

# 目 录

第 1 章 Struts 和基于 Eclipse 的配置与开发.....	1
1.1 几个基本的技术要点.....	1
1.2 Struts.....	2
1.3 框架 (FrameWork) .....	2
1.4 JSP 开发的两个 MVC Model 方法.....	3
1.4.1 MVC Model.....	3
1.4.2 MVC Model 1 .....	4
1.4.3 MVC Model 2 .....	5
1.4.4 Struts 中的 MVC Model.....	6
1.5 配置 Eclipse 下的 Struts 开发环境.....	7
1.5.1 配置 JDK 环境变量要点 .....	8
1.5.2 Eclipse 的安装很“绿色” .....	9
1.5.3 MyEclipse 的安装.....	9
1.5.4 Tomcat 的安装.....	13
1.5.5 测试开发环境.....	15
第 2 章 Struts 简介 .....	23
2.1 Struts 的工作流程 .....	23
2.2 关于 Struts 的实例 .....	25
2.2.1 添加 Struts 框架支持文件.....	25
2.2.2 视图层 V-View 的添加 .....	28
2.2.3 控制层 C-Controller 的添加.....	33
2.2.4 模型层 M-Model 的添加 .....	37
2.2.5 结尾前小小的改动 .....	40
2.2.6 struts-config.xml 文件 .....	40
2.2.7 部署项目并运行 .....	42
第 3 章 应用中的 C-Controller 控制层 .....	44
3.1 ActionServlet 类的作用.....	44
3.1.1 process()方法的执行过程 .....	47
3.1.2 process()方法执行过程总结 .....	48
3.2 Action 类的作用 .....	49

3.2.1	Action 的工作.....	49
3.2.2	在 Action 类中进行用户名验证的实例 .....	49
3.2.3	ActionErrors 和 ActionError 类的关系和使用 .....	54
3.3	ActionForward 类的功能及两种在 Eclipse 中创建 Action Forward 类的方法.....	56
3.3.1	ActionForward 类的功能 .....	56
3.3.2	在 Eclipse 中创建 ActionForward 类的两种方法.....	56
3.3.3	带参数的 ActionForward .....	57
3.4	使用 ForwardAction 进行页面或 Web 组件的跳转 .....	57
3.4.1	什么情况下使用 ForwardAction .....	58
3.4.2	一个 ForwardAction 类的实例.....	58
3.4.3	第二种创建 ForwardAction 类的方法.....	62
3.5	IncludeAction 让年久失用的 Web 组件复用 .....	62
3.5.1	使用 IncludeAction 包含 JSP 文件.....	63
3.5.2	使用 IncludeAction 包含进 Servlet 组件 .....	68
3.6	DispatchAction 将 Action 类变得更少 .....	75
3.7	LookupDispatchAction 实现一个表单包含多个提交按钮.....	79
3.8	用 SwitchAction 切换不同的 Struts 模块 .....	85
第 4 章	中心配置文件 struts-config.xml .....	94
4.1	Struts 1.2 版 struts-config.xml 文件结构 .....	94
4.2	struts-config.xml 配置文件中的子元素 .....	95
4.2.1	<data-sources />子元素 .....	95
4.2.2	<form-bean />子元素 .....	95
4.2.3	<global-forwards />子元素.....	95
4.2.4	<action-mappings>子元素.....	96
4.2.5	<message-resources>子元素.....	97
4.2.6	关于配置 struts-config.xml 文件 .....	97
4.2.7	元素详解 .....	99
4.2.8	attribute 和 name 的区别.....	105
第 5 章	V-View 视图层中的 ActionForm .....	107
5.1	ActionForm 类的结构 .....	107
5.2	ActionForm 生命周期 .....	108
5.3	DynaActionForm 使用方法 .....	109
5.4	DynaActionForm 实例.....	110
5.5	Action 和 ActionForm 配置精例 .....	115
5.5.1	完整的 action 功能 .....	115
5.5.2	仅有 Form 的 action 超级链接功能 .....	116
5.5.3	仅有 Action 的 action 执行链接式请求后就转发.....	116

5.5.4 仅有 JSP 的 action .....	117
5.5.5 两个 action 对应一个 Form.....	117
5.5.6 两个 action 对应两个 Form.....	118
5.6 ActionForm 中文乱码问题解决方案.....	118
5.7 Struts 中的 ActionErrors.....	119
<b>第 6 章 Struts-HTML 标签库 .....</b>	<b>120</b>
6.1 普通 HTML 与 Struts 中 HTML 标签的区别及 Struts 标签公共特征 .....	120
6.2 显示 Struts-HTML 标签的 Snippets 窗口 .....	121
6.3 <html:base />设置相对根路径 .....	123
6.3.1 标签简介 .....	123
6.3.2 使用示例 1 .....	123
6.3.3 使用示例 2 .....	124
6.4 <html:cancel />取消当前的提交.....	126
6.4.1 标签简介 .....	126
6.4.2 使用示例 .....	126
6.5 <html:checkbox />复选框.....	130
6.5.1 标签简介 .....	130
6.5.2 使用示例 .....	130
6.5.3 <html:checkbox />复选框在 Action 中状态的改变 .....	132
6.5.4 <html:checkbox />复选框的初始化 .....	133
6.5.5 在 Action 中通过数据库的数据控制<html:checkbox />复选框的选中状态.....	134
6.6 <html:errors />出错提示.....	138
6.6.1 显示局部错误信息 .....	138
6.6.2 显示全局错误信息 .....	140
6.6.3 生成错误信息在不同版本使用上的区别 .....	142
6.7 <html:file />文件上传功能.....	142
6.7.1 定制 JSP 页面.....	142
6.7.2 定制 ActionForm 类.....	142
6.7.3 设计重要的 Action 类.....	143
6.7.4 查看 struts-config.xml 配置文件 .....	145
6.8 <html:form />表单提交 .....	145
6.9 <html:hidden />保密的数据传送 .....	147
6.9.1 开发实例 .....	147
6.9.2 <html:hidden>如何设置默认值.....	150
6.10 <html:html>定义 HTML 文件 .....	151
6.11 <html:image>定义图像提交按钮 .....	152
6.12 <html:img>在页面上显示图像.....	153
6.13 <html:link>变幻莫测的超级链接 .....	157

6.13.1	Link type 为 action 的情况.....	158
6.13.2	Link type 为 forward 的情况.....	160
6.13.3	Link type 为 href 的情况.....	161
6.13.4	Link type 为 page 的情况.....	162
6.13.5	带参数超级链接的情况.....	163
6.13.6	在<html:link>中嵌入 JSP 脚本.....	165
6.13.7	带参数超级链接的问题解决实例.....	165
6.13.8	给 Struts 中<html:link>标签加确认对话框.....	168
6.13.9	用<html:link>标签生成 BBS 主题列表功能.....	168
6.14	<html:errors>的更新版<html:messages>.....	174
6.15	<html:multibox />分组类型的复选框.....	181
6.15.1	用<html:multibox />做一个选择“爱好”的实例.....	181
6.15.2	<html:multibox />初始化时即呈 checked 状态.....	183
6.16	用<html:select />和<html:option />实现下拉列表.....	187
6.16.1	用<html:select />和<html:option />实现下拉列表.....	187
6.16.2	<html:select />和<html:option />实现列表单选.....	188
6.16.3	<html:select />和<html:option />实现列表多选.....	188
6.16.4	<html:select />和<html:option />标签设置下拉列表初始化值.....	189
6.16.5	<html:select />和<html:option />设置列表单选初始化值.....	190
6.16.6	<html:select />和<html:option />设置列表多选初始化值.....	190
6.16.7	如何获取<html:select />和<html:option />下拉列表单选值.....	192
6.16.8	如何获取<html:select />和<html:option />列表多选值.....	194
6.17	使用<html:options>动态生成<html:select />和<html:option>列表内容.....	195
6.17.1	将数据库的内容动态生成<html:select />和<html:option>列表内容.....	196
6.17.2	初始化<html:select />和<html:options>列表生成的内容.....	198
6.18	使用<html:optionsCollection>动态生成 <html:option>列表内容.....	198
6.18.1	使用<html:optionsCollection>动态生成<html:option>列表内容实例.....	198
6.18.2	使用<html:optionsCollection>标签中的 property 属性来生成下拉列表.....	202
6.19	<html:password>、<html:text>、<html:textarea>标签的使用.....	205
6.19.1	<html:password>标签的 redisplay 属性实例.....	205
6.19.2	使用 style 的 CSS 样式改变<html:text>标签的外观.....	207
6.19.3	<html:textarea>标签的使用.....	208
6.20	<html:radio>标签的使用.....	208
6.21	<html:submit>和<html:reset>标签的使用.....	209
6.22	总结.....	209
第 7 章	Struts-Logic 标签库.....	210
7.1	<logic:iterate>标签的功能.....	210
7.1.1	打印数组中的内容.....	211

7.1.2 打印 HashMap 中的内容 .....	211
7.1.3 打印 ArrayList 中的内容 .....	212
7.2 <logic:redirect>重定向的标签 .....	213
7.3 <logic:forward>转发的标签 .....	214
7.4 <logic:empty>和<logic:notempty>标签的作用 .....	214
7.5 <logic:present>标签的作用和与<logic:empty>的区别 .....	215
7.6 用<logic:equal>和<logic:notEqual> 判断等于和不等 .....	216
7.6.1 使用<logic:equal>和<logic:notEqual>判断变量 .....	217
7.6.2 使用<logic:equal>和<logic:notEqual>判断 Bean 的属性值 .....	217
7.7 用<logic:lessEqual>和<logic:lessThan> 判断小于等于和小于 .....	218
7.7.1 <logic:lessEqual>和<logic:lessThan>判断变量 .....	218
7.7.2 <logic:lessEqual>和<logic:lessThan>判断 Bean 的属性值 .....	219
7.8 用<logic:greaterEqual>和<logic: greaterThan> 判断大于等于和大于 .....	220
7.8.1 <logic:greaterEqual>和<logic: greaterThan>判断变量 .....	220
7.8.2 <logic:greaterEqual>和<logic: greaterThan>判断 Bean 的属性值 .....	221
<b>第 8 章 Struts-Bean 标签库 .....</b>	<b>223</b>
8.1 Bean 标签库的功能 .....	223
8.2 <bean:write />标签打印 Bean 中的属性值 .....	223
8.2.1 <bean:write>标签打印变量 .....	223
8.2.2 <bean:write>标签打印 Bean 的 property 属性值 .....	224
8.2.3 <bean:write>标签 format 属性的应用 .....	227
8.2.4 <bean:write>标签 filter 属性的应用 .....	227
8.3 <bean:parameter />标签读取 HTTP 请求的参数 .....	228
8.3.1 使用<bean:parameter />标签读取单个 http 参数 .....	228
8.3.2 使用<bean:parameter />标签读取数组型 HTTP 参数 .....	229
8.4 <bean:message>标签显示资源文件中的文本消息 .....	230
8.5 在<bean:define>标签中定义变量 .....	235
8.5.1 在<bean:define>标签中定义字符串常量 .....	236
8.5.2 <bean:define>标签复制 Bean .....	236
8.5.3 用<bean:define>标签复制现有 Bean 的属性给新的 Bean 属性 .....	236
<b>第 9 章 关于 Struts 的其他内容 .....</b>	<b>237</b>
9.1 Struts 资源文件国际化 .....	237
9.1.1 MyEclipse 保存资源文件的编码哨兵 .....	237
9.1.2 使用 MyEclipse 资源文件的插件 jinto .....	237
9.2 在 Struts 的 URL 中传递中文参数 .....	238
9.3 从不同的资源文件中显示信息 .....	240
9.3.1 从不同的资源文件中显示信息的实例 .....	240

9.3.2 优化新建资源文件目录结构 .....	243
9.4 没有登录不能访问非 index.jsp 的 JSP 页面 .....	244
9.5 设置应用的默认页面 .....	252
9.6 URL 重写技术 .....	254
9.7 使用 Struts 多语言切换的情况 .....	259
9.8 添加 Struts 包的操作 .....	263
9.9 实现跨页表单的提交 .....	265
<b>第 10 章 简易论坛模型的实例 .....</b>	<b>274</b>
10.1 实例目标 .....	274
10.2 功能模块简介 .....	274
10.3 模块设计 .....	275
10.3.1 用户注册 .....	276
10.3.2 显示主题列表 .....	284
10.3.3 用户登录 .....	295
10.3.4 修改个人用户信息 .....	297
10.3.5 查询用户 .....	301
10.3.6 删除用户 .....	307
10.3.7 删除主题及删除回复 .....	308
10.4 总结 .....	309

# 第 1 章

## Struts 和基于 Eclipse 的配置与开发

### 1.1 几个基本的技术要点

在概述什么是 Struts 之前，我们要先弄清楚几个相关的技术术语：Apache、ASF、Jakarta、Tomcat。

#### 1. Apache

Apache 是一种开放源代码的 Web 服务器，它的功能类似于微软的 IIS，但要远远超过 IIS，不管从安全性、跨平台性，还是一些企业级的部署和应用。Apache 作为自由软件之一，像其他自由软件一样，它们都是由许许多多的自由开发人员投入了大量的时间和精力来实现并逐步完善的，这也是 Apache 能成为最流行的 Web 服务器原因之一。不过从第一个版本以来，尽管不断有新的漏洞被发现，但由于其 OpenSource（开放源代码）的特点，漏洞总能被很快修补，因此，其安全性还是相当高的，使用也是最普遍的。

#### 2. ASF

ASF（Apache Software Foundation）是 Apache 软件基金组织的缩写。随着 Apache 服务器的广泛应用，现在的 Apache 已经不仅代表一个软件，而是具有一些开放源代码及企业级应用的软件项目机构。Apache 软件基金会（ASF）正式创建于 1999 年。

#### 3. Jakarta

ASF 这个组织包含了很多软件项目，Jakarta 是 ASF 旗下的一套 Java 解决方案的开源软件项目的名称，它包括了很多子项目，Tomcat、Ant、Struts 等也是 Apache 下的开源项目，同时也是 Jakarta 的关联项目。Jakarta 里的项目主要面向 Java 技术。

#### 4. Tomcat

Tomcat 是一个免费开源的 Servlet 容器，它是 Apache 基金会 Jakarta 项目中的一个核心项目，由 Apache、Sun 和其他一些公司及个人共同开发而成。由于有了 Sun 公司的参与和支持，最新的 Servlet 和 JSP 规范总能在 Tomcat 中得到体现。Tomcat 不仅仅是一个 Servlet 容器，它也具有传统的 Web 服务器的功能：处理 HTML 页面，简单的域名管理，配置 JNDI 等。但是与 Apache 相比，它的处理静态 HTML 的能力远不如 Apache。可以将 Tomcat 和

Apache 集成在一起，让 Apache 处理静态 HTML，而 Tomcat 处理 JSP 和 Servlet，这样可以发挥各自的所长。

## 1.2 Struts

Struts 是 Jakarta 的一个子项目，它提供了一种方法，可以在一个 Web 应用程序中同时使用 Java Server Pages (JSP) 和 Servlet。它的目的是要解决完全由 JSP 或完全由 Servlet 实现的传统应用程序中固有的问题，例如，用 JSP 很难将 Java 代码同网页的数据显示分开，也很容易将 Java 代码同 HTML 混在一起，结果做出的项目运行速度较慢，而且在后期维护中工作量较大。

Struts 只是一个 MVC 框架 (Model/View/Controller Framework)，用于快速开发 Java Web 应用程序，这样以分三层的结构来开发软件项目，不但使开发的结构明了，而且还有助于项目的维护。Struts 的重点在 C (Controller) 控制端，也为 V (View) 视图端提供了一系列定制的标签 (Custom Tag)。但 Struts 几乎没有涉及 M (Model) 模型端，所以 Struts 可以采用 Java 实现的任何形式的商业逻辑。

MVC (Model/View/Controller) 模式是国外用得比较多的一种设计模式，最早在 Smalltalk 编程语言中出现。MVC 包括三类对象：Model 是应用对象，也就是功能逻辑；View 是它在屏幕上的表示，也就是 UI 界面；Controller 定义用户界面对用户输入的响应方式，相当于请求和应答的事件。

使用 Struts 是免费的。

## 1.3 框架 (FrameWork)

框架是一个应用程序的半成品，框架提供了可在应用程序之间共享的可复用的公共结构。开发者把框架融入到他们自己的应用程序中，并加以扩展，以满足他们特定的需要。框架和工具包的不同之处在于：框架提供了一致的结构，而不仅仅是一组工具类。框架其实就是一组组件，供你选用以完成你自己的系统。简单地说就是使用别人搭好的舞台，你来做表演。而且，框架一般是成熟的，不断升级的软件。

可以说，一个框架是一个可复用的设计构件，它规定了应用的体系结构，阐明了整个设计、协作构件之间的依赖关系、责任分配和控制流程，表现为一组抽象类以及其实例之间协作的方法。

打个比方，应用在建筑技术中比较常见的就是钢架结构，以前在建设一所厂房的时候，都是一砖一瓦堆起来，工期时间长不说，而且不坚固；现在的工业技术，使用钢架结构，支起房屋的框架，然后在这个钢架结构上可以设计出不同样式，不同风格的厂房，比如不同颜色的屋顶，不同金属板的墙壁等，这样不仅有利于房屋的拆装和维护，而且在构造时也非常快捷，具有自由性。

可见，建筑工程中的技巧也可以完全应用在软件工程中，给你一个软件的框架，在这个框架之上可以做自定义的功能。



## 1.4 JSP 开发的两个 MVC Model 方法

不管对于 Win 32 软件还是 Web 开发,设计模式的应用无处不在,在 JSP 的标准发布之后,使用 JSP 开发的软件项目大致上可以分为两个方法,或者叫两种模型: MVC Model 1 和 MVC Model 2。

### 1.4.1 MVC Model

MVC 英文即 Model-View-Controller,即把一个应用的输入、处理、输出流程按照 Model、View、Controller 的方式进行分离,这样一个应用被分成 3 个层——模型层、视图层、控制层。

视图 (View) 代表用户交互界面,对于 Web 应用来说,可以概括为 HTML 界面,但有可能为 XHTML、XML 和 Applet。随着应用的复杂性和规模性的不断变化,界面的处理也变得具有挑战性。一个应用可能有很多不同的视图, MVC 设计模式对于视图的处理仅限于视图上数据的采集和处理,以及用户的请求,而不包括在视图上的业务流程的处理。业务流程的处理交予模型 (Model) 处理。比如一个订单的视图只接受来自模型的数据并显示给用户,以及将用户界面的输入数据和请求传递给控制和模型。

模型 (Model): 就是业务流程/状态的处理以及业务规则的制定。业务流程的处理过程对其他层来说是黑箱操作,模型接受视图请求的数据,并返回最终的处理结果。业务模型的设计可以说是 MVC 最主要的核心。目前流行的 EJB 模型就是一个典型的应用例子,它从应用技术实现的角度对模型做了进一步的划分,以便充分利用现有的组件,但它不能作为应用设计模型的框架。它仅仅告诉你按这种模型设计就可以利用某些技术组件,从而减少了技术上的难度。对一个开发者来说,就可以专注于业务模型的设计。 MVC 设计模式告诉我们,把应用的模型按一定的规则抽取出来,抽取的层次很重要,这也是判断开发人员是否优秀的依据。抽象与具体不能隔得太远,也不能太近。 MVC 并没有提供模型的设计方法,而只告诉你应该组织管理这些模型,以便于模型的重构和提高重用性。在这里可以用面向对象编程来做比喻, MVC 定义了一个顶级类,告诉它的子类你只能做这些,但没法限制你能做这些。这点对编程人员来说非常重要。业务模型还有一个很重要的模型,那就是数据模型。数据模型主要指实体对象的数据保存(持续化)。比如将一张订单保存到数据库,从数据库获取订单。我们可以将这个模型单独列出,所有有关数据库的操作只限制在该模型中。

控制 (Controller) 可以理解为从用户接收请求,将模型与视图匹配在一起,共同完成用户的请求。划分控制层的作用也很明显,它清楚地告诉你,它就是一个分发器,选择什么样的模型,选择什么样的视图,可以完成什么样的用户请求。控制层并不做任何的数据处理。例如,用户单击一个连接,控制层接受请求后,并不处理业务信息,它只把用户的信息传递给模型,告诉模型做什么,选择符合要求的视图返回给用户。因此,一个模型可能对应多个视图,一个视图可能对应多个模型。

模型、视图与控制器的分离,使得一个模型可以具有多个显示视图。如果用户通过某

个视图的控制器改变了模型的数据,所有其他依赖于这些数据的视图都应反映到这些变化。因此,无论何时发生了何种数据变化,控制器都会将变化通知所有的视图,导致显示的更新。这实际上是一种模型的变化——传播机制。模型、视图、控制器三者之间的关系和各自的主要功能如图 1-1 所示。

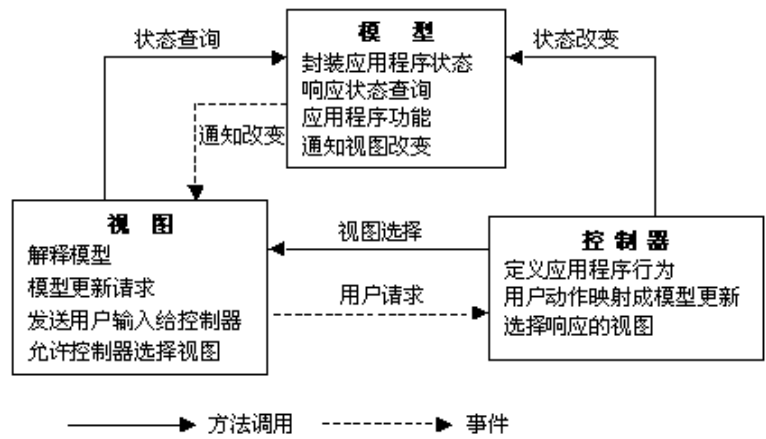


图 1-1 MVC 组件关系和功能

1.4.2 MVC Model 1

在最初开发 JSP 项目过程中,使用最广泛的方法就是 JSP+JavaBean,虽然这是一种对于大型软件项目来说是过时的技术,但这样的方法直到现在有些公司还在乐此不疲地使用,也许是由于项目的需要。虽然这里要学习 Struts,但 JSP+JavaBean 的开发方法还要熟练掌握,它是学习 Struts 的基本功。MVC Model 1 在 UI 图形用户界面端 JSP 文件里,夹杂着大量的 JSP 脚本和 HTML 语言代码及一些 JavaScript 脚本,这就加大了程序的调试、维护的难度。想想,如果在一个包含 10 万行的 JSP 文件中,调试程序的情景是多么可怕。

从图 1-2 可以看出,JSP 是整个应用系统的门户,主要做 3 个重要工作。

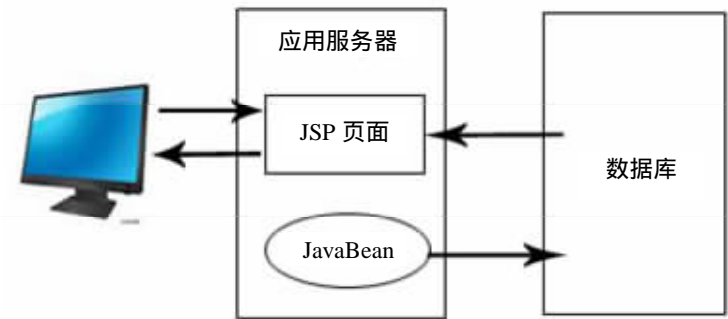


图 1-2 MVC Model 1

- 负责与客户端的所有通信。
- 处理所有的请求。
- 处理所有的答复。

在处理答复的时候，从数据库中存取数据有如下两种方式。

- JSP 自己直接去存取。
- JavaBean 来完成。

因为 JavaBean 可以被放在一个请求上下文或者用户会话中，这样就可以在不同的 JSP 页面之间通信。当然这种模型在开发需求不是很复杂、规模较小的 Web 应用的情况下有很大的优势，比如，JSP 页面可以非常容易地结合业务逻辑（`<jsp:useBean>`）、服务端处理过程（`<jsp:scriptlet>`）和 HTML（`<html>`）。在 JSP 页面中同时实现显示、业务逻辑和流程控制，从而可以快速地完成应用开发。

从工程化的角度考虑，它也有一些不足之处：把表现层和业务逻辑层的程序代码揉和在一起，不利于以后的维护工作以及开发角色的分配，所以这种模式只能适合于小的系统开发和单人开发。

### 1.4.3 MVC Model 2

由于 MVC Model 1 将表现层和业务逻辑层程序代码揉和在一起，容易产生对软件开发的维护、管理、调试不利的一面，Servlet/JSP 规范的 0.92 版描述了一个应用中使用 Servlet 和 JSP 的架构，这也就是现在所说的 MVC Model 2，该模型结构即 JSP+Servlet+JavaBean。该 MVC Model 2 又称做以 Servlet 为中心（Servlet Centric）的设计模型。

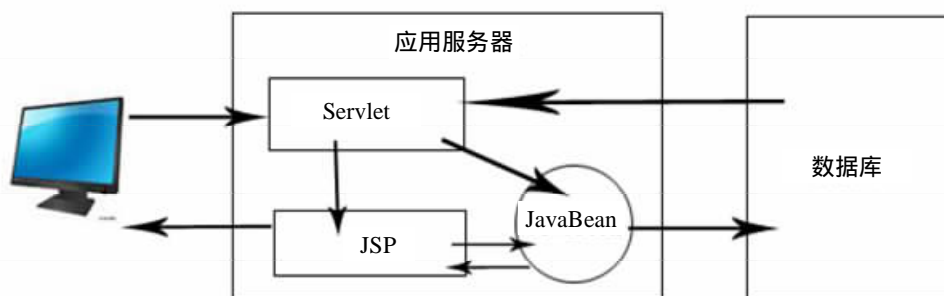


图 1-3 MVC 2 模型

对比图 1-3 与图 1-2 的结构，从 JSP 这个角度来看，JSP 页面至少少了两个任务，即获取和处理用户的请求，因为 Servlet 相当于控制器（Controller）角色，它负责接收客户端请求并处理此请求，将它传递给合适的 JSP，而 JSP 则负责显示给用户。所以 JSP 页面这时候主要做以下两件事情。

- JavaBean 直接与数据库打交道取得数据后，JSP 从 JavaBean 中读取数据。
- 把结果返回给客户端。

根据 MVC Model 2，Servlet 处理数据存取和导航流，JSP 处理表现。MVC Model 2 强制性准则使 Java 软件工程师和 HTML 用户界面设计者分别工作于他们所熟悉的部分。MVC Model 2 应用功能的一部分发生改变并不会牵连其他部分也跟着发生改变。例如，HTML 设计者可以改变软件的外观和风格，并不需要改变后端 Servlet 的工作方式。

任何一种解决方案都是双刃剑，在我们获得的同时必须有一定的付出。MVC Model 2 也带来了如下一些实际的问题。

- 必须基于 MVC 组件的方式重新思考和设计应用结构。原来通过建立一个简单的 JSP 页面就能实现的应用现在变成了多个步骤的设计和实现过程。
- 所有的页面和组件必须在 MVC 框架中实现，所以必须进行附加的开发工作。

以上两点也在 Struts 框架中存在，但并不能掩盖 Struts 优点的光辉。

#### 1.4.4 Struts 中的 MVC Model

Struts 已经提供了一个非常好的 MVC 框架，利用 Struts 开发 MVC 系统时可以大大加快开发速度。在开发时可以采用的开发流程如下。

- (1) 收集和定义应用需求。
- (2) 基于数据采集和显示的原则定义和开发用户界面的需求。
- (3) 为每一个用户界面 JSP 文件定义访问路径。
- (4) 定义 ActionMapping，建立到应用业务逻辑之间的联系。
- (5) 开发满足用户界面需求的所有支持对象。
- (6) 基于每一个用户界面需求提供的数据属性来创建对应的 ActionForm 对象。
- (7) 开发被 ActionMapping 调用的 Action 对象。
- (8) 开发应用业务逻辑对象 (Bean, EJB 等)。
- (9) 对应 ActionMapping 设计的流程创建 JSP 页面。
- (10) 建立合适的配置文件 struts-config.xml 和 web.xml。
- (11) 开发/测试/部署。

具体在使用 Struts 框架时，对应各个部分的主要开发工作如下。

- Model 部分：采用 JavaBean 和 EJB 组件，设计和实现系统的业务逻辑。根据不同的请求从 Action 派生具体 Action 处理对象，完成“做什么”的任务来调用由 Bean 构成的业务组件。由 ActionForm 的派生类实现对客户端表单数据的封装及简单的检验。
- Controller 部分：Struts 为我们提供了核心控制部分的实现。只需要配置 ActionMapping 对象即可完成 URI 地址的映射及匹配用户界面表单和 ActionForm 类的对应关系。
- View 部分：为了使用 Model 中的 ActionForm 对象，必须用 Struts 提供的自定义标记创建 HTML 表单，即利用 Struts 提供的自定义标记库编写用户界面把应用逻辑和显示逻辑分离。Struts 框架通过这些自定义标记建立了 View 和 Model 之间的联系。Struts 的自定义标记还提供了很多定制页面的功能。
- 需要编辑两个配置文件：web.xml 和 struts-config.xml。通过它们配置 Struts 系统中各个模块之间的交互。

图 1-4 描述了一个 Struts 应用程序的简要执行流程。

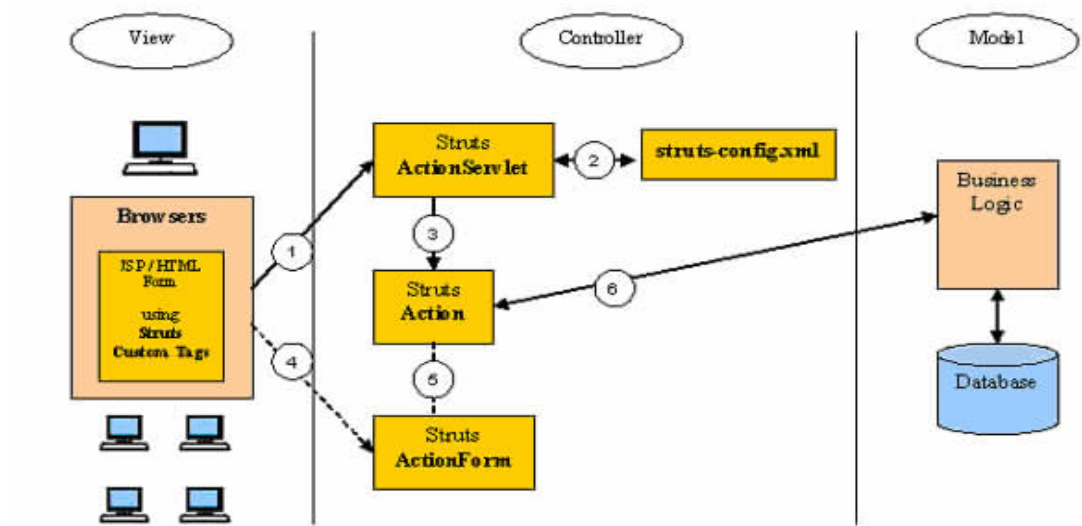


图 1-4 Struts 框架 MVC 流程图

(1) 客户端浏览器发出请求。

(2) 服务器端的 Struts 中心类 ActionServlet 找到 struts-config.xml 文件，并放入内存，将文件中的内容作为请求路径映射。

(3) ActionServlet 类在 struts-config.xml 文件中找到相关的请求路径映射后，实例化 ActionForm 类，将前台传进来的表单域打包成 Bean。

(4) 转到 Action 类进行业务逻辑功能的实现，比如增、删、改、查数据库中的数据。

(5) 通过 struts-config.xml 文件的映射，找到逻辑处理结束后显示给客户端用户看到的页面，转发功能的实现是通过 ActionForward 对象。

由图 1-4 所示的结构来看，Struts 框架可以将软件的功能进行分层，这样可以将精通不同技术的人员工作模块化，比如精通 UI 用户界面设计的人员可以只管 View 视图层，而精通 Java 程序设计的程序员可以开发 Model 层的代码。

## 1.5 配置 Eclipse 下的 Struts 开发环境

Java 开发工具对版本匹配的要求非常严格，比如在 Eclipse 上安装 MyEclipse 插件，这个 MyEclipse 插件的版本必须是针对 Eclipse 的某一个版本，否则要安装不成功，要不运行时出现莫名的错误，请读者留意。

本文使用的软件版本如下。

- eclipse3.2.1.zip
- jdk-1\_5\_0\_08-windows-i586-p.exe
- MyEclipseEnterpriseWorkbenchInstaller\_5.1.0GA\_E3.2.1.exe
- apache-tomcat-5.5.20.exe

由于 Java 环境配置的特殊性，请按如下的顺序进行配置与安装，否则某些软件由于找不到相应的环境配置文件而安装失败。

- (1) 安装 jdk-1\_5\_0\_08-windows-i586-p.exe。
- (2) 配置 JDK 环境变量。
- (3) 解压 eclipse3.2.1.zip。
- (4) 安装 MyEclipseEnterpriseWorkbenchInstaller\_5.1.0GA\_E3.2.1.exe。
- (5) 安装 apache-tomcat-5.5.20.exe。

### 1.5.1 配置 JDK 环境变量要点

这里假设 JDK 安装到计算机的 H:\JDK 目录下。

安装 JDK 结束后，用鼠标右键单击“我的电脑”，在弹出的菜单中选择“属性”，单击“高级”选项卡中的“环境变量”按钮，然后在“系统变量”中单击“新建”按钮进行如下的环境变量设置。

变量名：java\_home

变量值：H:\JDK

变量名：classpath

变量值：H:\JDK\lib\tools.jar.;H:\JDK\lib\dt.jar

变量名：path

变量值：;H:\JDK\bin

这里需要注意的是环境变量名为“path”的设置，由于系统中自带“path”环境变量及环境变量值，所以在“path”环境变量值结尾处添加“;H:\JDK\bin”，分号“;”起分隔的作用，环境变量名“path”起到在对应的目录中寻找 Java 相关命令的作用，比如 javac.exe 作用是编译 Java 源文件，Java.exe 的作用是运行 Java 程序。

“ClassPath”用来寻找 Java 类的路径，如同 DOS 里面的“set path”命令一样。另外“classpath”环境变量值中有一个英文句号“.”，代表当前的路径，即在当前目录寻找 Java 的类，如果没有找到则按“classpath”中的路径继续寻找。而“java\_home”则是 JDK 的安装目录。

注意：“系统环境变量”和“用户环境变量”之间是有区别的，“系统环境变量”针对的是所有登录到计算机上面的用户，这些用户都可使用“系统环境变量”中的内容。而“用户环境变量”则针对某个用户，比如用户 A 登录计算机设置“用户环境变量”，而用户 B 登录计算机后，用户 A 设置的“用户环境变量”对用户 B 无效。

### 1.5.2 Eclipse 的安装很“绿色”

Eclipse 是一个开放源代码的、基于 Java 的可扩展开发平台。就其本身而言,它只是一个框架和一组服务,用于通过插件组件构建开发环境。幸运的是,Eclipse 附带了一个标准的插件集,包括 Java 开发工具(Java Development Tools, JDT)。

虽然大多数用户很乐于将 Eclipse 当做 Java IDE (Integrated Development Environment, 集成开发环境)来使用,但 Eclipse 的目标不仅限于此。Eclipse 还包括插件开发环境(Plug-in Development Environment, PDE),这个组件主要针对希望扩展 Eclipse 的软件开发人员,因为它允许构建与 Eclipse 环境无缝集成的工具。由于 Eclipse 中的每样东西都是插件,对于给 Eclipse 提供插件,以及给用户提供一个一致和统一的集成开发环境而言,所有工具开发人员都具有同等的发挥场所。

这种平等和一致性并不仅限于 Java 开发工具。尽管 Eclipse 是使用 Java 语言开发的,但它的用途并不限于 Java 语言。例如,它支持诸如 C/C++、COBOL 和 Eiffel 等编程语言的插件。

Eclipse 的前身是 IBM 的 Visual Age for Java。在把这个项目免费赠送给 Eclipse 社团([www.eclipse.org](http://www.eclipse.org))前,IBM 已经投入超过 4000 万美元进行研发。Eclipse 社团的创始人还包括 Borland、Merant、QNX Software Systems、Rational Software、Red Hat、SuSE、TogetherSoft 和 Webgain 等国际知名 IT 信息公司,后来加入的还有 Oracle 等公司,实力相当雄厚。如今,IBM 通过附属的研发机构 Object Technologies International(简称 OTI),继续领导着 Eclipse 的开发。

由于 Eclipse 运行在 Windows 上的版本是一个 ZIP 文件,所以直接解压就可以使用了,还是比较“绿色”的。

### 1.5.3 MyEclipse 的安装

MyEclipse 是一款功能强大的 J2EE 集成开发环境,支持代码编写、配置、测试以及调试。程序功能包括:HTML 智能编辑器,Struts,JSF,CSS,JavaScript,SQL,Hibernate,带有自动完成与语法高亮显示功能的 J2EE 编辑器。在 Eclipse 中安装这个插件可以大大加快开发 J2EE 的效率。它是商业的,使用需要付费。

本书使用的 MyEclipse 版本如下。

MyEclipseEnterpriseWorkbenchInstaller\_5.1.0GA\_E3.2.1.exe

双击安装文件后,出现欢迎界面,如图 1-5 所示。

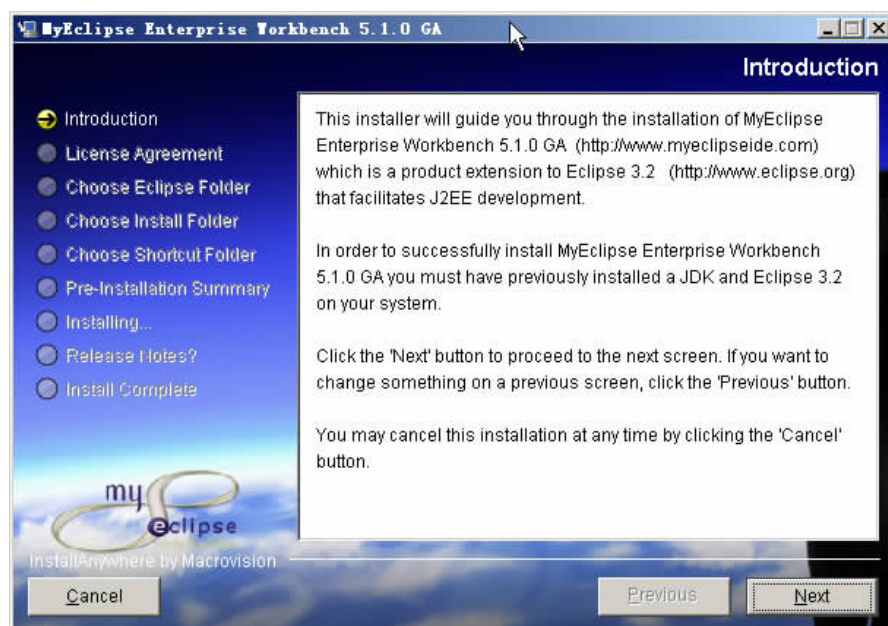


图 1-5 MyEclipse 欢迎界面

单击“Next”按钮继续安装，出现如图 1-6 所示的软件许可界面。

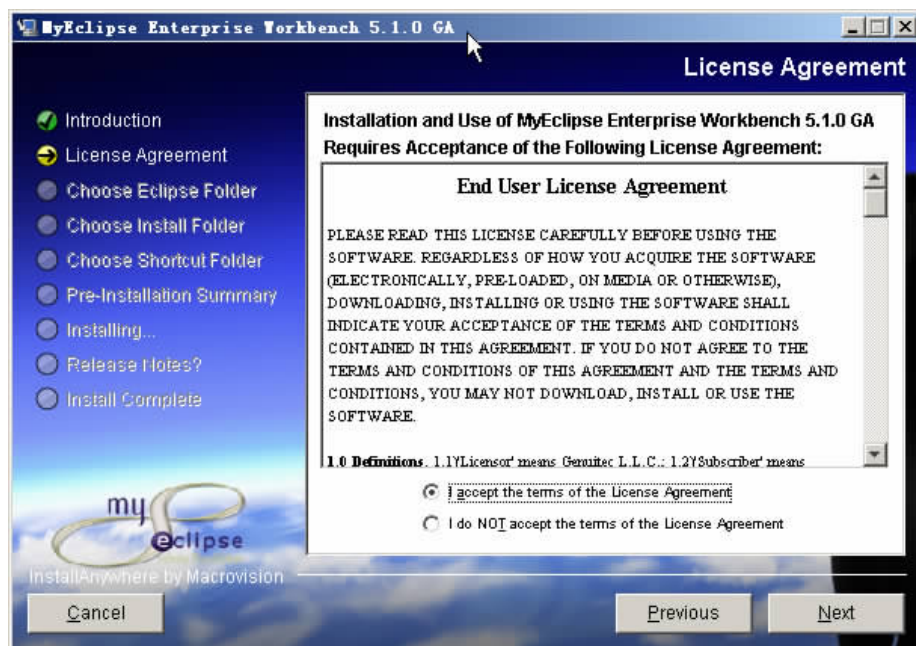


图 1-6 MyEclipse 软件许可界面

出现软件使用许可窗口后，选择“I accept the...”单选按钮，并单击“Next”按钮继续安装，出现设置 Eclipse 的安装目录的界面，如图 1-7 所示。



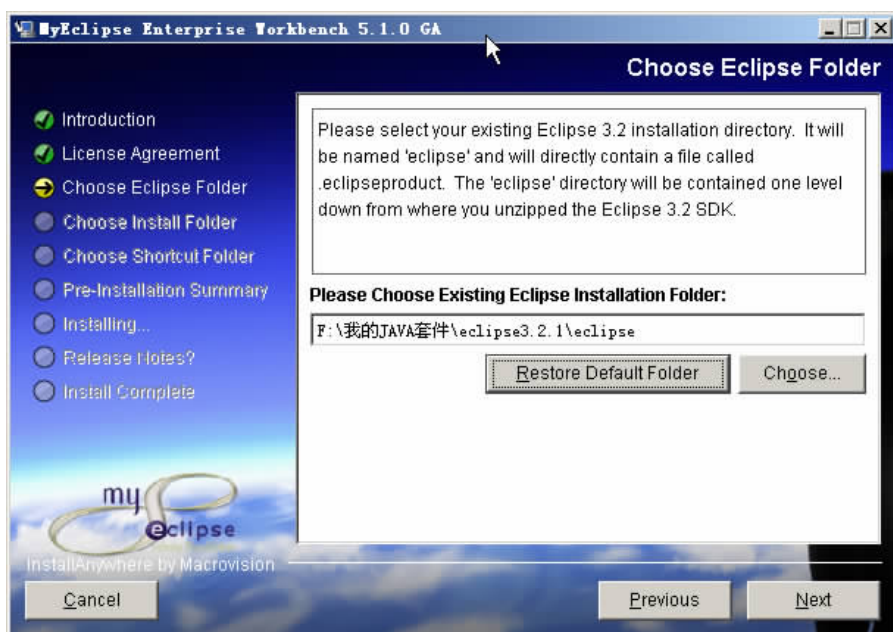


图 1-7 设置 Eclipse 安装目录

选择 Eclipse 的安装路径后，单击“Next”按钮后出现新的窗口，如图 1-8 所示，这回设置的是 MyEclipse 的安装目录。

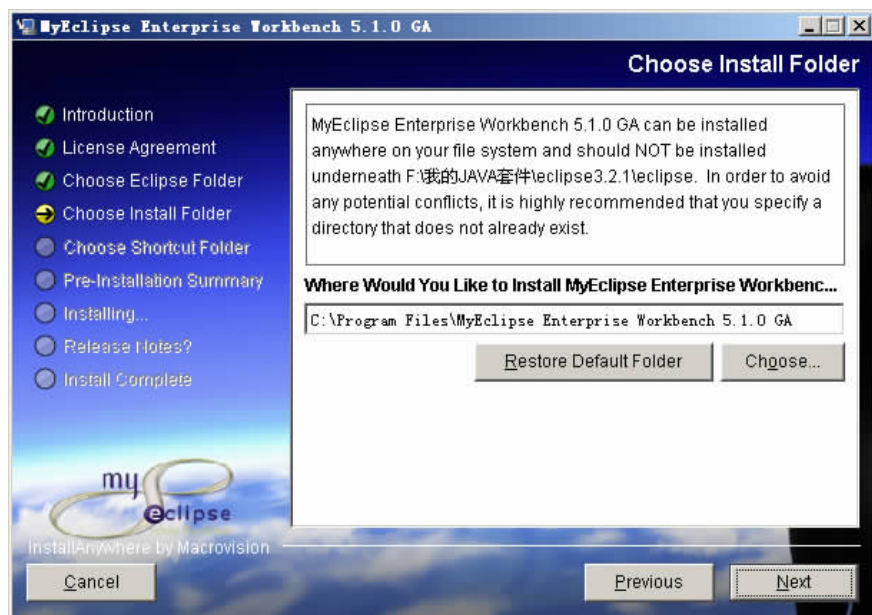


图 1-8 设置 MyEclipse 安装目录

设置完 MyEclipse 的安装路径后，单击“Next”按钮，出现如图 1-9 所示的界面设置，MyEclipse 创建图标选项，笔者选择不创建，单击“Next”按钮继续安装。

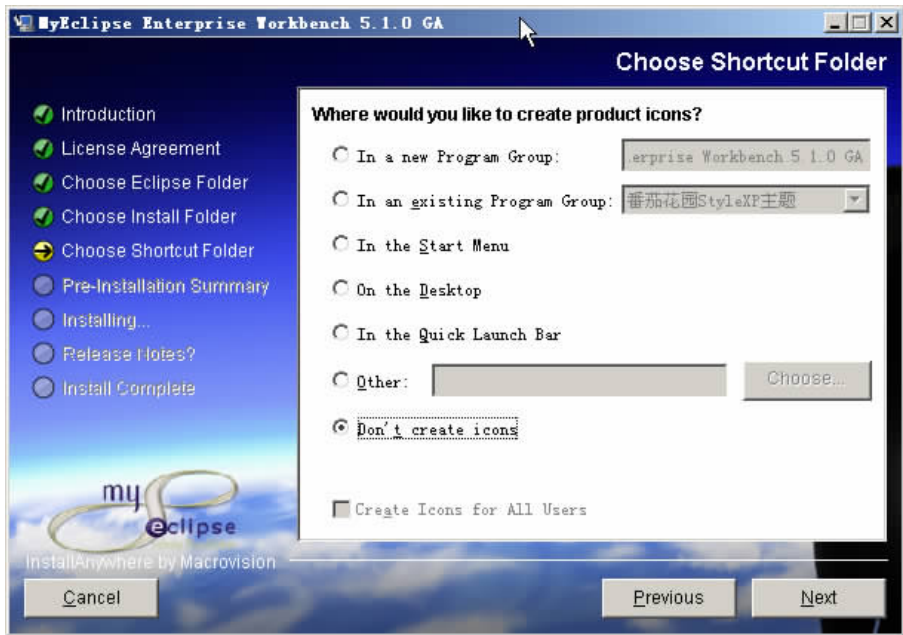


图 1-9 设置 MyEclipse 创建图标选项

这是最后一步了，里面显示欲安装软件的一些信息，如图 1-10 所示，单击“Install”按钮进行软件的安装。

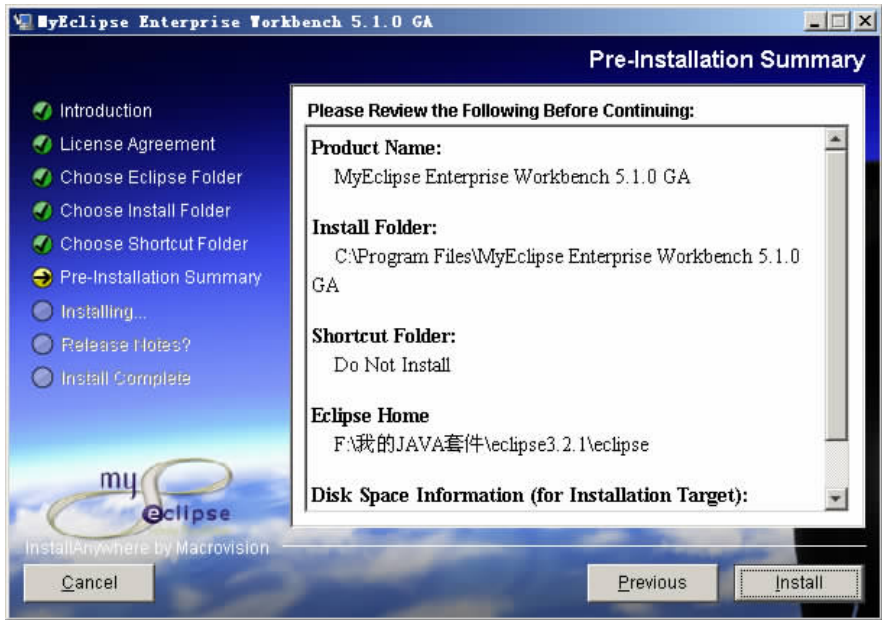


图 1-10 显示 MyEclipse 安装信息

等待安装结束吧！☺。

### 1.5.4 Tomcat 的安装

Tomcat 是一个非常好的 JSP 容器，支持 JSP 的技术规范，而且占用资源少，使用方便，在本地开发 JSP 程序时推荐使用。

本书使用的 Tomcat 版本如下。

apache-tomcat-5.5.20.exe

双击安装文件，显示欢迎界面，如图 1-11 所示。



图 1-11 Tomcat 的欢迎界面

单击“Next”按钮继续安装，出现如图 1-12 所示的窗口。

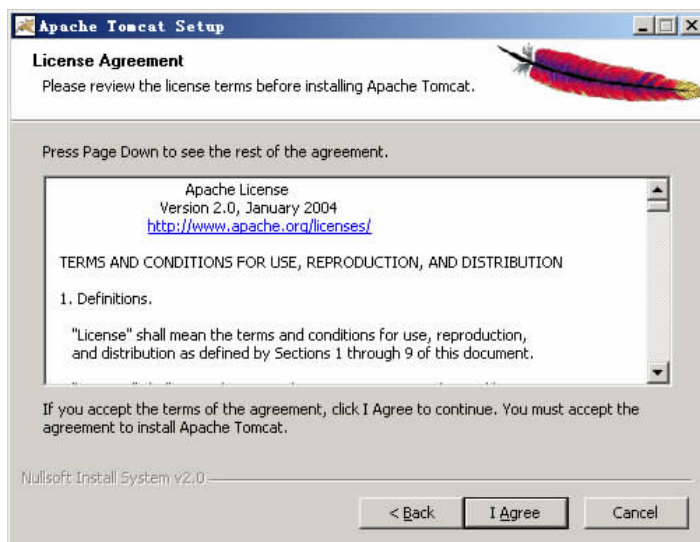


图 1-12 Tomcat 软件许可

在显示软件安装许可窗口中，单击“ I Agree ”按钮继续安装，出现如图 1-13 所示的窗口，这里保持默认的安装，不需要更改。

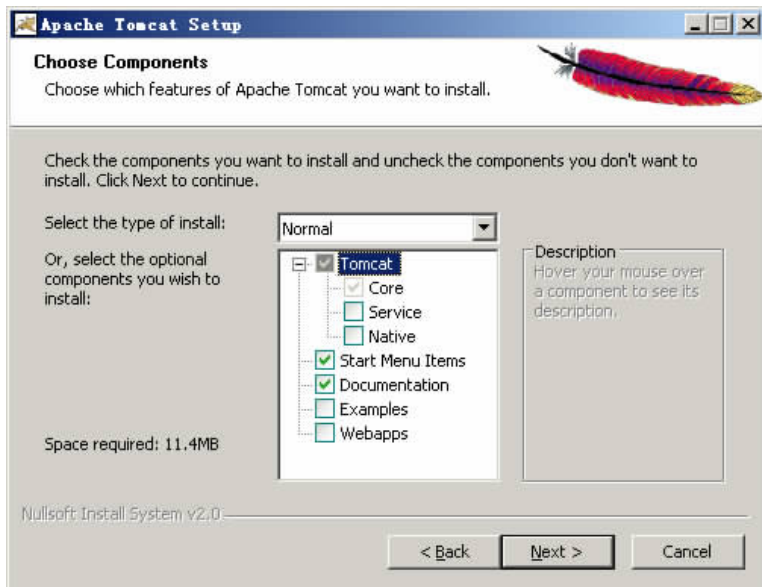


图 1-13 Tomcat 安装组件选项

单击“ Next ”按钮继续安装，出现如图 1-14 所示的窗口，可以设置 Tomcat 的安装目录。

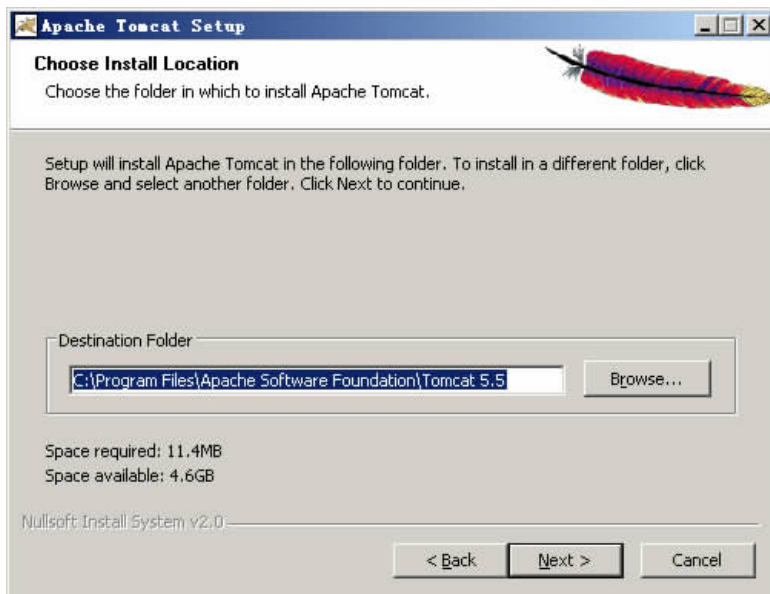


图 1-14 Tomcat 安装路径

设置了安装目录，单击“ Next ”继续安装，出现如图 1-15 所示的窗口，这里保持默认值，不需要更改。

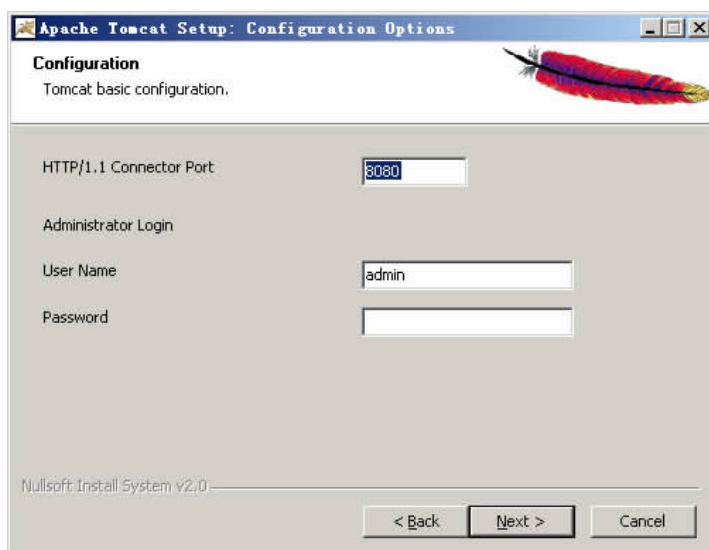


图 1-15 Tomcat 端口及用户登录设置

单击“Next”按钮继续安装，出现如图 1-16 所示的窗口，这里一定要选择 JDK 安装目录中的 JRE 目录。

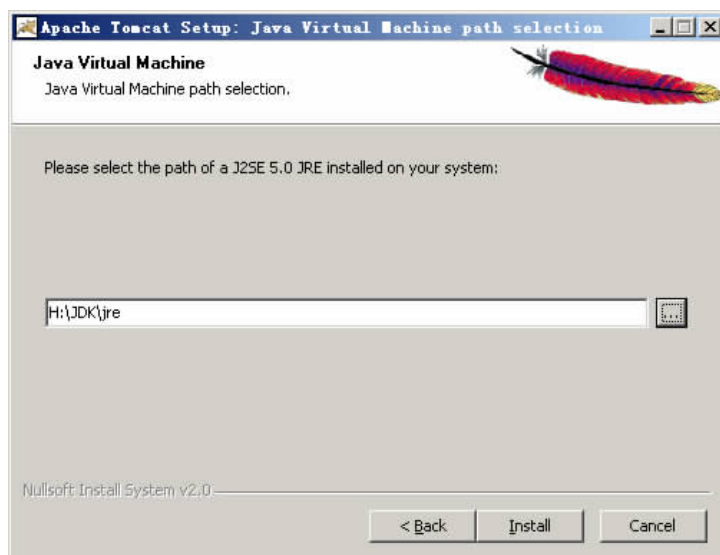


图 1-16 设置 Tomcat 的 JRE 目录

选择了 JDK 目录中的 JRE 目录后，单击“Install”按钮进行软件安装。

### 1.5.5 测试开发环境

双击 Eclipse.exe 文件进入 IDE 开发环境，在这里我们要测试一下 JSP 文件是否正常运行。

启动 Eclipse.exe 后出现如图 1-17 所示的窗口。

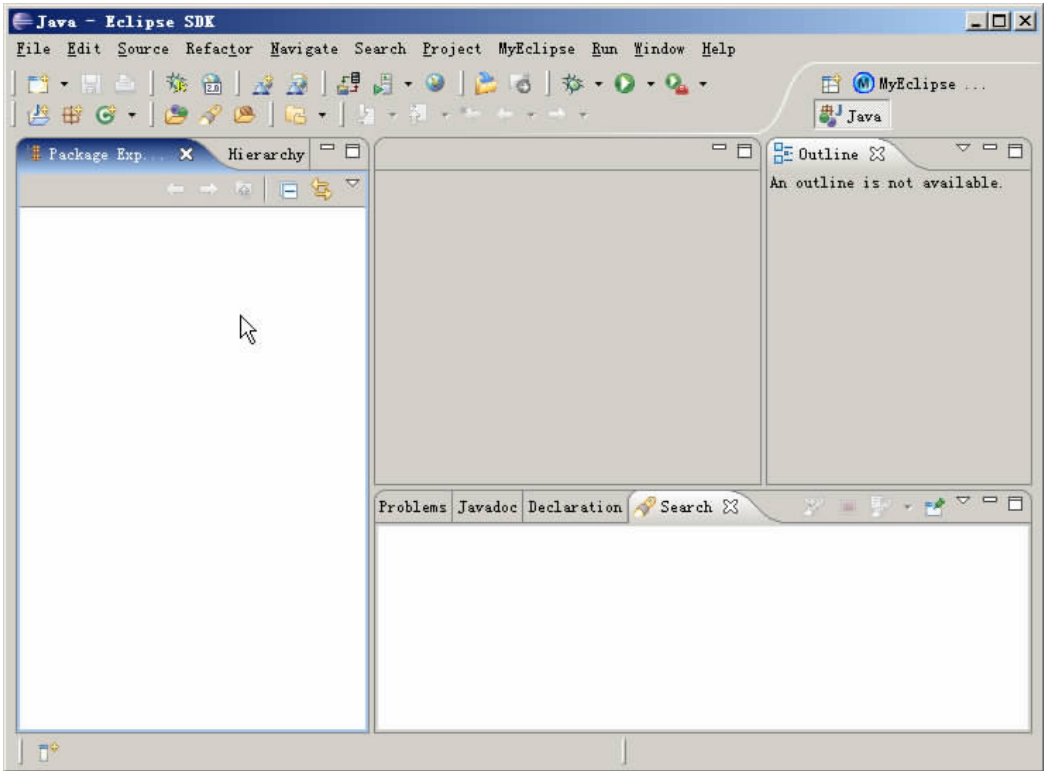


图 1-17 Eclipse 窗口

选择 “ File ” “ new ” “ Project ” 新建一个工程项目，出现如图 1-18 所示的窗口。

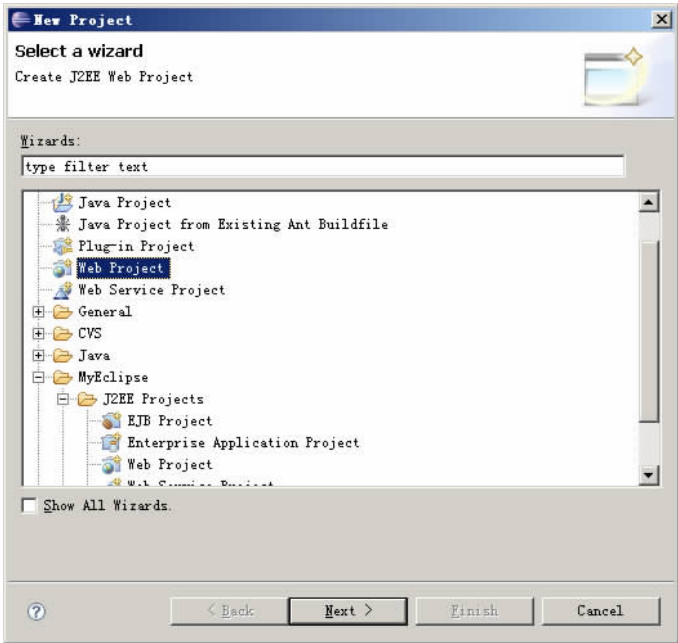


图 1-18 Eclipse 新建项目窗口



在这里我们要测试 JSP 页面，所以选择“Web Project”项目，单击“Next”按钮后出现如图 1-19 所示的窗口。

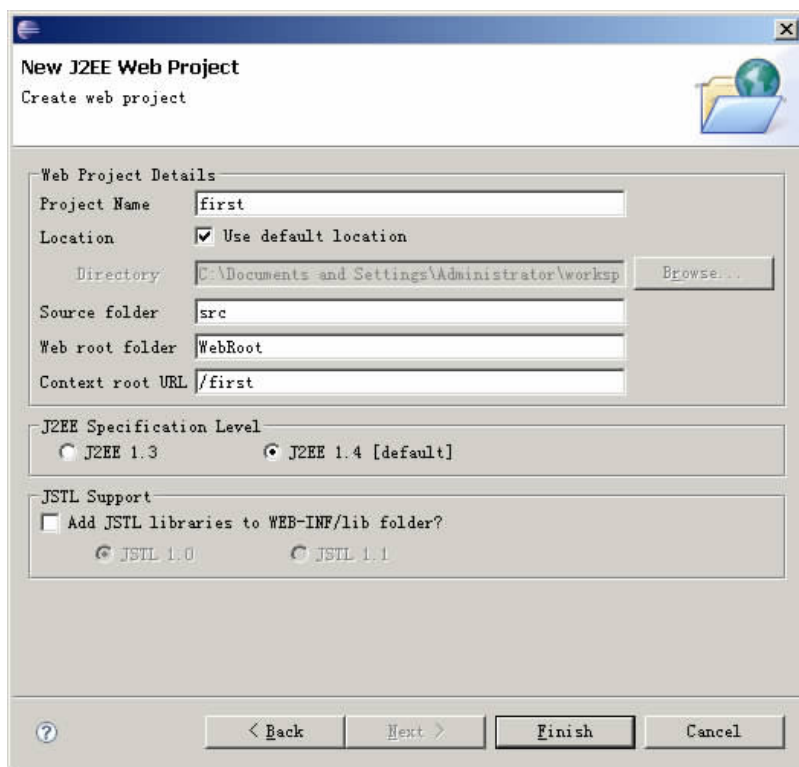


图 1-19 Eclipse 项目信息窗口

在“Project Name”文本框中输入项目名称“first”后单击“Finish”按钮，完成新建项目。

这时在左边的“Package Explorer”页中出现 first 项目，如图 1-20 所示。

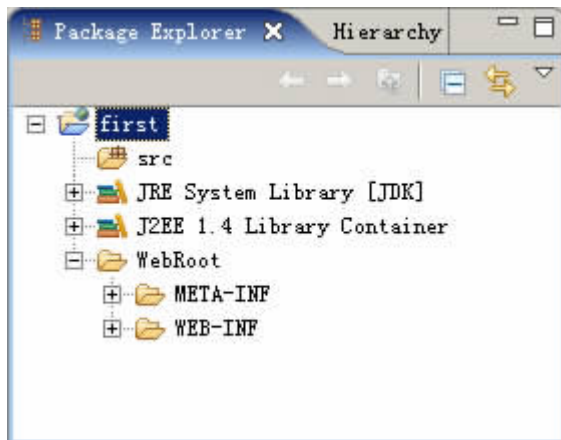


图 1-20 first 项目

项目已经创建完成，这时我们新建一个 JSP 页面，用鼠标右键单击“WebRoot”项，然后在弹出的菜单中选择“New”“JSP”命令来新建一个 JSP 网页文件，如图 1-21 所示。

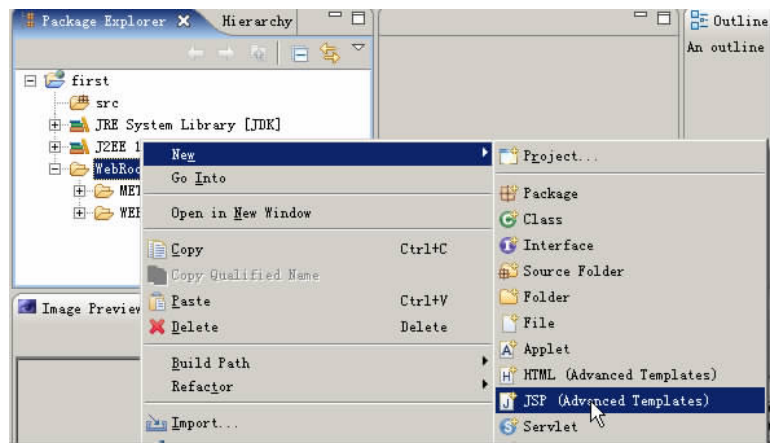


图 1-21 新建一个 JSP 页面

出现如图 1-22 所示对话框，输入 JSP 的文件名称“one.jsp”，其他保持默认设置。

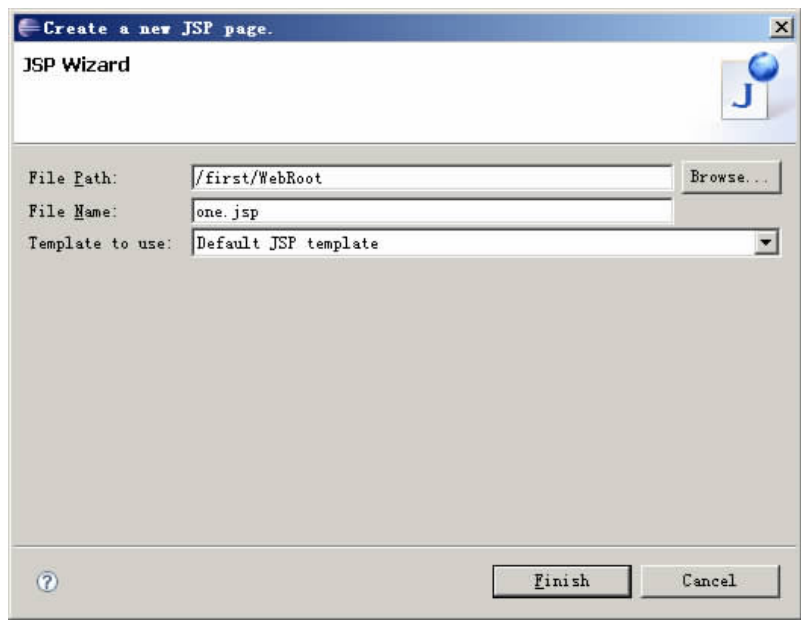


图 1-22 设置 JSP 文件信息

单击“Finish”按钮完成创建 JSP 页面。这时我们发现在“Package Explorer”页中的“WebRoot”节点下方出现“one.jsp”文件，证明创建文件成功。

此时可以发现软件的标题内容改变，正在使用的软件不是 Eclipse，而是 MyEclipse，MyEclipse 是 Eclipse 的一个插件，如图 1-23 所示。

如图 1-23 所示，在代码编辑区自动打开了新建的“one.jsp”文件。



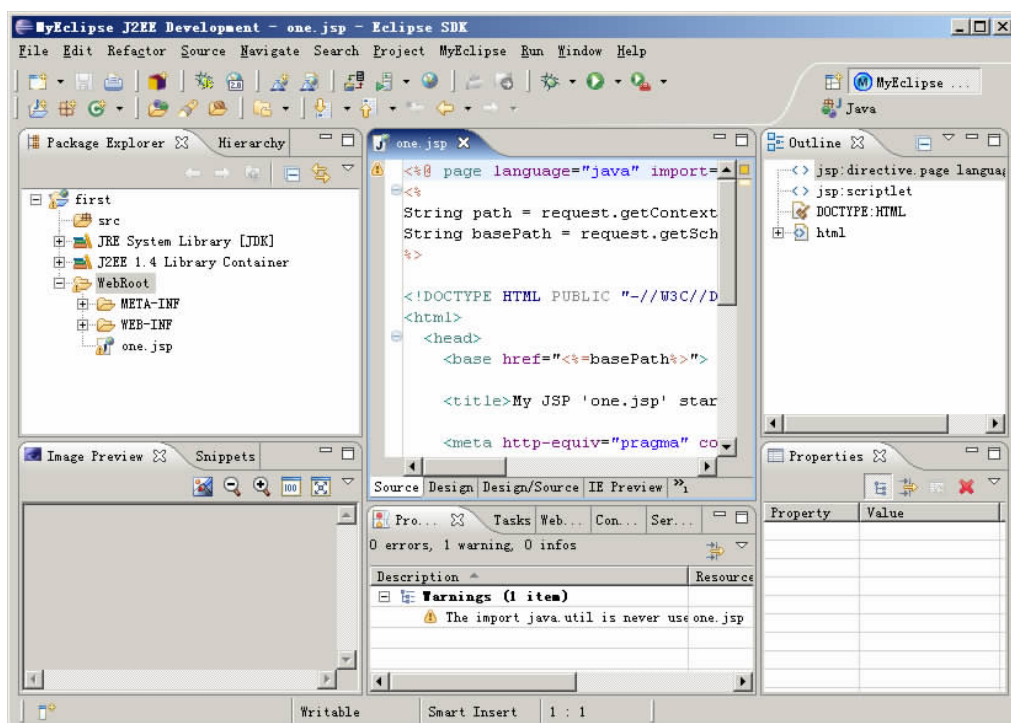


图 1-23 正在使用 MyEclipse

### MyEclipse 与 Tomcat 集成

到这一步，我们已经成功安装了 MyEclipse 和 Tomcat，下一步就需要将 Tomcat 与 MyEclipse 进行集成，单击“Window”“Preferences”菜单后弹出窗口如图 1-24 所示。

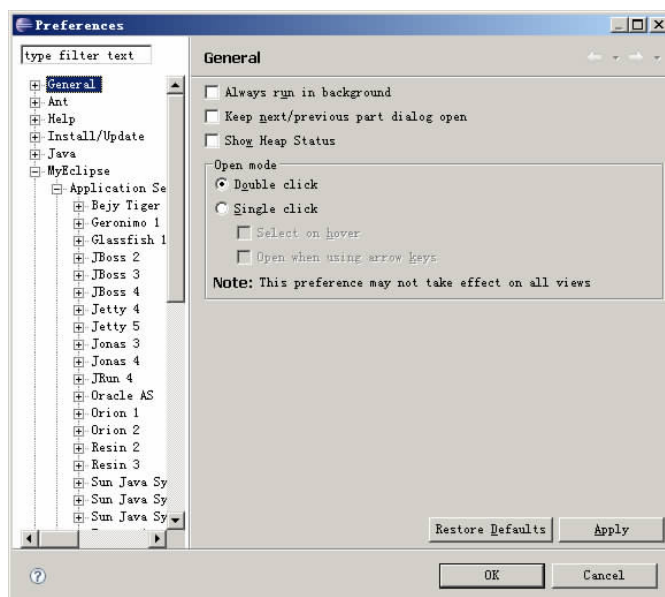


图 1-24 Eclipse 属性设置

选择 “MyEclipse” 节点中的 “Tomcat5” 节点，然后按照如图 1-25 所示进行设置。

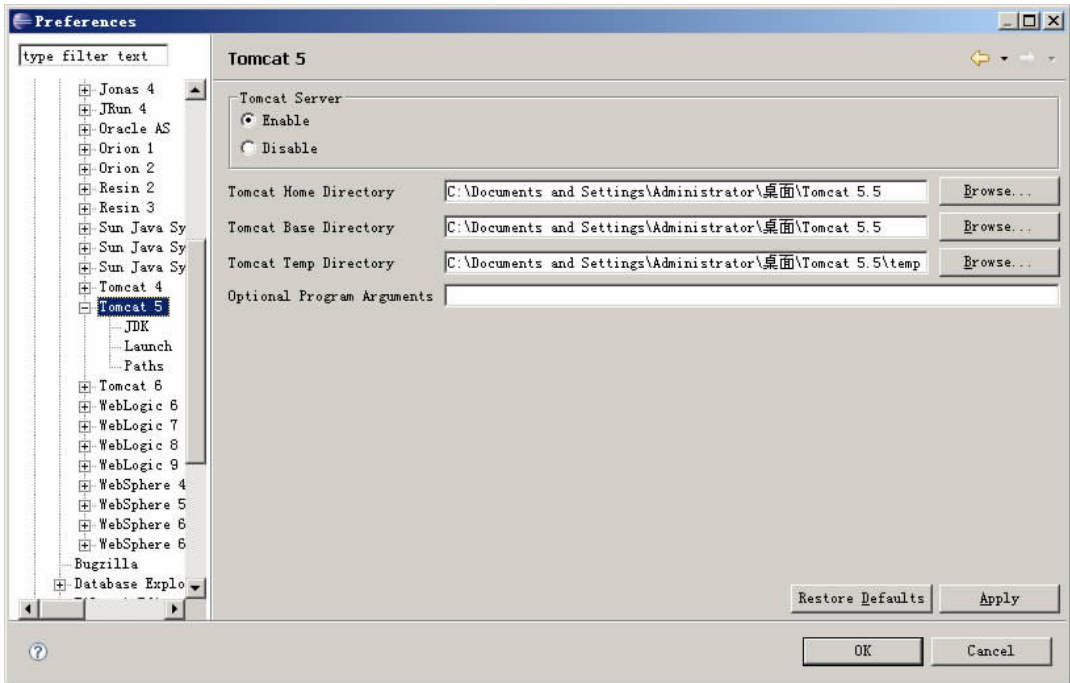




图 1-25 Tomcat 属性设置

在如图 1-25 所示窗口中设置 “Tomcat Server” 的状态为 “Enable”，设置 “Tomcat Home Directory” 的内容，即 Tomcat 的安装目录，设置完成后单击 “OK” 按钮。

在 “one.jsp” 文件的 <body> 和 </body> 之间加入 JSP 脚本：

```
<body>
<%
out.println("this is my first jsp page!~");
%>
</body>
```

保存更改的文件。

下一步我们就要部署这个 JSP 项目了，如果 MyEclipse 安装正确，则在工具栏里会出现按钮组 ，单击  按钮进行项目的部署，出现如图 1-26 所示的对话框。

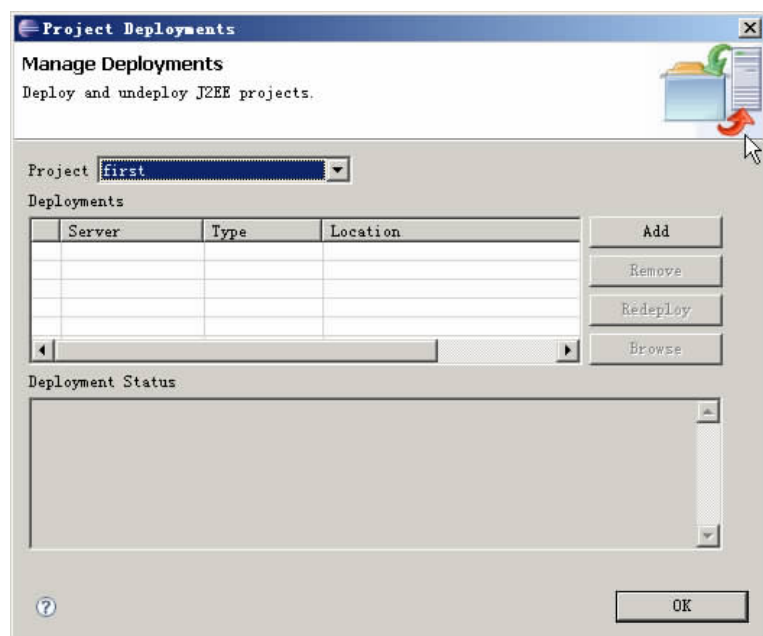


图 1-26 项目部署

单击“Add”按钮，添加 JSP 的容器，弹出如图 1-27 所示的对话框。

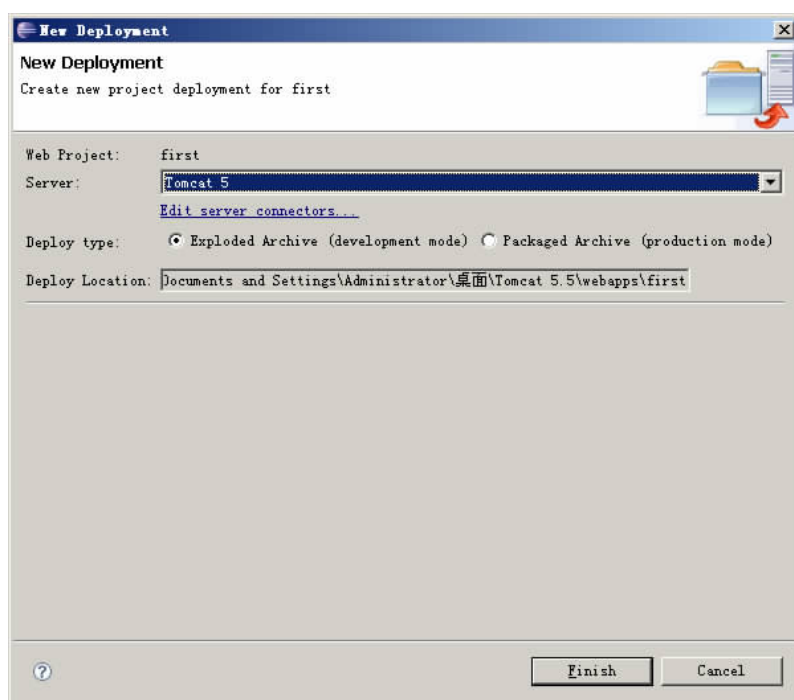


图 1-27 选择 JSP 容器

在这里选择刚才设置的 Tomcat 5 的 Web 容器，单击“Finish”按钮后，界面如图 1-28 所示。

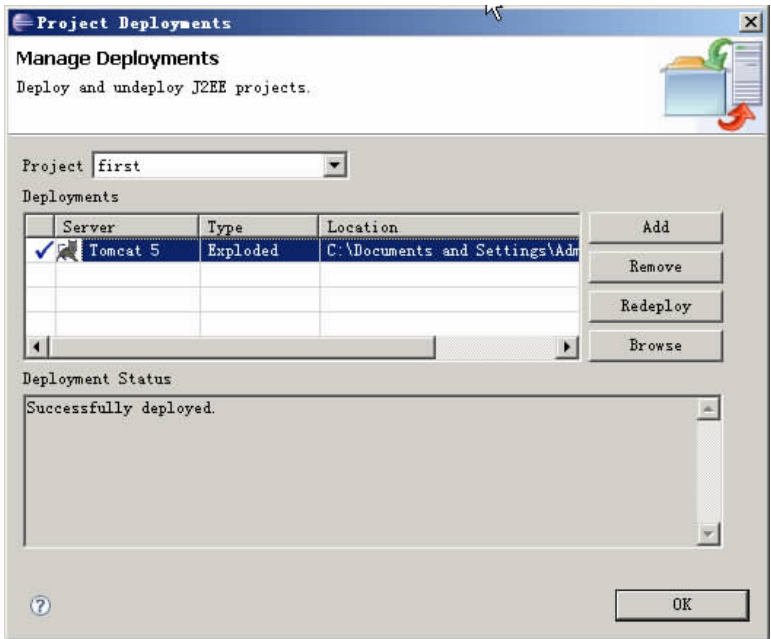


图 1-28 设置完成 JSP 容器

在如图 1-28 所示界面中可以对这个部署进行更改、删除和重新部署功能，单击“OK”按钮后，部署完成。

单击 按钮组中的 按钮来启动 Tomcat 容器。打开一个 IE 窗口，输入地址：

http://localhost:8080/first/one.jsp

出现如图 1-29 所示的运行效果，证明环境配置、项目部署、启动服务成功。

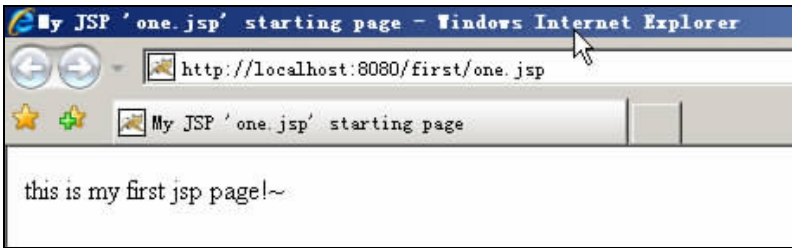


图 1-29 部署成功

至此，关于软件的安装、开发环境的配置、新建 JSP 文件、Tomcat 与 MyEclipse 的集成、项目的部署、启动 Tomcat 的介绍已经结束，下一步让我们了解如何开发一个简单的 Struts 实例吧。

# 第 2 章

## Struts 简介

### 2.1 Struts 的工作流程

Struts 的运行机制有自己的特点，但万变不离其宗，MVC Model 2 的设计概念也正是 Struts 的根本。

当在浏览器中输入一个 URL 地址后，Struts 运行过程大体如下。

(1) 服务器端接受客户端用户的 URL 请求并响应，此时在浏览器中显示页面。比如，这个用户的 URL 请求是 login.jsp，显示登录页面。

(2) 在 login.jsp 中输入“用户名”和“密码”后，单击“登录”按钮提交表单。“用户名”和“密码”的 HTML 文本域的 name(名字属性)分别是“username”和“password”。

(3) 一个表单提交请求产生，由服务器端的 ActionServlet 类进行接收。ActionServlet 类会查找在服务器启动时就加载到内存的 Struts-Config.xml 配置文件(这个文件中的内容是由程序员来维护的)，根据文件中的内容来进行路径与功能的映射查询。这个 ActionServlet 类相当于十字路口的路标，负责指引方向。

(4) ActionServlet 类将接收到的客户端请求的表单打包成一个 ActionForm 类，这个类就是使用 set 或 get 方法的 Java Bean，存取的当然就是“用户名”和“密码”了。在这个 ActionForm 类中有两个方法：validate 和 reset。方法 validate 的功能是简单地检验传进来的“用户名”和“密码”的正确性，比如是否为空。方法 reset 是初始化 get 或 set 的变量。

关于 Struts 的例程如下。

```
public class LoginForm extends ActionForm {
    private String password;

    private String username;

    public ActionErrors validate(ActionMapping mapping,
        HttpServletRequest request) {
//一些校验语句
        return null;
    }

    public void reset(ActionMapping mapping, HttpServletRequest request) {
        password="";
    }
}
```

```
        username="";    }

    public String getPassword() {
        return password;
    }

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

    public String getUsername() {
        return username;
    }

    public void setUsername(String username) {
        this.username = username;
    }
}
```

(5) 在服务器内存的 Struts-Config.xml 文件中去寻找由哪个“功能模块”来进行用户名和密码的详细验证,比如与数据库中的内容进行对比。这里所指的“功能模块”就是 Struts 中的 Action 类。

(6) 在 Action 类中,

```
public ActionForward execute(
    ActionMapping mapping,
    ActionForm form,
    HttpServletRequest request, HttpServletResponse response)
```

方法从 ActionForm 中取得“用户名”和“密码”来进行一些检验。检验处理结束后需要返回一个 ActionForward 对象,这个 ActionForward 对象的功能相当于 JSP 中<jsp:forward>的转发功能。

(7) 如果校验成功,转到 true.jsp 登录成功页面;如果校验不成功,转到 false.jsp 页面。

如图 2-1 所示为上面过程的工作流程。

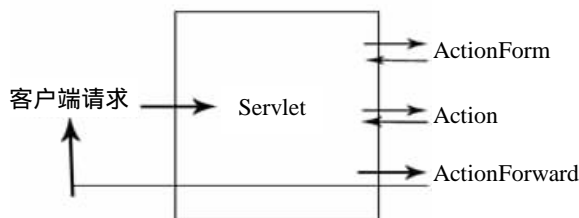


图 2-1 Struts 的工作流程

执行过程大体如下：

客户请求 进入 ActionServlet 中心控制类查找 Struts-Config.xml 路径关系映射 封装成 ActionForm

执行业务功能逻辑 Action ActionForward 转发返回到客户端

不难看出，Struts 运行的中心就是 Struts-Config.xml 文件，它是心脏。

上面就是一个最典型的 Struts 应用程序的执行步骤。

到此一个 Struts 请求和响应的流程结束。也许读者不明白上面的一些代码的含义，没有关系，下面介绍一个简单的实例，步入 Struts 的世界吧！

## 2.2 关于 Struts 的实例

本节通过一个 Struts 实例来巩固前面所学的知识。

### 2.2.1 添加 Struts 框架支持文件

(1) 建立一个 Web Project 项目工程。

新建一个 Project 项目，操作步骤如图 2-2 所示。

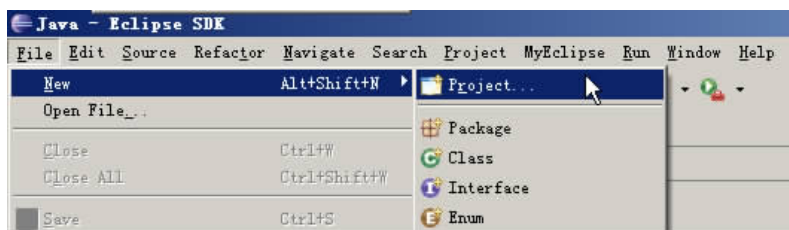


图 2-2 新建一个项目

弹出如图 2-3 所示的窗口界面。

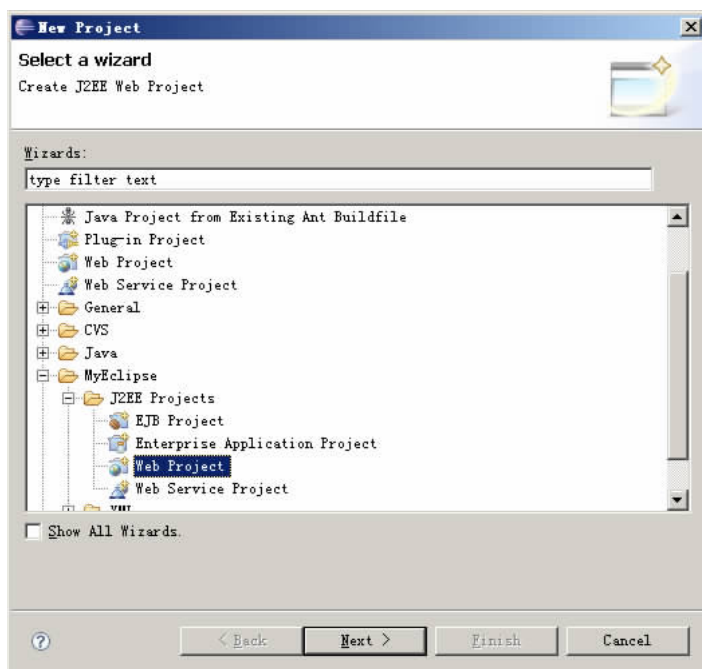


图 2-3 新建项目界面

双击 “Web Project” 选项后，弹出项目配置窗口，如图 2-4 所示。

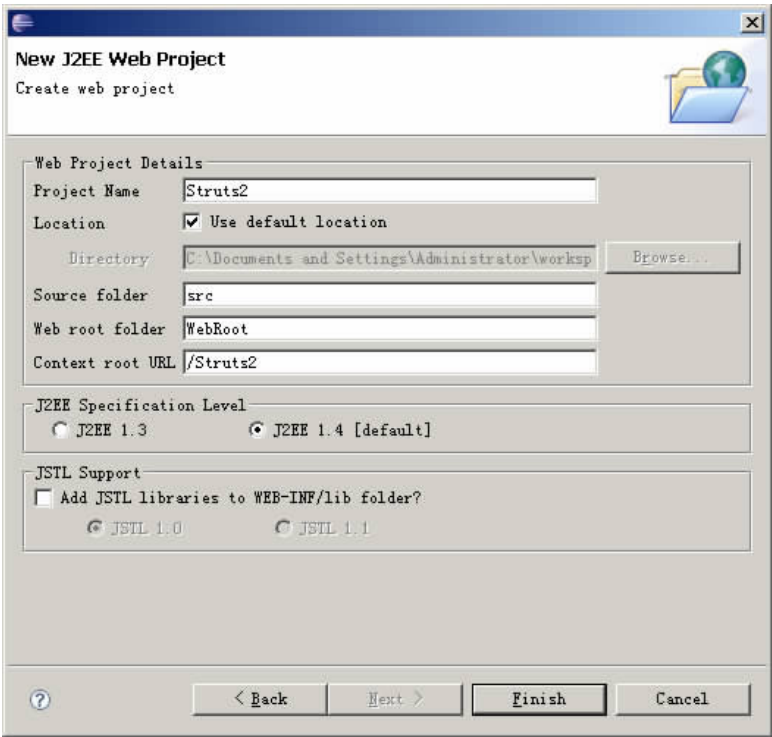


图 2-4 项目配置窗口

在 Project Name 项目名称中写入 “Struts2”，单击 “Finish” 按钮后如图 2-5 所示，显示出 Struts2 的工程结构。

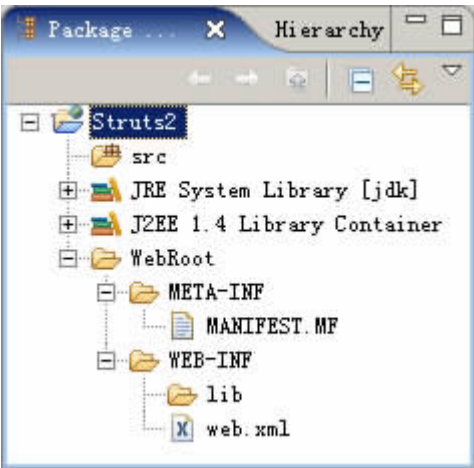


图 2-5 Struts2 工程结构

(2) 在 “Struts2” 上单击鼠标右键，添加 Struts 框架必需的支持文件，如图 2-6 所示。



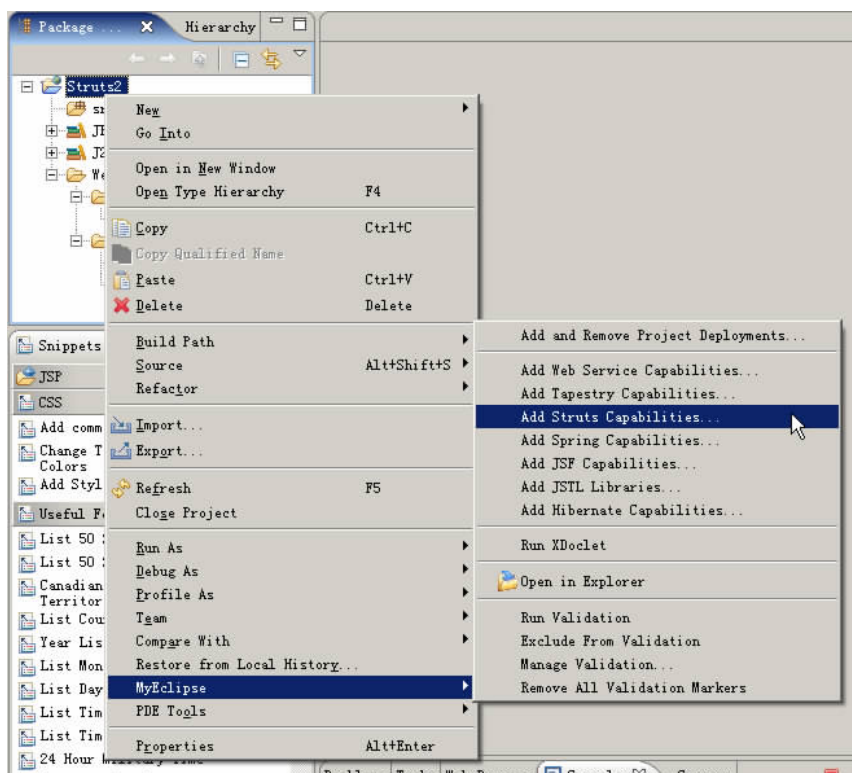


图 2-6 添加 Struts 框架必需的文件

(3) 出现如图 2-7 所示的界面。

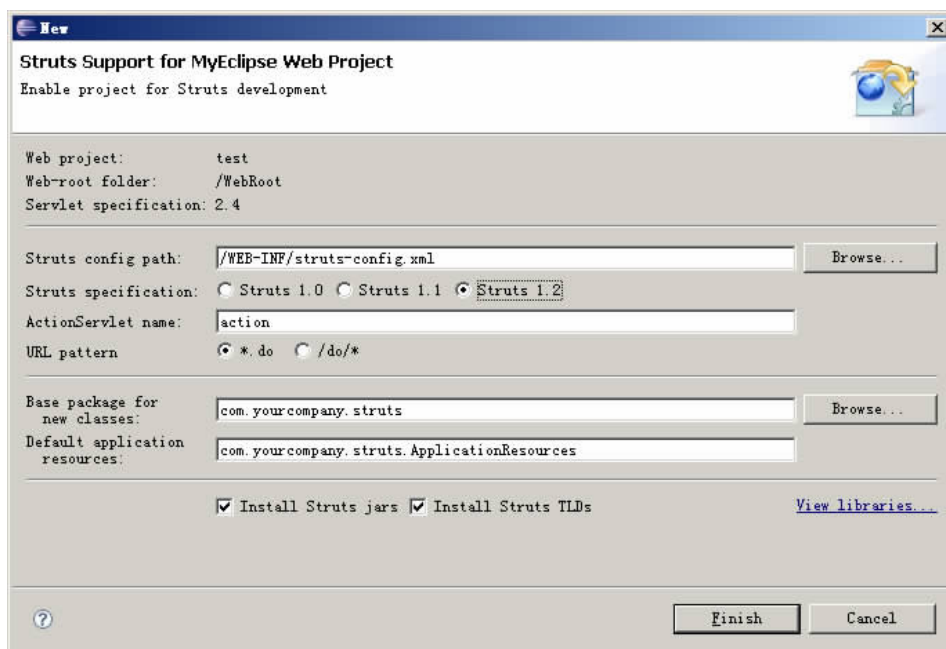


图 2-7 Struts 框架设置

- Struts config path：设置 struts-config.xml 文件的存放路径。
- Struts specification：设置使用 Struts 的版本，这里使用 Struts 1.2 版本。
- ActionServlet name：设置 ActionServlet 的名字，如下面所示的代码。

```
<servlet-name>action</servlet-name>  
<servlet-class>org.apache.struts.action.ActionServlet</servlet-class>
```

- URL pattern：设置 URL 的模式，访问的映射地址形式。
- Base package for new classes：给当前 Struts 项目自动新建的一个包设置包名。
- Default application resources：设置默认使用的资源文件名称。

(4) 这里保持默认设置，不需要改动任何选项，单击“Finish”按钮，应用设置。

(5) 经过了几秒的软件自动操作后，出现如图 2-8 所示的界面。

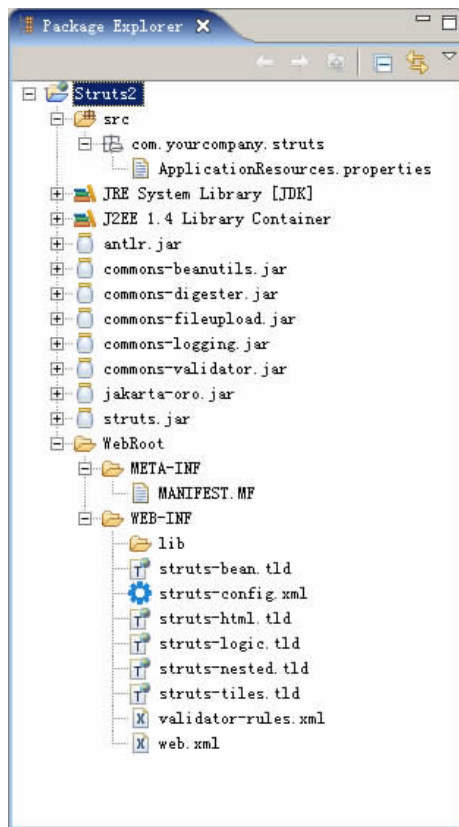


图 2-8 支持 Struts 框架的项目结构

到此，我们已经将当前的项目设置成为支持 Struts 的项目了。

### 2.2.2 视图层 V-View 的添加

在此要做一个登录的界面，里面包含“用户名”和“密码”的表单。

在 Struts2 项目中的“WebRoot”上单击鼠标右键，在弹出的菜单中选择“New”“Other”命令，如图 2-9 所示。

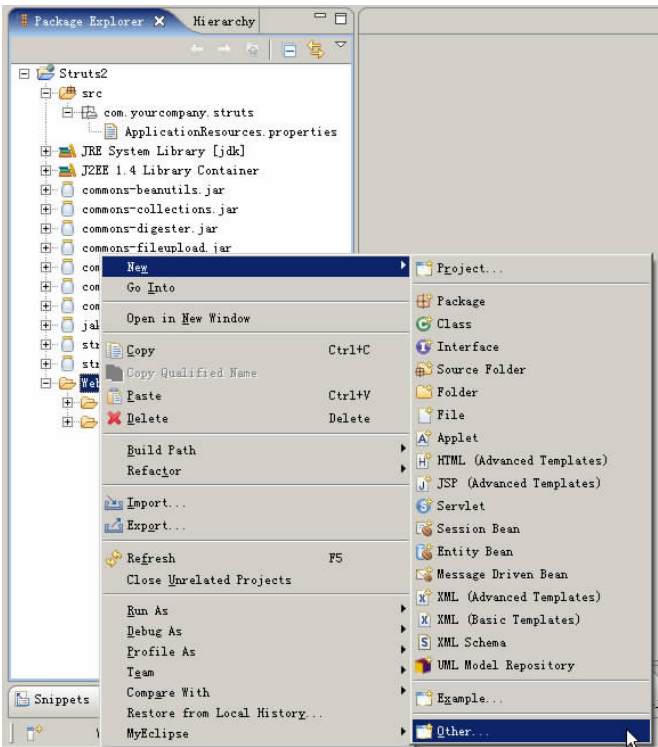


图 2-9 将要新建一个 ActionForm

依次展开 “MyEclipse” “Web-Struts” “Struts 1.2” “Struts1.2 Form” 节点，如图 2-10 所示。

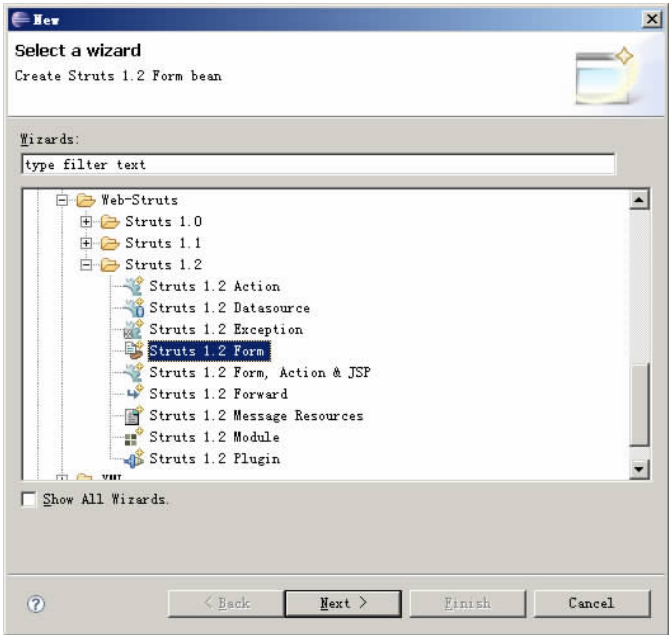


图 2-10 新建一个 ActionForm

单击“Struts 1.2 Form”后，出现如图 2-11 所示的设置窗口。

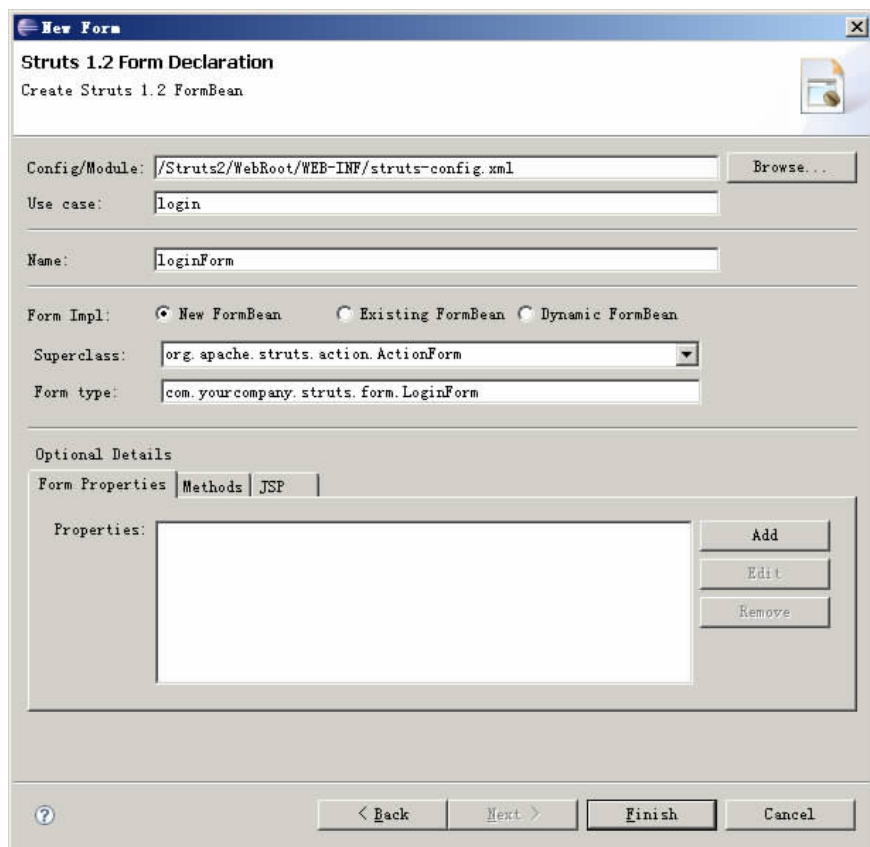


图 2-11 ActionForm 设置

在这个窗口中我们需要设置如下 4 项内容。

- (1) 设置“Use case”为“login”。
- (2) 设置“Superclass”为“org.apache.struts.action.ActionForm”。
- (3) 添加“用户名”和“密码”表单项。

单击“Form Properties”页中的“Add”按钮，出现如图 2-12 所示的界面。



图 2-12 创建用户名属性

设置 “Name” 为 “username”，也就是输入用户名表单文本域的名字，设置 “Type” 为 “java.lang.String”，设置 “JSP input type” 属于表单中的 “text” 类型，相当于 HTML 中的：

```
<input name="username" type="text" id="username">
```

单击 “Add” 按钮后，继续设置 password 表单的属性，如图 2-13 所示。



图 2-13 创建密码属性

单击 “Add” 按钮，添加密码属性，到这一步，用户名和密码已经设置完毕，单击 “Close” 按钮关闭当前窗口。

“Form Properties” 页中的内容如图 2-14 所示。

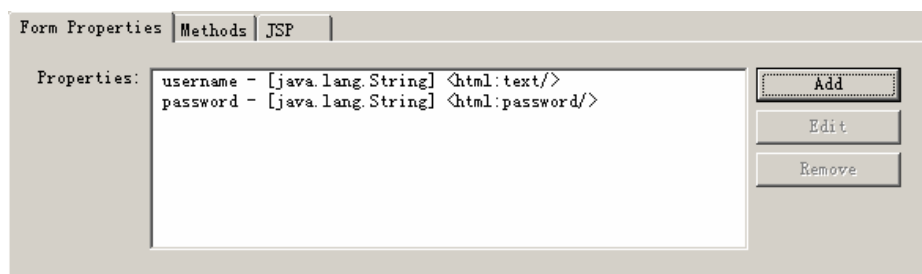


图 2-14 设置完成的用户名和密码属性

(4) 设置 JSP 页，如图 2-15 所示。



图 2-15 创建一个 JSP 文件

到此，已经完成了 ActionForm 和 JSP 页面的创建设置，单击“Finish”按钮结束设置。发现如图 2-16 所示的窗口中已经添加了一个 LoginForm.java 文件。

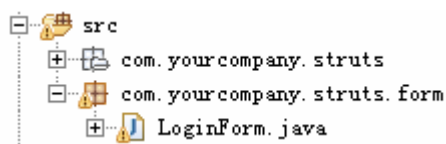


图 2-16 添加的 LoginForm.java 文件

LoginForm.java 文件的程序代码如下。

```
package com.yourcompany.struts.form;

import javax.servlet.http.HttpServletRequest;
import org.apache.struts.action.ActionErrors;
import org.apache.struts.action.ActionForm;
import org.apache.struts.action.ActionMapping;

public class LoginForm extends ActionForm {

    /** password property */
    private String password;

    /** username property */
    private String username;

    public ActionErrors validate(ActionMapping mapping,
        HttpServletRequest request) {
        // TODO Auto-generated method stub
        return null;
    }

    public void reset(ActionMapping mapping, HttpServletRequest request) {
        // TODO Auto-generated method stub
    }

    public String getPassword() {
        return password;
    }

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

    public String getUsername() {
        return username;
    }

    public void setUsername(String username) {
        this.username = username;
    }
}
```

从上面的程序可以看出，Struts 将前端页面表单中的内容封装成 Java Bean，每一个这样的 Bean 都是 ActionForm 的子类。

在 WebRoot 节点下自动新建了一个目录 form 和一个文件 login.jsp，如图 2-17 所示。

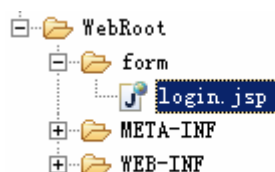


图 2-17 WebRoot 目录结构

login.jsp 文件的程序代码如下。

```
<%@ page language="java" pageEncoding="ISO-8859-1"%>
<%@ taglib uri=http://jakarta.apache.org/struts/tags-bean
prefix="bean"%>
<%@ taglib uri=http://jakarta.apache.org/struts/tags-html
prefix="html"%>

<html>
  <head>
    <title>JSP for LoginForm form</title>
  </head>
  <body>
    <html:form action="/[ACTION_PATH]">
      password : <html:password property="password"/><html:errors
        property="password"/><br/>
      username : <html:text property="username"/><html:errors
        property="username"/><br/>
      <html:submit/><html:cancel/>
    </html:form>
  </body>
</html>
```

在 Struts 的 MVC 模型中的 V-View 视图包含 jsp 文件和 ActionForm 类。

自动生成的<html:form>表单的 action 目标资源却是默认的"/[ACTION\_PATH]"，action 可以在以后的步骤中进行设置。

### 2.2.3 控制层 C-Controller 的添加

控制层的作用是指向程序的流程，如果在 V-View 视图层输入正确的“用户名”和“密码”后就能显示 true.jsp 页面，如果失败则显示 false.jsp 页面。

#### 1. 创建 JSP 文件

在这里，要另外创建 true.jsp 和 false.jsp 这两个 JSP 文件。

(1) 用鼠标右击“WebRoot”节点中的“form”节点，在弹出的菜单中选择“New”“JSP (Advanced Templates)”命令，如图 2-18 所示。

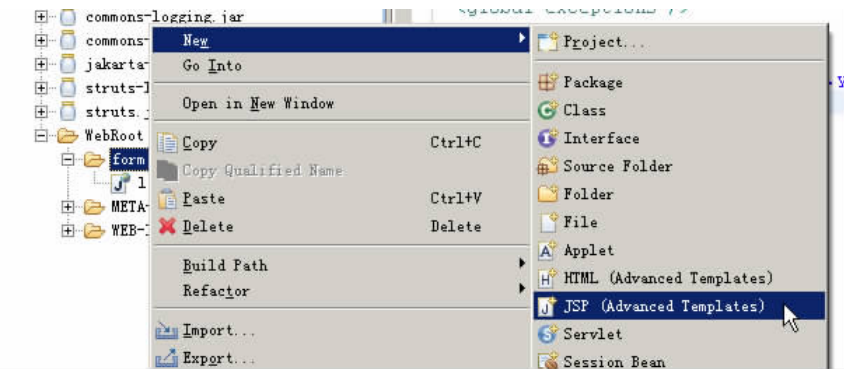


图 2-18 创建一个 JSP 文件

出现如图 2-19 所示的界面，在“File Name”文本框中输入“true.jsp”，其他保持默认设置。

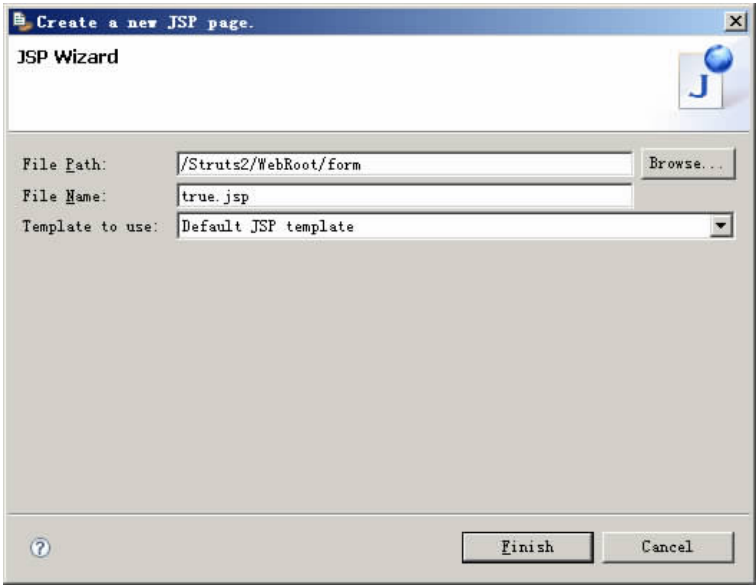


图 2-19 创建 true.jsp 文件

单击“Finish”按钮后即创建完成一个 true.jsp 文件。  
更改 true.jsp 文件中标记<body>和</body>之间的内容如下。

```
<body>
welcome you!~
</body>
```

(2) 重复第 (1) 步，继续创建 false.jsp 文件。更改文件内容如下。

```
<body>
username or password wrong!~
</body>
```



## 2. 创建 ActionForward 转发

在此，我们需要创建两个 Forward 转发：一个转发到 true.jsp，另一个转发到 false.jsp。

这里为什么要使用 Struts 中特有的 ActionForward 呢？因为 Struts 最主要的优点就是后期维护方便，如果一直持续使用开发 JSP 的习惯，比如在 JSP 页面中经常出现如下代码。

```
<jsp:forward page="true.jsp">
```

如果是中小型的项目这样写可以，但如果在大项目中存在这样的语句，这句代码在 10 000 个 JSP 页面每一页都有的话，假如客户的业务功能逻辑突然改变，改成：

```
<jsp:forward page="newtrue.jsp">
```

则不得不改动 10 000 个 JSP 文件中的转发，这时改动量是 10 000 次。

使用 ActionForward 的优点就是提供 URL 逻辑名称和 URL 物理地址匹配对应的一种方法，改变 URL 物理地址并不影响 URL 逻辑名称。如果转发的物理地址改变，只需要改变 ActionForward 中的 URL 物理地址，但我们在程序中使用的 10 000 个 URL 名称并没有改变。结合上面的小示例，如果使用 Struts 中的 ActionForward，则只改动了 1 次代码，和使用 JSP 的方法改动 10 000 次相比是不是有很少的工作量呢？

(1) 现在开始创建刚才提及的在 Struts 中具有神秘性质的 ActionForward。双击“WebRoot”节点的“Web-INF”节点中的“struts-config.xml”文件，如图 2-20 所示。

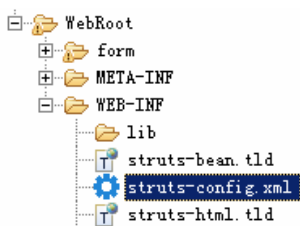


图 2-20 struts-config.xml 文件位置

(2) 打开“struts-config.xml”文件后，选择“Design”模式，如图 2-21 所示。

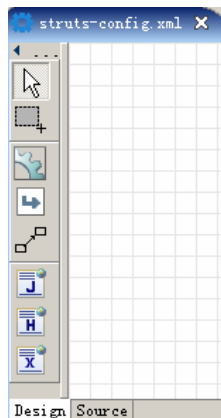


图 2-21 “Design”模式

(3) 在空白处单击鼠标右键，在弹出的菜单中选择“New”“Forward”命令，新建一个 Forward 转发，如图 2-22 所示。

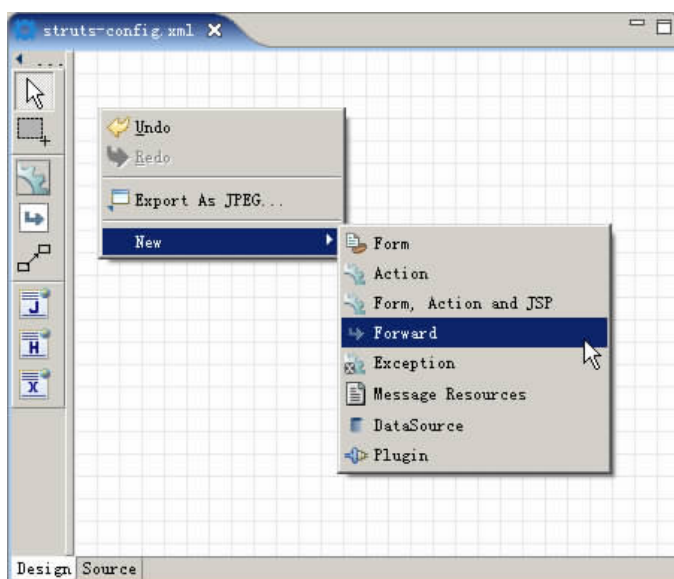


图 2-22 新建一个 Forward 转发

(4) 出现如图 2-23 所示的窗口，设置“Name”为“true”，即 Forward 的 URL 逻辑名称；设置“Path”为“/form/true.jsp”，即 Forward 的物理 URL 地址。单击“Finish”按钮，设置结束。

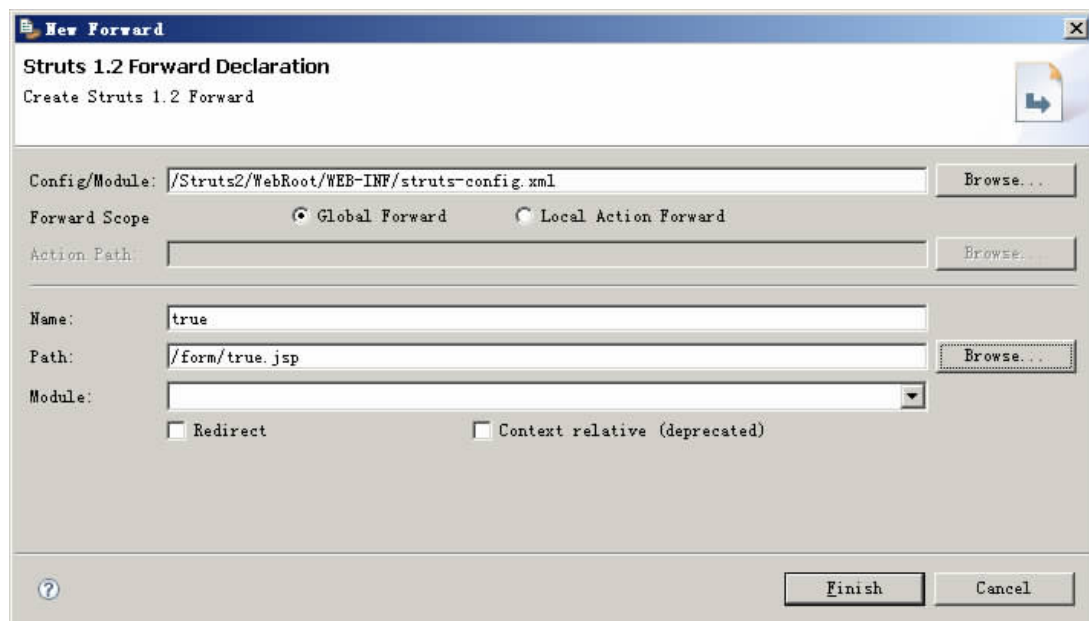


图 2-23 新建 Forward 选项

(5) 重复上一步, 再创建一个“Name”为“false”, “Path”为“/form/false.jsp”的 Forward。

(6) 保存 struts-config.xml 文件。

控制层 C-Controller 设置完成。

控制层 C-Controller 中的转发主要使用 ActionForward 来实现, 它是 Struts 中的一个类。

#### 2.2.4 模型层 M-Model 的添加

虽然 Struts 是一个基于 MVC 模式的 Web 开发框架, 但模型层 M-Model 的数据持久化技术需要第三方的框架来实现, 比如 Hibernate 等。Struts 的侧重点是 V(视图)层和 C(控制)层, 当然这也不是一个坏消息, M(模型)层的功能实现通过第三方的框架会使软件的设计更加合理, 把别人精华的东西应用到项目中, 何乐而不为呢?

在这一小节创建一个“伪 M 层模型”。

(1) 在 Struts2 项目中的“WebRoot”节点上单击鼠标右键, 在弹出的菜单中选择“New”“Other”命令, 如图 2-24 所示。

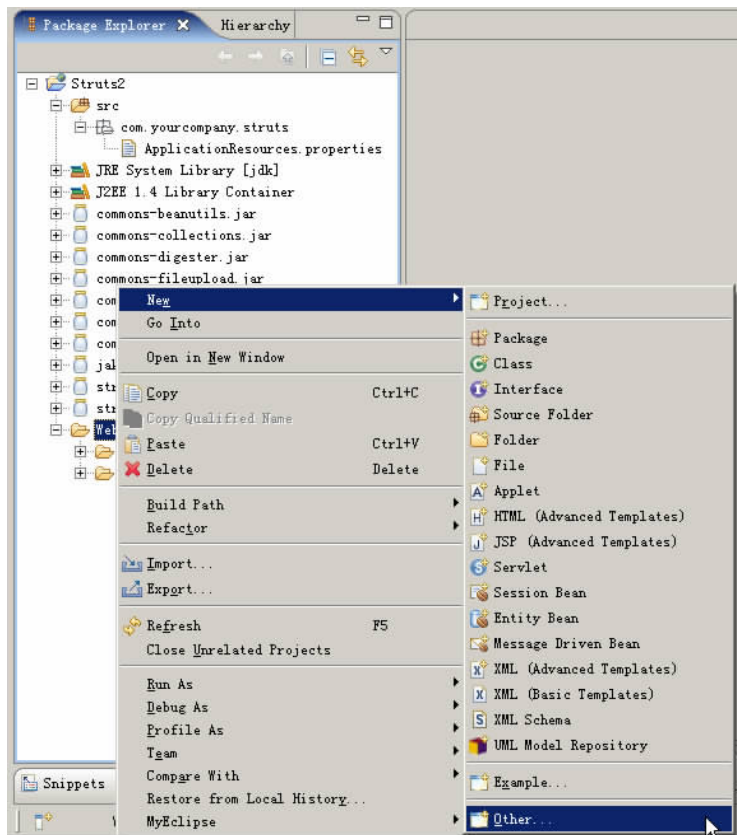


图 2-24 新建一个 Action

(2) 在打开的窗口中依次单击“ MyEclipse ” “ Web-Struts ” “ Struts 1.2 ” “ Struts1.2 Action ”, 如图 2-25 所示。

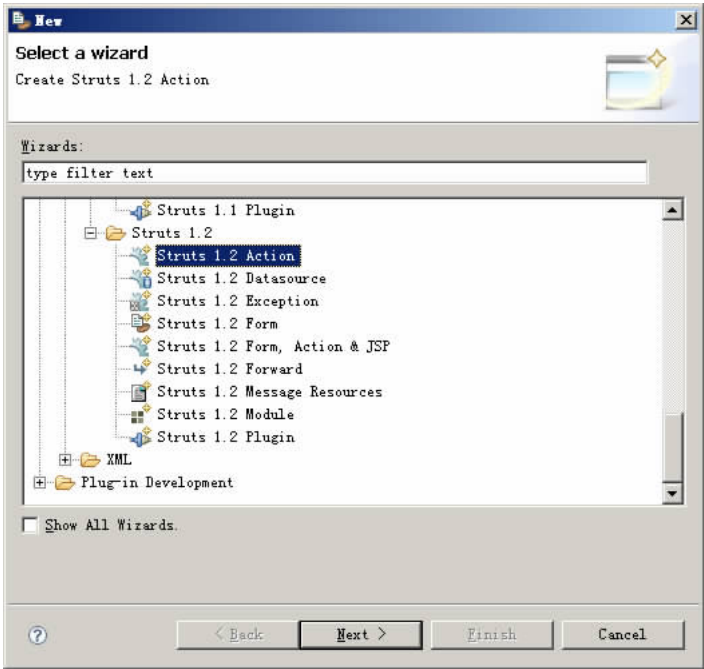


图 2-25 新建一个 Action

(3) 单击 Struts 1.2 Action 后，出现如图 2-26 所示的设置窗口。

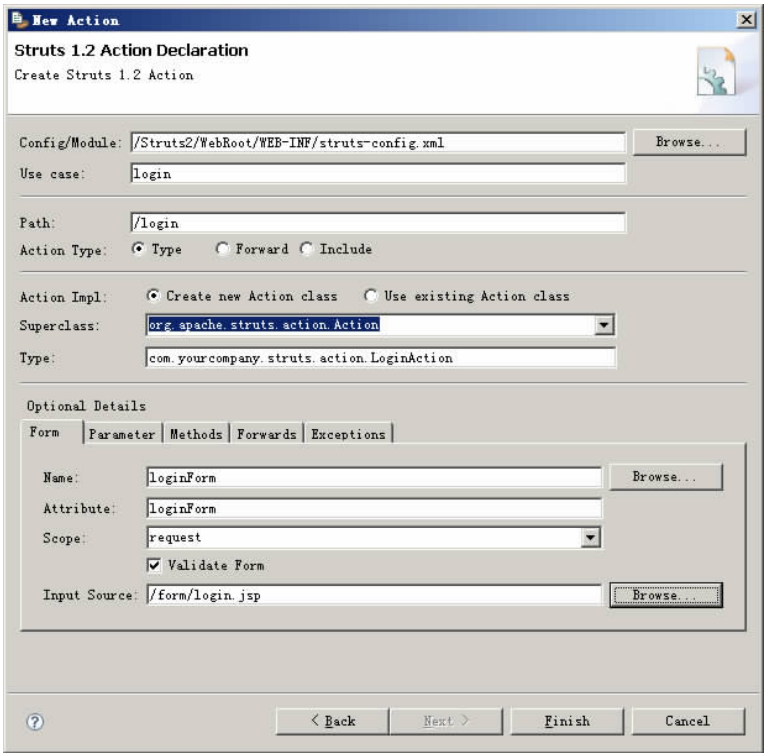


图 2-26 Action 设置

- 设置 Use case 为 login。”使用 login.do 的形式来访问这个 Action。
- 设置 Form 页中的 Name 属性为 loginForm。”接收 LoginForm.java 封装的 Java Bean。
- 设置 Input Source 为 /form/login.jsp。”得知是由哪个 JSP 页面发出的请求。
- 也可以设置这个 Java Bean 的 Scope 的范围，这里选择默认的 “request。”

(4) 单击 “Finish” 按钮应用设置。

这时请保存自动添加 XML 代码的 struts-config.xml 文件。

(5) 编写 M(模型)层的逻辑功能程序，双击 LoginAction.java 文件，如图 2-27 所示。

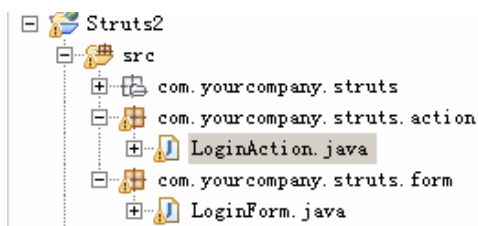


图 2-27 LoginAction.java 文件位置

(6) 增加 “用户名” 和 “密码” 检验程序代码，LoginAction.java 程序代码如下。

```
package com.yourcompany.struts.action;

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 com.yourcompany.struts.form.LoginForm;

public class LoginAction extends Action {

    public ActionForward execute(ActionMapping mapping, ActionForm form,
        HttpServletRequest request, HttpServletResponse response) {
        LoginForm loginForm = (LoginForm) form; // TODO Auto-generated method
        // stub
        String username = loginForm.getUsername();
        String password = loginForm.getPassword();
        // 这里的用户名和密码假设为"abc"和"xyz"
        if ((username.equals("abc")) && (password.equals("xyz"))) {
            return mapping.findForward("true");
        } else {
            return mapping.findForward("false");
        }
    }
}
```

当这段程序运行完成后，Action 根据条件的判断，会有选择性地执行某一个 ActionForward。中心控制类 ActionServlet 会使用这个 ActionForward 的物理路径来进行路径的跳转，从而改变不同的视图。在 Struts 中，ActionForward 的路径映射存放在 ActionMapping 类中，它的内容来源于当软件运行时首先存入内存的 struts-config.xml 文件。

Struts 中的 Action 类主要负责以下 4 项工作。

- 获得 ActionForm 中属性信息。
- ActionForm 属性信息的逻辑验证。
- 利用这些属性信息进行的业务逻辑运算。
- 根据业务逻辑运算结果和当前状态选择转发路径。

### 2.2.5 结尾前小小的改动

在前面我们已经创建了 MVC 的 3 个模块，项目结束前还需要改动 login.jsp 文件中的程序，将原来的程序：

```
<html:form action="/[ACTION_PATH]">
```

改成：

```
<html:form action="/login.do">
```

这样，我们就将前端的 V（视图）端与 M（模型）端进行连接了。

### 2.2.6 struts-config.xml 文件

struts-config.xml 文件的代码如下所示。

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE struts-config PUBLIC "-//Apache Software Foundation//DTD Struts
Configuration 1.2//EN" "http://struts.apache.org/dtds
/struts-config_1_2.dtd">

<struts-config>
  <data-sources />
  <form-beans >
    <form-bean name="loginForm"
      type="com.yourcompany.struts.form.LoginForm" />
  </form-beans>

  <global-exceptions />
  <global-forwards >
    <forward name="true" path="/form/true.jsp" />
    <forward name="false" path="/form/false.jsp" />
  </global-forwards>

  <action-mappings >
    <action
      attribute="loginForm"
      input="/form/login.jsp"
```

```
name="loginForm"
path="/login"
scope="request"
type="com.yourcompany.struts.action.LoginAction" />
</action-mappings>

<message-resources
parameter="com.yourcompany.struts.ApplicationResources" />
</struts-config>
```

研究一下这个总控制中心的重要文件。

(1) 定义 `com.yourcompany.struts.form.LoginForm` 类的别名。

```
<form-beans >
  <form-bean name="loginForm"
    type="com.yourcompany.struts.form.LoginForm" />
</form-beans>
```

这段代码的功能是定义 `com.yourcompany.struts.form.LoginForm` 类的别名，相当于实例变量的名字。

(2) 定义 `ActionForward`。

```
<global-forwards >
  <forward name="true" path="/form/true.jsp" />
  <forward name="false" path="/form/false.jsp" />
</global-forwards>
```

这段代码的功能是定义 `ActionForward`，在 Struts 中，`Forward` 分为两种：一种是局部的，另一种是全局的。局部的 `Forward` 只是在当前的 `Action` 对象中有效，在其他的 `Action` 类中无效。这里定义的是全局的，既然是全局的，那么也就是所有的 `Action` 都能使用这些 `Forward` 概念。

(3) 配置 `Action`。

```
<action
  attribute="loginForm"
  input="/form/login.jsp"
  name="loginForm"
  path="/login"
  scope="request"
  type="com.yourcompany.struts.action.LoginAction" />
```

- `input`：指定从哪个页面提交请求。
- `path`：`action` 的路径，例如 `<html:form action="/login.do">` 也可以使用 `<html:form action="/login">` 的形式。
- `name`：指定当前的 `Action` 由哪个 `ActionForm` 传进 Java Bean。

`attribute` 和 `name` 有着微妙的区别。

`ActionForm` 被存储在一定的 `scope` 中（`request` 或 `session`，通过 `action` 的 `scope` 属性来配置），当我们在配置时，指定 `name` 而不指定 `attribute`，那么指定的 `name` 值就作为 `ActionForm`

存储在 scope 中的 key 值，这样就可以在 action 中通过 `HttpServletRequest.getAttribute("指定的 name 属性值")` 来获得这个 `ActionForm`；当既配置了 name 又配置了 attribute 时，那么 `ActionForm` 存储在 scope 中的 key 值就采用 attribute 属性指定的值，这时要通过 `HttpServletRequest.getAttribute("指定的 attribute 属性值")` 来获得 `ActionForm`，此时通过 `HttpServletRequest.getAttribute("指定的 name 属性值")` 是不能获得 `ActionForm` 的。在实际开发中，一般不必去理会 attribute 属性的设置。

另外，当不再需要存储在 request 或 session 中的 `ActionForm` 时，我们就应该把这个 `ActionForm` 从 request 或 session 中删除，而不应该再让 `ActionForm` 随 request 或 session 传递到页面中，这时就要用到这个 key 值了。

示例代码（`action.execute()`方法中）如下。

```
if ("request".equals(mapping.getScope())) {  
    request.removeAttribute(mapping.getName());  
} else if ("session".equals(mapping.getScope())) {  
    request.getSession().removeAttribute(mapping.getName());  
}
```

当然，当配置了 attribute 属性时，上面示例代码中的 `mapping.getName()` 就要用 `mapping.getAttribute()` 替代了。

所以，是否配置 attribute 属性就决定了 `ActionForm` 存储在 scope 中的 key 值是采用 name，还是采用 attribute。

### 2.2.7 部署项目并运行

（1）在浏览器的地址栏中输入 `http://localhost:8080/Struts2/form/login.jsp`，出现如图 2-28 所示的界面。



图 2-28 浏览器中的登录页面

（2）在“password”中输入正确密码“xyz”，在“username”中输入正确用户名“abc”后，单击“Submit”提交按钮，此时用户名和密码都为正确的，转到如图 2-29 所示的页面。



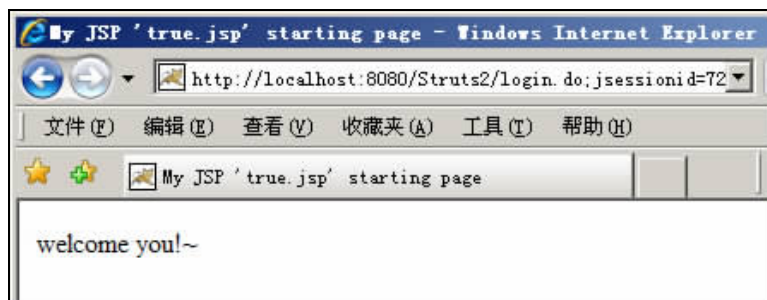


图 2-29 正确页面

如果输入错误的用户名和密码，则转到如图 2-30 所示的页面。



图 2-30 错误页面

好了，第一个 Struts 的实例已经成功完成了。

读者是不是感觉开发这么一个简单的登录检验功能要比使用纯 JSP 开发要麻烦许多？这就是 Struts 的特点，前期开发麻烦一些，但对以后的维护工作还是相当有利的。

下一章一起研究 Struts 中的控制器吧！

# 第 3 章

## 应用中的 C-Controller 控制层

### 3.1 ActionServlet 类的作用

ActionServlet 类是 Struts 的控制中心类，所有的 URL 地址映射、ActionForm 的匹配、Action 的执行都需要这个类来进行导航。下面是 ActionServlet 类的继承关系结构。

```
java.lang.Object
|
+--javax.servlet.GenericServlet
|
+--javax.servlet.http.HttpServlet
|
+--org.apache.struts.action.ActionServlet
```

Struts 提供了一个默认版本的 ActionServlet 类，开发者可以继承这个类，覆盖其中的一些方法来达到某些特殊处理的需要。ActionServlet 类继承自 javax.servlet.http.HttpServlet 类，所以在本质上它和一个普通的 Servlet 没有区别，你完全可以把它当做一个 Servlet 来看待，只是在其中完成的功能不同罢了。

ActionServlet 主要完成如下功能。

- 将一个来自客户端的 URI 映射到一个相应的 Action 类。
- 如果这个 Action 类是第一次被调用，那么实例化一个对象并放入缓存。
- 如果在配置文件（struts-config.xml）中指定了相应的 ActionForm，那么从 Request 中抓取数据填充 FormBean。
- 调用这个 Action 类的 execute() 方法，传入 ActionMapping 对象的一个引用和对应的由表单域封装的 ActionForm 类，以及由容器传给 ActionServlet 的 HttpServletRequest 请求处理和 HttpServletResponse 应答处理对象。

默认版本的 ActionServlet 会从配置文件 web.xml 中读取如下初始化参数。

- application：应用使用的资源包（resources bundle）的基类。
- factory：用于创建应用的 MessageResources 对象的 MessageResourcesFactory 的类名。默认是 org.apache.struts.util.PropertyMessageResourcesFactory。
- config：Struts 的配置文件，默认是 /Web-INF/struts-config.xml。注意这里是与应用 Context 关联的相对路径。

- `content` :定义了默认的内容类型和编码格式,它会被自动地设置到每个 `response` 中,如果 JSP/Servlet 中没有明确的设置,默认是 `text/html`。
- `debug` :调试信息的级别,默认为 0,比当前级别高的调试信息会被记录到日志文件中。
- `detail` :与 `debug` 的作用类似,只是这个 `detail` 是 `initMapping()` 专用的。调试信息会被打印到 `System.out` 文件中,而不是日志文件。
- `formBean` : `ActionFormBean` 的实现类,默认为 `org.apache.struts.action.ActionFormBean`。
- `forward` :应用中使用的 `ActionForward` 类,默认为 `org.apache.struts.action.ActionForward`。
- `locale` :指定了默认使用的 `Locale` 对象。设为 `true`,当得到一个 `session` 时,会自动在 `session` 中存储一个以 `Action.LOCALE_KEY` 标示的 `Locale` 对象。
- `mapping` :应用中使用的 `ActionMapping` 类,默认为 `org.apache.struts.action.ActionMapping`。
- `multipartClass` :文件上传使用的 `MutipartRequestHandler` 实现类。默认为 `org.apache.struts.upload.DiskMultipartRequestHandler`。
- `nocache` :如果设为 `true`,那么 `ActionServlet` 会自动在每个到客户端的响应中添加 `nocache` 的 HTML 头,这样客户端就不会对应用中的页面进行缓存。默认为 `false`。
- `null` :如果设置为 `true`,那么应用在得到一个未定义的 `message` 资源时,会返回 `null`,而不是返回一个错误信息。默认为 `true`。
- `maxFileSize` :文件上传的大小上限,默认为 250MB。
- `bufferSize` :文件上传时的缓冲区的大小,默认为 4MB。
- `tempDir` :设置用于上传时的临时目录。工作目录会作为一个 `Servlet` 环境 (`Context`) 的属性被提供。
- `validating` :在解析配置 XML 文件时是否进行有效性的验证。默认为 `true`。

在 `ActionServlet` 中应用了命令设计模式。

一个 `Servlet` 在由容器生成时,首先会调用 `init()` 方法进行初始化,在接到一个 HTTP 请求时,调用相应的方法进行处理。比如 GET 请求调用 `doGet()` 方法,POST 请求调用 `doPost()` 方法。

在 Struts 中,担任 MVC Model 2 控制器角色核心的是 `ActionServlet` 类,所有的请求都必须先通过它,当 `ActionServlet` 类收到 GET 或 POST 的请求时,其 `doGet()` 或 `doPost()` 会调用 `process()` 方法来处理请求。

```
protected void process(HttpServletRequest request,
                        HttpServletResponse response)
    throws IOException, ServletException {
    RequestUtils.selectApplication(request, getServletContext());
    getApplicationConfig(request).getProcessor().process(request,
response);
}
```

RequestUtils 是一个工具类，ActionServlet 调用其 selectApplication() 方法，由 request.getServletPath() 来取得请求路径以选择应用程序模块来处理请求，之后从 ApplicationConfig 对象中取得 RequestProcessor 对象，将使用者的请求委托它来进行处理。

通常将 ActionServlet 当做黑盒子，然而也可以继承 ActionServlet 来定义自己的控制器，但在 Struts 1.2 中大部分的请求已经委托给 RequestProcessor 来处理，继承 ActionServlet 来定义自己的控制器请求处理意义已经不大，通常的目的是重新定义 ActionServlet 的 init() 方法，增加自己的初始化动作。

```
public class CustomActionServlet extends ActionServlet {
    public void init() throws ServletException {
        super.init();

        // 增加自己的初始化动作
        .....
    }
}
```

对于一个 JSP 请求，服务器端自动调用 ActionServlet、ActionMapping、ActionForm、Action、ActionForward 等组件和运行它们相应的方法。

ActionServlet 的方法主要有：process()、InitApplication()、InitMapping()、InitDigester()、InitOther()。ActionServlet 提供了公共方法，可以被 Action 类实例使用。ActionServlet 包含允许增加或删除 ActionForm Beans、ActionForwards 和 ActionMappings 的方法。这些方法的基本形式如下所示。

```
public void addFormBean(ActionFormBean formBean)
public void removeBean(ActionFormBean formBean)
public void addForward(ActionForward forward)
public void removeForward(ActionForward formward)
public void addMapping(ActionMapping mapping)
public void removeMapping(ActionMapping mapping)
```

每个定义显示了方法的范围（都为 public）、方法返回的对象（都为 void）以及方法的参数，下列方法根据名字找到这些对象。

```
public ActionFormBean findFormBean(String name)
public ActionForward findForward(String name)
public ActionMapping findMapping(String name)
```

接下来的两个方法用于处理数据源。

```
public void addDataSource(String key, DataSource ds)
public DataSource findDataSource(String key)
```

findDataSource() 方法用名字查找数据源。数据源可能是在 Struts 配置文件中静态定义的，也可能是用 addDataSource() 方法动态增加的。

最后，可以用 destroy() 方法关闭 ActionServlet，并用 reload() 方法从 Struts 配置文件中重新装载信息到 ActionServlet 中。

### 3.1.1 process()方法的执行过程

默认的 RequestProcessor 类是 org.apache.struts.action.RequestProcessor，可以查看 process()方法的源代码来了解它做了哪些事情，如下面程序代码所示。

```
public void process(HttpServletRequest request,
                    HttpServletResponse response)
    throws IOException, ServletException {
    // 处理 contentType 为 multipart/form-data 的 POST 请求
    request = processMultipart(request);

    // 取得 URI 路径
    String path = processPath(request, response);
    if(path == null)
        return;
    .....

    // 确定客户端的位置，是否要将一个 Locale 对象储存在 session 中
    // 配合<controller>的 locale 属性使用
    processLocale(request, response);

    // 确定 contentType，默认是 text/html
    processContent(request, response);

    // 判断<controller>属性 nocache 是否被设置
    // 若是，在 response 中加入防止缓存的 header
    processNoCache(request, response);

    // 准备处理，默认返回 true，子类可以重新定义它以决定要不要继续处理
    if(!processPreProcess(request, response)) {
        return;
    }

    // 从 URI 路径确定 ActionMapping
    ActionMapping mapping = processMapping(request, response, path);
    if(mapping == null) {
        return;
    }
    .....

    // 处理 ActionForm，如果没有就新建一个，之后一直使用它
    ActionForm form = processActionForm(request, response, mapping);

    // 将浏览器的字段值填入 ActionForm
    processPopulate(request, response, form, mapping);

    // 判断是否执行 ActionForm 的 validate() 方法
    if(!processValidate(request, response, form, mapping)) {
```

```
        return;
    }

    // 判断<action>标记中的 forward 或 include 标记是否被设定，
    // 这两个标记在设定一个路径，其与 type 属性是互斥的，当设定
    // 其中一个属性时，调用 RequestDispatcher 的 forward()或
    // include()方法，其作用与设定 ForwardAction 或 IncludeAction 相同，
    // 直接进行逻辑处理，而不再使用 Action 对象进行接下来的处理
    if(!processForward(request, response, mapping)) {
        return;
    }
    if(processInclude(request, response, mapping)) {
        return;
    }

    // 处理 Action，如果没有就生成一个，之后一直使用它
    Action action = processActionCreate(request, response, mapping);
    if(action == null) {
        return;
    }

    // 执行 Action 的 execute()方法，并返回 ActionForward
    ActionForward forward = processActionPerform(request, response, action,
        for, mapping);

    // 处理 ActionForward
    processActionForward(request, response, forward);
}
```

以上的程序代码就是一个 Struts 运行的典型流程。

### 3.1.2 process()方法执行过程总结

一个 ActionServlet 类接收到客户请求时大体执行了如下的几个重要步骤。

- (1) 从 URI 路径确定 ActionMapping。
- (2) 处理 ActionForm，如果没有就新建一个，之后一直使用它。
- (3) 将浏览器的字段值填入 ActionForm。

(4) 判断是否执行 ActionForm 的 validate()方法，如果执行 validate()方法返回一个 NULL 值或返回一个不包含任何 ActionMessage 的 ActionErrors 对象，代表当前的验证数据为正确的，继续执行下一步。否则的话转回 input 属性所代表的页面，并在页面中显示 ActionErrors 对象中的错误信息。

- (5) 处理 Action，如果没有就生成一个，之后一直使用它。
- (6) 执行 Action 的 execute()方法，并返回 ActionForward。
- (7) 处理 ActionForward。

## 3.2 Action 类的作用

在 Struts 中, `ActionServlet` 担任分配工作的中心控制器角色, 实际上的工作是交给 `Action` 对象来进行的, `ActionServlet` 由 `ActionMapping` 得知所使用的 `Action` 对象, 将工作交给它, 并在最后由 `Action` 对象得到一个 `ActionForward` 对象, `ActionServlet` 使用这个 `ActionForward` 来知道下一个 `Forward` 的对象, 并执行处理。

### 3.2.1 Action 的工作

对于 Struts, 一个 `ActionMapping` 只会生成一个 `Action` 对象, 当请求到达时, 会检查所需的 `Action` 对象是否存在, 如果不存在则生成一个, 之后一直使用它, 由于 `Action` 对象会一直存在, 所以使用 `Action` 对象必须注意到执行期安全问题。

`Action` 对象是执行工作的对象, 它主要的工作如下。

- 验证使用者的进程状态。
- 进一步验证窗体对象的信息。
- 更新应用程序中对象的状态。
- 处理客户端的请求。
- 返回 `ActionForward` 给 `ActionServlet` 转发到下一个视图。

为了能够重用 `Action` 对象, 通常建议不要在 `Action` 中加入过多的逻辑, 普遍认为 `Action` 对象也应属于控制器角色, 除了以上必要的工作之外, 建议将与业务相关的工作交给辅助类, 减少 `Action` 对象中的逻辑, 比如存取数据库时在 `Action` 类中增加 `import db.java` 的 `Java Bean`, 以使得 `Action` 对象更适合做它自己的工作。

基本上不是只有 `Action` 对象应保持内容的清晰, 就 Struts 而言, 它的工作是辅助使用 MVC Model 2 架构, 每个相关的组件除了完成 MVC 模型每层的目的作用之外, 没有其他的作用, Struts 建立起来的架构也鼓励重用, 应当将与业务相关的逻辑交由其他的对象来处理, 而不是交给 Struts 组件。

可以在 `Action` 类中对数据库进行增、删、改、查等操作。

在上一章所介绍的实例中, 在 `Action` 类中对“用户名”和“密码”进行了验证, 这样的一个验证就是一个 `Action`, 那么是不是每一个逻辑功能都要有一个 `Action` 呢? 其实不然, `Action` 类中可以聚合相似的功能, 例如在一个 `Action` 类中可以对数据库进行增、删、改、查操作, 而不需要创建 4 个 `Action` 类, 这样的功能使用起来非常方便, 在本书的后面会有专门的章节进行介绍。

每一个 `Action` 的子类都要实现 `execute()` 方法。

### 3.2.2 在 Action 类中进行用户名验证的实例

在没有 Struts 框架时, 经常使用 `Java Bean+Servlet` 来进行用户名的验证, 在这里, 先做一个简单的用户名登录认证的实例, 当然, 在 Struts 中 `Servlet` 已经被 `Action` 类代替了, 但从本质上来讲, `Action` 其实就是一个 `Servlet`。

在本书以后的实例中，将对界面操作的过程介绍越来越少，大部分的操作已经在第 2 章中详细介绍过。

(1) 在 MyEclipse 中新建一个工程 Struts3.2.2，再新建 JSP、Action、ActionForm 文件，如图 3-1 所示。

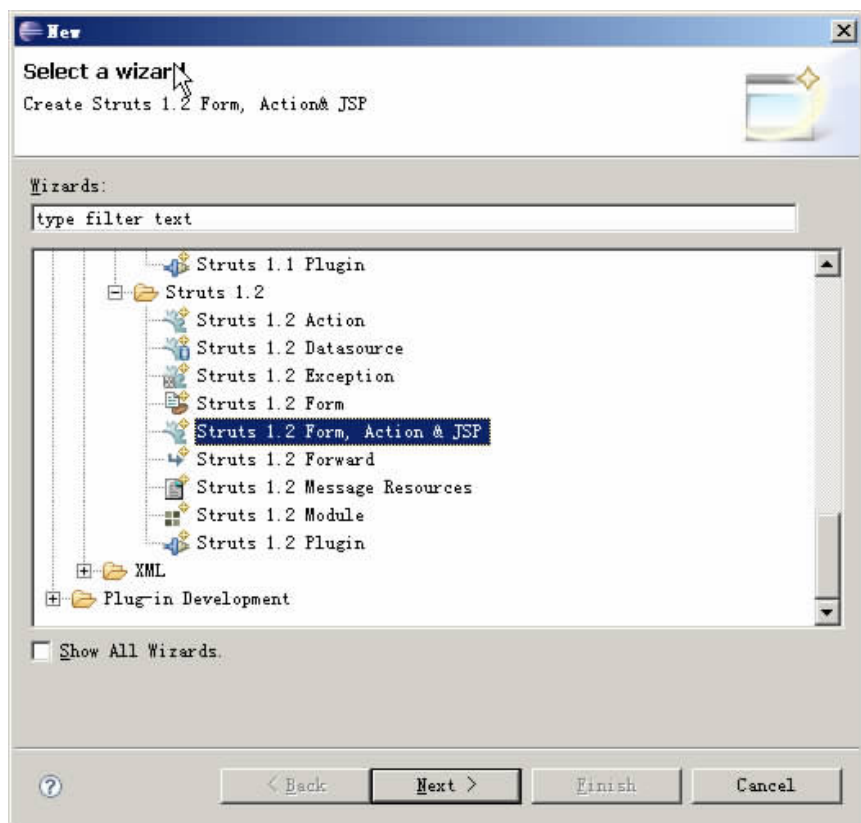


图 3-1 新建 JSP、Action、ActionForm 文件

(2) 单击“Next”按钮后，出现如图 3-2 所示的界面。

(3) 在该窗口中的“Use case”文本框中输入“login”，设置“Superclass”类为“ActionForm”，再添加一个 username 的表单域，数据类型为 String，表单类型为<html:text/>，设置完 ActionForm 后，还需要创建对应的 JSP 文件，如图 3-3 所示。

(4) 创建 JSP 文件后，单击“Next”按钮转到配置 Action 类的窗口中，在 Action 窗口中不需要任何的配置，只需使用默认的配置即可，单击“Finish”按钮新建 JSP、Action、ActionForm 完成。



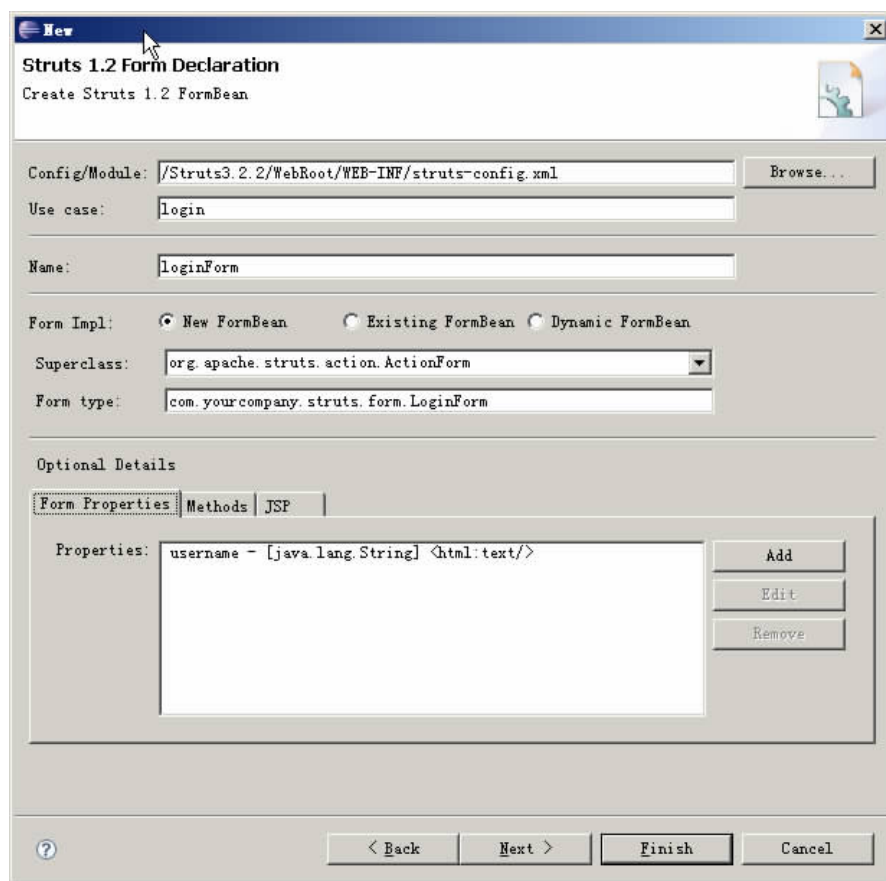


图 3-2 配置 ActionForm 和 JSP 选项

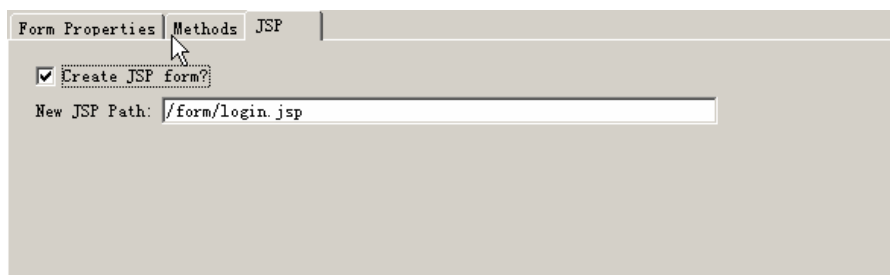


图 3-3 新建对应的 JSP 文件

生成的 JSP 程序代码如下所示。

```
<%@ page language="java" pageEncoding="ISO-8859-1"%>
<%@ taglib uri=http://jakarta.apache.org/struts/tags-bean
prefix="bean"%>
<%@ taglib uri=http://jakarta.apache.org/struts/tags-html
prefix="html"%>

<html>
<head>
```

```
<title>JSP for LoginForm form</title>
</head>
<body>
  <html:form action="/login">
    username : <html:text property="username"/><html:errors
    property="username"/><br/>
    <html:submit/><html:cancel/>
  </html:form>
</body>
</html>
```

从 JSP 的代码<html:form action="/login">可以得知，表单要提交的路径是/login，这个路径得在 struts-config.xml 文件中进行配置。本节的后面将介绍 struts-config.xml 配置文件的内容。

在 JSP 文件中有表单的提交，但在 Struts 中，在常规的情况下也应该有 ActionForm 类的封装，下面的程序代码就是封装前台 JSP 表单提交过来的数据，并打包成 javabean，生成了针对 username 用户名域的 get 和 set 方法。

```
package com.yourcompany.struts.form;

import javax.servlet.http.HttpServletRequest;
import org.apache.struts.action.ActionErrors;
import org.apache.struts.action.ActionForm;
import org.apache.struts.action.ActionMapping;

public class LoginForm extends ActionForm {
    private String username;

    public ActionErrors validate(ActionMapping mapping,
        HttpServletRequest request) {
        // TODO Auto-generated method stub
        return null;
    }

    public void reset(ActionMapping mapping, HttpServletRequest request) {
        // TODO Auto-generated method stub
    }

    public String getUsername() {
        return username;
    }

    public void setUsername(String username) {
        this.username = username;
    }
}
```

通过上面的程序将前台表单数据封装成 ActionForm 后，通过 ActionServlet 类传递给 Action 进行功能业务逻辑处理，在 Action 中必须在 execute()方法中写入相关的功能代码，

比如在本例中需要对用户名进行检验,然后再返回一个 ActionForward 类的对象,转发到处理完成后的结果显示页。Action 类的程序代码如下。

```
package com.yourcompany.struts.action;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import org.apache.struts.action.Action;
import org.apache.struts.action.ActionError;
import org.apache.struts.action.ActionErrors;
import org.apache.struts.action.ActionForm;
import org.apache.struts.action.ActionForward;
import org.apache.struts.action.ActionMapping;

import com.yourcompany.struts.form.LoginForm;

public class LoginAction extends Action {
    public ActionForward execute(ActionMapping mapping, ActionForm form,
        HttpServletRequest request, HttpServletResponse response) {

        response.setCharacterEncoding("GBK");

        LoginForm loginForm = (LoginForm) form; // TODO Auto-generated method
        // stub
        String username = loginForm.getUsername();

        ActionErrors errors = new ActionErrors();
        if (!username.equals("abc")) {
            errors.add("username", new ActionError("username_wrong"));
            this.saveErrors(request, errors);
            return mapping.getInputForward();
        }
        return mapping.findForward("succ");
    }
}
```

首先在 Action 类中将前台传递进来的字符编码转换成 GBK 编码,然后再判断用户名是不是“abc”,如果不是“abc”的话,则生成错误的信息,这个错误的信息通过 ActionErrors 类来实现,这个类就是包装所有的错误信息,通过它的 add 方法来进行添加,虽然 ActionErrors 类有些像错误信息的容器,但错误信息的实际文本内容却是通过 ActionError 类来从资源文件中取得的,这样可以做到国际化,可以使资源文件中的内容可以随着语言版本的不同而更改资源文件中的错误提示内容,不像以前在 JSP+Java Bean 开发时将错误提示硬编码在程序代码中。添加完错误信息后,再通过 saveErrors 方法将错误信息保存到 request 对象中,最后再由 mapping.findForward 方法转发到目标页,一个请求和应答的处理过程随即结束。如果 username 等于“abc”的话,则通过代码 mapping.findForward("succ") 转发到 succ 逻辑页中。

### 3.2.3 ActionErrors 和 ActionError 类的关系和使用

在 LoginAction.java 文件的 execute()方法中进行用户名的验证，主要的程序代码如下。

```
ActionErrors errors = new ActionErrors();
    if (!username.equals("abc")) {
        errors.add("username", new ActionError("username_wrong"));
        this.saveErrors(request, errors);
        return mapping.getInputForward();
    }
```

在上面的程序代码中，首先创建一个 ActionErrors 的对象，这个对象的功能是保存出错信息的容器。然后判断用户名是不是等于“abc”，如果不是，则往 errors 对象中添加错误信息实际的文本内容，errors.add()方法的第一个参数是<html:errors>标记的 property 的名字，就是在指定 property 的<html:errors>标签中显示信息，在这里请看 JSP 页面文件中的<html:errors property="username"/>代码，说明出错信息显示在这个标签中。一个 ActionErrors 对象可以通过使用 add()方法加入多个 ActionError 对象。errors.add()方法的第二个参数是包含出错信息的实际文本内容，这个文本实际内容保存在 ActionError 对象中，如图 3-4 所示为资源文件的位置。

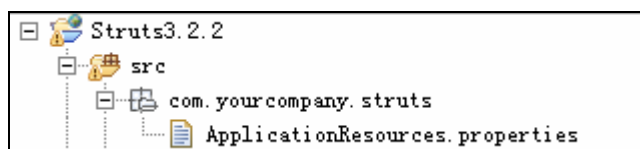


图 3-4 资源文件位置

这个资源文件是维护出错信息内容的，里面的出错信息需要程序员手动输入维护，可以输入所有外语语种，这样可以将出错信息进行国际化。当然此文件并不是只针对保存出错信息的，在这里面可以输入任何想要的国际化文字，比如按钮的标题等。在本实例中的资源文件配置内容如下。

```
username_wrong=your username wrong!~
```

可以看到，资源文件的格式还是非常简单的，等号左边是资源信息的名称，也就是大多数人所谓的 key，右边是要显示信息的内容，也就是 value 值。

在这里，如果出错，要在 login.jsp 页面中的<html:errors property="username"/>位置显示出错信息，所以就要使用 errors.add("username", new ActionError("username\_wrong"));这样的功能语句。

接下来，要将这个 errors 对象保存到 request 对象中，即 this.saveErrors(request, errors);，并且转发给 login.jsp 页面 return mapping.getInputForward();。

否则如果用户名等于“abc”的话，转发到 return mapping.findForward("succ");中。

下面是 struts-config.xml 文件内容。

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE struts-config PUBLIC "-//Apache Software Foundation//DTD Struts
```

```
Configuration 1.2//EN" "http://struts.apache.org/dtds
/struts-config_1_2.dtd">

<struts-config>
  <data-sources />
  <form-beans >
    <form-bean name="loginForm"
      type="com.yourcompany.struts.form.LoginForm" />

  </form-beans>

  <global-exceptions />
  <global-forwards />
  <action-mappings >
    <action
      attribute="loginForm"
      input="/form/login.jsp"
      name="loginForm"
      path="/login"
      scope="request"
      type="com.yourcompany.struts.action.LoginAction" >
      <forward name="succ" path="/form/succPage.jsp" />
    </action>

  </action-mappings>

  <message-resources
parameter="com.yourcompany.struts.ApplicationResources" />
</struts-config>
```

在“struts-config.xml”配置文件中的 Action 中可以看到<forward name="succ" path="/form/succPage.jsp" />，这个 forward 是局部的，只属于当前的 Action，只有当前的 Action 类可以使用，其他的 Action 类不能使用。

如果在浏览器界面的表单文本域中输入非“abc”后，则出现如图 3-5 所示的出错提示。

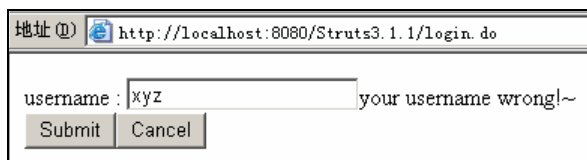


图 3-5 出错提示

如果输入“abc”，则出现如图 3-6 所示的正确效果。



图 3-6 正确效果

## 3.3 ActionForward 类的功能及两种在 Eclipse 中创建 Action Forward 类的方法

下面首先介绍 ActionForward 类的功能。

### 3.3.1 ActionForward 类的功能

在 3.2.2 小节中,使用了 ActionForward 转发对象来转发到相应的 Web 页面中,当然也可以将转发的目标定为 Servlet、其他 JSP 页面或一个 Action 对象。

它的功能有些类似于如下 JSP 命令:

```
<jsp:forward>
```

在 Action 对象中执行功能逻辑代码结束时必须返回一个 ActionForward 对象来更换视图:

```
public ActionForward execute(ActionMapping mapping, ActionForm form,
    HttpServletRequest request, HttpServletResponse response)
```

上面的 execute 方法必须返回一个 ActionForward 对象的实例。

### 3.3.2 在 Eclipse 中创建 ActionForward 类的两种方法

创建 ActionForward 有两种方法,一种是使用更改配置文件 struts-config.xml 的方法,另一种是实例化 new ActionForward 方法。

(1)更改配置文件 struts-config.xml 方法,其实也就是使用 Eclipse 通过图形化的界面来实现的,在以前的例程中已经操作过,大体步骤为:将 struts-config.xml 文件置为 design 模式,在空白处单击鼠标右键,在弹出菜单中选择“New”“Forward”命令,出现如图 3-7 所示的窗口。

在这里面需要注意的是 Global Forward 和 Local Action Forward 的区别:前者是全局的,后面的是局部的。局部的 Forward 属于某一个 Action 对象,其他的 Action 对象不能访问。

如果出现全局和局部的 Forward 重名的情况,优先使用 Action 中局部的 Forward 对象。

(2)实例化 New ActionForward 方法其实也就是在软件运行的过程中使用 Java 程序来动态地创建一个 ActionForward 的对象。示例程序如下所示。

```
public ActionForward execute(ActionMapping mapping, ActionForm form,
    HttpServletRequest request, HttpServletResponse response) {
    // TODO Auto-generated method stub
    ActionForward myForward = new ActionForward();
    myForward.setName("gaohongyan");
    myForward.setPath("/form/myjsp.jsp");
    return myForward;
}
```

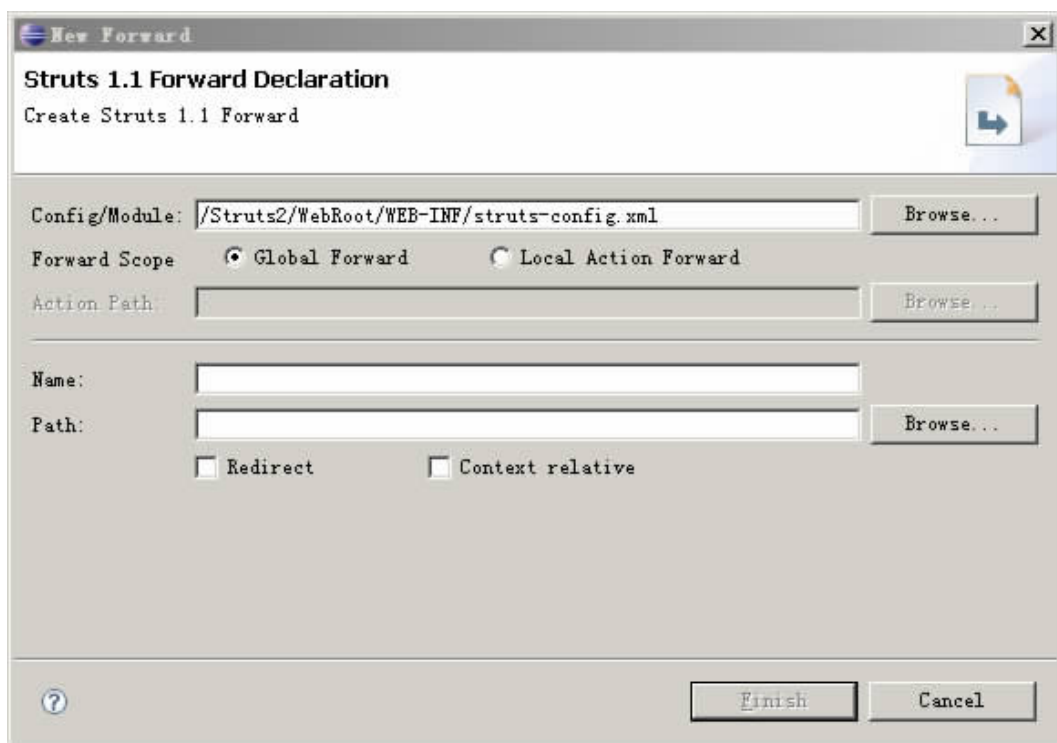


图 3-7 新建一个 ActionForward 转发

在具体的开发过程中推荐使用第 1 种方法，因为第 2 种方法没有将这个 ActionForward 作为全局性的共享，不符合 Struts 的 MVC 中组件复用的规范。

当然，第 1 种方法也是 Struts 官方推荐的方法。

提示：如果在转发完成后，在浏览器中出现空白的页面，那么请检查一下配置文件 struts-config.xml 中的 ActionForward 配置是否正确，看看配置文件中 Forward 中的 name 和代码中的 name 是否一致。如果出现书写不一致的情况，那么浏览器将不显示出异常信息，而是显示空白网页。

### 3.3.3 带参数的 ActionForward

有时需要在 Action 中将 ActionForward 的转发夹带上参数，这种情况应该如何解决呢？在 Action 类中使用如下类似的代码。

```
String path = mapping.findForward("any_forward").getPath();
ActionForward forward = new ActionForward(path + "&abc=123", true);
return forward;
```

## 3.4 使用 ForwardAction 进行页面或 Web 组件的跳转

ForwardAction 的功能类似于 <jsp:forward> 标签。

本小节要介绍的 ForwardAction 类和前面学过的 ActionForward 类功能非常相似，甚至会因为两个单词的位置调换而经常使人容易搞混。

在 Struts 框架包中有很多附加标准的 Action 类，比如 ForwardAction 类，正如这个类的名字一样，它的功能就是转发到其他的 Web 组件。这个 Web 组件可以是一个 Action、JSP、Servlet 或者 URI 的资源。

#### 3.4.1 什么情况下使用 ForwardAction

ForwardAction 功能与超级链接相同。

有些时候，在开发项目时，单击超级链接后只是想要从一个页面跳转到另一个页面，在 MVC Model 2 的架构中，直接使用页面或资源的路径来进行调用并不是一个好的主意，比如<a>这样的超级链接，这会使得控制器没有机会处理相关的请求事宜。在 Struts 的规则里，所有的用户请求都要经过控制器，由控制器来决定具体使用哪个 Web 组件或新的用户视图。如果使用 Struts 的控制器，不使用<a>html 的标记的话，使用 ForwardAction 类就可以完成这个功能，即超级链接。

如果只是想得到将当前的页面像类似 HTML “超级链接”一样，链接到其他页面的效果，不必提交到一个 Action，而直接提交到一个 forwardaction.jsp 文件中时，就要使用 ForwardAction 了。

下一小节，大家一起来做一个 ForwardAction 的实例吧！

#### 3.4.2 一个 ForwardAction 类的实例

##### （1）新建 Web Project 项目

新建一个 Web Project，项目名称为 Struts 3.4.2。

##### （2）新建 JSP 文件

新建两个 JSP 文件：1.jsp 和 forwardaction.jsp。

在新建 JSP 文件时，一定要选择该 JSP 页面支持 Struts 1.2，如图 3-8 所示。

如果支持 Struts 1.2 的 JSP 文件会加入如下的 Struts 标签。

```
<%@ taglib uri="http://struts.apache.org/tags-bean" prefix="bean" %>
<%@ taglib uri="http://struts.apache.org/tags-html" prefix="html" %>
<%@ taglib uri="http://struts.apache.org/tags-logic" prefix="logic" %>
<%@ taglib uri="http://struts.apache.org/tags-tiles" prefix="tiles" %>
```

如果创建的文件没有这 4 个标签，请手动加入。

##### （3）新建一个 ForwardAction 类

新建一个 ForwardAction 类和新建一个 Action 类步骤相似。

新建一个 Action 类，如图 3-9 所示。



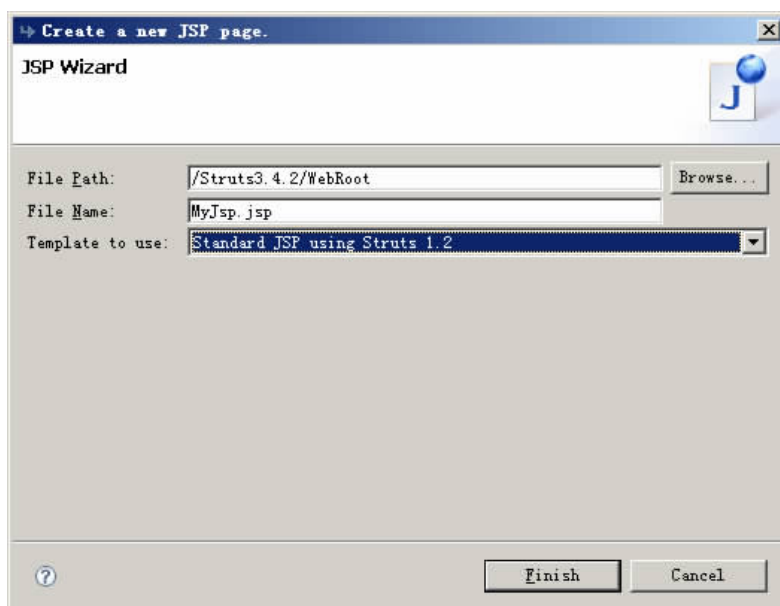


图 3-8 选择支持 Struts 1.2 的 JSP 文件

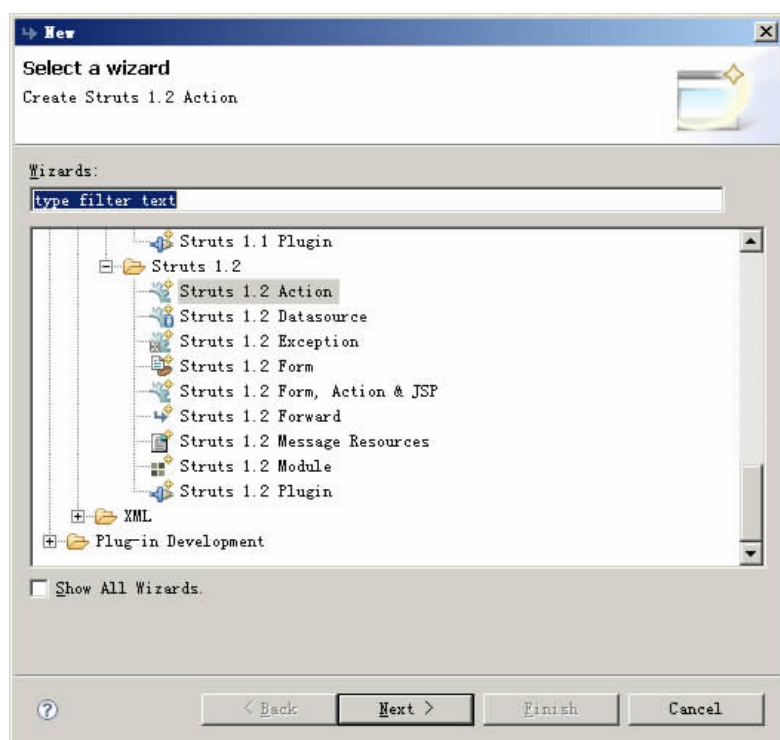


图 3-9 新建一个 ForwardAction 类

单击“Next”按钮后出现如图 3-10 所示的窗口。

在如图 3-10 所示的窗口中选择“Use existing Action class”单选按钮，然后单击如图 3-11 所示的“Browse”按钮。

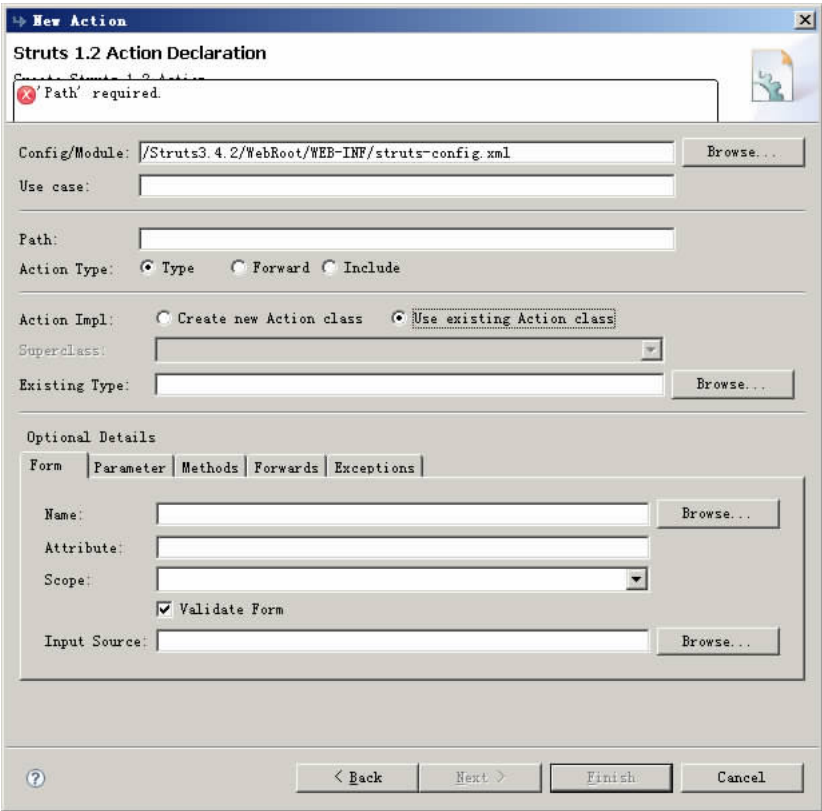


图 3-10 设计 ForwardAction



图 3-11 单击“Browse”按钮

出现如图 3-12 所示的窗口，输入 ForwardAction，选择第一项后，单击“OK”按钮。

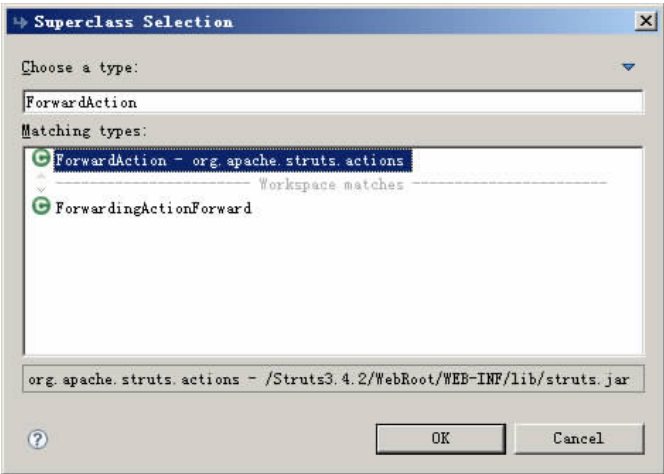


图 3-12 选择 ForwardAction 类

这时在新建 Action 的窗口上方出现一个 path 没有设置的提示，如图 3-13 所示。



图 3-13 path 错误提示

设置 path 为：“/to ForwardAction”。

最后一步也就是设置转发到某一个 JSP 文件了，设置属性如图 3-14 所示。



图 3-14 设置“Parameter”参数

单击窗体下方的“Finish”按钮，新建一个 ForwardAction 类结束。

Struts-config.xml 文件内容如下。

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE struts-config PUBLIC "-//Apache Software Foundation//DTD Struts
Configuration 1.2//EN" "http://struts.apache.org/dtds
/struts-config_1_2.dtd">

<struts-config>
  <data-sources />
  <form-beans />
  <global-exceptions />
  <global-forwards />
  <action-mappings >
    <action
      parameter="/forwardaction.jsp"
      path="/toForwardAction"
      type="org.apache.struts.actions.ForwardAction" />
  </action-mappings>

  <message-resources
    parameter="com.yourcompany.struts.ApplicationResources" />
</struts-config>
```

#### (4) 更改 JSP 文件内容

将 1.jsp 中的<body>内容更改如下。

```
<body>
<html:link action="/toForwardAction.do">toForwardAction</html:link>
</body>
    将 forwardaction.jsp 中的<body>中的内容更改如下：
<body>
hello!~
</body>
```

部署应用程序，并运行。

可以看到单击超级链接后浏览器的地址栏出现：<http://localhost:8080/Struts3.4.2/toForwardAction.do>。网页显示的数据为 forwardaction.jsp 文件的内容。

如果不用这个例子直接使用超级链接的话也可以实现，但有两个缺点。

- 不是 MVC 模式规范。
- 暴露目标地址页。

### 3.4.3 第二种创建 ForwardAction 类的方法

也可以在创建 Action 时，有针对性地选择 Forward 项来创建 ForwardAction 类，如图 3-15 所示。

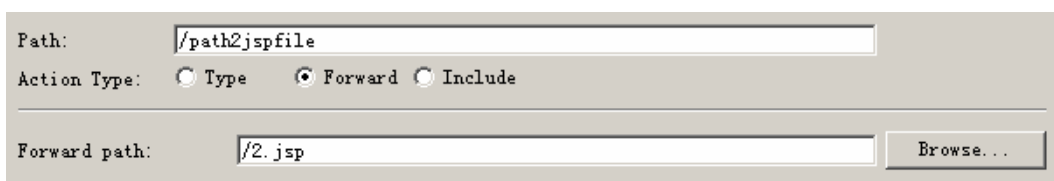


图 3-15 创建 ForwardAction

只需要设置如图 3-14 所示的 3 项内容即可，设置完成单击“Finish”按钮后，struts-config.xml 文件内容如下。

```
<action forward="/2.jsp" path="/path2jspfile" />
```

这时就可以通过 1.jsp 文件中的：

```
<html:link action="/path2jspfile.do">to 2.jsp</html:link>
```

链接到 2.jsp 文件。

这个配置方法是不是比较简单？使用这种吧！

## 3.5 IncludeAction 让年久失用的 Web 组件复用

在使用 JSP 开发的过程中，经常使用：

```
<%@ include file="" %>
```

命令来将静态的文件包含进当前的 JSP 文件中，这是静态的包含，服务器对此的处理是该指令只是静态包含文件，服务器将对包含进来的文件进行编译，只是编译。

本节要学习的是 IncludeAction 类。从拼写上讲`<%@ include file="" %>`和 IncludeAction 类的写法非常相似，但它们的功能却相差很大，一个为静态，一个为动态。

IncludeAction 类和 JSP 命令中的：

```
<jsp:include page="" flush="true/false" />
or
<jsp:include page="" flush="true/false" >
<jsp:param name="" value="" />
.....
</jsp:include>
```

功能类似。

`<jsp:include>` 动作或 Servlet 中的 `RequestDispatcher` 的 `include()` 方法的功能与 IncludeAction 大体一样，都是将其他的 Web 组件包含进当前的项目中，可以传递参数。

`<jsp:include>` 会自行判断包含的文件是动态的还是静态的，可以同时处理这两种文件。

在 JSP 网页中，尽管可以直接通过`<include>`指令包含另一个 Web 组件，但是 Struts 框架提倡先把请求转发给控制器，再由控制器来负责包含其他 Web 组件。IncludeAction 类提供了包含其他 Web 组件的功能。与 ForwardAction 一样，Web 组件通过 IncludeAction 类来包含另一个 Web 组件，可以充分利用 Struts 控制器的预处理功能。

在大多数的 JSP 转到 Struts 项目中时，有很多 Servlet 和 JSP 的组件变得不可用，因为 Servlet 虽然是 MVC 的模式，但和 Struts 要表示的模型还有一些差别，纯 JSP 这种形式的项目不通过控制器处理，不是纯的 MVC 模式。

那么如果在新建的 Struts 项目中还需要老项目的 JSP 和 Servlet 中的 Web 组件时，该怎么办，不会要重新写这些功能逻辑去专门针对 Struts 框架的项目吧？那样工作量太大了，其实不需要，只要使用 Struts 自带的 IncludeAction 类就可以实现重用老 Web 组件的功能。

下面做两个这样的例子。

### 3.5.1 使用 IncludeAction 包含 JSP 文件

在这个例子中，我们使用一个老的 JSP 文件 `old.jsp`。

在这个 `old.jsp` 文件中，需要得到传进来的 `name` 参数来进行一些功能化的处理。

下面开始创建基于 Struts 框架的新项目。

(1) 创建 ActionForm 类和 `new.jsp` 页面。

新建一个 ActionForm，如图 3-16 所示。

选择“Struts 1.2” “Struts 1.2 Form”选项后，单击“Next”按钮创建一个 ActionForm 类，出现如图 3-17 所示的窗口。

在这里我们需要设置 4 项。

- 设置 Use case 为“new。”
- 设置 Superclass 为“org.apache.struts.action.ActionForm。”
- 设置 Form Properties 属性页中的 Properties 为“name”，数据类型为“String。”
- 设置 JSP 属性页中的 New Jsp Path 为“new.jsp。”

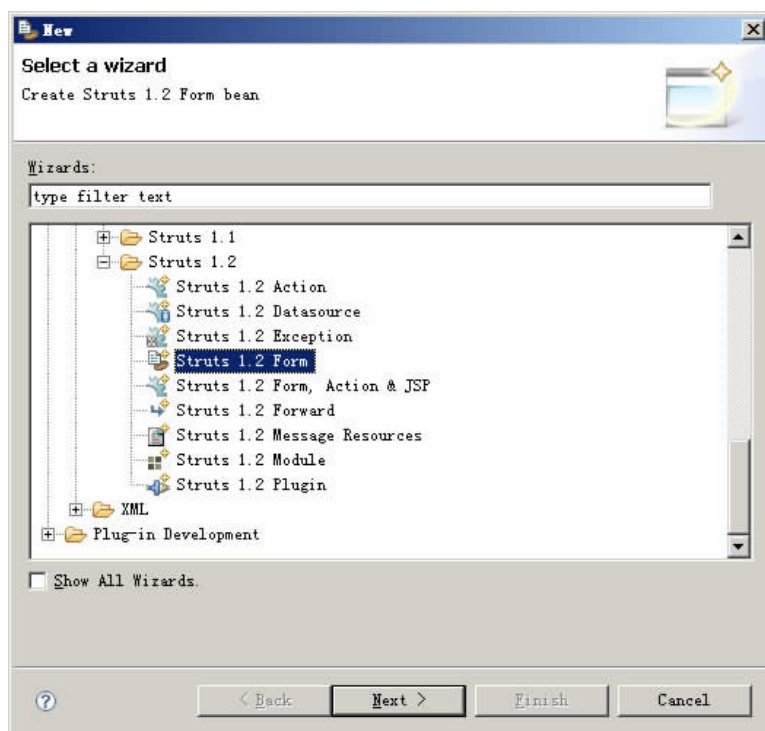


图 3-16 新建一个 ActionForm

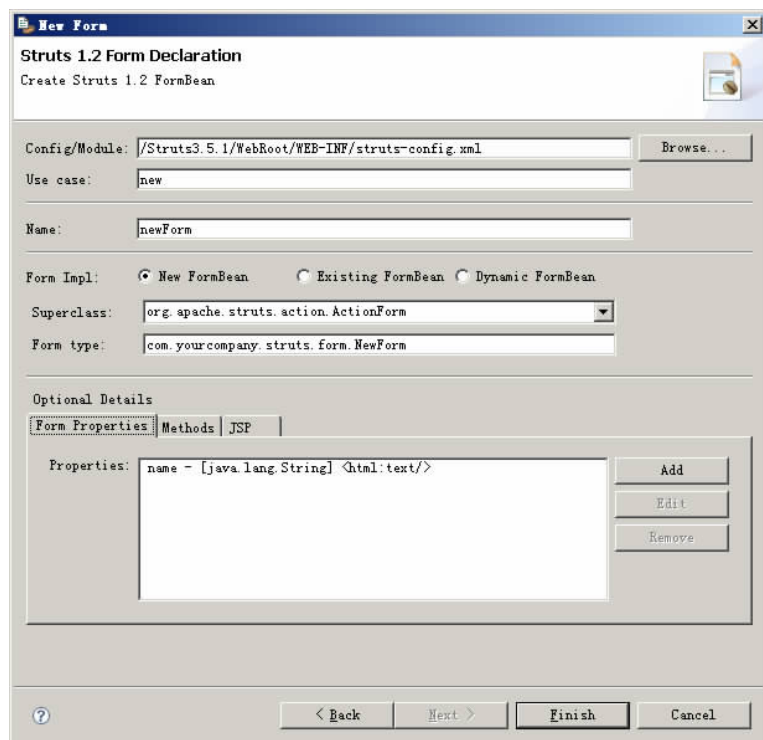


图 3-17 设置 ActionForm 类

设置完成后单击“Finish”按钮完成设置。

双击 new.jsp 文件，编辑<html:form>标记内容如下。

```
<html:form action="/old.do">
```

这里面 action 中的“/old.do”是以后步骤中在 struts-config.xml 配置的映射路径，也就是指向 IncludeAction 类的路径。

创建 ActionForm 类和 new.jsp 页面结束。

(2) 创建一个模拟的老 JSP 文件，old.jsp 内容如下。

```
<%@ page language="java" pageEncoding="ISO-8859-1"%>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html:html lang="true">

  <body>
<%
String name=request.getParameter("name");
out.println("this is old.jsp page"+"<br>");
out.println("name parameter value is =" +name+"<br>");
%>
  </body>
</html:html>
```

(3) 在 struts-config.xml 文件中配置 IncludeAction 类。

创建一个 IncludeAction 类和创建一个 Action 类一模一样，如图 3-18 所示。



图 3-18 创建一个 IncludeAction 类

单击“Next”按钮后，如图 3-19 所示。

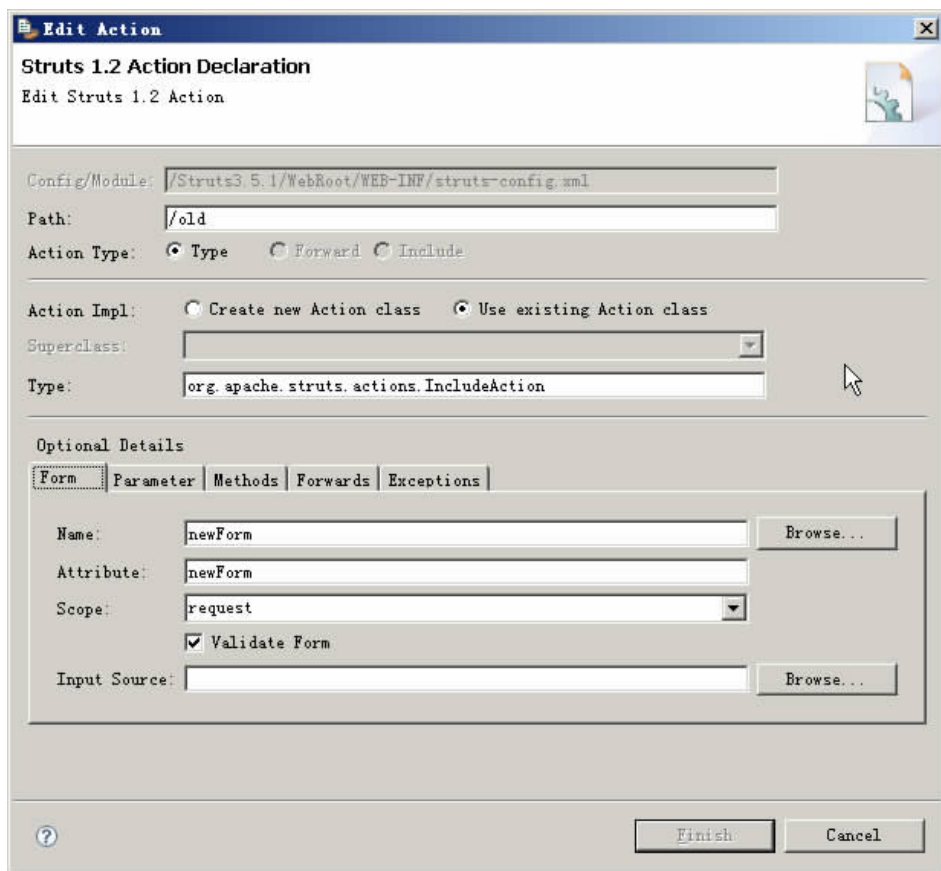


图 3-19 配置 IncludeAction

在如图 3-18 所示的界面中需要设置 5 项。

- 设置“Path”为“/old”，这个路径也就是前面<html:form>中的<html:form action="/old.do">路径。
- 设置“Action Impl”为“Use existing Action class。”
- 设置“Type”为“org.apache.struts.actions.IncludeAction。”
- 设置 Form 属性页中的 Name 属性。方法是单击 Name 右边的 Browse 按钮，进行自动设置。
- 最后一步，也是比较重要的，就是设置包含进哪个 JSP 页面，如图 3-20 所示。

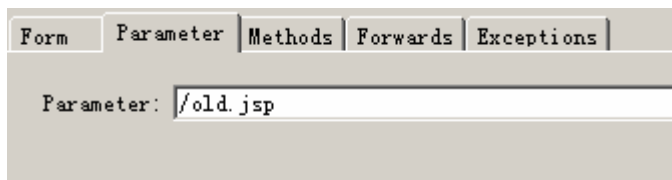


图 3-20 设置 Parameter 属性页



设置完成后单击“Finish”按钮，结束设置。

Struts-config.xml 文件内容如下。

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE struts-config PUBLIC "-//Apache Software Foundation//DTD Struts
Configuration 1.2//EN" "http://struts.apache.org/dtds
                                /struts-config_1_2.dtd">

<struts-config>
  <data-sources />
  <form-beans >
    <form-bean name="newForm"
              type="com.yourcompany.struts.form.NewForm" />

  </form-beans>

  <global-exceptions />
  <global-forwards />
  <action-mappings >
    <action
      attribute="newForm"
      name="newForm"
      parameter="/old.jsp"
      path="/old"
      scope="request"
      type="org.apache.struts.actions.IncludeAction" />

  </action-mappings>

  <message-resources
parameter="com.yourcompany.struts.ApplicationResources" />
</struts-config>
```

部署项目，启动 Tomcat 服务。

打开浏览器后，输入相应的 URL 地址，显示如图 3-21 所示。



图 3-21 输入 gaohongyan

单击“Submit”按钮后出现如图 3-22 所示的界面。



图 3-22 转到/old.do

在这个例子中将以前老的 old.jsp 文件通过 Struts 中的控制器集成到新的 Struts 项目中了，这样通过控制器来中转可以有效地进行集中控制及维护。

在这个例子中使用了 Form 表单的提交，也可以使用超级链接的提交，在超级链接的 URL 中写入?name=gaohongyan 类似的参数，例如：

```
<html:link action="/old.do?name=ghyghost">html:link to  
old.jsp</html:link>
```

### 3.5.2 使用 IncludeAction 包含进 Servlet 组件

新建一个项目名称为 Struts3.5.2 的项目。添加必要的 Struts 框架文件。

(1) 新建一个 Servlet 类。

用鼠标右键单击 src 节点，在弹出的菜单中选择“New”“Package”命令，如图 3-23 所示。

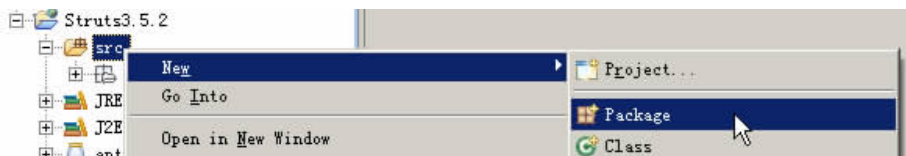


图 3-23 新建一个 Package 包

输入包名“myservlet”，如图 3-24 所示。

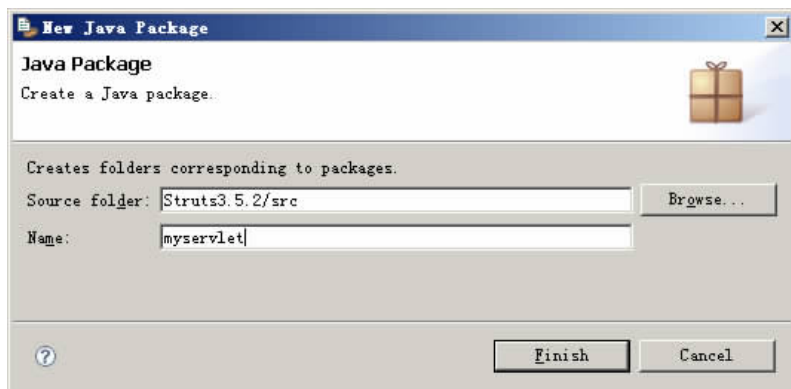


图 3-24 输入包名

单击“Finish”按钮，再在包中新建一个 Servlet，用鼠标右键单击“myservlet”，在弹出的菜单中选择“New”“Servlet”命令，如图 3-25 所示。

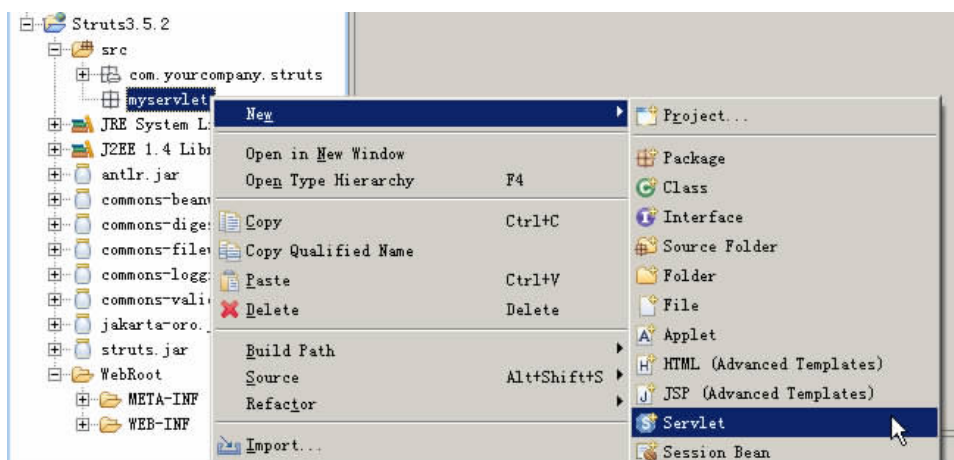


图 3-25 新建一个 Servlet

出现如图 3-26 所示的窗口。

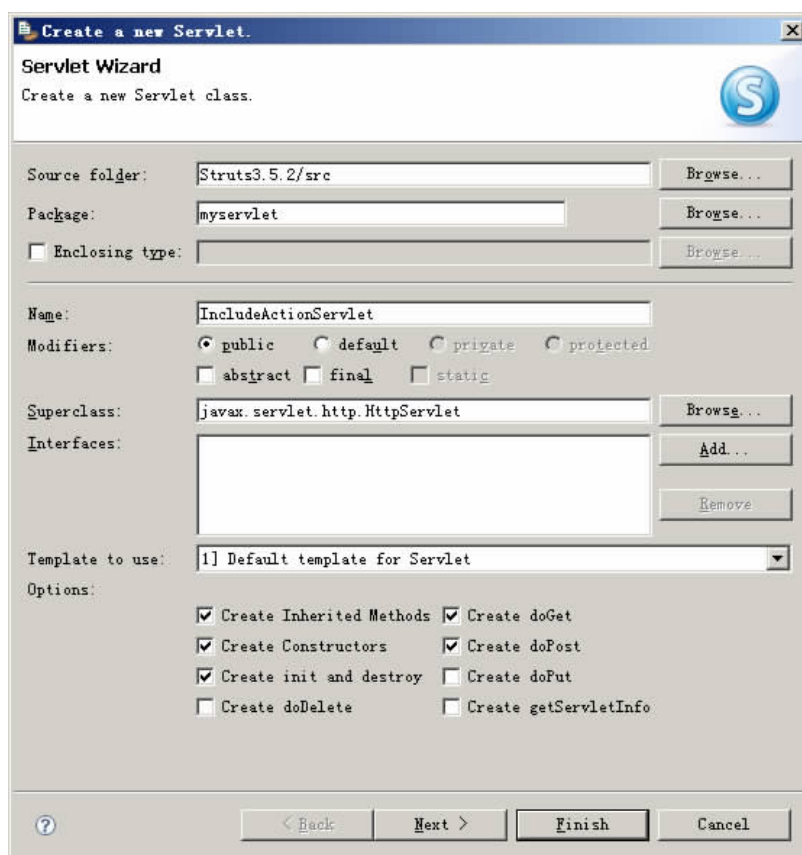


图 3-26 配置 Servlet

只需要输入 Servlet 的 “Name”，即类名 IncludeActionServlet。单击 “Finish” 按钮，结束设置。

(2) 编辑 Servlet 文件内容如下。

```
package myservlet;

import java.io.IOException;
import java.io.PrintWriter;

import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

public class IncludeActionServlet extends HttpServlet {

    public IncludeActionServlet() {
        super();
    }

    public void destroy() {
        super.destroy(); // Just puts "destroy" string in log
        // Put your code here
    }

    public void doGet(HttpServletRequest request,
                      HttpServletResponse response)
        throws ServletException, IOException {
        doPost(request, response);
    }

    public void doPost(HttpServletRequest request,
                      HttpServletResponse response)
        throws ServletException, IOException {

        response.setContentType("text/html");
        PrintWriter out = response.getWriter();
        out
            .println("<!DOCTYPE HTML PUBLIC \"-//W3C//DTD HTML 4.01\n\" Transitional//EN\">");
        out.println("<HTML>");
        out.println("  <HEAD><TITLE>A Servlet</TITLE></HEAD>");
        out.println("  <BODY>");
        out.print("name value is=" + request.getParameter("name"));
        out.println("  </BODY>");
        out.println("</HTML>");
        out.flush();
        out.close();
    }
}
```

```
public void init() throws ServletException {  
    // Put your code here  
}  
}
```

在 Servlet 文件中只是通过 doPost 方法取得 name 参数的值，并打印出来。

(3) 创建一个提交数据的 JSP 页面和相对应的 Struts 的 ActionForm，如图 3-27 所示。

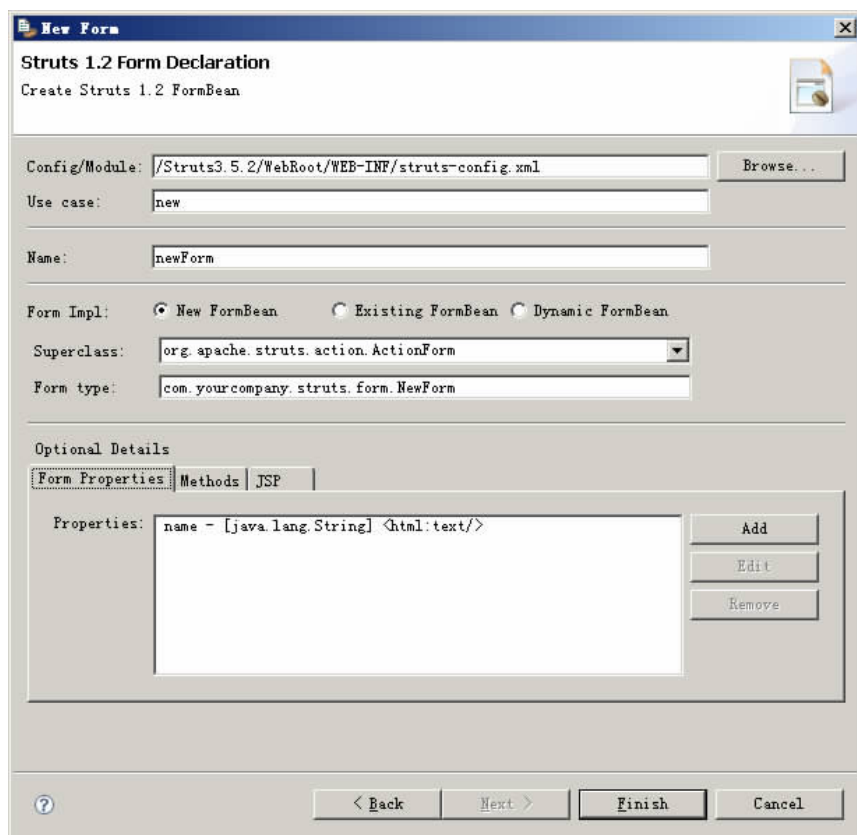


图 3-27 创建 JSP 及 ActionForm

设置“JSP”属性页如图 3-28 所示。



图 3-28 设置“JSP”属性页

(4) 编辑 JSP 文件内容如下。

```
<html:form action="/old.do">
```

(5) 创建一个 IncludeAction 类。

在 struts-config.xml 文件的 design 模式下，在空白处单击鼠标右键，新建一个 Action，如图 3-29 所示。

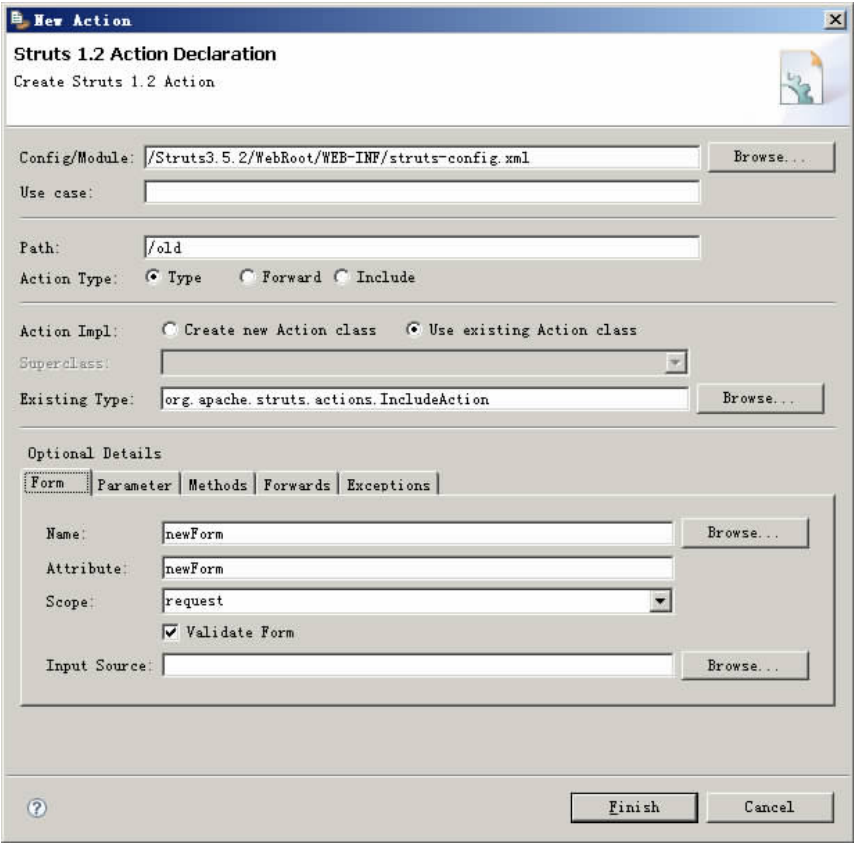


图 3-29 新建一个 IncludeAction

设置 “Parameter” 属性页内容如图 3-30 所示。

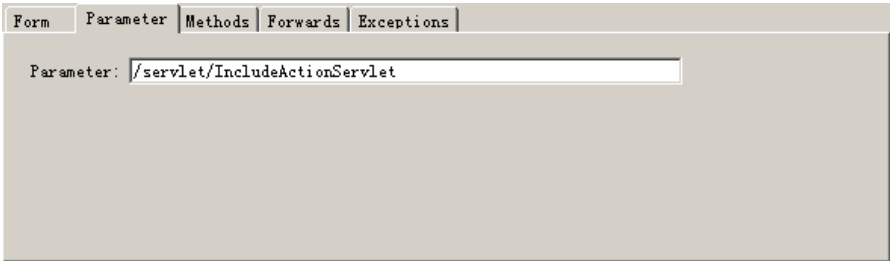


图 3-30 设置 “Parameter” 属性页

里面的路径就是要提交的目的 URL，也就是模拟老项目中 Servlet 所在的路径位置。

那有读者会问，为什么这样添加？因为添加这样的内容是要对照 web.xml 文件的路径映射关系的。在这个项目中 web.xml 文件内容如下。

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns="http://java.sun.com/xml/ns/j2ee"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="2.4"
  xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee
http://java.sun.com/xml/ns/j2ee/web-app_2_4.xsd">
  <servlet>
    <servlet-name>action</servlet-name>
    <servlet-class>org.apache.struts.action.ActionServlet
      </servlet-class>
    <init-param>
      <param-name>config</param-name>
      <param-value>/Web-INF/struts-config.xml</param-value>
    </init-param>
    <init-param>
      <param-name>debug</param-name>
      <param-value>3</param-value>
    </init-param>
    <init-param>
      <param-name>detail</param-name>
      <param-value>3</param-value>
    </init-param>
    <load-on-startup>0</load-on-startup>
  </servlet>
  <servlet>
    <description>This is the description of my J2EE component</description>
    <display-name>This is the display name of my J2EE
      component</display-name>
    <servlet-name>IncludeActionServlet</servlet-name>
    <servlet-class>myservlet.IncludeActionServlet</servlet-class>
  </servlet>

  <servlet-mapping>
    <servlet-name>action</servlet-name>
    <url-pattern>*.do</url-pattern>
  </servlet-mapping>
  <servlet-mapping>
    <servlet-name>IncludeActionServlet</servlet-name>
    <url-pattern>/servlet/IncludeActionServlet</url-pattern>
  </servlet-mapping>
</web-app>
```

程序<url-pattern>/servlet/IncludeActionServlet</url-pattern>就是我们要指向的 URL 路径。所以在上面的 Parameter 要这么写。

本示例中的 struts-config.xml 文件内容如下。

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE struts-config PUBLIC "-//Apache Software Foundation//DTD Struts
```

```
Configuration 1.2//EN" "http://struts.apache.org/dtds
/struts-config_1_2.dtd">

<struts-config>
  <data-sources />
  <form-beans >
    <form-bean name="newForm" type="com.yourcompany.struts.form.NewForm"
  />

  </form-beans>

  <global-exceptions />
  <global-forwards />
  <action-mappings >
    <action
      attribute="newForm"
      name="newForm"
      parameter="/servlet/IncludeActionServlet"
      path="/old"
      scope="request"
      type="org.apache.struts.actions.IncludeAction" />

  </action-mappings>

  <message-resources
parameter="com.yourcompany.struts.ApplicationResources" />
</struts-config>
```

单击“Finish”按钮，设置结束。

保存项目，部署项目。

输入 `http://localhost:8080/Struts3.5.2/new.jsp` 地址后，如图 3-31 所示。



图 3-31 显示 new.jsp 页面

单击“Submit”按钮后，出现如图 3-32 所示的窗口。

从这两个例子中可以看出，省略了使用 Action 类，直接使用 IncludeAction 类来将老项目中的 Web 模块有效地集成到新的项目中，这样可以省却人力物力，大大缩短项目开发周期。

IncludeAction 和 ForwardAction 都有相似的功能——换页面。





图 3-32 显示结果页面

但 IncludeAction 类不只是“换地址”，还有提交数据的功能，而 ForwardAction 类则不然，只有“换地址”，即转发的功能。

### 3.6 DispatchAction 将 Action 类变得更少

在此将相似的功能代码放到一个 Action 中去。

在前几章的项目实例中通常都是用 execute 方法来完成对业务逻辑的处理及页面的转发。通常在一个 Action 中只能完成一种业务逻辑的操作。如果要完成多个业务逻辑（比如添加、删除等）功能相近的业务逻辑就没有办法了吗？答案是否定的，可以通过在页面中定义一个隐藏变量，在不同的页面要求处理不同的业务逻辑的时候可以赋予这个变量不同的值，并在 execute 方法中通过对变量值的判断来完成不同的业务逻辑操作。

举例来说，首先在页面中定义一个隐藏变量。

```
<html:hidden property="operAt" />
```

然后定义一个 JavaScript 函数，可以通过单击提交按钮，在函数体里面修改它的值。

```
<SCRIPT>
function set(key) {
with(document.forms[0]){
operAt.value=key;
}
}
</SCRIPT>
```

当单击提交按钮时便触发该事件，修改变量的值。

```
<html:submit onclick="set('save');">SAVE</html:submit>
```

在后台 execute 又如何处理相关逻辑呢？如下所示。

```
String operAt = myForm.getOperAt();
if (operAt.equals("create")) { ...
if (operAt.equals("save")) { ...
```

很简单吧！虽然说这样做可以实现多个业务逻辑在同一个 Action 中实现，可是带来的代价便是代码的冗长，不易理解。比如在做一个论坛时，经常会对文章进行新建文章、编辑文章、删除文章、隐藏文章等操作，这时如果按照以前学习过的 Struts 的方法，就不得

不分别在每个功能上都要建立一个 Action 类来进行功能化的实现。这样不仅使 Action 类在项目结构中变得越来越多，而且在调试、维护项目上也有不便之处，这里 Struts 提供了一个非常方便的类，即 DispatchAction 类来处理这个问题。

要使用 DispatchAction 必须让 Action 类继承 DispatchAction 类。DispatchAction 类是一个抽象类，它实现了父类（Action）的 execute()方法，所以它的子类就不用实现这个方法了，只需要专注于实际操作的方法。

一个 DispatchAction 类的实例

（1）新建一个 1.JSP 文件，文件内容如下。

```
<%@ page language="java" pageEncoding="ISO-8859-1"%>

<%@ taglib uri="http://struts.apache.org/tags-bean" prefix="bean"%>
<%@ taglib uri="http://struts.apache.org/tags-html" prefix="html"%>
<%@ taglib uri="http://struts.apache.org/tags-logic" prefix="logic"%>
<%@ taglib uri="http://struts.apache.org/tags-tiles" prefix="tiles"%>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html:html lang="true">

<body>
    <html:link page="/iWantDo.do?method=newdb">new</html:link>
    <html:link page="/iWantDo.do?method=deletedb">delete</html:link>
    <html:link page="/iWantDo.do?method=updatedb">update</html:link>
    <html:link page="/iWantDo.do?method=hidedb">hide</html:link>
</body>
</html:html>
```

在程序代码中：

```
<html:link page="/iWantDo.do?method=newdb">new</html:link>
```

method 是参数，这个参数的值代表了要执行相对应的方法，这里执行的是 newdb 方法。这些方法在同一个 Action 类中。

（2）这一步也是最简单的一步，就是要创建一个 DispatchAction 类。

新建一个 Action 类，如图 3-33 所示。

在这里要设置如下 3 项。

- Use case：设置路径映射。
- 设置 Superclass 为 org.apache.struts.actions.DispatchAction。
- 设置 Parameter 属性页中的内容。

（3）新建一个 ActionForward。

```
<global-forwards >
    <forward name="toForward" path="/1.jsp" />
</global-forwards>
```

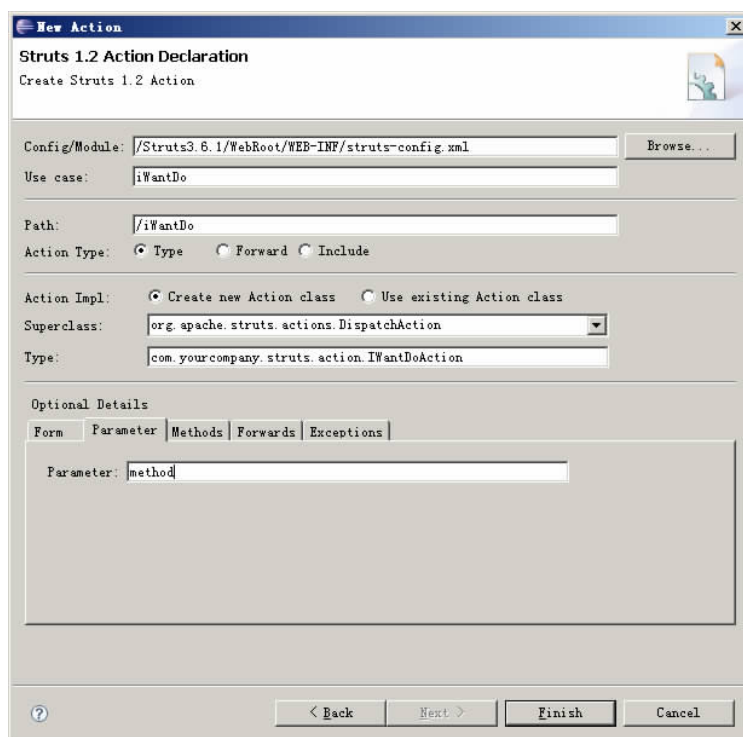


图 3-33 配置 DispatchAction 属性

(4) 更改 IWantDoAction 内容如下。

```
package com.yourcompany.struts.action;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import org.apache.struts.action.ActionForm;
import org.apache.struts.action.ActionForward;
import org.apache.struts.action.ActionMapping;
import org.apache.struts.actions.DispatchAction;

public class IWantDoAction extends DispatchAction {
    public ActionForward newdb(ActionMapping mapping, ActionForm form,
        HttpServletRequest request, HttpServletResponse response) {
        System.out.println("call newdb method");
        return mapping.findForward("toForward");
    }
    public ActionForward deletedb(ActionMapping mapping, ActionForm form,
        HttpServletRequest request, HttpServletResponse response) {
        System.out.println("call deletedb method");
        return mapping.findForward("toForward");
    }
    public ActionForward updatedb(ActionMapping mapping, ActionForm form,
        HttpServletRequest request, HttpServletResponse response) {
        System.out.println("call updatedb method");
    }
}
```

```

        return mapping.findForward("toForward");
    }
    public ActionForward hidedb(ActionMapping mapping, ActionForm form,
        HttpServletRequest request, HttpServletResponse response) {
        System.out.println("call hidedb method");
        return mapping.findForward("toForward");
    }
}

```

这里面的方法名称和

```

<html:link page="/iWantDo.do?method=newdb">new</html:link>
<html:link page="/iWantDo.do?method=deletedb">delete</html:link>
<html:link page="/iWantDo.do?method=updatedb">update</html:link>
<html:link page="/iWantDo.do?method=hidedb">hide</html:link>

```

相互对应，即单击其中的一个链接后就执行指定 Action 中的方法。



图 3-34 效果图

(5) 部署应用程序并运行，效果如图 3-34 所示。

超级链接 new 的 URL 地址为 `http://localhost:8080/Struts3.6.1/iWantDo.do?method=newdb`。即执行

```
public class IWantDoAction extends DispatchAction
```

类中的

```

    public ActionForward newdb(ActionMapping mapping, ActionForm form,
        HttpServletRequest request, HttpServletResponse response) {
        System.out.println("call newdb method");
        return mapping.findForward("toForward");
    }
}

```

方法。

程序运行结果可以在 Eclipse 的 Console 控制台中输出：

```
call newdb method
```

当然也可以在 newdb() 方法中输入一些 JDBC 的逻辑代码，这就取决于应用的需求了。

如果在配置文件中已经设定了 Parameter，而请求的时候没有加此参数，则 DispatchAction 类会调用 unspecified() 方法，此方法是 Protected，也就是说，程序员可以在自己类的里面覆盖 (override) 这个方法，也就成了默认方法。

## 3.7 LookupDispatchAction 实现一个表单包含多个提交按钮

本小节一起讨论 LookupDispatchAction 类的使用。

从类的名称上看, LookupDispatchAction 大概是 DispatchAction 的子类, 而且这种猜测也是十分正确的。它们从功能上有许多相似的地方。

通常它主要应用于在一个表单中有多个提交按钮而这些按钮又有一个共同的名字的情况, 而这些按钮的名字要和具体的 Action Mapping 中的 Parameter 的值对应(关于这个知识点做了实例就会知道)。这个 Action 类可以将据将类似功能的 Action 合并到一起。它与 DispatchAction 类的作用差不多, 唯一不同的是: 这个类通过资源包中的 key 作为请求参数来进行对方法的映射, 而 DispatchAction 是通过请求参数来选择方法的, 在 LookupDispatchAction 的设计时, SUBMIT 的 value 与方法名必须一致。

如果使用这个类, SUBMIT 的 value 是资源包中 key 所对应的值。

一个 LookupDispatchAction 类的使用实例

(1) 创建一个有一个提交按钮的 Form 表单。

新建一个 Struts 1.2 Form, 如图 3-35 所示。

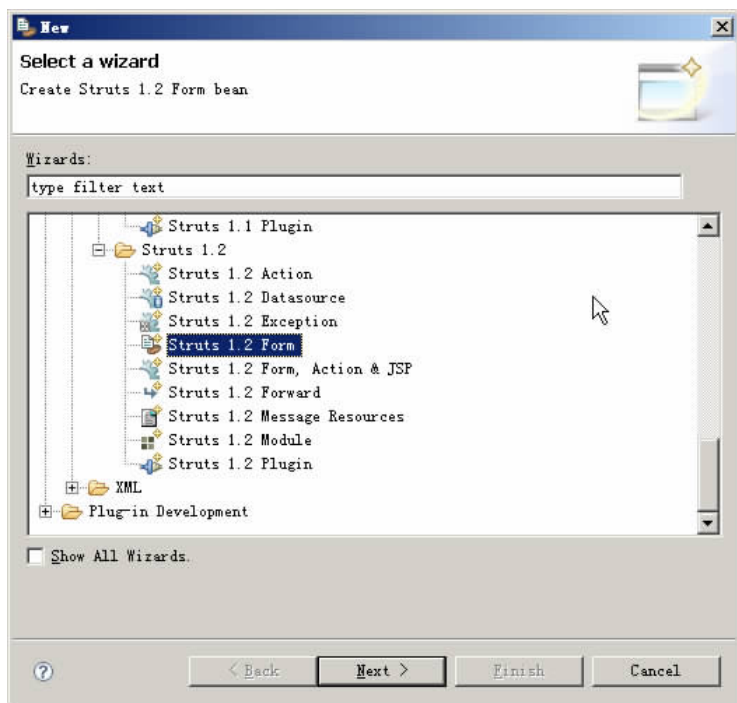


图 3-35 创建一个 Struts1.2 Form

单击“Next”按钮后，出现如图 3-36 所示的窗口，在这个窗口中需要设置 4 项参数。

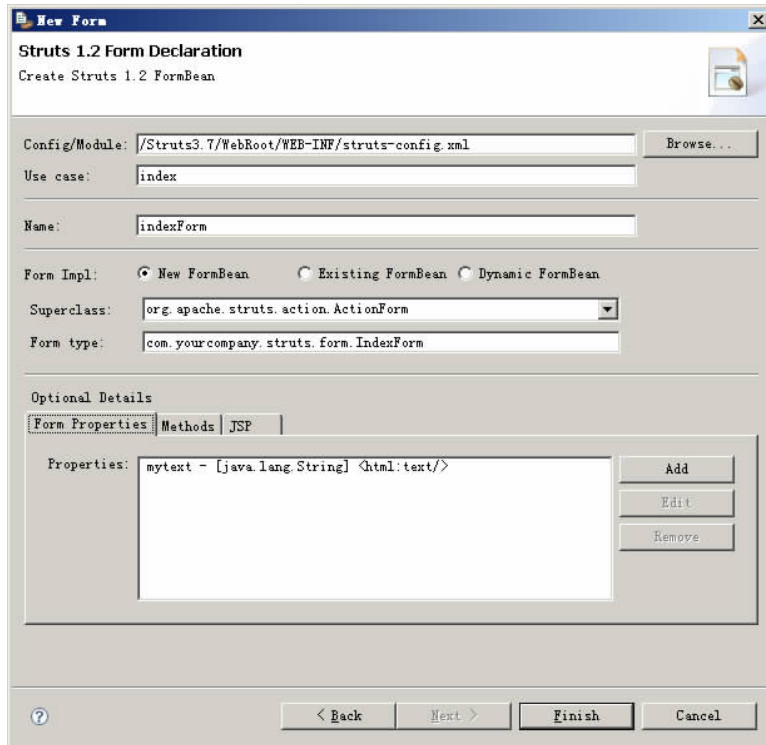


图 3-36 设置 Form

- Use case。
- Superclass。
- Form Properties 属性页。
- JSP 属性页。

设置内容如图 3-36 和图 3-37 所示。



图 3-37 设置 JSP 属性页

(2) 加入一个提交按钮，更改 JSP 文件内容。

双击 index.jsp 文件，添加一个提交按钮，更改后的 JSP 文件内容如下。

```
<%@ page language="java" pageEncoding="ISO-8859-1"%>
<%@ taglib uri="http://jakarta.apache.org/struts/tags-bean"
```

```
prefix="bean"%>
<%@ taglib uri="http://jakarta.apache.org/struts/tags-html"
prefix="html"%>
<html>
  <body>
    <html:form action="/myLookupDispatchAction.do">
      mytext<html:text property="mytext" />
      <html:errors property="mytext" />
      <br />
      update<html:submit property="action">
        <bean:message key="update_button" />
      </html:submit>
      delete<html:submit property="action">
        <bean:message key="delete_button" />
      </html:submit>
    </html:form>
  </body>
</html>
```

从程序代码中可以看出，JSP 页面有 2 个提交按钮，在本例中，提交按钮的名称通过 `<bean:message key="delete_button" />` 标签来制定，这个名称来源于资源文件。

(3) 添加两个 JSP 文件，显示 Forward 后的内容。

update.jsp 内容如下。

```
<%@ page language="java" import="java.util.*" pageEncoding="ISO-8859-1"%>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <body>
update db!~
  </body>
</html>
```

delete.jsp 内容如下。

```
<%@ page language="java" import="java.util.*" pageEncoding="ISO-8859-1"%>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <body>
delete db!~
  </body>
</html>
```

(4) 设置资源文件中的内容。

编辑资源文件内容如图 3-38 所示。

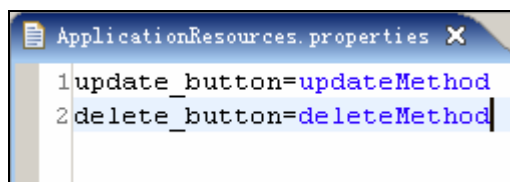


图 3-38 编辑资源文件

资源文件为什么要这么设计呢？为什么必须写这些内容呢？请看下面的代码：

```
update<html:submit property="action">
    <bean:message key="update_button" />
</html:submit />
delete<html:submit property="action">
    <bean:message key="delete_button" />
</html:submit />
```

上面的代码是 index.jsp 文件中的内容，这里面的 bean:message 中的 key 必须和资源文件中的 key 对应（这也许是死记硬背的方法）。

（5）设计 LookupDispatchAction 类。

新建一个 Action，如图 3-39 所示。

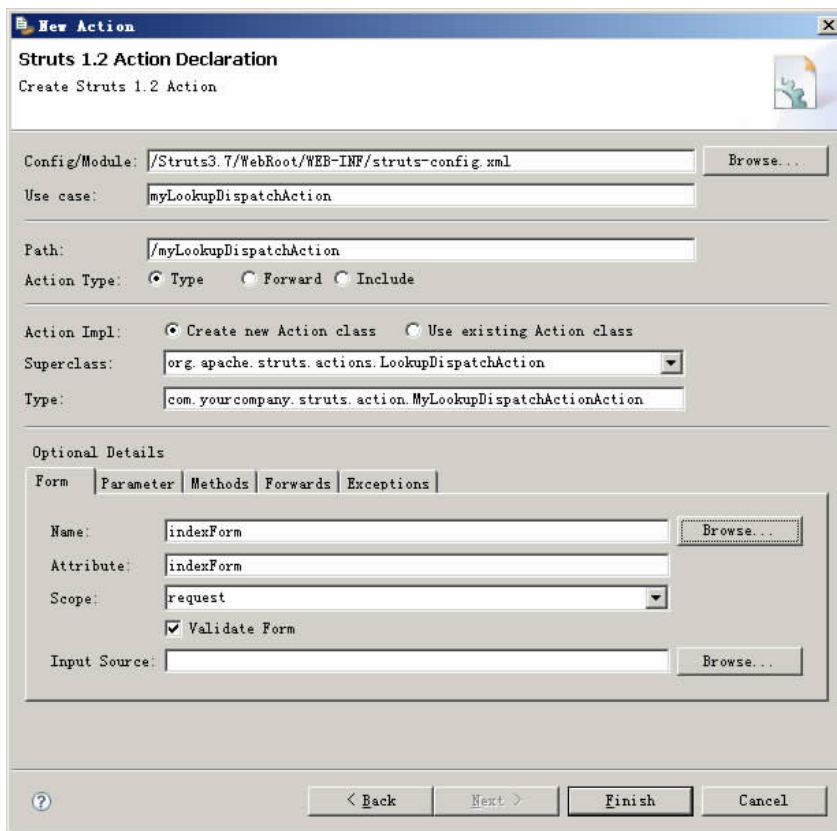


图 3-39 设计 LookupDispatchAction 类

在这里需要设置 4 项。

- Use case。
- Superclass。
- Parameter 属性页中的 Parameter 中的参数值 Action。



### ■ 设置 Form 属性页中的 ActionForm。

单击 “ Finish ” 按钮结束设置。

struts-config.xml 文件内容如下。

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE struts-config PUBLIC "-//Apache Software Foundation//DTD Struts
Configuration 1.2//EN" "http://struts.apache.org
/dtds/struts-config_1_2.dtd">

<struts-config>
  <data-sources />
  <form-beans >
    <form-bean name="indexForm"
               type="com.yourcompany.struts.form.IndexForm" />
  </form-beans>

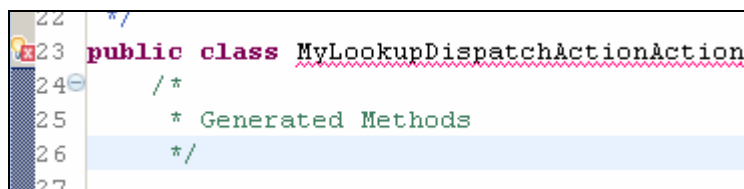
  <global-exceptions />
  <global-forwards >
    <forward name="update" path="/update.jsp" />
    <forward name="delete" path="/delete.jsp" />
  </global-forwards>

  <action-mappings >
    <action
      attribute="indexForm"
      name="indexForm"
      parameter="action"
      path="/myLookupDispatchAction"
      scope="request"
      type="com.yourcompany.struts.action.MyLookupDispatchActionAction"
    />
  </action-mappings>

  <message-resources
    parameter="com.yourcompany.struts.ApplicationResources" />
</struts-config>
```

### (6) 编辑 MyLookupDispatchActionAction.java 文件。

在上步单击 “ Finish ” 按钮后出现 MyLookupDispatchActionAction.java 的文件其实是有错误的，这在 Eclipse 中已经标注了，如图 3-40 所示。



```
22  */
23  public class MyLookupDispatchActionAction
24    /*
25     * Generated Methods
26     */
27
```

图 3-40 出错的文件

在左边出现一个错误标记，说明文件有错。出错的原因其实很简单，就是因为 MyLookupDispatchAction 类没有完全实现父类 LookupDispatchAction 的方法。

解决办法是：单击左边错误标记，出现一个提示框，选择第一个命令，实现所有方法，如图 3-41 所示。

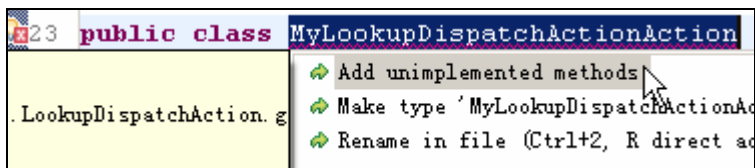


图 3-41 实现所有方法

程序自动添加了：

```
protected Map getKeyMethodMap() {  
    // TODO Auto-generated method stub  
    return null;  
}
```

在 getKeyMethodMap 方法中还要完善一下代码，添加相应的语句。

在这个类中还要删除方法：

```
public ActionForward execute(ActionMapping mapping, ActionForm form,  
    HttpServletRequest request, HttpServletResponse response) {  
    // TODO Auto-generated method stub  
    return null;  
}
```

完整正确的 Java 文件程序如下。

```
package com.yourcompany.struts.action;  
  
import java.util.Hashtable;  
import java.util.Map;  
  
import javax.servlet.http.HttpServletRequest;  
import javax.servlet.http.HttpServletResponse;  
  
import org.apache.struts.action.ActionForm;  
import org.apache.struts.action.ActionForward;  
import org.apache.struts.action.ActionMapping;  
import org.apache.struts.actions.LookupDispatchAction;  
  
public class MyLookupDispatchActionAction extends LookupDispatchAction {  
  
    public ActionForward updateMethod(ActionMapping mapping,  
        ActionForm form, HttpServletRequest request,  
        HttpServletResponse response) {  
  
    }  
}
```

```

        System.out.println("call updateMethod methos");
        return mapping.findForward("update");
    }

    public ActionForward deleteMethod(ActionMapping mapping,
        ActionForm form,HttpServletRequest request,
        HttpServletResponse response) {
        System.out.println("call deleteMethod methos");
        return mapping.findForward("delete");
    }

    protected Map getKeyMethodMap() {
        Map map=new Hashtable();
        map.put("update_button", "updateMethod");
        map.put("delete_button", "deleteMethod");
        return map;
    }
}

```

程序代码：

```

protected Map getKeyMethodMap() {
    Map map=new Hashtable();
    map.put("update_button", "updateMethod");
    map.put("delete_button", "deleteMethod");
    return map;
}

```

的功能是在 Action 类中通过取得前端相应提交按钮的名称来对应执行 map 对象 value 值中的方法，比如 updateMethod 或 deleteMethod 方法。

(7) 创建两个 Forward 对象。

```

<global-forwards>
  <forward name="update" path="/update.jsp" />
  <forward name="delete" path="/delete.jsp" />
</global-forwards>

```

(8) 部署并运行程序。

单击两个提交按钮，执行不同的功能。

在通常情况下，使用 DispatchAction 类的情况多一些。

### 3.8 用 SwitchAction 切换不同的 Struts 模块

在大型的项目开发中经常将大的软件分割成几个小模块，当每个功能模块开发完毕后要做的第一件事就是整合，在整合的过程中，模块与模块之间的切换是非常重要的操作。

但上面的开发方式笔者感觉在 Win 32 中开发比较合适，基于对 IDE 的操作及 Web 开发中 Struts 开发的特性，还是先做“总模块”再做“分模块”，这样比较顺手，而且合乎开

发流程逻辑。

从一个模块转换至另一个模块有两种方法，一种方法是使用相对于 Context 的路径来进行 URL 链接，这种方法比较大的缺点是没有经过 Servlet 中心控制器。另外一种方法比较符合 Struts 的 MVC 规范，也就是使用 SwitchAction 类，这也是本节要学习的内容，它需要在请求中带两个参数，一个是 prefix 用来指定模块前缀名称，一个是 page 用来指定相对于模块的资源路径。

下面就来做一个 SwitchAction 类的实例。

### SwitchAction 类的实例

针对前面概述的知识，需要创建一个“总模块”，然后再创建“分模块”这样的顺序来进行开发。

在这个实例中，要执行以下几步。

- (1) 创建一个工程 Struts3.8.1。
- (2) 像以前添加的 Struts 框架文件版本一样，添加 1.2 版的框架支持文件。
- (3) 展开“WebRoot” “New” “Folder” 节点，依次创建两个文件夹：login 和 showuser，这两个文件夹就是分模块了，从英语词意上就能看出，一个是 login 登录模块，一个是显示用户信息的模块，如图 3-42 所示。

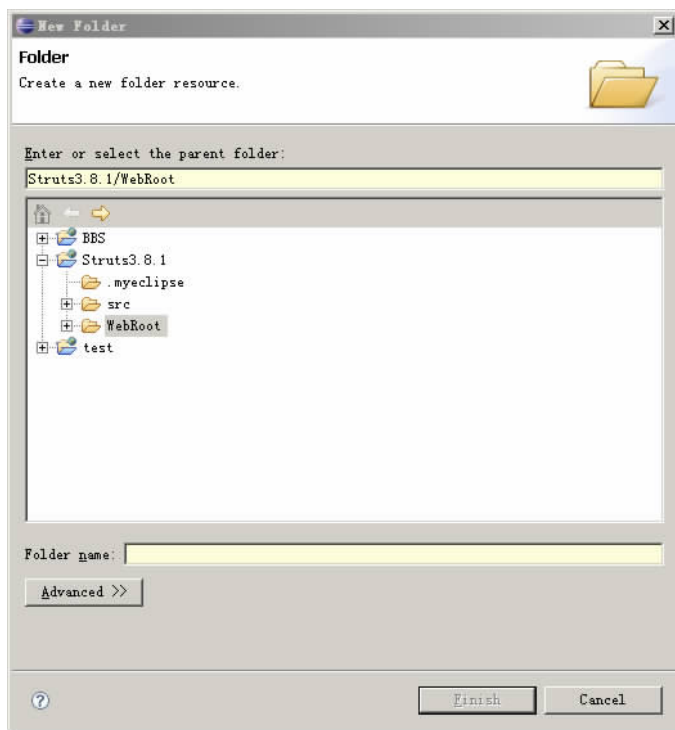


图 3-42 依次新建两个模块文件夹

(4) 创建每个模块的配置文件。其实在 Struts 中多模块开发也就是有多个 XML 配置文件，怎么样让多个配置文件能互相交互，也就是其目的所在。新建一个 login 模块，如图

3-43 所示。

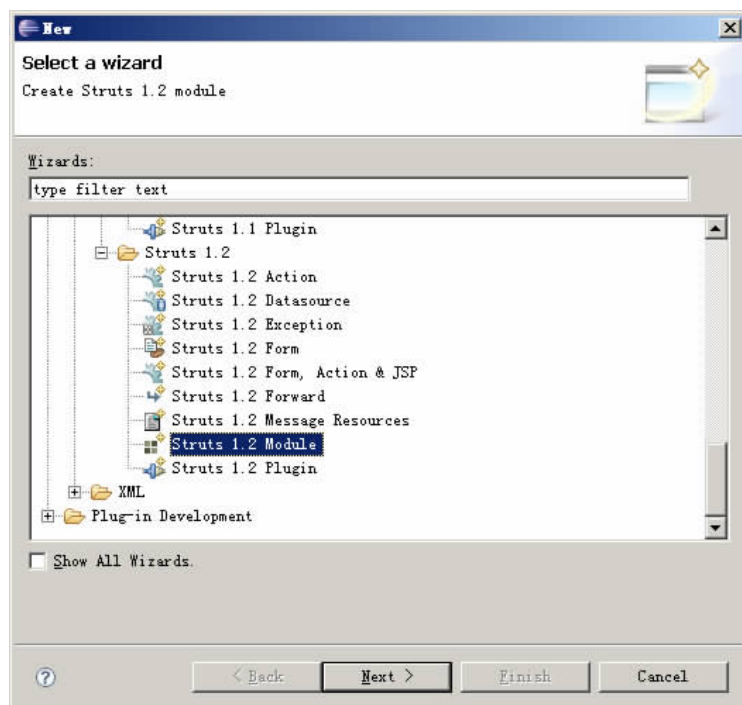


图 3-43 新建 login 模块

单击“Next”按钮后开始配置模块属性，如图 3-44 所示。

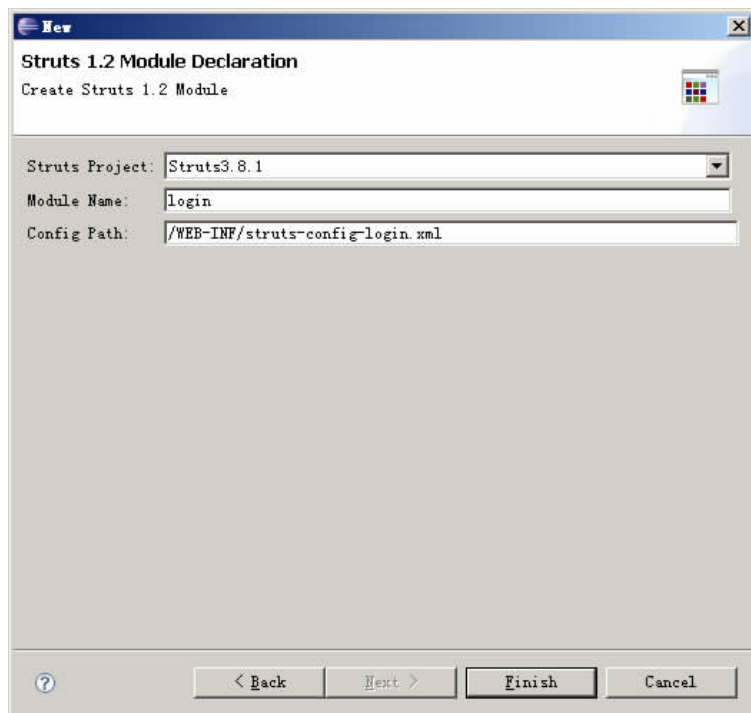


图 3-44 配置 login 模块属性

设置完成后单击“Finish”按钮应用设置。

(5) 重复第 4 步创建 showuser 模块。

(6) 创建两个模块后的 MyEclipse 目录结构如图 3-45 所示。

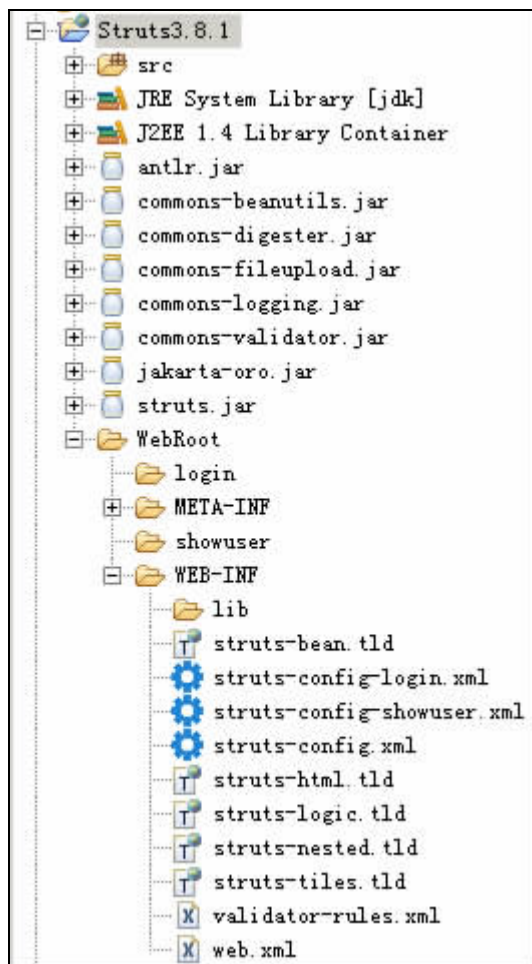


图 3-45 包含两个模块的目录结构

(7) 前 6 步的数据可以放到 CVS 工具中去了,这时可以针对每个程序员的不同开发,下载不同的模块,每个程序员只开发自己的 XML 配置文件,并不影响默认的 struts-config.xml 文件。

(8) 在 login 模块中添加 login.jsp 文件并更改文件内容和新建一个指向 login.jsp 的 ForwardAction,当然这个新建的 ForwardAction 一定要在 struts-config-login.xml 文件中配置。新建的 ForwardAction 配置如图 3-46 所示。

设置 Parameter 属性页内容,如图 3-47 所示。

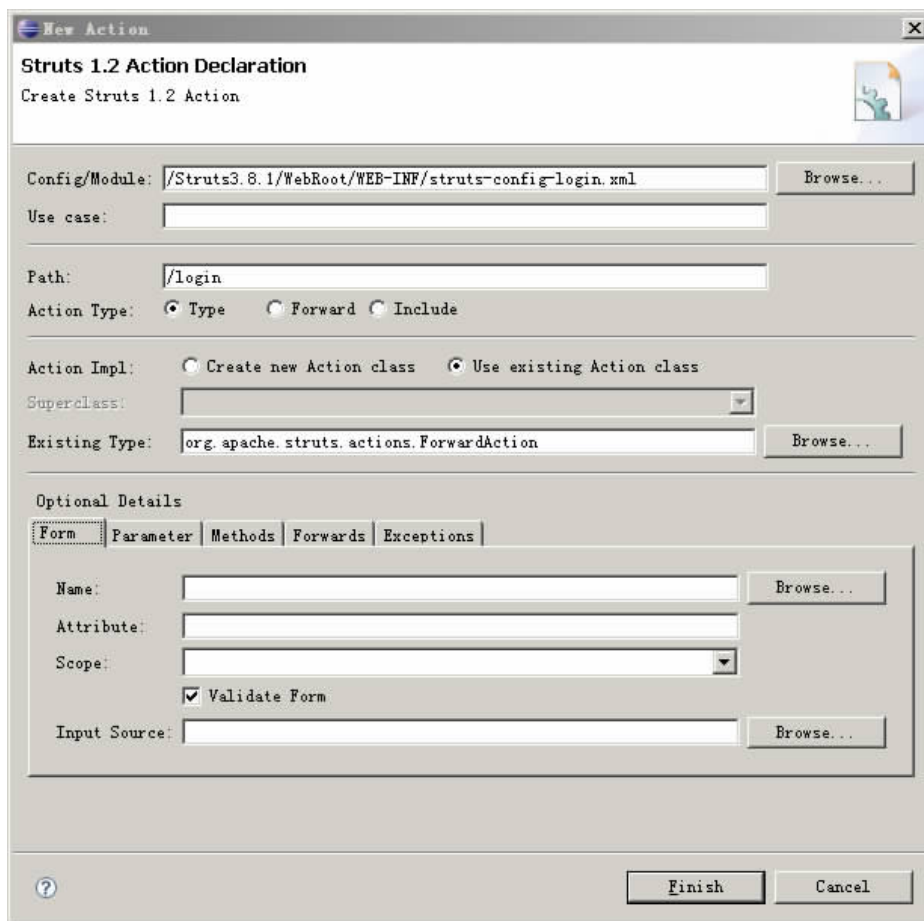


图 3-46 新建一个 login 的 ForwardAction

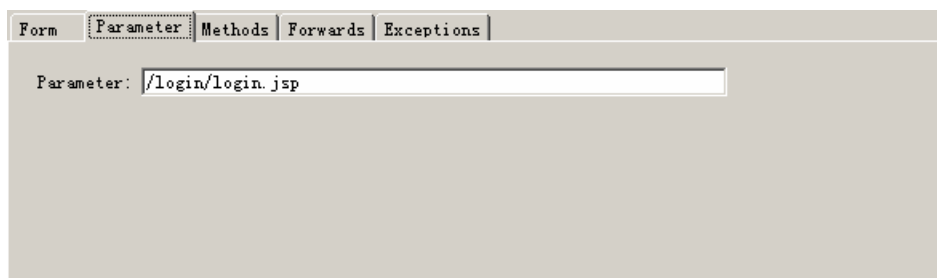


图 3-47 设置 login 的 Parameter 属性

struts-config-login.xml 文件代码如下。

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE struts-config PUBLIC "-//Apache Software Foundation
//DTD Struts Configuration 1.2//EN"
"http://struts.apache.org/dtds/struts-config_1_2.dtd">

<struts-config>
```

```
<data-sources />
<form-beans />
<global-exceptions />
<global-forwards >
</global-forwards>

<action-mappings >
  <action
    parameter="/login/login.jsp"
    path="/login"
    type="org.apache.struts.actions.ForwardAction" />
</action-mappings>

<controller />
</struts-config>
```

(9) 在 showuser 模块中添加 showuser.jsp 文件并更改文件内容和新建一个指向 showuser.jsp 的 ForwardAction。当然这个新建的 ForwardAction 一定要在 struts-config-showuser.xml 文件中配置。

struts-config-showuser.xml 文件内容如下所示。

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE struts-config PUBLIC "-//Apache Software Foundation
//DTD Struts Configuration 1.2//EN"
"http://struts.apache.org/dtds/struts-config_1_2.dtd">

<struts-config>
  <data-sources />
  <form-beans />
  <global-exceptions />
  <global-forwards >
  </global-forwards>

  <action-mappings >
    <action
      parameter="/showuser/showuser.jsp"
      path="/showuser"
      type="org.apache.struts.actions.ForwardAction" />
  </action-mappings>

  <controller />
</struts-config>
```

(10) 在 WebRoot 节点下新建的 JSP 文件 index.jsp 是项目的主界面。在这个 index.jsp 文件中有两个链接分别指向 login 模块的 login.jsp 和 showuser 模块的 showuser.jsp。

index.jsp 代码如下。



```
<%@ page language="java" pageEncoding="gb2312"%>

<%@ taglib uri="http://struts.apache.org/tags-bean" prefix="bean"%>
<%@ taglib uri="http://struts.apache.org/tags-html" prefix="html"%>
<%@ taglib uri="http://struts.apache.org/tags-logic" prefix="logic"%>
<%@ taglib uri="http://struts.apache.org/tags-tiles" prefix="tiles"%>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html:html lang="true">

<body>
  <html:link page="/toOtherModule.do?prefix=/login&";
            page=/login.do">指向 login 模块的/login.do</html:link>
  <html:link
    page="/toOtherModule.do?prefix=/showuser&";
    page=/showuser.do">指向 showuser 模块的/showuser.do</html:link>
</body>
</html:html>
```

(11) 在 struts-config.xml 默认配置文件中新建一个 Action，如图 3-48 所示。

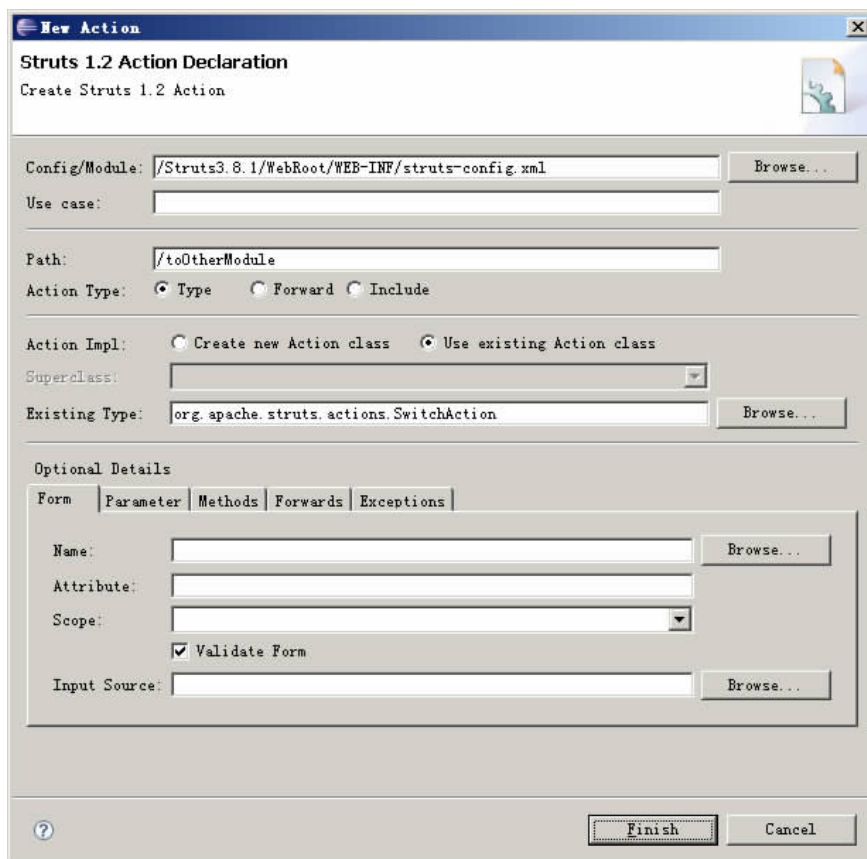


图 3-48 新建一个 SwitchAction 类

单击“Finish”按钮后，struts-config.xml 文件内容如下。

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE struts-config PUBLIC "-//Apache Software Foundation
//DTD Struts Configuration 1.2//EN"
"http://struts.apache.org/dtds/struts-config_1_2.dtd">

<struts-config>
  <data-sources />
  <form-beans />
  <global-exceptions />
  <global-forwards />
  <action-mappings >
    <action path="/toOtherModule"
              type="org.apache.struts.actions.SwitchAction" />
  </action-mappings>

  <message-resources
parameter="com.yourcompany.struts.ApplicationResources" />
</struts-config>
```

(12) 部署项目，启动服务并运行。

这时可以单击超级链接后转到相应模块的相应路径，并显示相应的 JSP 文件内容。如果想将 login 模块中的 login.jsp 文件转到默认模块，该如何实现呢？

(1) 在 struts-config-login.xml 文件添加如下代码。

```
<action path="/index" type="org.apache.struts.actions.SwitchAction" />
```

(2) 在默认的配置文件中 struts-config.xml 添加以下代码。

```
<action
  parameter="/index.jsp"
  path="/index"
  type="org.apache.struts.actions.ForwardAction" />
```

(3) 更改 login.jsp 文件内容如下。

```
<%@ page language="java" import="java.util.*" pageEncoding="gb2312"%>

<%@ taglib uri="http://struts.apache.org/tags-bean" prefix="bean"%>
<%@ taglib uri="http://struts.apache.org/tags-html" prefix="html"%>
<%@ taglib uri="http://struts.apache.org/tags-logic" prefix="logic"%>
<%@ taglib uri="http://struts.apache.org/tags-tiles" prefix="tiles"%>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>

  <body>
    this login.jsp page
    <html:link page="/index.do?prefix=&page=/index.do">
```

```
                                转到默认模块</html:link>
    </body>
</html>
```

在<html:link page="/index.do?prefix=&page=/index.do">代码中,前一个 index.do 代表子模块中的 SwitchAction 的路径,而后一个 index.do 代表默认模块中的 ForwardAction 的路径,这点需要留意。

部署项目并运行,单击超级链接就可以转到默认模块的 index.jsp 页面了。

在前面学习了一些 Struts 控制器及一些实例后,会发现那些 XML 文件的使用的确是非常重要的,下一章一起了解配置文件的大体结构。

# 第 4 章

## 中心配置文件 struts-config.xml

在前面的所有实例中，都使用了 struts-config.xml 文件来进行 Struts 程序执行流程的配置，虽然通过英文的翻译能大体了解在 struts-config.xml 文件中内容的功能，但还是不够细化，本章着重讲解 struts-config.xml 文件的结构，揭开这个中心控制文件的面纱。

Struts 的核心是 struts-config.xml 配置文件，在这个文件里描述了所有的 Struts 组件。在这里包括配置主要的组件及次要的组件，下面是 Struts 1.2 版的 struts-config.xml 文件简单结构。

### 4.1 Struts 1.2 版 struts-config.xml 文件结构

struts-config.xml 文件结构的程序如下。

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE struts-config PUBLIC "-//Apache Software Foundation
//DTD Struts Configuration 1.2//EN"
"http://struts.apache.org/dtds/struts-config_1_2.dtd">

<struts-config>
  <data-sources />
  <form-beans />
  <global-exceptions />
  <global-forwards />
  <action-mappings />
  <message-resources
parameter="com.yourcompany.struts.ApplicationResources" />
</struts-config>
```

以上各元素的顺序是非常重要的，struts-config.xml 配置文件必须按照这个顺序进行配置，否则在容器启动的时候就会出错，因为其 struts-config\_1\_2.dtd 文件详细的结构规定顺序如下。

```
<!ELEMENT struts-config (display-name?, description?, data-sources?,
form-beans?, global-exceptions?, global-forwards?, action-mappings?,
controller?, message-resources*, plug-in*)>
```

但在大多数的情况下，很少手动去更改这个配置文件，都是使用 MyEclipse 的图形界面去维护这个配置文件中的内容。

## 4.2 struts-config.xml 配置文件中的子元素

struts-config.xml 配置文件中的元素很多，下面只介绍 MyEclipse 自动生成的那些重要的配置元素。

### 4.2.1 <data-sources />子元素

定义 data-source 元素设置了具体的数据源，可以使用这个数据源来连接数据库，使用数据源来连接数据库有很多优点，比如：事务处理，连接效率，管理方便，J2EE 核心技术 JNDI 实现等。

但由于种种原因，Struts 官方推荐不要在 Struts 中使用 DataSource（数据源），而改用其他的 ORM 映射工具来实现，比如 Hibernate 等。

### 4.2.2 <form-bean />子元素

<form-bean />用来定义将要绑定到 Action 的 FormBean 的实例。  
语法如下。

```
<form-beans>
  <form-bean name="FormBean 在配置文件中唯一的名称"
    type="相对应的 ActionForm 类的名称" />
</form-beans>
```

实例

```
<form-bean name="indexForm"
  type="com.yourcompany.struts.form.IndexForm" />
```

### 4.2.3 <global-forwards />子元素

全局转发可以定义几个<forward/>子元素，Struts 首先会在<action-mappings>元素中查找对应的<forward>，若找不到，则到全局转发配置中找。

语法如下。

```
<global-forwards>
<forward name="唯一的转发的逻辑名称"
  path="要转发到的物理资源路径" />
</global-forwards>
```

除了 name 及 path 属性之外，还有一个 redirect 属性，如果 redirect 设为 true，则用 HttpServletResponse.sendRedirect()方法进行重定向的操作，否则用 RequestDispatcher.forward()方法来转发，默认为 false。

注意：如果为 true，则用 HttpServletResponse.sendRedirect()方法，此时存储在原来的 HttpServletRequest 中的值将会丢失。

#### 实例

```
<global-forwards >
  <forward name="toindex" path="/index.jsp" />
</global-forwards>
```

#### 4.2.4 <action-mappings>子元素

它可以定义几个<action />子元素，主要是定义 Action 实例到 ActionServlet 类中。语法如下。

```
<action-mappings>
<action path="请求的路径"
type="action 使用 Action 子类的名字"
name="与这个 action 关联的 FormBean 的名称">
<forward name="转发 1" path="转发 1 的相对上下文的物理路径"/>
<forward name="转发 2" path="转发 2 的相对上下文的物理路径"/>
</action>
</action-mappings>
```

<action/>属性及其描述信息如下。

- path：在浏览器的 URL 中输入的请求路径。
- type：连接到相对应的 Action 全称。
- name：与请求相关联的 Action Bean，在<form-bean/>中定义 name 名（可选的）。
- scope：指定 ActionForm Bean 的作用域（session 或 request），默认为 session（可选的）。
- input：当 Bean 发生错误时返回的页面（可选的）。
- className：指定一个调用这个 Action 类的 ActionMapping 类的全名。默认用 org.apache.struts.action.ActionMapping（可选的）。
- forward：指定处理相应请求转发时所对应的 JSP 页面（可选的）。
- include：如果没有 forward 的时候，它起 forward 的作用（可选的）。
- validate：若为 true，则会调用 ActionForm 的 validate()方法，否则不调用，默认为 true（可选的）。

#### 实例

```
<action-mappings>
<action path="/ghyghostlookupAction"
type="gaohongyan.LookupAction"
name="ghyghostLookupForm"
scope="request"
validate="true"
input="/index.jsp">
```

```
<forward name="success" path="/succ.jsp" />
<forward name="faliue" path="/false.jsp" />
</action>
</action-mappings>
```

#### 4.2.5 <message-resources>子元素

在 struts-config.xml 文件中用<message-resources />元素来定义消息资源。其语法如下。

```
<message-resources
    parameter="com.yourcompany.struts.ApplicationResources" />
```

<message-resources />元素属性及其描述信息如下。

- Parameter：给定资源文件全名。
- ClassName：定义处理消息资源的类名的全名。默认是 org.apache.struts.config.MessageResourcesConfig。
- Factory：定义 MessageResourcesFactory 类的全名。默认是 org.apache.struts.util.property.MessageResourcesFacotry。
- Key：定义绑定在这个资源包中的 ServletContext 的属性主键。默认值是 Action.MESSAGES\_KEY。
- Null：如果为 true，找不到消息 key 时，则返回 null，默认是 true。

实例（1）

```
<message-resources parameter="ghyghost.ApplicationResources" />
```

实例（2）

```
<message-resources
    parameter="ghyghostMessageResources"
    null="false" />

<message-resources
    key="other_RESOURCE_KEY"
    parameter="otherImageResources"
    null="false" />
```

#### 4.2.6 关于配置 strus-config.xml 文件

（1）Struts-config.xml 的每个部分的配置规范主要由该 XML 文件所使用的 DTD 文件决定。所使用的 DTD 版本在文件头部分，所以掌握好 DTD 是了解该配置文件的最好起点。如：

```
<!DOCTYPE struts-config PUBLIC "-//Apache Software Foundation
//DTD Struts Configuration 1.2//EN"
"http://struts.apache.org/dtds/struts-config_1_2.dtd">
```

不同的版本所定义的元素和属性不同。

(2) Struts-config.xml 文件中的参数是与次序相关的, 所以需要将子元素按正确的顺序放置。不过系统会根据 DTD 文件生成一个按顺序排列的模板。

(3) 一般主要包括以下几个部分。

- data-sources
- form-bean
- global-forwards
- action-mapping
- message-resource

(4) Struts-config.xml 文件 DTD 的一部分。

```
<!ELEMENT struts-config(data-sources?,form-beans?,  
global-exceptions?,global-forwards?,  
action-mappings?,controller?,  
messages-resources,plug-in)>
```

实际上, DTD 只有两种选项: ELEMENT 和 ATTLIST。

- ELEMENT 是用来描述 XML 文件的总体语法(元素的名字是什么、它的内部可以包含哪些标记)。
- ATTLIST 值定义了标记可包括的属性。

<!ELEMENT tag-name (subtags)>中的 subtags 是其他 ELEMENT 项的引用。

Subtag 后跟一个“?”表示父标记中可以包括一个或者零个这样的标记。

如果在标记名后面跟一个星号, 表示父标记中可以包括多个或零个这样的标记。

如果没有跟符号, 表示有且仅有一个标记可以放到父标记中, 此外, 标记的顺序是非常重要的, 标记出现的次序必须符合它们在列表中的定义。

属性 ATTLIST 可以以任意次序出现。有 4 个参数: 标记名、属性名、属性的类型、默认值。字符串“IMPLIED”意味着属性值不是必需的。“REQUIRED”意味着基于该 DTD 的 XML 无论在何种情况下都应该提供这个属性。

(5) 表单 Bean (form-bean)

form-beans 和 form-bean 标记被用来告知 Struts, ActionForm 类与哪个唯一标识相关联, 它还可以用来定义动态 form-DynaForm。

(6) Action 映射 (action-mappings)

Struts 使用 action 标记来将表单、action 以及转发连接在一起。它们被捆绑在 action-mappings 之内。

- 属性 path 被用于同 action 的请求相匹配, action 的路径应该没有任何后缀。如在表单中指定 action="/my/regist.do"。它将匹配 path 为/my/regist。
- 属性 forward 和 include 可以用来将控制权直接传递到新路径, 而不是直接处理 action。通过指定它的路径, 属性 input 允许将 action 属性重定向到用以输入表单值的表单。



- 属性 name 用于指定 FormBean 的名字。
- 属性 forward、exception 和 set-property 标记可以在 action 的内部局部使用，它们归属于所定义的 Action。Set-property 标记对于向 action 传递信息非常有用。例如，如果一个单独的 Action 类可被用来处理若干不同的表单，可以使用 set-property 标记来告诉 action 要处理的是哪一个表单。

(7) Struts-config.xml 与 Struts 其他元素的相互连接。

可以从 JSP 页面的角度开始考虑。当页面使用带有 Action 属性的<html:form>标记时，使用该属性从 config 文件中找到相应的<action>标记。这个标记接下来定义了表单类，它允许 JSP 页面使用<html:text>以及其他表单输入标记和表单 Bean 相关联；Action 类用来处理验证之后的结果。

#### 4.2.7 元素详解

```
<struts-config>
<!--
```

名称：data-sources。

描述：data-sources 元素定义了 Web App 所需要使用的数据源。

数量：最多 1 个。

子元素：data-source。

```
-->
<data-sources>
<!--
```

名称：data-source。

描述：data-source 元素定义了具体的数据源。

数量：任意多个。

属性：

(1) key：当需要配置多个数据源时，相当于数据源的名称，用来对数据源彼此间进行区别。

(2) type：可以使用的数据源实现的类，一般来自如下 4 个库。

- Poolman：开放源代码软件。
- Expresso：Jcorporate。
- JDBC Pool：开放源代码软件。
- DBCP：Jakarta 组织。

```
-->
<data-source key="firstOne"
  type="org.apache.commons.dbcp.BasicDataSource">
<!--
```

名称：set-property。

描述：用来设定数据源的属性。

属性：

(1) autoCommit：是否自动提交，可选值为 true/false。

(2) description：数据源描述。

(3) driverClass：数据源使用的类。

(4) maxCount：最大数据源连接数。

(5) minCount：最小数据源连接数。

(6) user：数据库用户。

(7) password：数据库密码。

(8) url：数据库 url。

```
-->
<set-property property="autoCommit" value="true"/>
<set-property property="description" value="Hello!"/>
<set-property property="driverClass" value="com.mysql.jdbc.Driver"/>
<set-property property="maxCount" value="10"/>
<set-property property="minCount" value="2"/>
<set-property property="user" value="root"/>
<set-property property="password" value=""/>
<set-property property="url"
  value="jdbc:mysql://localhost:3306/helloAdmin"/>
</data-source>
</data-sources>

<!--
```

名称：form-beans。

描述：用来配置多个 ActionForm Bean。

数量：最多 1 个。

子元素：form-bean。

```
-->
<form-beans>
<!--
```

名称：form-bean。

描述：用来配置 ActionForm Bean。

数量：任意多个。

子元素：form-property。

属性：

(1) className 指定与 form-bean 元素相对应的配置类，一般默认使用 org.apache.struts.config.FormBeanConfig，如果自定义，则必须继承 FormBeanConfig。

(2) name 必备属性。为当前 form-bean 制定一个全局唯一的标识符，使得在整个 Struts 框架内，可以通过该标识符来引用这个 ActionForm Bean。

(3) type：必备属性。指明实现当前 ActionForm Bean 的完整类名。

```
-->
<form-bean name="Hello" type="myPack.Hello">
<!--
```

名称：form-property。

描述：用来设定 ActionForm Bean 的属性。

数量：根据实际需求而定，例如 ActionForm Bean 对应的一个登录 Form 中有两个文本框，即 name 和 password，ActionForm Bean 中也有这两个字段，此处编写两个 form-property 来设定属性。

属性：

(1) className：指定与 form-property 相对应的配置类，默认是 org.apache.struts.config.FormPropertyConfig，如果自定义，则必须继承 FormPropertyConfig 类。

(2) name：所要设定的 ActionForm Bean 的属性名称。

(3) type：所要设定的 ActionForm Bean 的属性值的类。

(4) initial：当前属性的初值。

```
-->
<form-property name="name" type="java.lang.String"/>
<form-property name="number" type="java.lang.Integer" initial="18"/>
</form-bean>
</form-beans>

<!--
```

名称：global-exceptions。

描述：处理异常。

数量：最多一个。

子元素：exception。

```
-->

<global-exceptions>
  <!--
```

名称：exception。

描述：具体定义一个异常及其处理。

数量：任意多个。

属性：

(1) className 指定对应 exception 的配置类，默认为 org.apache.struts.config.ExceptionConfig。

- (2) handler：指定异常处理类，默认为 org.apache.struts.action.ExceptionHandler。
- (3) key：指定在 Resource Bundle 中描述该异常的消息 key。
- (4) path：指定当发生异常时，进行转发的路径。
- (5) scope：指定 ActionMessage 实例存放的范围，默认为 request，另外一个可选值是 session。
- (6) type：必须要有，指定所需要处理异常类的名字。
- (7) bundle：指定资源绑定。

```
-->  
  
<exception  
key="hello.error  
path="/error.jsp"  
scope="session"  
type="hello.HandleError"/>  
</global-exceptions>  
  
<!--
```

名称：global-forwards。

描述：定义全局转发。

数量：最多一个。

子元素：forward。

```
-->  
  
<global-forwards>  
<!--
```

名称：forward。

描述：定义一个具体的转发。

数量：任意多个。

属性：

(1) className：指定和 forward 元素对应的配置类，默认为 org.apache.struts.action.ActionForward。

(2) contextRelative：如果为 true，则指明使用当前上下文，路径以 “/” 开头，默认为 false。

(3) name：必须配有。指明转发路径的唯一标识符。

(4) path：必须配有。指明转发或者重定向的 URI。必须以 “/” 开头，具体配置要与 contextRelative 相应。

(5) redirect：为 true 时，执行重定向操作；否则执行请求转发。默认为 false。

```
-->  
<forward name="A" path="/a.jsp"/>  
<forward name="B" path="/hello/b.do"/>
```

```
</global-forwards>
```

```
<!--
```

名称：action-mappings。

描述：定义 action 集合。

数量：最多一个。

子元素：action。

```
-->
```

```
<action-mappings>
```

```
<!--
```

名称：action。

描述：定义了从特定的请求路径到相应的 Action 类的映射。

数量：任意多个。

子元素：exception，forward（两者均为局部变量）。

属性：

（1）attribute：制定与当前 Action 相关联的 ActionForm Bean 在 request 和 session 范围内的名称（key）。

（2）className：与 Action 元素对应的配置类。默认为 org.apache.struts.action.Action-革时代 Mapping。

（3）forward：指定转发的 URL 路径。

（4）include：指定包含的 URL 路径。

（5）input：指定包含输入表单的 URL 路径，表单验证失败时，请求会被转发到该 URL 中。

（6）name 指定和当前 Action 关联的 ActionForm Bean 的名字。该名称必须在 form-bean 元素中定义过。

（7）path：指定访问 Action 的路径，以“/”开头，没有扩展名。

（8）parameter：为当前的 Action 配置参数，可以在 Action 的 execute()方法中通过调用 ActionMapping 的 getParameter()方法来获取参数。

（9）roles：指定允许调用该 Action 的安全角色。多个角色之间用逗号分割。处理请求时，RequestProcessor 会根据该配置项来决定用户是否有调用该 Action 的权限。

（10）scope：指定 ActionForm Bean 的存在范围，可选值为 request 和 session。默认为 session。

（11）type：指定 Action 类的完整类名。

（12）unknown：值为 true 时，表示可以处理用户发出的所有无效的 Action URL。默认为 false。

（13）validate：指定是否要先调用 ActionForm Bean 的 validate()方法。默认为 true。

注意：在如上属性中，forward、include、type 三者相斥，即三者在同一 Action 配置中只能存在一个。

```
-->
<action path="/search"
    type="addressbook.actions.SearchAction"
    name="searchForm"
    scope="request"
    validate="true"
    input="/search.jsp">
    <forward name="success" path="/display.jsp"/>
    </action>
</action-mappings>

<!--
```

名称：controller。

描述：用于配置 ActionServlet。

数量：最多一个。

属性：

- (1) bufferSize：指定上传文件的输入缓冲的大小。默认为 4096。
- (2) className：指定当前控制器的配置类。默认为 org.apache.struts.config.ControllerConfig。
- (3) contentType：指定相应结果的内容类型和字符编码。
- (4) locale：指定是否把 Locale 对象保存到当前用户的 session 中，默认为 false。
- (5) processorClass：指定负责处理请求的 Java 类的完整类名。默认为 org.apache.struts.action.RequestProcessor。
- (6) tempDir：指定文件上传时的临时工作目录。如果没有设置，将使用 Servlet 容器为 Web 应用分配的临时工作目录。
- (7) noCache：为 true 时，在相应结果中加入特定的头参数 Pragma、Cache-Control、Expires 防止页面被存储在浏览器的缓存中，默认为 false。

```
-->

<controller
contentType="text/html; charset=UTF-8"
    locale="true"
    processorClass="CustomRequestProcessor">
</controller>

<!--
```

名称：message-resources。

描述：配置 Resource Bundle。

数量：任意多个。

属性：

(1) className：指定和 message-resources 对应的配置类。默认为 org.apache.struts.config.MessageResourcesConfig。

(2) factory：指定资源的工厂类，默认为 org.apache.struts.util.PropertyMessageResourcesFactory。

(3) key：null，parameter。

```
-->

<message-resources null="false" parameter="defaultResource"/>
<message-resources key="images" null="false" parameter="ImageResources"/>

<!--
```

名称：plug-in。

描述：用于配置 Struts 的插件。

数量：任意多个。

子元素：set-property。

属性：

className：指定 Struts 插件类。此类必须实现 org.apache.struts.action.PlugIn 接口。

```
-->

<plug-in className="org.apache.struts.validator.ValidatorPlugIn">
<!--
```

名称：set-property。

描述：配置插件的属性。

数量：任意多个。

属性：

(1) property：插件的属性名称。

(2) value：该名称所配置的值。

```
-->
<set-property
property="pathnames"
value="/Web-INF/validator-rules.xml,/Web-INF/vlaidation.xml"/>
</plug-in>

</struts-config>
```

#### 4.2.8 attribute 和 name 的区别

(1) 应用前提：attribute 只有在设置了 name 后才有意义。

(2) attribute 可以实现对象的重用，即如果设置了 attribute 属性，在创建 actionform 时，会先去查找相应的 scope 中是否有此对象，如果有，则重用，否则创建新的对象。

(3) 当将创建的 actionForm 保存到相应的 scope 中时，用一个更有意义的名字来访问它时，它就有意义了。

例如，配置 FormAction：

```
<form-bean name="employee" type="Employee"/>
```

配置 Action：

```
<action  
attribute="validEmployee"  
name="employee"  
type="EmployeeAction"  
scope="request"  
path="/employee">
```

这样就可以用 validEmployee 在 JSP 页面中访问了，而不是用 employee，这在同一个 Form 在不同情况下有不同的作用时，意义更明显。

本章讲述了一些关于 struts-config.xml 配置文件的内容，熟练使用配置文件，是 Struts 程序员的一个必备技能。



# 第 5 章

## V-View 视图层中的 ActionForm

在前几章的例子中，频繁地接触 ActionForm 类，这个类在 Struts 中扮演着非常重要的角色，它可以将前台表单传递到 Action 处理前进行有效的数据内容检验，还可以将表单域封装成 ActionForm 的 Bean 的形式，即设置者 set 和 get 访问者模式。

### 5.1 ActionForm 类的结构

如果想在 Struts 框架中使用 ActionForm 类的功能，则必须继承自 public class NewForm extends ActionForm 父类，而父类 ActionForm 不可以实例化，例程如下。

```
package com.yourcompany.struts.form;

import javax.servlet.http.HttpServletRequest;
import org.apache.struts.action.ActionErrors;
import org.apache.struts.action.ActionForm;
import org.apache.struts.action.ActionMapping;

public class NewForm extends ActionForm {
    private String name;

    public ActionErrors validate(ActionMapping mapping,
        HttpServletRequest request) {
        // TODO Auto-generated method stub
        return null;
    }

    public void reset(ActionMapping mapping, HttpServletRequest request) {
        // TODO Auto-generated method stub
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }
}
```

在 ActionForm 的子类 NewForm 类中有两个必要的方法：validate 和 reset。

Validate 方法简单地检验表单数据有效性，在该方法中需要返回一个 ActionErrors 类型的实例，如果返回 ActionErrors 实例的内容为 null 或者实例内错误元素的个数为 0，即没有包含任何的错误，就表示检验通过。如果为非 null 或者元素出错个数大于 0，那么再转回 input 的那个 Web 页面，在页面上显示错误提示。

reset 方法在封装成 ActionForm 前进行域属性初始化。在这里需要注意一点的是，千万不要将 JDBC 操作的代码放到 ActionForm 中，这大大破坏了基于 MVC 框架的工作分配目的。

当声明了一个 ActionForm Bean，在调用 Action 类的方法之前要做以下 4 步。

- (1) 检查这个 Bean 在一定的范围内是不是有一个实例。
- (2) 如果没有，那么在一定范围内 (request or session..) 生成该实例，即新建一个实例。
- (3) 对于 request 的参数，如果和 Bean 中的属性相匹配，那么该 Bean 的实例调用 setter 函数。
- (4) 修改的 ActionForm Bean 会被 Action 获得。在 Action 中执行业务逻辑代码。

当然在上面的例程中将属性固化在 ActionForm 类中，这样如果有上百个属性则会有相对应的 set 和 get 方法，这样对 ActionForm 维护相当不利，但在 Struts 中可以使用动态表单类 DynaActionForm 来解决这个情况。

## 5.2 ActionForm 生命周期

在 HTTP 请求/响应中，ActionForm 的生命周期取决于以下 4 种状态。

- Request：在一个请求周期内有效。就是从你单击页面上的一个按钮开始到服务器返回响应页面为止（包括响应页面）。
- Session：在一个用户与服务器建立连接的整个过程中有效。
- Application：在整个 Web 应用程序内有效。
- Page：仅在一个 JSP 页面内有效。

Struts 中的 ActionForm 类的生命周期有两种范围：request 和 session。

如果在当前的请求/响应生命周期中有效它属于 request 类型。这种情况当从一个 Web 资源转发到另一个 Web 资源时，ActionForm 的实例一直存在。但服务器把响应传到客户端的时候，ActionForm 的实例及它所包含的数据就被清除。

如果想让 ActionForm 的实例一直存在就放到 session 中。它在整个的 HTTP 会话中都存在，除非关闭浏览器。

## 5.3 DynaActionForm 使用方法

DynaActionForm 提供了一种方便的机制，从根本上消除了编写 ActionForm 的需要。DynaActionForm 允许动态地设置表单属性。这意味着能够在 struts-config.xml 文件中定义属性并且将表单类型设置为 org.apache.struts.action.DynaActionForm。

DynaActionForm 使用 Apache 公共项目中的 DynaBean 完成这些操作。这一动态的行为是通过反射（reflection）与哈希图（Hashmaps）提供的。

DynaActionForm 是在 struts-config.xml 文件中使用<form-bean>与<form-property>标记定义的，如下程序代码所示。

```
<form-bean name="insertDynaForm"
  type="org.apache.struts.action.DynaActionForm">    <form-property
  name="artist" type="java.lang.String"/>
<form-property name="title" type="java.lang.String"/>
<form-property name="genre" initial="Dance" type="java.lang.String"/>
</form-bean>
```

动态表单的属性与标准的 ActionForm 的属性类似。属性 name 是用于索引 Action 中的表单 Bean，并且 type 用于指定实例化的类。当使用类 DynaActionForm 时，<form-bean>的动态属性自动默认为真（true）。对于 DynaActionForm，要用<form-property>元素指定表单的所有属性。<form-property>元素中的 name 是指属性名称，type 是指 Bean 属性使用 Java 的实现类的类名。如果这个属性是索引类型的，可在 type 后添加“[]”。在上面的程序代码中，应该注意最后一个属性 genre 的<form-property>定义，设置了初始值（或叫默认值）为“Dance”。这个值也会在 DynaActionForm 中的 reset()方法被调用时作为默认值设置，并允许在表单中设置默认值的机制。如果在 initial 属性中没有指定任何值，那么所有原始类型的初值被设置为 0，如果是对象则初值为 null（空）。

使用 DynaActionForm 非常方便，主要的好处就是只需写非常少的代码。就像其他表单一样，前面的代码例子是使用表单所需的全部代码。需要知道的一件事就是验证，当使用 DynaActionForm 时，假定在某处进行了验证处理，这与 ActionForm 有些不同。你可以在自己的 Action 中实现验证，但这是一个更好的方法。

进行验证，可用 DynaValidatorForm 或者 DynaValidatorActionForm，这两个类都在 org.apache.struts.validator package 包中。通过扩展 DynaActionForm，可以得到基于 XML 文件的基本值域的验证。验证是基于输入验证器的 key。key 是来自于 struts-config.xml 文件的 name 属性。它应当与 validation.xml 文件中的表单元素的 name 属性匹配。

如果在一个 ActionForm 中有大量的表单项需要录入，比较姓名、年龄、出生日期等域属性，那么不得不在 ActionForm 中写入大量的 set 和 get 方法。如果表现层的业务功能经常改动，比如加入籍贯、婚否等这些变化，还需要在 ActionForm 中进行代码的增删，这样的操作增加了程序的维护周期和复杂度，应对这样的情况，使用 DynaActionForm 类再合适不过了。

## 5.4 DynaActionForm 实例

动态 DynaActionForm 与标准的 ActionForm 用途如出一辙，都是为了能够作为 DTO（Data Transfer Object）对象来与 Action 做数据传递，所不同的是动态 DynaActionForm 并没有一个专门的类作为 DTO 对象的载体，取而代之的是在 Struts 配置文件 FormBean 中增加属性配置项，来达到与 DTO 对象相同的作用。

- (1) 新建一个 Struts5.4 项目。
- (2) 加入 Struts 1.2 框架支持文件。
- (3) 新建一个 Form、Action 和 JSP，如图 5-1 所示。

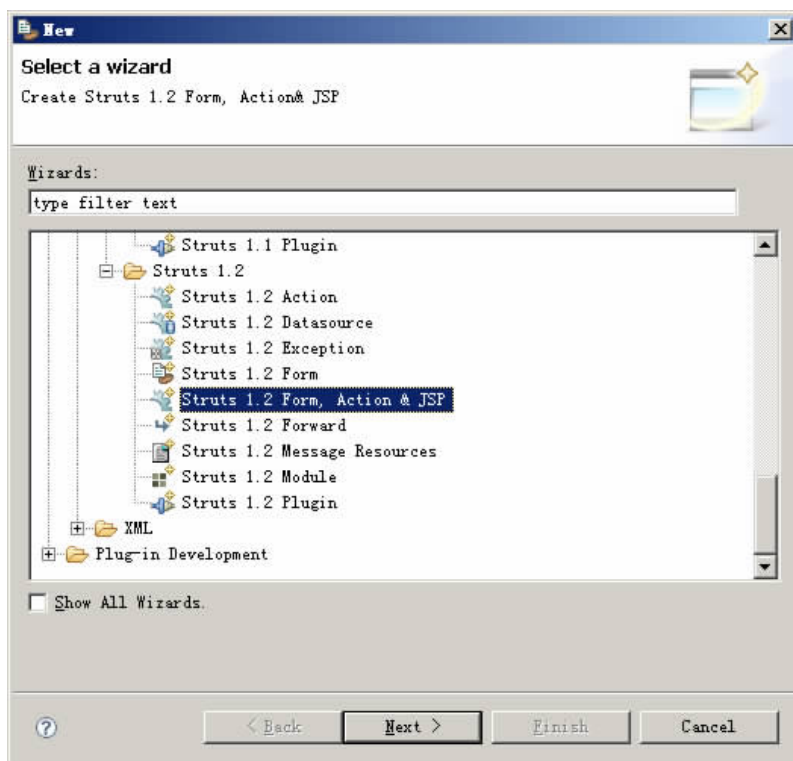


图 5-1 新建

- (4) 单击“Next”按钮，配置 ActionForm，如图 5-2 所示。
- 设置 Use case 为 login；设置 Form Impl 类型为动态的，即 Dynamic FormBean。  
添加两个属性 username 和 password。  
在 JSP 页中创建 JSP 文件，如图 5-3 所示。

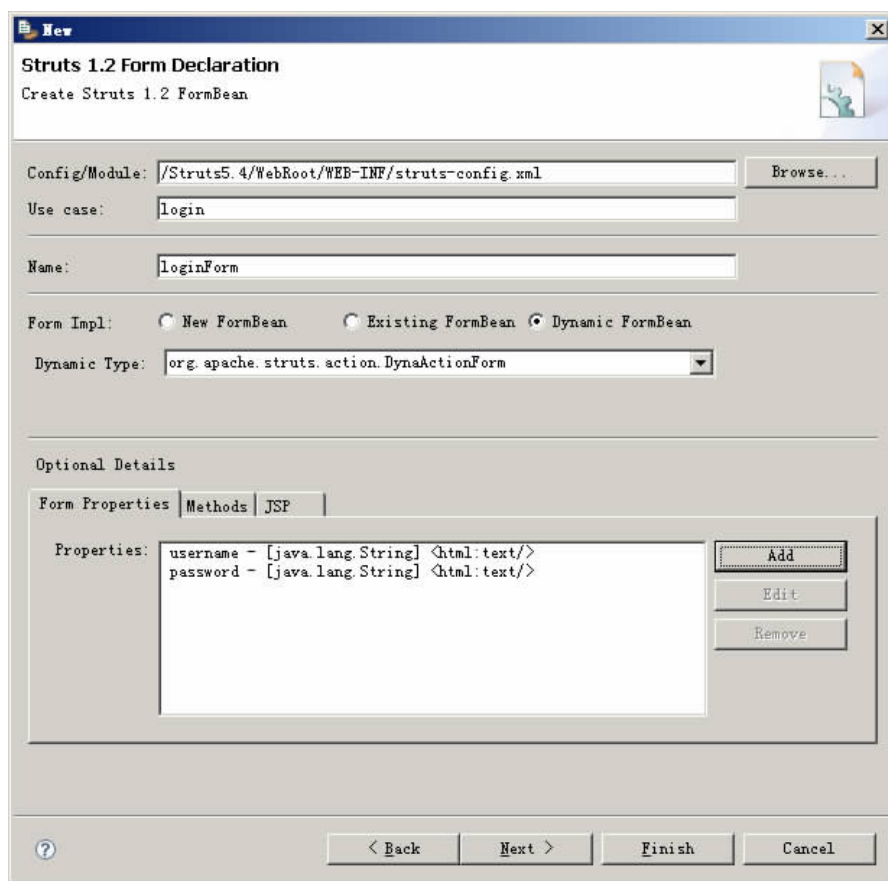


图 5-2 配置 ActionForm



图 5-3 创建 JSP 文件

单击“Next”按钮继续设置，出现如图 5-4 所示窗口，在本窗口中不需要设置任何属性，单击“Finish”按钮应用设置。

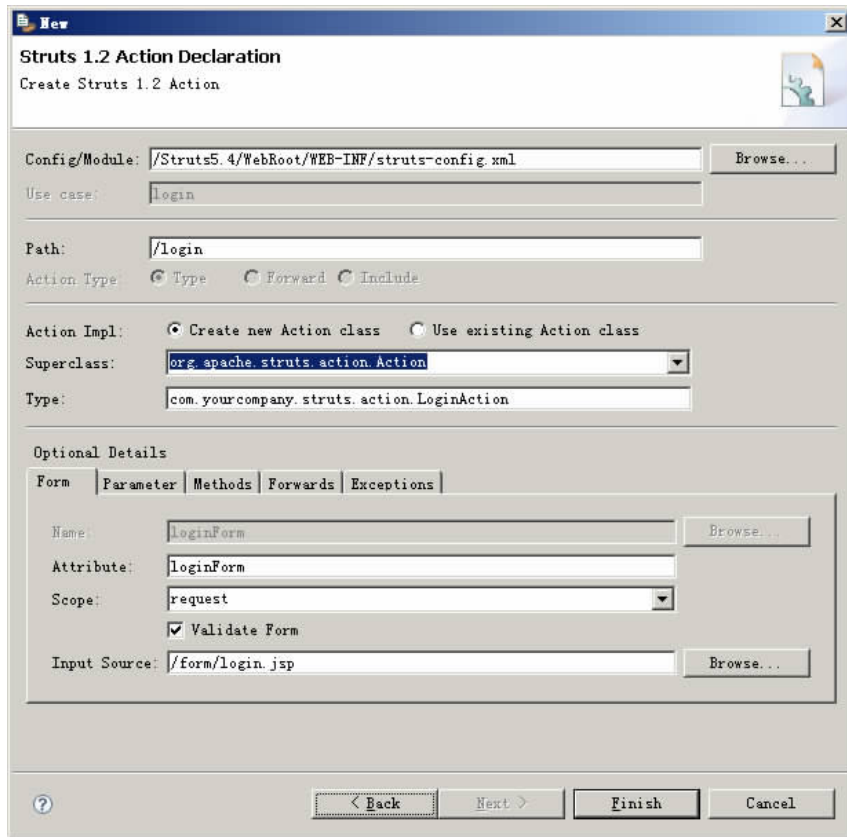


图 5-4 设置 Action

生成后的 JSP 文件内容为：

```
<%@ page language="java" pageEncoding="ISO-8859-1"%>
<%@ taglib uri=http://jakarta.apache.org/struts/tags-bean
prefix="bean"%>
<%@ taglib uri=http://jakarta.apache.org/struts/tags-html
prefix="html"%>

<html>
  <head>
    <title>JSP for DynaActionForm form</title>
  </head>
  <body>
    <html:form action="/login">
      password : <html:text property="password"/><html:errors
        property="password"/><br/>
      username : <html:text property="username"/><html:errors
        property="username"/><br/>
      <html:submit/><html:cancel/>
    </html:form>
  </body>
</html>
```

生成后的 XML 配置文件内容如下。

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE struts-config PUBLIC "-//Apache Software Foundation
//DTD Struts Configuration 1.2//EN"
"http://struts.apache.org/dtds/struts-config_1_2.dtd">

<struts-config>
  <data-sources />
  <form-beans >
<form-bean name="loginForm"
            type="org.apache.struts.action.DynaActionForm">
    <form-property name="password" type="java.lang.String" />
    <form-property name="username" type="java.lang.String" />
  </form-bean>

</form-beans>

<global-exceptions />
<global-forwards />
<action-mappings >
  <action
    attribute="loginForm"
    input="/form/login.jsp"
    name="loginForm"
    path="/login"
    scope="request"
    type="com.yourcompany.struts.action.LoginAction" />

</action-mappings>

  <message-resources
parameter="com.yourcompany.struts.ApplicationResources" />
</struts-config>
```

从配置文件中可以看到程序代码如下。

```
<form-beans >
<form-bean name="loginForm"
            type="org.apache.struts.action.DynaActionForm">
    <form-property name="password" type="java.lang.String" />
    <form-property name="username" type="java.lang.String" />
  </form-bean>

</form-beans>
```

ActionForm 的类型为 `org.apache.struts.action.DynaActionForm`，即动态表单。它有两个属性 `username` 和 `password`，数据类型都为 `String`。注意，在动态表单中设置域属性数据类型时一定要使用基本数据类型的外包类：

- `java.lang.BigDecimal`，`java.lang.BigInteger`，`java.lang.Boolean`。

- java.lang.Byte , java.lang.Character , java.lang.Class。
- java.lang.Double , java.lang.Float , java.lang.Integer。
- java.lang.Long , java.lang.Short , java.lang.String。
- java.sql.Data , java.sql.Time , java.sql.Timestamp。

如果想查看 Form Bean 还有没有其他的可设置属性怎么办？很简单，将光标移到 name 的前面，按下“Alt+/”键，这时就会出现下拉菜单，里面就是当前元素可设置的属性了，如图 5-5 所示。

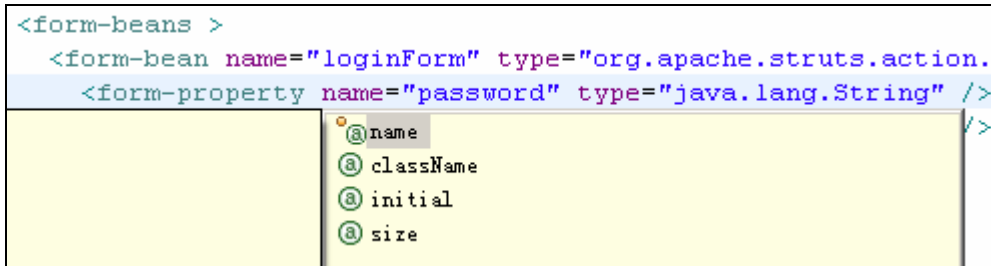


图 5-5 可用属性列表

配置文件中的 Action 元素和以前使用过的大体一样，没有什么特别之处。生成的 Action 类如下。

```
package com.yourcompany.struts.action;  
  
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.DynaActionForm;  
  
public class LoginAction extends Action {  
    public ActionForward execute(ActionMapping mapping, ActionForm form,  
        HttpServletRequest request, HttpServletResponse response) {  
        DynaActionForm loginForm = (DynaActionForm) form;  
        // TODO Auto-generated method stub  
        return null;  
    }  
}
```

在代码中可以看到，Form 被强制转换成 DynaActionForm 类型了。怎么取 Form 中的数据和页面转发呢？请看下面的程序代码。

```
public ActionForward execute(ActionMapping mapping, ActionForm form,  
    HttpServletRequest request, HttpServletResponse response) {  
    DynaActionForm loginForm = (DynaActionForm) form;  
    // TODO Auto-generated method stub
```



```
String username=(String)loginForm.get("username");
String password=(String)loginForm.get("password");
System.out.println(username+" "+password);
//继续处理这二个变量后进行转发
//return mapping.findForward("Other_PATH");
return null;
}
```

从总的开发顺序上来讲,和普通的 ActionForm 并没有什么大的区别。

部署,启动服务。

事物有其好处也有其坏处,那么动态表单也有两个非常突出的缺点。

### (1) 数据验证

在提及普通的 ActionForm 的时候曾经讲到过可以在 ActionForm 的 validate()方法中对用户输入的信息进行简单校验,而在动态 ActionForm 中却从未提到。这是因为如果想要实现动态 ActionForm 的校验必须做很多工作。首先必须引入 Validator 验证框架来帮助项目完成数据校验工作,其次由于某些数据的校验,还有可能为其编写数据校验项来帮助 Validator 框架完成数据校验工作。这与普通的 ActionForm 相比,动态 ActionForm 显得繁琐而笨拙。

### (2) 可配置

动态 ActionForm 的可配置性无疑是其存在的优势之一,但也有其缺点,因为可配置的 FormBean 与 Action 的耦合度还是相当大的。也就是说,由于 Action 类引用了如“String userID=(String)form.get("userID");”这样的代码,那么当我们增加或删除 Form 中的属性的时候就必须更改我们的 Action 类,这与普通的 ActionForm 的做法是一样的,失去了其可配置的优势所在。

总结:推荐使用普通的 ActionForm 去履行 DTO 的职责,因为那样会更方便地解决应用问题。动态 ActionForm 只是一点小尝试,不推荐使用。

在浏览器中输入相应的用户名和密码,然后提交,这时在 MyEclipse 的控制台 Console 中出现刚才输入的数据了。

DynaActionForm 适合界面业务经常改动的情况,总之它和 ActionForm 都有优缺点,至于在实际项目情况中使用哪个类就由业务逻辑决定了。

## 5.5 Action 和 ActionForm 配置精例

学习到这里,已经能将 Action 和 ActionForm 的工作目的搞清楚了,下面看几个比较常用且非常经典的配置实例。

### 5.5.1 完整的 action 功能

代码如下。

```
<action path="/aFullAction"
        type="somePackage.someActionClass">
    name="someForm"
```

```
input="someJSP.jsp"
<forward name="successful" path="someJSP.jsp"/>
<forward name="failed" path="someOtherJSP.jsp"/>
</action>
```

首先，Struts 的 ActionServlet 接收到一个请求，然后根据 struts-config.xml 的配置定位到相应的 mapping（映射）；接下来如果 Form 的范围是 request 或者在定义的范围中找不到这个 Form，创建一个新的 Form 实例；取得 Form 实例以后，调用其 reset()方法，然后将表单中的参数放入 Form，如果 validate 属性不为 false，调用 validate()方法；如果 validate()返回非空的 ActionErrors，将会被转到 input 属性指定的 URI，如果返回空的 ActionErrors，那么执行 Action 的 execute()方法，根据返回的 ActionForward 确定目标 URI。这样做的效果是：execute()仅当 validate()成功以后才执行；input 属性指定的是一个 URI。

### 5.5.2 仅有 Form 的 action 超级链接功能

代码如下。

```
<action path="/aFormOnlyAction"
  type="org.apache.struts.actions.ForwardAction"
  name="someForm"
  input="someJSP.jsp"
  parameter="someOtherJSP.jsp"
/>
```

首先，Struts 会在定义的 scope 搜寻 someForm，如果找到则重用，如果找不到则新建一个实例；取得 Form 实例以后，调用其 reset()方法，然后将表单中的参数放入 Form，如果 validate 属性不为 false，调用 validate()方法；如果 validate()返回非空的 ActionErrors，将会被转到 input 属性指定的 URI，如果返回空的 ActionErrors，那么转到 parameter 属性指定的目标 URI。

这样做的效果是：没有 Action 类可以存放业务逻辑，所以所有需要写入的逻辑都只能写到 Form 的 reset()或者 validate()方法中。validate()的作用是验证和访问业务层。因为这里的 Action 映射不包括 forward（也没有意义），所以不能重定向，只能用默认的那个 forward。仅有 Form 的 Action 可以用来处理数据获取并 forward 到另一个 JSP 来显示。

### 5.5.3 仅有 Action 的 action 执行链接式请求后就转发

代码如下。

```
<action path="/anActionOnlyAction"
  type="somePackage.someActionClass">
  input="someJSP.jsp"
  <forward name="successful" path="someJSP.jsp"/>
  <forward name="failed" path="someOtherJSP.jsp"/>
</action>
```

首先, ActionServlet 接收到请求后, 取得 action 类实例, 调用 execute()方法; 然后根据返回的 ActionForward 在配置文件中找 forward, forward 到指定的 URI 或 action。

这样做的效果是: 没有 Form 实例被传入 execute()方法, 于是 execute()必须自己从请求中获取参数。Action 可以被 forward 或者重定向。这种 action 不能处理通过 HTML Form 提交的请求, 只能处理链接式的请求。这种形式适合单击超级链接后, 转换 URL 前想做一些事情的情况。

#### 5.5.4 仅有 JSP 的 action

代码如下。

```
<action path="/aJSPOnlyAction"
      type="org.apache.struts.actions.ForwardAction"
      parameter="someOtherJSP.jsp"
/>
```

首先, ActionServlet 接到请求后调用 ForwardAction 的 execute()方法, execute()根据配置的 parameter 属性值来 forward 到哪个 URI。

这样做的效果是: 没有任何 Form 被实例化, 比较现实的情形可能是 Form 在 request 更高级别的范围中定义; 或者这个 action 被用做在应用程序编译好后充当系统参数, 只需要更改这个配置文件而不需要重新编译系统。

#### 5.5.5 两个 action 对应一个 Form

代码如下。

```
<action path="/anAction"
      type="somePackage.someActionClass">
  name="someForm"
  input="someJSP.jsp"
  <forward name="successful" path="/anotherAction.do"/>
</action>
<action path="/anotherAction"
      type="somePackage.someOtherActionClass">
  name="someForm"
  input="someOtherJSP.jsp"
  <forward name="successful" path="someResultJSP.jsp"/>
</action>
```

就每个单独的 action 来讲, 处理上并没有和完整的 action 有什么实质的区别。这个组合模式可以被用来传递命令对象( Form )。需要注意的是在后一个 action 中同样会调用 Form 的 reset()和 validate()方法, 因此必须确保 Form 中的信息不被重写。

处理的方式大致分为两种: (1) 在 request 中放入一个指示器表明前一个 action 有意向后一个 action 传递 Form, 从而在后一个 action 可以保留那个 Form 中的值, 这一方式只能在使用 forward 时使用。(2) 当使用 redirect 而不是 forward 时, 可以把指示器放在 session 或更高的级别, 在请求/应答的最后阶段将这个标记清除。

### 5.5.6 两个 action 对应两个 Form

代码如下。

```
<action path="/anAction"
    type="somePackage.someActionClass">
    name="someForm"
    input="someJSP.jsp"
    <forward name="successful" path="/anotherAction.do"/>
</action>
<action path="/anotherAction"
    type="somePackage.someOtherActionClass">
    name="someForm"
    input="someOtherJSP.jsp"
    <forward name="successful" path="someResultJSP.jsp"/>
</action>
```

这个组合方式与前一种在流程上没有太大区别，只是现在对于两个 action 分别提供了 Form，于是代码看上去更加清晰。这样可以分别处理 Web 应用程序的输入和输出。值得注意的是，后一个 action 同样会尝试往 Form 中写入那些参数，不过可以这样处理：(1) 在后一个 Form 中使用另一套属性名；(2) 只提供 getter 而不提供 setter。

大致的处理过程如下。

前一个 action 接收输入、验证，然后将数据写入业务层或持久层，重定向到后一个 action 中，后一个 action 手动地从业务层/持久层取出数据，写入 Form（通过其他方式），交给前台 JSP 显示。

这样做的好处是不必保留输入 Form 中的值，因此可以使用 redirect 而不是 forward。这样就降低了两个 action 之间的耦合度，同时也避免了不必要的重复提交。

## 5.6 ActionForm 中文乱码问题解决方案

在 ActionForm 遇到中文乱码问题时，可以有两种解决办法，一种是在 ActionForm 类中，另一种是在 Action 类中。

### (1) Action 类中解决办法

```
ActionForm cat = (ActionForm)cat;
String name = new String(cat.getName().getBytes("ISO-8859-1"), "UTF-8");
cat.setName(name);
```

以上代码写在 Action 中的 execute() 方法中。

### (2) ActionForm 类中解决办法

在 ActionForm 中的 reset 方法中加入如下代码即可。

```
public void reset(ActionMapping actionMapping, HttpServletRequest
    httpRequest)
{
```

```
try
{
    httpRequest.setCharacterEncoding("GBK");
}
catch (UnsupportedEncodingException ex)
{
}
}
```

## 5.7 Struts 中的 ActionErrors

比如在 Action 类中写入下面代码。

```
ActionErrors errors = new ActionErrors();
errors.add(ActionErrors.GLOBAL_ERROR,new
ActionError("error.buildsite.session_null"));
saveErrors(request,errors); //使用 saveErrors 方法将错误信息保存进 request 中
return mapping.findForward("buildsite");
//再转发到 buildsite 相对应的物理路径中
```

该代码使用全局错误标志 `ActionErrors.GLOBAL_ERROR`，这时就可以在 JSP 页面的 `<html:errors/>` 里显示错误信息了。不要设置 `html:errors` 的 `property` 属性。

`new ActionError("error.buildsite.session_null")` 代码中的参数是资源文件中的 `key` 值。

本章一起讨论了关于 ActionForm 的一些知识点，下一章将要讲述的是关于 HTML 标签库的使用，也就是制作 JSP 界面时用到的一些 Struts 重要标记元素，HTML 标签库在开发 Struts 项目时很重要。

# 第 6 章

## Struts-HTML 标签库

在 Struts 的开发中，接触 Web 界面的开发尤为频繁，一些表单数据提交前的设计也很重要，在 Struts 框架支持文件中，会发现有一个 HTML 的标签库，这点在前几章的实例中已经有所接触。那么普通的 HTML 标记与 Struts 中的 HTML 标签有什么区别呢？

### 6.1 普通 HTML 与 Struts 中 HTML 标签的区别及 Struts 标签公共特征

#### 1. 普通 HTML 与 Struts 中 HTML 的区别

##### (1) 功能的扩展

Struts 是对 HTML 标记的功能拓展，它不仅包含 HTML 的基本属性，而且还定义了一些扩展的方法和属性，比如 Struts 中的 HTML 可以对 ActionForm 进行自动的添充等。在 struts-html.tld 这个文件中就有其所有标签的所有属性的描述。

##### (2) 国际化

简短文字的描述可以通过资源文件的形式进行国际化，这点 HTML 标记做得不到位。

##### (3) 定制化集成性

Struts 中的 HTML 标签可以与 ActionForm 和 Action 非常无缝地进行功能性的配合，这点 HTML 标记没有完全实现。Struts 中的 HTML 标签定制化得非常好。

#### 2. 标签的公共特征

在 Struts 的框架中，有很多属性具有共同的功能意义，最常见的就是下面 4 种属性。

- id：命名自定义标签创建时的脚本变量名。
- name：指出关键字值，在该关键字下可以找到一个存在的 Bean。如果给出了 scope 属性，则仅仅在 scope 中查找。否则，根据标准的顺序在各种 scope 中查找（如 page、request、session、application）。
- property：指出 Bean 中的某个属性，可以在其中检索值。如果没有标明，则使用对象本身的值。
- scope：定义了 Bean 在哪个范围（如 page、request、session、application）中被查找。如果没有标明按顺序查找，脚本变量（见 id）将在相同的范围中创建。

## 6.2 显示 Struts-HTML 标签的 Snippets 窗口

新建一个 Web Project 项目,添加 Struts 1.2 框架支持文件后,可以在 MyEclipse 中显示出当前 Struts 版本支持的所有 Struts-HTML 标签。

(1) 选择“Window” “Show View” “Other”菜单项,如图 6-1 所示,打开如图 6-2 所示的窗口,选择“ MyEclipse Enterprise Workbench ” “ Snippets ”选项。

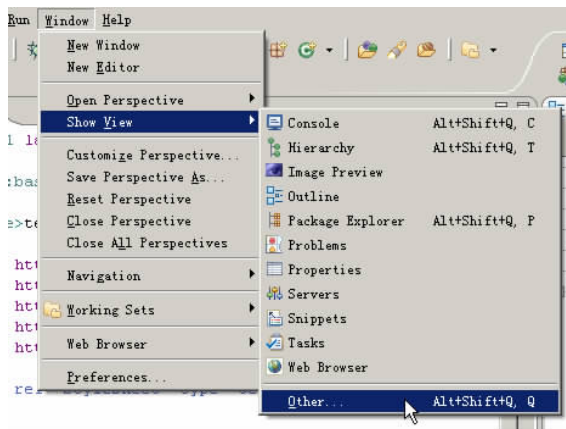


图 6-1 菜单位置

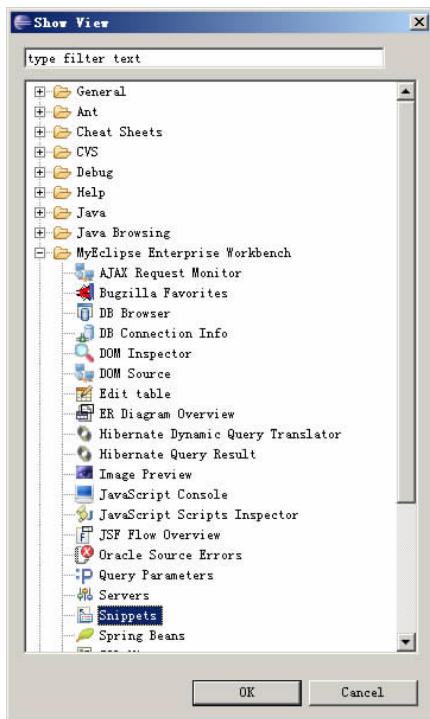


图 6-2 Snippets 选项

(2) 在 MyEclipse 的左边出现一个面板页，如图 6-3 所示。

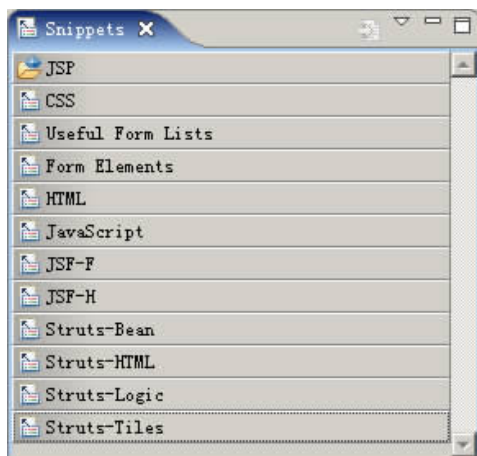


图 6-3 Snippets 面板页

在这个面板中有 MyEclipse 支持的所有框架的标签，比如 JSF 等，还有一些普通的 HTML 和 JavaScript 的代码模板。单击 Struts-HTML 子面板后，出现如图 6-4 所示的界面。

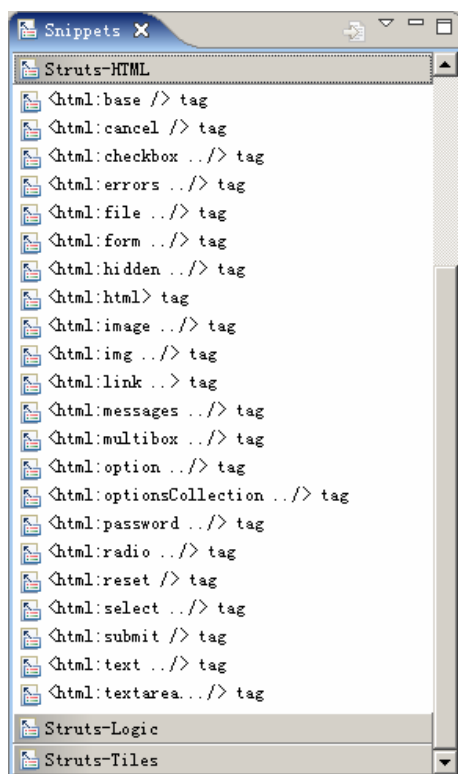


图 6-4 Struts-HTML 子面板

在本节以后的内容中，将使用实例介绍这里面所有的 Struts-HTML 标签。下面开始逐一介绍。



## 6.3 <html:base />设置相对根路径

### 6.3.1 标签简介

本标签不是必须使用的。

<html:base />标记和 HTML 中的<base>标记功能一模一样，主要用来生成一个相对的 URL 路径。它转换成 HTML 代码的位置在标记<head></head>中。建议查看本小节程序实例生成的 HTML 代码。

它将在输出的 HTML 文件的 head 中添加进类似<base href="http://server:port/requestURI"/>的 HTML 标记。浏览器通过 base 标签来把当前 HTML 文件中所有的相对 URL 转换成绝对 URL。

<html:base />的属性如图 6-5 所示。

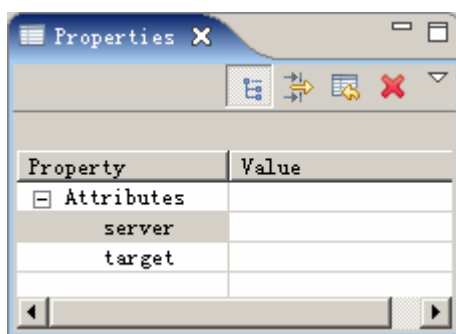


图 6-5 <html:base />属性

从图中可以看到该标签的属性比较少，只有两个，使用也非常方便。

- server 属性是设置相对根路径的 URL。
- target 设置单击超级链接后的打开方式，和 HTML 语言中<a>标记的 target 属性功能一样。

### 6.3.2 使用示例 1

- (1) 新建一个 Web 项目 Struts6.2.1。
- (2) 添加 Struts 1.2 框架支持文件。
- (3) 新建一个 JSP 文件，名称为 index.jsp，该 JSP 文件支持 Struts 1.2。

更改 index.jsp 文件内容如下：

```
<%@ page language="java" pageEncoding="ISO-8859-1"%>

<%@ taglib uri="http://struts.apache.org/tags-bean" prefix="bean"%>
<%@ taglib uri="http://struts.apache.org/tags-html" prefix="html"%>
<%@ taglib uri="http://struts.apache.org/tags-logic" prefix="logic"%>
<%@ taglib uri="http://struts.apache.org/tags-tiles" prefix="tiles"%>
```

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html:html lang="true">
<head>
  <html:base server="www.google.com" target="_blank" />

  <title>index.jsp</title>

  <meta http-equiv="pragma" content="no-cache">
  <meta http-equiv="cache-control" content="no-cache">
  <meta http-equiv="expires" content="0">
  <meta http-equiv="keywords" content="keyword1,keyword2,keyword3">
  <meta http-equiv="description" content="This is my page">
  <!--
  <link rel="stylesheet" type="text/css" href="styles.css">
  -->

</head>

<body>
  <a href="welcome.jsp">to end.jsp</a>
</body>
</html:html>
```

在上面的程序代码中，更改了<html:base>标签的内容：

```
<html:base server="www.google.com" target="_blank" />
```

本语言的功能是将当前页面的所有相对路径的根路径设置为 www.google.com，并且打开超级链接的方式为\_blank 类型。

(4) 部署。启动 Tomcat 后，超级链接指向的路径其实是：http://www.google.com:8080/Struts6.2.1/welcome.jsp。

从中不难看出<html:base>标签和 HTML 的<base>标记功能一样，都是设置基础网址。

由于 Struts 框架的特殊处理，在本例中去掉这个<html:base>标签功能没有影响，这个标签有无均可，但为了以后的功能完善性，建议加入本标签，以免遇到路径错误，增加不必要的调试时间。

### 6.3.3 使用示例 2

在 6.3.2 的实例中只是简单地介绍在单纯的 JSP 页面中如何使用本标签，但如果在 Action 的情况下，本标签的功能有没有作用呢？

(1) 新建一个 Web 项目 Struts6.2.2。

(2) 新建一个 JSP 网页 index.jsp。更改<body>中内容如下：

```
<body>
  <a href="/index.do">to end.jsp</a>
</body>
```

(3) 新建一个 Action，设置如图 6-6 所示。

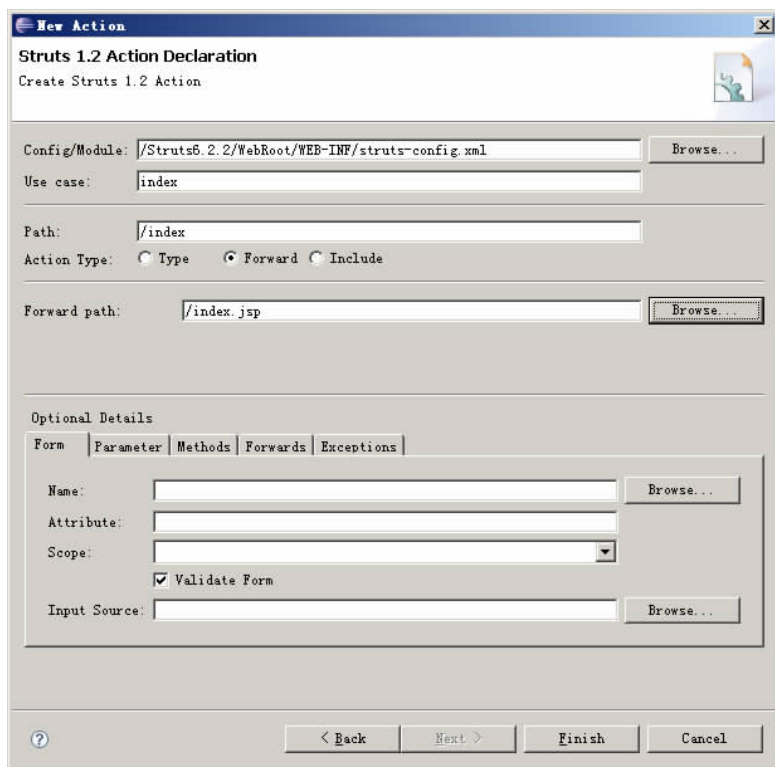


图 6-6 设置 Action

(4) 单击“Finish”按钮应用设置。

struts-config.xml 文件内容如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE struts-config PUBLIC "-//Apache Software Foundation
//DTD Struts Configuration 1.2//EN"
"http://struts.apache.org/dtds/struts-config_1_2.dtd">

<struts-config>
  <data-sources />
  <form-beans />
  <global-exceptions />
  <global-forwards />
  <action-mappings >
    <action forward="/index.jsp" path="/index" />

  </action-mappings>

  <message-resources
parameter="com.yourcompany.struts.ApplicationResources" />
</struts-config>
```

(5) 部署应用程序，启动 Web 服务。

这时发现在本例中使用与不使用<html:base>标签都有一样的效果，则以后在项目开发中可以在有<html:base>标记的情况下，并且设置了 server 属性为第三方的地址时，效果才生效，否则加和不加此标签的功能一样。

## 6.4 <html:cancel />取消当前的提交

### 6.4.1 标签简介

本标签必须使用在<html:form>标签中。

<html:cancel>标签生成一个取消按钮。当单击该按钮后 Action Servlet 会绕过相应的 Form Bean 的 validate()方法，同时将控制权交给相应的 Action。在该 Action 中可以使用 Action.isCancelled(HttpServletRequest)方法判断是否被取消了，如果返回 true 表示这个 Action 被取消了，否则表示这个 Action 没有被取消。

请注意，如果修改了<html:cancel>标签的 Property 属性值，那么 Struts 提供的 Cancel 探测机制就失效了，程序员必须自定义提供类似的机制。

### 6.4.2 使用示例

当单击了提交按钮后，又临时想取消当前的操作，可以使用下面的方法步骤。

(1) 新建一个 Web 项目 Struts6.3。

(2) 新建一个 Form、Action、JSP，如图 6-7 所示。

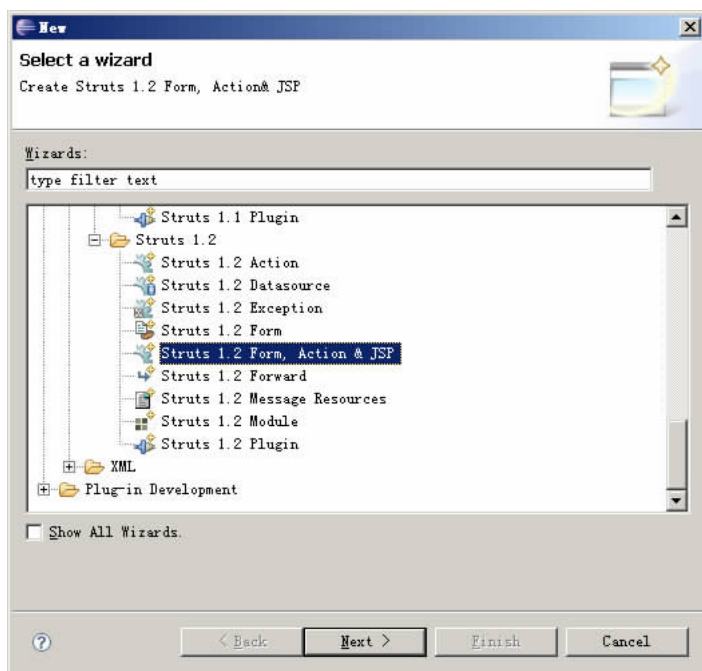
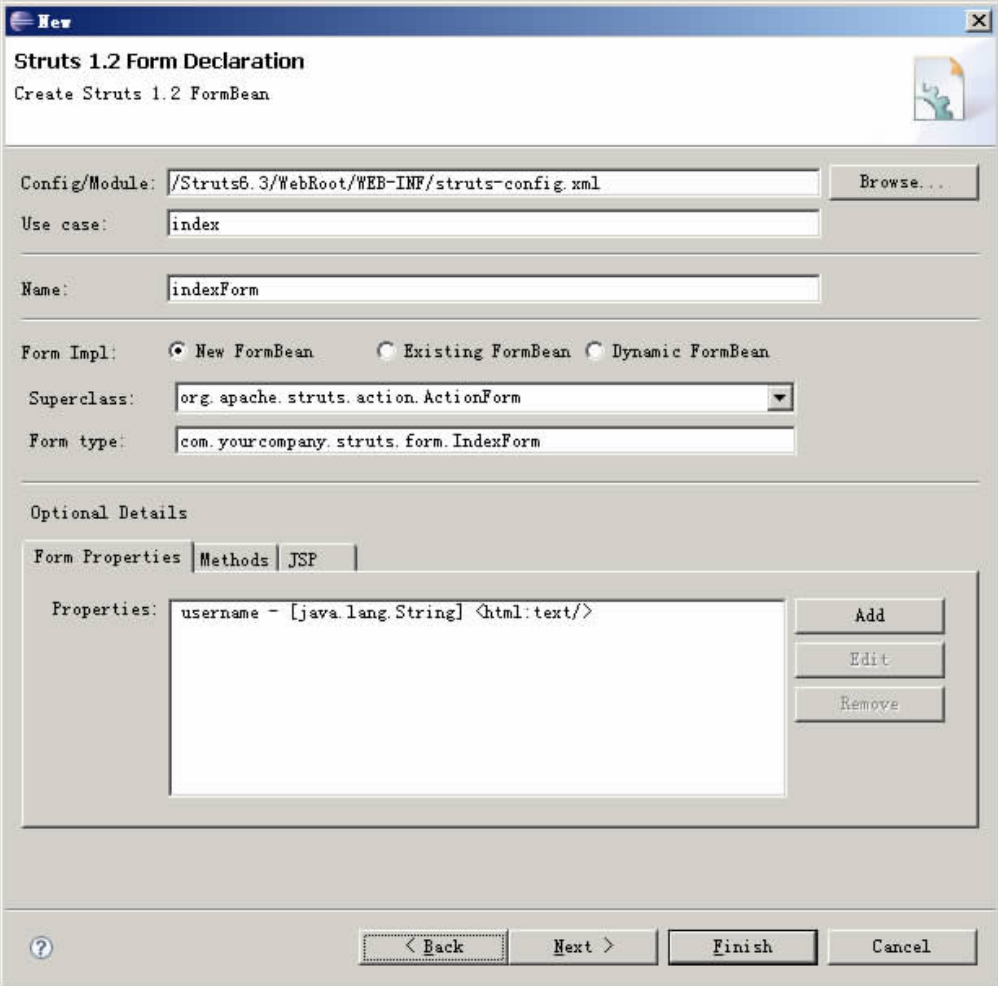


图 6-7 新建 Form、Action、JSP

(3) 单击“Next”按钮后出现如图 6-8 所示的窗口，并设置如下。



The image shows the "Struts 1.2 Form Declaration" dialog box. It has a title bar with a question mark icon and a close button. The main area is divided into several sections. At the top, it says "Create Struts 1.2 FormBean". Below this, there are three text fields: "Config/Module:" with the value "/Struts5.3/WebRoot/WEB-INF/struts-config.xml" and a "Browse..." button; "Use case:" with the value "index"; and "Name:" with the value "indexForm". Below these is a section for "Form Impl:" with three radio buttons: "New FormBean" (selected), "Existing FormBean", and "Dynamic FormBean". Below the radio buttons are two text fields: "Superclass:" with the value "org.apache.struts.action.ActionForm" and a dropdown arrow; and "Form type:" with the value "com.yourcompany.struts.form.IndexForm". Below this is a section titled "Optional Details" with three tabs: "Form Properties" (selected), "Methods", and "JSP". Under the "Form Properties" tab, there is a text area labeled "Properties:" containing the text "username - [java.lang.String] <html:text/>". To the right of the text area are three buttons: "Add", "Edit", and "Remove". At the bottom of the dialog are four buttons: "< Back", "Next >", "Finish", and "Cancel".

图 6-8 设置 ActionForm

(4) 设置 JSP 属性页如图 6-9 所示。



The image shows the "Struts 1.2 JSP Properties" dialog box. It has a title bar with a question mark icon and a close button. The main area is divided into three tabs: "Form Properties", "Methods", and "JSP" (selected). Under the "JSP" tab, there is a checkbox labeled "Create JSP form?" which is checked. Below the checkbox is a text field labeled "New JSP Path:" with the value "/form/index.jsp".

图 6-9 设置 JSP 属性页

(5) 单击“Next”按钮后，出现如图 6-10 所示的窗口。

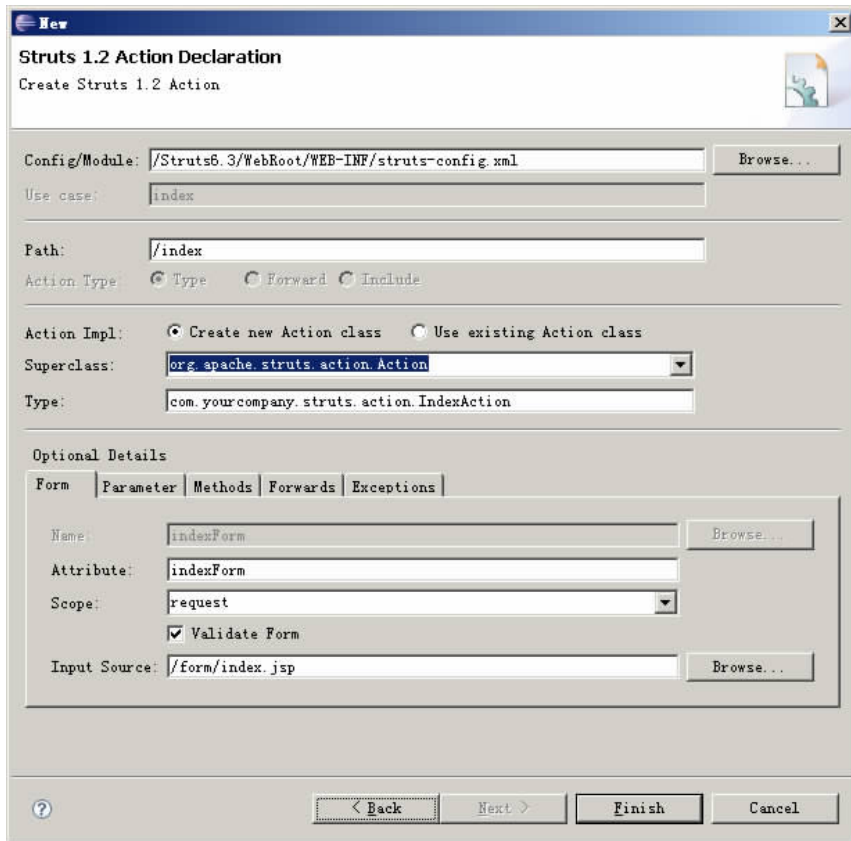


图 6-10 设置 Action

(6) 在图 6-10 中不需要设置任何选项，直接单击“Finish”按钮即可完成设置。生成的 JSP 文件内容如下。

```
<%@ page language="java" pageEncoding="ISO-8859-1"%>
<%@ taglib uri="http://jakarta.apache.org/struts/tags-bean"
prefix="bean"%>
<%@ taglib uri="http://jakarta.apache.org/struts/tags-html"
prefix="html"%>

<html>
  <head>
    <title>JSP for IndexForm form</title>
  </head>
  <body>
    <html:form action="/index">
      username : <html:text property="username"/><html:errors
        property="username"/><br/>
      <html:submit/><html:cancel/>
    </html:form>
  </body>
</html>
```

(7) MyEclipse 自动生成了 struts-config.xml 文件, 在这里需要手动添加两个 forward, 配置文件内容如下。

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE struts-config PUBLIC "-//Apache Software Foundation
//DTD Struts Configuration 1.2//EN"
"http://struts.apache.org/dtds/struts-config_1_2.dtd">

<struts-config>
  <data-sources />
  <form-beans >
    <form-bean name="indexForm"
      type="com.yourcompany.struts.form.IndexForm" />
  </form-beans>

  <global-exceptions />
  <global-forwards >
    <forward name="cancel" path="/form/cancel.jsp" />
    <forward name="succ" path="/form/succ.jsp" />
  </global-forwards>

  <action-mappings >
    <action
      attribute="indexForm"
      input="/form/index.jsp"
      name="indexForm"
      path="/index"
      scope="request"
      type="com.yourcompany.struts.action.IndexAction" />
  </action-mappings>

  <message-resources
    parameter="com.yourcompany.struts.ApplicationResources" />
</struts-config>
```

(8) 在前面做完所有的准备工作之后, 最后一步也就是更改 IndexAction.java 文件的内容了, 结果如下。

```
public ActionForward execute(ActionMapping mapping, ActionForm form,
    HttpServletRequest request, HttpServletResponse response) {
    IndexForm indexForm = (IndexForm) form; // TODO Auto-generated method
    // stub
    if (this.isCancelled(request)) {
        System.out.println("run cancel");
        return mapping.findForward("cancel");
    } else {
        System.out.println("run more code!~");
    }
}
```

```
    }  
  
    return mapping.findForward("succ");  
}  
}
```

在 `execute` 方法中使用了一个 `isCancelled` 方法来判断前台用户是否单击了“Cancel”按钮，如果单击则处理一些功能代码，比如转到 `Cancel`，否则转到 `succ`。

`<html:cancel>` 标记生成的 HTML 代码为：

```
<input type="submit" name="org.apache.struts.taglib.html.CANCEL"  
value="Cancel" onclick="bCancel=true;">
```

## 6.5 <html:checkbox /> 复选框

### 6.5.1 标签简介

`<html:checkbox>` 标签生成一个 checkbox。这里的 `value` 值可以是 `true`、`yes` 或 `on`，还可以是 `Boolean` 的数据类型。如果要提交 checkbox 分组类型的情况，应该考虑使用 `html:multibox` 标签。

对于程序 `<html:checkbox value="checkbox" property="checkbox"/>`，当传入的 Bean 里的 checkbox 的值（也就是 `property`）等于 checkbox 的 `value` 的值时，就会自动处于选中状态。

`<html:checkbox>` 标签有两个属性 `name` 和 `property`，这两个属性是用来做选中判断的，`name` 决定了当前 JSP 的 `pageContext` 中保存的对象，而 `property` 则是这个对象的一个属性，这个对象的类型必须有 `get/set` 方法的 Bean 或 `map`、`list` 等结构，这样通过这几个参数得到一个值，这个值就是作为是否选中校验的数据，如果这个值与 checkbox 标签的 `value` 属性的值相同，或这个值的 `String` 值等于 `true/yes/on` 中的任一个字符串值，那么都表示该 checkbox 被选中了。

注意：为了正确地处理没有选中的 checkbox，必须在 `reset()` 中设置对应的属性为 `false` 或设置非 `true/yes/on` 字符串，下面小程序段中将数据类型设置为 `Boolean`：

```
private boolean one = false;  
private boolean two = false;  
private boolean three = false;
```

### 6.5.2 使用示例

下面的代码示范了 `<html:checkbox>` 标签的用法，其中 `CheckboxForm` 中声明了 3 个 `Boolean` 类型的域，如下所示。

```
<html:checkbox name="checkboxForm" property="one">  
    One  
</html:checkbox>  
  
<html:checkbox name="checkboxForm" property="two">
```



```
Two
</html:checkbox>

<html:checkbox name="checkboxForm" property="three">
  Three
</html:checkbox>
```

如果选中 checkbox 后被提交则相应的属性的值为 true。

下面来做一个实例。

- (1) 新建一个 Web 项目 Struts6.4。
- (2) 新建一个 Form、Action、JSP，如图 6-11 所示。

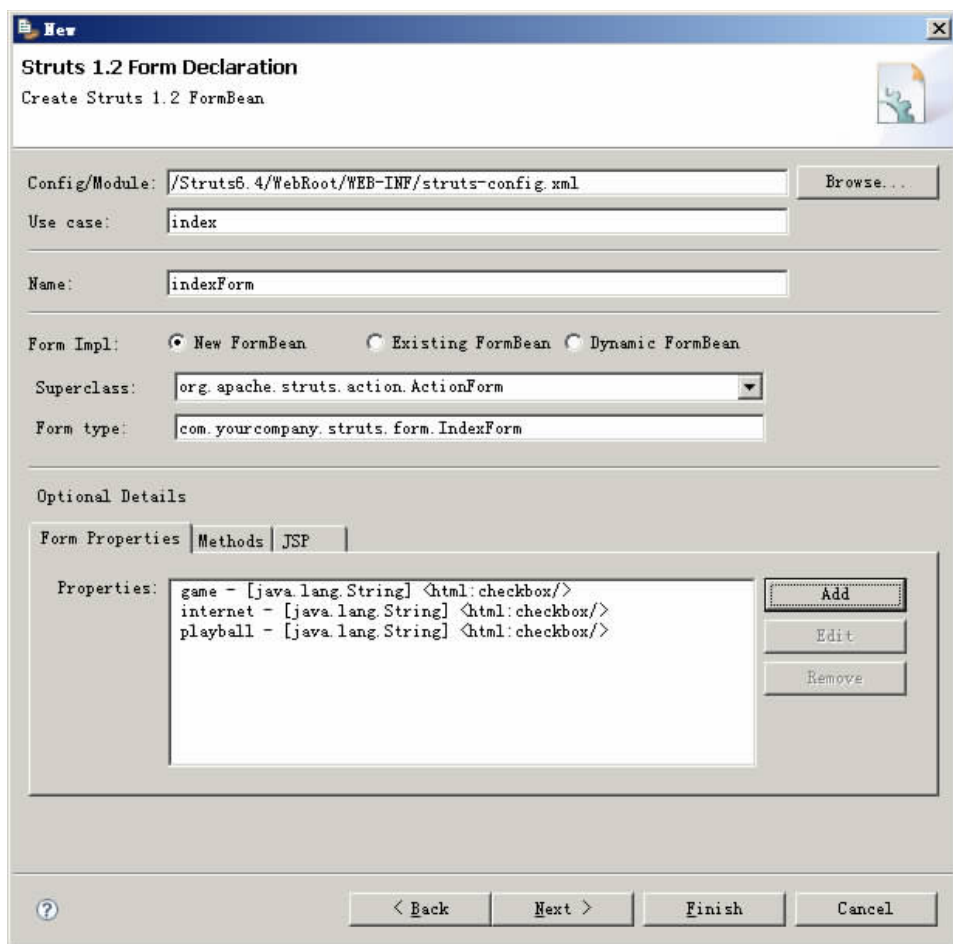


图 6-11 新建 Form、Action、JSP

在添加属性时，请注意改变属性的表单类型，这里选择 checkbox 类型，即复选框。

这里设计的是“爱好”功能，可以在里面选择 3 种爱好，但这 3 种爱好不是 checkbox 分组的情况。

- (3) 设置 JSP 属性页，创建 JSP 文件，如图 6-12 所示。



图 6-12 创建 JSP 文件

(4) 单击“Next”按钮后，设置 Action，在这里选择默认设置，单击“Finish”按钮，应用设置。

更改 IndexAction.java 文件如下：

```
public class IndexAction extends Action {
    public ActionForward execute(ActionMapping mapping, ActionForm form,
        HttpServletRequest request, HttpServletResponse response) {
        IndexForm indexForm = (IndexForm) form; // TODO Auto-generated method
                                                // stub
        System.out.println(indexForm.getPlayball());
        System.out.println(indexForm.getGame());
        System.out.println(indexForm.getInternet());

        return mapping.getInputForward();
    }
}
```

(5) 在 Action 中打印出复选框的状态。

(6) 部署程序，启动服务。

(7) 在浏览器界面中，将 playball 和 game 选择后，单击“提交”按钮，这时，在 Eclipse 的控制台输出了结果，如下所示。

```
on
on
null
```

从中可以看出，选中的复选框值为 on，没有选中的为 null。通过这样的测试，就可以在 Action 中对结果值进行进一步的处理了。

但是，发现一个问题：单击“提交”按钮转回 index.jsp 后，playball 和 game 是选中状态，而不是初始化的不选中状态，怎么样处理这样的情况呢？下面将要对其进行介绍。

### 6.5.3 <html:checkbox />复选框在 Action 中状态的改变

在 Struts 的<html:checkbox>标签中，将复选框选中后，在整个的会话中，将始终维持选中的状态，这就是上一小节的问题，怎么样解决呢？其实很简单，只需要在 Action 中增加一些代码即可。

针对上面的例子，改动 IndexAction.java 文件的代码如下。

```
public class IndexAction extends Action {
    public ActionForward execute(ActionMapping mapping, ActionForm form,
        HttpServletRequest request, HttpServletResponse response) {
        IndexForm indexForm = (IndexForm) form; // TODO Auto-generated method
        // stub
        System.out.println(indexForm.getPlayball());
        System.out.println(indexForm.getGame());
        System.out.println(indexForm.getInternet());

        indexForm.setPlayball(null);
        indexForm.setGame(null);
        indexForm.setInternet(null);

        return mapping.getInputForward();
    }
}
```

这时就会发现不会再出现永远选中的状态了。

#### 6.5.4 <html:checkbox />复选框的初始化

如果在页面显示的时候，希望某一个复选框配置为选中的状态，这样的情况怎么办呢？针对上面的例子，改动 IndexForm.java 文件中的 reset 方法内容如下。

```
public void reset(ActionMapping mapping, HttpServletRequest request) {
    // TODO Auto-generated method stub
    this.game="on";
    this.internet="on";
    this.playball=null;
}
```

将 game 和 internet 赋值为字符串“on”，其他变量赋值为 null，这时在浏览器中重新打开 index.jsp 文件，就会发现 game 和 internet 初始化为选中的状态，如图 6-13 所示。

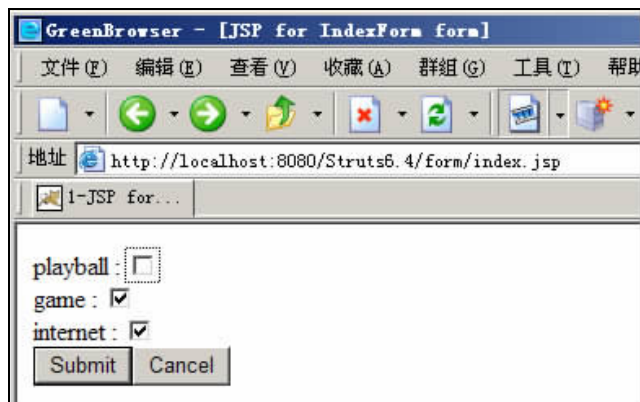


图 6-13 复选框初始化选中

在使用<html:checkbox>标签过程中,可以在 ActionForm 创建时,指定属性数据类型为 Boolean 类型,和使用 String 类型大同小异。本例使用 String 数据类型。

#### 6.5.5 在 Action 中通过数据库的数据控制<html:checkbox />复选框的选中状态

比如在开发一个 BBS 系统时,经常需要对用户的个人信息进行维护,开发这样的模块是大多数 BBS 必备的功能,那么怎么样通过从数据库中取出来的数据对 JSP 页面中的<html:checkbox>标签进行选中状态的控制呢?

比如个人信息模块需要维护一个“爱好”的 checkbox 的功能,这个“爱好”checkbox 也不是分组的情况,在本例中“爱好”的类型依然是 game、internet、playball,那么在 Action 中该怎么样处理才能同步数据库中的数据内容呢?

下面做一个实例来模拟这样的情况。

(1) 新建 JSP、Action、ActionForm,如图 6-14 所示。

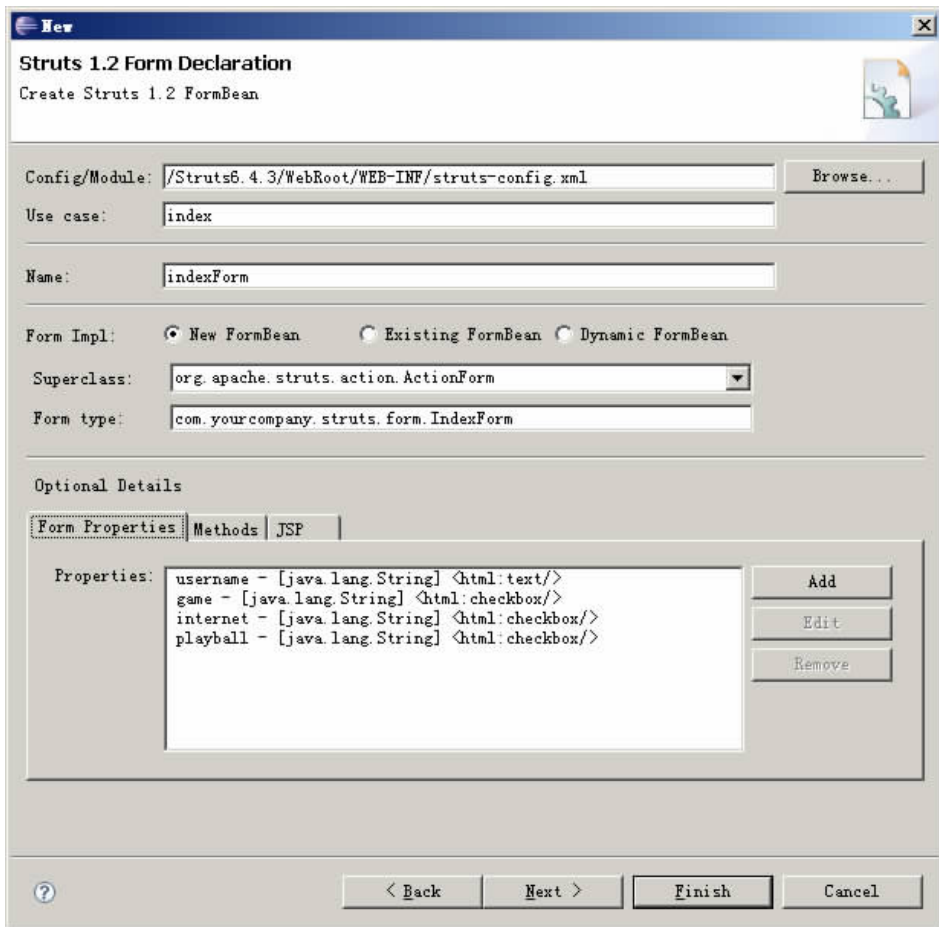


图 6-14 新建 JSP、Action、ActionForm

创建 JSP 文件如图 6-15 所示。



图 6-15 创建 JSP 文件

设置完成后单击“Next”按钮设置 Action，Action 使用默认设置，不需要更改任何的选项，单击“Finish”按钮应用设置。到这一步，注册功能的模块创建完成，本示例中假定数据表有 3 个字符串型字段 game、internet、playball。

## (2) 创建编辑模块。

再新建一个 JSP、Action、ActionForm，如图 6-16 所示。

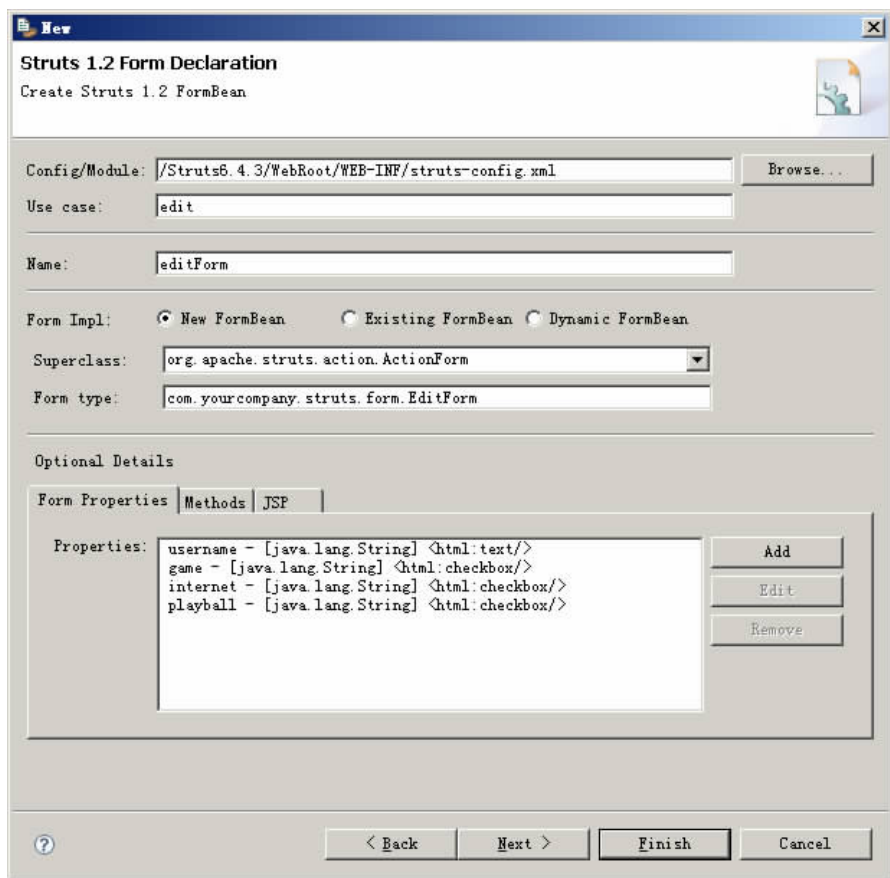


图 6-16 新建编辑模块

创建相对应的 JSP 文件，如图 6-17 所示。



图 6-17 新建编辑模块的 JSP 文件

在前面创建的/index 的 Action 映射的功能是注册，而/edit 的 Action 映射功能是编辑用户资料。

那么怎么样在 Action 中编写代码，才能在 edit.jsp 文件中显示出原来用户的信息，并且可以编辑修改呢？

(3) 更改 EditAction 类的程序代码如下：

```
public class EditAction extends Action {
    public ActionForward execute(ActionMapping mapping, ActionForm form,
        HttpServletRequest request, HttpServletResponse response) {
        EditForm editForm = (EditForm) form;
        // TODO Auto-generated method stub
        // 这里模拟通过 JDBC 的代码从数据表中的 3 个字段
        // 取出相应的 checkbox 字符串值
        // 本例中设置的 checkbox 的数据类型为 String
        // 所以 checkbox 的取值内容有 true/on/yes

        editForm.setGame("yes");
        editForm.setInternet("yes");
        editForm.setPlayball("no");
        return mapping.findForward("edit_page");
    }
}
```

在 EditAction 类中通过将 editForm 对象中的内容重新设置，然后转发 findForward 到 edit\_page 逻辑页，即可实现从数据库中控制页面的<html:checkbox>标签的选中状态。

(4) JSP 文件 edit.jsp 程序代码如下：

```
<%@ page language="java" pageEncoding="ISO-8859-1"%>
<%@ taglib uri="http://jakarta.apache.org/struts/tags-bean"
    prefix="bean"%>
<%@ taglib uri="http://jakarta.apache.org/struts/tags-html"
    prefix="html"%>

<html>
    <head>
        <title>JSP for EditForm form</title>
    </head>
```

```
<body>
  <html:form action="/edit">
    playball : <html:checkbox property="playball"/><html:errors
               property="playball"/><br/>
    game : <html:checkbox property="game"/><html:errors
           property="game"/><br/>
    username : <html:text property="username"/><html:errors
              property="username"/><br/>
    internet : <html:checkbox property="internet"/><html:errors
              property="internet"/><br/>
    <html:submit/><html:cancel/>
  </html:form>
</body>
</html>
```

(5) 配置文件 struts-config.xml 内容如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE struts-config PUBLIC "-//Apache Software Foundation//DTD Struts
Configuration 1.2//EN"
"http://struts.apache.org/dtds/struts-config_1_2.dtd">

<struts-config>
  <data-sources />
  <form-beans >
    <form-bean name="indexForm"
               type="com.yourcompany.struts.form.IndexForm" />
    <form-bean name="editForm"
               type="com.yourcompany.struts.form.EditForm" />
  </form-beans>

  <global-exceptions />
  <global-forwards >
    <forward name="edit_page" path="/form/edit.jsp" />
  </global-forwards>

  <action-mappings >
    <action
      attribute="indexForm"
      input="/form/index.jsp"
      name="indexForm"
      path="/index"
      scope="request"
      type="com.yourcompany.struts.action.IndexAction" />
    <action
      attribute="editForm"
      input="/form/edit.jsp"
      name="editForm"
      path="/edit"
```

```
scope="request"
type="com.yourcompany.struts.action.EditAction" />

</action-mappings>

<message-resources
parameter="com.yourcompany.struts.ApplicationResources" />
</struts-config>
```

(6) 部署项目，启动服务。

(7) 在地址栏中输入 `http://localhost:8080/Struts6.4.3/edit.do`。这时在浏览器的页面中通过数据库中的数据内容将 checkbox 设置为选中状态，本示例中设置了 `game` 和 `internet` 为选中状态，效果如图 6-18 所示。

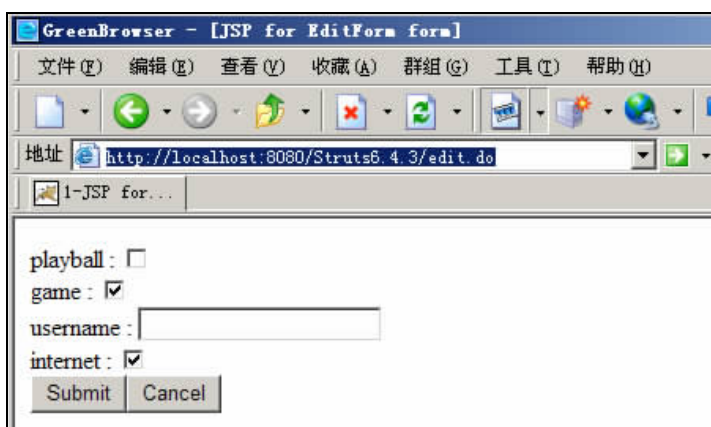


图 6-18 运行效果

在 `http://localhost:8080/Struts6.4.3/edit.do` 这个链接中可以设置用户的 ID 参数传递给 Action，Action 再通过 JDBC 的操作将对应 ID 的用户资料罗列出来，显示在 `edit.jsp` 文件中。

## 6.6 <html:errors /> 出错提示

是程序就有潜在的错误，就需要处理，需要提示给用户。`<html:errors>` 标签的作用就是显示一些提示及错误信息给用户。

在 Struts 中 `<html:errors>` 标签的使用还是非常普遍的，在以前的实例中都可以在 JSP 文件中看到它的身影，本小节着重介绍这个标签的两种使用方法。

### 6.6.1 显示局部错误信息

(1) 创建一个 JSP 文件，`index.jsp` 文件内容如下。

```
<%@ page language="java" pageEncoding="ISO-8859-1"%>
<%@ taglib uri="http://jakarta.apache.org/struts/tags-bean"
prefix="bean"%>
<%@ taglib uri="http://jakarta.apache.org/struts/tags-html"
```



```
prefix="html"%>

<html>
  <head>
    <title>JSP for IndexForm form</title>
  </head>
  <body>
    <html:form action="/index">
      username : <html:text property="username"/><html:errors
                  property="username"/><br/>
      <html:submit/><html:cancel/>
    </html:form>
  </body>
</html>
```

从上面的程序中可以看到，在 `<html:text property="username"/>` 的后面有代码：`<html:errors property="username"/>`，而且这个错误的标签链接的属性 `property` 为 `username`。这个 `property` 的值很重要，它是在 `Action` 中如果出错时要把错误显示在哪里的“路标”，`property` 属性的 `<html:errors/>` 标签是一条一条显示信息的。

资源文件 `ApplicationResources.properties` 内容如下。

```
showWrong=username is wrong!~
```

(2) 在本例中，当提交表单时，如果 `username` 表单中的数据内容不是 `gaohongyan` 时，就提示错误信息，否则转到其他的页面。

本例的 `Action` 类程序代码如下。

```
package com.yourcompany.struts.action;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import org.apache.struts.action.Action;
import org.apache.struts.action.ActionError;
import org.apache.struts.action.ActionErrors;
import org.apache.struts.action.ActionForm;
import org.apache.struts.action.ActionForward;
import org.apache.struts.action.ActionMapping;

import com.yourcompany.struts.form.IndexForm;

public class IndexAction extends Action {

    public ActionForward execute(ActionMapping mapping, ActionForm form,
        HttpServletRequest request, HttpServletResponse response) {
        IndexForm indexForm = (IndexForm) form; // TODO Auto-generated method
        // stub
        if (!indexForm.getUsername().equals("gaohongyan")) {
            System.out.println("false");
        }
    }
}
```

```
        ActionErrors error = new ActionErrors();
        error.add("username", new ActionError("showWrong"));
        this.saveErrors(request, error);
        return mapping.getInputForward();
    } else {
        System.out.println("true");
    }
    return null;
}
```

在 Action 中将传递过来的 username 变量取值，并判断是不是“gaohongyan”，如果不是的话，则创建一个 ActionErrors 对象来生成错误信息功能，并使用 add 方法来从资源文件中添加错误信息，最终使用 saveErrors 将错误保存到 request 中，并返回一个转发给 index.jsp。

关于代码：error.add("username", new ActionError("showWrong"));。

第 1 个参数指定 property 属性是 username 的<html:errors>标签中显示的信息，即<html:errors property="username"/>。

第 2 个参数新建一个错误对象。

通过这样的功能代码就可以实现在某一个<html:errors>中显示错误信息，这里就是指“局部”的，当然还有“全局”的。

### 6.6.2 显示全局错误信息

上一节介绍了“局部”显示的错误信息，当然还有“全局”显示错误信息的功能，步骤如下。

(1) 在 JSP 文件中增加如下代码：

```
<html:errors />
```

完整的 JSP 文件内容如下：

```
<%@ page language="java" pageEncoding="gb2312"%>
<%@ taglib uri="http://jakarta.apache.org/struts/tags-bean"
    prefix="bean"%>
<%@ taglib uri="http://jakarta.apache.org/struts/tags-html"
    prefix="html"%>

<html>
    <head>
        <title>JSP for IndexForm form</title>
    </head>
    <body>
        全局错误消息：#<html:errors />#。
        <html:form action="/index">
            write : <html:text property="write"/><html:errors
                property="write"/><br/>
```

```
<html:submit/><html:cancel/>
</html:form>
</body>
</html>
```

(2) 设置中文编码为 `pageEncoding="gb2312"`。

(3) 在 Action 中增加如下代码：

```
public ActionForward execute(ActionMapping mapping, ActionForm form,
    HttpServletRequest request, HttpServletResponse response) {
    IndexForm indexForm = (IndexForm) form; // TODO Auto-generated method
    // stub
    if (!indexForm.getWrite().equals("gaohongyan")) {
        System.out.println("false");
        ActionErrors error = new ActionErrors();
        error.add(error.GLOBAL_ERROR, new ActionError("showWrong"));
        this.saveErrors(request, error);
        return mapping.getInputForward();
    } else {
        System.out.println("true");
    }
    return null;
}
```

(4) 部署，启动 Web 服务。

(5) 浏览器中的效果如图 6-19 所示。



图 6-19 全局错误消息

在图中可以看到，可以在全局错误消息中写入任何的错误提示，只需要改动一条代码即可：

```
error.add(error.GLOBAL_ERROR, new ActionError("showWrong"));
```

由此可见，全局和局部的错误显示的转换还是非常方便的。

在“全局”错误提示这样的示例中，可以多次使用 `add` 方法来加入更多的错误提示信息。

### 6.6.3 生成错误信息在不同版本使用上的区别

在上面的例子中，如果非常留心的话，会发现在 Action 类中有一些代码是带“删除线”的，如果有“删除线”的代码，说明这些代码是过时的，有功能更好的类可以去代替实现。

在 Struts1.1 中使用 ActionErrors 和 ActionError 类。

在 Struts1.2 中使用 ActionMessages 和 ActionMessage 来替换 1.1 版本中相应的类。

## 6.7 <html:file />文件上传功能

在大多数的项目中，文件上传的功能很重要，比如留言板、BBS、Web 文件管理系统等这些常用文件上传功能的软件。

在 Struts 中可以使用<html:file />标签来实现文件的上传，由于 Struts 中有上传文件的组件类，在本例中只需要调用相应的功能类即可完成文件的上传。

### 6.7.1 定制 JSP 页面

新建一个 JSP 页面，设计代码如下：

```
<%@ page language="java" pageEncoding="ISO-8859-1"%>
<%@ taglib uri="http://jakarta.apache.org/struts/tags-bean"
    prefix="bean"%>
<%@ taglib uri="http://jakarta.apache.org/struts/tags-html"
    prefix="html"%>

<html>
  <head>
    <title></title>
  </head>
  <body>
    <html:form action="/update.do" enctype="multipart/form-data">
      <html:file property="theFile" />
      <html:submit />
    </html:form>
  </body>
</html>
```

新建一个 JSP 页面，里面包括<html:form>和一个<html:file>标签，在这里需要注意的是，要想实现文件上传功能，必须将<html:form>的 enctype 属性设置为 multipart/form-data。只有这样才能正确地识别要上传的是文件，并且针对这种上传文件的方式进行单独处理。

### 6.7.2 定制 ActionForm 类

新建一个 Form，代码如下。

```
package com.yourcompany.struts.form;

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;
import org.apache.struts.upload.FormFile;
import org.apache.struts.upload.MultipartRequestHandler;

public class IndexForm extends ActionForm {
    public static final String ERROR_PROPERTY_MAX_LENGTH_EXCEEDED =
        "org.apache.struts.webapp.upload.MaxLengthExceeded";

    protected FormFile theFile;

    public FormFile getTheFile() {
        return theFile;
    }

    public void setTheFile(FormFile theFile) {
        this.theFile = theFile;
    }

    public ActionErrors validate(ActionMapping mapping,
        HttpServletRequest request) {
        ActionErrors errors = null;
        // has the maximum length been exceeded?
        Boolean maxLengthExceeded = (Boolean) request
            .getAttribute(MultipartRequestHandler
                .ATTRIBUTE_MAX_LENGTH_EXCEEDED);
        if ((maxLengthExceeded != null) &&
            (maxLengthExceeded.booleanValue())) {
            errors = new ActionErrors();
            errors.add(ERROR_PROPERTY_MAX_LENGTH_EXCEEDED,
                new ActionError("maxLengthExceeded"));
        }
        return errors;
    }
}
```

在 Form 的 validate 方法中对文件上传功能进行一个提前预备。

### 6.7.3 设计重要的 Action 类

在上面的前台页面中，以及在 Form 中的简单性校验，最终的上传文件代码实现都要在 Action 类中进行，所以本例的 Action 作用很重要，代码如下。

```
package com.yourcompany.struts.action;

import java.io.ByteArrayOutputStream;
import java.io.FileOutputStream;
```

```
import java.io.InputStream;
import java.io.OutputStream;

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.upload.FormFile;
import com.yourcompany.struts.form.*;

public class IndexAction extends Action {
    public ActionForward execute(ActionMapping mapping,
                                ActionForm form,
                                HttpServletRequest request,
                                HttpServletResponse response)
        throws Exception {
        if (form instanceof IndexForm) {
            String encoding = request.getCharacterEncoding();
            if ((encoding != null) && (encoding.equalsIgnoreCase("utf-8")))
            {
                response.setContentType("text/html; charset=gb2312");
                //如果没有指定编码, 编码格式为 gb2312
            }
            IndexForm theForm = (IndexForm ) form;
            FormFile file = theForm.getTheFile(); //取得上传的文件
            try {
                InputStream stream = file.getInputStream(); //把文件读入
                String filePath = request.getRealPath("/"); //取当前系统路径
                // filePath = request.getRealPath(request.getRequestURI());
                //取当前系统路径
                ByteArrayOutputStream baos = new ByteArrayOutputStream();
                OutputStream bos = new FileOutputStream(filePath + "/" +
                    file.getFileName()); //建立一个上传文件的输出流
                //System.out.println(filePath+"/"+file.getFileName());
                int bytesRead = 0;
                byte[] buffer = new byte[8192];
                while ( (bytesRead = stream.read(buffer, 0, 8192)) != -1) {
                    bos.write(buffer, 0, bytesRead); //将文件写入服务器
                }
                bos.close();
                stream.close();
            } catch (Exception e) {
                System.err.print(e);
            }
            //request.setAttribute("dat",file.getFileName());
            return mapping.findForward("display");
        }
    }
}
```

```
        return null;
    }
}
```

将数据以 byte 数组的形式一点一点地、追加性地保存在服务器新建的文件中。保存成功后，findForward 进行界面的转发。

#### 6.7.4 查看 struts-config.xml 配置文件

配置文件内容如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE struts-config PUBLIC "-//Apache Software Foundation
//DTD Struts Configuration 1.2//EN"
"http://struts.apache.org/dtds/struts-config_1_2.dtd">

<struts-config>
    <data-sources />
    <form-beans>
        <form-bean name="indexForm"
                    type="com.yourcompany.struts.form.IndexForm" />
    </form-beans>

    <action-mappings>
    <action
        input="/form/index.jsp"
        name="indexForm"
        path="/update"
        scope="request"
        type="com.yourcompany.struts.action.IndexAction" />
    </action-mappings>

    <message-resources
        parameter="com.yourcompany.struts.ApplicationResources" />
</struts-config>
```

正确地配置后就可以在当前项目的根目录下找到刚才上传的文件，可以在 Action 类中通过更改创建文件时的路径来更改上传文件的保存路径。

## 6.8 <html:form /> 表单提交

在普通的 HTML 标记中的<form>，再到 Struts 中的<html:form>标签都在软件开发中起到了非常重要的作用，是数据提交处理必需的标记，当然，Struts 中的<html:form>标签属性却非常少，如图 6-20 所示。

在<html:form>的属性中比较重要的是 action、enctype、method、target。前 3 个属性在大多数的项目中能使用到，第 4 个属性 target 则是提交时窗口的状态。一起来做一个小实例。

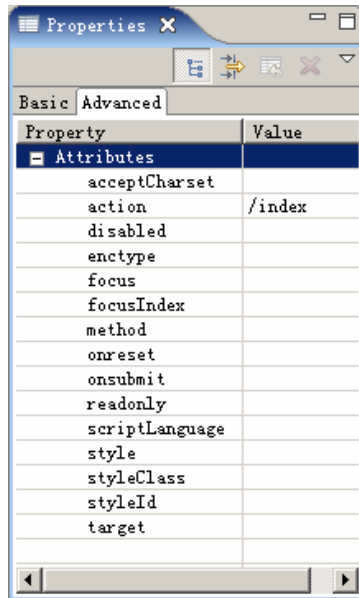


图 6-20 &lt;html:form&gt;属性

## 两个 JSP 页面的表单提交

(1) 设置 JSP 页面<html:form>的 target 属性：

```
<html:form action="/index" target="_blank">
    username : <html:text property="username" />
    <html:errors property="username" />
    <br />
    <html:submit />
    <html:cancel />
</html:form>
```

在这里将 action 的提交路径设置为/index，并且设置提交目标方式为\_blank，即新打开一个网页，然后两个 JSP 页面进行数据的提交交互。

(2) 设置 Action 文件内容如下。

```
public ActionForward execute(ActionMapping mapping, ActionForm form,
    HttpServletRequest request, HttpServletResponse response) {
    IndexForm indexForm = (IndexForm) form; // TODO Auto-generated method
    // stub
    System.out.println(indexForm.getUsername());
    return null;
}
```

这里只在 execute 方法中加入了打印传进来的 username 值的语句。

(3) 运行程序。

运行程序时单击“submit”提交按钮就会新打开一个浏览器窗口，并且在 MyEclipse 的控制台打印出所传递的文件内容。



## 6.9 <html:hidden />保密的数据传送

在前面学习过的表单标签中，大多数都是在 Web 页面中手动输入一些数据，然后提交再处理数据，但在大多数的情况下，还需要一个隐含式的表单来提交数据，在 Struts 中的 <html:hidden> 标签就是做这项工作的。

它和普通的 HTML 语言中的 <input type="hidden" name="hiddenField"> 标记功能一样。

在使用 Struts 的 <html:hidden> 标签的时候，同时也必须在 ActionForm 中定义它的域，而且需要提供 set 和 get 方法，以供 Action 类的处理。

### 6.9.1 开发实例

(1) 新建 Web 项目 Struts6.8.1。

(2) 添加属性。

在为 ActionForm 添加属性时，必须指定其类型为 hidden，如图 6-21 所示。

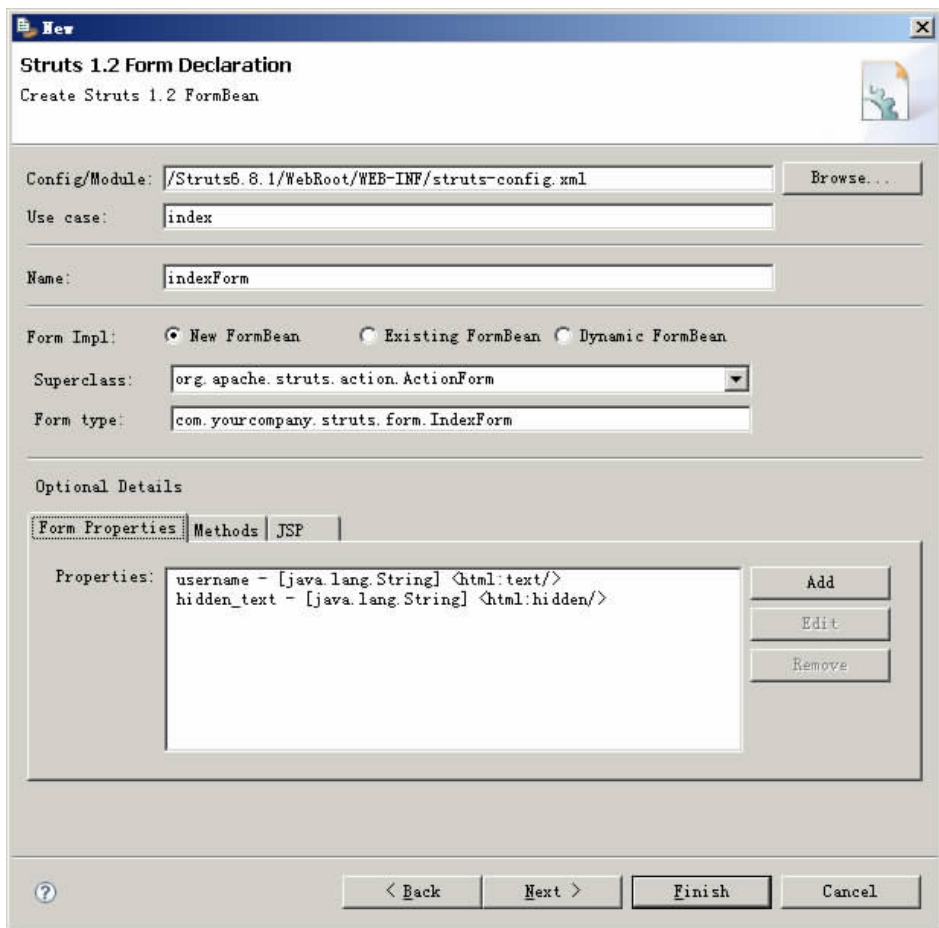


图 6-21 设置 hidden 类型

从图 6-21 中可以看出，设置了两个域：username 为<html:text>表单类型，另一个域 hidden\_text 为<html:hidden>表单类型。

(3) 在 JSP 属性页中创建 JSP 文件。

(4) 用默认的设置创建 Action。

(5) 单击“Finish”按钮完成设置。

(6) 在 ActionForm 中可以查看到 JSP 页面隐藏域所对应的实例变量，代码如下：

```
package com.yourcompany.struts.form;

import javax.servlet.http.HttpServletRequest;
import org.apache.struts.action.ActionErrors;
import org.apache.struts.action.ActionForm;
import org.apache.struts.action.ActionMapping;

public class IndexForm extends ActionForm {

    /** hidden_text property */
    private String hidden_text;

    /** username property */
    private String username;

    public ActionErrors validate(ActionMapping mapping,
        HttpServletRequest request) {
        // TODO Auto-generated method stub
        return null;
    }

    public void reset(ActionMapping mapping, HttpServletRequest request) {
        // TODO Auto-generated method stub
    }

    public String getHidden_text() {
        return hidden_text;
    }

    public void setHidden_text(String hidden_text) {
        this.hidden_text = hidden_text;
    }

    public String getUsername() {
        return username;
    }

    public void setUsername(String username) {
        this.username = username;
    }
}
```

在 ActionForm 中可以看到针对变量 hidden\_text 分别自动生成了 get 方法和 set 方法。

(7) JSP 页面内容如下：

```
<%@ page language="java" pageEncoding="ISO-8859-1"%>
<%@ taglib uri="http://jakarta.apache.org/struts/tags-bean"
    prefix="bean"%>
<%@ taglib uri="http://jakarta.apache.org/struts/tags-html"
    prefix="html"%>

<html>
  <head>
    <title>JSP for IndexForm form</title>
  </head>
  <body>
    <html:form action="/index">
      hidden_text : <html:hidden property="hidden_text"
                                value="<%= "10"%>" />

      <html:errors property="hidden_text" />
      <br />
      username : <html:text property="username" />
      <html:errors property="username" />
      <br />
      <html:submit />
      <html:cancel />
    </html:form>
  </body>
</html>
```

<html:hidden>标签必须放在<html:form>中。

在 JSP 文件中将<html:hidden>动态赋值为数字 10。

这时单击“提交”按钮后就可以处理传出去的隐藏域了。

(8) 在 Action 中显示这个隐藏域的值。

Action 中的程序代码如下：

```
public ActionForward execute(ActionMapping mapping, ActionForm form,
    HttpServletRequest request, HttpServletResponse response) {
    IndexForm indexForm = (IndexForm) form;
    // TODO Auto-generated method stub
    System.out.print(indexForm.getHidden_text());
    return null;
}
```

可以看到在 Action 中只添加了一个 println 语句，用来输出隐藏域的值。

(9) 部署项目，启动服务。

(10) 单击“提交”按钮后，可以在 MyEclipse 的控制台中输入字符串“10”，这个值即是隐藏域的值。

### 6.9.2 <html:hidden>如何设置默认值

在 JSP 页面中，通常都需要设置<html:hidden>标签的一个默认值，下面提供 5 种方法来实现这样的功能。

#### (1) 创建新的 Java Bean

在该方法中，<html:hidden>默认值的来源是<bean:define>定义的 Bean 的某一个属性值，如下面的程序代码：

```
<bean:define id="para" name="xxbean" property="xxproperty">
<html:hidden property="key" value="<%=para%>" />
```

上面的程序代码首先从 request 或 session 中取出 name 为 xxbean 的 Java Bean 实例，然后再取得其 xxproperty 属性值作为 para 变量的值，相当于将 xxbean Bean 中的 xxproperty 属性的值赋给变量 para，这样在下一行程序代码<html:hidden>中即可将 value 属性赋给 para 变量。

#### (2) 通过<bean:write>实现

例如，

```
<html:hidden property="key" value="<bean:write name="xxbean"
property="xxproperty">" />
```

<html:hidden>的默认值是从<bean:write>标签而来，而<bean:write>标签输出的是名字为 xxbean 的 Bean 的 xxproperty 属性。

#### (3) JSP 代码实现

例如：

```
<% String para =(String) pageText.getAttribute("你放进去那个 BEAN 属性");%>
<html:hidden property="key" value="<%=para%>" />
```

其实，这个例子和第(1)点中的例子大体一样，只不过在第(1)点的例子中将数据源放到了 request 或 session 中，并且数据类型使用的是 Java Bean，而在这里使用的数据源存放位置却是 pageText，即生命周期是本页面，并且数据类型是简单的字符串类型。

#### (4) 从 request 或 session 的 Bean 中获取

例如：

```
<html:hidden name="JavaBean" property="JavaBean 属性" />
```

Struts 标签的通用性很强，这里使用的是另外一种方法，即使用<html:hidden>的 name 和 property 属性来取得默认值，这里面 name 指的是存放在 session 或 request 中的对象别名，而 property 即是这个对象的某一个属性名称，使用这样的方法就没有必要使用<html:hidden>标签的 value 属性。建议使用这种方法来设置默认值。

#### (5) 在 ActionForm 中设置初始值

在基于 MVC 的 Struts 框架中，前台 JSP 页面有 HTML 类型的 Struts 标签时，就需要

在相对应的 ActionForm 中加入属性（域），这时就可以在访问 JSP 页面时，自动执行 ActionForm 中的 reset() 方法，而在 reset() 方法中可以针对 <html:hidden> 进行赋值。

例如：

```
public class TestForm extends ActionForm {  
    /*  
     * Generated fields  
     */  
  
    private String hidden;  
  
    public void reset(ActionMapping mapping, HttpServletRequest request) {  
        hidden="hidden_value";  
    }  
    .....  
}
```

而其相对应的 JSP 文件测试内容为：

```
<body>  
    <html:form action="/test">  
        hidden : <html:hidden property="hidden"/>  
                <html:errors property="hidden"/><br/>  
        <html:submit/><html:cancel/>  
    </html:form>  
</body>
```

上例是在 ActionForm 中对隐藏域赋予默认值，当然也可以在 Action 类中通过 set 方法对隐藏域进行赋值，这时访问的路径就是 \*.do 的形式，即先访问 Action，然后再转发到相应的 JSP 页面。

## 6.10 <html:html> 定义 HTML 文件

本标记与普通的 HTML 语言中的 <html> 标记功能一样。  
在 <html:html> 属性中主要有 3 个属性，如图 6-22 所示。

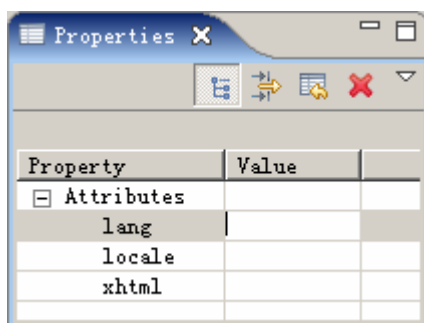


图 6-22 属性列表

- lang 属性的作用是当设置为 true 后，若没有 session 对象时，就根据 HTTP 请求中的 locale 信息来输出相应的语言信息，而不再强制创建一个新的 session。
- locale 属性在 Struts 1.2 中已经被 lang 所替代。
- xhtml 属性是定义本页是否遵守 xhtml 规范。

## 6.11 <html:image>定义图像提交按钮

在一些项目中，通常为了页面的美观程度，使用图片来代替方方正正的提交按钮，在 Struts 中可以使用<html:image>标签来实现。

图像提交按钮实例

(1) 在 JSP 页面中加入<html:image>标签，JSP 文件代码如下：

```
<body>
  <html:form action="/index">
    username : <html:text property="username" />
    <html:errors property="username" />
    <br />
    <html:submit />
    <html:cancel />
    <html:image page="/form/submit.bmp" property="submitBMP" />
  </html:form>
</body>
```

在<html:image>标签的属性 page 中设置图像的路径，property 是图像元素的名称 name。

(2) 在 Action 类中打印 username，代码如下：

```
public ActionForward execute(ActionMapping mapping, ActionForm form,
    HttpServletRequest request, HttpServletResponse response) {
    IndexForm indexForm = (IndexForm) form; // TODO Auto-generated method
    // stub
    System.out.println(indexForm.getUsername());
    return null;
}
```

(3) 部署项目，启动服务。

(4) 在浏览器中运行效果如图 6-23 所示。



图 6-23 运行效果

当前 index.jsp 文件的源代码如下。

```
<body>
  <form name="indexForm" method="post" action="/Struts6.11/index.do">
    username : <input type="text" name="username" value="">

    <br />
    <input type="submit" value="Submit">
    <input type="submit" name="org.apache.struts.taglib.html.CANCEL"
      value="Cancel" onclick="bCancel=true;">
    <input type="image" name="submitBMP"
      src="/Struts6.11/form/submit.bmp">

  </form>
</body>
```

可以看到<html:image>标签被转换为：

```
<input type="image" name="submitBMP" src="/Struts6.11/form/submit.bmp">
```

的形式。

单击图像按钮后，就完成了相当于单击 submit 按钮的提交效果。

如果想美化 Web 界面，那么请将方方正正的提交按钮用<html:image>来代替。

## 6.12 <html:img>在页面上显示图像

此标签的使用非常简单，它主要使用 page 属性。如：<html:img page="" />。

在 page 属性中设置图片的相对路径。例如：<html:img page="/form/submit.bmp" />。

在一些 BBS 的编辑用户资料中，可以在编辑时显示用户曾经选择的头像，这时就需要动态地将 page 赋值了，下面一起来做一个例子吧！

使用<html:img>标签动态改变用户头像

(1) 创建 Struts 6.11.1 项目，并且在 WebRoot 目录中放入两个 BMP 头像文件，如图 6-24 所示。

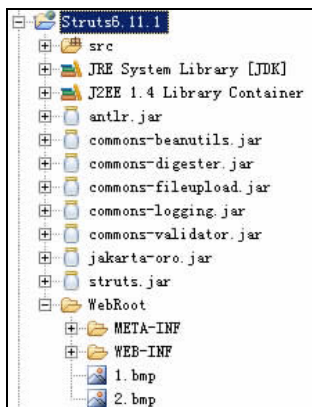


图 6-24 项目初期结构

(2) 新建一个 JSP 文件和一个 Action 类，JSP 文件内容代码如下。

```
<%@ page language="java" pageEncoding="gb2312"%>

<%@ taglib uri="http://struts.apache.org/tags-bean" prefix="bean"%>
<%@ taglib uri="http://struts.apache.org/tags-html" prefix="html"%>
<%@ taglib uri="http://struts.apache.org/tags-logic" prefix="logic"%>
<%@ taglib uri="http://struts.apache.org/tags-tiles" prefix="tiles"%>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html:html lang="true">
<head>
    <html:base />

    <title>toEditHeadPic.jsp</title>

    <meta http-equiv="pragma" content="no-cache">
    <meta http-equiv="cache-control" content="no-cache">
    <meta http-equiv="expires" content="0">
    <meta http-equiv="keywords" content="keyword1,keyword2,keyword3">
    <meta http-equiv="description" content="This is my page">
    <!--
    <link rel="stylesheet" type="text/css" href="styles.css">
    -->

</head>

<body>
    <%
        personalInfo abc = (personalInfo) request.
                                getAttribute("personalInfo");
        String path = abc.getHeadPic();
    %>
    <html:img page="<%=path%>" />
</body></html:html>
```

因为 Struts 的<html:img>标签不能嵌套<bean:write>标签，所以使用一种折中的办法解决这个问题。

(3) 创建 Action 类的程序代码内容如下：

```
package com.yourcompany.struts.action;

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;
```



```

public class ToEditHeadPicAction extends Action {

    public ActionForward execute(ActionMapping mapping, ActionForm form,
        HttpServletRequest request, HttpServletResponse response) {
        // 通过 JDBC 的操作, 找到 id 为 888 用户的全部信息, 包括头像字段、
        // 存放头像的 URI 路径
        System.out.println(request.getParameter("id"));
        // 假如用户以前曾经使用的头像是 1.bmp, 则在页面中显示 1.bmp 图片
        // 这里模拟 JDBC 的 DAO 操作, 将取出来的用户记录封装成一个用户信息的 Bean,
        // 然后再将这个 Bean 传到 JSP 页面中, 直接取出相应的属性即可

        personal.personalInfo headpic_ref = new personal.personalInfo();
        headpic_ref.setHeadPic("/1.bmp");
        request.setAttribute("personalInfo", headpic_ref);
        return mapping.findForward("newHeadpic");
    }
}

```

在 Action 中的 personalInfo 类是自定义包 personal 中的类, 负责将从数据库中提取到的记录字段中的内容封装成 Bean, 这里为了简短起见, 只有一个属性 headPic。

(4) personalInfo 类的程序代码如下。

```

package personal;

public class personalInfo {
    private String headPic = "";

    public void setHeadPic(String headPic) {
        this.headPic = headPic;
    }

    public String getHeadPic() {
        return headPic;
    }
}

```

(5) 配置文件 struts-config.xml 程序代码内容如下。

```

<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE struts-config PUBLIC "-//Apache Software Foundation
    //DTD Struts Configuration 1.2//EN"
    "http://struts.apache.org/dtds/struts-config_1_2.dtd">

<struts-config>
    <data-sources />
    <form-beans />
    <global-exceptions />
    <global-forwards >
        <forward name="newHeadpic" path="/show.jsp" />
    </global-forwards>
</struts-config>

```

```
</global-forwards>

<action-mappings>
  <action path="/toEditHeadPic"
          type="com.yourcompany.struts.action.ToEditHeadPicAction" />
</action-mappings>

<message-resources
  parameter="com.yourcompany.struts.ApplicationResources" />
</struts-config>
```

(6) 部署项目，启动服务。

(7) 程序运行效果如图 6-25 所示。



图 6-25 运行效果

(8) 单击超级链接后，出现正确的图片，效果如图 6-26 所示。



图 6-26 显示正确的头像

显示出正确的图片了，那么下一步可以通过 JSP 代码，当单击

猫咪的头像还是非常可爱的，这个实例结束了。

## 6.13 <html:link>变幻莫测的超级链接

<html:link>标签是生成 HTML 的<a>超级链接标记。

在 Web 开发中，超级链接是将 Web 组件与 Web 组件连接起来的最有效方法。

当然，使用超级链接去连接不同类型的组件、不同位置的组件，在使用时的方法也不一样。

在 JSP 页面上插入一个<html:link>标签，然后在 Properties 面板中出现了该标签的主要属性，如图 6-27 所示。

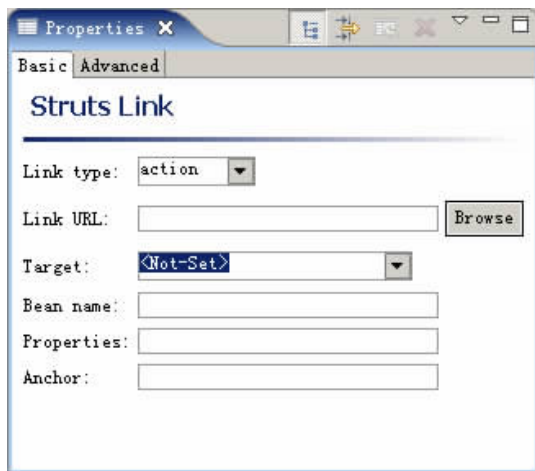


图 6-27 <html:link>主要属性

其中 Link type 属性尤为重要，它可以设置为 4 种，如图 6-28 所示。

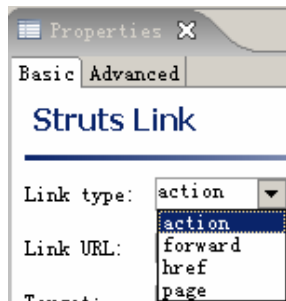


图 6-28 设置 Link type 类别

- forward：指定一个全局 ActionForward 的名称。另外如果 forward 是与 module 相关的，则该 forward 必须指向一个 action 而不能是一个页面。
- action：指定一个 Action 的名称。
- href：绝对路径，Struts 会直接使用这个值而不会对其进行任何处理。
- page：指定一个页面的相对路径，必须以 / 开始。

### 6.13.1 Link type 为 action 的情况

将 Link type 设置为 action 的情况是当单击超级链接后,希望转到 Action 类中进行功能代码的执行,在 Action 中实现一些功能。在大多数情况下,都将 action 设置为一个 ForwardAction 类,即转发,更换页面的内容,这样直接通过控制器就转到相应的 JSP 页面上,也可以直接指定到全局 Forward 上。

本实例模拟简单的投票功能,单击超级链接后执行一组相关的功能代码,然后再转发回到刚才的 index.jsp 页面。

(1) 新建 Web 工程 Struts6.12.1。

(2) 新建 JSP 页面,代码如下。

```
<%@ page language="java" pageEncoding="gb2312"%>
<%@ taglib uri="http://jakarta.apache.org/struts/tags-bean"
    prefix="bean"%>
<%@ taglib uri="http://jakarta.apache.org/struts/tags-html"
    prefix="html"%>

<html>
  <head>
    <title>JSP for IndexForm form</title>
  </head>
  <body>
    <html:form action="/index">
      username : <html:text property="username" />
      <html:errors property="username" />
      <br />
      <html:submit />
      <html:cancel />
    </html:form>
    <br />
    <html:link action="/toTouPiao.do">单击此链接进行投票</html:link>
  </body>
</html>
```

在<html:link>中设置 action 为一个路径映射,单击超级链接后转到/toTouPiao.do 路径,执行 Action 代码,也可以在这个<html:link>的 action 中写入参数,比如当前被投票人的数据库 ID 为 8 的情况:<html:link action="/toTouPiao.do?id=8">。

(3) 新建 Action, 名称为 toTouPiao, 代码如下。

```
public ActionForward execute(ActionMapping mapping, ActionForm form,
    HttpServletRequest request, HttpServletResponse response) {
    System.out.println("执行投票 JDBC 代码!~");
    return null;
}
```

(4) struts-config.xml 文件内容如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE struts-config PUBLIC "-//Apache Software Foundation
//DTD Struts Configuration 1.2//EN"
"http://struts.apache.org/dtds/struts-config_1_2.dtd">

<struts-config>
  <data-sources />
  <form-beans>
    <form-bean name="indexForm"
      type="com.yourcompany.struts.form.IndexForm" />
  </form-beans>

  <global-exceptions />
  <global-forwards />
  <action-mappings>
    <action attribute="indexForm" input="/form/index.jsp"
      name="indexForm" path="/index" scope="request"
      type="com.yourcompany.struts.action.IndexAction" />
    <action path="/toTouPiao"
      type="com.yourcompany.struts.action.ToTouPiaoAction" />
  </action-mappings>

  <message-resources
    parameter="com.yourcompany.struts.ApplicationResources" />
</struts-config>
```

在本例中所关心的是代码：

```
<action path="/toTouPiao"
  type="com.yourcompany.struts.action.ToTouPiaoAction" />
```

通过这个 action 映射，在单击超级链接时可以做任何的功能实现。

(5) 部署项目，启动服务。

运行效果如图 6-29 所示。



图 6-29 运行效果

单击超级链接后，在 MyEclipse 控制台上输出了字符串“执行投票 JDBC 代码！~”。

### 6.13.2 Link type 为 forward 的情况

超级链接的作用就是转换页面的功能,但在 Struts 的基于 MVC 模式中,如果不通过控制器来进行调配的话,显然不会遵守 MVC 规范,如果想遵守这个规范,那么就可以使用 `<html:link>` 的 `forward` 属性来通过控制器进行页面的转换了。

(1) 新建 JSP 页面 `index.jsp`, 内容如下。

```
<body>
    <html:link forward="toOtherPage">toOtherPage</html:link>
</body>
```

在 JSP 文件代码中,可以看到超级链接的 `forward` 指向了一个叫 `toOtherPage` 的 Forward 对象名称。

(2) 在 `struts-config.xml` 文件中创建这个 `forward`。`struts-config.xml` 创建结束后,代码如下。

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE struts-config PUBLIC "-//Apache Software Foundation//DTD Struts
Configuration 1.2//EN"
    "http://struts.apache.org/dtds/struts-config_1_2.dtd">

<struts-config>
    <data-sources />
    <form-beans />
    <global-exceptions />
    <global-forwards>
        <forward name="toOtherPage" path="/other.jsp" />

    </global-forwards>

    <action-mappings />
    <message-resources
        parameter="com.yourcompany.struts.ApplicationResources" />
</struts-config>
```

值得注意的是, `<html:link>` 的 `forward` 属性只引用 `Struts-config.xml` 配置文件中 `<global-forwards>` 全局内的 `<forward>` 子元素,如果引用 `<action>` 内的局部 `<forward>` 子元素,在运行时将会抛出异常:

```
Cannot create rewrite URL: Java.Net.MalformedURLException: Cannot retrieve
ActionForward
```

(3) 新建一个 `other.jsp` 文件,内容自定义。

(4) 部署项目,启动服务。

程序运行效果如图 6-30 所示。



图 6-30 项目运行效果

查看源程序后，可以看到当前的 index.jsp 转换成 HTML 语言的源程序，`<html:link>` 已经被转换成正确路径的 `<a>` 标记了，代码如下。

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html lang="zh-CN">
<head>
  <base href="http://localhost:8080/Struts6.12.2/index.jsp">

  <title>index.jsp</title>

  <meta http-equiv="pragma" content="no-cache">
  <meta http-equiv="cache-control" content="no-cache">
  <meta http-equiv="expires" content="0">
  <meta http-equiv="keywords" content="keyword1,keyword2,keyword3">
  <meta http-equiv="description" content="This is my page">
  <!--
  <link rel="stylesheet" type="text/css" href="styles.css">
  -->

</head>

<body>
  <a href="/Struts6.12.2/other.jsp">toOtherPage</a>
</body>
</html>
```

### 6.13.3 Link type 为 href 的情况

在 Web 项目中，所有的超级链接不可能都是本站的相对路径跳转，也有可能是其他的站点的 URL 路径，这时就得使用绝对路径来进行页面的转换，最典型的就是网站上的友情链接。本例要说明的就是使用 `<html:link>` 来实现绝对路径的超级链接功能。

(1) 新建 JSP 文件，添加如下代码：

```
<body>
  <html:link href="http://www.sohu.com">www.sohu.com</html:link>
</body>
```

(2) 部署项目，启动服务。

本例效果如图 6-31 所示。



图 6-31 绝对路径超级链接

查看 HTML 代码如下。

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html lang="zh-CN">
<head>
  <base href="http://localhost:8080/Struts6.12.3/index.jsp">

  <title>index.jsp</title>

  <meta http-equiv="pragma" content="no-cache">
  <meta http-equiv="cache-control" content="no-cache">
  <meta http-equiv="expires" content="0">
  <meta http-equiv="keywords" content="keyword1,keyword2,keyword3">
  <meta http-equiv="description" content="This is my page">
  <!--
  <link rel="stylesheet" type="text/css" href="styles.css">
  -->

</head>

<body>
<a href="http://www.sohu.com">www.sohu.com</a>
</body>
</html>
```

Struts 从<html:link>转换到<a>的标记很顺利。

#### 6.13.4 Link type 为 page 的情况

有些时候,可以不通过 MVC 的控制器直接转到另外一个 Web 组件中,比如 JSP 页面,这时可以使用 page 属性来进行相对路径的跳转。

(1) 新建 1.jsp 文件,内容如下。

```
<body>
  <html:link page="/2.jsp">to 2.jsp</html:link>
</body>
```

(2) 部署项目,启动服务后,单击浏览器中的超级链接后就转到 2.jsp 页面了,很简单吧。



### 6.13.5 带参数超级链接的情况

#### 1. 带固定参数的情况

如果要在 URL 或 URI 中包含请求参数,只要把请求参数加在 URL 或 URI 的末尾就可以了。例如:

```
<html:link page="/HtmlBasic.do?prop1=abc&prop2=123">
    Hard-code the url parameters
</html:link>
```

以上代码生成如下 HTML 内容:

```
<a href=/lib/HtmlBasic.do?prop1=abc&prop2=123">.....</a>
```

提示:在 HTML 中,&代表特殊字符"&"。

#### 2. 超级链接的参数值来自变量的情况

如果要在 URL 中包含一个请求参数,而这个参数的值存在于当前网页可访问的一个变量中,这样的情况怎么办?可以按以下方法来实现。

模拟这种情况,首先创建一个当前网页可访问的变量。例如,本例中创建了两个变量,一个是字符类型,另一个是 CustomerBean,它们存在于一个 page 生命周期范围内。

模拟代码 JSP 页面如下。

```
<%
    String stringBean = "Value to Pass ont URL";
    pageContext.setAttribute("stringBean", stringBean);
%>
<jsp:useBean id = "customerBean" scope="page"
              class="htmltaglibs.beans.CurstomerBean"/>
<jsp:setProperty name="customerBean" property="name" value="weiqin"/>
```

接着,把这两个变量作为请求参数,加入到 URL 或 URI 中。

```
<html:link page="/HtmlBasic.do"
           paramId="urlParamName"
           paramName="stringBean">
abc
</html:link>

<html:link page="/HtmlBasic.do"
           paramId="urlParamName"
           paramName="customerBean"
           paramProperty="name">
xyz
</html:link>
```

注意：针对前面的两个程序段实例，多了一个 paramProperty 属性，这里的 name 属性和 Property 属性与 bean:write 标签一样，一般来说 name 的值是一个被绑定在 request、session、application 或 page 范围内的对象的绑定 key 值，Property 属性的值则是这个对象的一个成员变量的名称，即属性。使用了 name 和 Property 之后，Struts 将在上述 4 种范围内，以 name 变量值为 key 查找对象，并将对象读出后，取出 Property 定义的成员变量的值。

<html:link>标签的 paramId 属性指定请求参数名，paramName 属性指定变量的名字。如果变量为 Java Bean，用 paramProperty 属性指定 Java Bean 的属性。

对于本例的 stringBean，请求参数值为 stringBean 的字符串值。对于 customerBean，指定了 paramProperty 属性，请求参数值为 customerBean 的 name 属性值。

以上代码生成如下 HTML 内容。

```
<a href="/HtmlBasic.do?urlParamName=Value to Pass on Url">
  abc
</a>

<a href="/HtmlBasic.do?urlParamName=weiqin">
  xyz
</a>
```

### 3. 超级链接多参数值来自 Map 对象变量的情况

如果要定义多个参数，就需要手动连接 URL，这就失去了使用标签的意义了。

如果在 URL 或 URI 中包含多个请求参数，而这些参数的值来自多个变量，需要先定义一个 Map 类型的 Java 类，如 java.util.HashMap，用它来存放请求变量。

例如：

```
<%
    java.util.HashMap myMap = new java.util.HashMap();
    myMap.put("myString", new String("myStringValue"));
    myMap.put("myArray", new String[]{"str1","str2","str3"} );
    pageContext.setAttribute("map", myMap);
%>
```

在以上代码的 HaspMap 中存放了两个对象 其中第二个对象是个字符串数组。HashMap 被存放在 PageContext 中。接下来就可以把这个 HashMap 作为请求参数，加入到 URL 或 URI 中。

```
<html:link page="/HtmlBasic.do" name="map">
  URL encode a parameter based on value in a Map
</html:link>
```

<html:link>标签的 name 属性指定包含请求变量的 HashMap 对象。HashMap 对象中的每一对“ key/value ”代表一对或多对“ 请求参数名/请求参数值 ”。以上代码生成如下的 HTML 内容。

```
<a href="/HtmlBasic.do?myString=myStringValue&myArray=str1&myArray=str2&myArray=str3">
    URL encode a parameter based on value in a Map
</a>
```

### 6.13.6 在<html:link>中嵌入 JSP 脚本

在一些情况下，有时在<html:link>标签中嵌入 JSP 脚本（Scriptlet），例如：

```
<html:link page="/test.do?code=<%=variable%>">add</html:link>
```

这种写法是错误的，无法编译。

上面的写法改成：

```
<html:link page="<%=>/test.do?code="+variable%>">add</html:link>
```

就可以被执行，但是要注意 URL 相对路径的问题。

虽然在 Struts 标签中嵌入 JSP 脚本不是真正意义上的 Struts 应用，但是有时在委曲求全的情况下也只能如此了，除非使用自定义标签。比如在 Form 表单中可能需要根据具体数据让某个字段是只读的，就可以用嵌入 JSP 脚本来实现。

```
<%
boolean rdonly=false;
if(2==2) rdonly=true;
%>
<html:text property="userid" readonly="<%=rdonly%>" />
```

### 6.13.7 带参数超级链接的问题解决实例

在上面的小节中，一起讨论了 3 种超级链接带参数的情况。

- 带固定参数的情况
- 超级链接的参数值来自变量的情况
- 超级链接多参数值来自 Map 对象变量的情况

在本小节中，将通过一个非常简单的实例，用代码来实现这 3 种情况的功能。

新建一个 ghyPackage 的包，然后新建 testBean 类，类代码如下。

```
package ghyPackage;
public class testBean {
    private String username = "suoyingxiu";

    public void setUsername(String username) {
        this.username = username;
    }

    public String getUsername() {
        return username;
    }
}
```

新建 JSP 文件，内容如下。

```
<%@ page language="java" pageEncoding="gb2312"%>

<%@ taglib uri="http://struts.apache.org/tags-bean" prefix="bean"%>
<%@ taglib uri="http://struts.apache.org/tags-html" prefix="html"%>
<%@ taglib uri="http://struts.apache.org/tags-logic" prefix="logic"%>
<%@ taglib uri="http://struts.apache.org/tags-tiles" prefix="tiles"%>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html:html lang="true">
<head>
    <html:base />

    <title>index.jsp</title>

    <meta http-equiv="pragma" content="no-cache">
    <meta http-equiv="cache-control" content="no-cache">
    <meta http-equiv="expires" content="0">
    <meta http-equiv="keywords" content="keyword1,keyword2,keyword3">
    <meta http-equiv="description" content="This is my page">
    <!--
    <link rel="stylesheet" type="text/css" href="styles.css">
    -->

</head>

<body>
    <html:link

href="http://www.sohu.com/showtext1.jsp?id=88&name=gaohongyan">
        带固定参数的情况</html:link>
    </br>

    <%
        String test = "gaohongyan";
        request.setAttribute("valueinto", test);
    %>
    <html:link href="http://www.sohu.com/showtext2.jsp" paramId="id"
        paramName="valueinto">超级链接的参数值从变量来的情况</html:link>
    </br>

    <jsp:useBean id="customerBean" scope="page"
        class="ghyPackage.testBean" />
    <jsp:setProperty name="customerBean" property="username"
        value="xiaosuo" />
    <html:link href=http://www.sohu.com/showtext3.jsp
        paramId="username"
        paramName="customerBean" paramProperty="username">
```

超级链接的参数值来自 Bean 中的属性的情况，代码如下。

```
</html:link>
</br>

<%
    java.util.HashMap myMap = new java.util.HashMap();
    myMap.put("myString", new String("myStringValue"));
    myMap.put("myArray", new String[] { "str1", "str2", "str3" });
    pageContext.setAttribute("map", myMap);
%>
<html:link href="http://www.sohu.com/showtext4.jsp" name="map">
```

超级链接的参数值来自 Map 中的属性的情况，代码如下。

```
</html:link>
</body>
</html:html>
```

程序运行后的 HTML 代码如下：

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html lang="zh-CN">
<head>
    <base href="http://localhost:8080/Struts6.12.7/index.jsp">

    <title>index.jsp</title>

    <meta http-equiv="pragma" content="no-cache">
    <meta http-equiv="cache-control" content="no-cache">
    <meta http-equiv="expires" content="0">
    <meta http-equiv="keywords" content="keyword1,keyword2,keyword3">
    <meta http-equiv="description" content="This is my page">
    <!--
    <link rel="stylesheet" type="text/css" href="styles.css">
    -->

</head>

<body>
    <a href="http://www.sohu.com/showtext1.jsp?id=88&name=gaohongyan">带固定参数的情况</a>
    </br>

    <a href="http://www.sohu.com/showtext2.jsp?id=gaohongyan">超级链接的参数值来自变量的情况</a>
    </br>

    <a href="http://www.sohu.com/showtext3.jsp?username=xiaosuo">超级链接的参数值来自 Bean 中的属性的情况</a>
    </br>
```

```
<a href="http://www.sohu.com/showtext4.jsp?myArray=str1&myArray=str2&myArray=str3&myString=myStringValue">
    超级链接的参数值来自 Map 中的属性的情况</a>
</body>
</html>
```

### 6.13.8 给 Struts 中<html:link>标签加确认对话框

示例如下面代码所示。

```
<html:link action="newsEdit.do?do=deleteNews" paramName="news"
paramProperty="id" paramId="id">Delete</html:link>
```

在上面的代码中，希望单击超级链接后出现一个对话框，这样的情况怎么办？其实 Struts 对 JavaScript 脚本的支持还是非常好的，改正如下。

```
<html:link action="newsEdit.do?do=deleteNews" paramName="news"
paramProperty="id" paramId="id" onclick="javascript:if(!confirm('test'))
return false;return true;"
>Delete</html:link>
```

这样，当单击链接后出现一个对话框就可以实现上述功能了，加上这个功能，鼠标的“误操作”将会大大减少，也使软件的交互性增强。

### 6.13.9 用<html:link>标签生成 BBS 主题列表功能

在本实例中，将用实际的程序来实现在 BBS 项目中生成主题列表的功能，列出主题列表后通过单击主题的超级链接而显示文章的内容。

(1) 新建一个 Web 项目 Struts 6.12.9。

(2) 新建一个贴子标题信息类，在该类中写入贴子的一些相关的数据，比如发帖者、回复次数等，程序代码如下。

```
package titleInfo;

public class titleInfos {
    private String id = ""; // 贴子在数据表中的 ID

    private String username = ""; // 发帖者

    private String title = ""; // 标题内容

    private String title_begindate = ""; // 发帖时间

    private String title_uptime = ""; // 回复次数

    private String title_looktime = ""; // 查看次数

    public void setUsername(String username) {
        this.username = username;
    }
}
```

```
}

public String getUsername() {
    return username;
}

public void setTitle(String title) {
    this.title = title;
}

public String getTitle() {
    return title;
}

public void setTitle_begindate(String title_begindate) {
    this.title_begindate = title_begindate;
}

public String getTitle_begindate() {
    return title_begindate;
}

public void setTitle_uptime(String title_uptime) {
    this.title_uptime = title_uptime;
}

public String getTitle_uptime() {
    return title_uptime;
}

public void setTitle_looktime(String title_looktime) {
    this.title_looktime = title_looktime;
}

public String getTitle_looktime() {
    return title_looktime;
}

public void setId(String id) {
    this.id = id;
}

public String getId() {
    return id;
}
}
```

从程序中可以看到，该 class 类存放在 package titleInfo 包中。

(3) 创建一个 Action 类，在 Action 中通过 JDBC 的代码将数据库中的标题内容及一些相关的数据存放在 titleInfos 类中，然后再添加到一个 arraylist 对象中，最后放入 request

对象中，传给前台的 JSP 页面，从而显示给用户，Action 程序代码如下：

```
/*
 * Generated by MyEclipse Struts
 * Template path: templates/java/JavaClass.vtl
 */
package com.yourcompany.struts.action;

import java.util.ArrayList;

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 titleInfo.titleInfos;

public class TitleListAction extends Action {

    public ActionForward execute(ActionMapping mapping, ActionForm form,
        HttpServletRequest request, HttpServletResponse response) {
        // 在本 Action 类的 execute() 方法中通过 JDBC 的操作将 title 标题的内容
        // 从数据库中提取出来后封装成 titleInfos 类，然后再放到 request 对象中
        // 这里只是模拟性的代码，不存在 JDBC 的操作
        titleInfos title_ref1 = new titleInfos();
        title_ref1.setId("1");
        title_ref1.setTitle("2007 年的第一场雪");
        title_ref1.setTitle_begindate("2007-2-19");
        title_ref1.setTitle_looktime("999");
        title_ref1.setTitle_uptime("1000");
        title_ref1.setUsername("高红岩");

        titleInfos title_ref2 = new titleInfos();
        title_ref2.setId("2");
        title_ref2.setTitle("2008 年的第一场雪");
        title_ref2.setTitle_begindate("2008-2-19");
        title_ref2.setTitle_looktime("999");
        title_ref2.setTitle_uptime("1000");
        title_ref2.setUsername("王雷");

        titleInfos title_ref3 = new titleInfos();
        title_ref3.setId("3");
        title_ref3.setTitle("2009 年的第一场雪");
        title_ref3.setTitle_begindate("2009-2-19");
        title_ref3.setTitle_looktime("999");
        title_ref3.setTitle_uptime("1000");
        title_ref3.setUsername("刘桂云");
    }
}
```



```
titleInfos title_ref4 = new titleInfos();
title_ref4.setId("4");
title_ref4.setTitle("2010 年的第一场雪");
title_ref4.setTitle_begindate("2010-2-19");
title_ref4.setTitle_looktime("999");
title_ref4.setTitle_uptime("1000");
title_ref4.setUsername("高风文");

ArrayList arraylist = new ArrayList();
arraylist.add(title_ref1);
arraylist.add(title_ref2);
arraylist.add(title_ref3);
arraylist.add(title_ref4);

request.setAttribute("titleInfo_List", arraylist);

return mapping.findForward("showTitle");
}
```

(4) 转发给 `return mapping.findForward("showTitle")` 后就需要在 JSP 页面中显示出标题的内容了, JSP 文件 `show.jsp` 程序代码如下。

```
<%@ page language="java" pageEncoding="gb2312"%>

<%@ taglib uri="http://struts.apache.org/tags-bean" prefix="bean"%>
<%@ taglib uri="http://struts.apache.org/tags-html" prefix="html"%>
<%@ taglib uri="http://struts.apache.org/tags-logic" prefix="logic"%>
<%@ taglib uri="http://struts.apache.org/tags-tiles" prefix="tiles"%>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html:html lang="true">
<head>
  <html:base />

  <title>show.jsp</title>

  <meta http-equiv="pragma" content="no-cache">
  <meta http-equiv="cache-control" content="no-cache">
  <meta http-equiv="expires" content="0">
  <meta http-equiv="keywords" content="keyword1,keyword2,keyword3">
  <meta http-equiv="description" content="This is my page">
  <!--
  <link rel="stylesheet" type="text/css" href="styles.css">
  -->

</head>

<body>
```

```
<table width="857" height="58" border="1">
  <tr>
    <td width="135">
      <div align="center">
        贴子 ID
      </div>
    </td>
    <td width="172">
      <div align="center">
        标题内容
      </div>
    </td>
    <td width="132">
      <div align="center">
        发贴者
      </div>
    </td>
    <td width="172">
      <div align="center">
        发贴时间
      </div>
    </td>
    <td width="172">
      <div align="center">
        回复次数
      </div>
    </td>
    <td width="101">
      <div align="center">
        查看次数
      </div>
    </td>
  </tr>

  <logic:iterate id="showTitle" name="titleInfo_List">
    <tr>
      <td>
        <bean:write name="showTitle" property="id" />
      </td>
      <td>
        <a href="showtitle.jsp?id=<bean:write
          name="showTitle" property="id" />"><bean:write
            name="showTitle" property="title" /> </a>
      </td>
      <td>
        <bean:write name="showTitle" property="username" />
      </td>
      <td>
        <bean:write name="showTitle"
          property="title_begindate" />
      </td>
    </tr>
  </logic:iterate>
</table>
```

```

        </td>
        <td>
            <bean:write name="showTitle"
                property="title_uptime" />
        </td>
        <td>
            <bean:write name="showTitle"
                property="title_looktime" />
        </td>
    </tr>
</logic:iterate>
</table>
</body>
</html:html>

```

根据 taglib 规范,嵌套在<html:link>内的<bean:write/>根本就不会被解释,作为字符串原样输出。程序通过在 JSP 文件中使用<logic: iterate>循环标签将 arraylist 中的内容显示出来。

当然也可以更改创建超链的源代码,实现不用嵌套即可使用<html:link>标签,而不使用<a>创建超级链接的功能,程序代码如下。

```

<td>
    <html:link page="/showtitle.jsp" paramId="id"
        paramName="showTitle"
        paramProperty="id">
        <bean:write name="showTitle" property="title" />
    </html:link>
</td>

```

(5) 相关的 struts-config.xml 配置文件内容如下。

```

<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE struts-config PUBLIC "-//Apache Software Foundation
    //DTD Struts Configuration 1.2//EN"
    "http://struts.apache.org/dtds/struts-config_1_2.dtd">

<struts-config>
    <data-sources />
    <form-beans />
    <global-exceptions />
    <global-forwards >
        <forward name="showTitle" path="/show.jsp" />
    </global-forwards>

    <action-mappings >
<action path="/titleList"
        type="com.yourcompany.struts.action.TitleListAction" />
    </action-mappings>

```

```
<message-resources
parameter="com.yourcompany.struts.ApplicationResources" />
</struts-config>
```

(6) 部署项目，启动服务。

(7) 在浏览器的地址栏中输入 `http://localhost:8080/Struts6.12.9/titleList.do` 地址后，按下回车键，显示如图 6-32 所示的效果。



图 6-32 运行效果

提示：在本实例中，使用[简单的 HTML 标记超级链接](#)来进行贴子内容的详细显示。但为了节省篇幅，本实例省略单击标题超级链接后显示详细贴子内容的步骤。

但如果在本例中的超级链接符合 MVC 规范的话，笔者建议使用 `<html:link forward>` 的形式来进行贴子内容的显示，这样不仅在浏览器地址栏中的扩展名为 `*.do`，隐藏了实际显示贴子内容的 JSP 文件名，而且还有利于以后的项目维护。

## 6.14 <html:errors>的更新版<html:messages>

本小节中的程序代码是模拟性的代码。

在讨论 `<html:messages>` 之前要先复习一下 ActionForm 和 Action 相关的知识，所有的出错信息都在这两个类中生成及处理。

ActionForm 是 JSP 页面元素的对象化，它将 Web 页面数据的完整性检查工作放在其中进行，例如使用者是否填写了所有的字段，ActionForm 中所有的属性是否被设定了，还可以重新定义 ActionForm 的 `validate()` 方法来进行这项工作，如下面程序代码所示。

```
import javax.servlet.http.*;
import org.apache.struts.action.*;

public class UserForm extends ActionForm {
    protected String name;
    protected String password;
```

```
public void setName(String name) {
    this.name = name;
}
public void setPassword(String password) {
    this.password = password;
}
public String getName() {
    return name;
}
public String getPassword() {
    return password;
}
public void reset(ActionMapping mapping, HttpServletRequest req) {
    name = null;
    password = null;
}

public ActionErrors validate(ActionMapping mapping,
                             HttpServletRequest request) {
    ActionErrors errors = new ActionErrors();

    if(getName() == null || getName().length() < 1) {
        errors.add("name", new ActionError("error.name.required"));
    }
    if(getPassword() == null || getPassword().length() < 1) {
        errors.add("password",
                  new ActionError("error.password.required"));
    }

    return errors;
}
```

当操作者提交表单，而一些表单域中的内容字段没有填写时，则请求中会包括参数名称，但是值为空字符串，如果 ActionForm 具有某些属性，而 Web 页面并没有发送对应的参数，则不会设定 ActionForm 中对应的属性，这些属性将为 null，方法 validate() 主要检查这两种情况。

validate() 方法会传回 ActionErrors 对象，ActionErrors 可以存储 ActionError 的出错信息，是一个保存出错信息的容器。每一个 ActionError 会查询资源文件中的 key-value 对应值，当 validate() 返回 ActionErrors 对象时，ActionServlet 就不会继续进行接下来的工作，而是进入 struts-config.xml 所设定的映射，如下面程序代码所示。

```
<global-forwards>
    <forward
        name="welcome"
        path="/Welcome.do"/>
</global-forwards>
```

```
<form-beans>
  <form-bean
    name="userForm"
    type="onlyfun.caterpillar.UserForm"/>
</form-beans>

<action-mappings>
  <action
    path="/Welcome"
    type="org.apache.struts.actions.ForwardAction"
    parameter="/pages/Welcome.jsp"/>

  <action
    path="/LoginAction"
    type="onlyfun.caterpillar.LoginAction"
    name="userForm"
    validate="true"
    input="/pages/Welcome.jsp">
    <forward name="greeting" path="/pages/greeting.jsp"/>
  </action>
</action-mappings>
```

为了使用 `validate()` 方法，`<action>` 中的 `validate` 属性必须设定为 `true`，而 `input` 属性也是必要的，当 `validate()` 传回 `ActionErrors` 时，就会传送至 `input="/pages/Welcome.jsp"` 所设定的位置，`ActionErrors` 中的错误信息，可以使用 `<html:errors/>` 标签的这个形式来显示。

`ActionForm` 中验证了属性为 `null` 及空字符串的可能，这是数据完整性的验证，接下来要验证数据的正确性，是否符合所设定的名称与密码，更改 `Action` 类 `LoginAction`，如下面程序代码所示。

```
import javax.servlet.http.*;
import org.apache.struts.action.*;
import org.apache.commons.beanutils.*;

public class LoginAction extends Action {
    public ActionForward execute(ActionMapping mapping,
                                ActionForm form,
                                HttpServletRequest request,
                                HttpServletResponse response)
        throws Exception {

        String name = (String) PropertyUtils.getSimpleProperty(form,
            "name");
        String password = (String) PropertyUtils.getSimpleProperty(form,
            "password");

        if(!(name.equals("caterpillar") && password.equals("1234"))){
            ActionMessages messages = new ActionMessages();
            messages.add(ActionMessages.GLOBAL_MESSAGE,
```

```
        new ActionMessage("message.namepass.notmatched"));
        saveMessages(request, messages);
        return mapping.findForward("welcome");
    }
    else {
        request.getSession().setAttribute("valid_user", form);
        return mapping.findForward("greeting");
    }
}
```

在这个程序中使用了 `org.apache.commons.beanutils` 中的 `PropertyUtils` 类别来协助获取 `ActionForm` 中的值，优点是不用考虑 `ActionForm` 真正的类型，`PropertyUtils` 会自动帮我们判断，`getSimpleProperty()` 传回的是 `Object`，而要将之转换为 `String`。

`ActionMessages` 是 Struts 1.1 所新增的类，它变成了 `ActionErrors` 的父类，同样，`ActionMessage` 也是 Struts 1.1 新增的类别，它是 `ActionError` 的父类别，数据的格式与完整性检查在 `ActionForm` 中已经验证了，接下来在 `Action` 中检查是否符合名称与密码，如果不符合就加入相关的错误信息。

在 Struts 1.1 中特意将 `Message` 与 `Error` 有所区别，所谓的 `Error` 指使用者的输入在完整性或格式等上有误，而 `Message` 是指输入的数据基本上没有错误，但不能符合后续的处理要求。

为了能够显示错误与提示，必须在 `application_zh.properties` 中加入 key-value 对应值，如下面代码所示。

```
# -- error --
error.name.required=没有输入名称
error.password.required=没有输入密码

#-- message --
message.namepass.notmatched=名称与密码不正确
```

为了能使用中文，记得使用 `native2ascii.exe` 工具进行转换，接下来看看 `Welcome.jsp` 是如何设计的，要注意 `<html:errors/>` 与 `<html:messages/>` 的使用，程序代码如下。

```
<%@ taglib uri="/tags/struts-bean" prefix="bean" %>
<%@ taglib uri="/tags/struts-html" prefix="html" %>
<%@page contentType="text/html; charset=Big5"%>
<html:html locale="true">
<head>
<title><bean:message key="welcome.title"/></title>
<html:base/>
</head>
<body bgcolor="white">
<html:errors/>

<html:messages id="messages" message="true">
    <bean:write name="messages"/>
</html:messages>
```

```
<h3>请登录</h3>

<html:form action="/Login">
    名称:<html:text property="name" size="20"/><br>
    密码:<html:password property="password" size="20"/><br>
    <html:submit/> <html:reset/>
</html:form>

</body>
</html:html>
```

如果由于 ActionForm 传回 ActionErrors 对象而返回到 Welcome.jsp 中, 则<html:errors/>标签会显示 ActionErrors 中的相关错误信息, 这时利用<html:messages/>来检查返回的内容是否也包括 ActionMessages 对象, 如果有的话就取出并使用<bean:write/>标签显示出来。

下面是程序运行时未填写字段所显示的错误信息的一个例子, 已被转换成 HTML 源程序, 程序代码如下。

```
<html lang="zh">
<head>
<title>哈啰! Struts! </title>
<base href="http://localhost:8080/HelloStruts/pages/Welcome.jsp">
</head>
<body bgcolor="white">
<UL>
<LI>没有输入名称
</LI><LI>没有输入密码
</LI></UL>

<h3>请登入</h3>

<form name="UserForm" method="post" action="/HelloStruts/Login.do">
    名称:<input type="text" name="name" size="20" value=""><br>
    密码:<input type="password" name="password" size="20" value=""><br>
    <input type="submit" value="Submit"> <input type="reset" value="Reset">
</form>

</body>
</html>
```

注意到 ActionErrors 在 Struts 1.2 之后会被标示为 deprecated (过时), 将来可能会以 ActionMessages 取代, 所以<html:errors/>在将来必须用下面的方式来取代。

```
<html:messages id="msg" >
    <bean:write name="msg"/>
</html:messages>
```

在之前的例子中, 在<html:messages/>的属性上设定 message 为 true, 这表示显示 ActionMessages 的内容, 程序代码如下。



```
<html:messages id="messages" message="true">
  <bean:write name="messages"/>
</html:messages>
```

### <html:messages>显示信息的实例

在本节将做一个关于标签<html:messages>的实例，在本实例中将对 ActionForm 和 Action 的出错信息分别进行处理，并在页面上显示。

(1) 新建一个 Form、JSP、Action，将 case 名称设置为 index。

(2) 在 ActionForm 中设置出错提示代码如下。

```
public ActionErrors validate(ActionMapping mapping,
    HttpServletRequest request) {
    // TODO Auto-generated method stub
    ActionErrors errors = new ActionErrors();
    if (!username.equals("gaohongyan")) {
        errors.add(ActionMessages.GLOBAL_MESSAGE, new ActionMessage(
            "usernamewrong"));
    }
    return errors;
}
```

上面代码的功能是如果发现传进来的用户名不是“gaohongyan”就进行全局性的错误提示，并且返回一个 Errors 对象，而且 Action 的代码也不执行。

(3) 更改 Action 的程序代码如下。

```
public ActionForward execute(ActionMapping mapping, ActionForm form,
    HttpServletRequest request, HttpServletResponse response) {
    IndexForm indexForm = (IndexForm) form; // TODO Auto-generated method
    // stub

    ActionErrors messages = new ActionErrors();
    if (!indexForm.getPassword().equals("xyz")) {
        messages.add(ActionMessages.GLOBAL_MESSAGE,
            new ActionMessage("passwordwrong"));
        this.addMessages(request, messages);
        return mapping.getInputForward();
    }

    return null;
}
```

上面这段程序是在 Action 中判断传进来的密码是不是为“xyz”，如果不是则创建一个错误提示对象 messages，然后再通过 addMessages 方法来保存到 request 对象中，再进行转发。

(4) 在资源文件中更改内容如下。

```
usernamewrong=your username is wrong!~
passwordwrong=your password is wrong!~
```

(5) 更改 JSP 页面代码如下。

```
<%@ page language="java" pageEncoding="ISO-8859-1"%>
<%@ taglib uri="http://jakarta.apache.org/struts/tags-bean"
    prefix="bean"%>
<%@ taglib uri="http://jakarta.apache.org/struts/tags-html"
    prefix="html"%>

<html>
    <head>
        <title>JSP for IndexForm form</title>
    </head>
    <body>
        <html:form action="/index">
            password : <html:text property="password" />
            <html:errors property="password" />
            <br />
            username : <html:text property="username" />
            <html:errors property="username" />
            <br />
            <html:submit />
            <html:cancel />
        </html:form>

        <html:messages id="errors">
            <bean:write name="errors" />
        </html:messages>

        <html:messages id="messages" message="true">
            <bean:write name="messages" />
        </html:messages>
    </body>
</html>
```

在 JSP 页面中显示出错误提示。其中<html:messages>的 id 属性是在 Action 或 ActionForm 中的 ActionMessages 对象的名称,然后再通过<bean:write>打印出错误对象中的错误信息。

<html:messages>的 message 属性如果不设定为 true,会输出 ActionErrors 中所存储的错误提示,Errors 代表的是一个操作方面的错误,例如错误操作导致使用者名称或密码为空(当然也许是用户故意设置的)。

<html:messages>的 message 属性如果设定为 true,会输出 ActionMessages 中所存储的信息,Messages 表示一个提示信息,也许使用者输入了不正确的信息,例如在输入名称与密码时打错了字,程序会提示使用者输入了不正确的信息。

## 6.15 <html:multibox />分组类型的复选框

开发者经常需要在页面上设计出分组式的复选框，比如下面例子使用 HTML 语言的 checkbox 表单的“爱好”分组复选框。

```
<form name="form1" method="post" action="">
  <input name="aihao" type="checkbox" id="aihao" value="internet">
  上网
  <input name="aihao" type="checkbox" id="aihao" value="game">
  打游戏
  <input name="aihao" type="checkbox" id="aihao" value="playball">
  玩足球
  <input name="aihao" type="checkbox" id="aihao" value="lookbook">
  看书
</form>
```

在这里，4 个复选框的名字（name）都为 aihao（爱好），而它们的 value 却不同，但在 Struts 中怎么样设计出这样的分组复选框呢？

使用<html:multibox>标签即可实现这样的功能。

### 6.15.1 用<html:multibox />做一个选择“爱好”的实例

（1）新建一个 JSP 文件，文件内容如下。

```
<%@ page language="java" pageEncoding="gb2312"%>

<%@ taglib uri="http://struts.apache.org/tags-bean" prefix="bean"%>
<%@ taglib uri="http://struts.apache.org/tags-html" prefix="html"%>
<%@ taglib uri="http://struts.apache.org/tags-logic" prefix="logic"%>
<%@ taglib uri="http://struts.apache.org/tags-tiles" prefix="tiles"%>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html:html lang="true">
<head>
  <html:base />

  <title>index.jsp</title>

  <meta http-equiv="pragma" content="no-cache">
  <meta http-equiv="cache-control" content="no-cache">
  <meta http-equiv="expires" content="0">
  <meta http-equiv="keywords" content="keyword1,keyword2,keyword3">
  <meta http-equiv="description" content="This is my page">
  <!--
  <link rel="stylesheet" type="text/css" href="styles.css">
  -->
```

```
</head>

<body>

    <html:form action="/multibox.do" method="post">
        <html:multibox property="aihao" value="internet" />上网<br />
        <html:multibox property="aihao" value="game" />打游戏<br />
        <html:multibox property="aihao" value="playball" />玩球<br />
        <html:submit />
    </html:form>
</body>
</html:html>
```

在 JSP 文件中加入如下的程序代码。

```
<html:form action="/multibox.do" method="post">
    <html:multibox property="aihao" value="internet" />上网<br />
    <html:multibox property="aihao" value="game" />打游戏<br />
    <html:multibox property="aihao" value="playball" />玩球<br />
    <html:submit />
</html:form>
```

在这里使用<html:multibox>标签来实现分组复选框，property 是 ActionForm 的属性名称，属性的数据类型必须是数组。

(2) 新建一个 Action，并写入如下的程序。

```
public ActionForward execute(ActionMapping mapping, ActionForm form,
    HttpServletRequest request, HttpServletResponse response) {
    DynaActionForm multibox = (DynaActionForm) form;
    // TODO Auto-generated

    // method stub
    Object obj = multibox.get("aihao");
    String[] checkList = (String[]) obj;
    for (int i = 0; i < checkList.length; i++)
        System.out.println(checkList[i]);

    return null;
}
```

在该程序中，通过使用动态 DynaActionForm 类的方法，来取得 aihao 中所有被选项的 value 值，然后再循环打印在 MyEclipse 的控制台中。

(3) 配置 struts-config.xml。

为了使用动态 DynaActionForm 及 action 映射功能，还要更改配置文件如下。

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE struts-config PUBLIC "-//Apache Software Foundation
    //DTD Struts Configuration 1.2
    //EN" "http://struts.apache.org/dtds/struts-config_1_2.dtd">
```

```
<struts-config>
  <data-sources />
  <form-beans >
<form-bean name="multibox"
           type="org.apache.struts.action.DynaActionForm">
  <form-property name="aihao" type="java.lang.String[]" />
</form-bean>

</form-beans>

<global-exceptions />
<global-forwards />
<action-mappings >
  <action
    attribute="multibox"
    name="multibox"
    path="/multibox"
    scope="request"
    type="com.yourcompany.struts.action.MultiboxAction" />
</action-mappings>

  <message-resources
parameter="com.yourcompany.struts.ApplicationResources" />
</struts-config>
```

在<form-property>中，其 aihao 的数据类型为 String[]字符串数组类型。

(4) 部署应用程序，启动服务。

运行效果如图 6-33 所示。



图 6-33 运行效果

选择复选框，然后单击“Submit”按钮，在 MyEclipse 的控制台中查看结果。

#### 6.15.2 <html:multibox />初始化时即呈 checked 状态

(1) 新建 JSP、Action、Form，如图 6-34 所示。

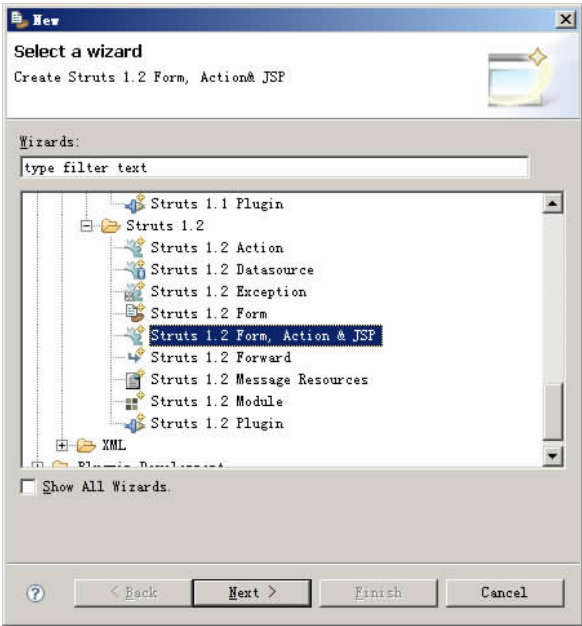


图 6-34 创建 JSP、Action、Form

(2) 单击“Next”按钮后，在弹出的窗口中按如图 6-35 所示进行设置。

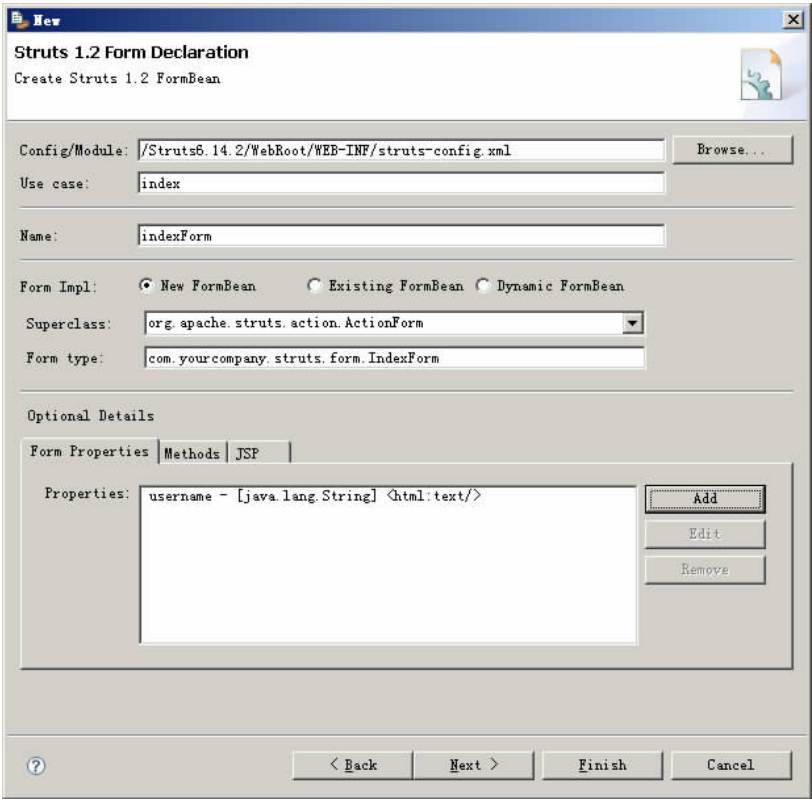


图 6-35 设置 ActionForm

(3) 设置 JSP 属性页生成 JSP 文件，如图 6-36 所示。

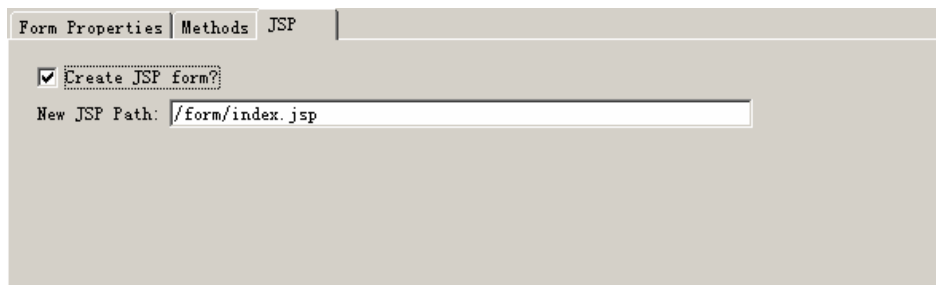


图 6-36 创建 JSP 页面

(4) 单击“Next”按钮后，使用默认的设置进行配置 Action，单击“Finish”按钮应用创建。

(5) 更改 JSP 文件内容如下。

```
<%@ page language="java" pageEncoding="gb2312"%>
<%@ taglib uri="http://jakarta.apache.org/struts/tags-bean"
    prefix="bean"%>
<%@ taglib uri="http://jakarta.apache.org/struts/tags-html"
    prefix="html"%>

<html>
  <head>
    <title>JSP for IndexForm form</title>
  </head>
  <body>
    <html:form action="/index">
      username : <html:text property="username" />
      <html:errors property="username" />
      <br />
      <html:submit />
      <html:cancel />
    </br>
    <html:multibox property="aihao" value="internet" />上网<br />
    <html:multibox property="aihao" value="game" />打游戏<br />
    <html:multibox property="aihao" value="playball" />玩球<br />
    </html:form>
  </body>
</html>
```

在<html:form>中添加了分组复选框代码：

```
<html:multibox property="aihao" value="internet" />上网<br />
<html:multibox property="aihao" value="game" />打游戏<br />
<html:multibox property="aihao" value="playball" />玩球<br />
```

(6) 更改 ActionForm 代码来实现初始化选中状态，程序代码如下。

```
package com.yourcompany.struts.form;

import javax.servlet.http.HttpServletRequest;
import org.apache.struts.action.ActionErrors;
import org.apache.struts.action.ActionForm;
import org.apache.struts.action.ActionMapping;

public class IndexForm extends ActionForm {

    /** username property */
    private String username;

    private String[] aihao = { "internet", "game" };

    public ActionErrors validate(ActionMapping mapping,
        HttpServletRequest request) {
        // TODO Auto-generated method stub
        return null;
    }

    public void reset(ActionMapping mapping, HttpServletRequest request) {
        // TODO Auto-generated method stub
    }

    public String getUsername() {
        return username;
    }

    public void setUsername(String username) {
        this.username = username;
    }

    public void setAihao(String[] aihao) {
        this.aihao = aihao;
    }

    public String[] getAihao() {
        return aihao;
    }
}
```

在 ActionForm 中添加了类变量，并且进行值初始化：

```
private String[] aihao = { "internet", "game" };
```

这句代码的功能就是初始化分组复选框中哪个复选框为选中的状态，这里在打开页面时就将 internet 和 game 复选框呈现选中的状态。

(7) 自定义在 Action 中写入处理数据的代码。

(8) 部署项目，启动服务。

运行效果如图 6-37 所示。





图 6-37 运行效果显示

## 6.16 用<html:select />和<html:option />实现下拉列表

Struts 中的下拉列表标签必须嵌套在<html:form>标签中，包括如下。

- <html:select>
- <html:option>
- <html:options>
- <html:optionsCollection>

使用时嵌套如下：

```
<html:select property="ActionForm.property">
<html:option>或者<html:options>或者<html:optionsCollection>
</html:select>
```

其中 property 为 ActionForm 中对应的一个属性。

### 6.16.1 用<html:select />和<html:option />实现下拉列表

通过<html:select />和<html:option />标签的合作，则可以实现在 HTML 语言中的下拉列表的功能。

HTML 语言中的下拉列表代码如下。

```
<select name="select">
  <option value="0432">吉林市</option>
  <option value="0431">长春市</option>
  <option value="024">沈阳</option>
</select>
```

使用 Struts 中的<html:select />和<html:option />标签实现下拉列表代码如下。

```
<html:select property="select_property">
  <html:option value="0432">吉林</html:option>
  <html:option value="0431">长春</html:option>
  <html:option value="024">沈阳</html:option>
```

```
</html:select>
```

<html:select>标签中的 property 属性是 ActionForm 中的类变量名称，即属性名称。从程序代码来看，和普通的 HTML 语言中的<select>标记实现方法非常类似。

### 6.16.2 <html:select />和<html:option />实现列表单选

在 HTML 语言中实现列表单选的源代码如下。

```
<select name="select" size="5">
  <option value="0432">吉林市</option>
  <option value="0431">长春市</option>
  <option value="024">沈阳</option>
</select>
```

上面的程序代码通过在<select>标记加了一个 size 属性后即可呈现出单选下拉列表表项的形式，如图 6-38 所示。

使用 Struts 中的<html:select>标签也可以实现列表单选，程序代码如下。

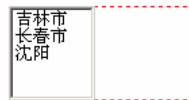


图 6-38 列表效果

```
<html:select property="select_property" value="024" size="3">
  <html:option value="0432">吉林</html:option>
  <html:option value="0431">长春</html:option>
  <html:option value="024">沈阳</html:option>
</html:select>
```

运行效果如图 6-39 所示。



图 6-39 使用<html:select>标签实现单选下拉列表表项

### 6.16.3 <html:select />和<html:option />实现列表多选

如果想在列表中实现多项选择的效果，这样的 HTML 实现代码如下。

```
<select name="select" size="5" multiple>
  <option value="0432">吉林市</option>
  <option value="0431">长春市</option>
  <option value="024">沈阳</option>
</select>
```

从程序中可以看到在原来加入 size 的基础上，还加入了属性 multiple，这样的代码即

可实现多项选择的功能，如图 6-40 所示。



图 6-40 多项选择列表表项

相应地使用 Struts 的 `<html:select>` 标签该如何实现列表多选呢？例程如下。

```
<html:select property="select_property" value="024"
             multiple="true" size="3">
  <html:option value="0432">吉林</html:option>
  <html:option value="0431">长春</html:option>
  <html:option value="024">沈阳</html:option>
</html:select>
```

从程序代码上来看，将 `multiple` 属性设置为 `true` 后，即可实现多选列表表项的效果。运行效果如图 6-41 所示。



图 6-41 使用 `<html:select>` 实现多选列表表项

#### 6.16.4 `<html:select />` 和 `<html:option />` 标签设置下拉列表初始值

在有些情况下，在打开页面的时候就需要设置 `<html:select>` 标签生成的下拉列表的初始值，这样的情况该如何处理呢？很简单，程序代码如下。

```
<html:select property="select_property" value="024">
  <html:option value="0432">吉林</html:option>
  <html:option value="0431">长春</html:option>
  <html:option value="024">沈阳</html:option>
</html:select>
```

程序运行效果如图 6-42 所示。



图 6-42 设置下拉列表的初始化值

6.16.5 <html:select />和<html:option />设置列表单选初始化值

例程如下。

```
<html:select property="select_property" value="024" size="3">
  <html:option value="0432">吉林</html:option>
  <html:option value="0431">长春</html:option>
  <html:option value="024">沈阳</html:option>
</html:select>
```

程序运行效果如图 6-43 所示。



图 6-43 列表单选初始化值

6.16.6 <html:select />和<html:option />设置列表多选初始化值

当<html:select>中存在 multiple="true"属性时，其<html:select>与 ActionForm 相对应的类变量必须为数组类型，如果想初始化列表多选，提供如下两种方法。

- 使用 JSP 代码设置下拉列表中的值选择。
- 在 ActionForm 中设置数组的初始值，然后通过 Action 进行 Forward 即可。

下面使用第 2 种方法来实现列表多选的初始化值。

- (1) 新建 Web 项目 Struts 6.15.6。
  - (2) 新建一个 JSP、Action、ActionForm，如图 6-44 所示。
- 在 JSP 属性页中创建 JSP 文件后，单击“Next”按钮使用默认的设置配置 Action，单

击“Finish”按钮应用设置。

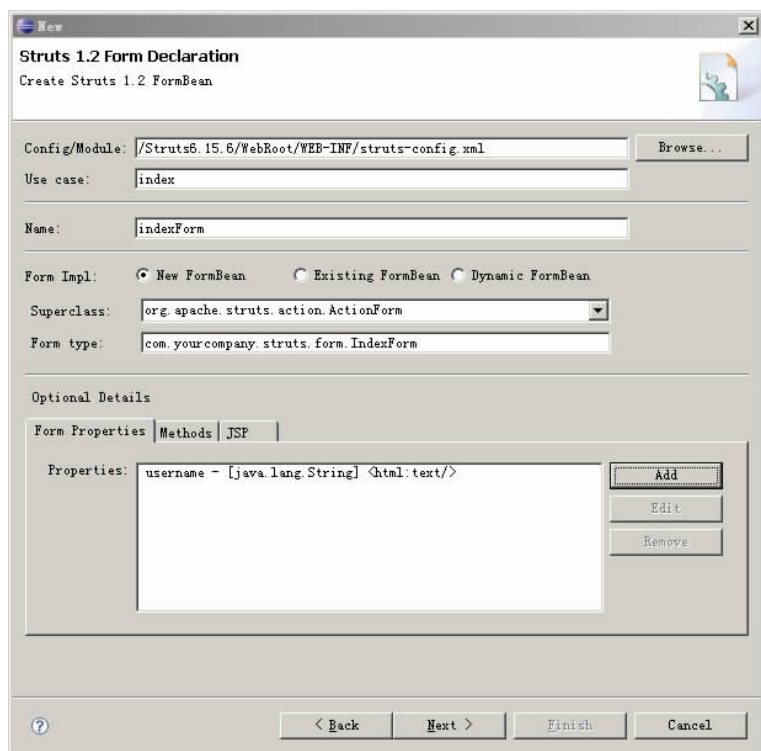


图 6-44 新建一个 JSP、Action、ActionForm

(3) 因为在本例中需要设置多选列表的初始值，所以必须在 JSP 文件中添加相应的 `<html:select>` 标签。在 JSP 文件的 `<html:form>` 内增加如下代码。

```
<html:select property="select" multiple="true"
    size="3">
    <html:option value="0432">吉林</html:option>
    <html:option value="0431">长春</html:option>
    <html:option value="024">沈阳</html:option>
</html:select>
```

其中 `<html:select>` 的 `property` 属性为 `select`，该列表为多选列表，所以相应地在 ActionForm 中要新建一个变量名为 `select` 的字符串数组类型的类变量，还要创建其 `set` 和 `get` 方法。

(4) 在 ActionForm 中增加程序如下所示。

```
private String select[];

public void setSelect(String[] select) {
    this.select = select;
}

public String[] getSelect() {
```

```
        return select;
    }
```

新建字符串数组变量 select，创建 set 和 get 方法。

(5) 在本示例中，需要将多选列表设置为初始值，所以在 ActionForm 中将 select 变量设置为初始值，才能达到本例要表达的效果。改动程序如下。

```
private String select[]={"024","0431"};
```

(6) 保存项目，部署项目，启动服务。

(7) 运行效果如图 6-45 所示。



图 6-45 运行效果

注意：介绍到这里，需要提醒读者的是，在使用 JSP 文件中的 `<html:select>` 标签时，必须在 ActionForm 中创建相对应的变量及变量的 set 和 get 方法，如果是下拉列表和单选列表，则在 ActionForm 中创建字符串类型即可，如果是多选列表的话，则必须在 ActionForm 中创建相对应的字符串数组类型及这个数组类型的 set 和 get 方法。

#### 6.16.7 如何获取 `<html:select />` 和 `<html:option />` 下拉列表单选值

(1) 像以前的项目开始一样，新建一个 JSP、Action、ActionForm，其中 ActionForm 的设置如图 6-46 所示。

在 JSP 属性页创建 JSP 文件，接下来单击“Next”按钮，使用默认设置的 Action，设置完成后单击“Finish”按钮应用设置。

(2) 新 Form 目录中的 index.jsp 文件中的 `<html:select>` 标签代码改成如下程序。

```
aihais : <html:select property="aihais">
    <html:option value="playball">ball</html:option>
    <html:option value="playgame">game</html:option>
    <html:option value="playbook">book</html:option>
</html:select>
```

通过改动的程序代码，增加了 `<html:option>` 标签来实现下拉列表中的下拉选择内容。

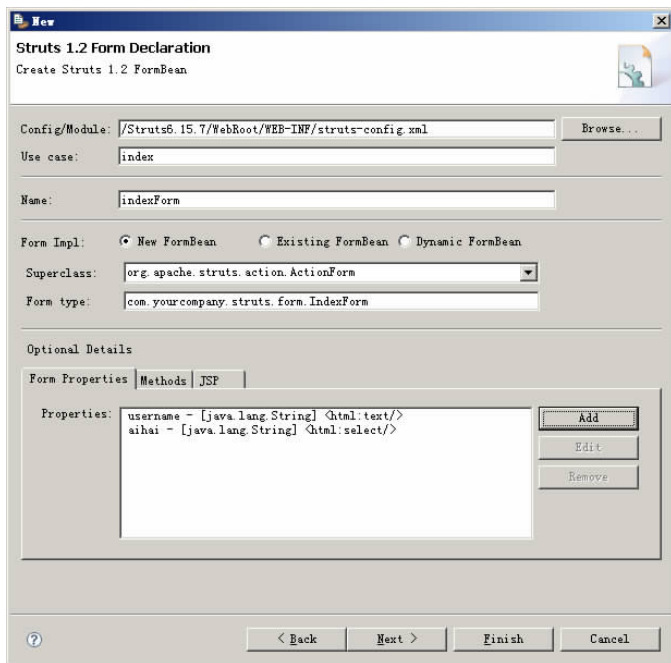


图 6-46 设置 ActionForm

(3) 更改 Action 代码如下。

```
public ActionForward execute(ActionMapping mapping, ActionForm form,
    HttpServletRequest request, HttpServletResponse response) {
    IndexForm indexForm = (IndexForm) form; // TODO Auto-generated method
    // stub
    System.out.print(indexForm.getUsername());
    System.out.print(indexForm.getAihai());
    return null;
}
```

在 Action 类的 execute 方法中打印出前台选择的爱好和用户名的值。

(4) 部署项目，启动服务。

(5) 在浏览器中输入相应的数据后单击“Submit”按钮，在 MyEclipse 中出现输入的用户名和爱好，显示效果如图 6-47 所示。

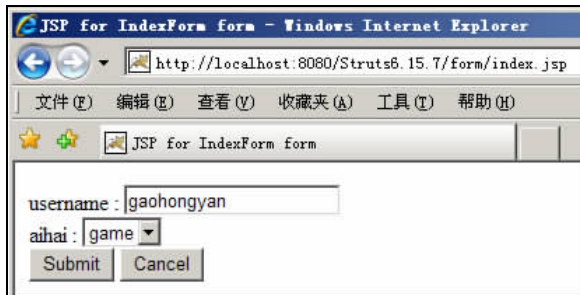


图 6-47 浏览器中显示效果

<html:select>的下拉列表和<html:select>的列表单选的程序代码方式一样，只是在 Action 选择相应的值即可。

#### 6.16.8 如何获取<html:select />和<html:option />列表多选值

(1) 按照上一节的第(1)步来创建应用程序。

(2) 创建完成后更改 JSP 文件代码如下。

```
aihao:<html:select property="aihao" multiple="true">
  <html:option value="playball">ball</html:option>
  <html:option value="playgame">game</html:option>
  <html:option value="playbook">book</html:option>
</html:select>
```

更改<html:select>的标记为多选列表。既然是多选的情况，那么其中的值肯定是 String[] 类型，所以还必须进入 ActionForm 中进行代码的更改。

(3) 更改 ActionForm 后的程序代码如下。

```
package com.yourcompany.struts.form;

import javax.servlet.http.HttpServletRequest;
import org.apache.struts.action.ActionErrors;
import org.apache.struts.action.ActionForm;
import org.apache.struts.action.ActionMapping;

public class IndexForm extends ActionForm {

    /** aihao property */
    private String aihao[];

    /** username property */
    private String username;

    public ActionErrors validate(ActionMapping mapping,
        HttpServletRequest request) {
        // TODO Auto-generated method stub
        return null;
    }

    public void reset(ActionMapping mapping, HttpServletRequest request) {
        // TODO Auto-generated method stub
    }

    public String getUsername() {
        return username;
    }

    public void setUsername(String username) {
        this.username = username;
    }
}
```



```
public void setAihao(String[] aihao) {  
    this.aihao = aihao;  
}  
  
public String[] getAihao() {  
    return aihao;  
}  
}
```

设置 aihao 为 String[] 字符串数组类型，并且加入 set 和 get 方法。

(4) 更改 Action 类代码如下。

```
public ActionForward execute(ActionMapping mapping, ActionForm form,  
    HttpServletRequest request, HttpServletResponse response) {  
    IndexForm indexForm = (IndexForm) form; // TODO Auto-generated method  
    // stub  
    String aihao[] = indexForm.getAihao();  
    for (int i = 0; i < aihao.length; i++) {  
        System.out.print(aihao[i]);  
    }  
    return null;  
}
```

(5) 部署项目，启动服务

(6) 程序运行如图 6-48 所示。

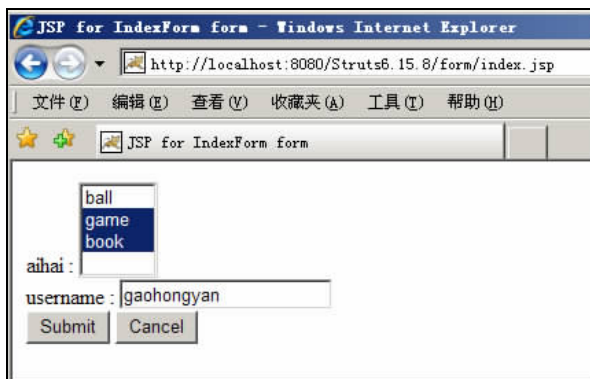


图 6-48 多项选择列表的运行效果

## 6.17 使用<html:options>动态生成<html:select /> 和<html:option>列表内容

当用户遇到需要从数据库中将处理后的数据放到 JSP 页的下拉列表、列表单选或列表

多选的情况，该如何处理呢？使用<html:options>标签。

在 MyEclipse 的面板中并没有出现<html:options>的标签，其实它在 Struts 中起到的作用还是非常重要的。

#### 6.17.1 将数据库的内容动态生成<html:select />和<html:option>列表内容

这里做一个实例，动态地往下拉列表中添加城市的名称和区号。

(1) 新建一个 JSP、Action、ActionForm，如图 6-49 所示。

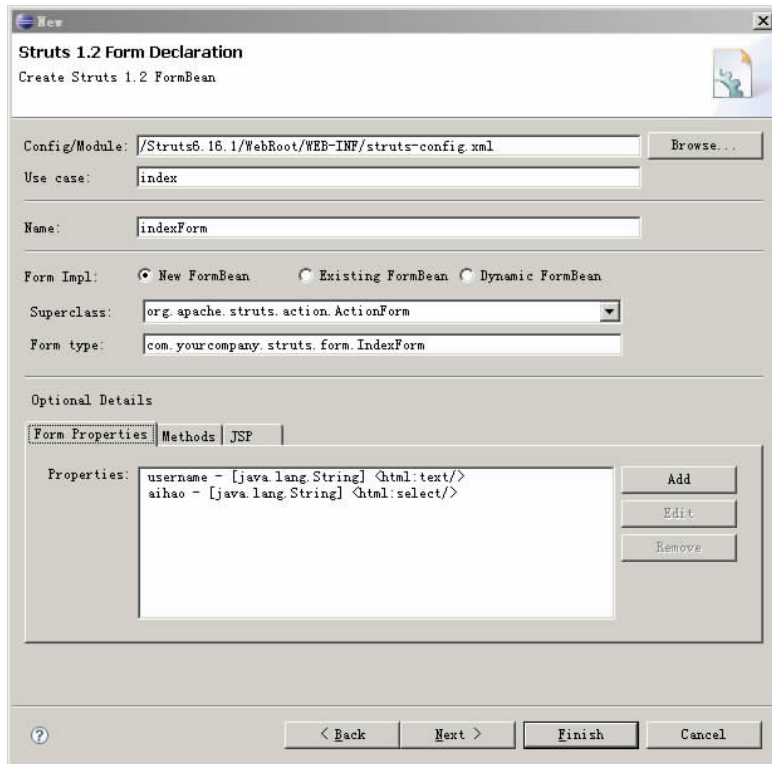


图 6-49 新建一个 JSP、Action、ActionForm

添加完两个属性 username、aihao 后，单击“JSP”属性页创建 JSP 文件，单击“Next”按钮，使用默认的设置创建 Action 类，再单击“Finish”按钮创建应用。

(2) 更改 JSP 文件内容如下。

```
<%@ page language="java" pageEncoding="gb2312"%>
<%@ page import="java.util.*,org.apache.struts.util.*"%>
<%@ taglib uri="http://jakarta.apache.org/struts/tags-bean"
    prefix="bean"%>
<%@ taglib uri="http://jakarta.apache.org/struts/tags-html"
    prefix="html"%>

<html>
  <head>
    <title>JSP for IndexForm form</title>
```

```
</head>
<body>
    <%
        ArrayList al = new ArrayList();
        al.add(new LabelValueBean("吉林市", "0432"));
        al.add(new LabelValueBean("长春市", "0431"));
        al.add(new LabelValueBean("沈阳市", "024"));
        pageContext.setAttribute("al", al);
    %>
    <html:form action="/index">
        aihao : <html:select property="aihao">
            <html:options collection="al" property="value"
                labelProperty="label"/>
        </html:select>
        <html:errors property="aihao" />
        <br />
        username : <html:text property="username" />
        <html:errors property="username" />
        <br />
        <html:submit />
        <html:cancel />
    </html:form>
</body>
</html>
```

注意：在本 JSP 文件中，假定可以通过 JDBC 语句从数据库中查询出城市及区号的数据。

在本文件中使用 ArrayList 类封装下拉列表中的内容，LabelValueBean 类是 Struts 框架特有的类，这个类有两个参数：第一个参数是 label，即显示在下拉列表中的文本内容；第二个参数是下拉列表中选项所对应的值，属性 collection 指定一个实现了 List 接口的对象。

将 al 对象添加到 pageContext 对象中，使在本页中有效。

使用<html:options collection="al" property="value" labelProperty="label"/>程序代码将 ArrayList 对象中的内容显示出来。

(3) 部署项目，启动服务。

(4) 程序运行效果如图 6-50 所示。



图 6-50 运行效果

本示例中只是将 ArrayList 对象 al 放到 pageContext 中，在有 JDBC 情况下，可以将这样的代码放在 Action 类中，然后在 request 对象中加入属性，Forward 相应的 JSP 文件后再取出并显示。

#### 6.17.2 初始化<html:select />和<html:options>列表生成的内容

(1) 在 ActionForm 中进行初始化，代码如下。

```
private String aihao="024";
```

(2) 在 Action 中通过 ActionForm 的 set 方法进行初始化。

(3) 在 JSP 页面初始化，例程如下。

```
<html:select property="aihao" value="0431">
<html:options collection="al" property="value" labelProperty="label"/>
</html:select>
```

具体使用哪种方法就要看实际的情况了。

## 6.18 使用<html:optionsCollection>动态生成 <html:option>列表内容

<html:optionsCollection>标签和<html:options>的用法很相似。

例如：

```
<html:select property="custId">
<html:optionsCollection property="customers"
                        label="name" value="custId" />
</html:select>
```

其中，optionsCollection 标签中的 property 指属性，label 指属性中的全局变量，value 也是指属性中的全局变量，区别就是 label 显示的是下拉列表的文字，而 value 则是选择下拉项的值。这个标签也有一个 name 属性，其值是存放在 pageContext/request/session/application 中的 Bean 的名称。

这个标签和 org.apache.struts.util.LabelValueBean 结合得很好，如果把 label 和 value 都放到这个对象中，可以很简单地进行应用：

```
<html:select property="custId">
<html:optionsCollection property="customers" />
</html:select>
```

建议在项目中直接使用 LabelValueBean 类。

在 Struts 1.2 版本中，推荐尽量使用<html:optionsCollection>标签进行动态内容的创建，应该逐步放弃使用<html:options>。

### 6.18.1 使用<html:optionsCollection>动态生成<html:option>列表内容实例

在本例中使用<html:optionsCollection>标签生成性别列表的功能。

(1) JSP 文件内容如下。

```
<%@ page language="java" pageEncoding="gb2312"%>
<%@ page import="mypackage.test,java.util.*,org.apache.struts.util.*"%>
<%@ taglib uri="http://jakarta.apache.org/struts/tags-bean"
    prefix="bean"%>
<%@ taglib uri="http://jakarta.apache.org/struts/tags-html"
    prefix="html"%>

<html>
  <head>
    <title>JSP for IndexForm form</title>
  </head>
  <body>
    <%
      ArrayList al = new ArrayList();
      al.add(new LabelValueBean("吉林市", "0432"));
      al.add(new LabelValueBean("沈阳市", "024"));
      al.add(new LabelValueBean("长春市", "0431"));
      pageContext.setAttribute("al", al);
    %>

    <%
      ArrayList otheral = new ArrayList();
      test tt1 = new test();
      tt1.setSexLabel("男");
      tt1.setSexValue("1");

      test tt2 = new test();
      tt2.setSexLabel("女");
      tt2.setSexValue("0");

      otheral.add(tt1);
      otheral.add(tt2);

      pageContext.setAttribute("otheral", otheral);
    %>

    <html:form action="/index">
      username : <html:text property="username" />
      <html:errors property="username" />
      <br />
      <html:select property="aihao">
        <html:optionsCollection name="al" />
      </html:select>
```

```
<html:select property="otheraihao">
    <html:optionsCollection name="otheral" label="sexLabel"
        value="sexValue" />
</html:select>

<html:submit />

<html:submit />

<html:cancel />
</html:form>
</body>
</html>
```

(2) 第三方的 Bean 定义如下。

```
package mypackage;

public class test {
    private String sexLabel;

    private String sexValue;

    public void setSexLabel(String sexLabel) {
        this.sexLabel = sexLabel;
    }

    public String getSexLabel() {
        return sexLabel;
    }

    public void setSexValue(String sexValue) {
        this.sexValue = sexValue;
    }

    public String getSexValue() {
        return sexValue;
    }
}
```

(3) ActionForm 代码如下。

```
/*
 * Generated by MyEclipse Struts
 * Template path: templates/java/JavaClass.vtl
 */
package com.yourcompany.struts.form;

import javax.servlet.http.HttpServletRequest;
import org.apache.struts.action.ActionErrors;
import org.apache.struts.action.ActionForm;
```

```
import org.apache.struts.action.ActionMapping;

/**
 * MyEclipse Struts Creation date: 03-29-2007
 *
 * XDoclet definition:
 *
 * @struts.form name="indexForm"
 */
public class IndexForm extends ActionForm {
    /**
     * Generated fields
     */

    /** username property */
    private String username;

    private String aihao;

    private String otheraihao;

    /**
     * Generated Methods
     */

    /**
     * Method validate
     *
     * @param mapping
     * @param request
     * @return ActionErrors
     */
    public ActionErrors validate(ActionMapping mapping,
        HttpServletRequest request) {
        // TODO Auto-generated method stub
        return null;
    }

    /**
     * Method reset
     *
     * @param mapping
     * @param request
     */
    public void reset(ActionMapping mapping, HttpServletRequest request) {
        // TODO Auto-generated method stub
    }

    /**
     * Returns the username.

```

```
*
 * @return String
 */
public String getUsername() {
    return username;
}

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

public void setAihao(String aihao) {
    this.aihao = aihao;
}

public String getAihao() {
    return aihao;
}

public void setOtheraihao(String otheraihao) {
    this.otheraihao = otheraihao;
}

public String getOtheraihao() {
    return otheraihao;
}
}
```

### 6.18.2 使用<html:optionsCollection>标签中的 property 属性来生成下拉列表

property 属性代表<html:optionsCollection>的 name 属性中的 Bean 有一个属性为 Collection 类型，则可以通过使用<html:optionsCollection>的 property 属性来进行数据内容的遍历。看以下程序代码。

(1) JSP 程序代码如下。

```
<%@ page language="java" pageEncoding="gb2312"%>
<%@ page import="mypackage.*, java.util.*, org.apache.struts.util.*"%>
<%@ taglib uri="http://jakarta.apache.org/struts/tags-bean"
    prefix="bean"%>
<%@ taglib uri="http://jakarta.apache.org/struts/tags-html"
    prefix="html"%>

<html>
```



```
<head>
    <title>JSP for IndexForm form</title>
</head>
<body>
    <%
        ArrayList al_ref = new ArrayList();
        al_ref.add(new LabelValueBean("吉林市", "0432"));
        al_ref.add(new LabelValueBean("沈阳市", "024"));
        al_ref.add(new LabelValueBean("长春市", "0431"));

        test1 t1_1 = new test1();
        t1_1.setAl(al_ref);

        pageContext.setAttribute("al", t1_1);
    %>
    <html:form action="/index">
        username : <html:text property="username" />
        <html:errors property="username" />
        <br />
        <html:select property="aihao">
            <html:optionsCollection name="al" property="al" />
        </html:select>
        <html:submit />
        <html:cancel />
    </html:form>
</body>
</html>
```

(2) Bean 的代码如下。

```
package mypackage;

import java.util.ArrayList;

public class test1 {
    private ArrayList al;

    public void setAl(ArrayList al) {
        this.al = al;
    }

    public ArrayList getAl() {
        return al;
    }
}
```

(3) ActionForm 的程序代码如下。

```
/*
 * Generated by MyEclipse Struts
```

```
* Template path: templates/java/JavaClass.vtl
*/
package com.yourcompany.struts.form;

import javax.servlet.http.HttpServletRequest;
import org.apache.struts.action.ActionErrors;
import org.apache.struts.action.ActionForm;
import org.apache.struts.action.ActionMapping;

/**
 * MyEclipse Struts Creation date: 03-29-2007
 *
 * XDoclet definition:
 *
 * @struts.form name="indexForm"
 */
public class IndexForm extends ActionForm {
    /**
     * Generated fields
     */

    /** username property */
    private String username;

    private String aihao;

    /**
     * Generated Methods
     */

    /**
     * Method validate
     *
     * @param mapping
     * @param request
     * @return ActionErrors
     */
    public ActionErrors validate(ActionMapping mapping,
        HttpServletRequest request) {
        // TODO Auto-generated method stub
        return null;
    }

    /**
     * Method reset
     *
     * @param mapping
     * @param request
     */
    public void reset(ActionMapping mapping, HttpServletRequest request) {
```

```
// TODO Auto-generated method stub
}

/**
 * Returns the username.
 *
 * @return String
 */
public String getUsername() {
    return username;
}

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

public void setAihao(String aihao) {
    this.aihao = aihao;
}

public String getAihao() {
    return aihao;
}
}
```

部署项目后，则可以看到下拉列表的内容。

## 6.19 <html:password>、<html:text>、 <html:textarea>标签的使用

<html:password>、<html:text>和<html:textarea>标签的使用非常普遍，它们主要的功能就是实现用户输入，而且它们的使用也非常简单，并没有像类似<html:select>那样复杂的操作。

### 6.19.1 <html:password>标签的 redisplay 属性实例

<html:password>标签将生成 HTML 的<input type="password" name="textfield">密码输入框标记。

它的使用和<html:text>一样方便，主要的属性就是 property，对应于 ActionForm 中的

成员变量属性。但它有一个 `redisplay` 属性却很重要，它的取值为 `true` 或 `false`，如果设置为 `false` 时，这时表示 `name` 或 `property` 属性或 `value` 属性指定的值将不能用于填充控件。用实例验证一下。

(1) 像以前创建 Struts 项目一样，新建一个 JSP、Action、ActionForm，并且添加属性 `username` 和 `password`，如图 6-51 所示。

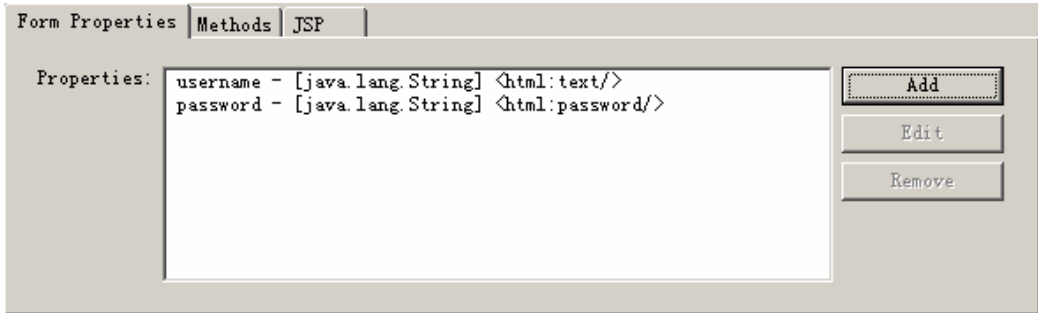


图 6-51 添加两个属性 `username` 和 `password`

在“JSP”属性页创建 JSP 文件，使用默认的设置创建 Action 类，单击“Finish”按钮完成设置。

(2) 更改 Action 代码如下。

```
public ActionForward execute(ActionMapping mapping, ActionForm form,
    HttpServletRequest request, HttpServletResponse response) {
    IndexForm indexForm = (IndexForm) form; // TODO Auto-generated method
    // stub
    return mapping.getInputForward();
}
```

(3) 部署项目，启动服务。

(4) 运行效果如图 6-52 所示。

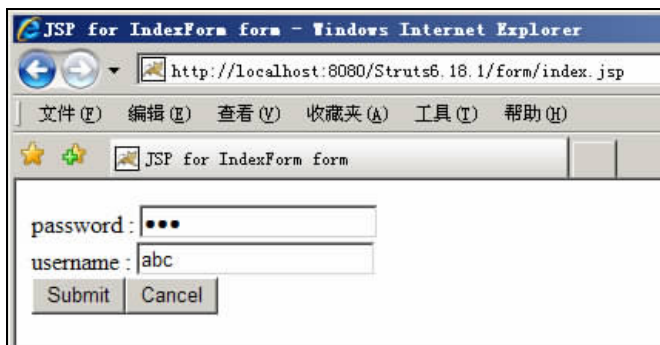


图 6-52 运行效果

在 `username` 和 `password` 表单中输入内容后，单击“Submit”按钮进行提交。

(5) 这时发现浏览器的地址栏内容改变, 并且 username 和 password 表单中的内容并没有改变, 如图 6-53 所示。



图 6-53 没有改变的 username 和 password

(6) 在一些情况下, 希望在 Action 里面生成 ActionErrors, 返回 input 的页面后使 `<html:password>` 标签中的内容清空, 这时就得使用 `redisplay` 属性了。

(7) 更改 JSP 文件的 `<html:password>` 标签内容:

```
<html:password property="password" redisplay="false"/>
```

(8) 刷新当前的浏览器页面, 这时发现 `<html:password>` 中的数据被清空了, 也就是没有将 ActionForm 中属性的内容自动添充。

#### 6.19.2 使用 style 的 CSS 样式改变 `<html:text>` 标签的外观

为了网页的美观, 通常都需要将 `<html:text>` 标签的边框、底色或边框颜色进行自定义风格的设置, 在普通的 HTML 标签中可以简单地使用 `style` 属性进行 CSS 样式的设置, 当然在 Struts 的 `<html:text>` 标签中, 或者 `<html:password>` 和 `<html:textarea>` 标签中也可以使用 CSS 样式来定义表单的外观。例程如下。

```
<html:text property="username" style="height:18px;width:110px;
border:1px solid #ffffff;background-color:#0099CC;font-size:9pt;
font-family:verdana;color:#ffffff"/>
```

在 `<html:text>` 标签中使用 CSS 样式和普通的 HTML 表单使用 CSS 样式一模一样, 外观如图 6-54 所示。

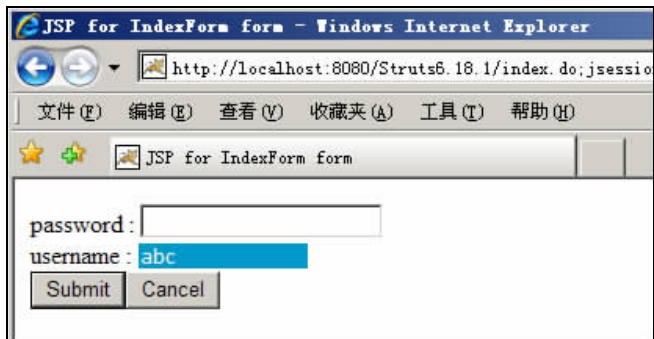


图 6-54 改变后的 `<html:text>` 标签的外观

### 6.19.3 <html:textarea>标签的使用

一个标准的<html:textarea>标签代码：

```
<html:text name="" property="" rows="" cols="" />
```

像所有的其他标签一样，都有 name、property 属性，这里的 name 属性是指在 request 或 session 中 Bean 或变量的名字，如果没有 name 这个属性的话，则默认 property 里面的值是其对应 ActionForm 的属性。rows 和 cols 则是设置标签的高和宽，即行和列数。

## 6.20 <html:radio>标签的使用

下面举一个代码的实例。

```
<html:radio idName=" idName " value="value" name=" name"  
property=" property " />
```

上面的<html:radio>标签代码表示在输出时，<html:radio>标签输出为 HTML 的<input type="radio">标记，name 输出为 name="name"。对于 value 的输出，当不指定 idName 时，value="value"；当指定 idName 时，输出是 Bean 名为" idName "、属性名为"value"的属性值，当 Bean 名为"name"，属性名为"property"的属性值等于上述 value 的输出值时，输出 checked="checked"。示例程序如下。

```
<html:radio property="sex" value="1"/>男  
<html:radio property="sex" value="0"/>女
```

上面程序将 radio 形式分成单选分组，在同一组内只能选择一个 radio，如果想实现分组的效果，则需要将 property 设置为相同即可。

在上面的程序中并没有设置性别为男或女的默认值，如果想设置<html:radio>标签的默认值怎么办？下面将给出答案。

设置<html:radio>标签的默认值

改动<html:radio>标签相对应的 ActionForm 中的属性初始值即可，例如：

```
private String sex="1";
```

程序运行结果如图 6-55 所示。

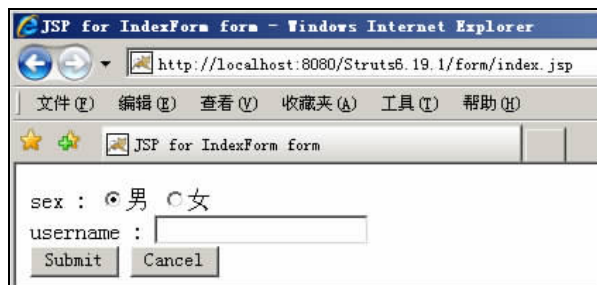


图 6-55 设置<html:radio>标签初始值

## 6.21 <html:submit>和<html:reset>标签的使用

这两个标签的主要功能就是生成提交按钮和一个复位当前列表内容的重置按钮，它们的作用和 HTML 语言中的相对应的标记功能一样，在使用上没有什么扩展之处，开发时不需要多余的设置。

## 6.22 总结

本章一起学习了 Struts 中的 HTML 标签的使用，相信通过这一章的学习，读者对 Struts 标签的熟悉程度会进一步加深，学习 Struts 所有标签的方法大致一样，但想更多地了解标签更多的功能，请留意 MyEclipse 右边 Properties 面板中的属性列表。

# 第 7 章

## Struts-Logic 标签库

在 Struts 框架中包含 Logic 标签库，该标签库主要的功能就是用来做一些逻辑上的处理，比如循环、判断等。

在 MyEclipse 的 Snippets 面板中有 Struts-Logic 标签列表，如图 7-1 所示。

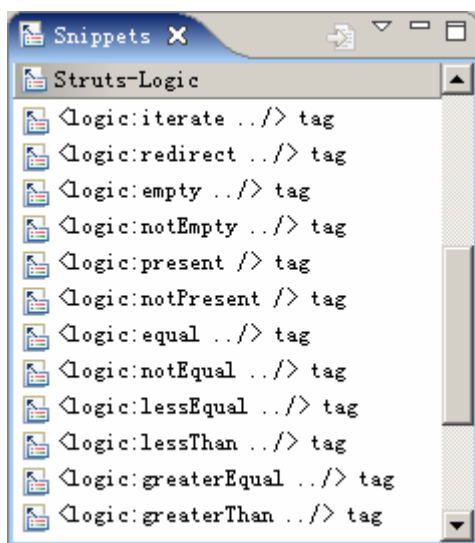


图 7-1 Struts-Logic 标签列表

在本章中，将详细介绍这些标签的使用。

### 7.1 <logic:iterate>标签的功能

<logic:iterate>标签的功能就是创建一个循环语句，通常将数组 Collection、Map、List 对象中的内容进行输出，使用标签省略了在 JSP 页面中写入<%.....%>大量的 JSP 脚本，也使得 JSP 页面更加整洁，程序的代码更加容易进行维护。

下面是一个<logic:iterate>程序代码的例子。

```
<logic:iterate id="prinfo" name="c_prinfo" scope="session">
  <bean:write name="prinfo" property="prname" />
</logic:iterate>
```



prinfo 是 session 中 JavaBean 的别名；c\_prinfo 是 session 中集合对象的名称；scope 将从 session 中获得<bean:write>的 name 与<logic:iterate>标签中的 id 值对应；property 代表这个 Bean 的属性名，将获得这个属性的值并打印显示出来。

### 7.1.1 打印数组中的内容

打印数组中的内容的 JSP 文件例程如下。

```
<body>
  <%
    String[] ghyghost = { "1", "2", "3", "4", "5" };
    pageContext.setAttribute("ghyghost_id", ghyghost);
  %>
  <logic:iterate id="write_al" name="ghyghost_id">
    <bean:write name="write_al" />
    <br>
  </logic:iterate>
</body>
```

程序首先创建一个 String[] 字符串数组对象，并进行初始化，然后将 ghyghost 对象放入 pageContext 上下文 page 对象中。再通过<logic:iterate>标签将 ghyghost\_id 对象中的数组打印出来，首先使用<logic:iterate>标签中的 name 属性在 pageContext 中寻找是否有 ghyghost\_id 的对象，如果有，取出来，并命名一个别名——write\_al，这时就可以使用<bean:write>标签打印出别名为 write\_al 中的数组内容了。

运行效果如图 7-2 所示。

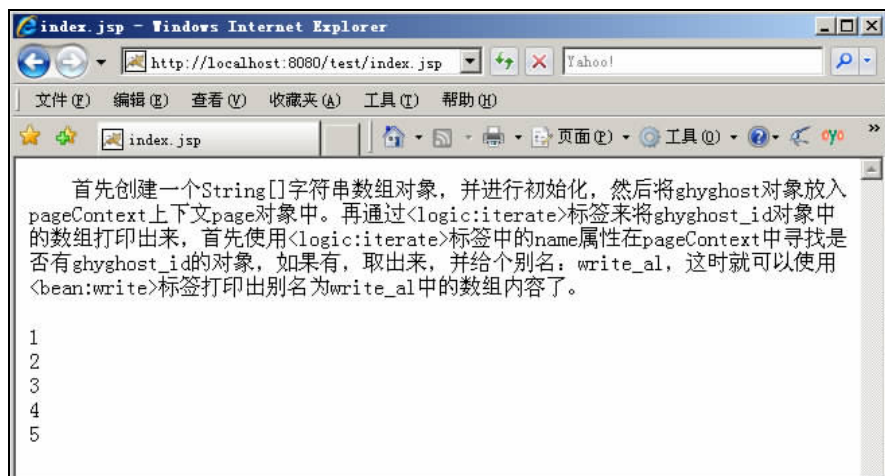


图 7-2 打印数组内容

### 7.1.2 打印 HashMap 中的内容

打印 HashMap 中的内容的 JSP 文件例程如下。

```
<body>

<%
    HashMap hm = new HashMap();
    hm.put("1", "高红岩 1");
    hm.put("2", "高红岩 2");
    hm.put("3", "高红岩 3");
    hm.put("4", "高红岩 4");
    hm.put("5", "高红岩 5");
    pageContext.setAttribute("hm_id", hm);
%>
<logic:iterate id="write_all" name="hm_id">
    <bean:write name="write_all" property="key" />:
        <bean:write name="write_all" property="value" />:
    <br>
</logic:iterate>
</body>
```

程序首先声明一个 HashMap 对象 hm，然后在 hm 对象中放入相应的 key 和 value，再放入到 pageContext 上下文 page 对象中。通过<logic:iterate>标签的 name 属性从上下文中取得 hm\_id 的内容，然后再命名一个别名为 write\_all，使用<bean:write>标签打印出 write\_all 中的内容，由于打印的是一个 HashMap 对象，里面有 key 和 value，所以得在<bean:write>标签中加入 property 属性为 key 和 value 的形式，以打印出 key 及其相对应的 value 的值。

运行效果如图 7-3 所示。

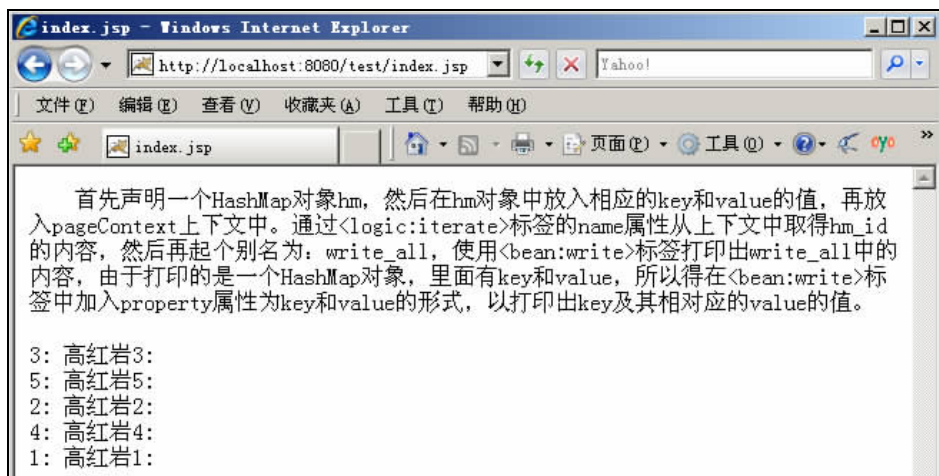


图 7-3 打印 HashMap 的内容

### 7.1.3 打印 ArrayList 中的内容

打印 ArrayList 中的内容的 JSP 文件例程如下。

```
<body>
<%
    ArrayList al_ref = new ArrayList();
```

```
al_ref.add("高红岩 1");
al_ref.add("高红岩 2");
al_ref.add("高红岩 3");
al_ref.add("高红岩 4");
al_ref.add("高红岩 5");
pageContext.setAttribute("al_id", al_ref);

%>
<logic:iterate id="write_all" name="al_id">
  <bean:write name="write_all" />
  <br>
</logic:iterate>
</body>
```

程序创建一个 `ArrayList` 对象 `al_ref`，然后使用 `add` 方法进行 `al_ref` 对象的填充，再放入 `pageContext` 中。通过 `<logic:iterate>` 标签来进行循环，并通过 `<bean:write>` 打印出 `ArrayList` 中的内容，通过代码可以看到，打印 `ArrayList` 和打印 `String[]` 的程序代码大体一样。

如果 `ArrayList` 中存放的是 `Bean` 时，可以使用如下类似的代码来进行输出。

```
<bean:write name="write_all" property="bean_property" />
```

运行效果如图 7-4 所示。

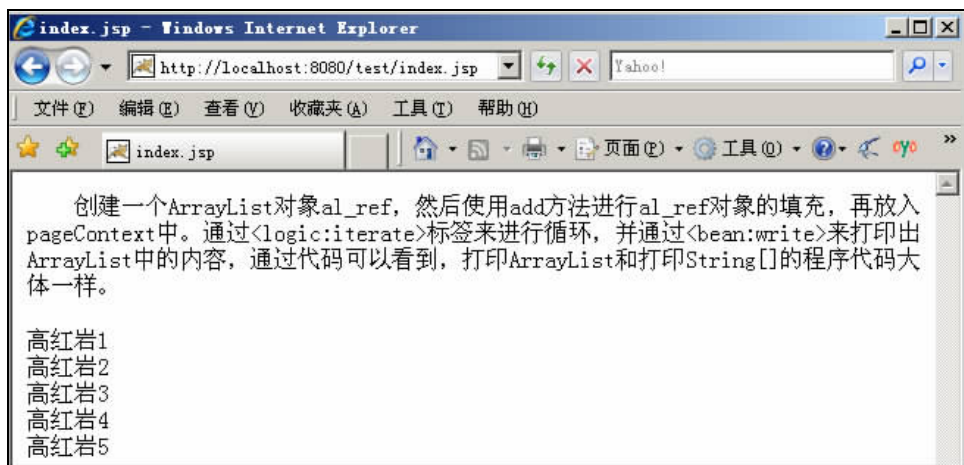


图 7-4 打印 `ArrayList` 内容

## 7.2 `<logic:redirect>` 重定向的标签

标签 `<logic:redirect>` 和 HTTP 中的重定向功能一样，它的使用方法很简单，它具有的属性如图 7-5 所示。

`<logic:redirect>` 标签的主要属性为：`action`、`forward`、`href`、`page`，和标签 `<html:link>` 中的属性功能一样，在重定向的过程中也可以传递参数，通过 `paramId`、`paramName`、`paramProperty` 这 3 个属性合作传递参数。

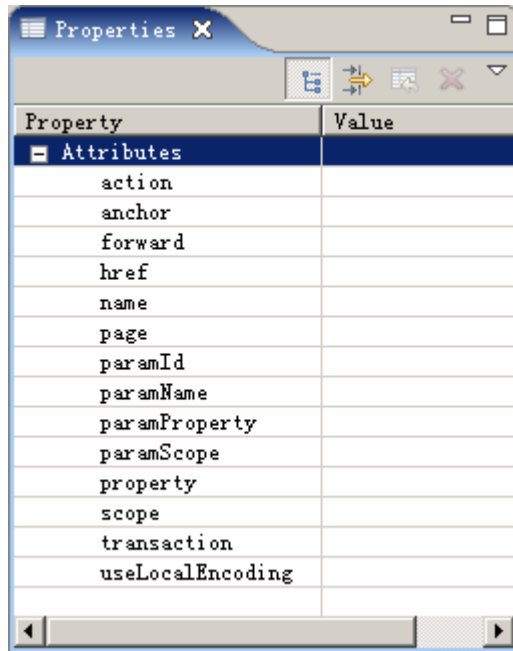


图 7-5 &lt;logic:redirect&gt;标签属性

### 7.3 <logic:forward>转发的标签

在 Struts 中通过<logic:forward>标签来实现 JSP 的转发功能，例程如下。

```
<logic:forward name="other_forward">
```

name 属性的内容是在 struts-config.xml 配置文件中的全局 forward 的名称。

### 7.4 <logic:empty>和<logic:notempty>标签的作用

在普通的 JSP 脚本中，通过判断一个对象是否为 null 或者是否为""都需要使用 if 语句，这样在有 Struts 的<logic:empty>标签的情况下，不会在 JSP 中出现<%%>脚本，JSP 页面变得整洁，并且易于维护。

<logic:empty>和<logic:notEmpty>标签的实例

JSP 文件例程如下。

```
<body>
  <%
    String ghyghost = "gaohongyan";
    String ghynull = "";
    pageContext.setAttribute("ref_1", ghyghost);
    pageContext.setAttribute("ref_2", ghynull);
  %>
```

```
<logic:empty name="ref_2">
is null<br>
</logic:empty>
<br>
<logic:notEmpty name="ref_1">
    <bean:write name="ref_1" />
    <br>
not null
</logic:notEmpty>
</body>
```

程序首先创建一个 ghyghost 的字符串变量并赋值为 gaohongyan，再声明一个 ghynull 的变量初始化为空""，然后分别将这2个对象放入 pageContext 对象中，再通过<logic:empty>和<logic:notEmpty>标签来进行字符串的 null、""判断和非空的判断。如果非空，使用<bean:write>标签打印出字符串中的内容。

运行效果如图 7-6 所示。

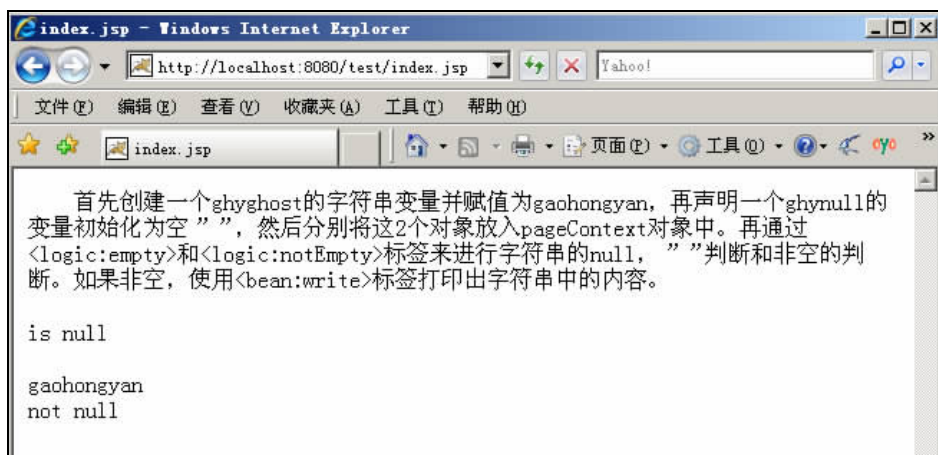


图 7-6 <logic:empty>和<logic:notEmpty>标签的效果

## 7.5 <logic:present>标签的作用和与<logic:empty>的区别

<logic:present>和<logic:empty>，它们的用法大致相同，唯一的不同点是：两者在对空字符串的处理上存在着不同。下面是示例程序的代码。

```
<logic:notPresent name="users">
notpresent
</logic:notPresent>

    <logic:notEmpty name="users">
notempty
</logic:notEmpty>

    <logic:empty name="users">
```

```
empty
</logic:empty>

    <logic:present name="users">
present
</logic:present>
```

创建一个 JSP 文件，并输入上面的程序代码，部署项目，启动服务，输入正确的 URL 地址后，由于第一次访问该 JSP 的时候 users 没有定义，并且也不在 page、request、session、application 任何一个作用域中，因此输出的结果为 notpresent 和 empty。

下面增加一个 Action，让该 Action 在 index.jsp 之前执行，然后再跳转到 index.jsp 中，同时在该 Action 的 execute 方法中增加如下代码。

```
String userName = "";
request.setAttribute("users", userName);
return new ActionForward("/index.jsp");
```

这里将 userName 保存在 request 中，key 为 users，再将请求转发至 index.jsp 中，由于 userName 的值为一个空字符串，转发过后，输出的值为 empty 和 present。

这就说明虽然存在 users 这个对象，但这个 users 对象为空，所以结果为 empty 和 present。这里再做一次改动来查看 <logic:present> 和 <logic:empty> 标签的区别。

将 Action 的 execute 方法中的代码改为：

```
String userName = null;
request.setAttribute("users", userName);
return new ActionForward("/hello.jsp");
```

这次不同的是 userName 不再为空字符串了，而是 null 值，当转发至 index.jsp 后，输出的值为 notpresent 和 empty。

对比这两次实验，可以得出结论：对于没有在 page、request、session、application 中定义或者没有分配内存空间（null 值）的变量，这两个标记处理的方法是一致的，都会认为此变量不存在（notpresent）或者为空（empty）。而对于空字符串""值，它们的处理就不一样了，<logic:present> 标签认为空字符串仍然是存在的，也就是说，只要引用了一块内存空间的变量，<logic:present> 就会返回 present，而 <logic:empty> 则认为空字符串仍然为空，由此得出，在 <logic:empty> 看来，变量不仅仅要引用一块内存空间，而且该地址空间的值不能为空字符串，否则都认为该变量为空，都会返回 empty。

## 7.6 用<logic:equal>和<logic:notEqual>

### 判断等于和不等于

- <logic:equal> 标签的作用是判断变量或 Bean 的属性值是不是等于某一个常量。
- <logic:notEqual> 标签的作用是判断变量或 Bean 的属性值是不是不等于某一个常量。

### 7.6.1 使用<logic:equal>和<logic:notEqual>判断变量

例程如下。

```
<body>
  <%
    String ghyghost = "gaohongyan";
    String ghyghostnot = "gaohongyan";
    pageContext.setAttribute("var", ghyghost);
    pageContext.setAttribute("varnot", ghyghostnot);
  %>
  <logic:equal name="var" value="gaohongyan">
    等于 gaohongyan
  </logic:equal>
  <br>
  <logic:notEqual name="varnot" value="abc">
    不等于 abc
  </logic:notEqual>
</body>
```

显示结果如图 7-7 所示。



图 7-7 显示效果

### 7.6.2 使用<logic:equal>和<logic:notEqual>判断 Bean 的属性值

例程如下。

```
<body>
  <%
    abc test_ref1 = new abc();
    test_ref1.setName("gaohongyan");

    abc test_ref2 = new abc();
    test_ref2.setName("gaohongyan");

    pageContext.setAttribute("var", test_ref1);
    pageContext.setAttribute("varnot", test_ref2);
  %>
  <logic:equal name="var" property="name" value="gaohongyan">
    等于 gaohongyan
  </logic:equal>
```

```
<br>
<logic:notEqual name="varnot" property="name" value="abc">
不等于 abc
</logic:notEqual>
</body>
```

在这里面使用了 Bean，名称为 abc，abc 的源代码如下。

```
package mypackage;
public class abc {
    private String name;

    public void setName(String name) {
        this.name = name;
    }

    public String getName() {
        return name;
    }
}
```

如果用到这个 abc 的类，还需要 i 导入该类。程序运行效果如图 7-8 所示。



图 7-8 判断 Bean 中的属性

## 7.7 用<logic:lessEqual>和<logic:lessThan>

### 判断小于等于和小于

- <logic:lessEqual>判断变量或 Bean 的属性值是否小于等于指定的常量。
- <logic:lessThan>判断变量或 Bean 的属性值是否小于指定的常量。

#### 7.7.1 <logic:lessEqual>和<logic:lessThan>判断变量

JSP 源代码如下。

```
<body>
<%
    Integer number = 100;
```



```
Integer numbertot = 100;
pageContext.setAttribute("num", number);
pageContext.setAttribute("numnot", numbertot);

%>
<logic:lessEqual name="num" value="100">
小于等于 100
</logic:lessEqual>
<br>
<logic:lessThan name="numnot" value="101">
小于 101
</logic:lessThan>

</body>
```

输出结果如图 7-9 所示。



图 7-9 输出结果

### 7.7.2 <logic:lessEqual>和<logic:lessThan>判断 Bean 的属性值

JSP 代码如下。

```
<body>
<%
    abc abc_ref1 = new abc();
    abc_ref1.setNumber(100);

    abc abc_ref2 = new abc();
    abc_ref2.setNumber(100);

    pageContext.setAttribute("num", abc_ref1);
    pageContext.setAttribute("numnot", abc_ref2);
%>
<logic:lessEqual name="num" property="number" value="100">
小于等于 100
</logic:lessEqual>
<br>
<logic:lessThan name="numnot" property="number" value="101">
小于 101
</logic:lessThan>

</body>
```

Bean 的源代码如下。

```
package mypackage;
public class abc {
    private int number;

    public void setNumber(int number) {
        this.number = number;
    }

    public int getNumber() {
        return number;
    }
}
```

运行效果如图 7-10 所示。



图 7-10 运行效果

## 7.8 用<logic:greaterEqual>和<logic: greaterThan>判断大于等于和大于

- <logic:greaterEqual>标签的功能是判断变量或 Bean 的属性值是不是大于等于一个常量。
- <logic: greaterThan>标签的功能是判断变量或 Bean 的属性值是不是大于一个常量。

### 7.8.1 <logic:greaterEqual>和<logic: greaterThan>判断变量

JSP 源代码如下。

```
<body>
    <%
        Integer abc_ref1 = 100;
        Integer abc_ref2 = 102;
        pageContext.setAttribute("num", abc_ref1);
        pageContext.setAttribute("numnot", abc_ref2);
    %>
```

```
%>
<logic:greaterEqual name="num" value="100">
大于等于 100
</logic:greaterEqual>
<br>

<logic:greaterThan name="numnot" value="101">
大于 101
</logic:greaterThan>

</body>
```

运行效果如图 7-11 所示。



图 7-11 运行效果

### 7.8.2 <logic:greaterEqual>和<logic: greaterThan>判断 Bean 的属性值

JSP 源代码如下。

```
<body>
<%
    abc abc_ref1 = new abc();
    abc_ref1.setNumber(100);

    abc abc_ref2 = new abc();
    abc_ref2.setNumber(102);

    pageContext.setAttribute("num", abc_ref1);
    pageContext.setAttribute("numnot", abc_ref2);
%>
<logic:greaterEqual name="num" property="number" value="100">
大于等于 100
</logic:greaterEqual>
<br>

<logic:greaterThan name="numnot" property="number" value="101">
大于 101
</logic:greaterThan>

</body>
```

Bean 源代码如下。

```
package mypackage;  
public class abc {  
    private int number;  
  
    public void setNumber(int number) {  
        this.number = number;  
    }  
  
    public int getNumber() {  
        return number;  
    }  
}
```

程序运行效果如图 7-12 所示。



图 7-12 运行效果

# 第 8 章

## Struts-Bean 标签库

### 8.1 Bean 标签库的功能

在 JSP 页面中通过<bean>标签库,可以输出存放在上下文中的 Bean 的 property 属性值,比如 request、session。例如通常在 JSP 页面中,尽量不要出现 ResultSet 的对象,应该在将处理的数据结果封装在 Java 的 Bean 中,再放入 request 对象中,转给 JSP 文件后再通过<bean:write>的标签库打印出 Bean 中的属性值。还可以使用<bean>标签库打印出参数值等功能。

在 MyEclipse 中的<bean>标签列表如图 8-1 所示。

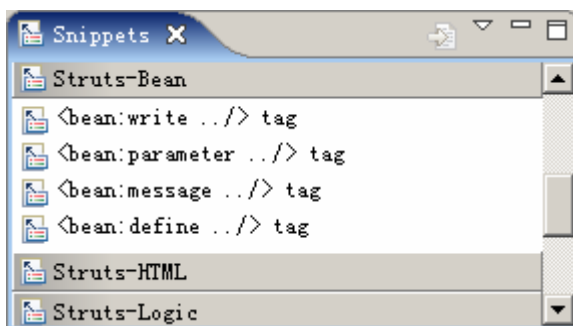


图 8-1 Struts 的 Bean 列表

本章将通过实例一起来学习如图 8-1 所示的 4 个 Bean 的标签。

### 8.2 <bean:write />标签打印 Bean 中的属性值

<bean:write>标签的功能就是打印出 Bean 的属性值,类似于 out.print()方法。它会调用 Bean 的 toString()方法来进行字符串的输出,如果 Bean 没有该方法,会调用 Object 类的 toString()方法。

#### 8.2.1 <bean:write>标签打印变量

<bean:write>标签打印变量的示例如下。

```
<body>
  <%
    String test = "my name is gao hong yan";
    request.setAttribute("myname", test);
  %>
  <bean:write name="myname" />
</body>
```

在上面的程序示例中，通过定义一个 test 的字符串变量后再初始化内容，将这个 test 变量放入到 request 对象中，再通过<bean:write>标签进行输出，运行效果如图 8-2 所示。



图 8-2 显示结果

### 8.2.2 <bean:write>标签打印 Bean 的 property 属性值

- (1) 新建一个 Web 项目 Struts 8.2.2。
- (2) 添加 Struts 1.2 框架支持文件。
- (3) 新建一个 package 包，包名为 mypackage。
- (4) 在 mypackage 包中新建一个类，类的名字为 mybean，代码如下。

```
package mypackage;

public class mybean {
    private String username;

    public void setUsername(String username) {
        this.username = username;
    }

    public String getUsername() {
        return username;
    }
}
```

- (5) 新建一个 JSP 文件，文件名称为 showname.jsp，代码内容如下。

```
<body>
  <bean:write name="showname" property="username" />
</body>
```

在 JSP 页面中通过使用<bean:write>标签打印存放在上下文的 Bean 对象 showname 中的 username 属性的值，该实例中的 showname 对象是在 Action 中创建的。

(6) 新建一个 Action，如图 8-3 所示。

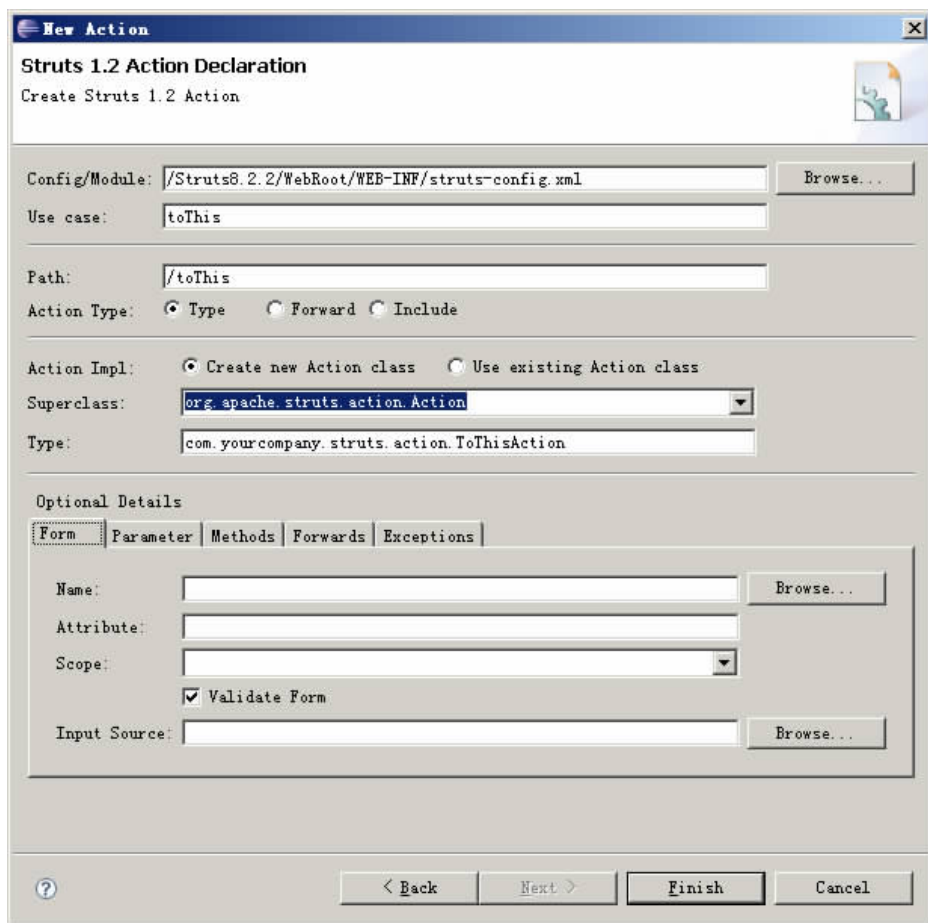


图 8-3 新建一个 Action

设置完成后，单击“Finish”按钮进行应用设置，创建 Action。

(7) 在“Forwards”属性页中，创建 Forward 转发，单击下方的“Forwards”属性页后，单击“Add”按钮进行添加，设置 Forward 内容如图 8-4 所示。

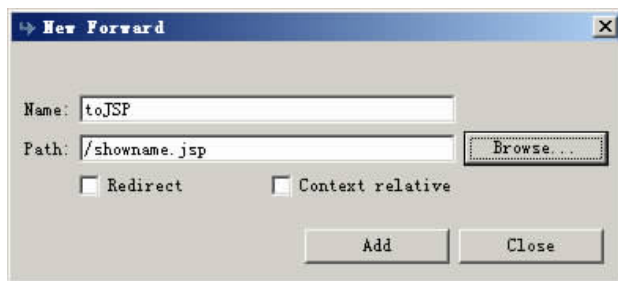


图 8-4 新建一个 Forward

单击“New Forward”对话框中的“Add”按钮后添加一个 Forward 对象，单击“Close”按钮，关闭当前的窗口。

(8) 单击“Finish”按钮应用设置。

(9) 更改 Action 代码如下：

```
public ActionForward execute(ActionMapping mapping, ActionForm form,
    HttpServletRequest request, HttpServletResponse response) {
    mypackage.mybean mybean_ref=new mypackage.mybean();
    mybean_ref.setUsername("gaohongyan");
    request.setAttribute("showname", mybean_ref);
    return mapping.findForward("toJSP");
}
```

在 Action 的代码中新建一个 mybean 的实例 mybean\_ref，设置 username 变量的值，再转发到 toJSP 逻辑路径中。

(10) 配置文件 struts-config.xml 内容如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE struts-config PUBLIC "-//Apache Software Foundation
    //DTD Struts Configuration 1.2 EN"
    //"http://struts.apache.org/dtds/struts-config_1_2.dtd">

<struts-config>
  <data-sources />
  <form-beans />
  <global-exceptions />
  <global-forwards />
  <action-mappings >
    <action path="/toThis"
      type="com.yourcompany.struts.action.ToThisAction">
      <forward name="toJSP" path="/showname.jsp" />
    </action>

  </action-mappings>

  <message-resources
parameter="com.yourcompany.struts.ApplicationResources" />
</struts-config>
```

(11) 部署项目，启动服务。

(12) 在浏览器的地址栏中输入 <http://localhost:8080/Struts8.2.2/toThis.do>。程序运行效果如图 8-5 所示。



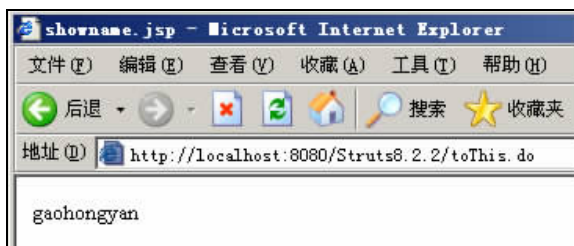


图 8-5 程序运行效果

### 8.2.3 <bean:write>标签 format 属性的应用

<bean:write>标签的 format 属性是格式化输出内容的。如果使用<bean:write>标签输出的是小数位数比较大的 Double 类型的话，在默认情况下它会截断多余的小数位数来显示，这样显示的结果如果不是想要的指定的小数位数，这时就需要进行格式化，如千分位格式化等。即输出为 13,160,000.00 这类的格式。

可以设置<bean:write>标签的 format 属性以处理截断多余小数位数的情况，即 format="0,000.00"。

新建一个 JSP 文件，输入如下代码。

```
<body>
  <%
    Double doubles = new Double(3.123456789123456789);
    request.setAttribute("doubles", doubles);
  %>
  <bean:write name="doubles" />
</body>
```

程序输出结果为 3.123456789123457，可以看出后面的 89 的小数位数被截断。

更改程序如下：

```
<body>
  <%
    Double doubles = new Double(3.123456789123456789);
    request.setAttribute("doubles", doubles);
  %>
  <bean:write name="doubles" format="0.000"/>
</body>
```

程序加入了 format 属性，运行结果为 3.123。在 format 属性中设置小数位数为 3，即 3 个 0。

### 8.2.4 <bean:write>标签 filter 属性的应用

filter 属性的作用是控制在输出 HTML 表单时所显示的页面效果。

- 当 filter 属性为 true 时，输出存放在 Bean 中相关属性的 HTML 标记的源代码里。
- 当 filter 属性为 false 时，将存放在 Bean 中相关属性以 HTML 代码所生成的效果显示。

举一个示例，新建一个 JSP 文件，内容如下：

```
<body>
  <%
    String table = "<table border=2><tr><td>my name is gao hong
                    yan</td></tr></table>";
    request.setAttribute("table", table);
  %>
  <bean:write name="table" filter="true" />
  <br>
  <br>
  <bean:write name="table" filter="false" />
</body>
```

程序运行结果如图 8-6 所示。



图 8-6 程序运行结果

从结果中可以看出，设置为 true 表示输出 HTML 的标记内容，而设置为 false 则显示出 HTML 代码所生成的效果。

## 8.3 <bean:parameter /> 标签读取 HTTP 请求的参数

在 JSP 的程序代码段中，经常使用如下代码：

```
<%
String id = request.getParameter("id");
%>
```

来取得 HTTP 参数的值，在 Struts 中替换的功能是使用 <bean:parameter> 标签。

### 8.3.1 使用 <bean:parameter /> 标签读取单个 http 参数

新建一个 JSP 文件，代码如下。

```
<%@ page language="java" pageEncoding="ISO-8859-1"%>

<%@ taglib uri="http://struts.apache.org/tags-bean" prefix="bean"%>
<%@ taglib uri="http://struts.apache.org/tags-html" prefix="html"%>
<%@ taglib uri="http://struts.apache.org/tags-logic" prefix="logic"%>
<%@ taglib uri="http://struts.apache.org/tags-tiles" prefix="tiles"%>
```

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html:html lang="true">
<head>
  <html:base />

  <title>index.jsp</title>

  <meta http-equiv="pragma" content="no-cache">
  <meta http-equiv="cache-control" content="no-cache">
  <meta http-equiv="expires" content="0">
  <meta http-equiv="keywords" content="keyword1,keyword2,keyword3">
  <meta http-equiv="description" content="This is my page">
  <!--
  <link rel="stylesheet" type="text/css" href="styles.css">
  -->
</head>
<body>
  <bean:parameter id="par_value" name="id" value="" />
  <bean:write name="par_value" />
</body>
</html:html>
```

在程序代码中的<bean:parameter>标签设置了 3 个属性：id、name、value。其中 name 属性对应 HTTP 参数的名称；value 属性是如果 HTTP 中没有此参数的名称的话，使用默认的值是多少；而 id 属性指将 name 参数的值取出后放到哪个变量中，在本例中将参数 id 的值取出来后放入 par\_value 的字符串变量中，如果没有 id 参数的话，则默认值为""。

程序运行效果如图 8-7 所示。



图 8-7 程序运行效果

### 8.3.2 使用<bean:parameter />标签读取数组型 HTTP 参数

在<bean:parameter>标签中如果想取得数组型参数值的话，则需要设置 multiple 属性。这时<bean:parameter>标签的 id 属性不是一个字符串类型，而是一个字符串数组类型 String[]。新建一个 JSP 文件，代码如下。

```
<%@ page language="java" pageEncoding="ISO-8859-1"%>

<%@ taglib uri="http://struts.apache.org/tags-bean" prefix="bean"%>
<%@ taglib uri="http://struts.apache.org/tags-html" prefix="html"%>
<%@ taglib uri="http://struts.apache.org/tags-logic" prefix="logic"%>
```

```
<%@ taglib uri="http://struts.apache.org/tags-tiles" prefix="tiles"%>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html:html lang="true">
<head>
    <html:base />

    <title>index.jsp</title>

    <meta http-equiv="pragma" content="no-cache">
    <meta http-equiv="cache-control" content="no-cache">
    <meta http-equiv="expires" content="0">
    <meta http-equiv="keywords" content="keyword1,keyword2,keyword3">
    <meta http-equiv="description" content="This is my page">
    <!--
    <link rel="stylesheet" type="text/css" href="styles.css">
    -->

</head>
<body>
    <bean:parameter id="par_value" name="id" multiple="yes" />
    <%
    out.println(par_value.length+"<br>");
        for (int i = 0; i < par_value.length; i++)
            out.println(par_value[i] + "<br>");
    %>
</body>
</html:html>
```

在浏览器中输入地址：<http://localhost:8080/test/index.jsp?id=100&id=200&id=300>，显示结果如图 8-8 所示。

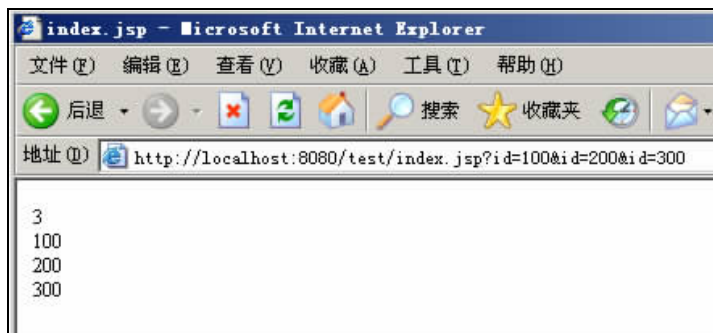


图 8-8 显示结果

## 8.4 <bean:message>标签显示资源文件中的文本消息

为了实现多语言跨平台的功能，需要将不同文字的内容集中放到资源文件中进行集中式的管理，这样不仅有利于项目的维护，还能够方便对文本资源的编码。

该标签用来从指定的 locale 中取回国际化的消息并输出，在这个过程中还可以传递参数。资源文件的 key 可以通过 key 直接指定，也可以通过 name 和 property 间接指定。

下面一起做一个显示资源文件中文本的示例吧！

(1) 新建 JSP 文件如下：

```
<%@ page language="java" pageEncoding="ISO-8859-1"%>

<%@ taglib uri="http://struts.apache.org/tags-bean" prefix="bean"%>
<%@ taglib uri="http://struts.apache.org/tags-html" prefix="html"%>
<%@ taglib uri="http://struts.apache.org/tags-logic" prefix="logic"%>
<%@ taglib uri="http://struts.apache.org/tags-tiles" prefix="tiles"%>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html:html lang="true">
<head>
    <html:base />

    <title>index.jsp</title>

    <meta http-equiv="pragma" content="no-cache">
    <meta http-equiv="cache-control" content="no-cache">
    <meta http-equiv="expires" content="0">
    <meta http-equiv="keywords" content="keyword1,keyword2,keyword3">
    <meta http-equiv="description" content="This is my page">
    <!--
    <link rel="stylesheet" type="text/css" href="styles.css">
    -->
</head>
<body>
    <bean:message key="Struts.begin" />
    <br>
    <br>
    <bean:message key="Struts.end"/>
</body>
</html:html>
```

<bean:message>标签中的 key 是指资源文件中的 key 的名称。

(2) 更改资源文件代码如图 8-9 所示。

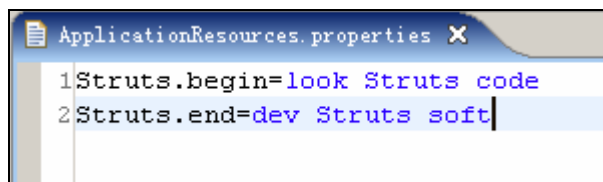


图 8-9 更改资源文件

(3) 程序运行结果如图 8-10 所示。



图 8-10 运行结果

用<bean:message>标签显示多资源文件的文本消息

在大的 Struts 项目中，对于不同功能、不同作用、不同语言的文本信息，为了管理上的方便，需要以多个资源文件的形式来进行管理，那么使用<bean:message>标签该如何正确实现显示多资源文件的文本内容呢？

- (1) 新建一个 Web 项目 Struts 8.4.1。
- (2) 本例是多资源文件的情况，所以需要创建多个资源文件。复制资源文件，在要复制的资源文件上单击鼠标右键，在弹出的菜单中选择“copy”命令，如图 8-11 所示。

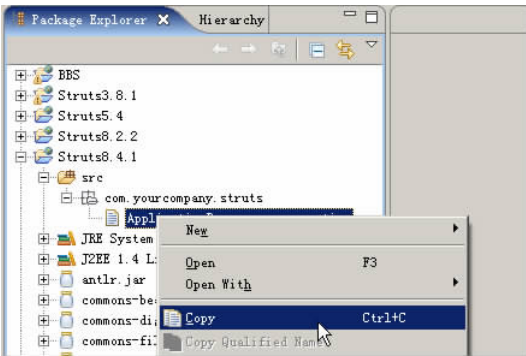


图 8-11 复制资源文件

- (3) 用鼠标右键单击“com.yourcompany.struts”节点，在弹出的菜单中选择“Paste”命令，粘贴资源文件，如图 8-12 所示。

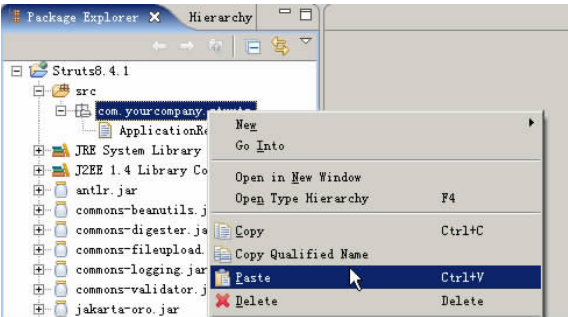


图 8-12 粘贴资源文件

(4) 弹出更改资源文件名的对话框, 如图 8-13 所示。

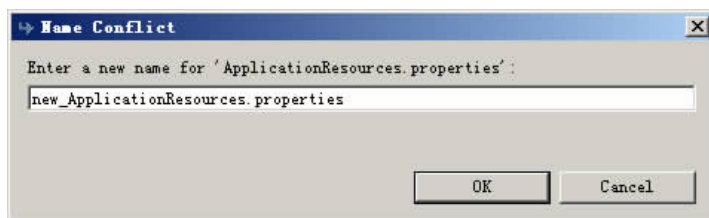


图 8-13 更改资源文件名

(5) 单击“OK”按钮后, 资源文件结构如图 8-14 所示。

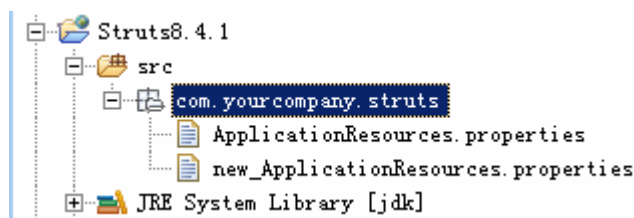


图 8-14 资源文件结构

(6) 更改资源文件内容, 前后效果如图 8-15 和图 8-16 所示。

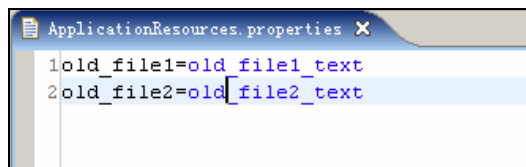


图 8-15 ApplicationResources.properties 资源文件内容——更改前

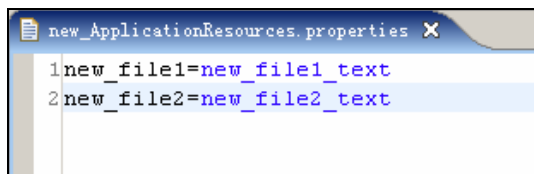


图 8-16 new\_ApplicationResources.properties 资源文件内容——更改后

(7) 更改配置文件 struts-config.xml 注册新的资源文件, 代码如下:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE struts-config PUBLIC "-//Apache Software Foundation
//DTD Struts Configuration 1.2 EN"
// "http://struts.apache.org/dtds/struts-config_1_2.dtd">

<struts-config>
  <data-sources />
  <form-beans />
  <global-exceptions />
  <global-forwards />
```

```
<action-mappings />
<message-resources
    parameter="com.yourcompany.struts.ApplicationResources" />
<message-resources
    parameter="com.yourcompany.struts.new_ApplicationResources"
    key="new"/>
</struts-config>
```

将新的资源文件定义别名为 new。

(8) 新建一个 JSP 文件，内容如下：

```
<%@ page language="java" pageEncoding="gb2312"%>

<%@ taglib uri="http://struts.apache.org/tags-bean" prefix="bean"%>
<%@ taglib uri="http://struts.apache.org/tags-html" prefix="html"%>
<%@ taglib uri="http://struts.apache.org/tags-logic" prefix="logic"%>
<%@ taglib uri="http://struts.apache.org/tags-tiles" prefix="tiles"%>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html:html lang="true">
<head>
    <html:base />

    <title>index.jsp</title>

    <meta http-equiv="pragma" content="no-cache">
    <meta http-equiv="cache-control" content="no-cache">
    <meta http-equiv="expires" content="0">
    <meta http-equiv="keywords" content="keyword1,keyword2,keyword3">
    <meta http-equiv="description" content="This is my page">
    <!--
    <link rel="stylesheet" type="text/css" href="styles.css">
    -->

</head>

<body>
    <bean:message key="new_file1" bundle="new" />
    <br>
    <bean:message key="new_file2" bundle="new" />
    <br>
    <bean:message key="old_file1" />
    <br>
    <bean:message key="old_file2" />
</body>
</html:html>
```

在 JSP 文件中，通过使用<bean:message>标签的 bundle 属性来定义显示哪个资源文件中的文本内容。



(9) 部署项目，启动服务，运行效果如图 8-17 所示。

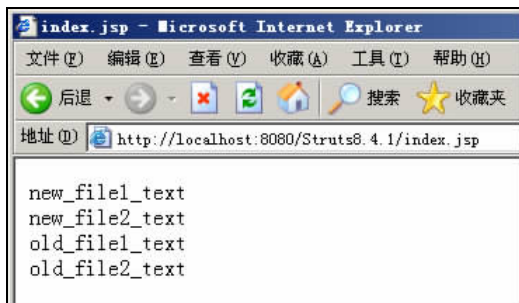


图 8-17 运行效果

## 8.5 在<bean:define>标签中定义变量

该标签用于定义一个新的变量，属性列表如图 8-18 所示。

Property	Value
<b>Attributes</b>	
id	
name	
property	
scope	
toScope	
type	
value	

图 8-18 属性列表

代码为<bean:define id="pagename" name="requestName"/>。

id 指的是 Bean 的变量名称，是唯一名字，在此定义的网页里面可以随便使用和访问；name 指的是从上一个页面或者 action 里面传过来的变量，可以是任何类型，也包括 Bean 类型。

例如，有一个 User 类，该类里面保存了 admin 管理员的个人信息，有身份证号（idcode）、性别（sex）、年龄（age）等，那怎么通过 Struts 的机制和标签去使用它呢？

- (1) 在 action 里面可以把 User 对象传出去，使用代码 request.setAttribute("user",User);。
- (2) 在 JSP 页面中可以使用下面的代码：

```
<bean:define id="admin" name="user"/>
```

- (3) 和 bean:write 搭配使用：

```
<bean:write name="admin" property="idcode"/>
<bean:write name="admin" property="sex"/>
<bean:write name="admin" property="age"/>
```

这样就可以输出当前 Bean 中的属性值了。

#### 8.5.1 在<bean:define>标签中定义字符串常量

例程如下。

```
<bean:define id="foo" value="This is a new String"/>
<bean:define id="bar" value='<%= "Hello, " + user.getName() %>' />
<bean:define id="last" scope="session" value='<%= request.getRequestURI() %>' />
```

通过在 value 属性中硬性地写入值，即可实现新建一个变量后，将变量的值初始化的功能。

#### 8.5.2 <bean:define>标签复制 Bean

例程如下。

```
<bean:define id="foo" name="bar" />
<bean:define id="baz" name="bop" type="com.mycompany.MyClass" />
```

通过设置 name 属性，即可实现 Bean 的复制，name 的属性值是存在于上下文的 Bean 的名称，可以通过使用 request 或 session 的 setAttribute 方法来存放 Bean。

#### 8.5.3 用<bean:define>标签复制现有 Bean 的属性给新的 Bean 属性

例程如下。

```
<bean:define id="foo" name="bar" property="baz" scope="request"
toScope="session" />
```

toScope 属性指新 Bean 的 scope，默认为 page，上段代码的功能是把名为 bar 的 Bean 的 baz 属性赋值给 foo，foo 的类型为 String（默认）。

# 第 9 章

## 关于 Struts 的其他内容

### 9.1 Struts 资源文件国际化

如果想要把 Struts 资源文件中的中文信息正确地显示到 JSP 页面上，那么就需要增加一些额外的工作，基于以前的情况，就得使用 `native2ascii` 命令进行编码的转换，可以实现将中文转成 Unicode 编码的功能，显示正确的中文信息。

#### 9.1.1 MyEclipse 保存资源文件的编码哨兵

在新建一个 Struts 项目时，都会自动地创建资源文件，但如果写入下面的这种形式时就会出现一个错误的提示，如图 9-1 所示。

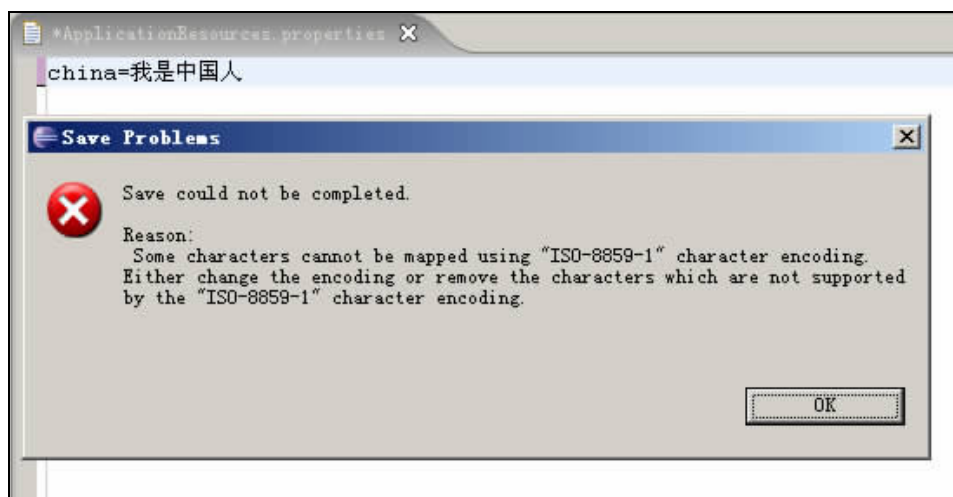


图 9-1 资源文件保存时的出错提示

出错的信息大概内容就是文件中的文字内容编码不对，不可以保存。针对这样的情况，使用传统的 `native2ascii` 命令就可以解决问题，但每编辑一次资源文件，就要使用一次 `native2ascii` 命令，是比较麻烦的。

#### 9.1.2 使用 MyEclipse 资源文件的插件 jinto

幸好，在使用 MyEclipse 时，使用针对国际化的第三方插件就可以解决上节遇到编码

及操作繁琐的问题。

进入下载地址 <http://www.guh-software.de/jinto.html>，选择下载 jinto 插件，该插件的使用方法非常简单，解压目录结构如图 9-2 所示。

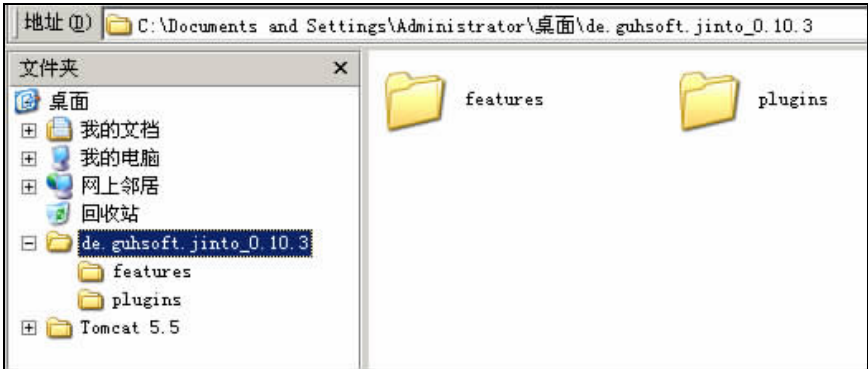


图 9-2 解压目录结构

将解压缩后的 features 和 plugins 目录复制到 MyEclipse 目录中的同名目录中即可，然后在启动 MyEclipse 时，双击资源文件后，就可以使用第三方的资源文件编码编辑器了，效果如图 9-3 所示。



图 9-3 运行效果

单击右上角的加号按钮和减号按钮来进行 key 的增加和删除的操作，编辑完成后单击“Save”按钮进行保存。

这时可以使用记事本来看一看，第三方的插件到底为我们做了一些什么样的操作呢？使用记事本打开资源文件后的内容如下。

```
china=\u6211\u662F\u4E2D\u56FD\u4EBA
```

可以看到原来中文的内容被第三方的插件自动转换编码了，以后再也不用手动使用命令去进行编码的转换，从而提高了开发效率。

## 9.2 在 Struts 的 URL 中传递中文参数

在大多数的情况下传递参数都为 `show.jsp?id=abc` 的形式，但如果传递的是中文参数则

需要做进一步的处理。

在 JSP 文件中如何传递中文化的 URL 参数呢？请看下面的示例。

(1) 第一部分 JSP 文件代码如下：

```
<%@ page language="java" pageEncoding="utf-8"%>
<%@ page import="java.net.URLEncoder"%>
<%@ taglib uri="http://struts.apache.org/tags-bean" prefix="bean"%>
<%@ taglib uri="http://struts.apache.org/tags-html" prefix="html"%>
<%@ taglib uri="http://struts.apache.org/tags-logic" prefix="logic"%>
<%@ taglib uri="http://struts.apache.org/tags-tiles" prefix="tiles"%>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html:html lang="true">

<body>
  <%
    String z = URLEncoder.encode("高红岩");
    pageContext.setAttribute("id", z);
  %>
  <html:link page="/2.jsp" paramName="id"
    paramId="name">2.jsp</html:link>
</body>
</html:html>
```

(2) 第二部分 JSP 文件代码如下：

```
<%@ page language="java" pageEncoding="utf-8"%>

<%@ taglib uri="http://struts.apache.org/tags-bean" prefix="bean"%>
<%@ taglib uri="http://struts.apache.org/tags-html" prefix="html"%>
<%@ taglib uri="http://struts.apache.org/tags-logic" prefix="logic"%>
<%@ taglib uri="http://struts.apache.org/tags-tiles" prefix="tiles"%>
<%@ page import="java.net.URLDecoder"%>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html:html lang="true">
<body>
  <%
    String name = URLDecoder.decode(request.getParameter("name"));
    out.println(name);
  %>
</body>
</html:html>
```

在传递中文参数时,必须先将中文进行编码,这里使用的是 java.net 包中的 URLDecoder 和 URLEncoder 两个类来进行中文参数编码的,可见传递中文参数和传递英文或数字参数的方法还是有所区别的,在使用上也要更加注意。

## 9.3 从不同的资源文件中显示信息

在大型的项目中，通常都需要配置多个资源文件，每个资源文件中的文本都是分类的，在使用 Struts 时，如何正确地显示出指定资源文件中的文本内容呢？

### 9.3.1 从不同的资源文件中显示信息的实例

(1) 新建一个 Struts 的 Web 项目 Struts9.3.1，并编辑自带的属性文件，如图 9-4 所示。

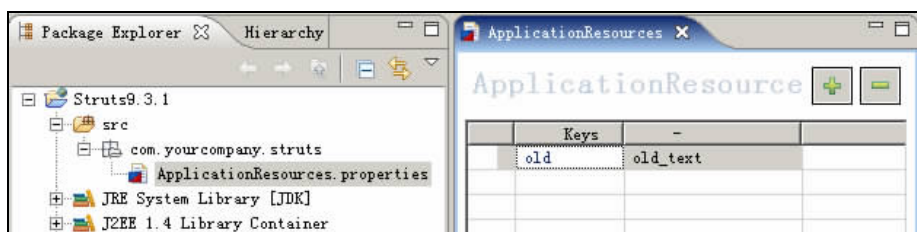


图 9-4 编辑自带的资源文件内容

(2) 新建一个资源文件，如图 9-5 所示。

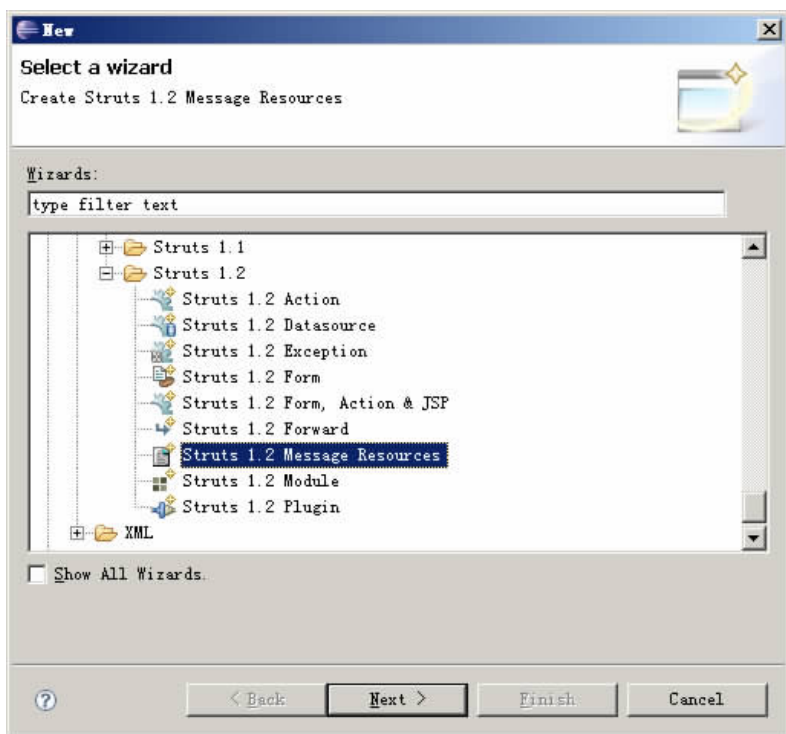


图 9-5 新建一个资源文件

在图 9-5 中单击“Next”按钮后，出现如图 9-6 所示的界面。

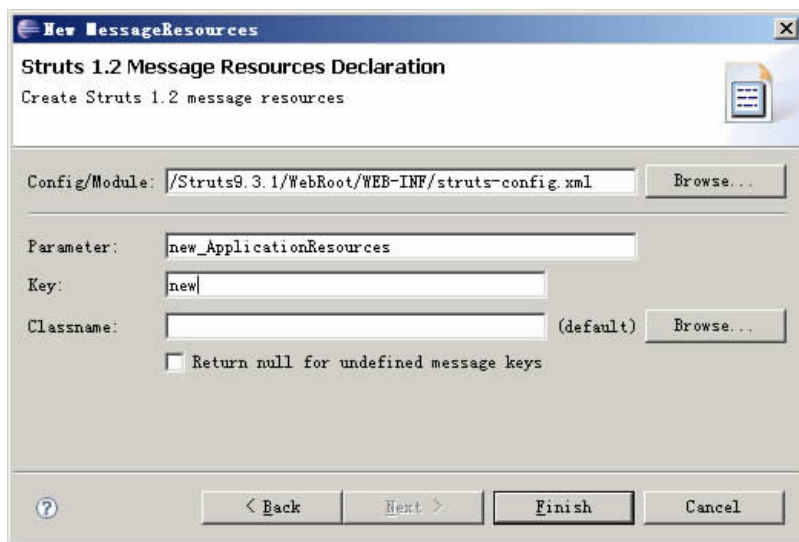


图 9-6 配置资源文件选项

其中在“Parameter”文本框中输入的值代表属性文件的主文件名，而在“Key”文本框中输入的值代表当前资源文件在系统中的别名。

输入成功后，直接单击“Finish”按钮创建应用。

(3) 新建一个资源文件后要在配置文件 struts-config.xml 中进行注册，当然注册的过程是 MyEclipse 自动帮我们创建的，注册后的 struts-config.xml 文件内容如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE struts-config PUBLIC "-//Apache Software Foundation
//DTD Struts Configuration 1.2 EN"
// "http://struts.apache.org/dtds/struts-config_1_2.dtd">

<struts-config>
  <data-sources />
  <form-beans />
  <global-exceptions />
  <global-forwards />
  <action-mappings />
  <message-resources
    parameter="com.yourcompany.struts.ApplicationResources" />
  <message-resources key="new" null="false"
    parameter="new_ApplicationResources" />
</struts-config>
```

(4) 新建一个支持 Struts 1.2 的 JSP 文件，更改代码如下：

```
<%@ page language="java" pageEncoding="utf-8"%>

<%@ taglib uri="http://struts.apache.org/tags-bean" prefix="bean"%>
<%@ taglib uri="http://struts.apache.org/tags-html" prefix="html"%>
<%@ taglib uri="http://struts.apache.org/tags-logic" prefix="logic"%>
```

```
<%@ taglib uri="http://struts.apache.org/tags-tiles" prefix="tiles"%>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html:html lang="true">
<head>
    <html:base />

    <title>index.jsp</title>

    <meta http-equiv="pragma" content="no-cache">
    <meta http-equiv="cache-control" content="no-cache">
    <meta http-equiv="expires" content="0">
    <meta http-equiv="keywords" content="keyword1,keyword2,keyword3">
    <meta http-equiv="description" content="This is my page">
    <!--
    <link rel="stylesheet" type="text/css" href="styles.css">
    -->

</head>

<body>
```

默认资源文件中的文本内容：

```
<bean:message key="old" />
<br>
<br>
```

新创建的资源文件中的文本内容：

```
<bean:message key="new" bundle="new" />
</body>
</html:html>
```

<bean:message>标签中的 bundle 属性是指定某个资源文件别名，在刚才创建资源文件时，配置别名为 new。在使用多个资源文件里的内容时，属性 bundle 属性很重要。

(5) 部署项目，启动服务，程序运行结果如图 9-7 所示。



图 9-7 程序运行结果



### 9.3.2 优化新建资源文件目录结构

在上一节创建资源文件时，发现目录结构如图 9-8 所示。

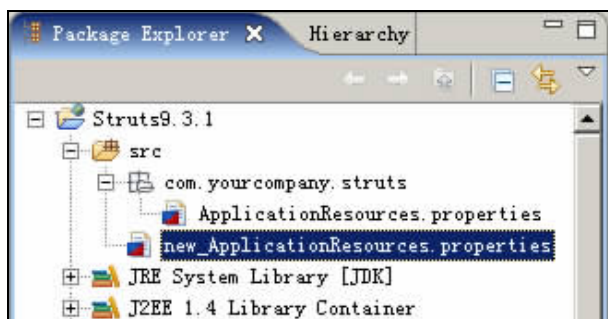


图 9-8 错位的目录结构

原本内容不一样，文件类型一样的资源文件所放的位置却不一样，该怎样处理这样的问题呢？

#### 1. 移动目录结构

将 `new_ApplicationResources.properties` 资源文件用鼠标拖动到 `com.yourcompany.struts` 节点中。

#### 2. 更改 `struts-config.xml` 配置文件

还需要在配置文件中指定资源文件新的目录结构，改变后的配置文件内容如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE struts-config PUBLIC "-//Apache Software Foundation
//DTD Struts Configuration 1.2 EN"
// "http://struts.apache.org/dtds/struts-config_1_2.dtd">

<struts-config>
  <data-sources />
  <form-beans />
  <global-exceptions />
  <global-forwards />
  <action-mappings />
  <message-resources
    parameter="com.yourcompany.struts.ApplicationResources" />
  <message-resources key="new" null="false"
    parameter="com.yourcompany.struts.new_ApplicationResources" />
</struts-config>
```

从代码中可以看出，原来的配置文件中的内容使用的是系统新建配置文件时默认的路径，在这个实例中需要在 `new_ApplicationResources` 的前面加入 `com.yourcompany.struts`。代码，这样才可以在改变新的资源文件路径后，正确地显示资源文件中的文本内容。

## 9.4 没有登录不能访问非 index.jsp 的 JSP 页面

在以往的项目开发中，针对这样的问题通常的解决方案是在每一个 JSP 页面中都判断一下 session 对象中有没有存在的对象，如果有，则显示\*.jsp 文件内容，否则使用 JSP 中的 response 对象进行重定向 response.sendRedirect("other.jsp")，或使用<jsp:forward>转发到登录的页面。

但是，如果在项目中有数个 JSP 页面，难道非要在每一个 JSP 页面中写入上面的代码吗？没有必要，使用 Servlet 的过滤功能即可方便快捷地实现这样的功能。

### 解决方案实例

(1) 新建一个支持 Struts 1.2 的 Web 项目 Struts9.4.1

(2) 在 src 中新建一个包 myFilter，再在该包中新建一个 loginFilter.java 类，loginFilter.java 的功能就是过滤 HTTP 的请求，代码如下。

```
package myFilter;

import java.io.IOException;

import javax.servlet.Filter;
import javax.servlet.FilterChain;
import javax.servlet.FilterConfig;
import javax.servlet.ServletException;
import javax.servlet.ServletRequest;
import javax.servlet.ServletResponse;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;

public class loginFilter implements Filter {

    private String onErrorUrl;

    public void init(FilterConfig filterConfig) throws ServletException {

        onErrorUrl = filterConfig.getInitParameter("onError");
        if (onErrorUrl == null || "".equals(onErrorUrl)) {
            onErrorUrl = "/index.jsp";
        }
    }

    public void doFilter(ServletRequest request, ServletResponse response,
        FilterChain chain) throws IOException, ServletException {
        HttpServletRequest req = (HttpServletRequest) request;
        HttpServletResponse res = (HttpServletResponse) response;

        HttpSession session = req.getSession();
        String username = (String) session.getAttribute("username");
        boolean bl = true;
```

```
        if (username == null) {
            bl = false;
        } else {
        }
        if (bl) {
            chain.doFilter(request, response);
        } else {
            req.getRequestDispatcher(onErrorUrl).forward(req, res);
        }
    }

    public void destroy() {
    }
}
```

在 `loginFilter.java` 类中，通过使用 `doFilter` 方法来进行 HTTP 请求的过滤，每当发出一个 HTTP 请求，过滤器先截获并进行处理，再将处理后的请求分发，以使得软件的流程能正常进行。在 `doFilter` 方法中首先取得 session 中的 key 为 `username`（用户名）的属性，如果该属性为空代表没有更新，则赋值 `bl` 为 `false`。接下来再判断 `bl` 变量的值，如果为 `true` 即为登录，就将请求进行分发：`chain.doFilter(request,response)`；否则使用转发代码转发到变量 `onErrorUrl` 所代表的页面中，该页面是在 `web.xml` 文件中通过参数传递进来的，如何传递进来的呢？在 `init` 方法中通过使用 `filterConfig.getInitParameter("onError")` 代码取得。

该过滤器的主要内容就是检查 session 中的 `username` 内容是否为空，为空则没有登录，转发到登录页面，如果不为空，则分发请求。

（3）在第（2）步配置完了过滤器的代码，这步还需要将过滤器的代码注册在系统中，需要改动 `web.xml` 文件，改动后完整的 `web.xml` 文件中的代码如下。

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns="http://java.sun.com/xml/ns/j2ee"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="2.4"
  xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee
    http://java.sun.com/xml/ns/j2ee/web-app_2_4.xsd">
  <servlet>
    <servlet-name>action</servlet-name>
    <servlet-class>
      org.apache.struts.action.ActionServlet
    </servlet-class>
    <init-param>
      <param-name>config</param-name>
      <param-value>/Web-INF/struts-config.xml</param-value>
    </init-param>
    <init-param>
      <param-name>debug</param-name>
      <param-value>3</param-value>
    </init-param>
    <init-param>
      <param-name>detail</param-name>
      <param-value>3</param-value>
  </servlet>

```

```
</init-param>
<load-on-startup>0</load-on-startup>
</servlet>
<servlet-mapping>
  <servlet-name>action</servlet-name>
  <url-pattern>*.do</url-pattern>
</servlet-mapping>

<filter>
  <filter-name>loginFilter</filter-name>
  <filter-class>myFilter.loginFilter</filter-class>
  <init-param>
    <param-name>onError</param-name>
    <param-value>/form/index.jsp</param-value>
  </init-param>
</filter>
<filter-mapping>
  <filter-name>loginFilter</filter-name>
  <url-pattern>/havesession/*</url-pattern>
</filter-mapping>
</web-app>
```

其中注册过滤器时只需要加入如下代码。

```
<filter>
  <filter-name>loginFilter</filter-name>
  <filter-class>myFilter.loginFilter</filter-class>
  <init-param>
    <param-name>onError</param-name>
    <param-value>/form/index.jsp</param-value>
  </init-param>
</filter>
<filter-mapping>
  <filter-name>loginFilter</filter-name>
  <url-pattern>/havesession/*</url-pattern>
</filter-mapping>
```

在注册过滤器的代码中，最重要的就是<filter-mapping>节点中的<url-pattern>的值/havesession/\*，该值的意思就是过滤（截获）/havesession/路径中的所有 HTTP 请求，获取所有 http://localhost:8080/test/havesession/下的所有请求，test 是工程项目的名称。

那么在本例中，为什么偏偏还要特别声明/havesession/请求路径呢？原因是：如果不加入一个识别的路径的话，则所有的请求都被过滤，这样当用户不登录即可查看新闻内容时，按功能逻辑讲，应该给用户显示出新闻信息，但恰恰是相反的，用户请求查看新闻的内容被过滤，并检查出 session 没有值，转发到登录页面。针对这样情况，就得在系统配置时在过滤器中分别配置当用户无登录即可查看新闻内容和用户必须登录查看新闻内容的两种条件了。

本例中所有以/havesession/开头的请求路径都属于必须登录查看新闻内容的条件。做到这步，已经将过滤器的 Java 类和 web.xml 配置完成。

(4) 新建一个 JSP、Action、Form，如图 9-9 所示。

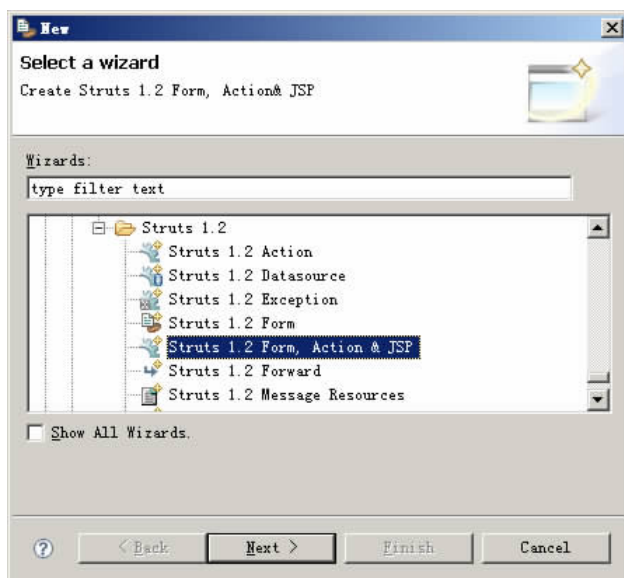


图 9-9 新建 JSP、Action、Form

单击“Next”按钮，配置 ActionForm 和 JSP 文件，如图 9-10 所示。

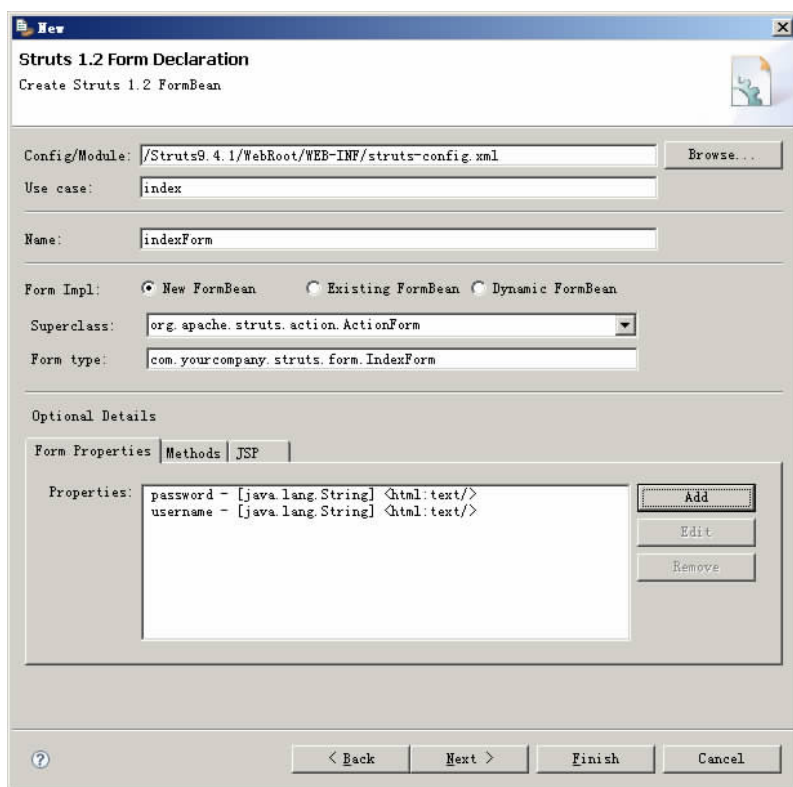


图 9-10 配置 ActionForm

在“JSP”属性页创建 JSP 文件，如图 9-11 所示。



图 9-11 创建 JSP 文件

单击“Next”按钮，继续操作，开始配置 Action，Action 的配置使用默认值，直接单击“Finish”按钮后完成配置。

(5) 更改/form/index.jsp 的程序代码如下。

```
<%@ page language="java" pageEncoding="gb2312"%>
<%@ taglib uri="http://jakarta.apache.org/struts/tags-bean"
    prefix="bean"%>
<%@ taglib uri="http://jakarta.apache.org/struts/tags-html"
    prefix="html"%>
<%@ taglib uri="http://jakarta.apache.org/struts/tags-logic"
    prefix="logic"%>

<html>
    <head>
        <title>JSP for IndexForm form</title>
    </head>
    <body>
        <logic:present name="username">
            欢迎 : <bean:write name="username" />
        </logic:present>
        <br>
        <html:form action="/index">
            password : <html:text property="password" />
            <html:errors property="password" />
            <br />
            username : <html:text property="username" />
            <html:errors property="username" />
            <br />
            <html:submit />
            <html:cancel />
        </html:form>
        <br>
        <html:link action="/havesession/show.do">to show.jsp
            page</html:link>
    </body>
</html>
```

在 JSP 程序中使用<login:present>标签来进行用户名的判断，如果在 session 中有 username 的对象则输出 username 变量的值。下面还使用了<html:link>标签来做一个链接，该链接的路径是/havesession/，代表当前链接的网页内容必须要登录才可以查看，参看 web.xml 的配置条件。

(6) 当单击 JSP 页面的提交按钮后，将当前的数据提交到 Action 中，更改后的 Action 完整代码如下。

```
package com.yourcompany.struts.action;

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.ActionError;
import org.apache.struts.action.ActionErrors;
import org.apache.struts.action.ActionForm;
import org.apache.struts.action.ActionForward;
import org.apache.struts.action.ActionMapping;

import com.yourcompany.struts.form.IndexForm;

public class IndexAction extends Action {
    public ActionForward execute(ActionMapping mapping, ActionForm form,
        HttpServletRequest request, HttpServletResponse response) {
        IndexForm indexForm = (IndexForm) form;

        ActionErrors error = new ActionErrors();

        if ("gaohongyan".equals(indexForm.getUsername())) {
            if ("123".equals(indexForm.getPassword())) {
                HttpSession session = request.getSession();
                session.setAttribute("username",
                    indexForm.getUsername());
            }
        } else {
            error.add("password", new
                ActionError("usernameorpasswordwrong"));
            error.add("username", new
                ActionError("usernameorpasswordwrong"));
        }

        if (error == null) {
            return mapping.getInputForward();
        } else {
            this.saveErrors(request, error);
            return mapping.getInputForward();
        }
    }
}
```

在 Action 中来判断用户名是不是 gaohongyan 和密码是不是 123, 该例将用户名和密码硬编码在程序中, 具体的项目开发实际是从数据库中提取的。

如果输入错误的用户名或密码, 单击“Submit”按钮后, 出现错误提示, 效果如图 9-12 所示。

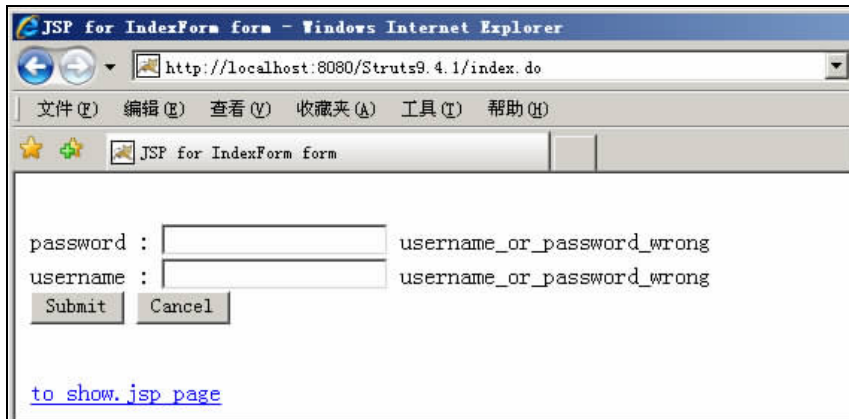


图 9-12 出错提示

如果输入正确的用户名 gaohongyan 和密码 123, 单击“Submit”按钮后, 出现如图 9-13 所示的效果。

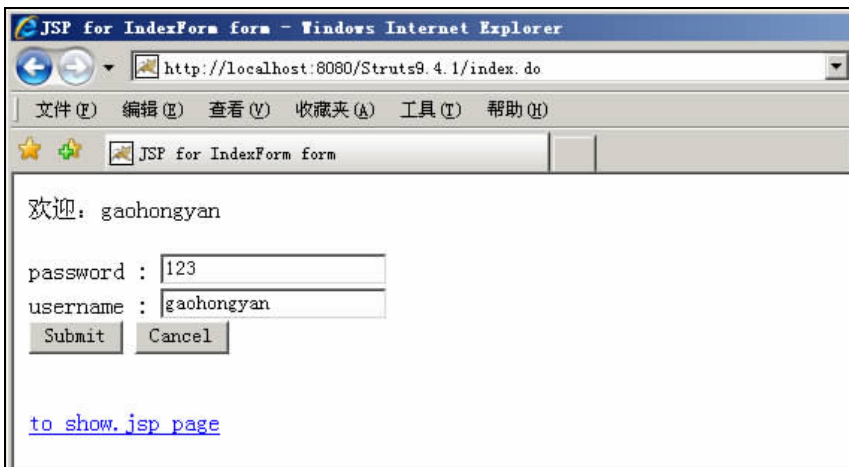


图 9-13 登录成功

(7) 新建一个 forward 的 action, 配置完成后的 struts-config.xml 文件内容如下。

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE struts-config PUBLIC "-//Apache Software Foundation
//DTD Struts Configuration 1.2 EN"
// "http://struts.apache.org/dtds/struts-config_1_2.dtd">

<struts-config>
  <data-sources />
```



```
<form-beans >
  <form-bean name="indexForm"
              type="com.yourcompany.struts.form.IndexForm" />
</form-beans>

<global-exceptions />
<global-forwards />
<action-mappings >
  <action
    attribute="indexForm"
    input="/form/index.jsp"
    name="indexForm"
    path="/index"
    scope="request"
    type="com.yourcompany.struts.action.IndexAction" />
  <action forward="/form/show.jsp" path="/havesession/show" />
</action-mappings>

<message-resources
  parameter="com.yourcompany.struts.ApplicationResources" />
</struts-config>
```

创建一个 forward 的 action，并且 path 为满足过滤条件的路径/havesession/\*。

(8) 在 form 目录中新建一个 show.jsp 文件。

(9) 部署项目，启动服务。

(10) 新打开一个浏览器窗口，输入：http://localhost:8080/Struts9.4.1/havesession/show.do，该 HTTP 请求被过滤器截获，发现在 session 中没有 username 的数据，则转到 index.jsp 页面中，可以看到地址栏中的地址和实际页面不对应，如图 9-14 所示。

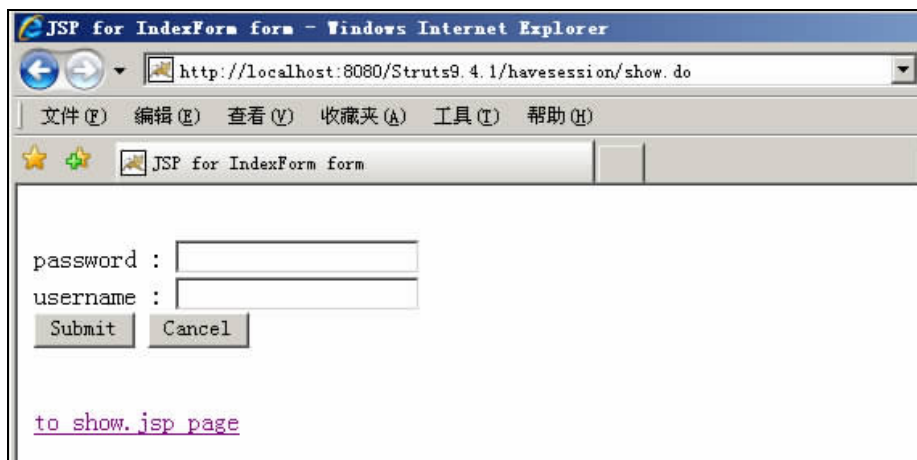


图 9-14 没有登录的效果

这时如果在“username”文本框中输入 gaohongyan，在“password”中输入 123 后单击“Submit”按钮，效果如图 9-15 所示。

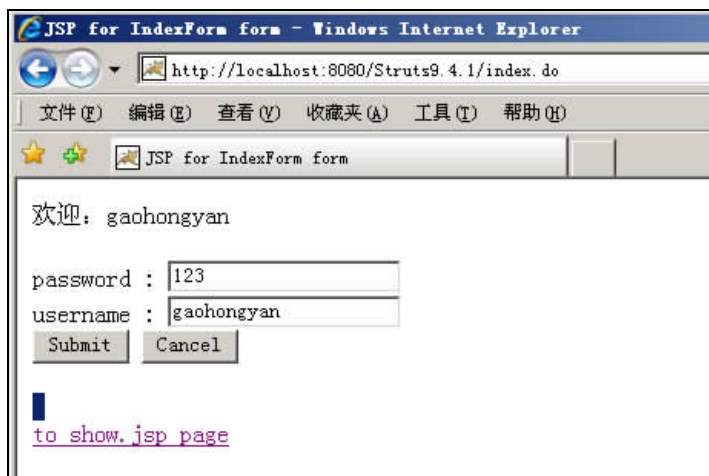


图 9-15 登录成功后转回

此时已经成功登录，这时单击下面的 to show.jsp page 的超级链接，出现如图 9-16 所示的界面。

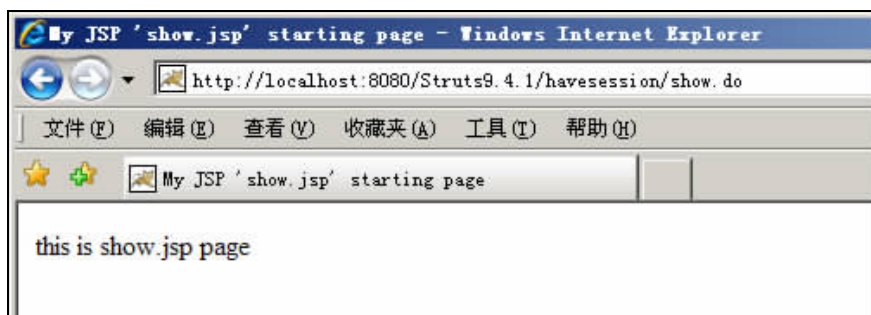


图 9-16 正确显示 show.jsp 文件内容

至此本例结束，把所有想要必须登录才可以看到的数据内容放到/havesession/请求路径中即可。

通过本例读者可以掌握在 Struts 中怎么样实现没有登录即转到 index.jsp 页面的效果。当然在具体的项目中，可以根据具体的细节来进行扩展。

## 9.5 设置应用的默认页面

如果需要一些特殊的需求，比如网站的默认起始页面就是 ghy.jsp，非 index.jsp，这样的情况，在 IIS 中可以很轻松地使用 GUI 工具进行实现，在 JSP 中简单地改动 web.xml 文件来达到这样的需求。

下面介绍设置应用的默认页面的实例。

(1) 新建 JSP、Action、ActionForm。JSP 的文件名为 ghy.jsp，文件存放在 form 目录下，项目文件结构如图 9-17 所示。

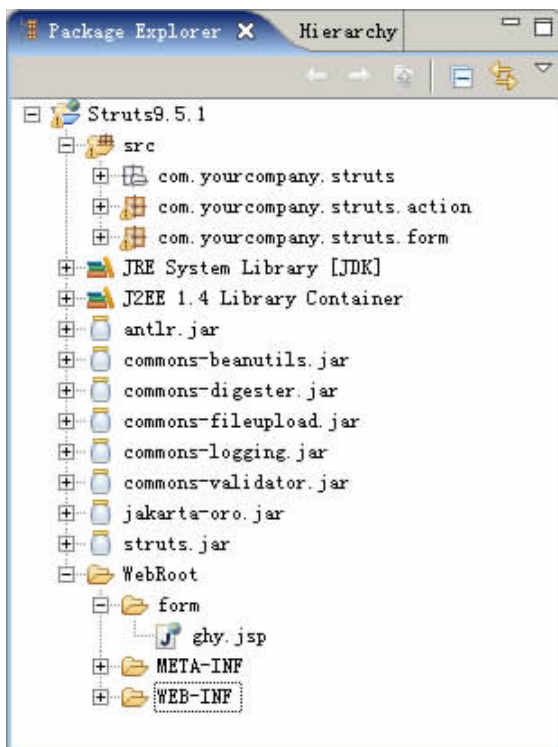


图 9-17 项目文件结构

(2) 更改 Web-INF 目录下的 web.xml 文件，内容如下。

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns="http://java.sun.com/xml/ns/j2ee"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="2.4"
  xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee
    http://java.sun.com/xml/ns/j2ee/web-app_2_4.xsd">
  <servlet>
    <servlet-name>action</servlet-name>
    <servlet-class>
      org.apache.struts.action.ActionServlet
    </servlet-class>
    <init-param>
      <param-name>config</param-name>
      <param-value>/Web-INF/struts-config.xml</param-value>
    </init-param>
    <init-param>
      <param-name>debug</param-name>
      <param-value>3</param-value>
    </init-param>
    <init-param>
      <param-name>detail</param-name>
      <param-value>3</param-value>
    </init-param>
    <load-on-startup>0</load-on-startup>
  </servlet>
</web-app>
```

```
</servlet>
<servlet-mapping>
    <servlet-name>action</servlet-name>
    <url-pattern>*.do</url-pattern>
</servlet-mapping>
<welcome-file-list>
    <welcome-file>/form/ghy.jsp</welcome-file>
</welcome-file-list>
</web-app>
```

(3) 部署项目, 启动服务, 这时在浏览器地址栏中输入 `http://localhost:8080/Struts9.5.1/`, 则显示如图 9-18 所示的运行效果。

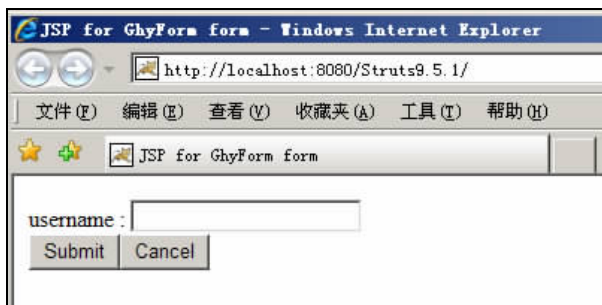


图 9-18 执行/form/ghy.jsp 文件

## 9.6 URL 重写技术

在开发 Web 技术中, URL 重写是设计一个商业网站的必要技术, URL 重写有很多非常优秀的特点, 在 JSP 技术中也有 URL 重写的插件, 使用这样的插件, 可以轻易地实现在 JSP 软件项目中的地址变换。

URL 重写技术主要有以下优点。

### 1. 满足搜索引擎的要求

某些搜索引擎不能支持动态页面的抓取, 大量的信息就不能被潜在用户搜索到。用 `UrlRewrite` 技术可以把 `http://server/news.asp?id=111` 变成 `http://server/news/111.htm` 这样的形式, 它们被搜索引擎收录的几率就大大增加了。Google 虽然可以抓取动态页面, 但是 Google 对动态页面的评分一般低于静态页面, 所以, 对大量信息发布的网站, 把网站地址改变成静态的绝对地址是值得的。

### 2. 隐藏技术实现, 提高网站的移植性

每个页面都挂着鲜明的 .asp、.jsp 开发语言的标记, 可以一眼让人看出网站使用什么语言开发的, 而且在改变网站语言的时候, 你需要改动大量的链接。可以用 `UrlRewrite` 技术隐藏语言技术的实现细节, 这样修改移植都很方便。

### 3. 满足美感的要求

对于追求完美主义的网站设计师，即使是网页的地址也要使其看起来简洁明快。形如 `http://server/news.asp?channel=3&id=111` 的网页地址，肯定不是完美主义者的观点，用 `UrlRewrite` 技术，可以把它变成 `http://server/news/3/111.htm`。

#### URL 重写技术实例

(1) 新建一个 Action，如图 9-19 所示。

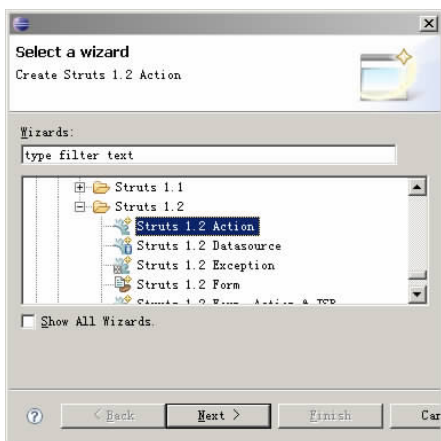


图 9-19 新建一个 Action

(2) 配置 Action，如图 9-20 所示。

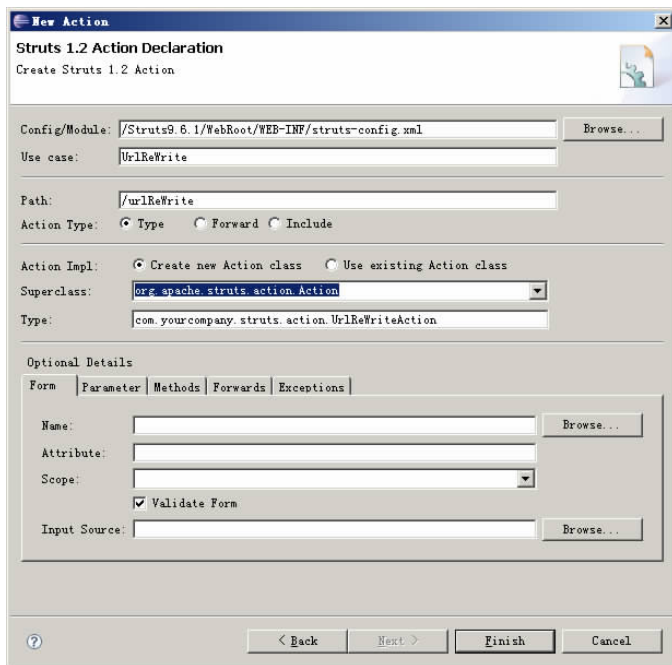


图 9-20 配置 Action

在配置 Action 时,将 path 路径设置为 `UrlRewrite`,在程序中访问这个 Action 的路径就是 `/UrlRewrite.do` 的形式了。

(3) 更改 Action 的 Java 代码如下。

```
package com.yourcompany.struts.action;

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;

public class UrlRewriteAction extends Action {

    public ActionForward execute(ActionMapping mapping, ActionForm form,
        HttpServletRequest request, HttpServletResponse response) {
        System.out.println("city=" + request.getParameter("city"));
        System.out.println("name=" + request.getParameter("name"));
        return null;
    }
}
```

(4) 新建一个 JSP 文件,内容如下。

```
<%@ page language="java" pageEncoding="gb2312"%>

<%@ taglib uri="http://struts.apache.org/tags-bean" prefix="bean"%>
<%@ taglib uri="http://struts.apache.org/tags-html" prefix="html"%>
<%@ taglib uri="http://struts.apache.org/tags-logic" prefix="logic"%>
<%@ taglib uri="http://struts.apache.org/tags-tiles" prefix="tiles"%>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html:html lang="true">

<body>
    <html:link page="/personal/jilin/gaohongyan">单击这个链接后,将 jilin 和
    gaohongyan 作为参数传递到 Action 中处理</html:link>
</body>
</html:html>
```

(5) 将下载下来的 `urlrewritefilter-2.6.zip` 目录解压,将里面的 JAR 文件和 XML 配置文件导入到项目工程的相对应的目录中,项目文件的结构如图 9-21 所示。

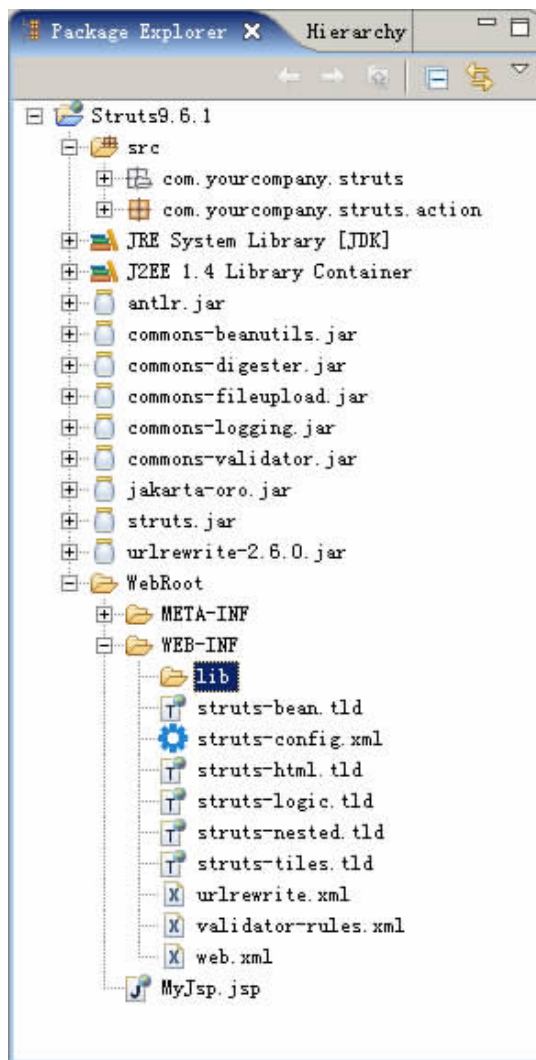


图 9-21 项目文件结构

(6) 更改 urlrewrite.xml 文件内容，添加规则，改动后的程序代码如下。

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE urlrewrite PUBLIC "-//tuckey.org//DTD UrlRewrite 2.6//EN"
    "http://tuckey.org/res/dtds/urlrewrite2.6.dtd">

<urlrewrite>

    <rule>
        <from>/personal/([a-z]+)/([a-z]+)</from>
        <to type="forward">/urlReWrite.do?city=$1&name=$2</to>
    </rule>

</urlrewrite>
```

(7) 部署项目，启动服务。

(8) 在浏览器的地址栏中输入 `http://localhost:8080/Struts9.6.1/MyJsp.jsp` 后出现如图 9-22 所示的界面。

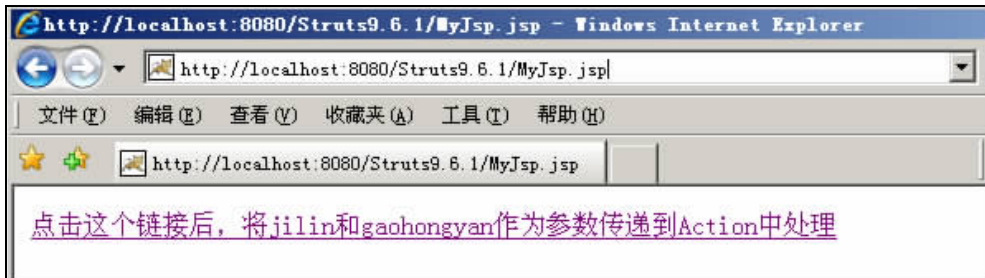


图 9-22 运行效果

(9) 查看 MyEclipse 的控制台，会发现打印出如下内容。

```
city=jilin  
name=gaohongyan
```

结果如图 9-23 所示。

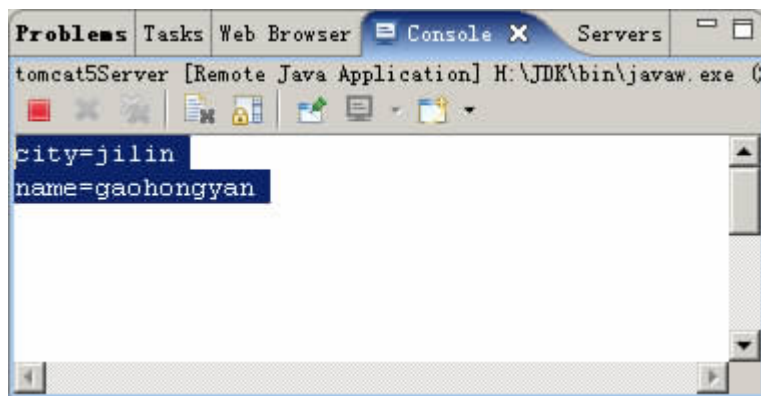


图 9-23 打印结果

在本例中使用了 Struts 的 \*.do 的 URL 重写，在这里面不仅可以将 URL 转到 Struts 的 Action 类中，也可以转到 \*.jsp 文件中，只需要将：

```
<rule>  
  <from>/personal/([a-z]+)/([a-z]+)</from>  
  <to type="forward">/urlRewrite.do?city=$1&name=$2</to>  
</rule>
```

改成：

```
<rule>  
  <from>/personal/([a-z]+)/([a-z]+)</from>  
  <to type="forward">/urlRewrite.jsp?city=$1&name=$2</to>  
</rule>
```



即可，但是为了实现基于 MVC 结构的项目，建议还是在 Action 中处理传参请求，再转发给相应的 JSP 页面进行显示数据。

关于本插件的使用及更多规则的实现，请登录 <http://tuckey.org/urlrewrite/> 进行参考。

## 9.7 使用 Struts 多语言切换的情况

在做外包的项目时，经常需要在页面上实现语言切换的功能，如果在没有国际化技术的支持下，需要针对一个网站做两种语言的版本，例如中文和日文，这种切换语言的技术如何在 Struts 中实现呢？

在 Struts 中可以设置资源文件的文件名称，来自动化实现多语言之间的方便切换，下面一起来做一个实例。

使用 Struts 多语言切换的实例

- (1) 新建一个 Web 项目 Struts9.7.1，添加 Struts 1.2 框架支持文件。
- (2) 使用 MyEclipse 自动新建了一个默认的资源文件，如图 9-24 所示。

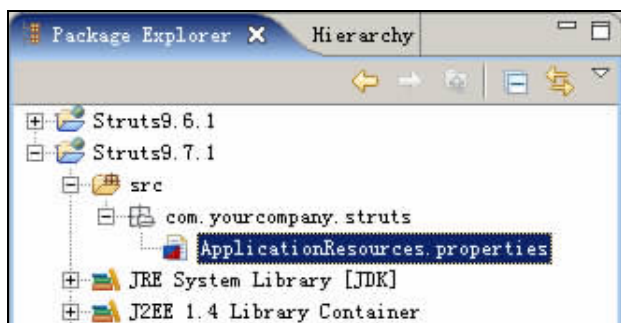


图 9-24 默认的资源文件

(3) 在这里不要使用默认的资源文件，需要改动一下文件的名称，在默认的资源文件上单击鼠标右键，在弹出的菜单中选择“Refactor” “Rename” 命令进行重命名，如图 9-25 所示。

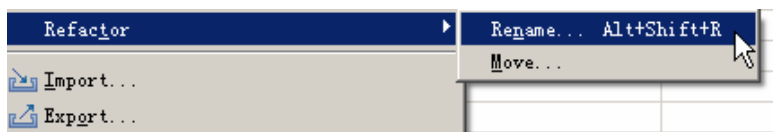


图 9-25 默认资源文件的重命名菜单

- (4) 将默认的资源文件重命名为：

```
ApplicationResources_en.properties
```

- (5) 编辑 ApplicationResources\_en.properties 文件中的内容为：

```
name=gaohongyan
```

(6) 英文版的资源文件新建完了,这次该新建中文版的资源文件了,用鼠标单击英文版的资源后文件,按“Ctrl+C”组合键进行复制,然后再按“Ctrl+V”组合键进行资源文件的粘贴,出现如图 9-26 所示窗口。

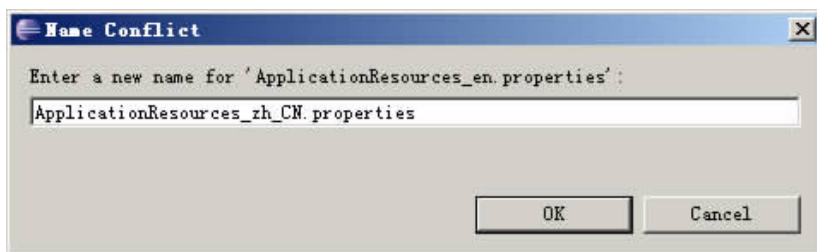


图 9-26 更改中文版资源文件名称

单击“OK”按钮后,即创建了英文和中文版的资源文件。  
项目资源文件结构如图 9-27 所示。

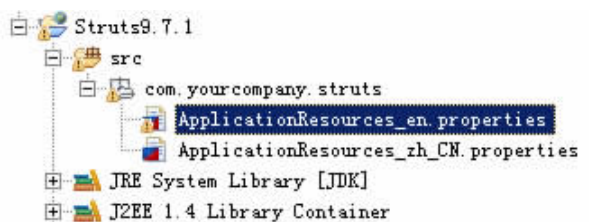


图 9-27 资源文件结构

(7) 新建一个 Action,在该 Action 中改变使用的资源文件,从而实现多语言的切换。  
设置 Action 的访问路径为/reLan。Action 的程序代码如下。

```
package com.yourcompany.struts.action;

import java.util.Locale;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import org.apache.struts.Globals;
import org.apache.struts.action.Action;
import org.apache.struts.action.ActionForm;
import org.apache.struts.action.ActionForward;
import org.apache.struts.action.ActionMapping;

public class ReLanAction extends Action {

    public ActionForward execute(ActionMapping mapping, ActionForm form,
        HttpServletRequest request, HttpServletResponse response) {
        String lan = request.getParameter("relan");
        if (lan.equals("1")) {
            request.getSession().setAttribute(Globals.LOCALE_KEY,
```

```
        Locale.CHINA);
    } else if (lan.equals("0")) {
        request.getSession().setAttribute(Globals.LOCALE_KEY,
            Locale.ENGLISH);
    } else {
        request.getSession().setAttribute(Globals.LOCALE_KEY,
            Locale.CHINA);
    }
    return mapping.findForward("index");
}
}
```

在 Action 中，通过取得 relan 的参数值来设置当前 Locale 的语种。

(8) 新建一个 JSP 文件，代码如下。

```
<%@ page language="java" pageEncoding="gb2312"%>

<%@ taglib uri="http://struts.apache.org/tags-bean" prefix="bean"%>
<%@ taglib uri="http://struts.apache.org/tags-html" prefix="html"%>
<%@ taglib uri="http://struts.apache.org/tags-logic" prefix="logic"%>
<%@ taglib uri="http://struts.apache.org/tags-tiles" prefix="tiles"%>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html:html lang="true">
<head>
    <html:base />

    <title>index.jsp</title>

    <meta http-equiv="pragma" content="no-cache">
    <meta http-equiv="cache-control" content="no-cache">
    <meta http-equiv="expires" content="0">
    <meta http-equiv="keywords" content="keyword1,keyword2,keyword3">
    <meta http-equiv="description" content="This is my page">
    <!--
    <link rel="stylesheet" type="text/css" href="styles.css">
    -->

</head>

<body>
    <html:link page="/reLan.do?relan=0">English</html:link>
    <html:link page="/reLan.do?relan=1">Chinese</html:link>
    <bean:message key="name" />
</body>

</html:html>
```

(9) struts-config.xml 文件内容如下。

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE struts-config PUBLIC "-//Apache Software Foundation
//DTD Struts Configuration 1.2 EN"
// "http://struts.apache.org/dtds/struts-config_1_2.dtd">

<struts-config>
  <data-sources />
  <form-beans />
  <global-exceptions />
  <global-forwards>
    <forward name="index" path="/index.jsp" />
  </global-forwards>

  <action-mappings>
    <action path="/reLan"
      type="com.yourcompany.struts.action.ReLanAction" />
  </action-mappings>

  <message-resources
    parameter="com.yourcompany.struts.ApplicationResources" />
</struts-config>
```

(10) 部署项目，启动服务，在浏览器地址栏中输入：<http://localhost:8080/Struts9.7.1/index.jsp>，出现如图 9-28 所示的窗口，在默认的情况下，使用的是中文版的资源文件，因为笔者电脑上的浏览器的语言种类默认为中文。如果使用英文的操作系统，则默认使用的是英文版的资源文件。



图 9-28 运行效果

这时单击 English 链接，则切换语言，运行效果如图 9-29 所示。



图 9-29 切换到英文 English 模式

## 9.8 添加 Struts 包的操作

在使用 MyEclipse 的过程中，导入以前的项目时，大多数的时候都会遇到找不到类的情况，如图 9-30 所示。

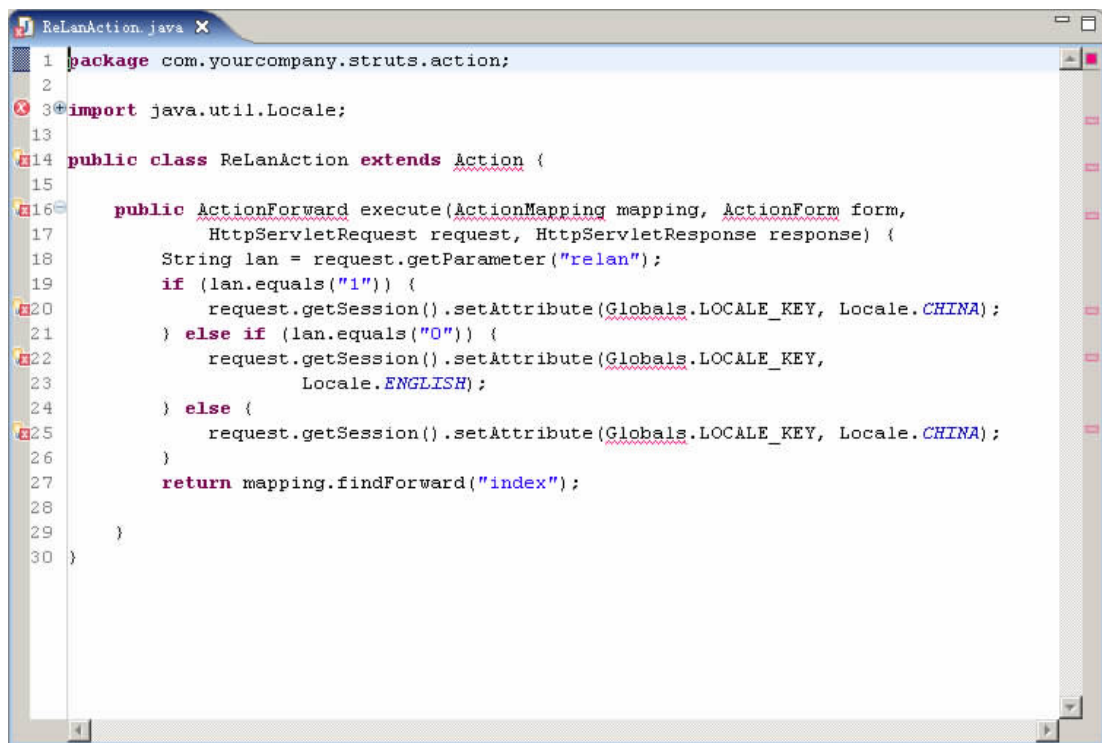


图 9-30 找不到类的错误提示

在这种情况下，在项目结构目录的 lib 目录中还可以看到 Struts 框架支持文件的存在，那是怎么回事呢？其实问题很简单，就是没有将 Struts 框架支持文件放到环境变量中去，所以出现上面的代码错误提示，找不到 Action 或 ActionForward 等的情况。

如何解决这样的问题呢？很简单！

用鼠标全选 lib 目录中的 jar 文件，如图 9-31 所示。

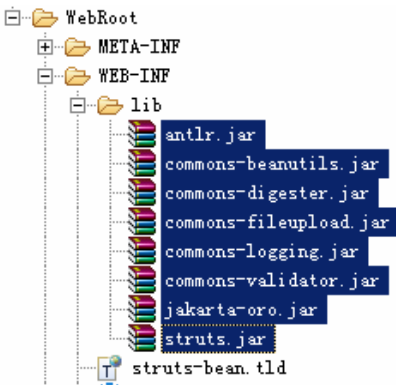


图 9-31 全选 jar 文件

全选后，在 jar 文件上面单击鼠标右键，在弹出的菜单中选择“ Build Path ” “ Add to Build Path ” 命令，如图 9-32 所示。

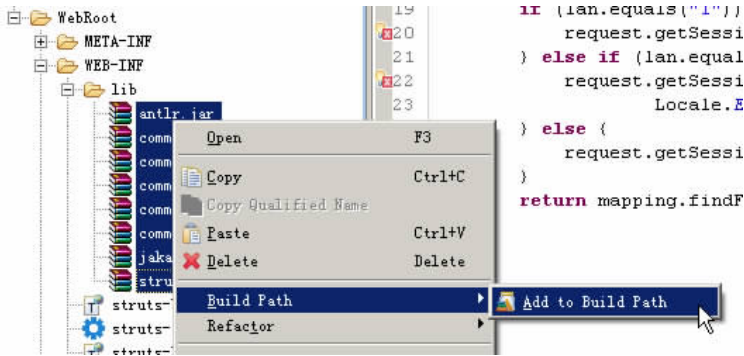


图 9-32 添加 Build Path 路径

原来以 win.rar 图标显示的 jar 文件变成如图 9-33 所示的图标。

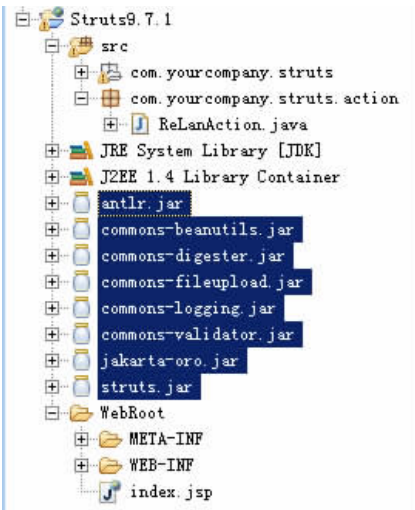


图 9-33 添加进 Build Path 路径的 jar 文件

这时再看 Action 类中的代码，没有错误提示了，能正确找到 Action 或 ActionForward 类了。

该办法可以使用在所有包含 jar 文件的项目中。

## 9.9 实现跨页表单的提交

在注册论坛账号的时候，由于注册的信息太多，以多个页面进行逐步的注册操作，这样的功能在 Struts 中也可以轻易实现，大概的原理就是一个 ActionForm 对应多个 Action，在 ActionForm 的验证方法中简单地检验表单数据的有效性。

实现跨页表单提交的实例

本例中需要注册的信息只有三项：姓名，密码，邮件。这三项需要在两个 JSP 页面中进行逐步注册，第一个页面注册用户名和密码，第二个页面注册邮件，最后一步也就是单击提交按钮进行数据库的持久化了。

(1) 在整个项目中，只有一个 ActionForm，所以在第一步就需要创建一个完整的 ActionForm，里面包括 name、password、email 域。

新建 ActionForm 如图 9-34 所示。

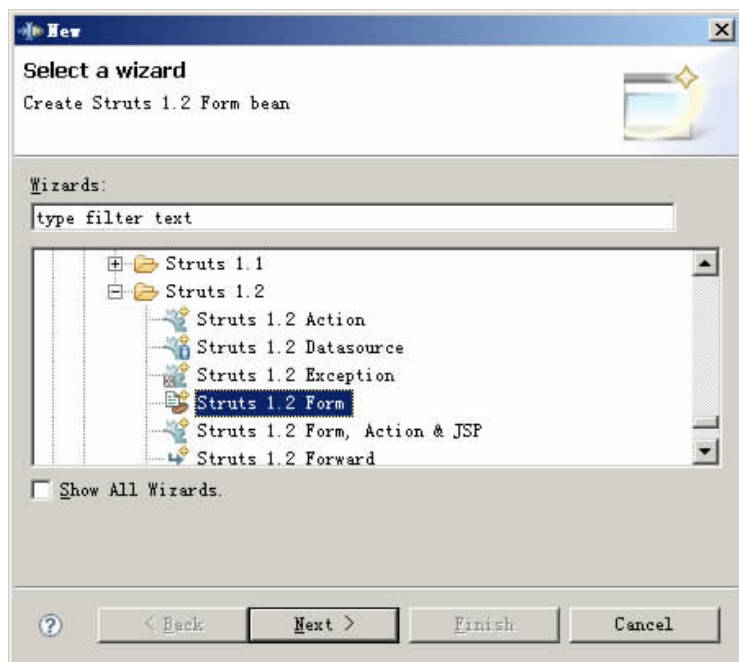


图 9-34 新建一个 ActionForm

单击“Next”按钮，开始配置这个 ActionForm。配置 ActionForm 如图 9-35 所示。

单击“Finish”按钮，完成创建 ActionForm 的操作。

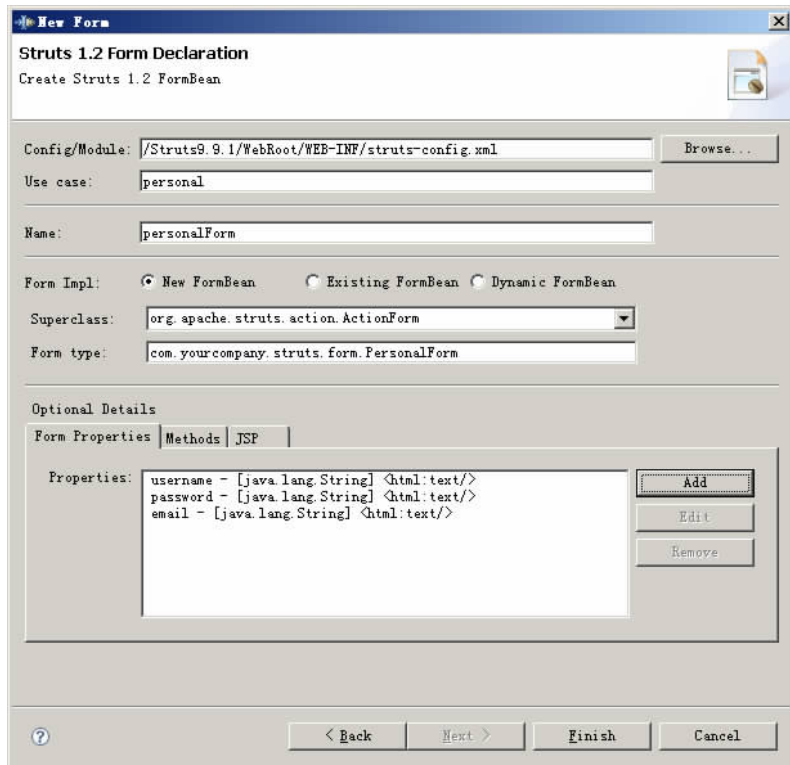


图 9-35 配置 ActionForm

(2) 由于跨页提交的表单是由多个 JSP 文件组合而成的, 在这一步需要创建注册用户名和密码的 JSP 文件, register1.jsp 代码如下。

```
<%@ page language="java" pageEncoding="ISO-8859-1"%>

<%@ taglib uri="http://struts.apache.org/tags-bean" prefix="bean"%>
<%@ taglib uri="http://struts.apache.org/tags-html" prefix="html"%>
<%@ taglib uri="http://struts.apache.org/tags-logic" prefix="logic"%>
<%@ taglib uri="http://struts.apache.org/tags-tiles" prefix="tiles"%>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html:html lang="true">
<body>
    <html:form action="/register1.do" method="post">
username:<html:text property="username" />
        <html:errors property="username" />
password:<html:text property="password" />
        <html:errors property="password" />
        <input type="hidden" name="page" value="1">
        <html:submit />
    </html:form>
</body>
</html:html>
```



(3) 在这一步需要创建注册邮件地址的 JSP 文件，register2.jsp 代码如下。

```
<%@ page language="java" pageEncoding="ISO-8859-1"%>

<%@ taglib uri="http://struts.apache.org/tags-bean" prefix="bean"%>
<%@ taglib uri="http://struts.apache.org/tags-html" prefix="html"%>
<%@ taglib uri="http://struts.apache.org/tags-logic" prefix="logic"%>
<%@ taglib uri="http://struts.apache.org/tags-tiles" prefix="tiles"%>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html:html lang="true">

<body>
    <html:form action="/register2.do" method="post">
email:<html:text property="email" />
        <html:errors property="email" />
        <input type="hidden" name="page" value="2">
        <html:submit />
    </html:form>
</body>
</html:html>
```

(4) 从上面两个 JSP 文件的 form 提交路径中可以看到，处理两个页面提交的 Action 分别为/register1 和/register2，在这一步就需要创建/register1 的 Action，如图 9-36 所示。

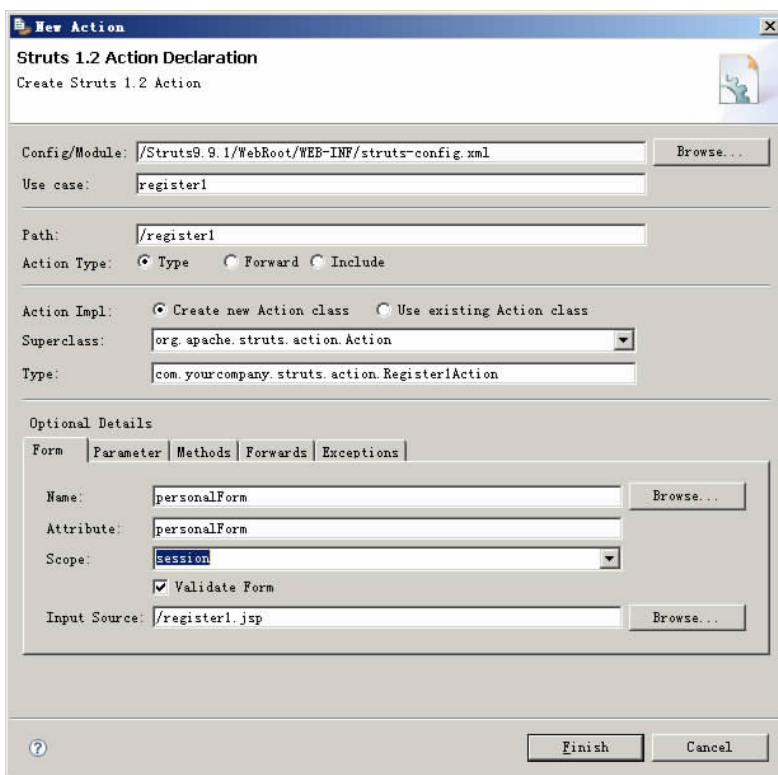


图 9-36 创建/register1 的 Action

(5) 在这一步需要创建/register2 的 Action，如图 9-37 所示。

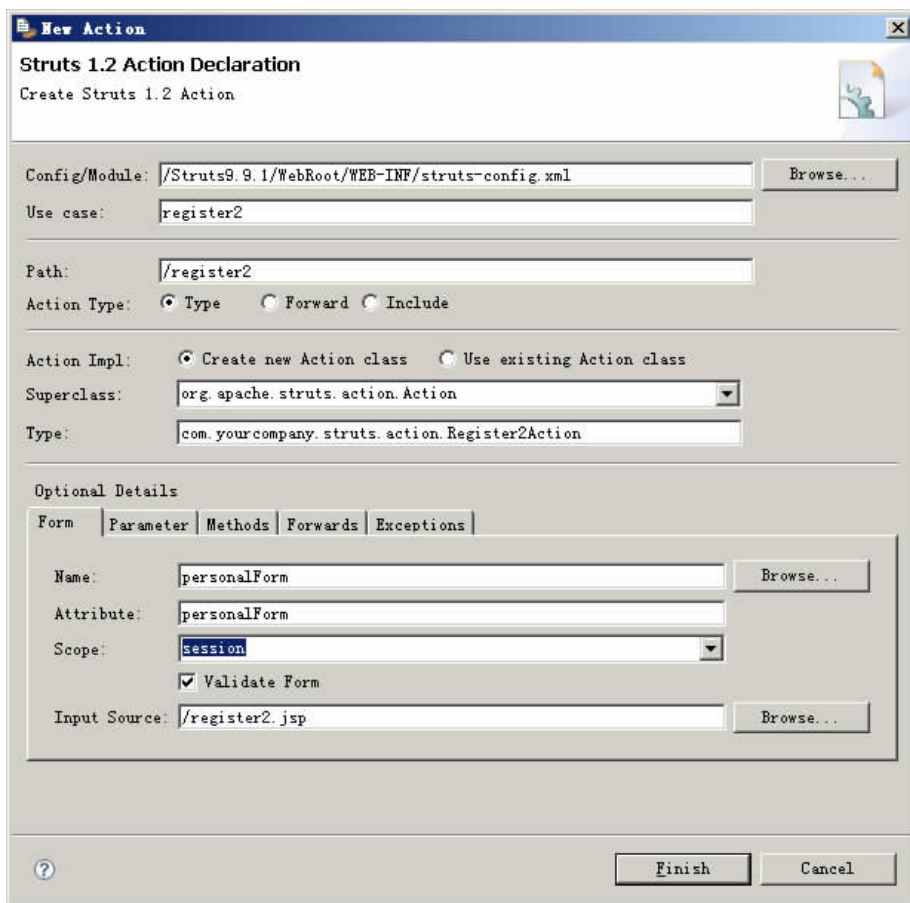


图 9-37 创建/register2 的 Action

(6) 创建完成 JSP 和 Action 后，需要在 ActionForm 的 validate 方法中写入针对表单域的验证代码，来防止用户在表单域中不输入内容而直接单击提交按钮，ActionForm 的完整代码如下。

```
package com.yourcompany.struts.form;

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 PersonalForm extends ActionForm {

    /** password property */
    private String password;
```

```
/** username property */
private String username;

/** email property */
private String email;

public ActionErrors validate(ActionMapping mapping,
    HttpServletRequest request) {
    ActionErrors error = new ActionErrors();
    if (request.getParameter("page").equals("1")) {
        if ("".equals(request.getParameter("username"))) {
            error.add("username", new ActionError("username null"));
        }
        if ("".equals(request.getParameter("password"))) {
            error.add("password", new ActionError("password null"));
        }
    }

    if (request.getParameter("page").equals("2")) {
        if ("".equals(request.getParameter("email"))) {
            error.add("email", new ActionError("email null"));
        }
    }
    return error;
}

public void reset(ActionMapping mapping, HttpServletRequest request) {
    // TODO Auto-generated method stub
}

public String getPassword() {
    return password;
}

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

public String getUsername() {
    return username;
}

public void setUsername(String username) {
    this.username = username;
}

public String getEmail() {
    return email;
}
```

```
public void setEmail(String email) {
    this.email = email;
}
}
```

在 `validate` 方法中通过使用隐藏域的值来取得是由哪个页面提交的处理请求，再根据这个值来判断用户名和密码或邮件的内容的合法性，如果有错误则返回 `ActionErrors` 的一个实例。

(7) 资源文件的内容如图 9-38 所示。

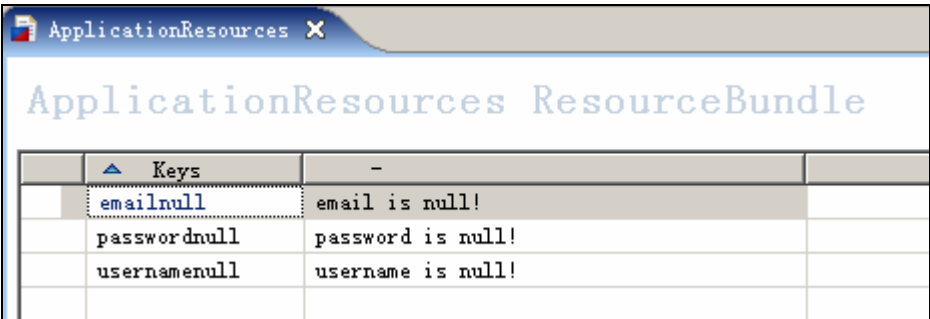


图 9-38 资源文件内容

(8) 如果用户名和密码的验证没有错误，则执行 `/register1.do` 的 Action，代码如下。

```
package com.yourcompany.struts.action;

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 com.yourcompany.struts.form.PersonalForm;

public class Register1Action extends Action {
    public ActionForward execute(ActionMapping mapping, ActionForm form,
        HttpServletRequest request, HttpServletResponse response) {
        PersonalForm personalForm = (PersonalForm) form;
        // TODO Auto-generated
        // method stub
        return mapping.findForward("step2");
    }
}
```

(9) 由第 (8) 步转到注册信息的第二个页面，显示注册邮件的内容，如果邮件输入通过验证，则只在 MyEclipse 的控制台中输出用户输入的个人信息，在这里可以使用数据库的持久化技术来保存用户的信息了，`/register2.do` 的 Action 代码如下。

```
package com.yourcompany.struts.action;

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 com.yourcompany.struts.form.PersonalForm;

public class Register2Action extends Action {

    public ActionForward execute(ActionMapping mapping, ActionForm form,
        HttpServletRequest request, HttpServletResponse response) {
        PersonalForm personalForm = (PersonalForm) form;
        // TODO Auto-generated

        // method stub
        System.out.print(personalForm.getUsername());
        System.out.print(personalForm.getPassword());
        System.out.print(personalForm.getEmail());
        return null;
    }
}
```

(10) struts-config.xml 配置文件内容如下。

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE struts-config PUBLIC "-//Apache Software Foundation
    //DTD Struts Configuration 1.2 EN"
    //"http://struts.apache.org/dtds/struts-config_1_2.dtd">

<struts-config>
    <data-sources />
    <form-beans>
        <form-bean name="personalForm"
            type="com.yourcompany.struts.form.PersonalForm" />
    </form-beans>

    <global-exceptions />
    <global-forwards>
        <forward name="step2" path="/register2.jsp" />
    </global-forwards>

    <action-mappings>
        <action attribute="personalForm" input="/register1.jsp"
            name="personalForm" path="/register1"
            type="com.yourcompany.struts.action.Register1Action" />
        <action attribute="personalForm" input="/register2.jsp"
            name="personalForm" path="/register2"
```

```
type="com.yourcompany.struts.action.Register2Action" />
</action-mappings>

<message-resources
    parameter="com.yourcompany.struts.ApplicationResources" />
</struts-config>
```

(11) 部署项目，启动服务。

在浏览器的地址栏输入 `http://localhost:8080/Struts9.9.1/register1.jsp`，出现如图 9-39 所示的界面。



图 9-39 注册第一步运行效果

如果在第一步中不输入任何内容而直接单击“Submit”提交按钮，则出现错误提示，如图 9-40 所示。



图 9-40 第一步错误提示

如果在“username”和“password”文本框输入内容后，单击“Submit”提交按钮，则转到注册 E-mail 的页面，如图 9-41 所示。



图 9-41 注册第二步运行效果

如果在注册第二步中也不输入任何的内容，则也会出现相应的错误提示，在“email”文本框中输入邮件地址后，单击“Submit”提交按钮，则由/register2.do 的 Action 处理，功能是在 Console 控制台上打印出注册的个人信息，如图 9-42 所示。

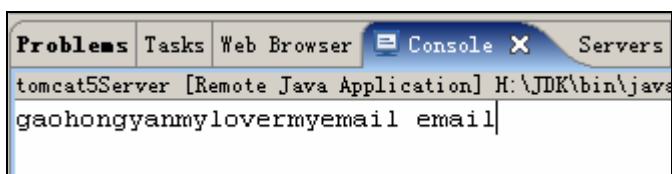


图 9-42 最后的结果

由于没有使用 println 方法，所以数据信息之间没有换行，但里面的数据的确是用户注册过的数据，并且也能正确地显示出来。

# 第 10 章

## 简易论坛模型的实例

本实例要求读者具有一定的 JSP 编程基础。

### 10.1 实例目标

- (1) 掌握 Struts 开发流程
- (2) 掌握 Action 的工作原理
- (3) 掌握 Struts-config.xml 文件的配置
- (4) 功能模块的完善及交互
- (5) 实现论坛模型的几个主要基本功能

### 10.2 功能模块简介

本节对本章实例的主要功能模块做一简单介绍。

#### 1. 用户注册

用户注册模块用于添加基本的用户信息。本实例有一个很典型的用户验证功能，在多数 Web 项目中，用户注册模块非常具有技术性。目前，在用户注册模块验证用户名是否重复时使用最多的技术是 Ajax，该技术基于无刷新页面的方式与数据库中的用户数据进行对比。

#### 2. 用户信息修改

用户信息修改模块的技术关键点在于 Struts 中的 HTML 类型的标签与数据库中的数据交互。如何将数据库中的数据完整不变地反应在基于 Struts 的 HTML 标签上？怎样实现对 HTML 标签动态地赋值？动态地存取数据？这些问题是在该模块中应该留意的知识点。

#### 3. 删除用户

删除用户模块仅仅是一个 DELETE 的操作，但其影响到相关的贴子及主题发表者与用户的关联性操作，如何做到删除用户时，与这个用户相关的贴子中的信息不被破坏，这也是软件设计上的一个小小经验。



#### 4. 主题编辑

主题编辑模块也是在开发留言板、论坛类项目时必做的一个模块，如果在发表主题时，一些程序代码里面有“<>”这样的符号，如何处理？怎样处理更安全？这也是本模块要关注的问题之一。

#### 5. 回复编辑

回复编辑其实和主题编辑的功能大体一样，它们的共同点就是基于用户权限使用“编辑”功能。只有管理员才可以编辑，只有管理员才会出现“编辑”超链接，这个权限的判断也是在代码中列为重点。

#### 6. 删除主题和删除回复

删除主题和删除回复也是一个值得留意之处，删除主题时要将回复一同删除，删除回复时要对回复与用户名之间的关联做一些善后的处理，请详细查看代码，体会设计上的解决方法。

#### 7. 权限管理

本实例的权限管理功能基本是一个很典型的实例，如果将本实例的权限管理与第 9.4 节进行联合设计，权限管理的功能会更加强大。当然，为了本实例代码流程的直观性，故本章没有加入此功能的实现。

#### 8. 用户登录

用户登录也是典型的权限认证和数据库操作的实例，代码的写法尽量不要出现 SQL 注入的情况，这也是登录模块要非常注意的特点。

#### 9. 查询用户资料

本模块具有一些 JDBC 与超级链接结合显示数据的基础实例，也是项目中最常用且必须掌握的知识点。而 JavaScript 语言在开发项目时也是非常重要的一个工具。

### 10.3 模块设计

本实例一共包括 10 个基本功能模块，通过这些模块的配置来完成一个实现论坛基本功能的软件项目实例。通过本实例，可以对 Struts 的理论知识进行实践，并将 Struts 开发流程进一步具体化，对以后开发大型的 Web 项目有非常好的推助作用。

程序运行效果如图 10-1 所示，页面布局使用 iframe 框架来实现，使用 Web 流行的左右分栏结构。

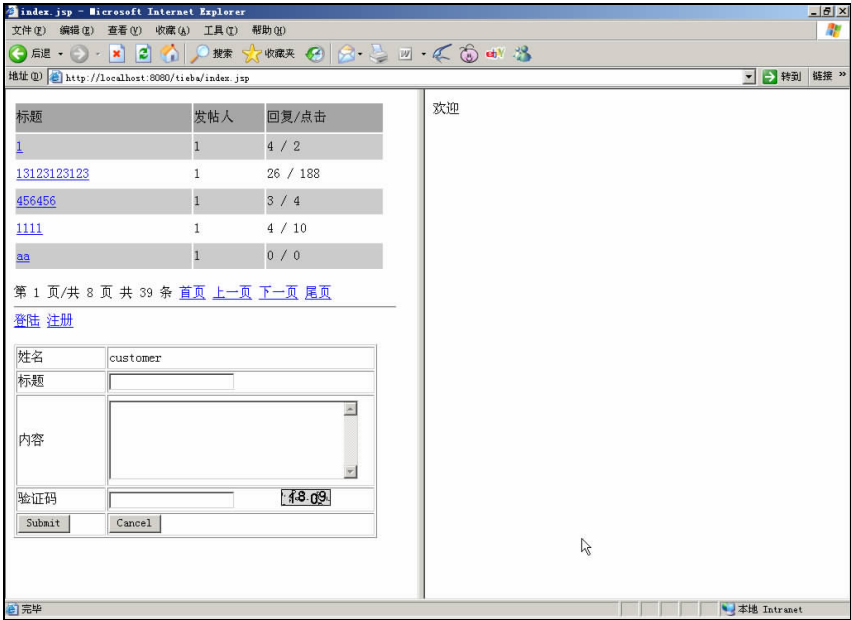


图 10-1 程序运行效果

在打开 `http://localhost:8080/tieba/index.jsp` 地址的时候，默认在右边的框架中出现欢迎页面，左边出现主题列表及分页、发表新主题的表单、验证码、提交按钮等常用功能。

### 10.3.1 用户注册

单击首页的“注册”超级链接，则出现注册用户的界面，在该界面中可以实现新用户的注册，填入必要的信息后单击“Submit”提交按钮，则在数据库中新添加一条用户信息的记录，如图 10-2 所示。



图 10-2 用户注册模块

注册页面的 JSP 源代码如下。

```
<%@ page language="java" pageEncoding="gb2312"%>
<%@ taglib uri=http://jakarta.apache.org/struts/tags-bean
    prefix="bean"%>
<%@ taglib uri=http://jakarta.apache.org/struts/tags-html
    prefix="html"%>

<html> <script type="text/javascript">
location.reload;
</script>
<head>
    <title>JSP for UsersForm form</title>
</head>
<body>
    <html:form action="/users.do">
    <table align="center">
    <tr>
        <td>姓名:</td>
        <td colspan="2"><html:text property="name"/>
            <html:errors property="name"/></td>
    </tr>
    <tr>
        <td>密码:</td>
        <td colspan="2"><html:password property="password"/>
            <html:errors property="password"/></td>
    </tr>
    <tr>
        <td>确认密码:</td>
        <td colspan="2"><html:password property="resetpw"/>
            <html:errors property="resetpw"/></td>
    </tr>
    <tr>
        <td>性别:</td>
        <td colspan="2"><html:radio property="sex" value="男">男
            </html:radio>
            <html:radio property="sex" value="女">女</html:radio></td>
    </tr>
    <tr>
        <td>年龄:</td>
        <td colspan="2"><html:text property="age"/>
            <html:errors property="age"/></td>
    </tr>
    <tr>
        <td>地址:</td>
        <td colspan="2"><html:text property="adds"/>
            <html:errors property="adds"/></td>
    </tr>
    <tr>
        <td>邮件:</td>
```



```

        ActionErrors errors = new ActionErrors();
        errors.add("name", new ActionError("namewrong1"));
        this.saveErrors(request, errors);
        return mapping.getInputForward();
    } else { //未注册
        int rs = ud.insertUsers(usersForm); //注册
        if (rs == 1) {
            System.out.println("4");
            hs.setAttribute("keys", "0");
            hs.setAttribute("dname", usersForm.getName());
            request.setAttribute("inf", "2");
            return mapping.findForward("tomessage"); // 正确
        } else {
            System.out.println("5");
            request.setAttribute("inf", "3");
            return mapping.findForward("tomessage"); // 出错
        }
    }
} else { System.out.println("2");
        ActionErrors errors = new ActionErrors();
        errors.add("yzm", new ActionError("yzmwrong"));
        this.saveErrors(request, errors);
        return mapping.getInputForward();
    }
}
}

```

在 Action 中首先使用语句 `if (usersForm.getYzm().equals(image))` 来判断验证码是否正确, 如果正确则继续进行注册过程。使用 UserDao 的实例 `ud` 判断用户是否注册, 如果注册过, 则返回错误信息, 如果没有注册使用 JDBC 代码操作数据库添加用户记录。

UserDao.java 的程序代码如下。

```

package com.zjy.struts.mypackage;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.ArrayList;
import java.util.Date;
import java.util.List;

import com.zjy.struts.form.UsersForm;

public class UserDao {
    public PreparedStatement ps = null;

    public Connection con = null;

```

```
public UserDao() {
    try {

        Class.forName("com.microsoft.jdbc.sqlserver.SQLServerDriver")
            .newInstance();
        String url = "jdbc:microsoft:sqlserver://localhost:1433;
            DatabaseName=luntan";
        String user = "sa";
        String password = "";
        con = DriverManager.getConnection(url, user, password);
    } catch (Exception e) {
        // TODO: handle exception
    }
}

public int insertUsers(UsersForm usersForm) {
    UsersForm uf = usersForm;

    int rs = 0;

    Date t = new Date();
    String ts = t.toLocaleString();

    try {
        ps = con.prepareStatement("insert into users(name,
            password, adds, sex, email, age, times)
            values(?,?,?,?,?,?,?)");
        ps.setString(1, bx(uf.getName()));
        ps.setString(2, uf.getPassword());
        ps.setString(3, bx(uf.getAdds()));
        ps.setString(4, bx(uf.getSex()));
        ps.setString(5, bx(uf.getEmail()));
        ps.setString(6, uf.getAge());
        ps.setString(7, ts);
        rs = ps.executeUpdate();

        System.out.print("insert right!");
    } catch (Exception e) {
        System.out.println(e);
        rs = 0;
        System.out.print("insert wrong!");
    } // 关闭Jdbc
    finally {
        try {
            ps.close();
            con.close();
        } catch (SQLException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
    }
}
```

```
    }  
}  
  
return rs;  
  
}  
  
public int selectName(String name) {  
    Statement stmt = null;  
    ResultSet rs = null;  
    int rst = 0;  
    try {  
  
        stmt = con.createStatement(ResultSet.TYPE_SCROLL_SENSITIVE,  
            ResultSet.CONCUR_READ_ONLY);  
        String sql = "select * from users where name='" + bx(name) + "'";  
        rs = stmt.executeQuery(sql);  
        if (rs.next()) {  
            rst = 1;  
            // System.out.print(rst);  
        }  
    } catch (Exception e) {  
        // TODO: handle exception  
        System.out.print(e);  
    } finally {  
        try {  
            rs.close();  
            stmt.close();  
            // con.close();  
        } catch (Exception e) {  
            // TODO: handle exception  
        }  
    }  
  
    return rst;  
}  
  
public UsersForm selectInf(String name) {  
    Statement stmt = null;  
    ResultSet rs = null;  
    UsersForm uf = new UsersForm();  
    try {  
  
        stmt = con.createStatement(ResultSet.TYPE_SCROLL_SENSITIVE,  
            ResultSet.CONCUR_READ_ONLY);  
        String sql = "select * from users where name='" + bx(name) + "'";  
        rs = stmt.executeQuery(sql);  
        if (rs.next()) {  
            uf.setId(rs.getInt(1));  
            uf.setName(bx(name));  
        }  
    }  
}
```

```
        uf.setAdds(rs.getString(4));
        uf.setSex(rs.getString(5));
        uf.setEmail(rs.getString(6));
        uf.setAge(rs.getString(7));
        // System.out.print(rst);
    }
} catch (Exception e) {
    // TODO: handle exception
    System.out.print(e);
} finally {
    try {
        rs.close();
        stmt.close();
        // con.close();
    } catch (Exception e) {
        // TODO: handle exception
    }
}

return uf;
}

public int updateUser(UsersForm usersForm) {
    UsersForm uf = usersForm;
    Statement stmt = null;
    int rst = 0;
    try {
        stmt = con.createStatement();
        String sql = "update users set password='" + uf.getPassword()
            + "',adds='" + uf.getAdds() + "',sex='" +
            bx(uf.getSex()) + "',email='" + uf.getEmail() + "',
            age='" + uf.getAge() + "' where name='" +
            uf.getName() + "'";
        rst=stmt.executeUpdate(sql);
        System.out.print("insert right!");
    } catch (Exception e) {
        System.out.println(e);
        rst = 0;
        System.out.print("insert wrong!");
    }// 关闭Jdbc
    finally {
        try {

            con.close();
        } catch (SQLException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
    }
}
```



```
        return rst;

    }

    public List selectAll () {
        List list = new ArrayList();
        Statement stmt = null;
        ResultSet rs = null;

        try {
            stmt = con.createStatement(ResultSet.TYPE_SCROLL_SENSITIVE,
                                       ResultSet.CONCUR_READ_ONLY);
            String sql = "select * from users";
            rs = stmt.executeQuery(sql);
            while(rs.next()) {
                UsersForm uf = new UsersForm();
                uf.setId(rs.getInt(1));
                uf.setName(rs.getString(2));
                uf.setAdds(rs.getString(3));
                uf.setSex(rs.getString(4));
                uf.setEmail(rs.getString(5));
                uf.setAge(rs.getString(6));
                list.add(uf);
            }
        } catch (SQLException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }

        return list;
    }

    public int deleteUser(String ids) {
        Statement stmt = null;
        int rst = 0;
        try {
            int id = Integer.parseInt(ids);
            stmt = con.createStatement();
            String sql = "delete from users where id=" + id;
            rst = stmt.executeUpdate(sql);
        } catch (SQLException e) {
            // TODO Auto-generated catch block
            rst = 0;
            e.printStackTrace();
        } finally {
            try {
                stmt.close();
            } catch (SQLException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }
        }

        return rst;
    }
}
```

```
}  
public String bx(String s) {  
    try {  
        byte b[] = s.getBytes("iso-8859-1");  
        s = new String(b);  
    } catch (Exception e) {  
    }  
    return s;  
}  
}
```

UserDao.java 程序代码在以后的项目中经常使用,请留意该 Bean 的技术及业务的实现。

在注册页面输入用户的必要信息后,单击“Submit”提交按钮,进行数据的提交,如果验证码及数据库验证通过,则出现如图 10-3 所示的界面。



图 10-3 注册成功页面

单击超级链接“返回主页”则用户注册成功,返回主页。

### 10.3.2 显示主题列表

单击“返回主页”后,出现主题列表的模块,在这里使用左右分栏的框架来实现,如图 10-4 所示。

index.jsp 的程序代码如下。

```
<%@ page language="java" pageEncoding="gb2312"%>  
  
<%@ taglib uri="http://struts.apache.org/tags-bean" prefix="bean" %>  
<%@ taglib uri="http://struts.apache.org/tags-html" prefix="html" %>  
<%@ taglib uri="http://struts.apache.org/tags-logic" prefix="logic" %>  
<%@ taglib uri="http://struts.apache.org/tags-tiles" prefix="tiles" %>  
  
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">  
<html:html lang="true">  
    <head>  
        <html:base />  
  
        <title>index.jsp</title>
```

```

<meta http-equiv="pragma" content="no-cache">
<meta http-equiv="cache-control" content="no-cache">
<meta http-equiv="expires" content="0">
<meta http-equiv="keywords" content="keyword1,keyword2,keyword3">
<meta http-equiv="description" content="This is my page">
<!--
<link rel="stylesheet" type="text/css" href="styles.css">
-->

</head>
<frameset rows="*" cols="500,*" framespacing="5" frameborder="yes"
        border="5">
    <frame src="show.do?method=showzhuti" name="leftFrame" >
    <frame src="show.do?method=showtie" name="mainFrame">
</frameset><noframes></noframes>
<noframes>
    <body>

        </body>
</html>

```

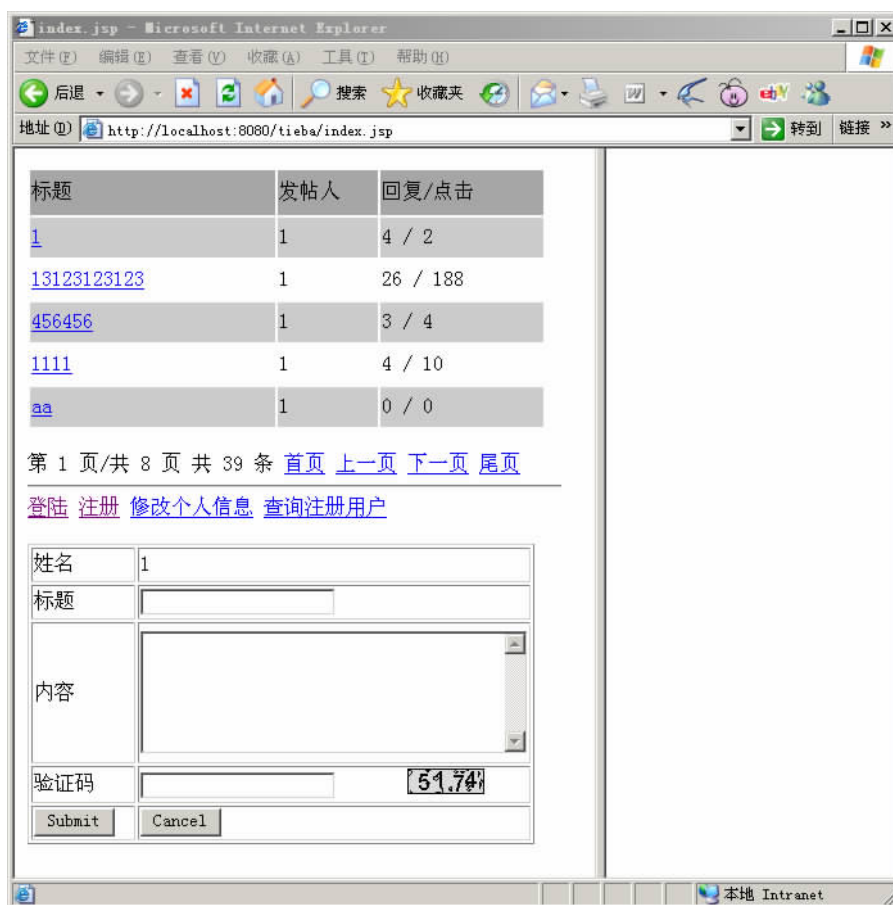


图 10-4 主题列表



```

        target="mainFrame">
        <bean:write name="ztlist"
            property="zhutitle" />
        </html:link>
    </td>
    <td>
        <bean:write name="ztlist" property="zhuname" />
    </td>
    <td>
        <bean:write name="ztlist" property="answer" />
        /
        <bean:write name="ztlist" property="chicks" />
    </td>
</tr>
</logic:iterate>
</table>

<br>

        第
        <bean:write name="zhutify" property="showpage" />
        页/共
        <bean:write name="zhutify" property="allpage" />
        页 共
        <bean:write name="zhutify" property="alllist" />
        条
        <html:link action="show.do?method=showzhuti" paramId="showpage"
            paramName="zhutify" paramProperty="fristpage">首页
        </html:link>
        <html:link action="show.do?method=showzhuti" paramId="showpage"
            paramName="zhutify" paramProperty="previouspage">上一页
        </html:link>
        <html:link action="show.do?method=showzhuti" paramId="showpage"
            paramName="zhutify" paramProperty="nextpage">下一页
        </html:link>
        <html:link action="show.do?method=showzhuti" paramId="showpage"
            paramName="zhutify" paramProperty="lastpage">尾页</html:link>
        <hr>
        <html:link forward="todenglu" target="_parent">登录</html:link>
        <html:link forward="tozhu" target="_parent">注册</html:link>
        <logic:notEqual name="dname" value="customer">
            <html:link action="show.do?method=showuser" target="_parent">
                修改个人信息</html:link>
        </logic:notEqual>
        <logic:equal name="keys" value="1">
            <html:link forward="tousers" target="mainFrame"> 查询注册用户
        </html:link>
        </logic:equal>
        <p>
            <logic:equal name="keys" value="1">
                </logic:equal>

```

```
<logic:equal name="keys" value="1"></logic:equal>
</p>
<html:form action="/zhutie.do">
  <table width="95%" height="228" border="1">
    <tr>
      <td width="25%" height="27">
        姓名
      </td>
      <td width="75%">
        <bean:write name="dname" />
      </td>
    </tr>
    <tr>
      <td height="26">
        标题
      </td>
      <td>
        <html:text property="title" />
        <html:errors property="title" />
      </td>
    </tr>
    <tr>
      <td height="110">
        内容
      </td>
      <td>
        <html:textarea property="text"
          cols="40" rows="6" />
        <html:errors property="text" />
      </td>
    </tr>
    <tr>
      <td height="24">
        验证码
      </td>
      <td>
        <html:text property="yzm" />
        <html:errors property="yzm" />
        &nbsp; &nbsp; &nbsp; &nbsp; &nbsp;
        <html:img page="/form/image.jsp" />
      </td>
    </tr>
    <tr>
      <td height="27">
        <html:submit />
      </td>
      <td>
        <html:cancel />
      </td>
    </tr>
  </table>
</html:form>
```

```

        </tr>
    </table>
</html:form>
</body>
</html>

```

在生成主题链接的 JSP 文件中，关键的程序代码如下：

```

<html:link paramId="id" paramName="ztlist" paramProperty="id"
    action="show.do?method=showtie" target="mainFrame">
    <bean:write name="ztlist" property="zhutitle"/>
</html:link>

```

通过使用<html:link>标签的target属性设置目标页打开的位置，这里是mainFrame框架，也就是框架右边的分栏的位置，并且使用paramId、paramName和paramProperty属性来拼装超级链接的参数名称及参数的取值。最后使用<bean:write>标签来显示出主题的文本内容，也是作为超级链接的文本。

在主题链接中使用JSTL标签来实现隔行不同颜色的显示功能。

huitie.jsp是框架右边的链接页面，代码如下。

```

<%@ page language="java" pageEncoding="gb2312"%>
<%@ taglib uri="http://struts.apache.org/tags-bean" prefix="bean"%>
<%@ taglib uri="http://struts.apache.org/tags-html" prefix="html"%>
<%@ taglib uri="http://struts.apache.org/tags-logic" prefix="logic"%>
<%@ taglib uri="http://struts.apache.org/tags-tiles" prefix="tiles"%>

<html>
    <script type="text/javascript">
location.reload;
</script>
    <head>
        <title>JSP for HuitieForm form</title>
        <meta http-equiv="Content-Type" content="text/html;
                                charset=gb2312"></head>
    <body>
        <table width="95%" height="170" border="1">
            <tr bgcolor="#A7A7A7">
                <td width="162" height="30">
                    楼主
                </td>
                <td width="420">
                    标题：
                    <bean:write name="zf" property="title" /><logic:equal
                        name="keys" value="1">
                        <html:link action="delete.do?method=deleteAll"
                            paramId="id"
                            paramName="zf" paramProperty="id"
                            target="_parent">删除</html:link>
                    </logic:equal>
                </td>
            </tr>
        </table>
    </body>
</html>

```

```

        </tr>
        <tr bgcolor="#CCCCCC">
            <td height="30">
                姓名 : <html:link action="show.do?method=showAllusers"
                    paramId="name"
                    paramName="zf" paramProperty="name"><bean:write name="zf"
                        property="name" /></html:link>

            </td>
            <td>
                时间 :
                <bean:write name="zf" property="times" />
            </td>
        </tr>
        <tr>
            <td height="100">
                内容 :
            </td>
            <td>
                <bean:write name="zf" property="text"
                    filter="false" />
            </td>
        </tr>
    </table>

    <hr>
    <table width="95%" height="170" border="1">
        <logic:iterate id="hl" name="hlist">

            <tr bgcolor="#A7A7A7">
                <td width="162" height="30">
                    第
                    <bean:write name="hl" property="id" />
                    楼
                </td>
                <td width="420">
                    回复<logic:equal name="keys" value="1">
                        <html:link action="delete.do?
                            method=deleteHuitie" paramId="id"
                            paramName="hl" paramProperty="parentid">
                            删除</html:link>
                    </logic:equal>
                </td>
            </tr>
            <tr bgcolor="#CCCCCC">
                <td height="30">
                    姓名 : <html:link action="show.do?
                        method=showAllusers" paramId="name"
                        paramName="hl" paramProperty="name"><bean:write name="hl"
                            property="name" /></html:link>

```



```

        </td>
        <td>
            时间 :
            <bean:write name="hl" property="times" />
        </td>
    </tr>
    <tr>
        <td height="100">
            内容 :
        </td>
        <td>
            <bean:write name="hl" property="text"
                filter="false" />
        </td>
    </tr>
</logic:iterate>
</table>
<hr>
第
<bean:write name="huitiefy" property="showpage" />
页/共
<bean:write name="huitiefy" property="allpage" />
页 共
<bean:write name="huitiefy" property="alllist" />
条
<html:link action="show.do?method=showhuitie" paramId="showpage"
    paramName="huitiefy" paramProperty="fristpage">首页
</html:link>
<html:link action="show.do?method=showhuitie" paramId="showpage"
    paramName="huitiefy" paramProperty="previouspage">上一页
</html:link>
<html:link action="show.do?method=showhuitie" paramId="showpage"
    paramName="huitiefy" paramProperty="nextpage">下一页
</html:link>
<html:link action="show.do?method=showhuitie" paramId="showpage"
    paramName="huitiefy" paramProperty="lastpage">尾页
</html:link>

<hr>

<html:form action="/huitie.do">
    <table width="95%" height="228" border="1">
        <tr>
            <td width="25%" height="27">
                姓名
            </td>
            <td width="75%">
                <bean:write name="dname" />
            </td>
        </tr>
    </table>
</html:form>

```

[illegible]

```
import org.apache.struts.action.ActionForm;
import org.apache.struts.action.ActionForward;
import org.apache.struts.action.ActionMapping;
import org.apache.struts.actions.DispatchAction;

import com.zjy.struts.form.UsersForm;
import com.zjy.struts.form.ZhutieForm;
import com.zjy.struts.mybean.Fenye;
import com.zjy.struts.mypackage.FenyeDao;
import com.zjy.struts.mypackage.HuitieDao;
import com.zjy.struts.mypackage.UserDao;
import com.zjy.struts.mypackage.ZhutiDao;
import com.zjy.struts.mypackage.ZhutieDao;

public class ShowAction extends DispatchAction {

    public ActionForward showtie(ActionMapping mapping, ActionForm form,
        HttpServletRequest request, HttpServletResponse response) {
        // TODO Auto-generated method stub

        String sid = request.getParameter("id");// 取主贴 id
        if (sid == null || "".equals(sid)) {
            return mapping.findForward("tomessage");
        } else {
            int id = Integer.parseInt(sid);
            ZhutieDao zd = new ZhutieDao();
            ZhutieForm zf = zd.setZhutie(id);// 取主贴内容
            FenyeDao fd = new FenyeDao();
            Fenye hfy = fd.selectHuitie("1", id);
            HuitieDao hd = new HuitieDao();
            List hlist = hd.setHuitie(hfy);// 取回贴内容
            HttpSession hs = request.getSession();
            hs.setAttribute("zf", zf);
            hs.setAttribute("huitiefy", hfy);
            hs.setAttribute("hlist", hlist);
            return mapping.findForward("tohuitie");
        }
    }

    public ActionForward showhuitie(ActionMapping mapping, ActionForm form,
        HttpServletRequest request, HttpServletResponse response) {
        // TODO Auto-generated method stub

        String showpage = request.getParameter("showpage");
        if (showpage == null || "".equals(showpage)) {
            showpage = "1";
        }
        HttpSession hs = request.getSession();
        Fenye hfyl = (Fenye) hs.getAttribute("huitiefy");
```

```
FenyeDao fd = new FenyeDao();
Fenye hfy = fd.selectHuitie(showpage, hfyl.getId());
HuitieDao hd = new HuitieDao();
List hlist = hd.setHuitie(hfy); // 取回贴内容
hs.setAttribute("huitiefy", hfy);
hs.setAttribute("hlist", hlist);
return mapping.findForward("tohuitie");
}

public ActionForward showzhuti(ActionMapping mapping, ActionForm form,
    HttpServletRequest request, HttpServletResponse response) {

    String sp = request.getParameter("showpage");
    if (sp == null || "".equals(sp)) {
        sp = "1";
    }
    FenyeDao fd = new FenyeDao();
    Fenye zfy = fd.selectZhutie(sp);
    ZhutiDao zd = new ZhutiDao();
    List list = zd.selectZhuti(zfy); // 取主题列表内容
    HttpSession hs = request.getSession();
    String dname = (String) hs.getAttribute("dname");
    if (dname == null || "".equals(dname)) {
        dname = "customer";
    }
    hs.setAttribute("zhutify", zfy);
    hs.setAttribute("zhutilist", list);
    hs.setAttribute("dname", dname);
    return mapping.findForward("tozhutie");
}

public ActionForward showuser(ActionMapping mapping, ActionForm form,
    HttpServletRequest request, HttpServletResponse response) {
    HttpSession hs = request.getSession();
    String dname = (String) hs.getAttribute("dname");
    UserDao ud = new UserDao();
    UsersForm ufi = ud.selectInf(dname); // 取用户信息
    hs.setAttribute("ufi", ufi);
    return mapping.findForward("toxiugai");
}

public ActionForward showAllusers(ActionMapping mapping,
    ActionForm form, HttpServletRequest request,
    HttpServletResponse response) {
    String name = request.getParameter("name");
    UserDao ud = new UserDao();
    UsersForm uf = ud.selectInf(name); // 取用户信息
    if (uf.getId() == 0) {
        request.setAttribute("u", "null");
        request.setAttribute("aul", null);
    }
}
```

```

    } else {
        request.setAttribute("aul", uf);
    }
    return mapping.findForward("tousers");
}
}

```

### 10.3.3 用户登录

单击主界面的登录超级链接，出现如图 10-5 所示的界面。



图 10-5 用户登录

denglu.jsp 程序代码如下。

```

<%@ page language="java" pageEncoding="gb2312"%>
<%@ taglib uri=http://jakarta.apache.org/struts/tags-bean
    prefix="bean"%>
<%@ taglib uri=http://jakarta.apache.org/struts/tags-html
    prefix="html"%>

<html>
<script type="text/javascript">
location.reload;
</script>
<head>
    <title>JSP for DengluForm form</title>
</head>
<body>
<html:messages id="msg" message="true">
<bean:write name="msg"/>
</html:messages>
    <html:form action="/denglu.do">
        <table border="1" align="center">
            <tr>
                <td>姓名:</td>

```

与 denglu.jsp 相对应的 Action 程序代码如下。

```
package com.zjy.struts.action;

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.ActionError;
import org.apache.struts.action.ActionErrors;
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 com.zjy.struts.form.DengluForm;
import com.zjy.struts.mypackage.DengluDao;

public class DengluAction extends Action {
    public ActionForward execute(ActionMapping mapping, ActionForm form,
        HttpServletRequest request, HttpServletResponse response) {
        DengluForm dengluForm = (DengluForm) form;
        // TODO Auto-generated
        HttpSession hs = request.getSession();
        String image = (String) hs.getAttribute("rand");
        if (dengluForm.getYzm().equals(image)) { // 判断密码
            DengluDao dd = new DengluDao();
```

```

int rst = dd.denglu(dengluForm);
if (rst == 0) { // 用户不存在
    ActionErrors errors = new ActionErrors();
    errors.add(ActionMessages.GLOBAL_MESSAGE,
        new ActionMessage("dengluwrong"));
    this.addMessages(request, errors);
    return mapping.getInputForward();
} else if (rst == 1) { // 普通用户
    hs.setAttribute("dname", dengluForm.getUsername());
    request.setAttribute("inf", "1");
    hs.setAttribute("keys", "0"); // 普通用户标记
    return mapping.findForward("tomessage");
} else { // 管理员
    hs.setAttribute("dname", dengluForm.getUsername());
    hs.setAttribute("keys", "1"); // 管理员标记
    request.setAttribute("inf", "0");
    return mapping.findForward("tomessage");
}
} else { // 验证码出错
    System.out.println("2");
    ActionErrors errors = new ActionErrors();
    errors.add("yzm", new ActionError("yzmwrong"));
    this.saveErrors(request, errors);
    return mapping.getInputForward();
}
}
}

```

输入正确的用户名和密码后，出现如图 10-6 所示的界面。



图 10-6 登录成功

#### 10.3.4 修改个人用户信息

登录成功后，可以单击“修改个人信息”超级链接来更改用户的个人资料信息，如图 10-7 所示。

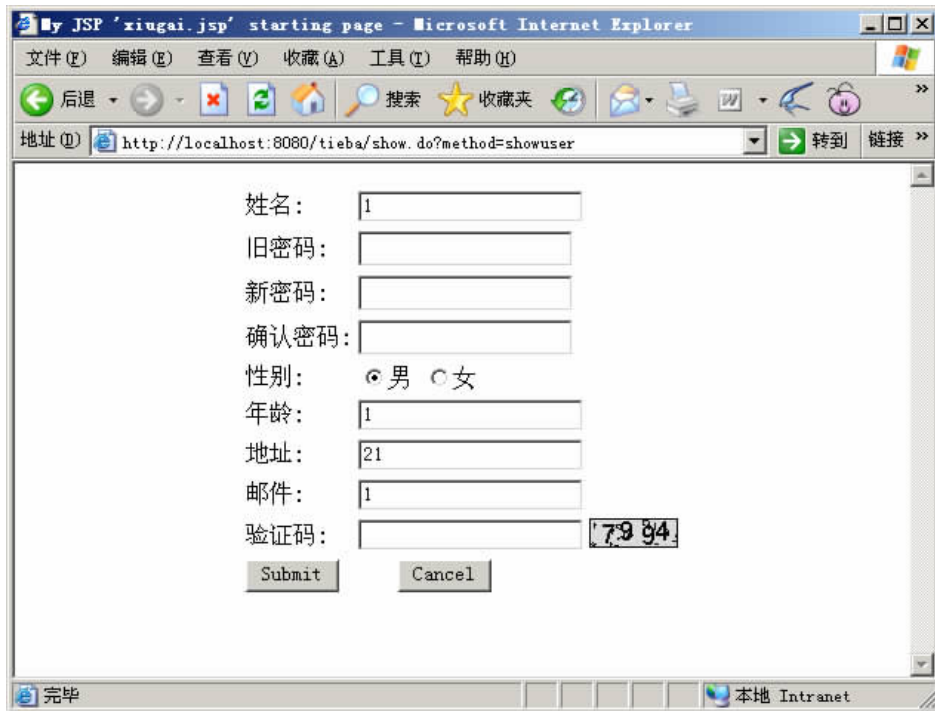


图 10-7 修改用户信息

更改完成后，单击“Submit”提交按钮进行数据的更改保存。  
其相对应的 xiugai.jsp 文件程序代码如下。

```
<%@ page language="java" import="java.util.*" pageEncoding="gb2312"%>

<%@ taglib uri="http://struts.apache.org/tags-bean" prefix="bean"%>
<%@ taglib uri="http://struts.apache.org/tags-html" prefix="html"%>
<%@ taglib uri="http://struts.apache.org/tags-logic" prefix="logic"%>
<%@ taglib uri="http://struts.apache.org/tags-tiles" prefix="tiles"%>
<%
    String path = request.getContextPath();
    String basePath = request.getScheme() + "://"
        + request.getServerName() + ":" + request.getServerPort()
        + path + "/";
%>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
    <script type="text/javascript">
location.reload;
</script>
    <head>
        <base href="<%=basePath%>">

        <title>My JSP 'xiugai.jsp' starting page</title>
```



```
<meta http-equiv="pragma" content="no-cache">
<meta http-equiv="cache-control" content="no-cache">
<meta http-equiv="expires" content="0">
<meta http-equiv="keywords" content="keyword1,keyword2,keyword3">
<meta http-equiv="description" content="This is my page">
<!--
<link rel="stylesheet" type="text/css" href="styles.css">
-->

</head>

<body>
  <logic:notEqual name="dname" value="customer">

    <html:form action="/xiugai.do">
      <table align="center">
        <tr>
          <td>
            姓名:
          </td>
          <td colspan="2">
            <bean:define name="dname" id="name" />

            <html:text property="name"
              value="<%=name.toString()%>" />
          </td>
        </tr>
        <tr>
          <td>
            旧密码:
          </td>
          <td colspan="2">
            <html:password property="oldpw" />
            <html:errors property="oldpw" />
          </td>
        </tr>
        <tr>
          <td>
            新密码:
          </td>
          <td colspan="2">
            <html:password property="password" />
            <html:errors property="password" />
          </td>
        </tr>
        <tr>
          <td>
            确认密码:
          </td>
```

```
<td colspan="2">
    <html:password property="resetpw" />
    <html:errors property="resetpw" />
</td>
</tr>
<tr>
    <td>
        性别:
    </td>
    <td colspan="2">
        <bean:define id="sex" name="ufi"
            property="sex" />
        <html:radio property="sex" value="男">男
        </html:radio>
        <html:radio property="sex" value="女">女
        </html:radio>
    </td>
</tr>
<tr>
    <td>
        年龄:
    </td>
    <td colspan="2">
        <bean:define id="age" name="ufi"
            property="age" />
        <html:text property="age"
            value="<%=age.toString()%>" />
        <html:errors property="age" />
    </td>
</tr>
<tr>
    <td>
        地址:
    </td>
    <td colspan="2">
        <bean:define id="adds" name="ufi"
            property="adds" />
        <html:text property="adds"
            value="<%=adds.toString()%>" />
        <html:errors property="adds" />
    </td>
</tr>
<tr>
    <td>
        邮件:
    </td>
    <td colspan="2">
        <bean:define id="email" name="ufi"
            property="email" />
        <html:text property="email"
```



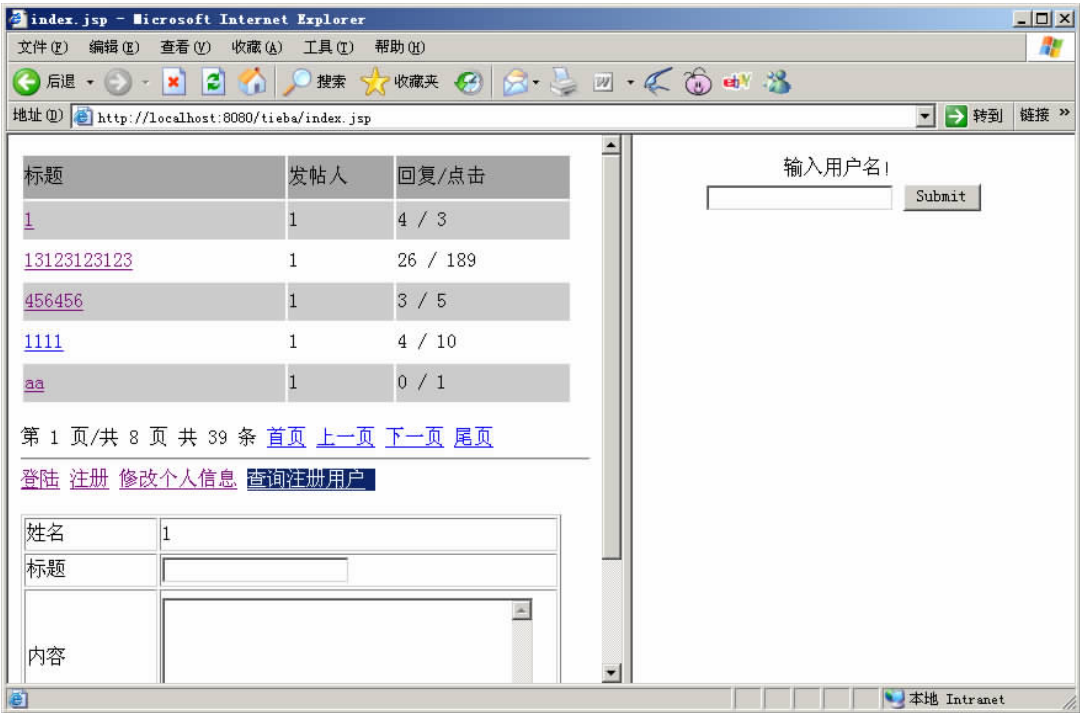


图 10-8 用户注册查询

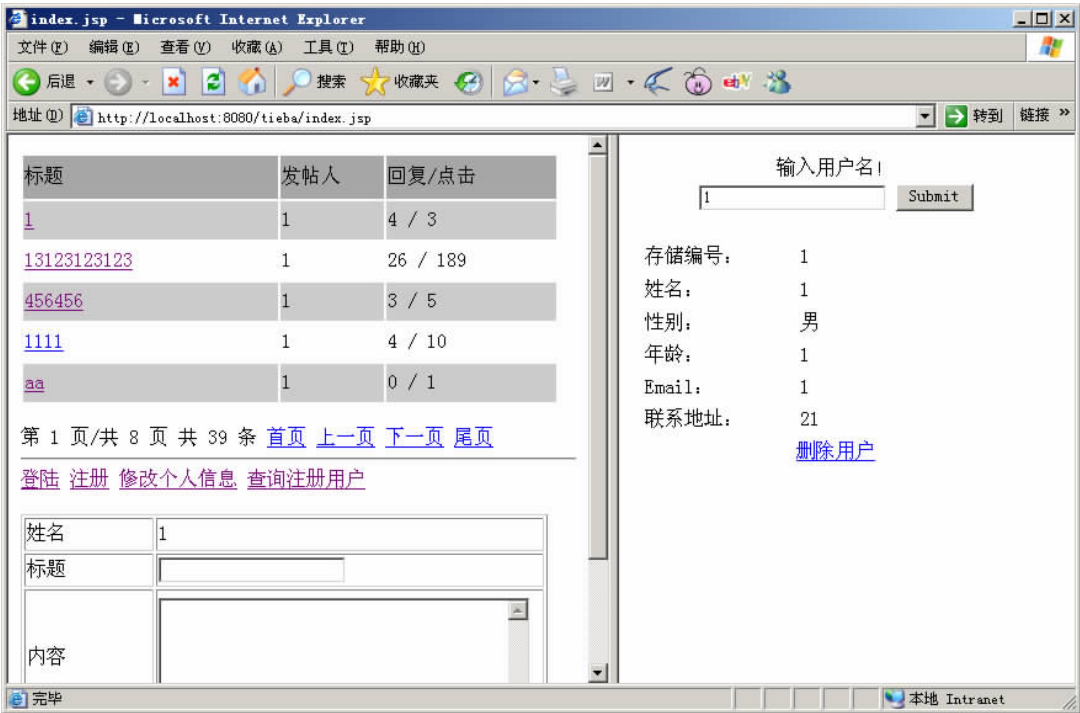


图 10-9 查询结果

查询用户名的 users.jsp 程序代码如下。

```
<%@ page language="java" pageEncoding="gb2312"%>

<%@ taglib uri="http://struts.apache.org/tags-bean" prefix="bean"%>
<%@ taglib uri="http://struts.apache.org/tags-html" prefix="html"%>
<%@ taglib uri="http://struts.apache.org/tags-logic" prefix="logic"%>
<%@ taglib uri="http://struts.apache.org/tags-tiles" prefix="tiles"%>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html:html lang="true">
<head>
    <html:base />

    <title>users.jsp</title>

    <meta http-equiv="pragma" content="no-cache">
    <meta http-equiv="cache-control" content="no-cache">
    <meta http-equiv="expires" content="0">
    <meta http-equiv="keywords" content="keyword1,keyword2,keyword3">
    <meta http-equiv="description" content="This is my page">
    <!--
    <link rel="stylesheet" type="text/css" href="styles.css">
    -->

</head>

<body>
    <logic:notPresent name="keys">
        <table width="95%" align="center">
            <tr>
                <td align="center">
                    你无此权限！
                </td>
            </tr>
            <tr>
                <td align="center">
                    <html:link forward="toindex">返回主页</html:link>
                </td>
            </tr>
        </table>
    </logic:notPresent>

    <logic:present name="keys">
        <logic:equal name="keys" value="1">
            <html:form action="/charUser.do">
                <table width="95%" height="34" align="center">
                    <tr>
                        <td align="center">
                            输入用户名！
                        </td>
                    </tr>
                </table>
            </html:form>
        </logic:equal>
    </logic:present>
</body>
</html>
```

```
<tr>
    <td align="center">
        <html:text property="name" />
        <html:submit />
    </td>
</tr>
<logic:equal name="u" value="null">
<tr>
    <td align="center">
        所查用户不存在！
    </td>
</tr>
</logic:equal>
</table>
</html:form>
<logic:present name="aul">
    <table width="95%" height="186" align="center">
        <tr>
            <td width="145">
                存储编号：
            </td>
            <td width="242">
                <bean:write name="aul" property="id" />
            </td>
        </tr>
        <tr>
            <td>
                姓名：
            </td>
            <td>
                <bean:write name="aul" property="name" />
            </td>
        </tr>
        <tr>
            <td>
                性别：
            </td>
            <td>
                <bean:write name="aul" property="sex" />
            </td>
        </tr>
        <tr>
            <td>
                年龄：
            </td>
            <td>
                <bean:write name="aul" property="age" />
            </td>
        </tr>
    </table>
</logic:present>
</table>
```

```

        <td>
            Email :
        </td>
        <td>
            <bean:write name="aul" property="email" />
        </td>
    </tr>
    <tr>
        <td>
            联系地址 :
        </td>
        <td>
            <bean:write name="aul" property="adds" />
        </td>
    </tr>
    <tr align="center">
        <td colspan="2">
            <html:link action="delete.do?
                method=deleteUser" paramId="id"
                paramName="aul" paramProperty="id">
                删除用户</html:link>
        </td>
    </tr>
    </table>
    </logic:present>
</logic:equal>

<logic:equal name="keys" value="0">
    <logic:present name="aul">
        <table width="95%" height="151" align="center">
            <tr>
                <td width="148">
                    姓名 :
                </td>
                <td width="210">
                    <bean:write name="aul" property="name" />
                </td>
            </tr>
            <tr>
                <td>
                    性别 :
                </td>
                <td>
                    <bean:write name="aul" property="sex" />
                </td>
            </tr>
            <tr>
                <td>
                    年龄 :
                </td>

```

```
<td>
    <bean:write name="aul" property="age" />
</td>
</tr>
<tr>
    <td>
        Email :
    </td>
    <td>
        <bean:write name="aul" property="email" />
    </td>
</tr>
<tr>
    <td>
        联系地址 :
    </td>
    <td>
        <bean:write name="aul" property="adds" />
    </td>
</tr>
<tr align="center">
    <td colspan="2">
        <html:link action="show.do?
            method=showhuitie">返回回贴列表
        </html:link>
    </td>
</tr>
</table>
</logic:present>
<logic:equal name="u" value="null">
<table width="95%" height="34" align="center">

    <tr>
        <td align="center">
            所查用户不存在 !
        </td>
    </tr>
    <tr>
        <td align="center">
            <html:link action="show.do?
                method=showhuitie">返回回贴列表</html:link>
        </td>
    </tr>

</table>
</logic:equal>
</logic:present>
</body>
</html:html>
```



用户资料的显示来源于 how.do，请查看以前章节的源代码。

### 10.3.6 删除用户

单击“删除用户”即可删除当前查询出来的用户资料，Action 的程序代码如下。

```
package com.zjy.struts.action;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import org.apache.struts.action.ActionForm;
import org.apache.struts.action.ActionForward;
import org.apache.struts.action.ActionMapping;
import org.apache.struts.actions.DispatchAction;

import com.zjy.struts.mypackage.HuitieDao;
import com.zjy.struts.mypackage.UserDao;
import com.zjy.struts.mypackage.ZhutieDao;

public class DeleteAction extends DispatchAction {

    public ActionForward deleteAll(ActionMapping mapping, ActionForm form,
        HttpServletRequest request, HttpServletResponse response) {
        // TODO Auto-generated method stub
        String id = request.getParameter("id");// 主贴 id
        System.out.println(id);
        ZhutieDao zd = new ZhutieDao();
        HuitieDao hd = new HuitieDao();
        int rst1 = zd.deleteZhutie(id);// 删除主贴
        if (rst1 == 0) { // 删除主贴失败
            request.setAttribute("inf", "10");
            return mapping.findForward("tomessage");
        } else { // 删除主贴成功
            hd.deleteAll(id);// 删除回帖
            request.setAttribute("inf", "11");
            return mapping.findForward("tomessage");
        }
    }

    public ActionForward deleteHuitie(ActionMapping mapping,
        ActionForm form, HttpServletRequest request,
        HttpServletResponse response) {
        // TODO Auto-generated method stub
        String id = request.getParameter("id");// 回帖 id
        System.out.println(id);
        HuitieDao hd = new HuitieDao();
        int rst = hd.deleteHuitei(id);// 删除回帖
        if (rst == 0) { // 删除回帖失败
            request.setAttribute("inf", "12");
        }
    }
}
```

```
        return mapping.findForward("tomessage");
    } else { // 删除回贴成功
        request.setAttribute("inf", "13");
        return mapping.findForward("tomessage");
    }
}

public ActionForward deleteUser(ActionMapping mapping, ActionForm form,
    HttpServletRequest request, HttpServletResponse response) {
    // TODO Auto-generated method stub
    String ids = (String) request.getParameter("id");// 用户 id
    UserDao ud = new UserDao();
    int rst = ud.deleteUser(ids);// 删除用户
    if (rst == 0) {
        request.setAttribute("inf", "14");
        return mapping.findForward("tomessage");
    } else {
        request.setAttribute("inf", "15");
        return mapping.findForward("tomessage");
    }
}
}
```

### 10.3.7 删除主题及删除回复

单击右框架中的“删除主题”及“删除回复”的超级链接时，可以删除主题和回复的内容，程序代码都在 delete.do 的 Action 中。

删除主题的代码如下。

```
public ActionForward deleteAll(ActionMapping mapping, ActionForm form,
    HttpServletRequest request, HttpServletResponse response) {
    // TODO Auto-generated method stub
    String id = request.getParameter("id");// 主贴 id
    System.out.println(id);
    ZhutieDao zd = new ZhutieDao();
    HuitieDao hd = new HuitieDao();
    int rst1 = zd.deleteZhutie(id);// 删除主贴
    if (rst1 == 0) { // 删除主贴失败
        request.setAttribute("inf", "10");
        return mapping.findForward("tomessage");
    } else { // 删除主贴成功
        hd.deleteAll(id);// 删除回贴
        request.setAttribute("inf", "11");
        return mapping.findForward("tomessage");
    }
}
```

删除回复的程序代码如下。

```
public ActionForward deleteHuitie(ActionMapping mapping,
    ActionForm form,HttpServletRequest request,
    HttpServletResponse response) {
    // TODO Auto-generated method stub
    String id = request.getParameter("id");// 回帖 id
    System.out.println(id);
    HuitieDao hd = new HuitieDao();
    int rst = hd.deleteHuitei(id);// 删除回帖
    if (rst == 0) {// 删除回帖失败
        request.setAttribute("inf", "12");
        return mapping.findForward("tomessage");
    } else {// 删除回帖成功
        request.setAttribute("inf", "13");
        return mapping.findForward("tomessage");
    }
}
```

## 10.4 总结

通过本实例的学习,读者可以对 Struts 有更深入的认识,Struts 是一个非常方便的 Web MVC 框架,基于这个框架来进行 Web 的开发是非常便捷迅速的,掌握 Struts 的原理对读者学习 JSF 有更深远的意义。