的，然后根据 Action 类返回的 ActionForward 对象又是怎么知道下一步要操作的对象？……。这一切都会在 struts-config.xml 配置文件中找到答案，这个配置文件是 struts 学习的关键，读者一定要非常熟练地掌握它。struts-config.xml 就像一个幕后的黑手，它悄悄地控制着这一切。

Struts 的这个配置文件和 ActionServlet 一起工作，来创建应用的控制层。通过下面的实例讲解让读者掌握 struts-config.xml 配置文件的配置和管理。

针对以上实例的 struts-config.xml 配置文件代码如下：

```
<?xml version="1.0" encoding="ISO-8859-1" ?>

<!DOCTYPE struts-config PUBLIC
          "-//Apache Software Foundation//DTD Struts Configuration 1.2//EN"
          "http://jakarta.apache.org/struts/dtds/struts-config_1_2.dtd">
<!-- -->
<struts-config>
<!-- ============================================== Form Bean Definitions -->
    <form-beans>
        <form-bean name="logonForm" type="cn.login.LoginForm"/>
    </form-beans>
<!-- ============================================ Global Forward Definitions -->
     <global-forwards>
        <forward
            name="login"
            path="/login.do"/>
        <forward
            name="logout"
            path="/logout.do"/>
    </global-forwards>
<!-- ========================================== Action Mapping Definitions -->
    <action-mappings>
        <action
            path="/loginAction"
            type="cn. login.LoginAction"
            name="loginForm"
```

```
                validate="true"
                scope="request"
                input="/login.jsp"
                >
                <forward name="success" path="/index.jsp"/>
                <forward name="fail" path="/login.jsp"/>
            </action>
            <action
                path="/logn"
                type="org.apache.struts.actions.ForwardAction"
                parameter="/login.jsp"
                scope="request">
            </action>
            <action
                path="/logout"
                type="cn.login.LogoutAction"
                scope="request">
                <forward name="return_index" path="/index.jsp"/>
            </action>
    </action-mappings>
<!-- ========================================= Message Resources Definitions -->
    <message-resources parameter="cn.login.MessageResources" />
</struts-config>
```

程序说明：

（1）Struts 配置文件可以帮助开发者以最小的努力对应用变更做出快速的反应。如果一个对象需要初始化为另外一个值，这时并不需要编辑、编译和部署一个 Java 类。许多配置细节都涉及到表现层。团体中工作于该层的人员可能不是 Java 工程师。使用 XML 文档可以使配置被页面设计员和项目管理者都能访问到。当需要 Java 工程师来创建或者修改应用中某些类时，可以转交给专门的人员，而配置这些对象则可以委托给其他人。使得工作分工很容易。

实践中，常常将不经常变更的基础 Java 类从经常变更的对象在运行时如何部署中分离出来。这就是受保护的变更原则。

（2）这个配置文件的前两句是通用的，可以从 Struts 提供的例子程序中复制过来。

（3）<form-beans>项是用来定义所有的用到的 ActionForm 类。这里把 LoginForm 类进行了定义，以后可以依次在<form-beans>…</form-beans>体内定义新的 ActionForm 类。常用的属性：

- name 属性给使用到的这个 LoginForm 类取一个别名。
- type 属性定义了 LoginForm 类所在位置，即指定 LoginForm 类的全名。

（4）<global-forwards>项中的<forward>子项定义了 JSP 页面中<html:link>标签所要连接的页面或者类。

- Name 属性指定的值和<html:link>标签中 forward 属性值相一致。
- Path 属性指定了<action>项中定义的 Action 类的别名。

（5）<action-mapping>内部定义的<action>项是非常重要的，由它定义 LoginForm 类和 LoginAction 类之间的关系。<action>项内有多个属性：

- Path 属性是定义这里用到的 LoginAction 类的别名，其中 login.jsp 页面中 form 标签中的 action 属性所指定的值和这里 path 属性值一致。在 IE 浏览器地址栏中，path 属性一般都带.do 后缀。
- Type 属性定义 LoginAction 类的全名。

- 例如在 LoginForm 类中实现了 Validator 方法来对表单中的数据进行初步的验证，这时必须在 struts-config.xml 配置文件的<action>元素中添加 validator=“true”。它表示在调用相应的 Action 类中的 execute()方法之前，ActionServlet 将调用 ActionForm 类中的 validator()方法进行输入检查。Struts 默认得 validator 属性为 false。另外还要设置 input 属性（例如上面 struts-config.xml 的配置），它表示当 bean 发生输入错误的时候（也就是在 LoginForm 类中有错误返回时）必须要返回的页面。在本例中，当发生输入错误时，会自动返回到 login.jsp 页面进行重新输入。
- parameter 属性直接指定需要跳转的页面，这时 type 属性必须设置为 org.apache.struts.actions.ForwardAction，也不需要指定 ActionForm 进行数据的封装。也就是不进行任何业务处理，只是页面转发。
- Name 属性指定的是 LoginAction 类所对应的 LoginForm 类的别名，这个值和<form-beans>项中的 name 属性值一致。Struts 就是通过这个属性来查找相对应的 ActionForm 类。
- Scope 定义了 ActionForm 类的作用域，一般都设置为 request。

（6）第一个<action>…</action>体中的<forward name="success" path="/index.jsp"/>定义了一个页面转向。

- name 是别名，LoginAction 类中的 mapping.findForward(Constants.SUCCESS) 中 Constants.SUCCESS 即为这里 name 属性值。
- Path 属性指定需要转移的页面。

（7）<message-resources>项定义了资源文件的位置，它对应 cn.login 包下的 MessageResources.properties 文件。

24.2.8　资源文件MessageResources.properties

在使用这个资源文件之前，第一步需要定义该文件的名称，这个文件会包含用默认语言编写的在程序中会出现的所有消息。这些消息以“关键字-值”的形式存储。如下所示：

```
error.username.required = 用户不存在
```

这个文件需要存储在类的路径下（在用户登录实例中，此资源文件存储在 WEB-INF\classes\cn\login 目录下，并在 struts-config.xml 配置文件进行了定义），而且它的路径要作为初始化参数传送给 ActionServlet 作为参数进行传递时，路径的格式要符合完整 Java 类的标准命名规范。例如，资源文件名是 MessageResources.properties，并且存储在 WEB-INF\classes 目录下，那么需要传递的参数值是 MessageResources。如果文件在 WEB-INF\classes\cn\login 中，那么参数值就应该是 cn.login.ApplicationResources。

为了实现国际化，所有的资源文件必须都存储在基本资源文件所在的目录中。基本资源文件包含的是用默认地区语言-本地语言编写的消息。如果基本资源文件的名称是 MessageResources.properties，那么用其他特定语言编写的资源文件的名称就应该是 MessageResources_xx_YY.properties(xx 为语言代码，是 ISO 编码，如英语是 en；YY 为地区名，如中国是 CN)。因此这些文件应包含相同的关键字，但关键字的值是用特定语言编写的。

ActionServlet 的区域初始化参数必须与一个 true 值一起传送，这样 ActionServlet 就会在用户会话中的 Action.LOCALE_KEY 关键字下存储一个特定用户计算机的区域对象。现在可以运行一个国际化的 web 站点，它可以根据用户计算机上的设置的区域自动以相应的语言显示。

用户还可以使用特定的字符串来替换部分消息，就象用 java.text.MessageFormat 的方法一样：

```
error.invalid.number = The number {0} is valid
```

用户可以把字符串{0}替换成任何需要的数字。

下面在资源文件中添加消息。上一章已经介绍了 jinto 插件，安装完这个插件之后，就可以在 Eclipse 中使用 Java ResourceBoudle Editor 编辑这创建的资源文件。如图 24.2 所示的编辑界面。

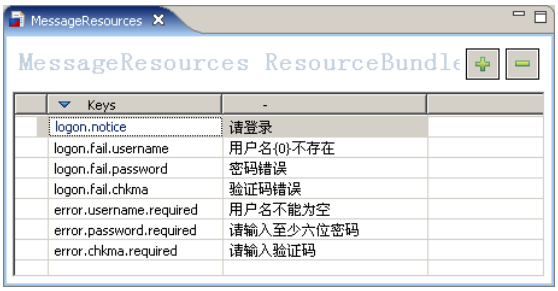

图 24.2　编写资源文件

在这个资源文件中总共添加了七个消息：logon.notice、logon.fail.username、logon.fail.password、logon.fail.chkma、error.username.required、error.password.required 和 error.chkma.required。可以查看 LoginAction 以及 LogoutAction 类对它们的引用。

在整个 Struts 开发过程中，最为重要的就是 Struts-config.xml 配置文件和标签库的编写。有关 Struts-config.xml 配置文件方面的知识在上一章节已经重点介绍过。

24.3　本章小结

本章通过一个用户登录实例具体演示了整个 Struts 开发过程，读者结合上一章有关 Struts 的理论知识可以更好的理解 Struts。Struts 是 MVC 开发模式的一个很好解决方案。下一章将向读者介绍另一个非常流行的技术：Hibernate 数据库持久层技术。