目录

概述

本文主要讲解什么是 Struts Framework，它的框架结构，组件结构，以及简单的配置讲解。

文章的包括了如下七大部分：

Framework 的概念和体系简介

Struts 的概念和体系结构

Struts 的工作原理和组件

Struts 配置文件简介

Struts 高级特性

Struts 标记库

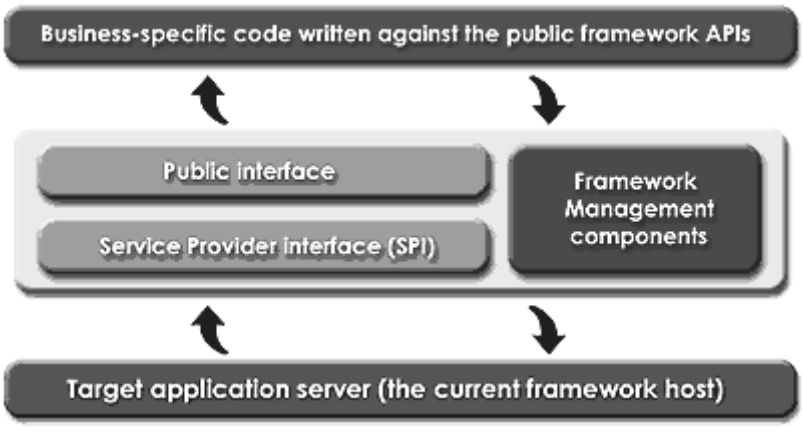
一个简单的示例

Framework 概念

一直以来我们都说 Struts 是一个 Web Framework。那么让我么先来看看什么是 Framework。

Framework 概念并不是很新了，伴随着软件开发的发展，在多层的软件开发项目中，可重用、易扩展的，而且是经过良好测试的软件组件，越来越为人们所青睐。这意味着人们可以将充裕的时间用来分析、构建业务逻辑的应用上，而非繁杂的代码工程。于是人们将相同类型问题的解决途径进行抽象，抽取成一个应用框架。这也就是我们所说的 Framework。

Framework 的体系提供了一套明确机制，从而让开发人员很容易的扩展和控制整个 Framework 开发上的结构。通常，Framework 的结构中都有一个“命令和控制”组件（"command and control" component）——Framework Factory and Manager。



Framework 体系

通过基于请求响应（Request-Response）模式的应用 Framework，基本上有如下几个表现逻辑结构组成。

控制器（Controller）——控制整个 Framework 中各个组件的协调工作。

业务逻辑层（Business Logic）——对 Framwork 本身来说，这里仅仅只是概念和几个提够服

务的基础组件，真正的实现与客户的业务逻辑接轨，还需要开发人员在 Framework 上再次扩展。

数据逻辑层（Data Logic）——绝大应用系统都需要涉及到数据交互，这一层次主要包括了数据逻辑和数据访问接口。对于数据逻辑来说，如果你了解数据建模（Data Modeling）可能就很容易理解。

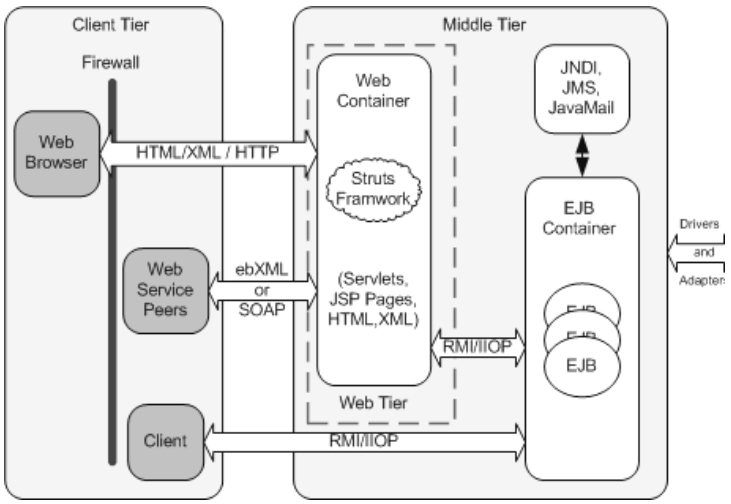
Struts 的概念和体系结构

Struts 有一组相互协作的类、Serlvet 以及 Jsp TagLib 组成。基于 Struts 构架的 web 应用程序基本上符合 JSP Model2 的设计标准，可以说是 MVC 设计模式的一种变化类型。根据上面对 framework 的描述，我们很容易理解为什么说 Struts 是一个 web framwork，而不仅仅是一些标记库的组合。但 Struts 也包含了丰富的标记库和独立于该框架工作的实用程序类。

Struts 有其自己的控制器（Controller），同时整合了其他的一些技术去实现模型层（Model）和视图层（View）。在模型层，Struts 可以很容易的与数据访问技术相结合，包括 EJB,JDBC 和 Object Relation Bridge。在视图层，Struts 能够与 JSP, Velocity Templates,XSL 等等这些表示层组件想结合。

Struts 的与 Web App 的关系

既然 struts 叫做 web framework，那么其肯定主要基于 web 层的应用系统开发。按照 J2EE Architecture 的标准，struts 应当和 jsp/servlet 一样，存在于 web container 一层。

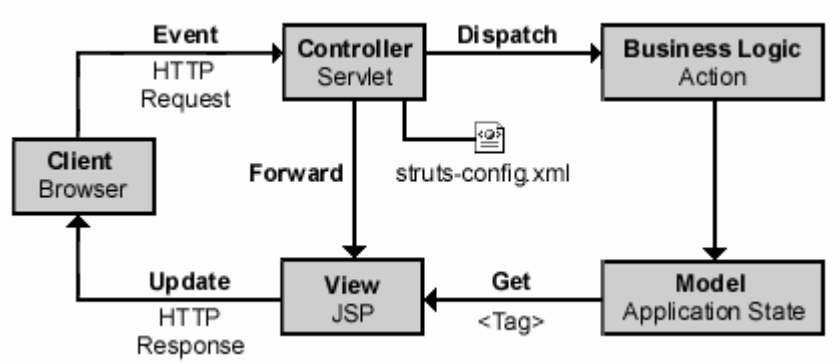


Struts 与 WebApp 的关系

Struts 的体系结构

我们说 struts framework 是 MVC 模式的体现，下面我们就从分别从模型、视图、控制

来看看 struts 的体系结构（Architecture）。下图显示了 struts framework 的体系结构响应客户请求时候，各个部分工作的原理。



Struts 体系结构

从视图角度（View）

首先，Struts 提供了 Java 类 org. apache.struts.action.ActionForm，Java 开发者将该类细分来创建表单 bean。在运行时，该 bean 有两种用法：

- 当 JSP 准备相关的 HTML，表单以进行显示时，JSP 将访问该 bean(它保存要放入表单中的值)。那些值是从业务逻辑或者是从先前的用户输入来提供的。
- 当从 Web 浏览器中返回用户输入时，该 bean 将验证并保存该输入以供业务逻辑或(如果验证失败的话)后续重新显示使用。

其次，Struts 提供了许多定制 JSP 标记，它们的使用简单，但是它们在隐藏信息方面功能强大。例如，除了 bean 名称和给定 bean 中每个段的名称之外，页面设计者不需要知道有关表单 bean 的更多信息。

从模型角度（Model）

Struts 虽然不直接有助于模型开发。在 Struts 中，系统模型的状态主要由 ActiomForm Bean 和值对象体现。

从控制器角度（Controller）

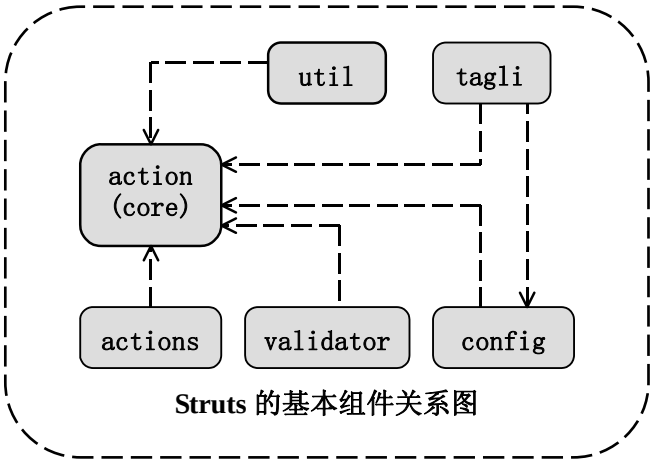
在 Struts framework 中， Controller 主要是 ActionServlet，但是对于业务逻辑的操作则主要由 Action、ActionMapping、ActionForward 这几个组件协调完成（也许这几个组件，应该划分到模型中的业务逻辑一块）。其中，Action 扮演了真正的控制逻辑的实现者，而 ActionMapping 和 ActionForward 则指定了不同业务逻辑或流程的运行方向。

Struts 的基本组件包

整个 struts 大约有 15 包，近 200 个类所组成，而且数量还在不断的扩展。在此我们不

能一一介绍，只能列举几个主要的简要的介绍一下。下表说明了目前 struts api 中基本的几个组件包，包括 action,actions,config,util,taglib,validator。下图则显现了这几个组件包之间的关系。其中 action 是整个 struts framework 的核心

org.apache.struts.action
基本上，控制整个 struts framework 的运行的核心类、组件都在这个包中，比如我们上面提到的控制器 ActionServlet。已经 Action,ActionForm,ActionMapping 等等。struts1.1 比 1.0 多了 DynaActionForm 类。增加了动态扩展生成 FormBean 功能
org.apache.struts.actions
这个包是主要作用是提供客户的 http 请求和业务逻辑处理之间的特定适配器转换功能，而 1.0 版本中的部分动态增删 FromBean 的类，也在 struts1.1 中被 Action 包的 DynaActionForm 组件所取代
org.apache.struts.config
提供对配置文件 struts-config.xml 元素的映射。这也是 sturts1.1 中新增的功能
org.apache.struts.util
Strtuts 为了更好支持 web application 的应用，体统了一个些常用服务的支持，比如 Connection Pool 和 Message Source。详细信息请参考 http://jakarta.apache.org/struts/api/org/apache/struts/util/package-summary.html
org.apache.struts.taglib
这不是一个包，而是是一个客户标签类的集合。下面包括 Bean Tags，HTML Tags，Logic Tags，Nested Tags，Template Tags 这几个用于构建用户界面的标签类。
org.apache.struts.validator
Struts1.1 framework 中增加了 validator framework，用于动态的配置 from 表单的验证。详细信息请参阅 http://home.earthlink.net/~dwinterfeldt/



Struts 的基本组件关系图

Struts framework 的工作原理和组件

对于 Struts 如何控制、处理客户请求，让我们通过对 struts 的四个核心组件介绍来具体说明。这几个组件就是：ActionServlet。Action Classes，Action Mapping（此处包

括 ActionForward)，ActionFrom Bean。

Struts ActionServlet 控制器对象

ActionServlet 继承自 javax.servlet.http.HttpServlet 类，其在 Struts framework 中扮演的角色是中心控制器。它提供一个中心位置来处理全部的终端请求。控制器 ActionServlet 主要负责将 HTTP 的客户请求信息组装后，根据配置文件的指定描述，转发到适当的处理器。

按照 Servelt 的标准，所有得 Servlet 必须在 web 配置文件（web.xml）声明。同样，ActoinServlet 必须在 Web Application 配置文件（web.xml）中描述，有关配置信息如下。

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

全部的请求 URI 以*.do 的模式存在并映射到这个 servlet，其配置如下：

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

一个该模式的请求 URI 符合如下格式：

```
http://www.my_site_name.com/mycontext/actionName.do
```

中心控制器为所有的表示层请求提供了一个集中的访问点。这个控制器提供的抽象概念减轻了开发者建立公共应用系统服务的困难，如管理视图、会话及表单数据。它也提供一个通用机制如错误及异常处理，导航，国际化，数据验证，数据转换等。

当用户向服务器端提交请求的时候，实际上信息是首先发送到控制器 ActionServlet，一旦控制器获得了请求，其就会将请求信息传交给一些辅助类（help classes）处理。这些辅助类知道如何去处理与请求信息所对应的业务操作。在 Struts 中，这个辅助类就是 org.apache.struts.action.Action。通常开发者需要自己继承 Aciton 类，从而实现自己的 Action 实例。

Struts Action Classes

ActionServlet 把全部提交的请求都被控制器委托到 RequestProcessor 对象。RequestProcessor 使用 struts-config.xml 文件检查请求 URI 找到动作 Action 标示符。

一个 Action 类的角色，就像客户请求动作和业务逻辑处理之间的一个适配器（Adaptor），其功能就是将请求与业务逻辑分开。这样的分离，使得客户请求和 Action 类之间可以有多个点对点的映射。而且 Action 类通常还提供了其它的辅助功能，比如：认证（authorization）、日志（logging）和数据验证（validation）。

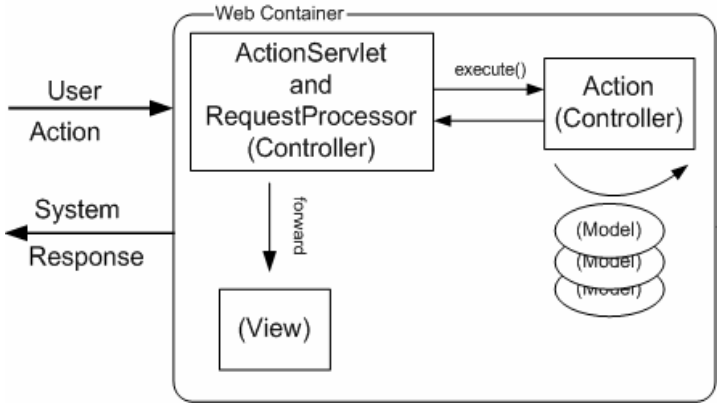
Action 最为常用的是 execute（）方法。（注意，以前的 perform 方法在 struts1.1 中

```
public ActionForward execute(ActionMapping mapping,
                             ActionForm form,
                             javax.servlet.ServletRequest request,
                             javax.servlet.ServletResponse response)
                      throws java.io.IOException,javax.servlet.ServletException
```

```
<action-mappings>
  <action  path="/logonAction"
           type="com.test.LogonAction"
           name="LogonForm"
           scope="request"
           input="logoncheck.jsp"
           validate="false">
     <forward name="welcome" path="/welcome.jsp"/>
     <forward name="failure" path="/logon_failure.jsp "/>
  </action>
</action-mappings>
```

已经不再支持），还有一个 execute（）方法，请参考 apidoc，在此不在说明。

当 Controller 收到客户的请求的时候，在将请求转移到一个 Action 实例时，如果这个实例不存在，控制器会首先创建，然后会调用这个 Action 实例的 execute（）方法。Struts Framework 为应用系统中的每一个 Action 类只创建一个实例。因为所有的用户都使用这一个实例，所以你必须确定你的 Action 类运行在一个多线程的环境中。下图显示了一个 execute（）方法如何被访问：



Action 实例的 execute()方法

注意，客户自己继承的 Action 子类，必须重写 execute（）方法，因为 Action 类在默认情况下是返回 null 的。

Struts Action Mapping

上面讲到了一个客户请求是如何被控制器转发和处理的，但是，控制器如何知道什么样的信息转发到什么样的 Action 类呢？这就需要一些与动作和请求信息相对应的映射配置说明。在 struts 中，这些配置映射信息是存储在特定的 XML 文件（比如 struts-config.xml）。

这些配置信息在系统启动的时候被读入内存，供 struts framework 在运行期间使用。在内存中，每一个<action>元素都与 org.apache.struts.action.ActionMapping 类的一个实例对应。下表就显示了一个登陆的配置映射。

上面的配置表示：当可以通过/logonAction.do（此处假设配置的控制器映射为*.do）提交请求信息的时候，控制器将信息委托 com.test.LogonAction 处理。调用 LogonAction 实例的 execute()方法。同时将 Mapping 实例和所对应的 LogonForm Bean 信息传入。其中 name=LogonForm，使用的 form-bean 元素所声明的 ActionForm Bean。有关 form-bean 的申明如下显示。

```
<form-beans>
  <form-bean  name="LoginForm"
              type="com.test.LoginForm"/>
</form-beans>
```

使用 ActionForward 导航

元素<forward>则表示了当 Action 实例的 execute()方法运行完毕或，控制器根据 Mapping 可将响应信息转到适当的地方。如上面现实，如果客户登陆成功，则调用 welcome forward，将成功信息返回到/welcome.jsp 页面。在你的 execute()方法的结尾可以使用下面的实例代码而返回 welcome forward。当然你的 welcome forward 必须在 action 元素属性中定义，正如上面所声明的那样。

```
return (mapping.findForward("welcome"));
```

ActionForward 对象是配置对象。这些配置对象拥有独一无二的标识以允许它们按照有意义的名称如“success”，“failure”等来检索。ActionForward 对象封装了向前进的 URL 路径且被请求处理器用于识别目标视图。ActionForward 对象建立自<forward>元素位于 struts-config.xml。下面是一个 Struts 中<forward>元素例子，属于<action>元素范围。

```
<action path="/editCustomerProfile"
type="packageName.EditCustomerProfileAction"
name="customerProfileForm" scope="request">
  <forward name="success" path="/MainMenu.jsp"/>
  <forward name="failure" path="/CustomerService.jsp"/>
</action>
```

基于执行请求处理器的 execute(…)方法的结果，当传递一个值匹配指定于<forward>元素中 name 属性的值的时候，下一个视图可以在 execute(…)方法中被开发者用方便的方法 org.apache.struts.action.ActionMapping.findForward(…)选择。ActionMapping.findForward(…)方法既从它的本地范围又从全局范围提供一个 ActionForward 对象，该对象返回至 RequestProcessor 以 RequestDispatcher.forward(…)或 response.sendRedirect(…)调用下一个视图。当<forward>元素有 redirect=“false”属性或 redirect 属性不存在的时候，RequestDispatcher.forward(…)被执行；当 redirect=“true”是，将调用 sendRedirect(…)方法。下例举例说明了 redirect 属性的用法：

```
<forward name="success" path="/Catalog.jsp" redirect="true"/>
```

如果 redirect=true, URL 建立如/contextPath/path 因为 HttpServletResponse.sendRedirect(…)中解释 URL 采用”/”开头相对于 servlet 容器根目录。

如果 redirect=false, URI 建立如/path 因为 ServletContext.getRequestDisptacher(…)采用虚拟目录相关 URL。

在此稍稍说一下有关 global-forwards 的概念。其在配置文件中描述了整个应用系统可以使用的 ActionForward，而不是仅仅是一个特定的 Action。

```
<global-forwards>
  <forward name="logout" path="/logout.do"/>
  <forward name="error"  path="/error.jsp"/>
</global-forwards>
```

Struts ActionForm Bean 捕获表单数据

在上面讲解 ActionServlet，Action Classes 和 Action Mapping 的时候，我们都提到了 ActionForm Bean 的概念。一个应用系统的消息转移（或者说状态转移）的非持久性数据存储，是由 ActionForm Bean 的负责保持的。

ActionForm 派生的对象用于保存请求对象的参数，因此它们和用户紧密联系。

一个 ActionForm 类被 RequestProcessor 建立。这是发生在已完成向前进到一个 URL，该 URL 为映射到控制器 servlet 而不是 JSP 和相应的动作映射指定的表单属性的。在这个情况下，如果没有在指定的活动范围内找到，RequestProcessor 将尝试寻找可能导致创建一个新 ActionForm 对象的表单 bean。该 ActionForm 对象在指定的活动范围内被用<action>元素的 name 属性找到；

RequestProcessor 将随后重新安排表单属性，用请求时参数填充表单，随即调用表单对象的 validate(…)方法以履行服务器端用户输入验证。仅当 ActionMapping 对象中 validate 属性被设为 true 时，validate(…)方法被调用；这就是默认的行为。
request.getParameterValues(parameterName)被用于得到一个 String[]对象，它用来表单填充；验证的结果应该是一个 ActionErrors 对象，用 org.apache.struts.taglib.html.ErrorsTag 来显示验证错误给用户。ActionForm 也可以被用于为当前用户保存即将被一个视图引用的中间模型状态。

当一个表单对象被 RequestProcessor 找到，它被传递到请求处理器的 execute(…)方法。一个 ActionForm 对象也可以被请求处理器建立。表单对象建立目的是提供中间模型状态给使用请求范围 JSP；这将确保对象不会在有效性过期后仍然存在。默认的，所有的表单都被保存为会话范围。会话中表单对象脱离有效性的存在可能导致浪费内存，同样的，请求处理器必须跟踪保存在会话中的表单对象的生命周期。一个好的捕获表单数据的实践是为横跨多用户交互的相关表单用一个单独的表单 bean。表单 bean 也可以在反馈的时候用来储存能够被自定义标签改变的中间模型状态。在视图中标签用法避免结合 Java 代码，因此要成一个好的任务划分，web 生产组主要处理标志，而应用开发组主要处理 Java 代码。标签因素退出访问中间模型状态的逻辑；当访问嵌套的对象或当通过聚集列举时这个逻辑可能很复杂。

注意：在 struts1.1 中，ActionForm 的校验功能，逐渐被剥离出来（当然依然可以使用）。使用了 validator framework 对整个应用系统的表单数据验证进行统一管理。相信信息请参考：http://home.earthlink.net/~dwinterfeldt

在 ActionForm 的使用中，Struts 提倡使用到值对象（Value Object）。这样将客户或开发人员，对数据状态与对象状态能够更加清晰的理解和使用。

对于每一个客户请求，Struts framework 在处理 ActionForm 的时候，一般需要经历如下几个步骤：

（1）检查 Action 的映射，确定 Action 中已经配置了对 ActionForm 的映射

（2）根据 name 属性，查找 form bean 的配置信息

（3）检查 Action 的 formbean 的使用范围，确定在此范围下，是否已经有此 form bean 的实例。

（4）假如当前范围下，已经存在了此 form bean 的实例，而是对当前请求来说，是同一种类型的话，那么就重用。

（5）否则，就重新构建一个 form bean 的实例

（6）form bean 的 reset()方法备调用

（７）调用对应的 setter 方法，对状态属性赋值

（８）如果 validatede 的属性北设置为 true，那么就调用 form bean 的 validate()方法。

（9）如果 validate（）方法没有返回任何错误，控制器将 ActionForm 作为参数，传给 Action 实例的 execute（）方法并执行。

注意：直接从 ActionFrom 类继承的 reset()和 validate()方法，并不能实现什么处理功能，所以有必要自己重新覆盖。

Struts 的其他组件

Struts framework 本身提供了很多可扩展的组件或 sub framework，方便的开发人员在其构架上构建 web 层的应用系统。比如 upload,collections ,logging 等等。让我们来看看两个比较重要的组件：validationg framework 和 struts taglib。有关其他组件请参考 Struts 用户手册（http://jakarta.apache.org/struts/userGuide）。

Validation Framework for Struts

在 struts1.1 中，新增了 validation framework。增加了对 form 数据提交的验证。将原本需要在 ActionFrom Bean 的 validate（）进行的验证通过配置文件的描述进行验证。

有关其详细信息，请参考 http://home.earthlink.net/~dwinterfeldt 。个人建议对于小型应用系统可以采用这种配置方式，但是对于应用系统中有大量 web 层表单应用的系统，并且业务需求变动比较大的，使用 validation framework 可能会加重开发难度、系统维护难度。可以借鉴 validation framework 的 Javascript Validator Tag。

Struts TagLib

struts 提供了一组可扩展的自定义标签库（TagLib），可以简化创建用户界面的过程。目前包括：Bean Tags，HTML Tags，Logic Tags，Nested Tags，Template Tags 这几个 Taglib。有关 Struts Taglib 的结构和使用，可以参考前面有关 Cutomer Tag Lib 的介绍，有关起详细资料，请参考

BeanUtils

这个组件的全称是 Bean Introspection Utilites。是属于 Jakarta Commons 项目组的。主要是帮助构建 javabean 的属性操作的（getter,setter），已经提供一种动态定义和访问 bean 的属性。有关详细信息，请参考。

http://jakarta.apache.org/commons/beanutils.html

如果各位对这方面有很兴趣，可以参考一些有关 java 反射（Reflectio）方面的资料。

Collections

这个组件主要是提供了一些集合或列表对象，在原有的 java collections framework 的基础上进行了扩展。详细资料请参考：
http://jakarta.apache.org/commons/collections.html 以及
http://cvs.apache.org/viewcvs/~checkout~/jakarta-commons/collections/STATUS.html?rev=1.13

Digester

这个组件翻译成中文的意思是“汇编”。其主要功能是根据 xml 配置文件，初始化系统的一些 java 类对象。Digester 帮助你指定 XML 与 java 对象之间映射模型，而且允许客户话定制映射规则（rules）。详细资料请参考
http://jakarta.apache.org/commons/digester.html

Struts 配置文件简介

Struts framework 根据配置文件使得 ServletAction， ActionMapping,Action ，ActionForm 这几个不同层次的组件相互交互，协调的工作。这些配置文件是在系统启动的时候，读入导内存中，供控制器使用的。

Struts framework 主要包括三部分的配置描述，一个是指定有关 Struts Controller 及其相关的的配置描述（Initialization Parameters），一个对 struts tag lib 的描述，一个是 struts 组件（ActionMapping,Action，ActionForm）之间相互映射协调的关系

有关 Struts Controller 及其相关的的配置描述

因为 Struts Controller 的主要类 ActionServlet 是继承自 HttpServlet，所以必须像配置一个 Servlet 那样在部署描述符(Web.xml)中配置 ActionServlet 类及其访问映射。

当您第一次创建基于 Struts 的 Web 应用程序时，将为您创建一个部署描述符，这通常就足够了。该文件包括下列条目：

- <servlet>条目定义用于 Web 应用程序的 servlet(在本例中，这是唯一的 servlet)：
  - <servlet-name> 和<servlet-class>指示 ActionServlet (标识为“操作”)接收 HTTP 请求并确定如何响应。
  - <init-param>表示 servlet 初始化参数.
    - “config”指示 ActionServlet 的行为由指定的配置文件来指导，该配置文件通常具有以下名称：
      \WEB-INF\struts-config.xml
    - “debug”具有整数值，它指示将有关处理的详细信息写至控制台的程度。
    - ”detail”具有整数值，它指示将“映射”详细信息(如后面所述)写至控制台的程度。

— <load-on-startup>导致在启动应用程序时装入 servlet。

- <servlet-mapping>元素标识这样的命名模式：当命名模式由 URL 进行匹配时，Web 服务器就将控制权移交给 ActionServlet。考虑下面各种情况：
  — 访问了 ActionServlet，原因是“操作”(<servlet-mapping>中的<servlet-name>元素的内容)与“操作”(<servlet>中的<servlet-name>元素的内容)相匹配。
  — <servlet-mapping>元素指定 URL 的结尾的命名模式。每个 URL 的开头都是应用程序上下文路径。按照惯例，ActionServlet 调用对象以响应与命名模式“*do”(其中“*”是通配符)一致的 URL。
- <welcome-file-list>元素指示获得初始控制权的特定于应用程序的代码；在本例中，Web 服务器直接从 Web Content 目录中调用 index.jsp。
- <error-page>元素指示显示哪个 JSP 来响应错误;在本例中，错误为如下所示：
  — 404 (找不到资源)
  — 500 (Web 服务器内部发生错误)
- 每个<taglib>元素都使相对 URL(相对于 Web.xml)与标记库描述符(相对于 Web 应用程序根目录)相关联。每个 JSP 都可以使用同一个 URL 来表示给定的标记库，而 Web.xml 确定引用了哪个文件。

有关 struts tag lib 的配置描述

如果你的 web application 打算使用 Struts 的 taglib，那么你有必要在 web.xml 中对 struts taglib 进行配置描述。

有关 Struts Action Mapping 的配置描述

作为先前描述的 web.xml 设置的结果，Web 应用程序服务器将请求的一个子集按路径发送至 ActionServlet，它通常调用一系列操作和 JSP。ActionServlet 的响应是基于配置文件 struts-config.xml 的内容的。有关其 DTD 文档的描述，请参考 http://jakarta.apache.org/struts/dtds/struts-config_1_1.dtd

一般 struts-config（version1.1）包含了如下几个部分：

（1）form-bean

（2）global-forwards

（3）action-mappings

（4）data-sources

我们知道，对于这样的一个请求（例如，表示为“/login.do”），执行下列步骤：

1、 寻找操作类对象（继承 org. apache.struts.action.Action 的类）

2、 ActionServlet 调用操作类对象的执行方法

操作类中的执行方法的特征符为如下所示：

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

- 映射对象（ActionMapping），它包含指示如何响应方法的每个可能结果的规则（“映射”）
- Struts 表单 bean（ActionForm），它保存发送至 HTML 表单或接收自 HTML 表单的数据
- 请求和响应对象（HttpServletReques/ HttpServletResponse）

3、 从执行方法返回 ActionForward 对象，用于指导 ActionServlet 接着访问哪个操作类或 JSP

返回的 ActionForward 对象中的信息取决于两个值：

- 方法的结果（如在“成功”或“故障”等字符串中所述）
- 映射对象，它包含从 Struts 配置文件中读取的信息

要弄明白某些运行时关系，要明白 struts-config.xml 该文件包括下面的一组条目：

- ＜form-beans＞标记标识每个表单 bean
- ＜action-mappings＞标记包括用于指导应用程序流的信息，每个＜action＞子标记都使相对 URL 与操作类和潜在的后续操作相关。

Form-bean 元素

Struts 配置文件中的一个示例＜form-bean＞子元素为如下所示：

```
<form-bean name=“registerForm”  type=“strutscommon.RegisterForm”/>
```

每个<form-bean>子元素都包括下列属性：

name

表单 bean 的名称，稍后在配置文件中会用到。ActionServlet 举例说明了该 bean（如果需要的话）并在将对 bean 的引用存储在请求或会话对象中时将该名称用作键。

type

类的全限定名称，它继承 org.apache.struts.action.ActionForm 该类必须在类路径中。接受“Struts 贸易样本”中的注册的表单 bean 包括 HTML 注册表单中每个字段的 getter 和 setter 方法。该 bean 还包括验证方法，如下节“验证”中所述。

Action 元素

Struts 配置文件中的一个示例＜action＞元素为如下所示：

```
<action path=“/register”
      type=“strutsEGL.RegisterAction”
    name=“registerForm”
    input=“/register.jsp”
    scope=“request”
    <forward name=“success”path=“/home.do”/>
    < forward name=“failure”path=“/register.jsp”/>
 </action>
```

每个＜action＞元素都包括下列属性中的某些属性或所有属性：

path

将请求指定为非限定 URL，不带文件扩展名（例如，“/register”）请求是根据＜action＞元素中的其它属性来处理的，并且是用户输入的结果或者是在 different ＜action＞元素中标识的转发的结果。

type

指定在发出请求时调用其执行方法的操作类的全限定名。该类必须在类路径中。注：不指定要实例化的类，可以通过使用 forward 属性来转发请求，该属性在“Struts 贸易样本”中未使用，并且与后面描述的＜forward＞子元素不相同。

name

用于保存发送至 HTML 表单或接收自 HTML 表单的数据表单 bean 的名称。

input

指定相对 URL（例如，“/register.do”或“/index.jsp”）必须包括后缀，如果表单 bean 的验证方法指示发生了输入错误，则会调用 URL；有关详细信息，参见下节的“验证”。

scope

指定将对表单 bean 的引用存储在哪个作用域中。其值为“会话”（缺省值）或“请求”。

Struts 配置文件中的每个＜action＞元素还包括子元素＜forward＞，它指定从方法结果至后续调用的映射。每个＜forward＞子元素都包括下列属性

name

指定导致在运行时使用当前映射的字符串（例如，“success”），但是只限于以下情况：在 type 中引用的操作类的执行方法使用完全相同的字符串来配置返回至 ActionServlet 的 ActionForward 对象。下面的执行方法不是很重要，但是会导致使用“success”映射：

```
public ActionForward exectue(
    ActionMapping mapping,
    ActoinForm form,
    HttpServletRequest request,
    HttpServletResponse response)
    Throws IOException,ServletException
{
    ActionForward forward=new ActionForward();
    Forward=mapping,findForward(“success”);
    return(forward)；
}
```

path

指定非限定 URL（例如，“/home.do” 或“/index.jsp”）必须包括文件扩展名，仅当使用当前映射时才会调用该 URL，转发操作类是根据 different<action>元素中的属性来处理的，尤其是，在其 path 属性标识相同 URL 的<action>元素中。

有必要提一下的是，在 struts1.1 中，提出了对 Multiple Application Support。在 struts 的早先版本中，只有一个 struts 配置文件，一般叫 struts-config.xml。但是，对

于越来越复杂的应用系统的发展，只有一个地方存放这个一个文件，对大型项目来说，使用和修改这个配置文件，使其成为了一个应用的瓶颈问题。在 struts1.1 中，你可以定义多了配置文件协同工作。

注：当用户或 ActionServlet 调用 JSP 时，请求是由 Web 应用程序服务器直接处理的不会受到 ActionServlet 的干预。

Struts 高级特性（Struts Advanced Feature）

验证

仅当在下列情况下才会在表单 bean 中对用户输入进行验证：

- 表单 bean 覆盖超类的验证方法
- Struts 配置文件中的<action>元素的验证属性显式地或者缺省设置为 TRUE。表单 bean 没有关于应用程序的业务逻辑的信息；因此该 bean 提供的验证仅限于一些相对简单的问题，例如，用户是否在每个必填字段中都输入了数据？Struts 框架的各种部件使得可以进行验证
- Struts 配置文件中的以下<action>子元素将导致使用表单 bean registerForm:

  <action path="/register"
  type="strutsEGL.RegisterAction"
  name="registerForm"
  input="/register.jsp"
  scope="request"
  <forward name="success" path="/home.do"/>
  <forward name="failure"path="/register.jsp"/>
  </action>

  如果缺少验证属性，则意味着当 ActionServlet 接收到来自 HTML 表单对"/register"的请求时，ActionServlet 将在接收用户数据的表单 bean 中调用验证方法。此验证在 ActionServlet 访问操作类之前进行。如果丢失了该方法，不会发生任何错误，在该情况下，验证总是会成功。
- 如果发生了错误，则表单 bean 的验证方法将举例说明错误类并将错误条目添加至该类。registerForm 的验证方法的一个子集为如下所示:

```
ActionErrors errors=new ActionErrors();
 If (username= =unll‖username.equals(""))
 {
  errors.add("register",
        new ActionError("error. Register.username"));
 }
  if (openingBalance<0.01)
 {
  errors.add("register",
```

new ActionError(“error. register. balance”));
}
return errors;
errors.add 方法包括两个参数：

property
用来标识错误类别的 Java 字符串。
如果想要在特定的输入字段或输入字段的子集发生错误时标识该错误，则指定属性值。例如，指定诸如“username”之类的值的优点在于：报告了错误的 JSP 中，可以在屏幕上靠近发生错误的字段的位置显示有关特定 HTML 字段的错误消息，但是，要指示所有错误都属于同一类别，可以对属性参数指定以下常量：
ActionErrors.GLOBAL_ERROR

error
包含从属性文件派生的“键－字符串”对的键的 ActionError 对象。当配置 ActionError 对象时，最多可以包括要代入字符串中用来替代{0}、{1}等的四个值。

- 如果从验证方法返回了错误，则 ActionServlet 指导对在<action>元素的输入属性中指定的对象或 JSP 进行处理；在本例中，将处理 register.jsp.
- JSP register.jsp. 包括用于显示从验证方法派生的任何错误消息的以下标记：

<html:errors/>

如果在未发生输入错误的情况下调用 JSP，则该标记不显示任何内容，而在 JSP 中将继续进行处理。但是，如果因发生验证故障而调用了 JSP，则为用户显示的内容将受到属性文件中是否包括下列键的影响：

— errors.header，它导致在所有错误消息前面显示一个字符串；或者
— errors.footer，它导致在所有错误消息后面显示一个字符串；或者
— errors.hiader，和 errors.footer 两者

例如，在 ApplicationResources.properties 中，errors.header 和 errors.footer 的“键－字符串”对以及先前显示的这两个消息键为如下所示，它们各自都在单独的一行上（但是分成了多行显示以便于您复查）：

errors.header=

The Action failed because of the following reason(s):
<ul class=“errors”>

error.register.username=<li>you must enter a User ID.

error.register.balance=
Your account must start with a positive balance.
Errors.footer=

如果在注册时用户对用户名输入了空白，对余额输入零，则用户将接收到一个包括两个错误的列表的屏幕：

The Action failed because of the following reason(s):

o You must enter a User ID.

o Your account must start with a positive balance.

可以为标记<html:errors/>指定属性以支持国际化或者只显示有关指定了给定属性值的错误的信息。通过使用相异属性值，可以在相邻的不同字段中显示每条错误消息而不是将所有错误置于单个列表中。

使用异常处理声明

要定义应用程序的逻辑流程，成熟的经验是推荐在代码之外，用配置的方法来实现，而不是写死在程序代码中的。在J2EE中，这样的例子比比皆是。从实现EJB的安全性和事务性行为到描述JMS消息和目的地之间的关系，很多运行时的处理流程都是可以在程序之外定义的。

Struts 创建者从一开始就采用这种方法，通过配置Struts的配置文件来定制应用系统运行时的各个方面。这一点在版本1.1的新特性上得到延续，包括新的异常处理功能。在Struts framework以前的版本中，开发人员不得不自己处理Struts应用中发生的错误情况。在最新的版本中，情况大大的改观了，Struts Framework提供了内置的一个称为ExceptionHandler 的类， 用于系统缺省处理action类运行中产生的错误。这也是在上一个技巧中我们提到的framework许多可扩展接口之一。

Struts缺省的 ExceptionHandler类会生成一个ActionError对象并保存在适当的范围（scope）对象中。这样就允许JSP页面使用错误类来提醒用户出现什么问题。如果你认为这不能满足你的需求，那么可以很方便的实现你自己的ExcepionHandler类。

具体定制异常处理的方法和机制

要定制自己的异常处理机制，第一步是继承org.apache.struts.action.ExceptionHandler类。这个类有2个方法可以覆盖，一个是excute()另外一个是storeException(). 在多数情况下，只需要覆盖其中的excute()方法。下面是ExceptionHandler类的excute()方法声明：

```
public ActionForward execute( Exception ex,
                              ExceptionConfig exConfig,
                              ActionMapping mapping,
                              ActionForm formInstance,
                              HttpServletRequest request,
                              HttpServletResponse response
  ) throws ServletException;
```

正如你看到的，该方法有好几个参数，其中包括原始的异常。方法返回一个ActionForward对象，用于异常处理结束后将controller类带到请求必须转发的地方去。

当然您可以实现任何处理，但一般而言，我们必须检查抛出的异常,并针对该类型的异常进行特定的处理。缺省的，系统的异常处理功能是创建一个出错信息，同时把请求转发到配置文件中指定的地方去。 定制异常处理的一个常见的例子是处理嵌套异常。假设该异常包含有嵌套异常，这些嵌套异常又包含了其他异常，因此我们必须覆盖原来的execute()方法，对每个异常编写出错信息。

一旦你创建了自己的ExceptionHandler 类，就应该在Struts配置文件中的部分声明这个类，以便让Struts知道改用你自定义的异常处理取代缺省的异常处理.

可以配置你自己的ExceptionHandler 类是用于Action Mapping特定的部分还是所有的Action对象。如果是用于Action Mapping特定的部分就在元素中配置。如果想让这个类可用于所有的Action对象,可以在 元素中指定。例如，假设我们创建了异常处理类CustomizedExceptionHandler用于所有的Action类, 元素定义如下所示：

```
<global-exceptions>
  <exception

handler="com.cavanes
s.storefront.Customi
zedExceptionHandler"

key="global.error.me
ssage"
```

在元素中可以对很多属性进行设置。在本文中，最重要的属性莫过于handler属性,handler属性的值就是自定义的继承了ExceptionHandler类的子类的全名。假如该属性没有定义，Struts会采用自己的缺省值。当然，其他的属性也很重要，但如果想覆盖缺省的异常处理的话，handler无疑是最重要的属性。

最后必须指出的一点是，你可以有不同的异常处理类来处理不同的异常。在上面的例子中，CustomizedExceptionHandler用来处理任何java.lang.Exception的子类. 其实，你也可以定义多个异常处理类，每一个专门处理不同的异常树。下面的XML片断解释了如何配置以实现这一点。

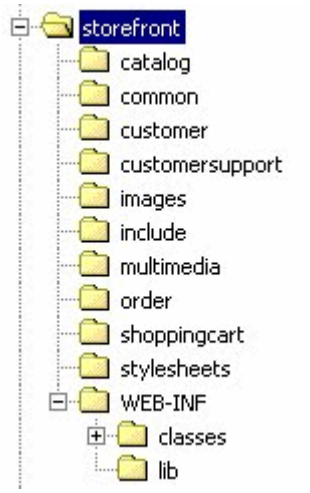
```
<global-exceptions>
  <exception

handler="com.cavanes
s.storefront.Customi
zedExceptionHandler"

key="global.error.me
ssage"
```

在这里，一旦有异常抛出，struts framework将试图在配置文件中找到ExceptionHandler，如果没有找到，那么struts将沿着该异常的父类链一层层往上找直到发现匹配的为止。因此，我们可以定义一个层次型的异常处理关系结构，在配置文件中已经体现了这一点。

使用应用模块（Application Modules）

Struts 1.1的一个新特性是应用模块的概念。应用模块允许将单个Struts应用划分成几个模块，每个模块有自己的Struts配置文件，JSP页面，Action等等。这个新特性是为了解决大中型的开发队伍抱怨最多的一个问题，即为了更好的支持并行开发允许多个配置文件而不是单个配置文件。

显然，当很多开发人员一起参加一个项目时，单个的Struts配置文件很容易引起资源冲突。应用模块允许Struts按照功能要求进行划分，许多情况已经证明这样更贴近实际。例如，假设我们要开发一个典型的商店应用程序。可以将组成部分划分成模块比如catalog（商品目录），customer（顾客），customer service（顾客服务），order（订单）等。每个模块可以分布到不同的目录下，这样各部分的资源很容易定位，有助于开发和部署。图1 显示了该应用的目录结构。



一个典型的商店应用程序的目录结构

注：如果你无需将项目划分成多个模块，Struts框架支持一个缺省的应用模块。这就使得应用程序也可以在1.0版本下创建，具有可移植性，因为应用程序会自动作为缺省的应用模块。

为了使用多应用模块功能，必须执行以下几个准备步骤：

- 为每个应用模块创建独立的Struts配置文件。
- **配置Web 部署描述符 Web.xml文件。**
- 使用org.apache.struts.actions.SwitchAction 来实现程序在模块之间的跳转.

创建独立的Struts配置文件

每个Struts应用模块必须拥有自己的配置文件。允许创建自己的独立于其他模块的Action，ActionForm，异常处理甚至更多。

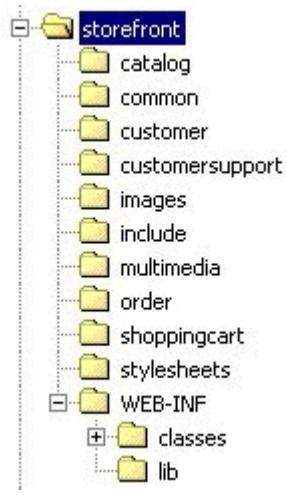
继续以上面的商店应用程序为例，我们可以创建以下的配置文件：一个文件名为struts-config-catalog.xml，包含catalog（商品目录）、items(商品清单)、和其它与库存相关的功能的配置信息；另一个文件名为struts- config-order.xml，包含对order（订单）和order tracking（订单跟踪）的设置。第三个配置文件是struts-config.xml,其中含有属于缺省的应用模块中的一般性的功能。

在为每个应用模块创建独立的配置文件之后，我们就有可能需要调用不同的模块中Action。为此必须使用Struts框架提供的SwitchAction类。Struts 会自动将应用模块的名字添加到URL,就如Struts 自动添加应用程序的名字加到URL一样。应用模块是对框架的一个新的扩充，有助于进行并行的团队开发。如果你的团队很小那就没必要用到这个特性，不必进行模块化。当然，就算是只有一个模块，系统还是一样的运作。

把 JSP 放到 WEB-INF 后以保护 JSP 源代码

为了更好地保护你的JSP避免未经授权的访问和窥视，一个好办法是将页面文件存放在Web应用的WEB-INF目录下。

通常JSP开发人员会把他们的页面文件存放在Web应用相应的子目录下。一个典型的商店应用程序的目录结构如图2所示。跟catalog （商品目录）相关的JSP被保存在catalog子目录下。跟customer相关的JSP，跟订单相关的JSP等都按照这种方法存放。



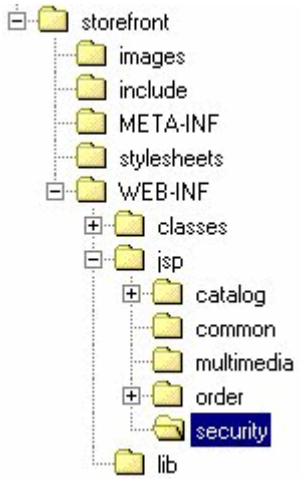
基于不同的功能 JSP 被放置在不同的目录下

这种方法的问题是这些页面文件容易被偷看到源代码，或被直接调用。某些场合下这可能不是个大问题，可是在特定情形中却可能构成安全隐患。用户可以绕过Struts的controller直接调用JSP同样也是个问题。

为了减少风险，可以把这些页面文件移到WEB-INF 目录下。基于Servlet的声明，WEB-INF

不作为Web应用的公共文档树的一部分。因此，WEB-INF 目录下的资源不是为客户直接服务的。我们仍然可以使用WEB-INF目录下的JSP页面来提供视图给客户，客户却不能直接请求访问JSP。

采用前面的例子，图3显示将JSP页面移到WEB-INF 目录下后的目录结构



JSP存放在 WEB-INF 目录下更为安全

如果把这些JSP页面文件移到WEB-INF 目录下，在调用页面的时候就必须把"WEB-INF"添加到URL中。例如，在一个Struts配置文件中为一个logoff action写一个Action mapping。其中JSP的路径必须以"WEB-INF"开头。如下所示：请注意粗体部分.

这个方法在任何情况下都不失为Struts实践中的一个好方法。是唯一要注意的技巧是你必须把JSP和一个Struts action联系起来。即使该Action只是一个很基本的很简单JSP，也总是要调用一个Action，再由它调用JSP。

最后要说明的是，并不是所有的容器都能支持这个特性。WebLogic早期的版本不能解释Servlet声明，因此无法提供支持，据报道在新版本中已经改进了。总之使用之前先检查一下你的Servlet容器。

使用 Prebuilt Action 类提升开发效率

Struts framework带有好几个prebuilt Action类，使用它们可以大大节省开发时间。其中最有用的是org.apache.struts.actions.ForwardAction 和 org.apache.struts.actions.DispatchAction.

使用 ForwardAction

在应用程序中，可能会经常出现只要将Action对象转发到某个JSP的情况。在上一点中曾提到总是由Action调用JSP是个好习惯。如果我们不必在Action中执行任何业务逻辑，却又想遵循从Action访问页面的话，就可以使用ForwardAction，它可以使你免去创建许多空的Action类。运用ForwardAction的好处是不必创建自己的Action类，你需要做的仅仅是在

Struts配置文件中配置一个Action mapping。

举个例子，假定你有一个JSP文件index.jsp ，而且不能直接调用该页面，必须让程序通过一个Action类调用，那么，你可以建立以下的Action mapping来实现这一点：

```
<action
  path="/logoff"
  type="org.apache.struts.webapp.example.LogoffAction">
  <forward name="success" path="/WEB-INF/jsp/index.jsp"/>
</action>
```

正如你看到的，当 /home 被调用时，就会调用ForwardAction 并把请求转发到index.jsp 页面.

再讨论一下不通过一个Action类直接转发到某个页面的情况，必须注意我们仍然使用元素中的forward属性来实现转发的目标。这时元素定义如下：

```
<action
   path="/home"
   forward="/index.jsp">
</action>
```

以上两种方法都可以节省你的时间，并有助于减少一个应用所需的文件数。

使用 DispatchAction

DispatchAction是Struts包含的另一个能大量节省开发时间的Action类。与其它Action类仅提供单个execute()方法实现单个业务不同，DispatchAction允许你在单个Action类中编写多个与业务相关的方法。这样可以减少Action类的数量，并且把相关的业务方法集合在一起使得维护起来更容易。

要使用DispatchAction的功能，需要自己创建一个类，通过继承抽象的DispatchAction得到。对每个要提供的业务方法必须有特定的方法signature。例如，我们想要提供一个方法来实现对购物车添加商品清单，创建了一个类ShoppingCartDispatchAction提供以下的方法：

```
public ActionForward addItem( ActionMapping mapping,
                              ActionForm form,
                              HttpServletRequest request,
                              HttpServletResponse response)
  throws Exception;
```

那么，这个类很可能还需要一个deleteItem()方法从客户的购物车中删除商品清单，还有clearCart()方法清除购物车等等。这时我们就可以把这些方法集合在单个Action类，不用为每个方法都提供一个Action类。

在调用ShoppingCartDispatchAction里的某个方法时，只需在URL中提供方法名作为参数值。就是说，调用addItem()方法的 URL看起来可能类似于：

http://myhost/storefront/action/cart?method=addItem

其中method参数指定ShoppingCartDispatchAction中要调用的方法。参数的名称可以任意配置,这里使用的"method"只是一个例子。参数的名称可以在Struts配置文件中自行设定。

Struts 标记库

定制 JSP 标记

Struts 提供了用来封装逻辑的各种定制 JSP 标记，因此页面设计者可以将主要精力花在页面的可视特征上，而不必主要考虑 Java 语法或其它 JSP 语法，在下列标识库描述符中引用了 Struts 标记：

Struts-bean.tld

使访问 bean 以及新 bean 的定义更容易,，为了实现国际化，应使用不同的属性文件

struts-html.tld

提供显示 HTML 对象（例如，表单、按钮和复选框）的简便方法

struts-logic.tld

支持逻辑构造，以便可以有条件地显示文本或者作为处理循环的结果来显示文本

struts-template.tld

支持使用在运行时可以修改的 JSP 模板

要在 JSP 文件顶部的＜taglib＞伪指令如下所示：

<%@ taglib uri="struts-html. tld" prefix="html"%>
<%@ taglib uri="struts-bean.tld"prefix="bean"%>
<%@ taglib uri="struts-logic.tld"prefix="logic"%>

每个＜taglib＞伪指令都具有与基于 web.xml 的＜ taglib＞标记中的 URL 相匹配的 URL。另外 JSP 中的每个 struts 标记都使用一个使标记与特定标记库描述符相关的前缀：

— 没有嵌套内容的标记可以采用以下格式：
<prefix:tagName attributesAndValues/>

— 嵌套内容是在一对标记之间嵌套的：
<prefix:tagName attributesAndValues />
</prefix:tagName>

prefix

在 JSP　taglib 伪指令中指定的前缀

tagName

标记的名称，如标记库描述符中所述；描述符条目指定提供标记逻辑的 Jave 类

attributesAndValues

— 系列属性与值的配对（是必需的或者是可选的），每个配对都包括一种属性、一个等号（没有前导或结尾空白）和一个引起来的字符串

文件 resource.jsp 包含 bean:message 标记的几个实例。以下是标记的示例用法:

<bean:message key="market. text. title"/>

资源束

在最简单的情况下，bean:message 标记解析为存储在根据属性文件创建的资源束中的字符串：

— 属性文件的名称是用来调用 ActoinServlet 的 web.xml "application"参数的值。如：

\WEB-INF\classes\ApplicationResources.properties

— 消息标记中的 key 属性指向属性文件中的"键－字符串"对；在本例中，指向下面的"键－字符串"对：

market. text.title=Current Market Conditions

可以采用各种方法来定制 bean:message 标记，以便（例如）JSP 在运行时引用不同的属性文件。标记提供了一种方法来支持多种语言以及最多将四个替代值插入到字符串中来代替｛0｝、｛1｝等等。

- 仅当指定的对象或值存在时，logic:present 标记才会导致显示嵌套的文本。在 register.jsp 中，仅当操作类创建了作为 tickerBean 引用（在任何作用域中）的 Java bean 时才为用户提供 HTML 表行。 Struts 标记为如下所示：

  <logic:present name="tickerBean">

  -->nested content for presentation<--

  </logic:present>

- Struts 标记允许很方便地访问 Java bean 内容。例如，以下标记将解析为存储在 tickerBean 中的值：

  <bean:write name=　"tickerBean"property="DJChange"/>

  tickerBean 的源代码在以下目录中：

  Trade\Java Source\tradeCommon\tickerBean.java

- HTML 表单与表单 bean 之间的数据传送是通过使用 html:form 和 html:text 标记来完成的。 register.jsp 中的输入表单是按如下所示构建的：

  <html:form action= "/register">

  -->nested form content with html: text tags<--

  </html:form action>

html:form 标记解析为 HTML FORM 标记并导致 html: text 标记引用适当的表单 bean；特别是在 path="/register"的 Struts 配置文件的 <action>标记中标识的表单 bean。

html:text 标记建立 HTML 输入字段。例如，以下标记确保在 HTML 输入字段与表单 bean 的用户名字段之间传送信息：

<html:text property="username"size="40"/>

JSP 视窗组件所使用的 struts 标记库由四类标记组成：

- Bean 标记：用来在 JSP 页中管理 bean
- 逻辑标记：用来在 JSP 页中控制流程
- HTML 标记：用来生成 HTML 标记，在表单中显示数据，使用会话 ID 对 URL 进行编程
- 模板标记：使用动态模板构造普通格式的页

Bean 标记

这个标记库中包含用于定义新 bean、访问 bean 及其属性的标记。Struts 框架提供了多种自定义标记用来在 JSP 页中处理 JavaBean。这些标记被封装在一个普通的标记库中，在文件 struts-bean.tld 中定义了它的标记库描述器。Bean 标记库将标记定义在四个子类别中：

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

Bean 复制标记

可定义新 bean，可复制现有 bean，还可从现有 bean 复制属性。

<bean:define>标记用来：

- 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>标记属性：

属性	描述
Id	新定义的 bean 脚本变量名称，必须设置
Type	定义引入脚本变量的类
Value	为 id 属性定义的脚本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：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大哦请求作用域中的另一个 bean:

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

定义脚本变量的标记

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，HTTP 标头等等。属性如下：

属性	描述
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

例如：

<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>标记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：

属性	描述
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

例如：

<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>标记将检索 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：

属性	描述
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。

显示 Bean 属性

标记库中定义了<bean:write>标记，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：

属性	**描述**
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。

消息标记和国际化

strtus 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。

用 strtus 实现国际化和本地化：

第一步要定义资源文件的名称,这个文件会包含用默认语言编写的在程序中会出现的所有消息。这些消息以“关键字-值”的形式存储，如下：

error.validation.location = The entered location is invalid

这个文件需要存储在类的路径下，而且它的路径要作为初始化参数传送给 ActionServlet 作为参数进行传递时，路径的格式要符合完整 Java 类的标准命名规范。比如，如果资源文件存储在 WEB-INF\classes 目录中，文件名是 ApplicationResources.properties，那么需要传递的参数值是 ApplicationResources。如果文件在 WEB-INF\classes\com\test 中，那么参数值就应该是 com.test.

ApplicationResources.

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

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

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

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

我们可以把字符串{0}替换成任何我们需要的数字。<bean:message>标签属性如下：

属性	描述
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}

我们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

逻辑标记

逻辑库的标记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记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记。这些标记封装在文件名为 struts-logic.tld 的标记包中。逻辑标记库定义的标记能够执行下列三个功能：

- 条件逻辑
- 重复
- 转发/重定向响应

条件逻辑

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记：

标记	功能
<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

属性	描述
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是”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。

第二类条件标记定义了两个标记：

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

它们的功能是在计算标记体之前判断特定的项目是否存在。标记的属性和属性值决定了要进行检查的项目。

属性	描述
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>
```

标记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。

第三类条件标记比较复杂，这些标记根据模板匹配的结果检查标记体的内容。换句话说，这些标记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>

这些标记允许 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：

属性	**描述**
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>
```

标记检查名为”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”）。

重复标记

在逻辑标记库中定义了<logic:iterate>标记，它能够根据特定集合中元素的数目对标记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。

属性	描述
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。

转发和重定向标记

转发标记

<logic:forward>标记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” />
```

重定向标记

<logic:redirect>标记是一个能够执行 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：

属性	描述
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

使用这个标记时至少要指定 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。

HTML 标记

Struts HTML 标记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

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<form>标记中使用的用来显示 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。

属性	**描述**
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	字段被鼠标点击
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点击
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：

属性	描述
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

表单标记

<html:form>标记用来显示 HTML 标记，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>标记能够包含与各种 HTML 输入字段相对应的子标记。

<html:form>标记属性如下：

属性	描述
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>标记属性进行显示的定义。

以下标记必须嵌套在<html:form>标记里

按钮和取消标记

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

属性	描述
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	按钮上的标记

复位和提交标记

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

文本和文本区标记

<html:text>和<html:textarea>标记分别 HTML 文本框和文本区，属性如下：

属性	描述
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>标记还有以下属性：

属性	描述
Maxlength	能够输入的最大字符数
Size	文本框的大小（字符数）

<html:textarea>标记特有的属性如下：

属性	描述
Rows	文本区的行数
Cols	文本区的列数

检查框和复选框标记

<html:checkbox>标记能够显示检查框控件。<html:multibox>标记能够显示 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：

属性	描述
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”.

文件标记

<html:file>标记可以显示 HTML 文件控件。属性如下：

属性	描述

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	按钮上的标记，这个按钮能够在本地文件系统中浏览文件

单选钮标记

<html:radio>标记用来显示 HTML 单选钮控件，属性如下：

属性	描述
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

隐藏标记

<html:hidden>标记能够显示 HTML 隐藏输入元素，属性如下：

属性	描述
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

密码标记

<html:password>标记能够显示 HTML 密码控件，属性如下：

属性	描述
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

选择标记

<html:select>标记能够显示 HTML 选择控件，属性如下：

属性	描述
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

选项标记(这个元素需要嵌套在<html:select>标记里)

<html:option>标记用来显示 HTML 选项元素集合，属性如下：

属性	描述
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)
labelProperty	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

我们看一下这个标记的一些例子：

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

标记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”/>

标记中 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。

显示错误信息的标记

<html:errors>标记能够与 ActionErrors 结合在一起来显示错误信息。这个标记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：

属性	描述
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。

其他 HTML 标记

struts HTML 标记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

这些标记的详细内容请参照 struts 文档。

模板标记

动态模板是模块化 WEB 页布局设计的强大手段。Struts 模板标记库定义了自定义标记来实现动态模板。

插入标记

<template:insert>标记能够在应用程序的 JSP 页中插入动态模板。这个标记只有一个 template 属性，用来定义模板 JSP 页。要插入到模板的页是有多个<template:put>标记来指定的，而这些标记被定义为<template:insert>标记的主体内容。

放置标记

<template:put>标记是<template:insert>标记内部使用的，用来指定插入到模板的资源。属性如下：

属性	描述
content	定义要插入的内容，比如一个 JSP 文件或一个 HTML 文件
direct	如果这个设置为 true，由 content 属性指定的内容将直接显示在 JSP 上而不是作为包含文件
Name	要插入的内容的名称
Role	如果设置了这个属性，只有在当前合法用户具有特定角色时才能进行内容的插入。

获得标记

在模板 JSP 页中使用<template:get>标记能够检索由<template:put>标记插入到 JSP 页的资源。属性如下：

属性	描述

Name	由<template:put>标记插入的内容的名称
Role	如果设置了这个属性，只有在当前合法用户具有特定角色时才能进行内容的检索

使用模板标记

首先编写一个模板 JSP 页，它将被所有的 web 页使用：

```
<html>
  <%@ taglib uri=”/template” prefix=”template” %>
  <head>
    <title></title>
  </head>
  <body>
    <table width=”100%” height=”100%” >
      <tr height=”10%”>
        <td>
          <template:get name=”header”/>
        </td>
      </tr>
      <tr height=”80%”>
        <td>
          <template:get name=”content”/>
        </td>
      </tr>
      <tr height=”10%”>
        <td>
          <template:get name=”footer”/>
        </td>
      </tr>
    </table>
  </body>
</html>
```

我们将这个文件命名为 template.jsp。这个文件使用<template:get>标记来获得由 JSP 页使用<template:put>标记提供的内容，并且将内容在一个 HTML 表格中显示出来。这三个内容是标题，内容和页脚。典型的内容 JSP 会是这样：

```
<%@ taglib uri=”/template” prefix=”/template” %>
<template:insert template=”template.jsp”>
  <template:put name=”header” content=”header.html”/>
  <template:put name=”content” content=”employeeList.jsp”/>
  <template:put name=”footer” content=”footer.html”/>
</template:insert>
```

这个应用程序 JSP 页使用<template:insert 标记来定义模板,然后使用<template:put>标记将特定内容名称指定的资源放到模板 JSP 页中。如果我们有上百个布局相同的页，但突然想改变这个模板，我们只需要改变 template.jsp 文件。

一个简单的示例

在这个指导中我们将 step by step 开发一个小的应用程序。你应该有一些 JSP 和 XML 的经验，并且有一个可以运行的应用服务器。

Struts 的安装

请先将 Struts.jar 和所有相关 common 拷贝到你应用程序的 lib 目录中，不用删除你的 struts 目录中的其他文件。结果如图 1 所示。

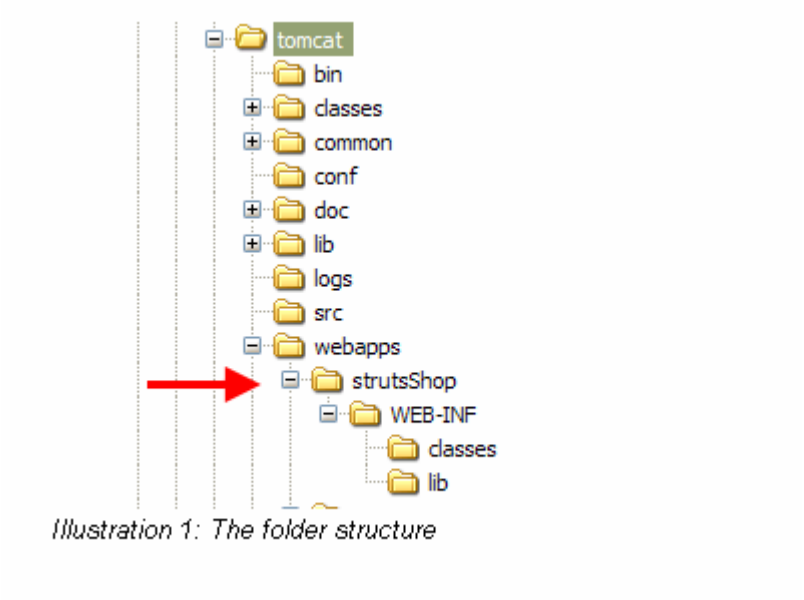


Illustration 1: The folder structure

第一个实验：简单的 JSP 页

现在我们要做一个简单的 JSP 页，用来确认至此我们的操作是正确的。

在 strutsShop 里建一个 BookView.jsp 的文件。内容如下，按图 2 所示在浏览器中运行：

First JSP

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

<html>
<head>
<title>Struts Tutorial: BookView</title>
</head>

<body bgcolor="white">
<h2>BookView</h2>

</body>
</html>
```

BookView

Illustration 2: Our First Page

第二个实验：struts 的国际化

我们将使用户能用本国的语言浏览预定义的文本，进而接触一些struts的功能。首先你要按照图 3 拷贝一些文件到WEB-INF目录下。在struts-html.tld文件里有我们要用的标签。这些我们在上接已经介绍了，你可以简短的回顾一下。

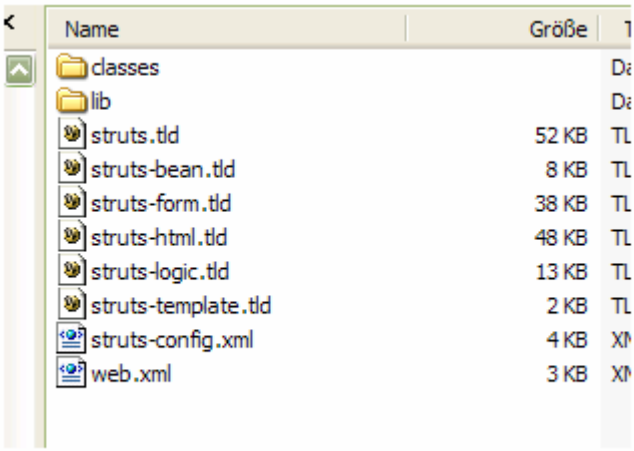


Illustration 3: Needed LTD files

struts-config.xml 的配置清单如下，这是一个标准的清单，你以后的程序都可以以此为基础进行扩展：

Empty struts-config.xml

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE struts-config PUBLIC
          "-//Apache Software Foundation//DTD Struts Configuration 1.0//EN"
          "http://jakarta.apache.org/struts/dtds/struts-config_1_0.dtd">
<struts-config></struts-config>
```

web.xml 的配置清单如下，这是一个标准的清单，你以后的程序都可以以此为基础进行扩展：

web.xml

```
<?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>

<servlet>
  <servlet-name>action</servlet-name>
  <servlet-class>org.apache.struts.action.ActionServlet</servlet-class>
  <init-param>
    <param-name>application</param-name>
    <param-value>ApplicationResources</param-value>
  </init-param>
    <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>
      <init-param>
        <param-name>validate</param-name>
        <param-value>true</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 Welcome File List -->
    <welcome-file-list>
      <welcome-file>index.jsp</welcome-file>
      <welcome-file>index.html</welcome-file>
    </welcome-file-list>

    <!-- Struts Tag Library Descriptor -->
  <taglib>
      <taglib-uri>
          /WEB-INF/struts-bean.tld
      </taglib-uri>
```

```
      <taglib-location>
          /WEB-INF/struts-bean.tld
      </taglib-location>
  </taglib>
  <taglib>
      <taglib-uri>
          /WEB-INF/struts-html.tld
      </taglib-uri>
      <taglib-location>
          /WEB-INF/struts-html.tld
      </taglib-location>
  </taglib>
  <taglib>
      <taglib-uri>
          /WEB-INF/struts-logic.tld
      </taglib-uri>
      <taglib-location>
          /WEB-INF/struts-logic.tld
      </taglib-location>
  </taglib>
  <taglib>
      <taglib-uri>
          /WEB-INF/struts-template.tld
      </taglib-uri>
      <taglib-location>
          /WEB-INF/struts-template.tld
      </taglib-location>
```

```
  </taglib>

  </web-app>
```

在 classes 目录下创建一个 ApplicationResources.properties 的文件(此文件名在 web.xml 中定义)，打开它，输入一行：index.title=Struts Tutorial。然后在创建一个 ApplicationResources_de.properties 文件，也输入一行：index.title=Struts Einführung。其实这两个文件就是当加载时会根据当前的浏览器而选择英文或德文，这里我们只能了解国际化过程来测试前者了。我们还需编写 BookView.jsp 文件，如下：

BookView.jsp: Introducing Internationalization

```
<%@ page language="java" %>
<%@ taglib uri="/WEB-INF/struts-bean.tld"  prefix="bean" %>
<%@ taglib uri="/WEB-INF/struts-html.tld"  prefix="html" %>
<%@ taglib uri="/WEB-INF/struts-logic.tld" prefix="logic" %>

<html:html locale="true">
<head>
<html:base/>
<title>
<bean:message key="index.title"/>
</title>
</head>
<body>Test Text</body>
</html:html>
```

可在浏览器中浏览，你可能需要重启你的 tomcat：

Illustration 4: Displaying the German title element

其实我们可以将国际化写在一个文件中，就是将参数写在一个属性文件中。（这是作者的意图，但我没有各种版本的 IE 进行实验）

struts 中的 Forms

在这一章我们将创建一个简单的 Bean(Book.java)和两个 JSP 页面，一个是创建新书的，另一个是显示它的，我们也会第一次使用 struts-config.xml 文件。

我们先在你的 classes 目录下创建如下 Book.java 文件。

Book.java

```
import java.util.Vector;

/*
    A simple book.
    @author stephan@stephanwiesner.de
*/
public class Book
{
    /** The title */
    private String title = "";
    /** We can have more than one author */
    private Vector authors = new Vector();
    /** The number of pages the book has */
    private int pages = 0;


    /** Standard constructor. */
    public Book()
    { }

    /** @param title The new Title   */
    public void setTitle(String title)
    {  this.title = title;   }

    /** @return The title. */
    public String getTitle()
    { return this.title; }


    /** @param pages The new number of pages. */
    public void setPages(int pages)
    { this.pages = pages; }


    /** @return The number of pages. */
    public int getPages()
    { return this.pages; }

    /**
        We don't want to work with the Vector here, as it is
        only a reference we would get!
        @param author Add another author
    */
    public void addAuthor(String author)
    { this.authors.add(author); }


    /**
        Pay attention not to use the wrong number.
        @param position The number of the author to remove.
    */
```

```
    public void removeAuthor(int position)
    { this.authors.remove(position); }

    /** @return The number of authors the book has. */
    public int getNumberOfAuthors()
    { return this.authors.size(); }
}
```

我们还需要创建新书的 JSP 页。我们将使用 title,auther 和 number of pages 三个字段，在此之前我们先要做一些工作，对于一个初学者这将有些难度。在你的 BookView 中加上以下内容：

```
        <html:form action="createBook" method="GET">
            Title:<html:text property="title" /> <br/>
            <html:submit property="submit"/>
        </html:form>
```

再次运行，你将得到图 5 所示内容，如果没有错，那你需要重启 tomcat；如果错误不一样，没关系，因为他都是没有在配置文件中找到 mapping 路径。

Illustration 5: Nice Errormessages are always close at hand

接下来我们需要第二个 JSP 页面 CreateBook.jsp,代码如下：

```
<%@ page language="java" %>
<%@ taglib uri="/WEB-INF/struts-bean.tld"  prefix="bean" %>
<%@ taglib uri="/WEB-INF/struts-html.tld"  prefix="html" %>
<%@ taglib uri="/WEB-INF/struts-logic.tld" prefix="logic" %>

<html:html locale="true">
<head>
   <html:base/>
   <title><bean:message key="index.title"/></title>
</head>

<body bgcolor="white">
<h2>Create a book</h2>


<html:errors/>

<html:form action="createBook.do" method="GET">
    Title:<html:text property="title" /> <br/>
    <html:submit property="submit"/>
</html:form>

</body>
</html:html>
```

在 classes 目录下创建一个 BookAction 文件：

BookAction.java

```
import javax.servlet.http.*;
import org.apache.struts.action.*;


/*
    The action for the creation of a book.
    @author stephan@stephanwiesner.de
*/
public final class BookAction extends Action
{

    /**
      @param mapping The ActionMapping used to select this instance
      @param form The optional ActionForm bean for this request (if any)
      @param req The non-HTTP request we are processing
      @param res The non-HTTP response we are creating
      @return Return an ActionForward instance describing where and how
          control should be forwarded, or null if the response has already
          been completed.
    */
    public ActionForward perform(ActionMapping mapping,
        ActionForm form,  HttpServletRequest req,
        HttpServletResponse res)
    {
        System.out.println("Start perform(" + form + ") . . ." );
        String title = req.getParameter("title");
        Book book = new Book();
        book.setTitle( title );
        System.out.println("After creation of book: " + book.getTitle() );

        req.setAttribute("BOOK", book);
        return mapping.findForward("bookCreated");
    }
}
```

它没按照 struts 要求编写仅仅创建一本书并给它标题。然后编写你的 struts-config.xml：

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

<!DOCTYPE struts-config PUBLIC
          "-//Apache Software Foundation//DTD Struts Configuration 1.0//EN"
          "http://jakarta.apache.org/struts/dtds/struts-config_1_0.dtd">


<struts-config>
  <form-beans>
    <form-bean name="bookForm" type="Book"/>
  </form-beans>

  <global-forwards>
    <forward   name="bookCreated"  path="/BookView.jsp"/>
  </global-forwards>

  <action-mappings>
    <action    path="/createBook"
               type="BookAction"
               name="bookForm"
               scope="request"
               input="/CreateBook.jsp">
    </action>
  </action-mappings>

</struts-config>
```

我们希望在 struts 中在 bookForm 和 Book 间建立连接。而且我们还用 bookCreated 定义了一个到 BookView.jsp 的转发。最后我们用 action=createBook.do 属性定义了我们的 form 做什么。关于 do:与接受 CreateBook.jsp 输入信息的 bookForm 相关的 bean，由 createBook 命令创建。

按图 6 编译你的类。由于我是初手，在这里遇见很多问题，所以耽误了很久，不过它让你学到很多东东。比如：javax.servlet 要用到 servlet.jar 包；javac 后跟 a.java， 而 java 后跟 a；还有我遇见了很原文提到的问题，很多都是由于自己编写(没有 copy)而造成的马虎。希望大家也能引起注意，到此除了 ActionForm 我们都已用到了。

Illustration 6: Using Struts.jar locally.

编译成功后，在你的 classes 目录下会增加两个文件：Book.class 和 BookAction.class.

重启你的 tomcat(每次改动 config 文件你都需要重启，改动注册表需要重启机器)。现在在你的浏览器里登陆 CreateBook.jsp，如图 7:

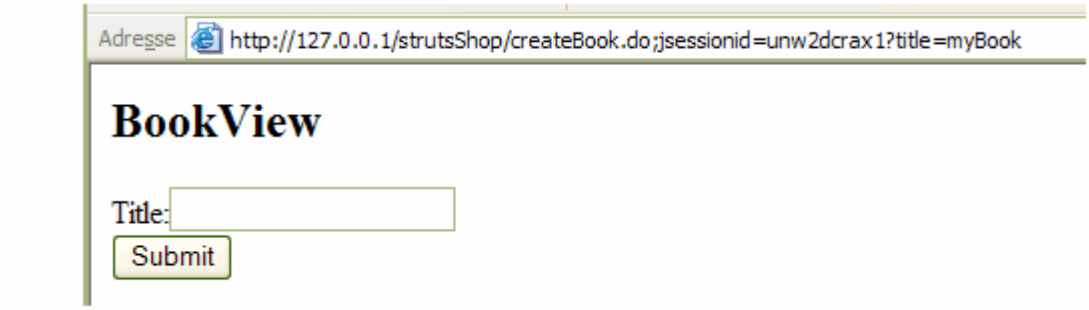


Illustration 7: Where is the book?

当你填写后提交，另你失望的是什么也没有得到。这是由于我们并没有 ActionForm bean.

struts:介绍ActionForm

这节我们继续做 ActionForm bean 来完成我们的例子。我们将用 ActionForm 得到合法的 book 的信息，并进行一些检验，例如没有输入标题等，向用户提供错误或成功的信息。我们还将得到 book 的属性并且能够更改它。

为此我们需要一个 ActionForm：它仅是一个简单的容器，没有应用程序逻辑，只有两个方法：reset(),validate().在 struts1.1 里，validate()方法被单独作为一个 validate.xml 文件。详见参考资料二。

现在我们做另一个类：BookForm.java。它将包含 book 的实例，并且有一些 getXXX 和 setXXX 的方法来访问它。关于内部的方法可看 struts 架构介绍。

BookForm.java

```
import javax.servlet.http.HttpServletRequest;
import org.apache.struts.action.*;

/*
    <b>ActionForm</b>
*/
public class BookForm extends ActionForm
{
    private Book book =new Book();
    String title = "Ye old Book";

    public void setTitle(String title)
    {
        book.setTitle(title);
    }
    public String getTitle()
    {
        return book.getTitle();
    }


    public void setBook(Book book) { this.book = book; }
    public Book getBook() { return this.book; }
```

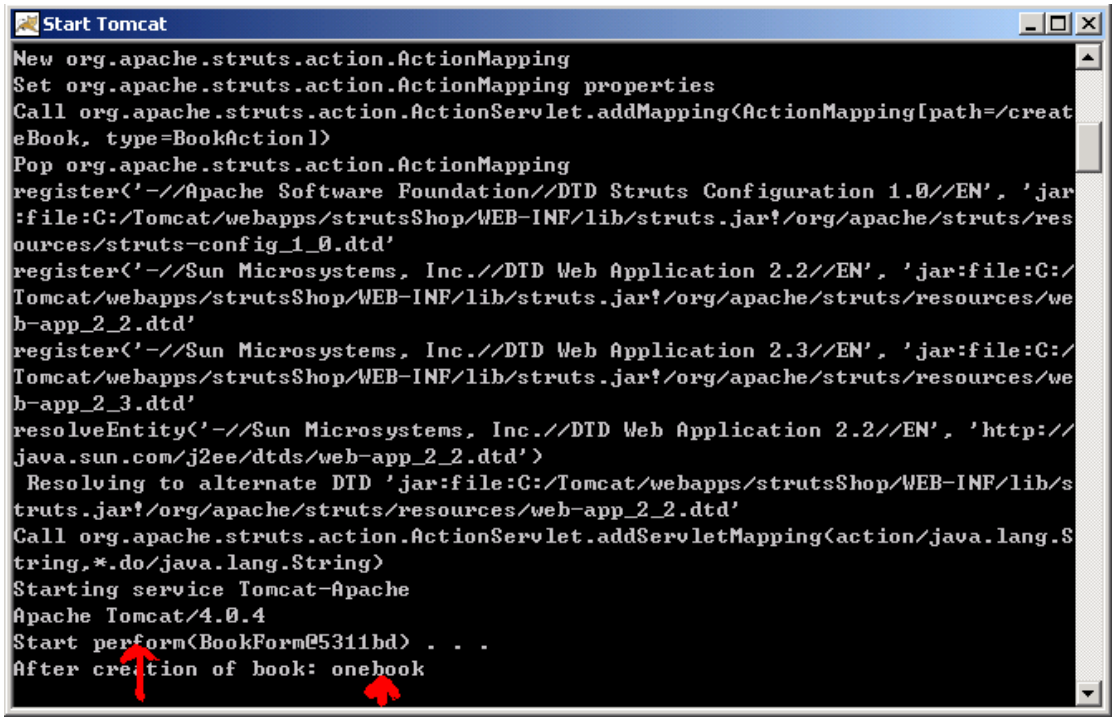
```
    /**
     * Reset all properties to their default values.
     *
     * @param mapping The mapping used to select this instance
     * @param request The servlet request we are processing
     */
    public void reset(ActionMapping mapping, HttpServletRequest request)
    {
        this.book = new Book();
    }



    public ActionErrors validate(ActionMapping mapping,
        HttpServletRequest request)
    {
        ActionErrors errors = new ActionErrors();
        if ((book.getTitle() == null) || (book.getTitle().length() < 3))
           { errors.add("Title", new ActionError("error.book.title")); }
        return errors;
    }
}
```

我们还要做一些额外的工作。去看struts-config.xml文件，我们需要用这个新类与Form关联，从而替代Book.java。因此我们必须改变form-beans: **<form-bean name="bookForm" type="BookForm"/>**.

另外我们还有定义一下当错误发生时的信息，在你的配置文件中输入：**error.book.title=Error**

现在编译你的类，重新启动 tomcat,重新登陆 CreateBook.jsp.，输入 onebook，你将在 tomcat 的 dos 窗口看见如下图所示：

你也可以在重输入其他的 title，看看 tomcat 的 dos 窗口有什么变化。



分离 Book 和 BookForm 的一个好方法

上一节我们使用一个新类 BookForm.java 去访问 Book.java,而不用 struts 直接连接到 Book.java。接下来，我们要解决在这两个类中重复输入 getXXX 和 setXXX。Struts 允许我们直接访问实例的方法。这很容易，但需要理解。我们给出改变的 CreateBook.jsp 的代码：

```
<html:form action="createBook.do" method="GET">
    Title:<html:text property="book.title" /> <br/>
    Pages:<html:text property="book.pages" /> <br/>
    <html:submit property="submit"/>
</html:form>
```

正如你看到的，除了将 title 改为 book.title，我们什么也没做。现在你可以去掉 BookForm.java 中的 getXXX 和 setXXX 方法了，如下图：（别忘记了也更改 BookView.jsp，否则在你的 tomcat 窗口下 After creation of book : null）

```
import javax.servlet.http.HttpServletRequest;
import org.apache.struts.action.*;

/*
    <b>ActionForm</b>
*/
public class BookForm extends ActionForm
{
    private Book book = new Book();

    public void setBook(Book book) { this.book = book; }
    public Book getBook() { return this.book; }

    /**
     * Reset all properties to their default values.
     *
     * @param mapping The mapping used to select this instance
     * @param request The servlet request we are processing
     */
    public void reset(ActionMapping mapping, HttpServletRequest request)
    {
        this.book = new Book();
    }

    public ActionErrors validate(ActionMapping mapping,
        HttpServletRequest request)
    {
        ActionErrors errors = new ActionErrors();
        if ((book.getTitle() == null) || (book.getTitle().length() < 3))
           { errors.add("Title", new ActionError("error.book.title")); }

        if (book.getPages() < 1)
           { errors.add("Page", new ActionError("error.book.page")); }
        return errors;
    }
}
```

你还应该改动BookAction.java，把String title = req.getParameter("title");中的title改为book.title,然后重新编译，这样你的tomcat下的After creation of book :none中的none就会出现你输入的名了。