第 23 章　使用Struts控制器简化MVC开发

本章将向读者介绍一种非常流行的 MVC 模式解决方案 Struts 技术。Struts 产品是 Apache 软件基金下的 Jakarta 项目的一部分，它是一种具体实现 MVC 的程序框架，其中采用了 Servlet 和 JSP 技术来实现。Struts 框架的使用简化了 MVC 模式的开发过程，下面将带领读者来详细认识 Struts。

本章要点包括以下内容：

- Struts 实现 MVC 开发模式
- Struts 的安装和配置
- 使 Dreamweaver 支持 Struts 标签功能
- Struts-config.xml 配置文件的详细介绍
- Struts 标签库的学习

23.1　Struts介绍

Struts 是把 Servlet、JSP 以及自定义标签等元素整合到一个统一的框架中，当开发者利用其进行开发时，就没有必要再自己编写全套的 MVC 框架代码了，这大大缩短了应用开发的周期。

正是由于 Struts 能充分满足应用开发的需求、简单易用、稳定可靠，Struts 已经成为 Web 应用框架事实上的标准。下面首先介绍 Struts 的框架结构和原理。

23.1.1　Struts的体系结构

在使用 Struts 开发 Web 应用之前，作者认为很有必要对 Struts 的整体框架结构以及原理有个总体的了解。Struts 的大致体系结构如图 23.1 所示。

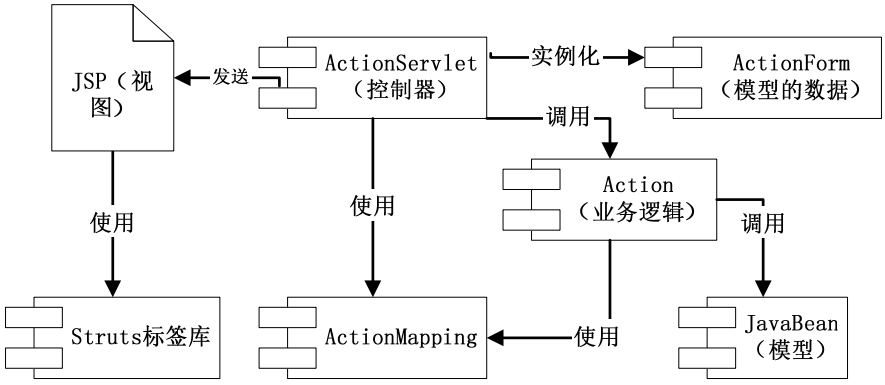

图 23.1　Struts 体系结构

结合图 23.1 详细介绍 Struts 的原理：

从左到右，分别是视图层（View）、控制器（Controller）和模型层（Model）。其视图层是通过 JSP 使用 Struts 标签库构建的；来自客户端的所有请求都统一由一个叫 ActionServlet（这里的 ActionServlet

代码，Struts 已经封装好了，可以直接使用）的 Servlet 接收，然后根据接收到的请求参数和 Struts 配置（struts-config.xml，很重要的一个配置文件，后面会重点介绍）中的 ActionMapping，将请求送给合适的 Action 去处理，等业务逻辑操作完之后，ActionServlet 再把处理结果返回给相应的视图进行显示；Action 则是 Struts 应用中真正干活的组件，开发人员一般都要在这里花费比较多的时间，这一层需要解决的是做什么的问题。Action 通过调用需要的业务组件（模型）来完成应用的业务，并把执行的结果以一个代表所需的显示响应的 JSP（或 Action）的 ActionForward 对象返回给 ActionServlet，再由 ActionServlet 来把响应显示给视图层。

图 23.1 显示的是 Struts 的体系结构图，那么 Struts 内部到底是怎么完成整个工作的，由图 23.2 可以很清楚的明白 Struts 的整个过程调用。

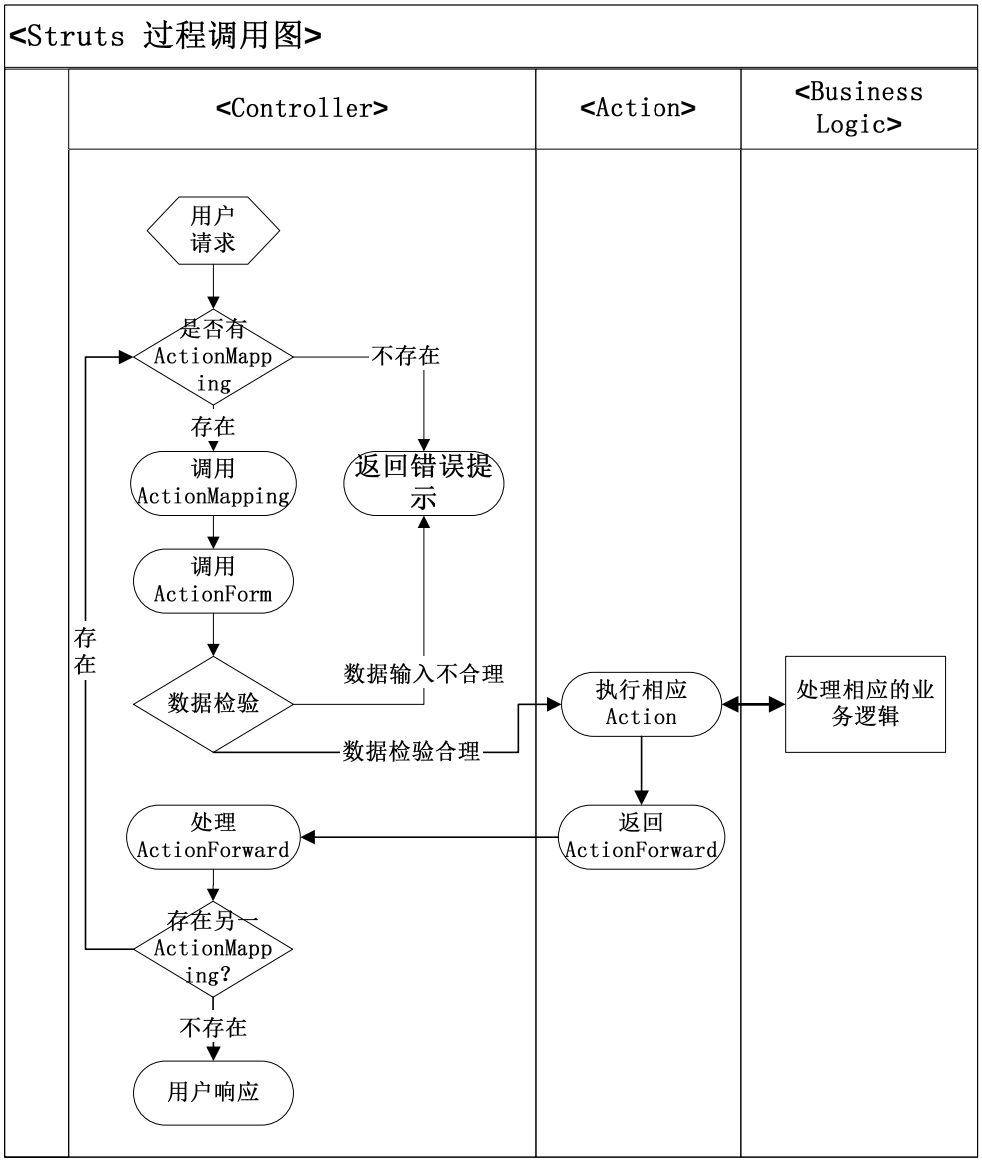

图 23.2　Struts 过程调用

下面结合图 23.2 详细介绍 Struts 处理用户请求的工作流程：

（1）首先客户端进行相应的请求，Struts 框架的中心控制器（叫 ActionServlet 的 Servlet 类，或者是用户自定义的 ActionServlet 类的子类）截获到客户请求。

（2）中心控制器 ActionServlet 根据在 struts-config.xml 文件中配置的映射信息将客户传递的数据填入相应的 ActionForm 组件对象，并对数据进行合理性校验。如果在配置文件中没有发现相应的映射信息将返回错误提示。

（3）如果 ActionForm 校验的数据不合理，将返回错误提示，否则 ActionServlet 把 ActionForm 对象传递给 Action 组件对象。

（4）Action 组件对象提取 ActionForm 组件对象中所包含的请求数据，必要时调用封装了业务逻辑的模型组件进行业务逻辑处理。

（5）Action 对象根据模型组件处理的结果返回相应的 ActionForward 对象给控制器 ActionServlet。

（6）控制器根据 Action 组件返回的 ActionForward 对象和 struts-config.xml 文件中的配置信息确定要跳转到的视图页面。

（7）视图访问模型组件返回的信息并进行相应的数据设置和显示，最后将生成的页面发送给客户端浏览器。

23.1.2　为什么使用Struts

其实 Model2 框架是可以自己搭建的，那么为什么还要使用 Struts 了，下面的理由是显而易见的：

（1）Struts 是建立在 MVC 这种公认的好模型之上，在 M、V 和 C 上都有涉及，但是它主要是提供了一个好的控制器和一套定制的标签库，也就是说 Struts 的着力点在 C 和 V 上。因此，Struts 天生就是 MVC 所带来的一系列优点，例如：结构层次分明、高可重用性、增加了程序的健壮性和可伸缩性。另外，使用 Struts 便于开发和设计分工，并提供了集中统一的权限控制、校验、国际化和日志等。

（2）它是一个开源项目，得到了很多程序大师和高手的持续呵护。并且已经经历了实战的检验，使其功能越来越强大，体系也日益完善。

（3）它对其他技术和框架显示出很好的融合性。例如，它已经和 titles 融为一体。

（4）Struts 是一个轻量级的产品技术，其中仅仅只有几个类和配置文件需要开发者去了解。

（5）允许为每个国家配置不同的信息资源，不同的语言翻译可以工作在他们自己的信息资源版本上。添加一个新国家语言的支持仅需要简单添加一个资源文件。

当然，和其他所有的技术产品一样，Struts 也不是十全十美的，例如：它对类和一些属性、参数的命名显得有些随意，给使用带来一些不便；还有，例如 Action 类中的 execute 方法只能接收一个 ActionForm 参数。但是瑕不掩瑜，这些并没有影响 Struts 被广泛使用。

23.2　Struts的安装和配置

介绍完了 Struts 的基本概念和原理后，接下来就要学习如何安装和配置 Struts，为 Struts 开发作准备。

23.2.1　Struts下载

本文使用的 Struts 版本为 1.2.4，下载的官方网站为：http://struts.apache.org/download.cgi。下载后的文件名为 struts-1.2.8-bin.zip。解压后，会出现如下的三个目录：

- contrib 目录.：这个目录是与 EL 和 JSTL 相关的文件内容。
- lib 目录：这个目录下包含很多需要使用到的核心文件，例如：JAR 包、TLD 以及 DTD 文件。

- webapps 目录：是关于 Struts 文档和例子的 WAR 文件。

23.2.2　Struts的安装和配置

由于本书介绍的是使用 Eclipse＋Lomboz 来构架 JSP 网站，所以接下来具体讲解如何在 Eclipse 中安装和配置 Struts，另外本章仍然以前面章节的用户登录应用作为实例。步骤如下：

（1）将 struts-1.2.8-bin.zip 解压，然后把 lib 目录下的所有 JAR 文件复制到 MyRegister 项目（这里仍然以第 12 章创建的 MyRegister 项目为例）的 Register/WEB-INF/lib 目录下。

（2）由于 Eclipse 并不能自动加载 WEB-INF/lib 目录下的 JAR 文件，所有这里需要手动的把复制到 WEB-INF/lib 目录下的 JAR 文件添加到 Eclipse 库引用中。方法如下：

首先右击 MyRegister 项目，选择“properties”命令，在弹出的“Properties for MyRegister”窗口左边的列表框中选择“Java Build Path”选项；然后再在右边窗口中选择“Libraries”选择卡，然后连续单击“Add JARs”按钮把 WEB-INF/lib 目录下的所有 JAR 文件添加到 Eclipse 库引用中。如图 23.3 所示。

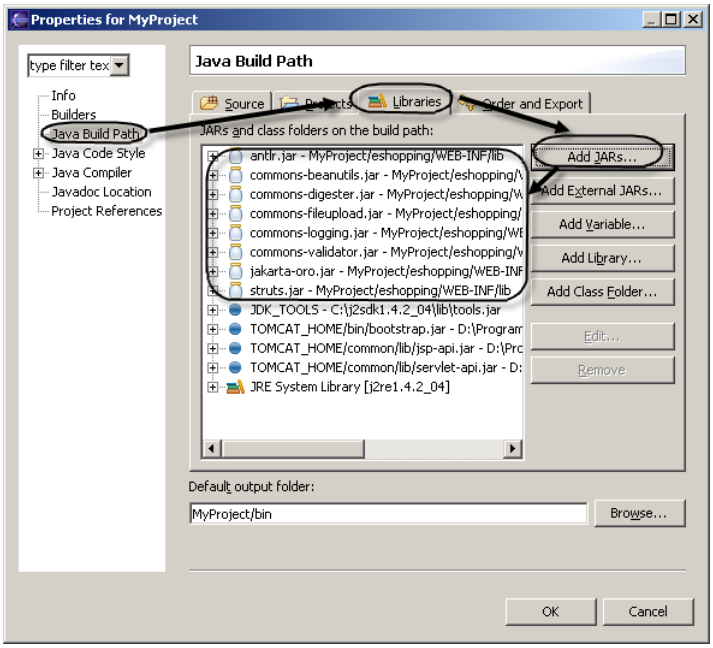

图 23.3　添加 JAR 文件到库引用中

（3）在 Register/WEB-INF 目录下创建一个 tlds 文件夹，将 struts-1.2.8-bin.zip 解压后 lib 目录下的所有 tld 文件（共五个文件）复制到新建的 tlds 文件夹中。这些 tld 文件就是在 Struts 开发过程中需要使用到的各种标签库。

（4）在创建 Register 模块时，就在 Register/WEB-INF 目录下生成了一个 web.xml 文件，这个文件是 Tomcat 的应用配置文件，这里需要对这个文件进行适当的修改。本文建议读者直接把 Struts 的例子程序 struts-blank.war 中的 web.xml 文件替换 eshopping/WEB-INF 目录下的 web.xml 文件，因为基本上所有的 struts 项目的 web.xml 文件配置都大同小异，所以只要对这个替换的 web.xml 作非常小的修改。一般 wen.xml 文件配置好之后，就很少再改动。

配置好的 web.xml 文件如下所示：

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE web-app
  PUBLIC "-//Sun Microsystems, Inc.//DTD Web Application 2.2//EN"
```

```
  "http://java.sun.com/j2ee/dtds/web-app_2_2.dtd">

<web-app>
  <display-name>Eshopping Application</display-name>

  <!-- Standard Action Servlet Configuration (with debugging) -->
  <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>2</param-value>
    </init-param>
    <init-param>
      <param-name>detail</param-name>
      <param-value>2</param-value>
    </init-param>
    <load-on-startup>2</load-on-startup>
  </servlet>

  <!-- Standard Action Servlet Mapping -->
  <servlet-mapping>
    <servlet-name>action</servlet-name>
    <url-pattern>*.do</url-pattern>
  </servlet-mapping>

  <!-- The Usual Welcome File List -->
  <welcome-file-list>
    <welcome-file>index.jsp</welcome-file>
  </welcome-file-list>

  <!-- Struts Tag Library Descriptors -->
  <taglib>
    <taglib-uri>/tags/struts-bean</taglib-uri>
    <taglib-location>/WEB-INF/tlds/struts-bean.tld</taglib-location>
  </taglib>

  <taglib>
    <taglib-uri>/tags/struts-html</taglib-uri>
    <taglib-location>/WEB-INF/tlds/struts-html.tld</taglib-location>
  </taglib>

  <taglib>
    <taglib-uri>/tags/struts-logic</taglib-uri>
    <taglib-location>/WEB-INF/tlds/struts-logic.tld</taglib-location>
  </taglib>
```

```
  <taglib>
    <taglib-uri>/tags/struts-nested</taglib-uri>
    <taglib-location>/WEB-INF/tlds/struts-nested.tld</taglib-location>
  </taglib>

  <taglib>
    <taglib-uri>/tags/struts-tiles</taglib-uri>
    <taglib-location>/WEB-INF/tlds/struts-tiles.tld</taglib-location>
  </taglib>

</web-app>
```

程序代码说明：

- <display-name>是 Web 应用的名称，当把应用程序发布到 Tomcat 服务器之后，就可以通过 Tomcat 的发布信息页面查看到这里定义的 Eshopping Application 名称。
- <servlet>定义了 struts 项目所需要使用到的 servlet 类，其中<servlet-class>子项定义了 ActionServlet 的 servlet 类。<param-value>子项定义了所要使用到的配置文件 struts-config.xml 所在位置，这里的 struts-config.xml 是 struts 学习的重点，在 struts 项目开发过程会不住地使用到这个文件。
- <servlet-mapping>是定义了访问 action URL 地址的后缀为*.do，例如，访问地址 http://localhost:8080/eshopping/LogonAction.do，就是需要访问一个 Action。
- <welcome-file-list>项定义了容器默认访问的主页，例如，当输入 http://localhost:8080/Register 地址时，容器会自动定位到 Web 模块主目录下的 index.jsp 文件。
- 最后<taglib>项定义各个 struts 中使用到的 TLD 标签。这里总共引入五个 TLD 标签。

（5）Register/WEB-INF 目录下创建一个 struts-config.xml 文件，这是 Struts 的核心配置文件（非常重要），它设置 ActionMapping 来控制着页面的流程转向，并且视图层和控制层就是通过这个配置文件融合在一起的。同样读者可以把 Struts 的例子程序 struts-blank.war 中的 struts-config.xml 文件复制过来，再进行适当的修改。struts-config.xml 配置内容会随着项目的进行而不断地需要添加，详细的讲解留在后面实例中介绍。

（6）在 MyRegister 项目的 j2src\cn\register 目录（必须是 j2src 目录下，但是读者可以选择其他子目录，读者应该知道这个目录下是存放 JavaBean 源文件的）下创建资源文件 MessageResources.properties，这个文件的主名读者可以任意取。

Struts 框架提供了一个很好用的以及灵活的消息系统。在 Java 或者 JSP 代码中，给定一个消息的关键字，消息文本就在运行的时候从属性文件中搜索到和关键字相关联的消息。

资源文件自身是一个平面的文本文件，每一行是一个关键字－值对。它的位置可以通过 struts-config.xml 文件进行配置。

它是和 Eclipse 中的资源文件一样，是用 UNICODE 编码的，可以安装一个国际化插件 jinto，下载地址为：http://www.guh-software.de/。在 Eclipse 中的安装方法和其他插件一样，可以参考第七章的内容。

对于资源文件的详细使用方法见下一章的实例讲解。

注意：很多 Hibernate 文档或者书籍都说将 MessageResources.properties 创建在 Register\WEB-INF\class 目录下，确实 struts 查找的是这个目录，但是本书建议创建在 j2src 目录下，这是因为 Eclipse 会自动将 MessageResources.properties 资源文件复制一份到 Register\WEB-INF\class 目录

下，并且保持两者同步。如果把 MessageResources.properties 资源文件直接创建在 Register\WEB-INF\class 目录下，你有时会惊讶地发现它不见了（如果你使用的 Eclipse 开发工具）。

至此，已经把 strtus 开发的前期工作做好了，接下来在实战中，就是要创建必要的 JSP 文件、以及对应的 ActionForm 和 Action 类，最重要的是 struts-config.xml 文件的配置。

23.3　使得Dreamweaver支持Struts标签

像 Dreamweaver 这些页面设计工具并不直接支持 Struts 的编程，即使用 Struts 标签编写的 JSP 页面并不能被 Dreamweaver 工具所识别和设计。这样使得 Struts 开发（特别是视图部分）带来了很多不便，有很多开发者就是由于这个原因而放弃使用 Struts。

其实 Dreamweaver 软件已经提供了一个插件，只要安装了这个插件就可以让 Dreamweaver UltraDev4 以及 Dreamweaver MX 页面设计工具识别出 Struts 的标签。本书使用的是 Dreamweaver MX。

23.4.1　下载插件

这个插件的全名是 ast-03.mxp，下载地址为 macromedia 公司网址 http://www.macromedia.com 或者 http://www.adobe.com/。打开网页之后，搜索关键字 Animalsgroup Struts taglibs Translator。如图 23.4 所示。

图 23.4　搜索页面

找到这个插件之后，把它下载下来。

23.4.2　安装插件

如果你已经安装了 Dreamweaver UltraDev4 或者 Dreamweaver MX 工具，就可以直接双击下载的 ast-03.mxp 插件，这样即可将该插件安装到 Dreamweaver 中。打开 Dreamweaver MX 软件，在菜单中选择“命令”|“扩展管理”命令，如果发现如图 23.5 所示，则说明这个插件已经安装成功。

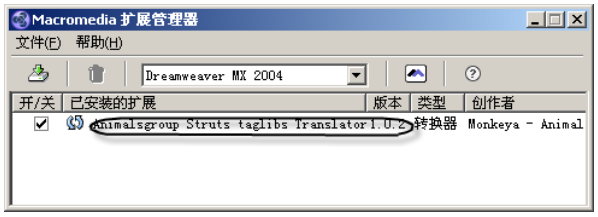

图 23.5　扩展管理器

23.4.3　在Dreamweaver中添加Struts标签

安装了 ast-03.mxp 插件之后，接下来还要在 Dreamweaver 中把 struts-1.2.8-bin.zip 解压后的 lib 目录下的所有 tld 文件添加进来。具体步骤如下：

（1）在 Dreamweaver 的主菜单中选择“编辑”|“标签库”命令，弹出如图 23.6 所示的“标签库编辑器”窗口。

图 23.6　标签库编辑器

（2）在 23.6 所示的窗口菜单中选择“从文件（*.tld、*.jar、*.zip）导入”命令，弹出“从文件（*.tld、*.jar、*.zip）导入”对话框，如图 23.7 所示。

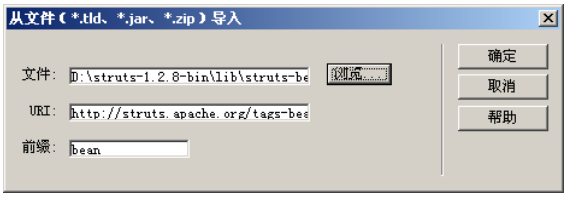

图 23.7　添加 tld 文件

（3）单击“浏览”按钮来添加 Struts 的 lib 目录下的所有 tld 文件，“URL”和“前缀”这两个文本框中的内容是不需要填写的，Dreamweaver 会自动通过添加的 tld 文件填写。最后单击“确定”按钮即完成此一个 tld 文件的添加。

由于总共有五个 tld 文件，所第三步骤需要依次操作五次。

注意：图 23.7 所示的“前缀”文本框中的内容必须和 JSP 文件头的<%@ taglib uri=”/tlds/struts-bean” prefix=”bean” %>**中的 prefix 属性值一致。在后面实例会重点讲解。**

23.5　Struts-config.xml配置文件

完成了 Struts 安装之后，下面重点讲解 Struts 中最为核心的两部分知识：Struts-config.xml 配置文件、Struts 标签库的使用。读者应该知道 struts 配置文件是一个用来部署 Java 对象的 xml 文档。struts-config.xml 配置文件和 TLD 标签的学习是整个 struts 最为重要的，也是最需要花费时间的。作者建议读者配合下一章的实例来学习下面将要介绍的这两部分重要内容。表 23.1 总结出来了配置文件中所使用到的配置元素：

表 23.1　struts-config.xml中所用的配置元素

配置元素	描述
data-sources	包含一个DataSource对象（JDBC2.0 标准扩展）的集合。

data-source	标识一个DataSrource对象，并且可以被实例化和进行配置，在servlet上下文中作为一个属性。
set-property	标识一个附加的JavaBean配置属性的方法名称和初始化值。
global-exceptions	描述一个可以被Action对象抛出的例外集合
exceptions	为一个例外类型注册一个ExceptionHandler。
form-beans	描述这个应用模块中所用的form bean描述符的集合。
form-bean	描述一个可以在<action>项中指定的ActionForm类。
form-properties	描述一个JavaBean属性，可以用来配置一个DynaActionForm实例或者其子类。
global-forwards	描述了一个对所有Action对象都能够作为返回值的ActionForward对象集合。
forward	描述一个能够被Action作为返回值的ActionForward对象。
action-mappings	描述一个能够用来处理和匹配ActionServlet定义到容器的url-pattern格式请求的ActinMapping对象集合。
action	描述一个ActionMapping对象，可以处理一个特定的URI请求。
controller	描述一个封装了应用模块运行时配置的控制器配置Bean。
message-resources	描述了该模块所要使用到的资源文件MessageResources对象所在位置。
plug-in	标识一个通用的plug-in模块的全限定类名，它接受应用的启动和推出事件的通知。

如今许多编程组件都以 XML 文件的方式进行资源配置，这样的好处也是显而易见的。下面对实际开发过程中比较常见的配置元素作重点介绍。

23.5.1　<global-exceptions>元素

为了在全部的 Action 对象类中进行一致的例外处理，可以在一个 struts-config.xml 配置文件中定义一个 ExceptionHandler（org.apache.struts.action.ExceptionHandler）。它可以在一个请求范围的属性下存储例外，为里外信息创建 ActionError 对象，并转发控制到 JSP 或者其他的 URI。Struts 的<html:errors/>标签会自动输出例外信息的本地化版本。所以你可以使用同一个页面来显示想用来显示的校验错误的例外错误信息。

如果需要的话，可以创建 ExceptionHandler 的子类以增加新的行为。

在定义一个例外时，你需要提供 Exception 类型类，消息关键字以及响应路径，具体设置如下：

```
<exception
    type=”org.apache.struts.webapp.example.ExpiredPasswordException”
    key=”error.password”
    path=”/error.jsp”/>
```

type 定义了一个 exception 类，用于处理例外。key 属性值调用资源文件中的关键字，path 设置响应的 JSP 文件。

23.5.2　<form-beans>元素

Struts 中的 ActionForm 类（org.apache.struts.action.ActionForm）提供了一个方便存储、通过 HTTP 请求提交输入属性值的方法。

Struts 配置文件提供了一个<form-bean>元素来归类一个模块所使用到的 ActionForm 类。具体的使用方法读者可以参考下一章将要介绍的实例。

某些特殊的 ActionForm，比如 DynaActionForms（org.apache.struts.action.DynaActionForm），它不需要创建类文件，但是需要在创建时传递额外的属性，然后由 Struts 根据 xml 配置，动态生成一个 ActionForm。

下面的是一个针对常规 ActionForm 以及 DynaActionForm 类的<form-bean>元素配置实例：

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

普通 ActionForm 类和动态 ActionForm 类存在明显的不同。在相应的 Action 类中，针对同样的 user_id 属性，从普通 ActionForm 类获取的方法为：String user_id = logonForm.getUser_id()；从动态 ActionForm 类中获取属性 user_id 的方法为：String user_id = (String)userForm.get(user_id)。

在<form-bean>子项中的 type 属性不一定要指定 DynaActionForm 类，也可以设置 DynaActionForm 类的多个子类，例如：DynaValidatorForm 类。它可以和 Validator 一起利用公共的 Validator 包提供自动校验的机制，使得可以在 Action 程序代码之外进行逻辑校验。

本文并不推荐多使用动态的 ActionForm，虽然它可以减少 ActionForm 类的创建（因为随着应用系统的开发，ActionForm 类会有很多），但是会增加 struts-config.xml 配置文件的负担，这时开发者不想见到的。

23.5.3　<global-forward>元素

在全局转发设置中的 path 属性既可以是 JSP 页面路径，也可以指定的是一个 Action 类的别名（例如上面实例中关于 Struts-config.xml 配置文件中的<globao-forwade>设置）。

下面是一个页面转换的实例代码：

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

这样设置之后，在页面中的<html:link forward="index">首页</html:link>，当单击这个“首页”链接时，会自动跳转到 index.jsp 页面。在之上实例中，通过全局设置，单击链接会转到一个 Action 类进行处理。

另外，在 Action 类中也可以通过这里的全局设置使用 mapping.findForward("index")来进行页面的转换。

这里全局的意思就是在所有的 JSP 页面以及所有不同 Action 类中都可以直接使用这里的页面全局转发。而在<action>项中进行的<forward>转发设置，只是针对单个相对应的 Action 类可以使用。具体理解见下一章的实例。

23.5.4　<action-mappings>元素

ActionForm 将应用需要的数据存储到一个 collect 中。ActionForward 归类那些应用要用到的 URI，而 ActionMapping 描述应用需要采取的操作、命令等。

Action 对象使用来处理操作的实际工作，但是一个操作包含大量的管理细节，ActionMapping 就是用来包装这些细节的。

<action-mapping> 元素描述了改应用需要用来处理请求的 ActionMapping 对象（org.apache.struts.action.ActionMapping）的集合。

基本所有定义的<action>元素至少包括下面几个属性：

- path：和相应 Action 类关联的路径引用名。
- type：指定 Action 类所在的位置。
- name：和相应 ActionForm 类关联的名称。

另外，有时还包括：

- validate 属性：在生命周期内，进行封装数据的初步校验。当 mapping 的 validate 属性设置为 true 的时候，ActionServlet 将调用 ActionForm 的 validate 方法。如果 validate 返回 false，请求将被转发到 input 属性表明的资源。
- Input 属性：当 validate 被设置为 true 的时候，重要的是提供了一个有效的输入路径，这也是当 ActionForm validate 方法返回 false 时，控制应该被传递的地方。
- Scope 属性：ActionForm bean 可以存储在当前的请求或者会话范围中（这样它可以服务于另外的请求）。虽然大部分开发人员使用请求范围来存储 ActionForm，框架的缺省设置却是会话范围。

其他属性在实际开发过程使用很少，这里就不一一介绍。

23.5.5　<Controller>元素

Struts 是允许多个 Web 应用模块共享同一个 servlet。但是每个模块确有自己的 struts 配置文件，并且是相对于其他模块进行独立开发的。

<controller>元素允许每个模块为 ActionServlet 标识一套不同的配置参数。例如你可以为每个模块插入不同的请求处理器，这样可以使得每个模块按自己的方式处理请求。

下面是一个<controller>元素的例子，它设置了 nocache 和 null 属性为 true，并且装入一个特制的请求处理器：

```
<controller>
  Nocache=”true”
  Null=”true”
  processorClass=”cn.register.struts.RequestProcessor” />
```

<controller>元素在实际开发过程使用比较少，所以读者只要基本了解即可。

23.5.6　<Message-Resources>元素

每个模块都有它自己默认的消息资源束，例如 JSP 标签等等。但也可以使用<message-resources>元素来部署应用系统所需要使用的额外资源。

关于<message-resources>元素的使用方法，前面实例已经仔细讲解了。这里不再重复。

23.5.7　<plug-in>元素

对 Action 类来说，可能需要特定的资源是比较少见的，但是，它可能需要使用一种非数据源兼容的连接池。或者需要创建一个应用类来为表单使用，又或者需要读入配置文件来创建一系列的对象，就像 struts-config.xml 文件所做的。

在比较常规的 Web 应用中，这些任务一般都是交付给 servlet 类来完成的。但是在 struts 应用中，是委托给 Action 类来完成的。

当 Action 希望实现或者销毁自己的资源的时候，它可以实现 PlugIn 接口（org.apache.struts.action.PlugIn）。这个接口定义了 Init()和 destroy()方法，在适当的时候，控制器会自动调用。

PlugIn 的 Action 类可以在 struts-config.xml 配置文件中通过<plug-in>元素进行配置。下面通过实例具体显示这个元素的使用方法，这个<plug-in>用来初始化 struts Validator：

```
<plug-in className=”org.apache.struts.validator.ValidatorPlugin”>
<set-property
    Property=”pathname” value=”/WEB-INF/validator-rules.xml”/>
<set-property
    Property=”pathname” value=”WEB-INF/validation.xml”/>
</plug-in>
```

23.5.8 <data-sources>元素

为了帮助开发人员进行数据库连接，其实 struts 也提供了一个数据源管理组件。也就是可以在 struts 中进行数据库连接的配置。

下面列举一个数据源配置的例子，是使用的 MySQL 数据库的 struts 缺省配置：

```
<data-sources>
<data-source>
<set-property property=”maxCount” value=”4”/>
<set-property property=”minCount” value=”2”/>
<set-property property=”description” value=”Artimus:MySQL Data Source Config”/>
<set-property property=”driverClass” value=”org.gjt.mm.mysql.Driver”/>
<set-property property=”url” value=”jdbc:mysql://localhost:3306/mydbs”/>
<set-property property=”autoCommit” value=”true”/>
<set-property property=”user” value=”root”/>
<set-property property=”password” value=”123456”/>
</data-source>
</data-sources>
```

如果所开发的数据库管理系统提供了自己的数据源可供 struts 使用（以上例子是通过 Tomcat 容器来配置数据库连接的），应该考虑直接使用这个实现。其实 struts 中使用的数据源执行效率并不高。

23.5.9 同一模块中使用多个配置文件

实际 Web 应用系统是非常庞大的，那么在配置文件 struts-config.xml 中需要配置的内容也是相当多的，这就会出现一个问题。单个的配置文件会变得非常冗长，当进行修改时，会非常容易出错。

Struts 也可以解决这个问题，它允许开发者通过声明可以创建多个 struts-config.xml 配置文件。这样开发者可以给系统中的每个子模块创建一个相应的配置文件，例如针对登录系统，可以创建一个名为 struts-cofig-logon.xml 配置文件。又由于系统一般都是按照子模块进行分工的，这样每个开发者可以使用自己的配置文件，以免发生混乱。

在生成了多个配置文件后，还需要在 web.xml 定义一下，修改 web.xml 如下：

```
<init-param>
    <param-name>config</param-name>
    <param-value>
                /WEN-INF/struts-config.xml,
```

```
                /WEB-INF/struts-config.login.xml,
                /WEB-INF/struts-config.register.xml
    </param-value>
</init-param>
```

各配置文件之间是以逗号隔开的。

但是这时又会发生另外一个问题，也就是各个开发者各自编辑自己的配置文件，但是 struts 在初始化的时候，是要把这几个定义的配置文件进行自动合并。这时就会非常容易出现命名冲突的问题。

为了避免这样的冲突，在全局配置文件 struts-cofig.xml 中可以使用全局的命名，但是各个模块的配置文件尽量在命名的前面加上字模块的名称。

划分多个配置文件，本文只建议在非常大型的应用系统中使用，而针对中小型应用系统，就没有必要根据各个子模块划分多个配置文件。

23.6　Struts标签库

Struts 提供了用来封装逻辑的各种定制 JSP 标签，主要包括五种标签库，其中最为常见的是 Bean、Logic 以及 HTML 三种标签库，每个标签库又包含了很多实用的标签。下面大致描述了各种标签库的功能：

- HTML（Struts-html.tld）：提供了显示各种 HTML 对象（例如：表单 form、按钮 button 和复选框 radio 等）的简单方法。
- Bean（Struts-bean.tld）：此标签库的使用使得 bean 以及新 bean 的定义更加的容易
- Logic（strut-logic.tld）：支持逻辑性的构造，以方便于有条件地显示某些文本或者作为处理循环的结果来显示文本。
- Nested（struts-nested.tld）：对于 struts 标签提供增强嵌套功能。
- Tiles（struts-tiles.tld）：Tiles 框架提供的动态模块标签库。

由于后两种标签库在实际开发过程中使用很少，所以本小节主要讲解前三种标签库的具体使用方法。

23.6.1　Bean标签库

这个标签库中包含了用于定义新 bean、访问 bean 以及其属性的各种标签。Struts 框架已经提供了多种自定义标签用来在 JSP 页中处理 JavaBean。这些标签被封装在一个普通的标记库中，然后在文件 struts-bean.tld 中定义了它的标签库描述器。Bean 标签库中的所有标签可以分成以下四个子类别：

- 创建和复制 bean 的标签。
- 脚本变量定义标签。
- bean 翻译标签。
- 消息国际化标签。

23.6.1.1　Bean 复制标签<bean:define>

此标签可定义新 bean、复制现有 bean，还可从现有 bean 中复制属性。<bean:define>标签主要的使用功能如下：

- 定义新字符串常数。
- 将现有的 bean 复制到新定义的 bean 对象。

- 复制现有 bean 的属性来创建新的 bean。

<bean:define>标签中的各个属性：

- id：新定义 bean 的变量名称，此属性是必须设置的。
- type：定义引入 bean 变量的类。
- value：为 id 属性定义的变量分配一个新的对象。
- name：目标 bean 的名称。若 value 属性没有设置，这个属性就必须设置。
- property：Name 属性定义的 bean 的属性名称，用来定义新的 bean。
- scope：源 bean 的作用域。若没有设置，搜索范围是从页作用域到应用程序作用域。
- toScope：目标 bean 的作用域。若没有设置，默认值是页作用域。

例如：通过<bean:define>标签来定义一个 bean：

```
<bean:define id="test" value="this is a test"/>
```

源 bean 在页作用域中被拷贝到请求作用域中的另一个 bean：

```
<bean:define id="targetBean" name="sourceBean"
scope="page" toScope="request"/>
```

23.6.1.2　定义脚本变量的标签<bean:cookie>

从多种资源中定义和生成脚本变量，这些资源包括 cookie、请求参数和 HTTP 标头等等。属性如下：

- Id：脚本变量和要定义的页作用域属性的名称。
- Name：cookie/标头/参数的名称。
- Multiple：如果这个属性设置了任意一个数值，所有匹配的 cookie 都会被积累并存储到一个 Cookie[](一个数组)类型的 bean 里。若无设置，指定 cookie 的第一个值将作为 Cookie 类型的值。
- Value：如果没有匹配的 cookie 或数值，就返回这个属性指定的默认值。下面还是通过一系列的实例讲解来加深读者都此标签的理解：

```
<bean:cookie id="myCookie" name="userName"/>
```

脚本变量名称是 myCookie，用来创建这个属性的 cookie 的名称是 userName。

```
<bean:header id="myHeader" name="Accept-Language"/>
```

脚本变量名称是 myHeader，请求标头的名称是 Accept-Language。

```
<bean:parameter id="myParameter" name="myParameter">
```

脚本变量名称是 myPatameter,它保存的请求参数的名称也是 myParameter。

<bean:include>标签将对一个资源的响应进行检索，并引入一个脚本变量和字符串类型的页作用域属性。这个资源可以是一个页，一个 ActionForward 或一个外部 URL。与<jsp:include>的不同是资源的响应被存储到一个页作用域的 bean 中，而不是写入到输出流。属性如下：

- Id：脚本变量和要定义的页作用域属性的名称。
- Page：一个内部资源。
- Forward：一个 ActionForward。
- Href：要包含的资源的完整 URL。

示例如下：

```
<bean:include id="myInclude" page="MyJsp?x=1"/>
```

脚本变量的名称是 myInclude，要检索的响应来自资源 MyJsp?x=1。

<bean:resource>标签将检索 web 应用中的资源，并引入一个脚本变量和 InputStream 或字符串类型的页作用域属性。如果在检索资源时发生问题，就会产生一个请求时间异常。属性如下：

- Id：脚本变量和要定义的页作用域属性的名称。
- Name：资源的相对路径。

- Input：如果这个属性不存在，资源的类型就是字符串。

例如：

```
<bean:resource id=”myResource” name=”/WEB-INF/images/myResource.xml”/>
```

脚本变量的名称是 myResource，要检索的资源的名称是 myResource.xml。

23.6.1.3　显示 Bean 属性的标签<bean.write>

标签库中定义了<bean:write>标签，用来将 bean 的属性输送到封装的 JSP 页写入器。这个标记与<jsp:getProperty>类似，属性如下：

- Name：要进行属性显示的 bean 的名称。
- Property：要显示的属性的名称。如果这个属性类有 java.beans.PropertyEditor,getAsText()或 toString 方法会被调用。
- Scope：Bean 的作用域，若没有设置，搜索范围是从页到应用程序作用域。
- Filter：如果设置 true，属性中的所有特殊 HTML 字符都将被转化为相应的实体引用。
- Ignore：如果设置 false，当发现属性时会产生一个请求时间异常，否则返回 null。

例如：

```
<bean:write name=”myBean” property=”myProperty” scope=”request”
filter=”true”/>
```

myBean 的属性 myProperty 将会被显示，作用域为请求，如果发现任何 HTML 特殊字符都将被转化为相应的实体引用。

23.6.1.4．消息标签<bean:message>和国际化

Struts 框架支持国际化和本地化。用户在他们的计算机中定义自己所在的区域，当 web 应用程序需要输出一条消息时，它将引用一个资源文件，在这个文件中所有的消息都使用了适当的语言。一个应用程序可能提供了很多资源文件，每个文件提供了用不同语言编写的消息。如果没有找到所选语言的资源文件，就将使用默认的资源文件。

struts 框架对国际化的支持是使用<bean:message>标记，以及使用 java.util 数据包中定义的 Locale 和 ResourceBundle 类来实现 Java2 平台对这些任务的支持。Java.text.MessageFormat 类定义的技术可以支持消息的格式。利用此功能，开发人员不需了解这些类的细节就可进行国际化和设置消息的格式。

使用<bean:message>标签在 JSP 页面调用资源文件中的各个消息，其中该标签中的属性如下：

- Key：资源文件中定义消息关键字。
- Locale：用户会话中存储的区域对象的属性名称。若没有设置，默认值是 Action.LOCALE_KEY。
- Bundle：在应用程序上下文中，存储资源对象的属性的名称。如果没有设置这个属性，默认值是 Action.MESSAGE_KEY。
- arg0：第一个替换参数值。
- arg1：第二个替换参数值。
- arg2：第三个替换参数值。
- arg3：第四个替换参数值。

例如：资源文件中定义了一个消息：

```
info.myKey = The numbers entered are {0},{1},{2},{3}
```

用户可使用下面的消息标记：

```
<bean:message key=”info.myKey” arg0=”5” arg1=”6” arg2=”7” arg3=”8”/>
```

这个信息标记输出到 JSP 页会显示为：The numbers entered are 5,6,7,8

23.6.2　逻辑标签logic

逻辑库的标签能够用来处理外观逻辑而不需要使用 scriptlet。Struts 逻辑标签库包含的标签能够有条件地产生输出文本，在对象集合中循环从而重复地产生输出文本，以及应用程序流程控制。它也提供了一组在 JSP 页中处理流程控制的标签。这些标签封装在文件名为 struts-logic.tld 的标签包中。逻辑标签库定义的标签能够执行下列三个功能：

- 条件逻辑。
- 重复。
- 转发/重定向响应。

23.6.2.1　条件逻辑标签

struts 有三类条件逻辑。

第一类可以比较下列实体与一个常数的大小：

- Cookie。
- 请求参数。
- bean 或 bean 的参数。
- 请求标头。

以下列出了这一类标签：

- <logic:equal>：如果常数与被定义的实体相等，返回 true。
- <notEqual>：如果常数与被定义的实体不相等，返回 true。
- <greaterEqual>：如果常数大于等于被定义的实体，返回 true。
- <lessEqual>：如果常数小于等于被定义的实体，返回 true。
- <lessThan>：如果常数小于被定义的实体，返回 true。
- <greaterThan>：如果常数大于被定义的实体，返回 true。

这一类的所有标记有相同的属性：

- Value：要进行比较的常数值。
- Cookie：要进行比较的 HTTP cookie 的名称。
- Header：要进行比较的 HTTP 请求标头的名称。
- Parameter：要进行比较的 HTTP 请求参数的名称。
- Name：如果要进行比较的是 bean 或 bean 的属性，则这个属性代表 bean 的名称。
- Property：要进行比较的 bean 属性的名称。
- Scope：Bean 的作用域，如果没有指定作用域，则它的搜索范围是从页到应用程序。

例如：

```
<logic:equal parameter=”name” value=”SomeName”>
The entered name is SomeName
</logic:equal>
```

判断名为“name”的请求参数的值是否为“SomeName”。

```
<logic:greaterThan name=”bean” property=”prop” scope=”page” value=”7”>
The value of bean.Prop is greater than 7
</logic:greaterThan>
```

判断在页的作用域中是否有一个名为“bean”的 bean，它有一个 prop 属性，这个属性的值是否大于 7。如果这个属性能够转化为数值，就进行数值比较，否则就进行字符串比较。

第二类条件标签定义了两个标签：

- <logic:present>。
- <logic:notPresent>。

它们的功能是在计算标签体之前判断特定的项目是否存在。标签的属性和属性值决定了要进行检查的项目。具体属性如下：

- Cookie：由这个属性指定的 cookie 将被检查是否存在。
- Header：由这个属性指定的请求标头将被检查是否存在。
- Parameter：由这个属性指定的请求参数将被检查是否存在。
- Name：如果没有设置 property 属性，那么有这个属性指定的 bean 将被检查是否存在。如果设置了，那么 bean 和 bean 属性都将被检查是否存在。
- Property：检查有 name 属性指定的 bean 中是否存在指定的属性。
- Scope：如果指定了 bean 的名称，这就是 bean 的作用域。如果没有指定作用域，搜索的范围从页到应用程序作用域。
- Role：检查当前已经确认的用户是否属于特殊的角色。
- User：检查当前已经确认的用户是否有特定的名称。

例如：

```
<logic:notPresent name=”bean” property=”prop” scope=”page”>
The bean property bean.prop is present
</logic:notPresent>
```

标签判断在页作用域中是否存在一个名为“bean”的 bean，这个 bean 有一个 prop 属性。

第三类条件标签比较复杂，这些标签根据模板匹配的结果检查标签体的内容。换句话说，这些标签判断一个指定项目的值是否是一个特定常数的子字符串：

- <logic:match>。
- <logic:notMatch>。

这些标签允许 JSP 引擎在发现了匹配或是没有发现时计算标记主体。属性如下：

- Cookie：要进行比较的 HTTP cookie 的名称。
- Header：要进行比较的的 HTTP 标头的名称。
- Parameter：要进行比较的的 HTTP 请求参数的名称。
- Name：若要对 bean 或 bean 的属性进行比较，这个属性是用户指定 bean 的名称。
- Location：如果设置了这个属性的值，将会在这个指定的位置(索引值)进行匹配。
- Scope：如果对 bean 进行比较，这个属性指定了 bean 的作用域。如果没有设置这个参数，搜索范围是从页到应用程序作用域。
- Property：要进行比较的 bean 的属性名称。
- Value：要进行比较的常数值。

例如：

```
<logic:match parameter=”name” value=”xyz” location=”1”>
The parameter name is a sub-string of the string xyz from index 1
</logic:match>
```

标签检查名为“name”的请求参数是否是“xyz”的子字符串，但是子字符串必须从“xyz”的索引位置 1 开始（也就是说子字符串必须是“y”或“yz”）。

23.6.2.2　重复标签

在逻辑标签库中定义了<logic:iterate>标签，它能够根据特定集合中元素的数目对标签体的内容进行重复的检查。集合的类型可以是 java.util.Iterator,java.util.Collection，java.util.Map 或是一个数组。有三种

方法可以定义这个集合：

- 使用运行时间表达式来返回一个属性集合的集合
- 将集合定义为 bean，并且使用 name 属性指定存储属性的名称。
- 使用 name 属性定义一个 bean，并且使用 property 属性定义一个返回集合的 bean 属性。

当前元素的集合会被定义为一个页作用域的 bean。属性如下，所有这些属性都能使用运行时表达式。

- Collection：如果没有设置 name 属性，它就指定了要进行重复的集合。
- Id：页作用域 bean 和脚本变量的名称，它保存着集合中当前元素的句柄
- Indexed：页作用域 JSP bean 的名称，它包含着每次重复完成后集合的当前索引
- Length：重复的最大次数。
- Name：作为集合的 bean 的名称，或是一个 bean 名称，它由 property 属性定义的属性，是个集合。
- Offset：重复开始位置的索引。
- Property：作为集合的 Bean 属性的名称。
- Scope：如果指定了 bean 名称，这个属性设置 bean 的作用域。若没有设置，搜索范围从页到应用程序作用域。
- Type：为当前定义的页作用域 bean 的类型。

例如：

```
<logic:iterate id=”currentInt”
collection=”<% =myList %>”
type=”java.lang.Integer”
offset=”1”
length=”2”>
<% =currentint %>
</logic:iterate>
```

代码将从列表中的第一个元素开始重复两个元素并且能够让当前元素作为页作用域和 java.lang.Integer 类型的脚本变量来使用。也就是说，如果 myList 包含元素 1，2，3，4 等，代码将会打印 1 和 2。

23.6.2.3　转发和重定向标签

转发标签<logic:forward>能够将响应转发给重定向到特定的全局 ActionForward 上。ActionForward 的类型决定了是使用 PageContext 转发响应，还是使用 sendRedirect 将响应进行重定向。此标记只有一个“name”属性，用来指定全局 ActionForward 的名称，例如：

```
<logic:forward name=”myGlobalForward”/>
```

重定向标签<logic:redirect>是一个能够执行 HTTP 重定向的强大工具。根据指定的不同属性，它能够通过不同的方式实现重定向。它还允许开发人员指定重定向 URL 的查询参数。属性如下：

- Forward：映射了资源相对路径的 ActionForward。
- Href：资源的完整 URL。
- Page：资源的相对路径。
- Name：Map 类型的页名称，请求，会话或程序属性的名称，其中包含要附加大哦重定向 URL（如果没有设置 property 属性）上的“名称-值”参数。或是具有 Map 类型属性的 bean 名称，其中包含相同的信息（没有设置 property 属性）。
- Property：Map 类型的 bean 属性的名称。Bean 的名称由 name 属性指定。
- Scope：如果指定了 bean 的名称，这个属性指定搜索 bean 的范围。如果没有设置，搜索范围从

页到应用程序作用域。

- ParamID：定义特定查询参数的名称。
- ParamName：字符串类型的 bean 的名称，其中包含查询参数的值(如果没有设置 paramProperty 属性)；或是一个 bean 的名称，它的属性(在 paramProperty 属性中指定)包含了查询参数值。
- paramProperty：字符串 bean 属性的名称，其中包含着查询参数的值。
- ParamScope：ParamName 定义的 bean 的搜索范围。

使用这个标签时至少要指定 forward，href 或 page 中的一个属性，以便标明将响应重定向到哪个资源。

23.6.3　HTML标签

Struts 的 HTML 标签可以大致地分为以下几个功能：

- 显示表单元素和输入控件。
- 显示错误信息。
- 显示其他 HTML 元素。

23.6.3.1　显示表单元素和输入控件

struts 将 HTML 表单与为表单操作而定义的 ActionForm bean 紧密联系在一起。表单输入字段的名称与 ActionForm bean 里定义的属性名称是对应的。当第一次显示表单时，表单的输入字段是从 ActionForm bean 中移植过来的，当表单被提交时，请求参数将移植到 ActionForm bean 实例。

所有可以在<html:form>标签中使用的用来显示 HTML 输入控件的内嵌标记都使用下列属性来定义 JavaScript 事件处理器。

- Onblur：字段失去了焦点。
- Onchange：字段失去了焦点并且数值被更改了。
- Onclick：字段被单击。
- Ondblclick：字段被鼠标双击。
- Onfocus：字段接收到输入焦点。
- Onkeydown：字段拥有焦点并且有键按下。
- Onkeypress：字段拥有焦点并且有键按下并释放。
- Onkeyup：字段拥有焦点并且有键被释放。
- Onmousedown：鼠标指针指向字段并且被单击。
- Onmousemove：鼠标指针指向字段并且在字段内移动。
- Onmouseout：鼠标指针指向控件，但是指针在元素外围移动。
- Onmouseover：鼠标指针没有指向字段，但是指针在元素内部移动。
- Onmouseup：鼠标指针指向字段，并且释放了鼠标按键。

<form>元素中能够被定义的其他一般属性有：

- Accesskey：定义访问输入字段的快捷键。
- Style：定义输入字段的样式。
- styleClass：定义输入字段的样式表类。
- Tabindex：输入字段的 tab 顺序。

表单标签<html:form>用来显示 HTML 标签，可以指定 AcitonForm bean 的名称和它的类名。如果没有设置这些属性，就需要有配置文件来指定 ActionMapping 以表明当前输入的是哪个 JSP 页，以及从映

射中检索的 bean 名和类。如果在 ActionMapping 指定的作用域中没有找到指定的名称，就会创建并存储一个新的 bean，否则将使用找到的 bean。

<form>标签能够包含与各种 HTML 输入字段相对应的子标签。

<html:form>标签属性如下：

- Action：与表单相关的操作。在配置中，这个操作也用来标识与表单相关的 ActionForm bean。
- Enctype：表单 HTTP 方法的编码类型。
- Focus：表单中需要初始化焦点的字段。
- Method：表单使用的 HTTP 方法。
- Name：与表单相关的 ActionForm bean 的名称。如果没有设置这个属性，bean 的名称将会从配置信息中获得。
- Onreset：表单复位时的 JavaScript 事件句柄。
- Onsubmit：表单提交时的 JavaScript 事件句柄。
- Scope：搜索 ActionForm bean 的范围。如果没有设置，将从配置文件中获取。
- Style：使用的格式。
- styleClass：这个元素的格式表类。
- Type：ActionForm bean 的完整名称。如果没有设置，将从配置文件获得。

例如：

```
<html:form action=”validateEmploee.do” method=”post”>
</html:form>
```

与表单相关的操作路径是 validateEmployee,而表单数据是通过 POST 传递的。对于这个表单来说，ActionForm bean 的其他信息，如 bean 名称类型，作用域，都是从表单指定操作的 ActionMapping 中检索得到的：

```
<form-beans>
<form-bean name=”empForm” type=”com.example.EmployeeForm”/>
</form-beans>
<action-mappings>
<action path=”/validateEmployee”
type=”com.example.ValidateExampleAction”
name=”empForm”
scope=”request”
input=”/employeeInput.jsp”>
<forward name=”success” path=”/employeeOutput.jsp”>
</action>
</action-mapping>
```

如果配置文件中包含上述信息，并且请求 URI 的*.do 被映射到 ActionServlet，与表单相关的 ActionForm bean 的名称，类型和作用域分别是 empForm,com.example.EmployeeForm 和 request.这些属性也可以使用<html:form>标签属性进行显示的定义。

以下标签必须嵌套在<html:form>标签里。

（1）按钮和取消标签：

<html:button>标签显示一个按钮控件；<html:cancel>标签显示一个取消按钮。属性如下：

- Property：定义在表单被提交时返回到服务器的请求参数的名称
- Value：按钮上的标签

（2）复位和提交标签：

<html:reset>和<html:submit>标签分别能够显示 HTML 复位按钮和提交按钮。

（3）文本和文本区标记：

<html:text>和< html:textarea>标签分别是 HTML 文本框和文本区，属性如下：

- Property：定义当表单被提交时送回到服务器的请求参数的名称，或用来确定文本元素当前值的 bean 的属性名称
- Name：属性被查询的 bean 的名称，它决定了文本框和文本区的值。如果没有设置，将使用与这个内嵌表单相关的 ActionForm 的名称。

<html:text>标签还有以下属性：

- Maxlength：能够输入的最大字符数。
- Size：文本框的大小（字符数）。

<html:textarea>标签特有的属性如下：

- Rows：文本区的行数。
- Cols：文本区的列数。

（4）检查框和复选框标签：

<html:checkbox>标签能够显示检查框控件。<html:multibox>标签能够显示 HTML 复选框控件，请求对象在传递检查框名称时使用的 getParameterValues()调用将返回一个字符串数组。属性如下：

- Name Bean 的名称，其属性会被用来确定检查是否以选中的状态显示。如果没有设置，将使用与这个内嵌表单相关的 ActionFrom bean 的名称。
- Property：检查框的名称，也是决定检查框是否以选中的状态显示的 bean 属性名称。在复选框的情况下，这个属性必须是一个数组。
- Value：当检查框被选中时返回到服务器的请求参数的值。

例如：

```
<html:checkbox property=”married” value=”Y”/>
```

一个名为 married 的检查框，在表单提交时会返回一个“Y”。

（5）文件标签：

<html:file>标签可以显示 HTML 文件控件。属性如下：

- Name：Bean 的名称，它的属性将确定文件控件中显示的内容。如果没设置，将使用与内嵌表单相关的 ActionForm bean 的名称。
- Property：这个属性定义了当表单被提交时送回到服务器的请求参数的名称，以及用来确定文件控件中显示内容的 bean 属性名称。
- Accept：服务器能够处理的内容类型集。它也将对客户浏览器对话框中的可选文件类型进行过滤。
- Value：按钮上的标签，这个按钮能够在本地文件系统中浏览文件。

（6）单选钮标签：

<html:radio>标签用来显示 HTML 单选钮控件，属性如下：

- Name：Bean 的名称，其属性会被用来确定单选钮是否以选中的状态显示。如果没有设置，将使用与这个内嵌表单相关的 ActionFrom bean 的名称。
- Property：当表单被提交时送回到服务器的请求参数的名称，以及用来确定单选钮是否以被选中状态进行显示的 bean 属性的名称
- Value：当单选钮被选中时返回到服务器的值

（7）隐藏标签：

<html:hidden>标签能够显示 HTML 隐藏输入元素，属性如下：

- Name：Bean 的名称，其属性会被用来确定隐藏元素的当前值。如果没有设置，将使用与这个内嵌表单相关的 ActionFrom bean 的名称。
- Property：定义了当表单被提交时送回到服务器的请求参数的名称，以及用来确定隐藏元素当前值的 bean 属性的名称。
- Value：用来初始化隐藏输入元素的值。

（8）密码标签：

<html:password>标签能够显示 HTML 密码控件，属性如下：

- Maxlength：能够输入的最大字符数。
- Name：Bean 的名称，它的属性将用来确定密码元素的当前值。如果没有设置，将使用与这个内嵌表单相关的 ActionFrom bean 的名称。
- Property：定义了当表单被提交时送回到服务器的请求参数的名称，以及用来确定密码元素当前值的 bean 属性的名称。
- Redisplay：在显示这个字段时，如果相应的 bean 属性已经被设置了数据，这个属性决定了是否显示密码的内容。
- Size：字段的大小。

（9）选择标签：

<html:select>标签能够显示 HTML 选择控件，属性如下：

- Multiple：表明这个选择控件是否允许进行多选。
- Name：Bean 的名称，它的属性确定了哪个。如果没有设置，将使用与这个内嵌表单相关的 ActionFrom bean 的名称。
- Property：定义了当表单被提交时送回到服务器的请求参数的名称，以及用来确定哪个选项需要被选中的 bean 属性的名称。
- Size：能够同时显示的选项数目。
- Value：用来表明需要被选中的选项。

（10）选项标签(这个元素需要嵌套在<html:select>标签里)：

<html:option>标签用来显示 HTML 选项元素集合，属性如下：

- Collection：Bean 集合的名称，这个集合存储在某个作用域的属性中。选项的数目与集合中元素的数目相同。Property 属性能够定义选项值所使用的 bean 属性，而 labelProperty 属性定义选项标记所使用的 bean 的属性。
- labelName：用来指定存储于某个作用域的 bean，这个 bean 是一个字符串的集合，能够定义<html:option>元素的标记(如果标志与值不相同)。
- label：Property 与 collection 属性共同使用时，用来定义了存储于某个作用域的 bean，这个 bean 将返回一个字符串集合，能够用来写入<html:option>元素的 value 属性。
- Name：如果这是唯一被指定的属性，它就定义了存储于某个作用域的 bean，这个 bean 将返回一个字符串集合，能够用来写入<html:option>元素的 value 属性。
- Property：这个属性在与 collection 属性共同使用时，定义了每个要显示选项值的独立 bean 的 name 属性。如果不是与 collection 属性共同使用，这个属性定义了由 name 属性指定的 bean 的属性名称(如果有 name 属性)，或是定义了一个 ActionForm bean，这个 bean 将返回一个集合来写入选项的值。

下面是这个标签的一些例子：

```
<html:option collection=”optionCollection” property=”optionValue”
labelProperty=”optionLabel”/>
```

标签假设在某个作用域中有一个名为 optionCollection 的集合，它包含了一些具有 optionValue 属性的独立的 bean，每个属性将作为一个选项的值。每个选项的标志由 bean 的 optionLabel 属性属性进行定义。

```
<html:option name=”optionValues” labelName=”optionLabels”/>
```

标签中 optionValues 代表一个存储在某个作用域中的 bean，它是一个字符串集合，能够用来写入选项的值，而 optionLabels 代表一个存储在某个作用域中的 bean，它也是一个字符串集合，能够用来写入选项的标志。

23.6.3.2　显示错误信息的标签

<html:errors>标签能够与 ActionErrors 结合在一起来显示错误信息。这个标签首先要从当前区域的资源文件中读取消息关键字 errors.header，然后显示消息的文本。接下去它会在 ActionErrors 对象(通常作为请求参数而存储在 Action.ERROR_KEY 关键字下)中循环，读取单个 ActionError 对象的消息关键字，从当前区域的资源文件中读取并格式化相应的消息，并且显示它们。然后它读取与 errors.footer 关键字相对应的消息并且显示出来。

通过定义 property 属性能够过滤要显示的消息，这个属性的值应该与 ActionErrors 对象中存储 ActionError 对象的关键字对应。属性如下：

- Bundle：表示应用程序作用域属性的名称，它包含着消息资源，其默认值 Acion.MESSAGE_KEY
- Locale：表示会话作用域属性的名称，它存储着用户当前登录的区域信息。其默认值是 Action.ERROR_KEY
- Name：表示请求属性的名称，它存储着 ActionErrors 对象。其默认值是 Action.ERROR_KEY
- Property：这个属性指定了 ActionErrors 对象中存储每个独立 ActionError 对象的关键字，它可以过滤消息

例子：

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

显示集合中所有的错误。

```
<html:errors property=”missing.name”/>
```

显示存储在 missing.name 关键字的错误。

23.6.3.3　其他 HTML 标签

struts HTML 标签还定义了下列标记来显示其他 HTML 元素：

- <html:html>：显示 HTML 元素。
- <html:img>：显示图象标记。
- <html:link>：显示 HTML 链接或锚点。
- <html:rewrite>：创建没有锚点标记的 URI。

这些标签的详细内容请参照 struts 文档。

23.7　本章小结

本章重点介绍了 Struts 的安装和配置，Struts 学习的核心是 struts-config.xml 配置文件和 TLD 标签，这些内容的学习就将要花费读者 90%的时间。控制器 ActionServlet 类 Struts 已经写好，直接使用即可。另外还需要开发者编写是 Action 和 ActionForm 类，以及资源文件*.properties。总体来说 struts 的使用是比较方便的。下一章节将以实例重点讲解如何使用 Struts 进行实例开发。