用户指南

用于 Struts 1.3.10

南磊 译

目录

关于 Struts 框架

Struts 是 Apache Software Foundation（阿帕奇软件基金会）的一个开源项目。它是一个基于标准技术（比如 Servlet，JavaBean，ResourceBundle，XML 还有 Apache Commons 组件）的灵活的控制层。Struts 框架可以帮助你的应用程序创建基于发布标准和成熟设计模式的可扩展开发环境。

Struts 提供了它自己的 Web 控制（Controller）组件，并且整合了其它技术来提供模型（Model）和视图（View）。对于模型来说，Struts 可以和标准的数据访问技术（比如 JDBC 和 EJB）和第三方开发包（比如 Hibernate，iBatis 或其它对象关系框架）来交互。对于视图，Struts 和 JSP（Java Server Pages）技术良好协作，也包含 JSTL，JSF，Velocity Template，XSLT 和其它视图展现系统。

Struts 控制器担当了沟通应用程序模型和 Web 视图之间的桥梁。当收到请求时，控制器可以调用 Action 类。Action 类和模型（或者最好是一个代表模型的门面）来咨询沟通来检查或更新应用程序的状态。

大多数时候，模型被表示为 JavaBean 的集合。典型的情况是，开发人员会使用 Apache Commons 组件的 BeanUtils 来转换 ActionForm 和模型对象（或门面）之间的数据。最好的情况是，模型自己处理这个“繁重”的任务，而 Action 类担当“交通警察”或适配器的角色。

Struts 是每个项目的最佳选择吗？

不。如果你只需要用极少数的页面来编写一个简单的应用程序，那么你可以选择“模型 1”的解决方案来处理。

但是，如果你要编写一个很复杂的应用程序，它有很多页面，也需要在日后来维护，那么 Struts 就可以帮助你。要了解模型 1 或 MVC/模型 2 对你来说是正确的选择，可以阅读理解 JSP 和模型 2 架构。

关于翻译

翻译时，参考官网的用户指南，并有机地结合 FAQ 和 HOWTO 部分到中文版中，力求做到涵盖全面。

前言 关键技术

"The time has come," the Walrus said, "To talk of many things: Of shoes -- and ships -- and sealing-wax -- Of cabbages -- and kings -- And why the sea is boiling hot -- And whether pigs have wings."

“现在是时候了”，海象说，“要谈论的事情有很多：比如鞋子，船，密封的蜡，白菜-国王，为什么海是滚烫的，还有猪会长翅膀吗。”

0.1 常规做法

这篇用户指南是为动态 Web 开发人员所写，并假设读者已经有 Java Web 应用程序开发的基础知识。在开始之前，你应该理解一些关键技术的基础，比如：

- HTTP，HTML 和用户代理（0.2 节内容）
- HTTP 请求/响应周期（0.3 节内容）
- Java 语言和应用程序框架（0.4 节内容）
- JavaBean（0.5 节内容）
- 属性文件和资源束（0.6 节内容）
- Java Servlet（0.7 节内容）
- JavaServer Pages 和 JSP 标签库（0.8 节内容）
- 可扩展标记语言（0.9 节内容）
- JAAS（0.10 节内容）
- 其它层面（0.11 节内容）

本章是定义这些技术的简要介绍，而不是详细地描述它们。为了你的方便，如果你想了解这些技术的详细内容，其链接是提供的

如果你对 Java 很属性，而不是这些技术，最佳的整体出发点是 Sun 的 J2EE 教程。这个教程也可以 PDF 格式文档下载。

如果你为其它平台创建过 Web 应用程序，你可能在需要的时候会访问其参考文档。Struts 框架使用的核心技术也被其它大多数 Java Web 开发产品所使用，所以背景信息在任何 Java 项目中都是有用的。

如果你不熟悉 Java 语言，那么最佳的起点就是 Java 教程。这和 J2EE 教程在某些地方会有重叠，但是这两份教程却是形影不离的。

要获得构建普通 Java 应用程序的更多内容，可以参考 New to Java Center。

0.2 HTTP，HTML 和用户代理

World Wide Web（万维网）是基于 Hypertext Transfer Protocol（HTTP，超文本传输协议）和 Hypertext Markup Language（HTML，超文本标记语言）之上构建的。用户代理，比如 Web 浏览器，使用 HTTP 请求 HTML 文档。之后，浏览器格式化并显示文档给它的用户。HTTP 的传输不局限于 HTML，但是 HTML 是 Web 和 Web 应用程序的通用语言。

在构建 Web 应用程序时，一些 Java 开发人员编写他们自己的 HTML。而有一些开发人员将这个责任留给页面设计人员。

要了解更多关于 HTTP，HTML 和用户代理的内容，请参考：

- HTML 入门（by Dave Raggett）
- HTTP 概述（J2EE 教程）
- HTTP/1.1 规范
- HTTP 基本的和摘要认证规范
- 状态管理机制规范（Cookie）

0.3 HTTP 请求/响应周期

Web 开发人员要了解 HTTP 协议的一个很重要部分就是请求/响应周期。要使用 HTTP 就要生成请求。HTTP 服务器，就像 Web 服务器一样，之后负责响应。当你构建 Web 应用程序时，你设计它来对 HTTP 请求做出反应，之后返回 HTTP 响应。

框架则抽象出了这些大部分的螺母和螺丝（抽象的比喻），而理解幕后所发生的事情是很重要的。

如果你对 HTTP 请求/响应周期不熟悉，我们强烈建议您先阅读 J2EE 教程中的 HTTP 概述。

0.4 Java 语言和应用程序框架

Struts 框架由流行的 Java 编程语言来编写。Java 是面向对象的语言，而 Struts 框架使用了很多面向对象的技术。此外，Java 原生支持*线程*的概念，这就允许了多于一个任务同时执行。能更好的理解 Java，特别是面向对象编程（OOP）和线程，将会帮助你从 Struts 框架和本指南中获益更多。

要了解更多关于 Java 和线程方面的内容，可以参考：

- 学习 Java 语言（Sun 的 Java 教程）
- 线程：同时做两个以上的工作（Sun 的 Java 教程）

即使之前你使用过 Java 语言和 OOP，它们也可以帮助你了解具体的编程挑战来创建和使用应用程序框架。要了解更多关于应用程序框架的内容，可以阅读经典白皮书：

- 设计可重用的类（by Ralph E. Johnson & Brian Foote）
- 面向对象应用程序框架（by Mohamed Fayad & Douglas C. Schmidt）

如果你正调查或审查服务器端框架，这些文献就会有特别的用处。

0.4.1 责任链

组织执行复杂处理流程的流行技术就是“责任链”设计模式，正如经典的“Gang of Four”设计模式书中所描述的那样。GoF 总结了责任链模式是“每次给出多于一个对象到处理请求的程序时，避免耦合请求发出者到它的接受者。链接接收对象并沿着链传递请求直到有一个对象处理它。”

CoR 模式帮助我们保持软件组件的松耦合。组件可以调用责任链，而不需要知道什么对象在链子上或它们是怎么实现的。更重要的是，我们可以调整链子而不需要改变调用者是如何调用链子的。在 1.3 版本中，默认的请求处理器，作为 Struts 框架的“核心”，就是责任链。

要实现这个链，请求处理器使用 Jakarta Commons 组件中的责任链组件，它提供了标准

的 CoR 模式实现，还有各种由责任链使用的上下文和命令对象的实现，来服务于请求。

要获取更多关于责任链的信息，可以参考：

- Jakarta Commons 的责任链
- Commons 的责任链一览
- Struts 1.3 更好的编码

0.5 JavaBean

像很多 Java 应用程序一样，很多框架对象被设计成 JavaBean。遵循 JavaBean 设计模式可以使得框架的类更容易使用，这对于 Java 开发人员和开发工具都是一样的。

尽管 JavaBean 最初是为可视化元素而创建的，这些对象设计模式在任意可重用的组件中被发现也是很有用的，就像被框架使用的那样。

要了解更多关于 JavaBean 的内容，可以参考：

- `java.sun.com` 网站的 JavaBean 组件架构文档页面，包括下载 JavaBean 1.01 规范的链接。
- JavaBean Trail（Sun 的 Java 教程）
- JSP 页面中的 JavaBean 组件（Sun 的 J2EE 教程）

0.5.1 反射和自我检查机制

反射机制是决定哪些成员字段和方法在对象中是可用的过程。自我检查机制是使用 JavaBean 的 API 进行反射的特殊形式。使用自我检查，我们可以决定 JavaBean 的哪些方法可以被其它对象访问。（比如 getter 方法和 setter 方法）

Struts 框架使用自我检查机制来转换 HTTP 参数到 JavaBean 的属性，并从 JavaBean 属性中填充 HTML 的字段。这个技术使得在 HTML 表单和 JavaBean 之间“往返”填充属性就变得十分容易。

要获取更多关于反射自我检查机制的内容，可以参考：

- Reflection Trail
- JavaBean API 规范的第八章

0.5.2 Map

JavaBean 存储数据作为属性，并可以通过其它方法作用于数据。JavaBean 是很灵活，很强大的对象，但是不能仅仅作为存储数据的对象来编程。另外一个流行的对象是 Map[`java.util.Map`]。Map 是简单的名-值对集合。Map 通常被用来“在幕后”作为存储动态数据的一种灵活的方式。

0.5.3 DynaBean

DynaBean 结合了 JavaBean 的可扩展性和 Map 的灵活性。定义即使是最简单的 JavaBean 也需要定义一个新的类并编写属性，并为每个属性编写两个方法。DynaBean 的属性可以通过 XML 描述符来配置。DynaBean 的虚拟属性不能被标准的 Java 方法所调用，但是可以和依

赖反射和自我检查机制的组件良好地协同工作。

在应用程序中，你可以使用 DynaBean 来描述 HTML 表单。这个策略可以避免创建常规的 JavaBean 子类来存储很少很简单的属性。

要获取更多关于 DynaBean 的内容，可以参考：

- Commons 的 BeanUtils 组件包描述和 Javadoc 文档
- Struts FormDef

0.6 属性文件和资源束

Java 应用程序，包括 Web 应用程序，通常是使用属性文件来配置的。属性文件是资源束的基础内容，框架可以使用来给应用程序提供消息资源。

要获取更多关于属性文件的内容，可以参考：

- 使用属性文件来管理程序属性（Sun 的 Java 教程）

Java 的资源束使用一个或多个属性文件提供国际化消息给用户，这是基于用户的语言环境 Locale 实现的。支持应用程序的本地化从底层的行为开始构建框架。

要了解更多关于本地化和资源束的内容，可以参考：

- 关于资源束类（Sun 的 Java 教程）

0.7 Java Servlet

因为 Java 是面向对象的语言，Java Servlet 平台努力转换 HTTP 到面向对象的形式。这个策略使得 Java 开发人员集中精力去解决应用程序要做什么就很容易了，而不是 HTTP 的运行机制。

HTTP 为被称为通用网管接口（Common Gateway Interface，CGI）的扩展服务器提供了标准的机制。服务器可以传递请求到 CGI 程序中，之后程序会传递回响应。同样，Java 服务器可以传递请求到 Servlet 容器中。容器可以填充请求或将请求传回 HTTP 服务器。容器通过检查它自己的 Servlet 列表来决定是否可以处理请求。若有一个 Servlet 为该请求所注册，容器就传递请求给 Servlet。

当有请求传入时，容器检查是否有为该请求注册的 Servlet。如果可以匹配上一个，请求就传给该 Servlet。如果没有找到，请求就返回给 HTTP 服务器。

容器的工作就是管理 Servlet 的生命周期。容器创建 Servlet，调用 Servlet 最后再处理掉 Servlet。

Servlet 通常是 `javax.servlet.http.HttpServlet` 的子类。Servlet 必须实现四个方法，这是由容器调用所需的：

- **public void init(ServletConfig config)**-当 Servlet 实例第一次被创建，并在任何请求被处理之前，由 Servlet 容器调用。
- **public void doGet(HttpServletRequest request, HttpServletResponse response)**-被调用来处理使用 HTTP 的 `GET` 协议收到的特定请求，这会生成一个对应的动态响应。
- **public void doPost(HttpServletRequest request, HttpServletResponse response)**- 被调用来处理使用 HTTP 的 `POST` 协议收到的特定请求，这会生成一个对应的动态响应。
- **public void destroy()**-当 Servlet 实例不再被服务所需时由 Servlet 容器调用，比如当 Web 应用程序取消部署或当整个容器被关闭时。

Struts 框架为应用程序提供一个准备可用的 Servlet[`org.apache.struts.action.ActionServlet`]。作为开发人员，只需编写 ActionServlet 需要的对象即可。但是理解 Servlet 要领和它们在 Java Web 应用程序中所扮演的角色也是很有用的。

要获取更多关于 Java Servlet 的相关内容，可以参考：

- `java.sun.com` 的 Java Servlet 技术页
- `java.sun.com` 的 Servlet 2.3 和 2.4 规范下载页
- J2EE 教程的 Java Servlet 技术

0.7.1 Servlet 和线程

为了提高性能，容器可以执行多线程 Servlet。仅有一个特定的 Servlet 实例被创建，而这个 Servlet 的每个请求都经过相同的对象。这个策略帮助容器可以更好的使用可用资源。这个权衡是 Servlet 的 `doGet()` 和 `doPost()` 方法必须以*线程安全*的方式来编程。

要获取更多关于 Servlet 和线程安全的信息，可以参考：

- 控制并发访问共享资源（Sun 的 J2EE 教程）

0.7.2 Servlet 上下文

ServletContext 接口[`javax.servlet.ServletContext`]定义了当 Servlet 运行时，其 Web 应用程序的观点。可以在 Servlet 中通过 `getServletConfig()` 方法来访问，而在 JSP 页面中就是内置变量 `application`。Servlet 上下文提供一些 API，在构建 Web 应用程序时是很有用的：

- *访问 Web 应用程序资源*-Servlet 可以使用 `getResource()` 和 `getResourceAsStream()` 方法来访问 Web 应用程序中静态资源文件。
- *Servlet 上下文属性*-上下文开辟了 Java 对象的存储空间，由字符串值的键来标识。这些属性是对整个 Web 应用程序来说是全局的，可以在 Servlet 中使用 `getAttribute()`，`getAttributeNames()`，`removeAttribute()` 和 `setAttribute()` 方法来访问。而对于 JSP 页面，Servlet 上下文属性也就是我们所知的“应用范围的 bean”。

要获取更多关于 Servlet 上下文的内容，可以参考：

- 访问 Web 上下文（Sun 的 J2EE 教程）

0.7.3 Servlet 请求

每个由 Servlet 处理的请求由一个 Java 接口所代表，通常是 `HttpServletRequest` [`javax.servlet.http.HttpServletRequest`]。request 接口提供了面向对象的机制来访问在底层 HTTP 请求中包含的所有信息，包括：

- *Cookie*-这个请求中包含的 Cookie 的集合，可以通过 `getCookies()` 方法来访问。
- *Header*-HTTP 包含在该请求中的 HTTP 头部信息，可以通过其名称来访问。也可以枚举所有包含的头部信息的名称。
- Parameter-请求参数，包含从 URL 的查询字符串部分和从嵌入在请求体（仅为 POST

方式提交）中的内容，可以通过名称来访问。

- *请求特点*-很多收到的 HTTP 请求的其它特点，比如使用的方法（通常是 GET 或 POST），使用的协议模式（“http”或“https”），还有相似的值。
- *请求 URI 信息*-处理的原始请求 URI 可以通过 `getRequestURI()` 方法来访问。此外，Servlet 容器传递的请求 URI 的组成部分（contextPath，servletPath 和 pathInfo）可以分别被访问。

用户信息-如果你使用了容器安全管理（可以参考 0.7.10 节内容），你可以查询认证用户的用户名，检索代表当前用户的 `Principal` 对象，还有当前用户是否对特殊角色进行了认证。

此外，Servlet 请求支持请求属性（对于 JSP 而言，这就是“请求范围的 bean”），类似于上面所描述的 Serlvet 上下文属性。请求属性通常可以用来从业务逻辑类来沟通状态信息，生成到视图组件（比如 JSP 页面）中，这会使用信息来产生对应的响应。

Servlet 容器保证了特定请求会被一个单独的 Servlet 线程来处理。因此，通常不用去担心访问请求属性的线程安全问题。

要获取更多关于 Servlet 请求的内容，可以参考：

- 从请求中获取信息（Sun 的 J2EE 指南）

0.7.4 Servlet 响应

Servlet 的一个主要目的是处理收到的 Servlet 请求[`javax.servlet.http.HttpServletRequest`]并转换它到对应的响应。这是调用 Servlet 响应接口[`javax.servlet.http.HttpServletResponse`]的合适方法来执行的。可用的方法允许你：

- *设置头部信息*-你可以设置将包含在响应中的 HTTP 头部信息。最重要的头部信息就是 `Content-Type` 头部，这会告诉客户端包含在响应体中的是什么样的信息。对于 HTML 页面的典型的设置是 `text/html`，对于 XML 文档的就是 `text/xml`。
- *设置 Cookie*-你可以在当前响应中添加 Cookie。
- *发送错误响应*-你可以使用 `sendError()` 方法发送 HTTP 错误状态（代替通常的内容页面）。
- *重定向到另外一个资源*-你可以使用 `sendRedirect()` 方法来重定向客户端到其它指定的 URL。

使用 Servlet 响应 API 的一个重要原则是你调用的任意操作头部信息或 Cookie 的方法**必须**在第一个内容缓冲区满并被刷新到客户端之前执行。这个限制的原因是这样的信息会在 HTTP 响应的开头来发送，所以尝试如在头部已经发送之后再添加头部信息的事情是不会有效果的。

当你使用模型 2 应用程序的表示页面时，那么通常你不会直接使用 Servlet 响应 API。在 JavaServer Pages 的情况下，Servlet 容器的 JSP 页面编译器会转换页面成为一个 Servlet。JSP servlet 呈现响应，在你插入 JSP 自定义标签的地方产生动态信息。

其它展示系统，比如 Struts 的 Velocity 工具，可能代理呈现响应到特定的 Servlet，但同样的模式也是如此。你可以创建模板，而动态响应会从模板中自动生成。

要获取更多关于 Servlet 响应的内容，可以参考：

- 构造响应（Sun 的 J2EE 教程）

0.7.5 过滤

如果你使用了基于 **2.3** 或更高版本 Servlet 规范的 Servlet 容器（比如 Tomcat 4.x），你可以利用新的过滤器 API[`javax.servlet.Filter`]的优点，它允许你组合一组组件来处理请求或响应。过滤器会汇总成一条链，在这条链中每个过滤器都有机会来处理请求和响应，在它的下一个过滤器之前或之后处理（Servlet 是最终被调用的）。

Struts 1.0，1.1 和 1.2 版本需要仅仅 2.2 或更高版本的 Servlet 规范实现的 Servlet 容器，所以那些版本没有使用到过滤器。从 1.3 版开始，容器需要支持 2.3 或更高版本的 Servlet 规范，那么 Struts 框架就可以在后期使用过滤器了。

要获取过滤器的更多内容，可以参考：

- 过滤请求和响应

0.7.6 Session

HTTP 的一个核心特性是*无状态的*。用其它话讲，就是没有为 HTTP 构建来标识从相同用户发出的后续请求并关联到之前请求上的机制。这就使得构建想要用户参与一些请求会话的应用程序有些困难了。

为了减轻这个困难。Servlet API 提供了一个编程的概念，称为 Session，由一个实现了 `javax.servlet.http.HttpSession` 接口的对象所代表。Servlet 会使用两种技术（Cookie 或 URL 重写）之一来保证从相同用户发出的下一个请求会包含这个 Session 的 *session id*，这样，存储在 Session 中的状态信息可以和多个请求相关联起来了。状态信息存储在 Session 的属性中（在 JSP 里，就是我们熟知的“会话范围 bean”）。

为了当用户未能完成交互而避免无限期的占用资源，Session 可以配置*超时间隔*。如果两次请求之间的时间超过这个间隔，Session 就会超时，所有的 Session 属性也会被移除。你可以在 Web 应用程序的部署描述符中定义默认的 Session 超时时间，也可以动态地调用 `setMaxInactiveInterval()` 方法来更改特定的 Session。

不像请求，你需要关注 Session 属性（这些 bean 提供的方法，而不是 Session 本身的 `getAttribute()` 和 `setAttribute()` 方法）的线程安全问题。从相同用户发出的多个请求就很容易了，因此它们也会访问同一个 Session。

另外一个重要的考虑点是 Session 属性在请求之*间*占用服务器的内存。这也会影响到应用程序可以同时支持多少个用户。如果应用程序需要同时包含很大数量的在线用户，那么就要考虑去减少使用 Session 属性了，努力控制总内存占用量来支持应用程序。

要获取更多关于 Session 的内容，可以参考：

- 维护客户端状态（Sun 的 J2EE 教程）
- `javax.servlet.http.HttpSession`

0.7.7 分发请求

Java Servlet 规范扩展了 HTTP 请求/响应生命周期，允许请求在资源之间被分发或转发。Struts 框架使用这个特性通过特殊的组件来传递请求，比如处理响应的一个方面。在通常的流程中，请求可以传递控制器对象，模型对象，最终到视图对象中并作为独立的请求/响应周期的一部分。

0.7.8 Web 应用程序

正如 HTTP 服务器可以用来托管几个不同的网站，Servlet 容器也可以用来托管多于一个的 Web 应用程序。Java Servlet 平台提供了一个设计良好的机制来组织和部署 Web 应用程序。每个应用程序在它自己的命名空间下运行，所以它们可以各自进行部署。Web 应用程序可以组装到一个 Web 应用程序存档或 WAR 文件中。单独的 WAR 文件可以上传到服务器之后自动部署。

0.7.9 Web 应用程序部署描述符（web.xml）

应用程序生命周期的很多方面可以通过称为 Web 应用程序部署描述符的 XML 文档来配置。这个描述符的模式，或 web.xml 是根据 Java Servlet 规范给定的。

0.7.10 安全

在 Web 应用部署描述符中一个可以配置的细节是容器安全管理。声明式安全可以用来保护从 URI 中过来匹配给定模式的请求。务实安全可以用来调整安全做出基于时间，调用的参数或 Web 组件内部状态的验证决定。它也可以用来限制基于数据库信息的认证。

要获得更多关于安全的信息，可以参考：

- J2EE 蓝图：安全
- SecurityFilter
- Acegi Security（现在为 Spring Security，译者注）
- 为 HTTP/HTTPS 转换的 Strtus SLL 扩展

0.8 JavaServer Pages，JSP 标签库和 JavaServer Faces

JavaServer Pages（JSP）是“从内到外转换成 Servlet”，创建和维护动态 Web 页面是很容易的。而不用放置你想写入到 HTTP 响应的 Java 的 `print` 语句中的内容，JavaServer Pages 中的所有内容可以写入到响应对象中，**除了**放置到特殊 Java 语句中的东西。

使用 **JavaServer Pages**，你可以在标准 HTML 中编写页面代码，使用 Java 语言的语句或是 JSP 标签来添加动态特性。Struts 框架的发布包中包含一些 JSP 标签，从一个 JSP 页面中访问框架特性是很容易的。

要获取更多关于 JavaServer Pages 和自定义 JSP 标签库的内容，可以参考：

- `java.sun.com` 的 JavaServer Pages 技术页
- 从 `java.sun.com` 下载 JSP 1.2 和 2.0 规范
- JavaServer Pages 技术（Sun 的 J2EE 教程）
- JSP 页面中的自定义标签（Sun 的 J2EE 教程）

很多时候，JSP 标签是和 JavaBean 联合使用的。应用程序发送 JavaBean 到 JSP，JSP 标签使用 bean 来为当时的用户自定义页面。要获取更多内容，可以参考 J2EE 教程的 JSP 页面中的 JavaBean 组件。

Struts 框架也可以和其它资源的 JavaServer Pages 标准标签库（JSTL）良好运行，比如 JSP

标签，Jakarta 标签库，Struts 布局和 Display 标签

Struts 框架中的一个可用组件是 Struts-EL。这个标签库是特殊设计的，可以和 JSTL 协同工作。特别是，它使用了和 JSTL 相同的“表达式语言”引擎来计算标签属性。这可以和原来 Struts 标签库来对比，它们只能对动态属性值使用“rtexprvalue”（runtime scriptlet expression，运行时脚本表达式）。

也有一些工具可以让 Struts 框架方便使用 XSLT 和 Velocity 模板。

Java 视野上的新星是 JavaServer Faces 技术。JSF 简化了构建 Java 服务器程序的用户界面，对于 Web 应用和桌面应用都可以。对于要分布过度 JSF 和 Struts 1 的开发团队，我们提供 Struts-Faces 标签库，这可以让你在已有的 Action 中使用 JSF 组件，所以你可以在页面中迁移到 JSF。

要获取 JSF 的开源实现，可以访问我们的兄弟项目，Apache MyFaces。

要获取更多关于 JSTL 和 JavaServer Faces 的信息，可以参考：

- JSTL 实践，第一部分（by Sue Spielman）
- JSF 核心（JavaServer Faces 资源）
- JavaServer Faces 资源（James Holmes.com）
- Apache MyFaces-JSF 规范的开源实现

0.9 可扩展标记语言（XML）

Struts 框架提供的特性依赖于对象的数量，通常是使用可扩展标记语言来编写的配置文件。XML 也被用来配置 Java Web 应用程序，所以这也是一个很熟悉的方法。

要了解更多关于 XML 是如何在 Java 应用程序中使用的，请参考 Java 的 XML 处理 API 指南。框架在内部能很好使用 API，这就是当开发人员使用 Struts 框架编写他们自己的应用程序时最多使用的东西。

0.9.1 描述符

当 Java 应用程序使用 XML 配置文件，元素通常是被用作*描述符*。应用程序不直接使用 XML 元素。元素用来创建和配置（或部署）Java 对象。

Java Servlet 平台使用 XML 配置文件来部署 Servlet（在其它事项中）。同样，Struts 框架使用 XML 配置文件来部署对象。

0.10 JAAS

Struts 框架可以和任意的用户认证方法一起工作，1.1 和后续的版本提供对标准 Java 认证和认证服务（JAAS）的直接支持。现在你可以在 action 连 action 的基础上指定安全策略。

要了解更多关于 JAAS 的信息，可以参考 Sun 的开发者网站产品页面。

在 Java Web 应用程序中处理安全的流行扩展，包括扩建应用程序，就是 SecurityFilter。

0.11 其它层面

Struts 框架提供 Web 应用程序的控制层面。开发人员可以使用这个层面和其它的标准技

术一起来提供业务处理，数据访问和展示层。

要了解诶更多关于创建业务和数据访问层的信息，可以参考第二章相关内容。

要了解更多关于创建视图层的信息，可以参考第三章的相关内容。

第一章 介绍

"Read the directions and directly you will be directed in the right direction."

阅读简介指导，那么你就能直接找到正确的帮助。

本框架文档为动态 Web 开发者所写，假设开发者有关于如何创建 Java Web 应用程序的工作经验。要获得关于更多底层的技术，可以参考关键技术入门。

1.1 回到过去！（Struts 简史）

当初发明 Java Servlet 的时候，很多程序员很快地意识到那是一个利器。它比标准的 CGI（Common Gate Interface，通用网关接口，译者注）更快更强大，有良好的可移植性，还有无限的扩展性。

但是无限制的使用 `println()` 语句来书写 HTML 并发送到客户端浏览器是很无聊的事情。解决的答案就是 Java Server Pages，它可以把 Servlet 代码嵌入到页面中。现在开发者可以很容易的混合 HTML 代码和 Java 代码，JSP 有所有 Servlet 的优点。编程无拘无束。

Java Web 开发很快就变成了“以 JSP 为中心”。这对它本身来说并不是一件坏事，但是对于流程控制和 Web 应用程序特有的问题来说，JSP 做的就太少了。

很明显，还需要一种范例...

很多聪明的开发者意识到 JSP 和 Servlet 可以**一同**使用来部署 Web 应用程序。Servlet 可以帮助控制流程而 JSP 可以关注编写 HTML 页面这讨厌的业务。在适当的时候，一起使用 JSP 和 Servlet 就被认为是模型 2 了（也就意味着，想必单独使用 JSP 就是模型 1 了）。

当然，Sun 公司也没有什么新的技术了...而且很快 JSP 的模型 2 被指出是遵循了经典的 Model-View-Controller（模型-视图-控制器）设计模式，MVC 模式是从过去的 Smalltalk MVC framework 中抽象出来的。Java Web 开发人员现在倾向于使用模型 2 和 MVC 的互换。在本指南中，我们使用 MVC 范式来描述框架架构，这可能是最好的模型 2/MVC 设计。

Apache Struts 项目最初在 2000 年 5 月由 Craig R. McClanahan 发起，为 Java 社区提供一个标准的 MVC 框架。在 2001 年 7 月，1.0 版本的 Struts 发布了，Java 模型 2 的开发就不一样了。

1.2 Model-View-Controller(MVC)设计模式

术语“MVC”源于 SmallTalk 的模型-视图-控制器框架。在 MVC 下，应用程序被视为含有三个不同的部分。问题域由模型代表。对用户的输出由视图代表。那么从用户的输入由控制器代表。

1.2.1 模型：系统状态和业务逻辑 JavaBean

基于 MVC 系统的模型部分可以被分成两个主要的子系统—系统的**内部状态**和可以改变状态的 **action**。

在语法方面，我们可能想把状态信息视为**名词**（描述信息）而把 action 视为**动词**（可以改变那些信息的状态）。

很多应用程序将系统的内部状态表现为一个或一组 JavaBean。Bean 的属性代表系统状态的细节。基于应用程序的复杂程度，那些 bean 可能是自我包含的（也知道怎么去持久它们自己的状态），或者它们可能是知道如何从其它组件中检索系统状态的门面。这个组件可能是数据库，搜索引擎，企业级 Java 实体 Bean（Entity EJB），LDAP 服务器或其它什么东西。

大规模应用程序经常将可能的业务操作集合表示为可以在 bean 中调用的方法，或者是 bean 维护状态信息。比如，你可能有一个购物车的 bean，为当前每个用户存储在会话（session）范围内，那些 bean 中的属性代表了当前用户决定购买物品的集合。这个 bean 可能有一个 `checkOut()` 方法来验证用户的信用卡并发送订单到库房来挑选货物和发送。其它系统将分别表示它们各自可用的操作，比如企业级会话 JavaBean（Session EJB）。

在小规模应用程序中，另一方面，可用的操作可能嵌在了 `Action` 类中，它们是框架控制层的一部分。这当逻辑非常简单时或在其它环境中没有考虑重用业务逻辑时，这就很有用。

框架的架构要足够灵活，以便来支持很多访问模型的操作，但是我们**强烈**建议将业务逻辑（“如何来完成”）从扮演控制（“做什么”）的 `Action` 类角色中分离出来。

对于很多关于将应用程序模型适应到框架中的内容，请参考第二章

1.2.2 视图：JSP 页面和表示组件

基于 Struts 的应用程序的*视图*部分通常是用 JavaServer Pages（JSP）技术来构建的。JSP 页面可以包含被称作是“模板文本”的静态 HTML（或者是 XML）代码，加之它有能插入基于解释特殊 action 标签的动态内容的能力。JSP 环境包含了一组标准的 action 标签，比如 `<jsp:useBean>`，它的作用在 JSP 规范中有描述。除了内建的 action，也有标准的方法来定义你自己的标签，这就是我们熟知的“自定义标签库”。

Struts 框架包含了一组自定义标签库，它们可以容易地来创建完全国际化的用户界面，并能自如地和 `ActionForm` bean 来交互。ActionForm 捕捉和验证什么样的输入内容是应用程序所需的。

要了解更多关于 Struts 标签库，和框架使用表现页面，请参考第三章的相关部分。此外文档关于标签库的部分在标签库子项目中也是有的。

1.2.4 控制器：ActionServlet 和 ActionMapping

Struts 框架提供应用程序的*控制器*部分。控制器关注于接收从客户端（典型的情况是用户使用 Web 浏览器）发出的请求，然后决定哪个业务逻辑功能来执行，之后将处理责任代理给下一阶段用户界面中合适的视图组件。Struts 框架中控制器的核心组件是一个 Servlet 类 `ActionServlet`。这个 Servlet 通过定义一组 `ActionMapping` 来配置。一个 ActionMapping 定义了基于接收到请求的请求 URI 所匹配的 `path` 和 Action 类指定的完全限定类名。所有的 Action 都是 `[org.apache.struts.action.Action]` 的子类。Action 封装调用业务逻辑类，解释输出，和查找视图组件来生成响应的最大控制权。最后 Struts 框架将处理结果分派给一个视图，最后在它的范围之外来呈现视图。

Struts 框架也支持使用超越必须配置并有额外属性 `ActionMapping` 类来操作控制器。这也就允许你为应用程序存储额外指定的信息，并利用 Struts 框架的其它特性。此外，Struts 还允许你定义逻辑“名称”，表示控制应该转发去的地方，比如 action 方法可以请求“主菜

单”页面，这样它就不需要知道对应 JSP 页面的实际所在路径了。这个特性可以极大地防住你分离控制逻辑（做什么）和显示逻辑（如何呈现）。

要了解更多关于控制层的信息，可以参考第四章。

1.3 框架控制流

Struts 框架提供了一些组件来弥补 MVC 风格应用程序的**控制**层的不足。这包含控制器组件（Servlet），开发者定义的请求处理程序，和一些支持对象。

Struts 标签库组件提供了对 MVC 应用程序中**视图**层的直接支持。这些标签中的一部分可以访问控制层对象。其它则是方便的通用标签，在编写应用程序时可以使用。其它标签库，包括 JSTL，也可以在 Struts 框架中使用。其它显示层技术，比如 Velocity Template 和 XSLT 也是可以在 Struts 框架中使用的。

MVC 应用程序中的**模型**层通常是对项目来自行指定的。Strtus 框架的设计使得很容易访问应用程序的业务端，但是留下这个部分对其它产品进行编程实现，比如 JDBC，Enterprise JavaBean，Object Relation Bridge 和 iBatis，仅举这几个例子。

现在我们来逐步解释这些组件是如何结合在一起的。

当初始化时，控制器解析配置文件（`struts-config.xml`），并使用它来部署其它控制层对象。总之，这些对象是来自 **Struts 配置**的。Struts 的配置定义（除其它事项）应用程序 ActionMapping`[org.apache.struts.action.ActionMappings]`的集合。

控制器组件咨询 ActionMapping 如何将 HTTP 请求路由到框架的其它组件中。请求可能被转发到 JavaServer Pages 或由应用程序开发者提供的 Action`[org.apache.struts.action.Action]`的子类中。通常情况下，请求首先被转发到 Action 中，之后到 JSP（或其它展示页面）中。这个映射帮助控制器将 HTTP 请求转入应用程序的 action 中。

独立的 ActionMapping`[org.apache.struts.action.ActionMapping]`通常会包含一组属性：

- **请求路径**（或“URI”）
- 作用于请求之上的**对象类型**（Action 子类）
- 其它需要的属性

Action 对象可以处理请求并对客户端（通常是 Web 浏览器）做出响应，并指示控制应该转发到什么地方。比如，如果登录成功，那么复杂登录的 Action 可能希望将请求转发到主菜单页面。

Action 可以访问应用程序的控制器组件，也可以访问其它成员的方法。当转发控制时，Action 对象可以直接转发一个或多个共享对象，包含 JavaBean，将它们放置到 Java Servlet 共享的标准上下文环境中。

比如，Action 对象可以创建购物车的 bean，并将商品加入购物车，将这个 bean 放置在 session 的上下文环境中，之后转发请求到另外一个映射中。那个映射肯会使用 JavaServer Pages 来展示用户购物车中的内容。因为每个客户端都有它独自的 session，那么它们就会有它们自己独立的购物车。

很多应用程序中的业务逻辑可以使用 JavaBean 来呈现。Action 可以调用 JavaBean 中的属性而不需要知道它实际是怎么工作的。这就封装了业务逻辑，所以 Action 就关注在错误处理和向哪里转发控制上了。

JavaBean 也可以用于管理输入表单。设计 Web 应用程序的关键的问题是保持和验证用户在请求中输入的信息是什么。你可以定义你自己的一组输入 bean 类，将它们作为

ActionForm [org.apache.struts.action.ActionForm]的子类。ActionForm 类使得存储和**验证**应用程序输入表单中的数据非常容易。ActionForm bean 会自动在标准，共享的上下文集合中存储，所以那些数据也可以被其它对象来使用，比如 Action 对象和其它 JSP。

Form bean 可以被 JSP 用于从用户收集数据…由 Action 对象用于验证用户输入的数据…再次由 JSP 来重新填充表单字段。在校验错误时，Struts 框架有一个对提出和显示错误信息的共享机制。

配置信息中的另外一个元素是 ActionFormBean[org.apache.struts.action.ActionFormBeans]。这是描述符对象的集合，它们用来在运行时创建 ActionForm 对象的示例。当映射需要一个 ActionForm 时，Servlet 通过名称查找 form-bean 描述符，并使用它来为指定类型创建 ActionForm 的实例。

这里是当从映射中请求调用是 ActionForm 时发生的事件序列：

- 控制器 Servlet 检索或创建 ActionForm bean 的实例。
- 控制器 Servlet 传递 bean 到 Action 对象中。
- 如果请求被用于提交输入页面，那么 Action 对象可以检验数据。如果需要的话，数据可以和要在页面显示的消息列表一同被送到后台的输入表单中。否则数据就被传递到业务层中。
- 如果请求用来创建输入页面，Action 对象可以使用任意输入页面可能需要的数据来填充 bean。

Struts 的标签库组件提供自定义标签，它们可以自动从 JavaBean 中填充字段。几乎所有的 JavaServer Pages 都需要知道的是使用什么字段名，并向哪里提交表单。

其它标签可以自动输出由 Action 或 ActionForm 排队的消息，仅仅需要集成到页面中。消息可以被设计成本地化的，并对用户本地化信息显示最佳的可用消息。

Struts 框架和标签库被设计成从地面行动来支持到 Java 平台中的国际化特性构建。所有的字段标签和消息可以从 message resource（消息资源）中来获得。为另外一种语言提供消息，简单将文件添加到资源束中即可。

国际化方法，消息资源的其它好处是在表单之间形成一致的标签，还有检查所有从核心位置获取标签和消息的能力。

对于最简单的应用程序，Action 对象可能有时处理和请求相关的业务逻辑。**然而在很多情况下，Action 对象应该调用其它对象，通常是 JavaBean 来处理真实的业务逻辑**。这就让 Action 关注于错误处理和流程控制，而不是业务逻辑。为了允许在其它平台上重用，业务逻辑 Javabean 不应该引用任何 Web 应用程序对象。Action 对象应该从 HTTP 请求中翻译所需的细节信息，并作为普通的 Java 对象传递给随后的业务逻辑 bean。

在数据库应用程序中，比如：

- 业务逻辑 bean 可以连接和查询数据库
- 业务逻辑 bean 返回结果给 Action
- Action 存储结果在请求中的 form bean 内
- JavaServer Page 以 HTML 格式来显示结果

Action 和 JSP 都不需要知道（或关心）结果来自何处。它们仅仅需要知道如何打包和显示它们即可。

在本文档中的其它部分包含了各种框架组件的详细介绍。Struts 标签库组件包含一些涵盖自定义标签各个方面的开发指南。一些示例应用程序捆绑在发布包中，它们会来演示它们是如何结合在一起的。

Struts 框架基于 Apache Software Foundation license（阿帕奇软件基金会许可）发布。代码是有版权的，但是可以自由用于任何应用程序。

第二章 构建模型组件

"If I had a world of my own, everything would be nonsense. Nothing would be what it is, because everything would be what it isn't. And contrary wise, what is, it wouldn't be. And what it wouldn't be, it would. You see?"

如果我有我自己的世界，那么所有的东西都是废话。没有什么是它本身，因为所有的东西都不是。相反，它是的也会不是，而不是的也会是，你明白吗？

2.1 概述

很多用于构建 Web 应用程序的需求文档都关注于*视图*。而你应该保证对每个提交请求所需的处理也要从模型角度定义来明晰。通常情况下，*模型*组件的开发者将会关注于 JavaBean 类的创建来支持所有需要的功能。这些所需的 bean 的确切性质由应用程序本身来确定，这很大程度上是基于应用的需求的，但通常它们可以被分成下面要讨论的几类。首先，我们来看一下关于 bean 和 JSP“范围”概念的简要概述。

2.2 JavaBean 和范围

在基于 Web 的应用程序中，JavaBean 可以存储在（）一些不同“属性”的集合中。每个集合都有它不同的生命周期规则，还有 bean 存储的可见性。而定义生命周期和可见性的规则被称为 bean 的*范围*。JavaServer Pages（JSP）规范定义了使用下列术语（这和 Servlet API 在括号中定义的概念是相同的）的选择范围：

页面 page-Bean 只能在单独的 JSP 页面中可见，也就是当前请求的生命周期。（service 方法的局部变量）

请求 request-Bean 在单独的 JSP 页面和在这个页面中或者由这个页面转发的任意页面或 Servlet。（request（请求）对象属性）

会话 session-Bean 在所有 JSP 页面和参与到特定用户会话的 Servlet 中可见，跨越一个或多个请求。（session（会话）对象属性）

应用 application-Bean 在 Web 应用程序的所有的 JSP 页面和 Servlet 部分都可见。（Servlet 上下文环境属性）

记住 JSP 页面和 Servlet 在相同 Web 应用程序中共享的相同 bean 的集合是很重要的。比如，在请求对象属性中存储的 bean 在 Servlet 中就会是这样：

```
MyCart mycart = new MyCart(...);
request.setAttribute("cart", mycart);
```

在 Servlet 要立即转发的 JSP 页面中是可见的，可以这样使用标准的 action 标签：

```
<jsp:useBean id="cart" scope="request"
class="com.mycompany.MyApp.MyCart"/>
```

2.3 ActionForm Bean

注意：ActionForm bean 通常有和模型 bean 属性对应的属性，form bean 本身应被视为

是控制器组件。因此，它们能够转换模型和视图层之间的数据。

Struts 框架通常假设已经定义为应用程序的输入表单了一个 ActionForm bean（也就是说，有一个扩展了 ActionForm 类的 Java 类）。ActionForm bean 有时也称作“form bean”。这些都是细粒度的对象，所以对每个表单都有一个 bean，或者是粗粒度的，那么一个 bean 服务于多个表单或是这个应用程序。

如果在 Struts 配置文件（参考 4.8 节）中声明了这样的 bean，那么 Struts 控制器 Servlet 将会自动为你在调用合适的 Action 方法前执行下面的服务：

- 在相应的范围（请求或会话），根据相应的键，检查相应类的 bean 的实例。
- 如果没有找到这样可用的 bean 的实例，那么就会自动创建一个新的实例并加入到响应的范围（请求或会话）内。
- 对于每个请求参数，其名称和 bean 的属性名相对应，那么对应的 setter 方法就会被调用。当你使用星号通配符来选择所有属性时，这个操作和标准的 JSP action 的 <jsp:setProperty>规则相似。
- 更新之后的 ActionForm bean 将会被传递到 Action 类[org.apache.struts.Action]的 execute 方法中，所以在系统状态和业务逻辑 bean 中，那些值就是可用的了。

要了解关于 Action 和 ActionForm bean 的编码详情，请参考第四章。

应该注意“form”（这里讨论的意义），并非必须和用户界面的单独 JSP 页面相对应。在很多应用中都可以有“form”（从用户的角度来说）扩展到多个页面，这很普通。比如，这样来想，向导风格的用户界面通常在安装新的应用程序时来使用。Struts 建议你定义单独的 ActionForm bean，它可以包含所有字段属性，而不管实际哪个页面的字段来展示。同样，同一形式的所有页面应该被提交的相同的 Action 类中。如果你遵循了这些建议，那么页面设计者可以重新在所有页面中安排字段，而不需要改变处理逻辑。

小型应用程序可能仅仅需要单独的 ActionForm 来服务于所有的输入表单。其它应用程序可以使用单独的 ActionForm 为应用程序的每个主要子系统。一些开发团队可能更想把 ActionForm 类从不同的表单或工作流中分离出来。要用多少 ActionForm 这都又你来决定。Struts 框架本身不关心这些事情。

2.4 系统状态 Bean

系统的实际状态通常表示为一个或多个 JavaBean 类，它们的属性定义了当前的状态。比如，购物车系统，会包含一个表示为每个独立购物者维护购物车的 bean，也会（除了其它事项）包含了购物者选择购买的商品的组合。分来来说，系统还会包含不同的 ben 来描述用户的个人资料信息（包含它们的信用卡和收货地址），还有可用的商品分类和它们当前的库存量。

对于小规模的系统，或者不需要长时间保持状态信息，系统状态 bean 的集合可能包含所有系统曾具有的特殊细节的知识。或者说，这是常有的情况，系统状态 bean 将会展示暂时存储在外部数据库（比如 CustomerBean 对象对应 CUSTOMERS 表中特定的一行）中的信息，如果需要的话就就创建或者从服务器内存中移除。在大型应用程序中，企业级 Java 实体 bean 也可以用于这个目的。

2.5 业务逻辑 Bean

你应该封装应用程序的功能逻辑作为方法在调用，JavaBean 的设计就是这个目的。这些方法可能是用于系统状态 bean 的相同类的一部分，或者它们可能在不同的类中，致力于逻辑的执行。在后一种情况中，你通常应该使用传递系统状态 bean 作为参数来操作这些方法。

为了代码的最大重用，业务逻辑 bean 应该被设计并实现为它们不需要知道自己在 Web 应用程序环境中被执行了。如果发现在 bean 的代码中不得不引入 `javax.servlet.*`类，那么可以尝试将这个业务逻辑放到 Web 应用程序环境中。考虑重新安排一些事情，那么 `Action` 类（控制器的一部分，如下所述）会从 HTTP 请求中翻译所有需要的信息，并在业务逻辑 bean 中调用属性的 setter 方法来处理，之后就可以调用 `execute` 方法了。这样的业务逻辑类可以在环境中重用而不是在它们被初始构建的 Web 应用程序中。

基于应用程序的复杂度和适用范围，业务逻辑 bean 可能普通的 JavaBean，它们可以作为参数来传递，和系统状态 bean 交互。或者普通的可以使用 JDBC 调用访问数据库的 JavaBean。对于大型应用程序，这些 bean 通常会用有状态/无状态的企业级 JavaBean（EJB）来代替。

对于在应用程序中使用的数据库的详情，可以参考 FAQ 章节中的如何访问数据库部分。

对于业务逻辑和数据访问框架的详情，可以参考关键技术入门。

2.5.1 DynaBean 动态 Bean

动态 bean 结合了 JavaBean 的可扩展性和 Map 的灵活行。定义简单 JavaBean 需要定义一个新的类并编写每个属性的两个方法。动态 bean 的属性可以通过 XML 描述符来配置。动态 bean 的虚拟属性不能被标准的 Java 方法所调用，但是可以依赖反射和自我检查组件来很好地使用。

在应用程序中，你可以使用动态 bean 来描述 HTML 表单。这个策略可以不去创建正式的 JavaBean 子类来存储简单的属性。

要了解更多关于动态 bean 的内容，请参考

- Commons 组件 BeanUtils 中包的描述和 JavaDoc 文档
- Struts 的 Form 参考（用于 Struts 动态 ActionForm 的插件）

2.6 Apache Commons 组件 Chain

组织执行复杂处理流程的流行技术就是“责任链”模式，在经典书籍“Gang of Four”设计模式中有描述（还有其它很多地方）。GoF 总结了责任链模式是“米面耦合请求发送者到它的接受者中，通过给予多余一个对象的机会来处理请求。接收对象链和沿着链传递请求，直到有一个对象可以处理它。”

CoR 模式帮助我们保持软件组件的松耦合。组件可以调用责任链，而不需要知道什么对象在链子上或它们是怎么实现的。更重要的是，我们可以调整链而不需要改变调用者是如何调用链的。对于 1.3 版本的 Struts 来说，默认的请求处理器，担当了框架的“核心”，就是一个责任链。

为了实现这个链，请求处理器使用 Apache Commons 中的责任链组件，它提供了一个标准的 CoR 模式的实现，还有各种服务于请求的上下文和命令对象的实现。

要获得更多关于责任链的信息，请参考

- Apache Commons 组件的责任链
- 看一看 Commons Chain
- 用 Struts 1.3 更好的编码

对于 Struts 1.3，Commons Chain 用来为框架构建默认的请求处理器。

第三章 构建视图组件

"What if I should fall right through the center of the earth... oh, and come out the other side, where people walk upside down."

如果我坠落地穿越地球的中心…哦，然后从另一侧出来，那边的人是倒着走的。

3.1 概述

Struts 框架提供了视图组件的基本支持，但是没有提供任何实际的视图组件。一些显示技术（参考 3.4 节）是可用的，包括 Cocoon，JSP，Belocity Template 和 XSLT。

3.2 国际化消息

很多年以前，应用程序开发者只能支持他们本国的用户来使用，只使用了一种（有时是两种）语言，只有一种展示数字量，比如日期，数字和货币值。然而，基于 Web 技术的爆炸式应用程序开发，还有这样应用程序在 Internet 和其它广泛访问的网络上部署，很多情况下已经呈现出无形国界的限制。这已经翻译（如果你会原谅这个双关语）需要的应用程序来支持国际化（通常称为“i18n”，因为 18 是在“i”和“n”之间的字母数量）和本地化。

Struts 框架在 Java 平台的标准可用类库之上构建的，它可以用来构建国际化和本地化的应用程序。我们熟悉的关键概念是：

- 语言环境 Locale-Java 类基本支持的国际化是 `Locale`。每个 `Locale` 代表一个特定的国家和语言（加上可选的语言变体）的选择，而且是一组对如数字和日期这些数值的格式化假定。
- 资源束 ResourceBundle-`java.util.ResourceBundle` 类提供了支持多语言环境消息的基本工具。参看 `ResourceBundle` 类的 Javadoc 文档，还有文档中有关 JDK 发布包的国际化信息来获取详情。
- 属性资源束 PropertyResourceBundle-`ResourceBundle` 的一个基本实现，它允许你使用“名=值”的语法来定义资源去初始化属性文件。这对在 Web 应用程序中使用消息来准备资源束是非常方便的，因为这些消息是一般是面向文本的。
- 消息格式 MessageFormat-类允许你在运行时使用指定参数来替换消息的字符串（在这种情况下，我们可以从资源束中检索）部分。这在创建表述语句时是非常有用的，但是在不同的语言中词语出现的顺序可能不同。消息中的占位符字符串`{0}`会被运行时第一个参数来替换，`{1}`会被第二个参数来替换等等。
- 消息资源 MessageResources-Struts 框架的类可以让你将一组资源束视作是一个数据库，而且允许你为特定的语言环境（通常是和当前用户相关的）请求特定的消息字符串，而不是服务器本身运行的默认的语言环境。

请注意在服务器端框架的 i18n 支持限制于给用户国际化文本和图片的**展示**。支持特定语言的**输入方法**（和语言一起使用，如日语，中文和韩语）是留给客户端设备的，通常就是 Web 浏览器了。

对于一个国际化的应用，遵循在你的平台上的 JDK 文档中国际化部分文档描述的步骤，来创建包含对每种语言的消息属性文件。示例会在后面来说明：

假设你的源代码在 com.mycompany.mypackage 包中来创建，那么它就存储在目录（相对于你的源码目录）名为 com/mycompany/mypackage 的路径下。要创建名为 com.mycompany.mypackage.MyApplication 的资源束，那么你应该创建如下的文件在目录 com/mycompany/mypackag 中：

- **MyApplication.properties**-包含服务器默认语言的消息。如果默认语言是英语，可能会有这样的配置项：prompt.hello=Hello
- **MyApplication_xx.properties**-包含不同语言的相同消息，而这种语言的的 ISO 编码是“xx”（参考 ResourceBundle 的 javadoc 页面链接到当前可用语言的列表）。对上面那个配置项使用法语版本的消息,那么就会是这样：prompt.hello=Bonjour。资源束文件你需要多少种语言就可以写多少个。

当通过 strtus-config.xml 配置文件来配置控制器 Servlet 时，你需要定义的一个啥东西是应用程序资源束的基础名称。在上面描述的示例中，这个基础名称就是 com.mycompany.mypackage.MyApplication。

```
<message-resources parameter="com.mycompany.mypackage.MyApplication"/>
```

那么在应用程序的类路径下发现资源束文件也是很重要的。另外一种方法是在应用程序的 classes 文件夹中存储 MyResources.properties 文件。那么就可以指定“myResources”作为应用程序的值。要小心如果的构建脚本删除类文件是作为“clean”目标的一部分时，它是不能被删除的。

如果要那么做，当编译应用程序并复制 src/conf 目录的内容到 classes 目录时，这里的 Ant 任务脚本可以运行：

```
<!-- 复制任意配置文件 -->
<copy todir="classes">
   <fileset dir="src/conf"/>
</copy>
```

3.3 Form 和 FormBean 的互动

注意：这里给出的示例使用 JSP 和自定义标记，ActionForm bean 和其它控制器组件是中立于视图的。Struts 框架也可以使用 Cocoon，Velocity Template，XSLT 和其它可以通过 Java Servlet 来呈现的展示技术。

3.3.1 自动表单填充

同一时间或其它时候，很多 Web 开发者使用标准的 HTML 功能来构建表单，比如<input>标记。用户所期望的交互式应用程序要有特定的行为，还有有和这些期望相关的错误处理--如果用户操作出错，那么应用程序应该允许它们来修改需要改变的部分--而不是重新输入当前页面或表单中剩余的部分。

当使用标准 HTML 和 JSP 页面编码实现这一期望是乏味和累赘的。比如，对于 username 字段的 input 元素可能会是这样（在 JSP 中）：

```
<input type="text" name="username" value="<%= loginBean.getUsername() >"/>
```

要输入正确就很难了，会迷惑 HTML 开发人员，它们对编程概念不熟悉，可能会在 HTML 编辑器中引发问题。相反，Struts 标签库提供了构建表单的综合措施，这是基于 JSP 1.1 的自定义标签库。上面的示例要使用 Struts 标签库就可以这么来呈现：

```
<html:text property="username"/>
```

就不需要明确指定从哪个 JavaBean 中获取初始值了。这是由 JSP 标记自动处理的，使用由 Struts 框架提供的便利。

HTML 表单有时要用于上传文件。很多浏览器支持它是通过<input type=”file”>元素，这会生成一个文件浏览按钮，但是它是受开发者来处理传入的文件的。Struts 框架处理这些“multipart”形式是以相同的方式来建立正常的形式。

比如要使用 Struts 框架来创建简单的登录表单，参考 FAQ 中如何创建 ActionForm。

3.3.2 自动表单验证

除了上面描述的表单和 bean 的交互，Struts 框架提供验证它收到的输入字段的额外方式。要使用这个特性，在 ActionForm 类中要覆盖下列方法：

```
validate(ActionMapping mapping,HttpServletRequest request);
```

validate 方法在 bean 属性填充之后由控制器 Servlet 调用，而且是在对应的 action 类的 execute 方法调用之前执行。validate 方法有下列可选项：

- 执行适当的验证发现没有问题--返回 null 或零长度的 ActionErrors 实例，之后控制器 Servlet 将会进行适当 Action 类的 execute 方法调用。
- 执行适当的验证发现了问题--返回一个包含 ActionMessage 的 ActionErrors 实例，其中包含应该显示的错误消息的键(在应用程序的 MessageResources 资源束中)。控制器 Servlet 将会存储这个数组在请求属性中，并合适<html:errors>标记的使用，之后将会转发到输入表单（为这个 ActionMapping 找到标识的 input 属性)。

正如之前提到的，这个特性是完全是可选的。默认的 validate 方法实现返回值是 null，而控制器 Servlet 会假设任何需要的验证是由 action 类完成的。

一个通用的方法使用 ActionForm 的 execute 方法来执行简单，初步验证，之后从 Action 中处理“业务逻辑”。

Struts 框架的验证器，会在下一部分来说明，会简单验证 ActionForm。

3.3.3 Struts 的验证器

配置验证器来执行表单验证是很容易的。

1. ActionForm bean 必须扩展 ValidatorForm。
2. 表单的 JSP 必须包含<html:javascript>标记来进行客户端验证。
3. 必须在 xml 文件中如下所示定义验证规则：

```
<form-validation>
   <formset>
      <form name="logonForm">
         <field property="username" depends="required">
            <msg name="required" key="error.username"/>
         </field>
      </form>
   </formset>
</form-validation>
```

msg 元素指的是当生成错误消息时使用的消息资源的键。

4. 最后，必须在 struts-config.xml 中开启 ValidatorPlugin，就像这样：

```
<plug-in className="org.apache.struts.validator.ValidatorPlugIn">
   <set-property property="pathnames"
      value="/org/apache/struts/validator/validator-rules.xml,
      /WEB-INF/validation.xml"/>
</plug-in>
```

注意：如果必须的表单属性是 Java 对象原始类型（比如 java.lang.Integer）的表示，那么必须设置 ActionServlet 的 init-parameter 中 convertNull 为 true。没有做这一步会导致必须的验证不会在那个字段上执行，因为它默认是 0。

要了解更多关于 Struts 框架验证器的信息，可以参考开发指南。

3.3.4 使用 Tiles 组织页面

Tiles 是一个强大的模板库，它允许你通过组合各种“tiles”构建来构建视图。这里是一个快速创建指南：

1. 创建一个/layout/layout.jsp 文件来包含应用的通用的外观和感觉：

```
<html>
   <body>
      <tiles:insert attribute="body"/>
   </body>
</html>
```

2. 创建/index.jsp 主页文件：

```
<h1>This is my homepage</h1>
```

3. 创建/WEB-INF/tiles-defs.xml 文件，是这样的：

```
<tiles-definitions>
   <definition name="layout" path="/layout/layout.jsp">
      <put name="body" value=""/>
   </definition>
   <definition name="homepage" extends="layout">
      <put name="body" value="/index.jsp"/>
   </definition>
<tiles-definitions>
```

4. 在 struts-config.xml 文件中建立 TilesPlugin：

```
<plug-in className="org.apache.struts.tiles.TilesPlugin">
   <set-property property="definitions-config"
      value="/WEB-INF/tiles-defs.xml"/>
</plug-in>
```

5. 在 struts-config.xml 文件中建立一个 action 映射来指向主页 tile：

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

TilesPlugin 配置了指定的 RequestProcessor 来决定请求视图是否是一个 tile 并且来处理它。注意我们设置了主页 tile 扩展了 root 板式 tile 并改变了 body 属性。Tiles 插入命名在 body 属性中的文件，最终加入到主板式中。

参考 Strtus Tiles 网站来获取更深层次示例。

3.4 表示框架

Struts 框架由很多表示技术来支持，也有大量的扩展很方便地来创建视图组件。一些流行的展示技术包括：

- Struts 标签库
- Struts Cocoon
- Velocity Struts
- Stxx for XLST

3.5 其它表示技术

尽管应用程序的外观和感觉可以完全基于标准的表示库的功能来构建，也许你需要使用其它技术来直接呈现一些响应。

3.5.1 图形渲染组件

一些应用程序需要动态生成图片，比如股票站点的价格图表。有两种不同的技术通常用于实现这些需求：

使用执行 Servlet 请求的 URL 来渲染超链接。Servlet 会使用图形库来渲染图形图像，合适的设置类型（比如 `image/gif`），之后将图形字节发回浏览器，就像接收到一个静态文件那样来展示它们。

必须下载一个 Java 小程序（Applet，译者注）来渲染 HTML 代码用于创建需要的图形。你可以在渲染代码中为小程序设置合适的初始化参数配置图形，或者你可以用小程序它自己连接的服务器来接收这些参数。

3.5.2 渲染文本

一些应用程序需要动态生成文本或标记，比如 XML。如果一个完整页面被呈现了，而且可以使用 PrintWritter 来输出，那么这在 Action 中就很容易做了：

```
response.setContentType("text/plain"); // 或者是 text/xml
PrintWriter writer = response.getWriter();
// 使用 writer 来呈现文本
return(null);
```

第四章 构建控制器组件

"It would be so nice if something made sense for a change."

如果一些东西做出了改变，那么这将是很好的。

4.1 概述

既然我们已经知道如何去构建应用程序的模型和视图组件了，现在就到了关注于控制器组件的时候了。Struts 框架包含实现了主要映射请求 URI 到 Action 类功能的 Servlet。因此，开发人员对控制器的主要相关责任就是：

- 编写 ActionForm 类去调和模型和视图之间的沟通。（可以参考 FAQ 中如何构建 ActionForm 的相关内容）
- 为每个可能收到的逻辑请求编写 Action 类（扩展了 `[org.apache.struts.action.Action]`类）。
- 为每个可能提交的逻辑请求配置 ActionMapping（在 XML 中）。XML 配置文件通常命名为 strtus-config.xml。

要部署应用程序，你还要做的是：

- 为应用程序升级 Web 应用程序的部署描述符文件（在 XML 中）来引用其它组件。
- 包括应用程序的额外组件。

后面两项内容可以在第五章中找到相关介绍。

4.2 ActionServlet

对于熟悉 MVC 架构的开发人员来说，ActionServlet 负责 C-控制器。控制器的职责是：

- 处理用户请求
- 根据请求决定用户想实现什么内容
- 从模型中（如果需要）取出数据给适合的视图
- 选择合适的视图来响应用户

控制器代理了很多繁重的工作给 Request Processor 和 Action 类。

除了作为应用程序的前端控制器，ActionServlet 示例也负责初始化和清理资源。当控制器初始化时，最初对于“配置”的 init-param（初始化参数，在 web.xml 的<servlet>或<filter>元素中配置，对于 Struts 框架来说，是 Servlet。译者注）来加载应用程序配置信息。之后它会枚举所有的 init-param 元素，查找那些元素名，看看哪个名字以 config/开头。对于每个元素，Struts 框架通过 init-param 来加载指定配置文件，并且指定一个“前缀”值给那个模块的 ModuleConfig 实例，这个实例由几块名在“config/”之后的 init-param 组成。比如，由 init-param config/foo 指定的模块前缀是“foo”。知道这点是非常重要的，因为这就是控制器如何决定给哪个模块处理请求的权限。要访问模块 foo，你可以使用这样的 URL：

```
http://localhost:8080/myApp/foo/someAction.do
```

由控制器生成的每个请求，方法 process(HttpServletRequest,HttpServletResponse)将会被调用。这个方法简单决定哪个模块应该来服务于请求，之后调用那个模块的 RequestProcessor 的 process 方法，传递相同的请求和响应对象。

4.2.1 Request Processor 请求处理器

RequestProcessor 是大部分为每个请求的核心处理发生的地方。从 1.3 版开始，默认的请求处理器（ComposableRequestProcessor）开始使用 Commons 组件 Chain，那是*责任链*模式（CoR）的实现。它被设计成 Strtus 1.2.x 版本无需替换的行为并为请求处理过程带来更大的灵活性和简便的自定义性。处理的每个步骤都被表示为链中的独立命令。通过插入，替换或移除操作，你可以定制 Struts 框架按照*你的*方式来运行。让我们看一*看默认的* ComposableRequestProcessor 链的命令顺序。这个命令类在 org.apache.struts.chain.commands 或 org.apache.struts.chain.commands.servlet 包中。

SelectLocale	为请求选择语言环境，如果某个请求已经选择好了，那么就将它放到请求对象中。
SetOriginalURI	存储请求对象中原请求的 URI。
RequestNoCache	如果合适，设置如下的响应头部：“Pragma”，“Cache-Control”和“Expires”。
RemoveCachedMessages	如果 message 的 isAccessed 方法返回 true，就移除任意存储在 session 的 Globals.MESSAGE_KEY 和 Globals.ERROR_KEY 下的任意 ActionMessages 对象。
SetContentType	如果请求，对所有响应设置默认的内容类型（用可选的字符编码）。
SelectAction	决定和这个路径相关的 ActionMapping。
AuthorizeAction	如果映射有一个相关的角色，保证请求客户端有确定的角色。如果客户端没有，那么就引发错误并终止请求的处理。
CreateActionForm	实例化（如果需要）和这个映射（如果有）相关的 ActionForm 并将它放到合适的范围呢。
PopulateActionForm	如果有，填充和这个请求相关的 ActionForm
ValidateForm	执行和这个请求（如果有）相关的 ActionForm 中的验证（如果请求）。
SelectInput	如果验证失败，选择合适的 ForwardConfig 来返回到输入页面。
ExecuteCommand	如果当前 ActionConfig 配置了，就查找和执行命令链。
SelectForward	如果这个映射代表跳转，跳转的路径由映射来指定。
SelectInclude	为当前的 action 映射选择包含 uri（如果有）。
PerformInclude	如果选择了，包含当前请求中调用这个路径的结果。
CreateAction	实例化由当前 ActionMapping（如果需要）指定类的实例。
ExecuteAction	这是 Action 的 execute 方法会被调用的点。
ExecuteForwardCommand	如果当前 ForwardConfig 配置了，查找和执行命令链。
PerformForward	最终，RequestProcessor 的 process 方法使用 Action 类返回的 ActionForward 来选择下一个资源（如果有）。很多时候，ActionForward 导向显示页面来呈现响应。

在 Struts 1.2.x 版本中，Tiles 处理需要配置框架使用 TilesRequestProcessor 实现。在 Struts 1.3 版本中，基于*链*的请求处理器使用简单设计到配置的 Tiles 来调用额外的 Tiles 命令，并部署 tiles 的 sub-project.jar。下面的命令在 org.apache.struts.tiles.commands 包中。

TilesPreProcessor	意图执行 Struts 1.2.x 版本中 TilesRequestProcessor 责任的命令类。

4.2.2 它们去哪儿了？

在 Struts 1.2.x 版本中，RequestProcessor 使用了一个类似但是不同的消息设置。这里我们提供一个表格来总结那些变化。

这个链试图去枚举（很多）在标准的 org.apache.struts.action.RequestProcessor 类中标准的请求处理，通过执行对应的独立的可组合命令任务来实现。下面的列表定义了在 processXxx 方法和命令之间的交叉引用来执行对应的功能：

processMultipart	集成到 Servlet 和遗留的类
processPath	SelectAction（也可以进行 processMapping）
processException	ExceptionCatcher/ExceptionHandler
processLocale	SelectLocale
processContent	SetContentType
processNoCache	RequestNoCache
processPreprocess	使用 optional="true"来查找命令，它的多次出现可以很容易地被添加来支持额外的在链上的任意点的处理钩而不需要修改标准的定义。
processCachedMessages	RemoveCachedMessages
processMapping	SelectAction（也可以进行 processPath）
processRoles	AuthorizeAction
processActionForm	CreateActionForm
processPopulate	PopulateActionForm
processValidate	ValidateActionForm/SelectInput
processForward	SelectForward
processInclude	SelectInclude/PerformInclude
processActionCreate	CreateAction
processActionPerform	ExecuteAction

4.3 ActionForm 类

ActionForm 代表了和用户交互的 HTML 表单，它们可能是一个或多个页面上的。你需要提供属性和 getter/setter 方法去访问它们并持有表单的状态。ActionForm 可以存储在 session（默认）或请求范围内。如果它们在 session 中，那么实现表单的 reset 方法在每次使用前来初始化表单是很重要的。Struts 框架从请求参数中设置 ActionForm 的属性，并将验证过的 form 发送给合适的 Action 的 execute 方法。

当你对 ActionForm bean 进行编码时，要记住下列原则：

- ActionForm 类本身不需要实现特定的方法。在整体架构中，它被用来识别这些特定 bean 扮演的角色。典型的情况是，一个 ActionForm bean 仅有属性 getter 和 setter 方法，而没有业务逻辑。
- ActionForm 对象也提供一个标准的验证机制。如果你覆盖了“stub”方法，并在标准的应用程序资源中提供了错误消息。那么 Strtus 框架将会自动验证来自表单（使用你编写的方法）的输入。请参考 3.3.2 节自动表单验证来获取详细信息。当然，

你可以忽略 ActionForm 的验证并在 Action 对象中来进行。

- 为表单中出现的每个字段定义属性（还有相关联的 getXxx 和 setXxx 方法）。字段名和属性名根据通常 JavaBean 的约定（请参考 java.beans.Introspector 类的 Javadoc 文档来了解一下这个信息）必须匹配。比如，一个名为 username 的输入字段可以让 setUsername 方法被调用执行。
- 按钮和其它表单中的控件也可以被定义成属性。这当表单提交时，可以帮助来决定哪个按钮或控件被选择了。要记住，ActionForm 是来代表数据项表单的，而不是数据 bean。
- 将 ActionForm bean 考虑成 HTTP 和 Action 之间的防火墙。使用验证方法来保证所有需要的属性都是存在的，而且它们包含合理的数值。ActionForm 如果验证失败将不会把控制权交给 Action 的。

你也可以在表单上放置一个 bean 的实例，使用嵌入的属性引用。比如，在 ActionForm 中可能有一个“customer”bean，之后在展示页面引用属性“customer.name”。这会对应 customer bean 的方法 customer.getName()和 customer.setName(String Name)。请参考 Apache Struts 的标签库开发指南来获取更多关于使用嵌入的语法。

警告：如果在表单中嵌入了存在的 bean 实例，要考虑可以从其中获取的属性。任意 ActionForm 中的公有可以接受单独字符串值的属性可以设置到查询字符串中。放置 bean 是很有用的，它可以在瘦“包装器”中影响业务状态并且只能暴露出需要的属性。包装器也可以提供一个过滤器来确定运行时属性没有设置为适合的值。

4.3.1 DynaActionForm 类

为应用程序每个 form 维护分开确定的 ActionForm 类是很浪费时间的。当所有的 ActionForm 所做的事情是收集并验证简单属性然后传递给业务 JavaBean 时，就更令人沮丧了。

这个瓶颈可以通过使用 DynaActionForm 类来缓解。代替了创建新的 ActionForm 子类和 bean 的每个属性新的 get/set 方法，你可以在 Struts 框架的配置文件中列出它的属性，类型和默认值。

比如，为 UserForm bean 添加下面的代码到 strtus-config.xml 中来存储用户给定的姓名。

```
<form-bean name="UserForm" type="org.apache.struts.action.DynaActionForm">
   <form-property name="givenName" type="java.lang.String" initial="John"/>
   <form-property name="familyName" type="java.lang.String"
        initial="Smith"/>
</form-bean>
```

DynaActionForm 支持的类型包括：

- java.math.BigDecimal
- java.math.BigInteger
- boolean 和 java.lang.Boolean
- byte 和 java.lang.Byte
- char 和 java.lang.Character
- java.lang.Class
- double 和 java.lang.Double
- float 和 java.lang.Float

- int 和 java.lang.Integer
- long 和 java.lang.Long
- short 和 java.lang.Short
- java.lang.String
- java.sql.Date
- java.sql.Time
- java.sql.Timestamp

你也可以指定这些类型的数组（比如 String[]）。你也可以指定确定的 Map 接口的实现类，比如 java.util.HashMap 或 List 接口的实现类，如 java.util.ArrayList。

如果不提供初始值，数字会被初始化为 0 而对象是 null。

在 JSP 页面中使用 Struts 标签库，DynaActionForm 对象的属性可以被引用，就像传统的 ActionForm 对象一样。不论在什么地方 Struts 标签引用“属性”，标签会自动使用 DynaActionForm 的属性，就像传统的 JavaBean 一样。也甚至可以使用 bean:define 来获取 DynaActionForm 的属性。（虽然，你不能使用 bean:define 来**实例化** DynaActionForm，因为它需要使用合适的 dyna-properties（动态属性，译者注）来建立。）

如果使用 Struts JSTL EL 标签库，那么默认情况下引用是不同的。只有传统 ActionForm 对象的属性可以直接通过 JSTL 表达式语言语法来访问。DynaActionForm 的属性必须通过稍有不同的语法来访问。JSTL EL 语法来引用 ActionForm 中的属性可以是这样：

```
${formbean.prop}
```

引用 DynaActionForm 的属性语法是：

```
${dynabean.map.prop}
```

map 属性是 DynaActionForm 的表示包含 DynaActionForm 属性的 HashMap。

DynaActionForm 用来作为一种通用问题的简单解决方案：ActionForm 使用简单属性和标准验证，你只需要传递这些属性给另外一个 JavaBean（也就是使用 BeanUtils.copyProperties(myBusinessBean,from)方法）。

DynaActionForm **不是** ActionForm 的简单替换。如果你在 Action 中需要访问 DynaActionForm 属性，那么需要使用 map 风格的访问期，比如 myForm.get(“name”)。如果你在 Action 中活跃地使用 ActionForm 对象，那么你会想使用传统的 ActionForm。

DynaActionForm 不能使用无参数的构造方法来实例化。为了去模拟额外的属性，在构建过程中涉及到了很多机制。必须依赖于 Struts 框架，通过 ActionMapping 来实例化 DynaActionForm。

如果需要的话，你可以扩展 DynaActionForm 来添加你需要的自定义验证和重置方法。仅仅在 strtus-config.xml 中指定你定义的子类即可。而不能混合传统属性和 DynaProperties。传统的 getter 或 setter 方法在 DynaActionForm 中不能被反射工具发现。

要和 Struts 校验器一起使用 DynaActionForm，指定 org.apache.struts.validator.ValidatorForm（或你自己写的子类）作为 form-bean 的 class。

当然，DynaActionForm 可能支持各种二进制类型，和 html:text 标签使用的属性应该也是字符串属性。

DynaActionForm 将开发人员从维护简单的 ActionForm 中解脱出来。要做更少的维护，可以尝试下面描述的 LazyActionForm 部分。

4.3.2 LazyActionForm 类

Struts 延迟 ActionFrom 包装了 LazyDynaBean（参考 Commons 组件 BeanUtils 的 JavaDoc 文档）。

并不是真的关于这个实现的很多都是 LazyDynaBean 的延迟行为，而且包装 LazyDynaBean 是在父 BeanValidatorForm 中处理的。真正要做的只有一件事情，就是填充索引属性，是一个 LazyDynaBean 在 get(name,index)方法上有关 List 类型。

延迟 DynaBean 提供一些延迟行为类型：

- ***Lazy* property addition 延迟属性添加**-不存在的属性自动被添加。
- ***Lazy* List facilities 延迟 List 的便利**-自动*增加* List 或数组来容纳设置的索引值。
- ***Lazy* List creation 延迟 List 创建**-如果不存在，自动为索引属性创建 List 或数组。
- ***Lazy* Map creation 延迟 Map 创建**-如果不存在，自动为映射属性创建 Map

使用*延迟* ActionForm 就是说你不需要在 struts-config.xml 中定义 ActionForm 的属性了。但也要注意，在请求中的所有东西填充到 ActionForm 中规避了普通的 Struts form 的*防火墙*功能。因此你应该仅*采用*这种你期望的 form 属性来用，而不是绑定填充所有的属性到业务层。

在 struts-config.xml 中预定义属性已经不是必须的了，话虽如此，对于*映射或索引*属性有时还是有用的。比如，如果你想从默认的 HashMap 或索引属性的数组中使用不同的 Map 实现，而不是默认的 List 类型：

```
<form-bean name="myForm"
    type="org.apache.struts.validator.LazyValidatorForm">
    <form-property name="myMap" type="java.util.TreeMap" />
    <form-property name="myBeans"
        type="org.apache.commons.beanutils.LazyDynaBean[]" />
</form-bean>
```

在 struts-config.xml 中定义*索引属性*的另外一个原因是如果你在现正索引属性时使用的验证器没有提交的，那么索引属性会是 null，这就会引起验证失败。在 strtus-config.xml 中预定义它们会导致零长度的索引属性（数组或 List）被初始化，来避免验证器在哪种情况下的问题。

这种验证实现使用了 ActionForm 的*名称*。如果你需要根据*路径*验证版本，那么就可以很容易地根据下面的写法来创建：

```
public class MyLazyForm extends LazyValidatorForm {
    public MyLazyForm () {
        super();
        setPathValidation(true);
    }
}
```

另外一种方法是使用 LazyDynaBean 或直接自定义 LazyDynaBean 的版本，而不是使用这个类，Struts 框架现在可以自动*包装*对象，它们不是 BeanValidatorForm 中的 ActionForm。比如：

```
<form-bean name="myForm" type="org.apache.commons.beanutils.LazyDynaBean">
    <form-property name="myBeans"
     type="org.apache.commons.beanutils.LazyDynaBean[]" />
</form-bean>
```

要获得更多代码示例，可以参考 LazyValidatorForm How-To。

4.3.3 Map 支持的 ActionForm

DynaActionForm 类提供在初始化阶段创建 ActionForm bean 的能力，这基于在框架配置文件中枚举属性列表。而很多 HTML 表达是在请求时动态生成的。因为这些表单的 ActionForm bean 的属性之前并非所有都是已知的，我们需要一种新方法。

Struts 框架允许你设置一个或多个应用程序的属性值作为 Map 而不是传统的原子对象。之后你可以从表单动态字段存储数据到 Map 中。这里给出一个 map 支持的 ActionForm 类的示例：

```
public FooForm extends ActionForm {
    private final Map values = new HashMap();
    public void setValue(String key, Object value) {
        values.put(key, value);
    }
    public Object getValue(String key) {
        return values.get(key);
    }
}
```

在和它对应的 JSP 页面中，你可以使用特殊的符号：mapname(keyname)访问存储在 map 变量 values 中的对象。在 bean 属性名中的括号表示了：

- Bean 的命名属性 mapname 是使用字符串（可能是 Map 支持的）编制索引的。
- 框架应该查找 get/set 方法来用字符串键参数去找正确的子属性的值。当然，当它调用 get/set 方法时，框架会使用括号中的 keyname 值。

这里给出一个简单示例：

```
<html:text property="value(foo)"/>
```

这会调用 FooForm 中的 getValue 方法，并用键值“foo”来查找属性值。使用动态字段名称来创建表单，你应该按照下面的做法进行：

```
<%
    for (int i = 0; i < 10; i++) {
        String name = "value(foo-" + i + ")";
%>
    <html:text property="<%= name %>"/>
    <br/>
<%
    }
%>
```

注意关于名称“value”没有什么特别之处。Map 支持的属性应该被命名为“property”，“thingy”，或其它任意你喜欢的 bean 属性的名称。在相同的 bean 中也可以使用多个 map 支持的属性。

除了 map 支持的属性之外，你还可以创建 list 支持的属性。通过在 bean 中创建用索引的 get/set 方法来实现：

```
public FooForm extends ActionForm {
    private final List values = new ArrayList();
    public void setValue(int key, Object value) {
        values.set(key, value);
    }
    public Object getValue(int key) {
        return values.get(key);
    }
}
```

在展示页面，访问在 list 支持属性中的独立数据项可以使用一个不同的特殊符号：listname[index]。Bean 属性名中的大括号表示了 bean 的命名属性 listname 是编制索引（可能支持 List）的，而框架应该需找 get/set 方法来使用索引参数去查找正确的子属性的值。

Map 支持的 ActionForm 给了你更大的灵活性，它们不支持可用语法的相同范围来使用常规的或 DynaActionForm。你也许在使用支持 map 的 ActionForm 时引用索引或映射属性（可以参考 Struts 标签库相关部分内容）时有困难。当验证器（从 Apache Struts 1.2.1 版开始）也不适用 map 支持的 ActionForm 时是合法的。

4.4 Action 类

Action 类定义了两个方法，在 Servlet 环境中可以执行：

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

因为很多开发团队使用 Struts 框架主要是构建 Web 应用程序，那么很多项目会仅仅使用“HttpServletRequest”版本的方法。非 HTTP 的 execute()方法是提供给非特定使用 HTTP 协议的应用程序的。

Action 类的目的是处理请求，执行 execute 方法，返回 ActionForward 对象来识别控制权应该转向（比如 JSP，Tile 定义，Velocity template，或其它 Action）何处来提供合适的响应。在 *MVC/模型 2* 设计模式下，典型的 Action 类通常会像如下 execute 方法来实现逻辑：

- 验证当前用户 session 的状态（比如，检查用户是否已经成功登录）。如果 Action

类发现用户没有登录，请求可以被转发到登录显示页面并要求输入用户名和密码。因用户尝试“在中间”（也就是说从浏览器的收藏夹进入）进入应用程序这就会发生，或者因为 session 超时，Servlet 容器创建了一个新的。

- 如果验证不完整，验证 form bean 的属性是必须的，如果发现问题，存储合适的错误信息的键作为 request 对象的属性，然后转发控制器回到输入表单，用户可以修正错误。
- 执行所需处理来处理请求（比如在数据库中存储一条记录）。这*可以*由嵌入在 Action 类中的逻辑代码来完成，但是通常应该调用合适的业务逻辑 bean 的方法来执行。
- 更新服务器端的对象，它可以用来创建用户界面（典型的请求范围和会话返回的 bean，基于你需要保持这些项目多长的可用事件）的下一页。
- 返回合适的 ActionForward 对象来标识展示页面并使用它生成响应，这是基于更新后的 bean 的。典型的情况是，你会通过调用收到的 ActionMapping 对象的 findForward 方法（如果使用这个映射的局部逻辑名），或者控制器 Servlet 本身（如果你使用应用程序中的全局逻辑名）获取一个指向这样对象的引用。

在 Apache Struts 1.0 中，Action 调用 perform 方法而不是新的 execute 方法。这些方法使用相同的参数，不同的只是抛出哪种异常。老的 perform 方法抛出 ServletException 异常和 IOException 异常。而新的 execute 方法仅仅抛出 Exception 异常。这个改变可以简化在 Apache Struts 1.1 时引入的声明异常处理特性。

Perform 方法在 Apache Struts 1.1 中仍然可用，不过已经标记为废弃的了。Apache Struts 1.1 方法仅仅调用新的 execute 方法并包装任意异常作为 ServletException 抛出。废弃的 perform 方法在 Apache Struts 1.2 中被移除了。

4.4.1 Action 类设计准则

当对 Action 类进行编码时，要记得下面的设计准则：

- **为多线程环境编写代码**-我们的控制器 Servlet 仅仅创建**一个 Action 的类实例**，而且使用这一个实例来服务所有的请求。因此，需要编写线程安全的 Action 类。遵循相同的准则你可以使用去编写线程安全的 Servlet。这里给出两个通用的准则，可以帮助你编写可扩展的，按成安全的 Action 类：
  - **仅仅使用局部变量**-帮助线程安全编码最重要的原则是在 Action 类中使用局部变量，**而不是实例变量**。局部变量在堆栈中为每个请求线程分配（通过 JVM），所以不用担心共享它们。Action 可以被分解成几个局部方法，所有所需的变量也是作为方法参数传递。这保证了线程安全，当 JVM 在内部处理这样的变量时，使用和调用堆栈相关的单独线程。
  - **保留资源**-作为一个通用的规则，分配稀缺资源并从相同用户（在用户的 session 中）中跨请求对象保存它们可能引起扩展性问题。比如，如果应用程序使用了 JDBC，而且为每个用户分配了一个独立的 JDBC 连接，当站点突然在斜线点展现出来时，那么就会引起一些扩展性问题。应该力争使用池管理和释放资源（比如数据库连接）而不是将控制转发到合适的视图组件-即使调用的 bean 方法抛出了异常。
- **不要抛出它，捕捉它**-用过一个商业站点，在输出信用卡号并点击付款按钮之后，仅仅是在页面有一个推展跟踪信息或抛出的异常？我们说这并不能激发信心。现在是你来处理这些应用程序错误的机会了-在 Action 类中。如果应用程序特定代码抛出了异常，你应该在 Action 类中捕获这些异常，在应用程序的日志中打印它们的

日志（servlet.log(“Error message”,exception)）并返回合适的 ActionForward。

避免创建冗长和复杂的 Action 类是很明智的。如果在 Action 类中嵌入了过多的逻辑，那么你会发现 Action 类就难于理解，维护，也不可能重用了。为了不创建过于复杂的 Action 类，通常一个好的做法是将持久层和“业务逻辑”移动到一个分开的应用层面。当 Action 类变得冗长和过程复杂时，那就是重构应用程序架构的最佳时机，并移除一些逻辑到另外一个概念层面；否则，你可能会留下一个不灵活的应用程序，可能只能在 Web 应用环境中来访问。Struts 框架应该被视作是在应用程序中实现 MVC 的**基础**。Struts 提供有用的控制层，但是它不是构建 MVC 应用的全功能且一应俱全的平台。

4.5 异常处理程序

当 Action 的 execute 方法抛出异常时，你可以定义一个 ExceptionHandler 来执行处理。首先，需要一个扩展了 org.apache.struts.action.ExceptionHandler 类的子类来覆盖 execute 方法。你的 execute 方法应该处理异常并返回 ActionForward 对象来告诉 Struts 框架下面转向哪里。之后在 struts-config.xml 中配置的处理程序：

```
<global-exceptions>
   <exception key="some.key" type="java.io.IOException"
     handler="com.yourcorp.ExceptionHandler"/>
</global-exceptions>
```

这个配置元素表述了 com.yourcorp.ExceptionHandler.execute 将会在任意 IOException 异常由 Action 抛出时被调用。Key 属性是消息资源文件中的 key，它可以用来检索错误消息信息。

如果 handler 属性没有指定，那么默认的处理程序在请求对象属性中存储异常，并用 Globals.EXCEPTION_KEY 作为全局变量的 key。

可能的“exception”元素的属性值如下：

属性	描述	默认值
bundle	和处理程序相关的消息资源文件的 Servlet 上下文属性。默认的属性是由 Globals.MESSAGE_KEY 字符串常量声明指定的值。	org.apache.strtus.Globals.MESSAGE_KEY
className	这个 ExceptionHandler 对象的配置 bean。如果指定了，className 必须是默认配置 bean 的子类。	org.apache.strtus.config.ExceptionConfig
extends	异常处理程序要继承配置信息的名称。	None
handler	异常处理程序的 Java 类的完全限定名。	Org.apache.strtus.action.ExceptionHandler
key	和处理程序消息资源束使用的键名，它可以为这个异常检索错误信息模板。	None
path	资源相对模块的 URI，如果这个异常发生，它会完成请求/响应。	None
scope	访问这个异常的 ActionMessage[org.apache.struts.action.ActionMessage]对象的上下文（“request”或“session”）	request
Type	使用这个处理程序注册的异常类型的 Java 类的完全	None

	限定名。	

你也可以在 action 定义下定义处理程序来覆盖全局异常处理器。

ExceptionHandler 的配置通常是为 java.lang.Exception 所写，所以当任意异常发生并记录异常数据日志时都会调用它。

4.6 PlugIn 类

PlugIn 接口扩展了 Action 的功能，所以应用程序可以很容易的钩住 ActionServlet 的生命周期。这个接口定义了两个方法，init()和 destroy()，这会在应用程序启动和关闭的时候各自被调用。插件 Action 的通用使用方法是当应用程序启动时配置或加载应用程序指定的数据。

在运行时，任何油 init 方法创建的资源将会被 Action 和业务层的类访问。PlugIn 接口允许你串接资源，但是不能提供任何访问它们的特殊方法。很多时候，资源将会存储在应用程序上下文的已知键名中，其它组件可以找打它。

PlugIn 在 Struts 框架的配置文件中使用<plug-in>元素来配置，可以参考 5.2.3 节来获取详细信息。

4.7 ActionMapping 实现

为了成功操作，控制器 Servlet 需要知道一些关于每个请求 URI 如何来映射到适合的 Action 类的信息。所需的内容都封装在了名为 *ActionMapping* 的 Java 类中，最重要的属性如下：

- type-这个映射使用的 Action 实现 Java 类的完全限定名。
- name-在配置文件中定义的，这个 action 会使用的 form bean 的名称。
- Path-选择这个映射的匹配请求的 URI 路径。参考下面的例子，映射的如何进行的还有如何使用通配符来匹配多个请求 URI。
- unknown-如果这个 action 应该配置成这个应用程序默认使用的，那么就设置为 true，来处理所有不被其它 action 处理的请求。在一个应用程序中，仅只能有一个 action 可以定义为默认的。
- validate-如果这个映射中和 action 相关的验证方法应该被调用的话，那么就设置为 true。
- forward-当映射被调用时，控制权传递后，请求 URI 要转向的路径。这也可以来声明类型属性。

4.8 编写 Action 映射

控制器 Servlet 怎么知道你想要的映射是什么样的？那么可以（但是很无聊）编写小的 Java 类仅仅实例化新的 ActionMapping 实例，并调用所有合适的 setter 方法。为了使这个处理更容易，Struts 框架使用 Commons 的 Digester 组件来解析基于 XML 描述的想要执行的映射并创建合适的对象，实例化到默认值。可以参考 Commons 组件的站点来获取更多关于 Digester 的信息。

开发人员的责任就是创建名为 struts-config.xml 的 XML 文件并把它放置到应用程序的 WEB-INF 目录下。这个文档的格式由文档类型定义（DTD）

http://struts.apache.org/1.3.10/dtds/struts-config_1_3.dtd 来维护。这一章节包含了你应该编写的典型的配置元素，作为开发应用程序的一部分。在 struts-config 文件中也可以放置其它元素来定制应用程序。可以参考第五章来获取更多关于 Struts 框架配置文件关于其它元素的信息。

控制器使用了这个文档的一个内部拷贝来解析配置；因特网连接对操作来说不是必须的。

最外层的 XML 元素必须是<struts-config>。在<struts-config>元素之内，有三个重要的元素来用作描述 action：

- <form-beans>
- <global-forwards>
- <action-mappings>

<form-beans>

这部分包含了 form bean 的定义。Form bean 是在运行时用于创建 ActionForm 实例的描述符。对于每个 form bean 都要使用<form-bean>元素，有下列重要的属性：

- name：这个 bean 的唯一标识符，在对应的 action 映射中，这可以用于引用 bean。通常，这也是请求或会话属性中存储这个 bean 的名称。
- type：使用这个 form bean 的 ActionForm 子类的完全限定名。

<global-forwards>

这部分包含了全局转发的定义。转发是 ActionForward 类的实例，可以从 Action 的 execute 方法返回。

这些映射逻辑名来指定资源（通常是 JSP 页面），允许你在应用程序中改变资源而不需改变它的引用。对于每个转发定义可以使用<forward>元素，有下列重要的属性：

- name：转发的逻辑名称。这在 Action 中的 execute 方法中用来转发到下一个合适的资源。比如：homepage。
- path：资源的上下文相关路径。比如：/index.jsp 或/index.do。
- redirect：tue 或 false（默认）。ActionSservlet 是否应该重定向到资源而不是转发？

<action-mappings>

这部分包含了 action 的定义。对于你想定义的每个映射，可以使用<action>元素。大多数 action 元素至少需要定义如下的几个属性：

- path：访问 action 的应用程序上下文相关的路径。
- type：Action 类的完全限定名。
- name：这个 action 使用的<form-bean>元素的名称。

其它常用属性有：

- parameter：“标准”的 Action 传递属性使用的一个通用属性设置值。
- roles：一个用逗号来分隔的列表，用来区分可以访问该映射的用户安全角色。

关于 action 元素中可以使用的元素的完整描述，可以参考配置 DTD 或在线 DTDDoc 文档，还有 ActionMapping 的 JavaDoc 文档。

4.8.1 ActionMapping 示例

下面是一个基于邮件阅读示例程序的 mapping 配置项。邮件阅读应用现在使用 DynaActionForm（动态 ActionForm）。但是在这个示例中，我们会展示一个传统的 ActionForm 来说明通常的工作流程。注意其它所有 action 的配置条目被排除在外了。

```
<struts-config>
    <form-beans>
       <form-bean name="logonForm"
          type="org.apache.struts.webapp.example.LogonForm" />
     </form-beans>
    <global-forwards type="org.apache.struts.action.ActionForward">
       <forward name="logon" path="/logon.jsp" redirect="false" />
    </global-forwards>
    <action-mappings>
       <action path="/logon"
         type="org.apache.struts.webapp.example.LogonAction"
         name="logonForm" scope="request" input="/logon.jsp"
         unknown="false" validate="true" />
    </action-mappings>
</struts-config>
```

首先是 form bean 的定义。“org.apache.struts.webapp.example.LogonForm”类的基本 bean 映射到逻辑名“logonForm”上。这个名称就是作为 form bean 的请求对象属性名使用的。

“global-forwards”部分用来为通用的展示页面创建逻辑名映射。这些转发配置每个都可以通过调用 action 映射实例来用，也就是 mapping.findForward(“logicalName”)。

你可以看到，这个映射匹配路径/logon（真实的情况是，因为邮件阅读示例应用使用扩展映射，在 JSP 页面中指定的请求 URI 将以/logon.do 来结尾）。当收到匹配这个路径的请求时，*LogonAction* 类的实例将会被创建（仅在第一次请求时）并使用。控制器 Servlet 会在请求范围内以键 logonForm 来查找 bean，如果需要，则创建和保存指定的类。

可选的但是很有用的是逻辑“forward”元素。在邮件阅读示例应用中，很多包含局部“success”和/或“failure”转发的 action 是 action 映射的一部分。

```
<!-- 编辑邮件订阅 -->
<action path="/editSubscription"
    type="org.apache.struts.webapp.example.EditSubscriptionAction"
    name="subscriptionForm" scope="request" validate="false">
    <forward name="failure" path="/mainMenu.jsp"/>
    <forward name="success" path="/subscription.jsp"/>
</action>
```

仅仅使用这两个额外的属性，Action 类几乎是完全独立于展示页面的真实名的。页面可以在重新设计时被重命名（比如），在 Action 类本身的影响却是微不足到的。如果“下一个”页面的名字很难进行编码成 Action 类，而这些类也需要被改变。当然，你可以为应用程序定义逻辑跳转属性。

配置文件包含一些其它元素，你可以使用它们来自定义应用程序，可以参考第五章来获取详细内容。

4.9 为页面使用 ActionMapping

当使用模块时，使用 ActionMapping 来转向页面是*必须*的。因为这样做是可以在请求中

调用控制器的唯一办法--而且是你想做的！控制器将应用程序的配置放置到请求对象中，这对所有的指定模块的配置数据都是可用的（包含你使用的消息资源，请求处理器等等）。

编写这样功能的最简单方法是使用 ActionMapping 的 forward 属性：

```
<action path="/view" forward="/view.jsp"/>
```

4.10 在 ActionMapping 中使用通配符

[从 Apache Struts 1.2.0 开始]应用程序的规模在增大，所以 action 映射的数量也会增多。通配符可以用于合并类似的映射到一个通用的映射中。

解释通配符的最好方法是展示一个示例并来看看它是如何工作的。这个示例修改了之前 4.8.1 节中的映射部分来使用通配符匹配所有以/edit 开始的页面：

```
<!-- 通用的 edit*映射 -->
<action path="/edit*"
    type="org.apache.struts.webapp.example.Edit{1}Action"
    name="{1}Form" scope="request" validate="false">
    <forward name="failure" path="/mainMenu.jsp"/>
    <forward name="success" path="/{1}.jsp"/>
</action>
```

在路径中的“*”允许映射匹配请求 URI，如/editSubscription，/editRegistration 或其它任意以/edit 开头的 URI，而/editSubscription/add 不能匹配上。URI 的一部分被通配符匹配之后，会代入 action 映射的各种属性，它的 action 转发替换为{1}。对于其它的请求，Struts 框架将会参考包含新值的 action 映射和 action 转发。

基于请求的映射匹配以它们在框架配置文件中出现的顺序为序。如果多于一个模式匹配，那么最后一个执行，所以匹配字符少的必须出现在匹配字符多的前面。如果请求 URL 可以匹配没有通配符的路径，那么没有通配符的匹配会被使用，这时顺序就不起作用了。

通配符可以包含一个或多个如下的特殊记号：

*	匹配 0 个或多个字符，不包括斜线（'/'）字符。
**	匹配 0 个或多个字符，包括斜线（'/'）字符。
\字符（character）	反斜线字符用作转义序列。因此*匹配字符型号（'*'），而\\匹配字符反斜杠（'\'）。

在 action 映射和 action 转发中，通配符匹配值可以用记号{N}来访问，N 值是从 1 到 9 表示通配符代替的匹配值。完整的请求 URI 可以使用记号{0}来访问。

Action 映射属性可以接受的匹配通配符字符串是：

- attribute
- catalog
- command
- forward
- include
- input
- multipartClass
- name
- parameter

- prefix
- roles
- suffix
- type

而且，action 映射属性（使用<set-property key="foo" value="bar">语法来设置）会在它们的 value 属性中接受通配符匹配字符串。

Action 转发属性可以接受的匹配通配符字符串是：

catalog

command

path

像 action 映射一样，action 转发属性（使用<set-property key="foo" value="bar">语法来设置）会在它们的 value 属性中接受通配符匹配字符串。

4.11 通用日志接口

Struts 框架本身没有配置日志--它都是通过 Commons 的 logging 组件在后台来完成的。默认的算法是搜索：

如果 Log4J 可用，就使用它。

如果 JDK1.4 可用，就使用它。

否则，使用 SimpleLog。

commons-logging 接口对很多不同的日志实现是一个超薄的桥接。这样做的目的是在任意单独日志实现中去除编译和运行时依赖。要获取更多关于当前支持的实现，可以参考 org.apache.commons.logging 包的描述。

因为 Struts 框架使用了 commons-logging，因此包含了必要的 JAR 文件，有可能你会有一闪而过的想法，*"我应该使用 commons-logging 吗？"*。答案（惊喜！）是要基于特定项目的需求了。如果其中的需求之一是改变日志实现而不对应用程序产生影响，那么 commons-logging 就是一个非常好的选择。

"很好！那我该怎么在代码中开始使用 commons-logging 呢？"

在代码中使用 commons-logging 是非常简单的-你所需要的操作仅仅是引入和声明两个重要的日志记录器。让我们来看一下：

```
package com.foo;
// ...
import org.apache.commons.logging.Log;
import org.apache.commons.logging.LogFactory;
...
public class Foo {
        // ...
        private static Log log = LogFactory.getLog(Foo.class);
        // ...
        public void setBar(Bar bar) {
                if (log.isTraceEnabled()) {
                        log.trace("Setting bar to " + bar);
                }
                this.bar = bar;
        }
        // ...
}
```

通常的做法是为每个类实例化一个单独的日志记录器，给记录器一个反射了在哪里使用的类的名称。示例就是使用类本身来构建。这就给记录器了 com.foo.Foo 的名称。按这种方式来做，可以让你很容易的看到输出是从哪里来的，所以你可以很快定位出问题的区域。此外，你也可以以非常精细的方式来开启/关闭日志记录。

比如在框架类中使用日志记录器的示例，参考上面邮件阅读示例程序的 Action 类。

第五章 配置应用程序

"An author doesn't necessarily understand the meaning of his own story better than anyone else."

作者不一定要比其他读者更好地理解他自己故事的意义。

5.1 概述

在你构建应用程序之前，你需要夯实基础。有一些安装任务你需要在部署应用程序之前来执行。这些包含了在配置文件和 Web 应用程序部署描述符中的组件。

5.2 配置文件

4.8 节的编写 Action 映射部分包含了编写配置文件的`<form-bean>`和`<action-mapping>`部分。这些元素通常在应用程序的开发中扮演了很重要的角色。配置文件中的其它元素基本上就是静态的了：将它们设置一次就不需要再做修改了。

这些“静态”的配置元素是：

- <controller>
- <message-resources>
- <plug-in>

5.2.1 控制器配置

`<controller>`元素允许你配置 ActionServlet。很多在 web.xml 文件中控制器参数都由 Servlet 初始化参数预定义了，但是也可以移入 `struts-config.xml` 这个部分，这就可以在相同的 Web 应用中配置不同的模块。要获取关于可用的参数详情可以参考 struts-config DTDDoc 或者下面的列表。

- **bufferSize 缓冲大小**-当处理文件上传时使用的输入缓冲的大小（字节为单位）。[4096]（可选的）
- **catalog 目录**-当为这个模块处理请求时使用的目录名称。[struts]
- **className 类名**-配置 bean 的类名。[org.apache.struts.config.ControllerConfig]（可选的）
- **command 命令**-执行处理请求的命令名称。[Servlet 标准]
- **contentType 内容类型**-对每个响应设置的默认内容类型（可选的字符编码）。可以被请求转向的 Action，JSP 或其它资源重写。[text/html]（可选的）
- **forwardPattern 转发模式**-替换模式了`<forward>`元素的“path”属性是如何映射到上下文相关的以斜线开头的 URL 的。这个值可以包含如下任意的组合：
  - *$M*-由这个模块的模块前缀替换。
  - *$P*-由被选择的<forward>元素的“path”属性替换。
  - *$$*-呈现文本的美元符号。

- *$x*-（“x”是上面没有定义的任意字符）默默地吞掉，留作将来使用。

如果没有指定，那么默认的转发模式是由之前转发的行为来构成的。[MP]（可选的）

- **inputForward 输入模式**-如果你想<action>元素的 input 属性是局部或全局的 ActionForward 名称，那么就设置成 true，这会用来计算最终的 URL。设置为 false 将<action>元素的 input 参数视作资源模块相关路径用于输入表单。[false]（可选的）
- **locale 语言环境**-如果你想在用户的 session 中存储当前不存在的 Locale 对象，那么就设置为 true。[true]（可选的）
- **maxFileSize 最大文件大小**-接受上传文件的最大文件大小（字节）。可以作为数字表达式，后面跟着“K”，“M”或“G”，这会被解释成千字节，兆字节，或千兆字节。[250M]（可选的）
- **memFileSize 最大内存文件大小**–在上传之后内容保留在内存中的文件的最大大小（字节）。文件大小超过一个限度时将会被写入到一些存储介质上，通常是硬盘。可以作为数字表达式，后面跟着“K”，“M”或“G”，这会被解释成千字节，兆字节，或千兆字节。[256K]
- **multipartClass 多类型类名**-在该模块中使用的多类型请求处理程序类的 Java 类的完全限定名。[org.apache.struts.upload.CommonsMultipartRequestHandler]（可选的）
- **nocache 不缓存**-如果你想让控制器添加 HTTP 头部信息设置从该模块出来的响应为不缓存，那么就设置为 true。[false]（可选的）
- **pagePattern 页面模式**-定义自定义标签的 page 属性如何使用它映射到对应资源的上下文相关的 URL 的替换模式。这个值可以包含下列任意的组合：
  - *$M*-由这个模块的模块前缀替换。
  - *$P*-由被选择的<forward>元素的“path”属性替换。
  - *$$*-呈现文本的美元符号。
  - *$x*-（“x”是上面没有定义的任意字符）默默地吞掉，留作将来使用。

如果没有指定，那么默认的页面模式是由之前的 URL 计算行为来构成的。[MP]（可选的）

- **processorClass 处理器类**-用于这个模块的 RequestProcessor 子类的 Java 类的完全限定名。[org.apache.struts.chain.ComposableRequestProcessor]（可选的）
- **tempDir 临时目录**-当处理文件上传时使用的临时的工作目录。[{由 Servlet 容器提供的目录}]

这个示例使用了一些控制器参数的默认值。如果你仅仅想要默认的行为，你可以完全忽略这些控制器的配置部分。

```
<controller processorClass="org.apache.struts.action.RequestProcessor"
      contentType="text/html"/>
```

5.2.2 消息资源配置

Struts 框架被构建来支持国际化（I18N）。你可以为应用程序定义一个或多个<message-resources>元素；模块自身可以定义它们自己的资源束。不同的资源束可以在应用程序中同时使用。“key”属性是用来指定想要的资源束的。

- **className 类名**-配置 bean 的类名。[org.apache.struts.config.MessageResourcesConfig]（可选的）

- **factory 工厂**-MessageResourcesFactory 类的类名。[org.apache.struts.util.PropertyMessageResourcesFactory]（可选的）
- **key 键名**-存储这个资源束的 Servlet 上下文属性的键名。[org.apache.struts.action.MESSAGE]（可选的）
- **null 空**-设置为 `false` 时，在应用程序中展示不存在的资源的键如“???keyname???”而不是 `null`。[true]（可选的）
- **escape 转义**-如果*转义处理*应该对错误消息字符执行，那么就设置为 `true`。[true]（可选的）
- **parameter 参数**-资源束的名称（必须的）

配置示例：

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

这会在文件 `MyWebAppResources.properties` 的默认键下创建消息资源束。不存在的资源键名会被展示为“???keyname???”。

默认的 `PropertyMessageResources` 实现可以在三种模式中操作：

- **Default 默认模式**-默认的，向后兼容的 Struts 行为（也就是通常使用的方式）。
- **JSTL JSTL 模式**-兼容 JSTL 查找消息。
- **Resource 资源模式**-兼容 Java 的 `PropertyResourceBundle` 来查找消息。

模式可以在 struts-config.xml 中来配置（要获取详情可以参考 `PropertyMessageResources` 的 JavaDoc 文档）：

```
<message-resources parameter="MyWebAppResources">
   <set-property key="mode" value="JSTL"/>
</message-resources>
```

5.2.3 插件配置

Struts 的插件可以在配置文件中使用`<plug-in>`元素来配置。这个元素仅有一个合法的属性，就是“className”，这是实现了 `org.apache.struts.action.PlugIn` 接口的 Java 类的完全限定名。

插件需要自身来配置，嵌套的`<set-property>`元素是可用的。

这是一个使用了 Tiles 插件的示例：

```
<plug-in className="org.apache.struts.tiles.TilesPlugin">
   <set-property property="definitions-config"
        value="/WEB-INF/tiles-defs.xml"/>
</plug-in>
```

5.3 为模块来配置应用程序

要利用模块特性的优点，需要的东西很少。只需要如下的几个步骤：

1.为每个模块准备配置文件。

2.通知控制器这些模块的配置。

3.使用 Action 来引用页面。

5.3.1 模块配置文件

回到 Struts 1.0 版，很少的“引导-启动”选项可以放置在 web.xml 文件中，大量的配置信息可以使用单独的 struts-config.xml 文件来完成。很显然，这不并适用于团队开发环境，因为多用户共享同一个配置文件是不现实的。

从 Struts 1.1 版开始，就有两个选择了：可以列出多个 struts-config 文件作为逗号分隔的列表，或者细分应用程序到不同的模块中。

随着模块化的到来，给定的模块有它自己的配置文件。这就意味着每个开发团队（每个模块相比仅又一个单独的团队来开发）都有它们自己的配置文件，那么当要去修改配置文件的时候就应该少了很多的争论。

5.3.2 通知控制器

从 Struts 1.0 版开始，可以列出应用程序的配置文件作为 web.xml 中 action servlet 的初始化参数。这在 1.1 版本中也是可以的，但是参数就可以扩展了。为了告诉模板关于不同模块的运行机制，你应该指定多个“config”初始化参数，这有轻微的变化。你仍然可以使用“config”来告诉 ActionServlet 关于“默认”模块的信息，但对于每个附加的模块，要列出名为“config/module”的初始化信息，而/module 就是模块的前缀（这当决定 URL 进入哪个给定的模块时会起作用，所以要选择一些有意义的信息!）。比如：

```
...
<init-param>
    <param-name>config</param-name>
    <param-value>/WEB-INF/conf/struts-default.xml</param-value>
</init-param>
<init-param>
    <param-name>config/module1</param-name>
    <param-value>/WEB-INF/conf/struts-module1.xml</param-value>
</init-param>
...
```

这里我们定义了两个模块。一个是“default”模块，由参数名“config”来标识，另外一个使用了基于给定参数名（“config/module1”）的模块前缀“/module1”。控制器被配置来在/WEB-INF/conf/（推荐放置所有配置文件的地方）下查找各自的配置文件。非常简单！

（struts-default.xml 和传统调用 struts-config.xml 是一样的。因为喜欢对称性，所以所有的模块配置文件都以 struts- *module.xml* 来命名。）

如果你想更改每个模块存储的页面，请参考 5.2.1 节中控制器的 forwardPattern 设置。

5.3.3 交换模块

从一个模块切换到另外一个模块有三种办法。

1.你可以使用附加 JAR 包中的 org.apache.struts.actions.SwitchAction。

2.你可以使用<forward>（全局或局部）来指定和上下文相关的属性的 true 值。

3.你可以指定“模块”参数作为 Struts JSP 超链接标记（包括 Img，Link，重写，或转发）的一部分

你可以这样来使用 org.apache.struts.actions.SwitchAction：

```
...
<action-mappings>
   <action path="/toModule"
        type="org.apache.struts.actions.SwitchAction"/>
...
</action-mappings>
...
```

现在，要变到 ModuleB，我们可以使用这样的 URI：

```
http://localhost:8080/toModule.do?prefix=/moduleB&page=/index.do
```

如果你使用“default”模式和“named”模式（比如“/moduleB”），你可以使用这样的 URI 换回“default”模式：

```
http://localhost:8080/toModule.do?prefix=&page=/index.do
```

这是一个全局转发的示例：

```
<global-forwards>
   <forward name="toModuleB" contextRelative="true"
       path="/moduleB/index.do" redirect="true"/>
  ...
</global-forwards>
```

你也可以使用声明在 ActionMapping 中的局部转发来做相同的事情：

```
<action-mappings>
    <action ... >
        <forward name="success" contextRelative="true"
            path="/moduleB/index.do" redirect="true"/>
  </action>
  ...
</action-mappings>
```

在超链接标签中使用模块参数也很简单：

```
<html:link module="/moduleB" path="/index.do"/>
```

这就是全部的内容了，希望能灵活运用模式-转换！

5.4 Web 应用程序部署描述符

创建应用程序的最后一个步骤是配置应用程序的部署描述符（存储在文件

WEB-INF/web.xml 中）来包含所有所需的框架或标签库组件。对示例应用程序使用部署描述符作为指南，我们来看看下面需要创建或修改的配置项。

5.4.1 配置 ActionServlet 实例

添加一个 action servlet 的配置项，还有合适的初始化参数。这样的配置项就像：

```
<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>
    <load-on-startup>1</load-on-startup>
</servlet>
```

Action servlet 支持的初始化参数下面会来描述。（你也可以在 ActionServlet 类的 JavaDoc 文档中来查找详细内容）如果你不提供初始化参数的值，那么方括号描述了假设的默认值。

- **chainConfig** -上下文相关或来加载 Commons 中的 chain 组件定义的类加载器路径，这是一个逗号分隔的列表。如果什么都没指定，默认的分类会使用 Struts 框架提供的。（从 1.3 版开始）
- **config** -包含默认模块配置信息的 XML 资源上下文相关的路径。这也可能是一个逗号分隔的配置文件列表。每个文件按顺序加载，它的对象追加到内部数据结构上。[/WEB-INF/struts-config.xml]

  警告-如果你在多个配置文件中定义了相同名称的对象，那么默认使用最后一个。

  警告-在相同模块中，插件不能加载多于一次。模块配置文件的集合应该仅加载一次插件。
- **config/${module}** -包含用于确定前缀（/${module}）的应用程序模块配置信息 XML 资源的上下文相关路径。对于多个应用程序模块，这可以被重复所需的次数。（从 1.1 版开始）
- **configFactory** – ModuleConfigFactory 的 Java 类名，用来创建 ModuleConfig 接口的实现。（从 1.3 版开始）[org.apache.struts.config.DefaultModuleConfigFactory]
- **convertNull** – 当填充 form 时，1.0 版行为的强制模拟。如果设置为“true”，数字的 Java 包装类类型（比如 java.lang.Integer）将会默认设置为 null（而不是 0）。（从 1.1 版开始）[false]
- **rulesets** – 附加的 org.apache.commons.digester.RuleSet 实例的类完全限定名的逗号分隔的列表，它应该被加入到 Digester 中，来处理 struts-config.xml 文件。默认情况下，处理标准配置元素的 RuleSet 被加载。（从 1.1 版开始）

validating – 我们应该使用验证 XML 解析器来处理配置文件（强烈建议）吗？[true]

警告：如果你为控制器 Servlet 定义了多于一个<servlet>元素或标准控制器 Servlet 的

子类，Struts 框架将不会正确处理。控制器 Servlet **必须**是 Web 应用程序中单例运行的。

5.4.2 配置 ActionServlet 映射

注意：本节的材料并不特定于 Struts。Servlet 映射的配置是定义在 Java Servlet 规范中的。本节描述了配置应用程序的最常见的方法。

定义控制器 Servlet 处理的 URL 有两种常用的方法—前缀匹配和扩展匹配。每种方法合理的映射项会在下面描述。

前缀匹配就是说你想所有的 URL 以特定值开始（在上下文路径部分之后）的会被 Servlet 处理。比如这样的配置项：

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

这就是说请求 URI 匹配路径`/logon`就可以是这样的：

```
http://www.mycompany.com/myapplication/do/logon
```

而`/myapplication`是应用程序部署的上下文路径。

另外一方面就是扩展映射，匹配请求 URI 到 action servlet 是基于 URI 结尾部分的字符串。比如，处理 JSP 的 Servlet 是映射`*.jsp`格式的，所以处理每个请求的 JSP 页面时它都会被调用。使用`*.do`扩展（这也暗示了要“做一些事情”），映射配置项就会是这样：

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

请求 URI 匹配路径`/logon`就可以是这样的：

```
http://www.mycompany.com/myapplication/logon.do
```

警告-如果你为控制器 Servlet 定义了多于一个的`<servlet-mapping>`元素，Struts 框架就不能正确处理了。

警告-如果你使用了从 1.1 版开始的新的模块支持的写法，那么应该小心**仅有**扩展映射是支持的。

5.4.3 配置 Struts JSP 标签库（Servlet 2.3/2.4）

Servlet 2.3 和 2.4 规范简化了标签库配置的开发。现在所需的仅仅是安装 Struts 标签库的 JAR 文件`struts-taglib.jar`到`/WEB-INF/lib`目录下，之后引用标签库的代码就像这样：

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

要注意必须在定义各种 tld（参考上面的示例）时使用完全的 uri，这样容易就可以知道去哪里查找标签的类文件。而不需要去修改`web.xml`文件或拷贝 tld 文件到任何应用程序的

目录中。

当然，用于老式容器的配置技术也会起作用。

5.5 向应用程序中添加框架组件

要使用 Struts 框架，必须复制 `struts-core-*.jar`（还有所有的 `commons-*.jar` 文件）到 `WEB-INF/lib` 目录中。要使用 Struts 标签库，还必须复制 `struts-taglib-*.jar`。同样，要使用其它可选的组件，比如 Extras，Tiles 或验证器，要复制响应的 `struts-*.jar` 文件。

补充：跨 Web 应用程序共享 JAR 文件

很多 Servlet 容器和应用服务器提供了在多个 Web 应用程序中共享 JAR 文件的便利。比如，Tomcat 4.1 允许你在`$CATALINA_HOME/shared/lib` or `$CATALINA_HOME/common/lib` 目录中放置 JAR 文件，那些 JAR 文件在所有的应用程序中就都可用了，而不需要单独放置在每个应用程序的`/WEB-INF/lib` 目录中了。通常情况下，共享是由创建独立的类加载器，它是每个独立应用程序类加载器（由容器创建）的父类完成的。

如果你有基于 Struts 框架的多个 Web 应用程序，考虑利用容器特性优点，放置 `struts.jar` 和 `commons-*.jar` 多个文件在共享目录中而不是在每个 Web 应用程序中，这点是很不错的。然而这种方式也有一些潜在和实际的问题：

- 从共享类加器中加载的类无法识别 Web 应用程序类加载器中的类，除非将程序特殊编写为使用线程上下文的类加载器。比如，Struts 框架会动态地加载 action 和 form bean 类，通常无法找到那些类。Struts 框架被设计成在大多数情况下是可以解决这些问题的，但对*所有情况*并不能完全保证好用。Struts 框架使用的 Commons 组件**没有**审查去捕捉可能发生问题的情况。
- 当类从共享的类加载器中加载时，类中的静态变量就编程全局的了。当底层代码假设静态变量在一个特定的 Web 应用程序（如果类从 Web 应用的类加载器中加载时这可能就是真的）中都是全局变量时，这可能会引发内部 Web 应用的冲突。框架和其依赖的 Commons 组件使用静态变量在维护单独 Web 应用程序中假定是可见的信息，这样的例子是很多的。共享 JAR 文件可以引发很多不期望的交互，也可能会引起不正确的做法。

当 JAR 文件像这样共享时，更新一个 Web 应用程序使用的 JAR 文件版本而不是所有的几乎是不可能的。此外，因为更新 Struts 版本通常需要重新编译应用程序，那么你不得不去重新编译所有的应用程序，而不是去管理它们的依赖。

尽管存在这些困难，共享 Struts 和 Commons 组件的 JAR 文件*也许*会有用。而这是**不**支持的配置方式。

如果你为 `ClassNotFoundException` 或 `NoClassDefFoundError` 异常发送错误报告，或其它相似的加载了错误版本类的情况，错误报告可能**不会**被处理，直到存在问题的 JAR 文件在它们推荐的位置，在 Web 应用程序的`/WEB-INF/lib` 子目录下。

5.6 日志

从 1.0 版本开始，日志功能就是相当限制的了。你可以使用 Servlet 初始化参数来设置

调试详情级的日志，所有日志消息就被写入由 Servlet 容器发出的 `ServletContext.log()` 输出。而到 Struts 1.1 版时，所有的日志消息都由框架本身来书写，还有它使用的 Commons 组件的日志，流经一个抽象的称为 Commons logging 的包装器，它可以用作任何日志实现的包装器。最常用的实现是将见到日志写到 `System.err`，Apache Log4J 日志包，或内建的 JDK 1.4 日志功能或后来的 java.util.logging 包。

这一部分并不想展开解释如何配置和使用 Commons 的 Loggin 组件。相反，我们关注于使用 Commons 的 Logging 组件连接 Struts 框架的有关细节。要获取使用 Commons 的 Logging 组件的完整文档，要咨询现在正在使用的日志系统的文档，附加 Commons 的 Logging 组件的 Javadoc 文档。

Commons 的 Logging 组件提供了由 `Log` 接口创建的日志信息的细粒度控制。按照管理，对框架（通常是 Commons 组件包）使用的 `Log` 实例用要记录日志消息类的完全限定名来命名。因此，由类 `RequestProcessor` 创建的日志消息，很自然，定向到日志记录器命名为 `org.apache.struts.action.RequestProcessor`。

这种方法的优点是你可以配置你想从哪个类获得输出的详细程度。但为每个可能的类维护这样的配置也是一个必须的负担，所以日志环境支持日志等级的概念。如果对特定类的详细成都的配置没有设置，那日志系统就会查找层次直到发现使用配置设置，或者如果没有为任何层次级别设置明确的配置，那么就使用默认的详细程度。在 `RequestProcessor` 类的消息例子中，日志系统会从如下记录器中寻找明确地设置，按这个顺序，直到发现：

- `org.apache.struts.action.RequestProcessor`
- `org.apache.struts.action`
- `org.apache.struts`
- `org.apache`
- `org`
- 日志实现的默认的日志细节程度

以相似的做法，从 `PropertyUtils`（来自 Commons 的 BeanUtils 组件库）类中消息的细节程度设置为按如下顺序查找配置信息：

- `org.apache.commons.beanutils.PropertyUtils`
- `org.apache.commons.beanutils`
- `org.apache.commons`
- `org.apache`
- `org`
- 日志实现的默认的日志细节程度

你可以无缝地从你自己的组件中集成框架和 Commons 组件包使用的相同日志实现，可以按照 4.11 节中介绍的内容来进行。如果你这样做了，那么强烈鼓励你为了最大的配置灵活性而遵循日志记录器（基于要记录日志消息的类名）的相同命名约定。

要获取把它们整合在一起的详细内容，可以参看 FAQ 中如何构建应用程序的内容。

第六章 其它

6.1 发布说明

1.3.10 版

介绍

这个部分包含了 Struts 中发生改变部分的发布说明，是从 1.3.9 版发布以来的变化。要保持框架所有的更新，可以订阅邮件列表（commits@struts.apache.org）。要预览我们未来的更新计划，请参考附录 A 开发路线图。

- **升级注意事项**在 Wiki 的升级页面来维护。Wiki 是一个社区维护的资源-请自由添加你的想法，那样所有人都可以从集体的经验中获益。

对于每个类库的版本要求，请参考 6.2 安装章节。

问题跟踪

Bug

- [STR-2630]标签库文档没有嵌入的 HTML 呈现。
- [STR-2802]当校验器不能在数组两个不同的索引之间测试时校验的问题。
- [STR-3026]由“Planet Struts”托管的项目不存在的连接。
- [STR-3052]Img 标签丢失对 actionId 的支持。
- [STR-3054]EL-示例的错误。
- [STR-3070]使用经典的 RequestProcessor 回发形式引起的 NPE 问题。
- [STR-3076]Passwoard 标签没有定义属性 onselect。
- [STR-3080]XLST 代替了 XSLT。
- [STR-3081]使用 LazyDynaBean 上传文件引起的 ClassCastException 异常。
- [STR-3082]当创建 form bean 失败时的不完整错误消息。
- [STR-3084]Tiles 示例页面的错误。
- [STR-3088] 当 N-长度的空字符串验证时没有被视为*null*
- [STR-3093]标签库 TLD 没有匹配标签实现。
- [STR-3097]生成站点失败。
- [STR-3110] 在 Weblogic Portal 上，ActionServlet.destroy() 方法中调用 PropertyUtils.clearDescriptors()方法时引起应用程序类加载器内存泄漏。
- [STR-3112]生成的 javascript 不能正确在控件数组上设置焦点。
- [STR-3143]Struts 的 jar 保留“in-use”来预防动态重新部署。
- [STR-3144]Bean 标签库文档很长时间都是不可读的。

[STR-3146]TLD 报告含有不可用的超链接。
[STR-3161]在高并发的环境中，Servlet 不能注入到新创建的 action 中。

改进

[STR-1946]html:form 标签的焦点和焦点索引问题。
[STR-1709]Form 标签：添加对不存在的在焦点 javascript 中的 form 字段的检查。
[STR-2810]自动补全属性

任务

[STR-3055]列出 EL 示例中的 disabled="${!empty.pageScope}"。
[STR-3134]更新 JSTL 从 1.0.2 版到至少 1.0.6 版本。
[STR-3150]更新 Commons 的 Chain 组件到 1.2 版本。
[STR-3163]更新 Commons 的 BeanUtils 组件。
[STR-3171]锁定 Maven 插件依赖。
[STR-3172]需要 Tomcat 5.5 来进行整合测试。

6.2 安装

"Would you tell me, please, which way I ought to go from here?"

"That depends a good deal on where you want to get to."

"I don't much care where"

"Then it doesn't much matter which way you go."

"so long as I get somewhere."

"Oh, you're sure to do that, if only you walk long enough."

“你能告诉我我该走哪条路吗？”
“那要看你想去什么地方了。”
“我不太关心是哪里。”
“那么你想走哪条路都没有关系。”
“只要我能到达就好。”
“哦，只要你走得足够远，你确定要那么做。”

必备软件

Struts 框架的二进制发布包需要安装第三方类库来运行。可能在你的系统中已经安装了

这些类库。要从源码开始构建框架，你还需要安装一些其它东西。完整的要求如下：

- **Java Development Kit JDK**-你**必须**为系统平台下载和安装 JDK（1.4 版本或更高）。定位 JDK 发布包的起点是 http://java.sun.com/j2se。要构建 Struts 框架，如果你使用 Maven 来构建系统，Java 1.4.2（当时最新版本的 JDK）推荐使用
- **Servlet 容器**-你**必须**下载并安装兼容 2.3 或更高版本 Servlet API 规范和 1.2 或更高版本 JavaServer Pages（JSP）规范的 serlvet 容器。流行的选择是下载 Apache 的 Tomcat，任何兼容的容器和 Struts 框架都能很好的工作。

Maven 构建系统-（可选的）如果你要从源代码构建 Struts 框架，就必须下载和安装 Apache Maven 2 来构建系统（2.0.4 或更高版本）。

安装类库发布包

首先，可以根据下面的介绍从这里来下载 Struts 的二进制发布包。只要，要确保已经下载和安装了上面介绍的必备软件包。

解压缩类库发布包到一个方便操作的目录下。（如果你从源码发布包构建的话，构建的结果会是解压的二进制发布包。）发布包包含如下内容：

- **lib/*.jar**-struts-*.jar 包含了由 Apache-Struts 发布的 Java 类。其它 JAR 文件包含框架从其它项目引入的包。当运行应用程序时，这些 JAR 应该是可用的，通常要将它们复制到应用程序的 `WEB-INF/lib` 目录下。

  警告-如果要在相同的 Servlet 容器中运行多个应用程序，那么可以将 `struts-core.jar` 文件放置到容器支持的共享资源库下。要知道这可能会引起 ClassNotFoundException 异常问题，除非应用程序的*所有*类都存储在共享资源库中。

要在应用程序中使用 Struts 类库，你可能需要如下的几个步骤：

- 从发布库中复制 `lib/*.jar` 文件到 Web 应用程序的 `WEB-INF/lib` 目录下。
- 修改 Web 应用程序的 `WEB-INF/web.xml` 文件，包含一个`<servlet>`元素来定义控制器 Servlet，还需要一个`<servlet-mapping>`元素来创建请求 URI 到这个 Serlvet 的映射关系。使用 Struts 邮件阅读器示例程序中的 `WEB-INF/web.xml` 文件来参考必须语法的详细示例。
- 创建一个 `WEB-INF/struts-config.xml` 文件来定义应用程序的 action 映射和其它特性。你可以使用从 Struts 示例应用的空白应用下的 `struts-config.xml` 文件来获取必须的语法定义。
- 在每个 JSP 页面的顶部可以使用 JSP 标签，在特定页面加入如下声明的 JSP 标签库：

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

- 当编译包括应用程序的 Java 类时，要保证包含了在类路径下的 JAR 文件（之前复制的），提交给编译器。

和 Servlet 容器一起安装框架

对于大多数的容器，你只需这么来做：

- 复制`/webapps` 目录中 WAR 文件到到容器的 `webapps` 目录下。
- 在某些情况下，如果容器正在运行，你可能需要重新启动容器。

在安全管理器下运行应用程序

很多应用服务器在 Java 安全管理器的控制下，使用严格授权 Web 应用程序可以执行什么类来执行 Web 应用程序。如果你和映射的属性来使用 form bean，可能会遇到完全异常，除非将下列授权加入到应用程序代码库的授权设置中：

```
permission java.lang.RuntimePermission
"accessDeclaredMembers";
```

可以参考应用程序服务器的问题来获取更多关于如何配置其它安全管理器授权的详情。

在各种容器上安装

- Bluestone Universal Business Server 7.2-参考附录 B1
- Borland Application Server 4.5-不需要其它步骤
- iPlanet Application Server-建议使用 Service Pack 2。要注意 Struts 邮件阅读示例程序的数据库对象在这个容器中是不兼容的
- iPlanet Web Server-参考附录 B2
- iPortal Application Server-参考附录 B3
- Jetty-参考附录 B4
- JRun-参考附录 B5
- Novell ExteNd Application Server 4.0+ -参考附录 B6
- Orion Application Server-参考附录 B7
- Resin 1.2+“standalone”-不需要其它步骤
- RexIP-不需要其它步骤
- SilverStream 3.7.1 及更高版本-参考附录 B8
- Tomcat 3.1 和之前版本-不推荐，使用 Tomcat 3.2.1 及更高版本
- Tomcat 3.2.1 和 Apache-参考附录 B9
- Tomcat 3.2.1+“standalone”-不需要其它步骤
- Tomcat 4.0 -不需要其它步骤
- Trifork Enterprise Application Server 3.3.x-不需要其它步骤
- Weblogic 5.1 sp8-参考附录 B10
- Weblogic 6.0+ -不需要其它步骤
- WebSphere-参考附录 B11
- WebSphere-示例应用程序的步骤，参考附录 B12

FAQ 和 HOWTO

FAQ

Kickstart FAQ（请先读我）

1.我们为什么需要 Struts?

当创建和维护应用程序来解决当前公共 Web 站点的需求和企业级交互时，Java 技术给开发者了一个严重刺激。Struts 结合了标准 Java 技术成为一个统一的框架。

2. Struts 是如何工作的？

Jave Servlet 被设计用来处理由 Web 浏览器发出的请求。服务器页面被设计来创建动态的 Web 页面，它们可以将广告牌站点转变为可交互的应用程序。Struts 使用了一个特殊的 Servlet 作为总机来路由来自 Web 浏览的请求到合适的服务器页面。在这条路上，称为 Action 的特殊 Java 类可以和数据访问框架或业务逻辑类交互。分层的架构使得企业级 Web 应用程序非常容易创建和维护。（要获得详细信息，可以参考完整版的“Struts 是如何工作的？”-后续的 FAQ）。

3. Struts 兼容其它 Java 技术吗？

是的。Apache Struts 项目是贡献给支持工业标准的。Struts 作为 Java 技术的整合者，当然是可以在“真实世界”中使用的。

4. 谁编写了 Struts?

Apache Struts 项目有一些积极的贡献者，来自世界各地并共同合作。几十个独立开发人员和贡献者贡献了 Struts 1.x 的代码库。有兴趣的开发人员和语言大师被要求来为项目贡献代码。

初始的 Struts 代码库（Struts 0.5）由 Craig R. McClanahan 在 2000 年 5 月创建并捐献给阿帕奇软件基金会。Craig 是 Struts 1.x 和 Tomcat 4 的主要开发者。Tomcat 4 是 Servlet 2.3 和 JSP 1.2 容器的官方参考实现。

作为 JSR 127 规范（JavaServer Faces）的共同领导之后，Craig 创建了另外一个基于 JavaServer Faces 的 Struts 版本，称为 Shale。新框架开始作为 Apache Struts 的自框架，但是之后 Shale 成为 Apeche 的顶级项目并有它自己的独立权力。Apache Shale 对于团队使用 JSF 作为基本技术开发复杂应用程序是一个很好的选择。（Apache Shale 已经下线，现在可以参考 attic，译者注）

5. Struts 是第一个 Java 的 Web 应用程序框架吗？

不，它的历史还不长。当 Struts 1.0 在 2001 年 6 月发布时，已经有了一些其它 Web 应用框架了，包括 Barracuda，Expresso，Maverick，Tapestry 和 Turbine，这里仅举几例。Struts 没有加入“绿色营地”。在那之前，也有在可用的框架直接的比较，现在也是如此。

6. Struts 是 Java 中的流行 Web 应用程序框架吗？

是的，由任何客观的衡量，Struts 继续称为最流行的 Java Web 应用程序框架。

在 2006 年时，Java 开发人员的绝大多数就业机会条件是要会 Struts。根据 OnJava 杂志，在 2005 和 2004 的读者调查中，Struts 的流行度仍保持稳定。同样，Struts 相关文章的发布数量在所有在线期刊的 2004 和 2005 年度中也保持恒定。（要获得完整列表，可以参考 Struts 中心网站）关于 Struts 的新书和修订版图书继续定期发布。

而一些人表征 Java Web 应用程序框架的空间为“支离破碎的”，真实情况是 **Web 开发人员使用 Struts 要比其它组合方案多的多**。这一观察在 2006，2005，2004 年时都是真实的。自从 Struts 在 2001 年发布其 1.0 版本开始就应该是真的了。

当然，一些开发人员发现 JavaServer Faces 可以有一种更快速的方法来编写新的应用程序，特别是适度的联网应用程序。这真是太好了！每个 Struts 的贡献者都想让所有开发人员从日常工作中收获更多，这比什么都重要。（我们也是开发人员！）当这些新的 JSF 应用程序变得复杂时，我们很高兴 Apache Shale 可以为 JSF 开发人员做些什么了，正如 Struts 为 JSP 开发人员所做的一样，直到永远。

与此同时，企业级 Web 开发人员拥有基于 Struts 的标准化开发模式可以保证 Struts 1 的新发布可以继续，即使我们可以用 Struts 2 来开创新的天地。

7. 我能从哪里获得 Struts 框架的拷贝？

下载 Apache Struts 产品的最佳地址是 struts.apache.org。

8. 我该如何来安装 Struts？

要使用 Struts 来开发应用程序，你通常可以在 Java 开发环境中添加 Struts 的 JAR 文件。之后就可以使用 Struts 的类库作为应用程序的一部分。我们提供了一个空白的 Struts 应用程序（在 `webapps` 目录下，打开 `struts-blank.war`），复制之后，你就可以获得一个自己的快速启动开发。

除了 Struts 的 jar 文件，空白应用程序也包含了其它 Struts 的类库 JAR 和依赖。所有依赖都兼容 Apache 许可。

因为 Struts 的完整源代码都是可用的，我们也提供自行编译 Struts 的 JAR 文件的介绍。（使用 Maven，构建 Struts 非常容易！）

Struts 应用程序通常可以使用标准的 WAR 文件来开发。在很多情况下，仅仅在应用服务器上部署 WAR 文件就可以了，它就会自动安装了。如果不可以，各种 Servlet 容器的手把手安装说明也是可以查看的。（请参考 6.2 节和附录 B 部分的介绍）

9. 什么时候我才需要在类路径中加入 Struts 的 JAR 文件？

当你使用 Struts 的类库来编译应用程序时，你必须将 Struts 的 JAR 文件放置到**编译器**可以查看的类路径下-它不需要是类路径环境变量。

为什么是一个很重要的区别？因为如果你在开发机制下使用 Servlet 容器来测试应用程序，当运行容器时，Struts 的 JAR 文件必须不能是类路径环境变量。（这是应为每个 Web 应用程序必须拥有它们自己的 Struts 类库，而如果在环境路径下也有，容器将会混淆它们。）

对于这个问题有一些通用的做法：

- **使用 Ant 或 Maven** 来构建项目-它可以很容易为编译器组合类路径。
- **使用 IDE**，你可以配置“类路径”来编译独立的类路径环境变量。
- **使用 shell 脚本**，暂时添加 struts-action.jar 到类路径下来编译，比如：
  javac -classpath /path/to/struts-action.jar:$CLASSPATH $@

10. Struts 有它自己的测试吗？

当前 Struts 有两个测试环境，来反射实际情况，一些东西可以静态地来测试，而一些需要在运行 Servlet 容器的环境中来完成。

对于静态的单元测试，我们使用 JUnit 框架。这些测试的源代码在源代码库中的“src/test”目录下，可以通过高级 build.xml 中“test.junit”目标来执行。这样的测试关注于独立方法的低级别功能，特别适合 org.apache.struts.util 中工具类的静态方法。在测试 Cecil 中，也有一些“模拟对象”类（在 org.apache.struts.mock 包中），所以你可以打包这些看起来像 Servlet API 和 Struts API 对象的东西来传递给这些测试的参数。

另外一个有价值的工具是 Struts 的测试用例，它提供了一个有用的对 Action 类的工具，可以和 JUnit 或 Cactus 来使用。

Newbie FAQ

1. 如何加载自定义的 chain 配置文件？

使用初始化参数 `chainConfig` 来为 ActionServlet 指定特定的逗号分隔的列表的 chain 配置文件。

这会覆盖 `chainConfig` 的默认值，所以除非你覆盖了默认的请求处理链，你必须在列表中包含默认的 chain 配置文件。

```
<init-param>
    <param-name>chainConfig</param-name>
    <param-value>org/apache/struts/chain/chain-config.xml,
            /WEB-INF/custom-chain-config.xml
   </param-value>
</init-param>
```

另外一个加载 chain 配置文件的方法，可以参考 Commons 的 Chain 组件的 Cookbook：从 Web 应用程序中加载目录。

2. 什么是模块化应用程序？相对模块是什么意思？

从 Struts 1.1 开始，框架支持多个应用程序模块。所有应用程序至少有一个根，或默认的模块。就像文件系统的根目录一样，默认应用没有名字。（或者使用空字符串来命名，这都按你的想法来。）仅用一个默认的模块来开发应用程序和使用 Struts 1.0 来开发应用程序没有什么不同。从 Struts 1.1 开始，你可以添加额外的模块到应用程序中，每个模块可以有它们自己的配置文件，消息资源等。每个模块按照默认模块的相同方式来开发。作为单独模块来开发的应用程序可以添加多模块应用程序，而且模块可以促进独立的应用程序而不需要改变。要获取更多关于配置应用程序来支持多模块的信息，可以参考 5.3 节内容。

但是要回答问题=:0)，一个模块的应用程序是使用了多于一个模块的应用程序。相对模块就是 URI 开始于模块级别，而不是上下文级别，或绝对 URL 级别。

- 绝对 URL：http://localhost/myApplication/myModule/myAction.do
- 相对上下文：myModule/myAction.do
- 相对模块：/myAction.do

Struts 示例应用程序是模块化的应用程序，由一些独立创建的应用程序组装而来。

3. 为什么一些类和元素名不直观？

框架规模一直在增长，因为它的发展，一些命名也漂移不定。

每日构建的一个好处是所有事情可以在编写完成后对开发社区都是可用的。而不好的一面是比如类名很早就被锁定了而之后就很难去改变了。

4. ActionForm 的走向是什么？

http://www.mail-archive.com/struts-user@jakarta.apache.org/msg19281.html

http://www.mail-archive.com/struts-user@jakarta.apache.org/msg19338.html

http://www.mail-archive.com/struts-user@jakarta.apache.org/msg20833.html

5. 为什么 ActionForm 是基类而不是接口？

最初编写 ActionForm 类的想法是利用 Java 单根继承限制的优点来使得它不能让用户做它不能做的事情。当时，EJB 开始流行，很多开发人员开始联合 EJB 远程使用 ActionForm，而最后的结果不令人满意。

从那以后，很多开发人员使用不同的方法来进行数据持久化，也有很多开发人员现在对为什么我们想从视图中分离出模型有了一个很好的理解。因此，我们计划引入更多的接口到框架中去。这只是当时的一个问题。

与此同时，DynaActionForm 缓解了开发人员维护简单 ActionForm 的问题。为了近乎零维护，尝试 LazyActionForm（第四章相关内容）和 Hubert Rabago 的 FormDef。

6. ActionForm 是真正的 JavaBean 吗？

Struts 框架使用的工具类（Commons 的 BeanUtils 组件从 1.1 版开始）需要 ActionForm 属性遵循 JavaBean 规范来编写访问器方法（get*，set*，is*）。因为 Struts 框架的 ActionForm 使用了 Introspection API，一些容易可能需要所有的 JavaBean 规范都要遵循，包括为每个子类声明“`implements Serializable`”（即实现可序列化的接口，译者注）。最安全做法是审查 JavaBean 规范（请参考前言部分内容）之后遵循规定的模式。

从 Struts 1.1 开始，你可以使用 DynaActionForm 和映射支持的 form，它们不是真正的 JavaBean。要获取更多内容，可以参考 4.3 节 Action Form 类和下一部分（Newbie FAQ-7）内容。

7. 我能和 ActionForm 一起使用其它 bean 或 hashmap 吗？

使得。要和 ActionForm 一起使用其它 bean 或 hashmap 是有很多方法的。

- ActionForm 可以有其它 bean 或 hashmap 作为属性。
- “值 Bean”或“数据转换对象”（DTO）可以独立对 ActionForm 使用来转换数据到视图层。

ActionForm 可以使用 Map 来支持“动态”属性（从 Struts 1.1 开始）

ActionForm（又名“form bean”）是真正的 JavaBean（有一些特殊的方法），框架为你创建它们并放置其到会话或请求范围内。并不阻止你使用其它 bean，或在你的 form bean 中来引用它们。让我们来看一些示例。

集合作为属性，假设你需要在应用程序的一个输入表单中展示一个可用颜色的下拉列表。你可以在 `ActionForm` 中包含一个字符串值的 `colorSelected` 属性来代表用户的选择，还有一个 `colorOptions` 属性作为 `Collection`（或字符串）的实现来存储可用的颜色选择。假设你已经为 `orderEntryForm` from bean 中的 `colorSelected` 和 `colorOptions` 属性定义了 getter 和 setter 方法，那么你就可以使用这段代码来呈现下拉菜单了：

```
<html:select property="colorSelected">
    <html:options property="colorOptions" name="orderEntryForm"/>
</html:select>
```

这个列表会使用 `orderEntryForm` 中的 `colorOptions` 集合里的字符串来填充，而用户选择的值会进入到 `colorSelected` 属性中并存储在随后的 `Action` 中。注意这里我们假设 `orderEntryForm` 的 `colorOptions` 属性已经被设置过了。

参考第 12 条（我该如何填充 form）来获取在呈现编辑表单之前如何设置 form bean 属性的介绍，属性要预先设定好。

`Action` 检索开放顺序（比如 `Order` 对象的 `ArrayList`）列表的独立 DTO 可以独立使用任意 form bean 中的 DTO 来转换查询结果到视图层。首先，Action 的 `execute` 方法执行查询过并将 DTO 放置到请求中：

```
ArrayList results =
businessObject.executeSearch(searchParameters);
request.setAttribute("searchResults",results);
```

之后视图可以使用请求对象中的键“searchResults”来引用 DTO 并进行迭代：

```
<logic:iterate id="order" name="searchResults"
    type="com.foo.bar.Order">
    <tr><td><bean:write name="order" property="orderNumber"/><td>
    <td>..other properties...</td></tr>
</logic:iterate>
```

也可以参考：Map 支持的 ActionForm（参考 4.3.3 节相关内容）（从 Struts 1.1 版开始）

8. 我该如何认证用户？

http://www.mail-archive.com/struts-user@jakarta.apache.org/msg24504.html

http://www.mail-archive.com/struts-user@jakarta.apache.org/msg22949.html

9. 我一定要在应用程序中使用 JSP 吗？

对这个问题的简短回答是：不，你不局限于使用 JavaServer Pages。

完整回答是你可以使用任意类型的展示技术，可以由 Web 服务器或 Java 容器返回。下面的列表包括了但不局限于：

- JavaServer Pages
- HTML 页面
- WML 文件
- Java Servlet
- Velocity 模板
- XML/XLST

一些用户仍然混合匹配表面上无关的技术，比如在相同的 Web 应用程序中引入 PHP。

10. ActionForm 是真正的 JavaBean 吗？

ActionForm 添加到 Servlet 范围（会话或请求）中作为 bean。那也就是说，对于某些可用的功能，你的 ActionForm 还需要一些小的简单规则。

首先，ActionForm bean 必须有一个没有参数的构造方法。这是因为 Struts 框架必须能够动态创建这些 form bean 类的新实例，而只知道类名所要求的。这不是繁重的限制，因为框架将会为你填充 form bean 的属性（从请求对象的参数中获得）。

第二，form bean 的字段对框架是可用的，通过提供公有的 getter 和 setter 方法，它们遵循了 JavaBean 规范的命名设计模式。对于大多数用户，这就是说可以为每个 form bean 的属性使用如下的约定俗成的写法：

```
private {type} fieldName;
public {type} getFieldName() {
    return (this.fieldName);
}
```

```
public void setFieldName({type} fieldName) {
    this.fieldName = fieldName;
}
```

注意-你*必须*为 ActionForm 中的可是别属性遵守上面展示的大写约定。在这个示例中的属性名是“fieldName”，这必须和输入字段中的属性名所对应。Bean 的属性可能有一个“getter”方法和一个“setter”方法（在一个 form bean 中，通常是二者都有），它们的名称以“get”或“set”开头，后面跟着首字母大写的属性名。（对于布尔值属性，使用“is”来代替“get”作为 gettter 方法的前缀也是合法的。）

高级的 JavaBean 用户会知道可以告诉系统想去为 getter 和 setter 方法使用不同的名称，可以使用和 form bean 相关的 `java.beans.BeanInfo` 类。通常情况下，遵循标准转换是很方便的。

警告-开发人员可能想去使用下列的技术之一，但它们其中的任意之一可能会引起属性不能被 JavaBean 的自我检查机制所识别，因此会引起应用程序行为混乱：

- *使用的 getter 和 setter 方法名不匹配*-如果你有一个 `getFoo()` 的 getter 方法，而 setter 方法是 `setBar()`，那么 Java 不会识别这些方法来引用相同的属性。相反，语言会认为你有一个只读属性叫“foo”而且有一个只可写的属性叫“bar”。
- *使用多于一个相同名称的 setter 方法*-Java 语言允许你“重载”方法，只要参数类型不同即可。比如，你可以在相同的类中有一个 `setStartDate(java.util.Date date)` 方法和一个 `setStartDate(String date)` 方法，编译后的代码之后哪个方法被调用是基于传递过来的参数类型。对 form bean 的属性这么来做将会阻止 Java 识别“startDate”属性。

如果你想 form bean 可以使用其它特性，那么还有一些其它的规则应该遵守，特别是在索引属性和映射属性方面。特殊的规则覆盖了文档其它部分的细节，特别是，索引属性，映射属性和索引标签。

要获取关于什么是 JavaBean 的完整解释和它可以做的所有事情，请参考 JavaBean 规范（版本 1.01）。

11. 我应该为每个 HTML 表单分离出 ActionForm bean 吗？

这是一个有趣的问题。作为新手，为每个 action 序列创建新的 `ActionForm` 是一个良好的实践。你可以是使用 `DynaActionForm` 来帮助减少所需的努力，或者使用 IDE 的代码生成工具。

要记住关于重用 form bean 的一些问题，比如：

- *验证*-基于当前执行的 action，你可能需要不同的验证规则。
- *持久*-要注意在 action 中填充的 form **不期望**在不同的 action 中重用。`struts-config.xml` 中的多个对相同 `ActionForm` 子类`<form-bean>`配置会有作用（特别是你想在 session 范围内存储 form bean）。还有，在请求范围内存储 form bean 可以阻止不期望的交互（也可以减少应用程序的内存占用，因为没有服务器端的对象需要在请求之前存储。）
- *复选框*-如果你想对布尔值属性（作为复选框提交的字段）做建议和复位，而当前展示用的页面没有对 form bean 的每个布尔值属性都有复选框，没有展示出的布尔值将会一直是 `false` 值。

工作流-form bean 最常见的重用需要是工作流。在沙箱外，Struts 框架对工作流的支持

有限制，但是通常的模式是对工作流页面的所有属性使用单独的 form bean。你可能需要对环境有（ActionForm，Action 等）一个很好的了解，这要在能够将使用单独的 form bean 组件顺利的工作流环境之前进行。

当你感觉更舒适时，有一些快捷方式可以使用来重用你的 ActionForm bean。这些快捷操作中的很多是基于你选择如何实现 Action/ ActionForm 组合的。

12. 如何填充 form？

填充 form 的最简单方式是有一个 Action，它的唯一目的是填充 ActionForm 并转发到 Servlet 或 JSP 中来在客户端呈现 form。单独的之后会用来处理提交的表单字段，通过声明相同 form bean 的名称来实现。

在邮件阅读示例应用中，作为 Struts 应用程序子项目的一部分，很好地说明了这个设计模式。要注意下面从 struts-config.xml 文件中获得的定义：

```
...
<form-beans>
...
<-- 注册 form bean -->
<form-bean name="registrationForm"
    type="org.apache.struts.webapp.example.RegistrationForm"/>
...
</form-beans>
...
<action-mappings>
...
<-- 编辑用户注册-->
<action path="/editRegistration"
    type="org.apache.struts.webapp.example.EditRegistrationAction"
    name="registrationForm" scope="request" validate="false"/>
...
<-- 保存用户注册 Save user registration -->
<action path="/saveRegistration"
    type="org.apache.struts.webapp.example.SaveRegistrationAction"
    name="registrationForm" input="registration" scope="request"/>
...
</action-mappings>
```

注意下列这种方法的特性：

- /editRegistration 和/saveRegistration 两个 action 使用了相同的 form bean
- 当进入/editRegistration action 时，Struts 框架会有一个预先创建的空的 form bean 实例，之后传递给 execute()方法。当表单呈现后创建 action 到预先配置的值，并设置对应的 form bean 属性。
- 当创建 action 完成配置 form bean 的属性后，应该返回 ActionForm，并指向这个 form 要展示的页面。如果你使用了 Struts 的 JSP 标签库，在<html:form>标签中的

action 属性会被设置成/saveRegistration，为了表单可以提交到处理的 action 中。

- 要注意创建 action（/editRegistration）关闭了创建时的表单验证。通常你会想在创建 action 的配置中包含这个属性，因为你不想真正处理结果--而只是想利用 Struts 框架预先为你创建正确的 form bean 实例的优点。

处理 action（/saveRegistration），换句话讲，没有设置 validate 属性，那么默认值就是 true。这会告诉 Struts 框架在调用处理 action 之前来执行相关 form bean 的验证。如果发生了任意的验证错误，Struts 框架会转发回输入页面（从技术上说，它转发回名为“registration”的 ActionForward，因为示例应用在<controller>元素中使用了 inputForward 属性--请参考描述 struts-config.xml 的文档来获取详细信息）而不是调用处理 action。

13. 我可以有不使用 form 的 Action 吗？

当然。如果 Action 不需要任何数据并且也不需要给转发到的视图或控制器组件可用的数据，那么就不主要 form 了。在 Struts 邮件阅读示例程序中，没有 ActionForm 的 Action 的一个很好示例是 LogoffAction：

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

这个 action 不需要数据，除了用户的 session，这也可以从 Request 对象中获取到，也不需要准备任何视图元素来展示什么，所以它不需要 form。

所以，你不能使用<html:form>标记而不使用 ActionForm。及时你想和简单的 action 使用<html:form>标记而也不需要输入，这个标签也期望使用一些类型的 ActionForm，即便是空的没有任何属性的子类。

14. 能给我一些使用 requiredif 校验器规则的示例吗？

首先，有一个新的校验器规则，称为 validwhen，这可能就是你想用的，因为它很简单并且很强大。在 1.1 版之后的首次发布包即可使用。下面展示的示例可以使用 validwhen 来编码：

```
<form name="medicalStatusForm">
<field property="pregnancyTest" depends="validwhen">
<arg0 key="medicalStatusForm.pregnancyTest.label"/>
    <var>
        <var-name>test</var-name>
        <var-value>((((sex == 'm') OR (sex == 'M')) AND
               (*this* == null)) OR (*this* != null))</test>
    </var>
</field>
```

我们假设有一个医疗信息表单，并有三个字段，sex，pregnancyTest 和 testResult。如果 sex 是“f”或“F”，pregnancyTest 是必须的。如果 pregnancyTest 不为空，那么 testResult 是

必须的。在 validation.xml 中的配置项就可以是这样：

```
<form name="medicalStatusForm">
<field property="pregnancyTest" depends="requiredif">
    <arg0 key="medicalStatusForm.pregnancyTest.label"/>
        <var>
            <var-name>field[0]</var-name>
            <var-value>sex</var-value>
        </var>
        <var>
            <var-name>fieldTest[0]</var-name>
            <var-value>EQUAL</var-value>
        </var>
        <var>
            <var-name>fieldValue[0]</var-name>
            <var-value>F</var-value>
        </var>
        <var>
            <var-name>field[1]</var-name>
            <var-value>sex</var-value>
        </var>
        <var>
            <var-name>fieldTest[1]</var-name>
            <var-value>EQUAL</var-value>
        </var>
        <var>
            <var-name>fieldValue[1]</var-name>
            <var-value>f</var-value>
        </var>
        <var>
            <var-name>fieldJoin</var-name>
            <var-value>OR</var-value>
        </var>
    </field>
    <field property="testResult" depends="requiredif">
        <arg0 key="medicalStatusForm.testResult.label"/>
            <var>
                <var-name>field[0]</var-name>
               <var-value>pregnancyTest</var-value>
            </var>
            <var>
                <var-name>fieldTest[0]</var-name>
               <var-value>NOTNULL</var-value>
            </var>
    </field>
```

```
</form>
```

15. 校验输入的最佳时间是什么？

这是一个非常好的问题。我们回想一下典型的中大型应用程序。如果我们从后台工作到视图，我们会有：

1）数据库：很多现代数据库会进行必须字段，重复记录，安全限制等的校验。

2）业务逻辑：这里你可以检查合法的数据关系，为解决特定问题产生意义的东西。

这就是框架活灵活现的地方了，此时系统也应该做了很好的防护。我们要做的就是使验证更友好，信息更丰富。记得有重复的验证也是可以的。

3）`ActionErrors validate(ActionMapping map, HttpServletRequest req)` 就是你可以进行验证并反馈给视图层的地方，所需的信息来修正任意错误。validate 在 form 被 `reset` 和 `ActionForm` 的属性从对应的视图输入设置之后运行。而且要记住你可以在 `action` 映射的 `struts-config.xml` 文件中使用 `validate="false"`将验证关闭。完成后会返回含有从 `ApplicationResources.properties` 文件中获取消息的 `ActionErrors` 集合

这里你可以访问请求对象，所以你可以看到哪种类型的 action 被请求来微调验证。<html:error>标签允许你在页面或关联特定属性的特定错误来转存所有的错误。`struts-config.xml` 文件的 `action` 中的 `input` 属性允许你发送验证错误到指定的 jsp/html/tile 页面。

4）你可以使用 `ValidatorForm` 或它的衍生来执行系统低等级验证或客户端反馈。这会生成 javascript 并给出简单数据项的错误的实际反馈到用户处。你可以在 `validator-rules.xml` 文件中对验证进行编码。要有效地使用特性，正则表达式的知识是必须的。

16. 在数据输入之前如何避免验证表单？

最简单的方法是准备两个 action。第一个 action 的职责是设置表单数据，比如空的注册屏幕。编写的第二个 action 是将注册数据写入数据库。框架将会小心调用验证并当验证未完成时返回用户正确的页面。

在 Struts 邮件阅读示例程序中的 EditRegistration 是这样描述的：

```
<action path="/editRegistration"
    type="org.apache.struts.webapp.example.EditRegistrationAction"
    attribute="registrationForm" scope="request"
    validate="false">
    <forward name="success path="/registration.jsp"/>
</action>
```

当/editRegistration action 被调用时，registrationForm 就被创建并加入到请求对象中，但是它的 validate 方法没有被调用。`validate` 属性的默认值是 `true`，所以如果你不想 action 触发表单验证，需要记住加入这个属性并设置为 `false`。

17. 如何创建工作流程向导？

基本的想法是通用 bean 的一系列前进，后退，取消和完成 action。推荐使用 LookupDispatchAction 来符合设计模式，并可以很容易地国际化。因为 bean 是共享的，每个选择都会向向导基础信息中添加数据。示例 struts-config.xml 如下：

```
<form-beans>
    <form-bean name="MyWizard" type="forms.MyWizard" />
</form-beans>
<!-- 向导的第一个页面（仅有前进可用）-->
<!-- 没有验证，因为结束 action 不可用-->
<actions>
    <action path="/mywizard1" type="actions.MyWizard"
        name="MyWizard" validate="false"
        input="/WEB-INF/jsp/mywizard1.jsp">
        <forward name="next" path="/WEB-INF/jsp/mywizard2.jsp" />
        <forward name="cancel"
            path="/WEB-INF/jsp/mywizardcancel.jsp" />
    </action>
<!-- 向导的第二个页面（后退，前进和完成）-->
<!-- 因为结束 action 是可用的，bean 应该被验证，如果后退 action 被请求的话，注意验证不是必须的，你可以延迟验证或进行条件验证-->
    <action path="/mywizard2" type="actions.MyWizard"
        name="MyWizard" validate="true"
        input="/WEB-INF/jsp/mywizard2.jsp">
        <forward name="back" path="/WEB-INF/jsp/mywizard1.jsp" />
        <forward name="next" path="/WEB-INF/jsp/mywizard3.jsp" />
        <forward name="finish"
            path="/WEB-INF/jsp/mywizarddone.jsp" />
        <forward name="cancel"
            path="/WEB-INF/jsp/mywizardcancel.jsp" />
    </action>
<!-- 向导的最后一页（仅有后退，完成和取消）-->
    <action path="/mywizard3" type="actions.MyWizard"
        name="MyWizard" validate="true"
        input="/WEB-INF/jsp/mywizard3.jsp">
        <forward name="back" path="/WEB-INF/jsp/mywizard2.jsp" />
        <forward name="finish"
            path="/WEB-INF/jsp/mywizarddone.jsp" />
        <forward name="cancel"
            path="/WEB-INF/jsp/mywizardcancel.jsp" />
    </action>
```

向导分块如下：

forms.MyWizard.java-持有所需信息的 form bean

actions.MyWizard.java-向导的 action，注意 LookupDispatchAction 的使用允许一个 action 类有很多方法。所要做的工作是“finish”方法。

mywizard[x].jsp-数据收集 jsp

mywizarddone.jsp-“成功”页面

mywizardcancel.jsp-“取消”页面

18. 如何链接 Action?

链接 action 可以在 struts-config.xml 文件的转发配置项中使用适当的映射来完成。假设有下面两个类：

```
/* com/AAction.java */
...
public class AAction extends Action{
    public ActionForward execute(ActionMapping mapping,
        ActionForm form, HttpServletRequest request,
        HttpServletResponse response) throws Exception{
            // Do something
            return mapping.findForward("success");
    }
}
/* com/BAction.java */
...
public class BAction extends Action{
    public ActionForward execute(ActionMapping mapping,
        ActionForm form, HttpServletRequest request,
        HttpServletResponse response) throws Exception{
            // Do something else
            return mapping.findForward("success");
    }
}
```

之后，你可以使用如下所示的 Struts 配置片段来链接这两个 Action：

```
...
<action-mappings type="org.apache.struts.action.ActionMapping">
    <action path="/A" type="com.AAction" validate="false">
        <forward name="success" path="/B.do" />
    </action>
    <action path="/B" type="com.BAction" scope="session"
              validate="false">
         <forward name="success" path="/result.jsp" />
       </action>
</action-mappings>
...
```

这里我们假设你使用了基于后缀（.do）的 Servlet 映射，因为模块支持需要它，所以这是推荐的形式。当你发送浏览器请求到 Web 应用程序，action 的名为 A.do（比如 http://localhost:8080/app/A.do），那么它会执行 AAction.execute()方法，之后会转发到“success”映射。

引发 BAction.execute()的执行的原因是配置文件中“success”的配置项<forward>使用了.do 后缀

当然编程链接 action 是可能的，但是使用 XML 配置文件能够“重排”Web 应用程序的结构的效能和易用性就更容易维护了。

作为一项规则，链接 action 是不推荐的。如果业务类是正确因素，你应该可以调用你需要从 Action 调用的任意方法。而不用剪切它们到 Rube Goldberg 控制设备。

如果必须要链接 action，要小心下面的事情：从第一个 action 中调用第二个 action 和从头开始调用第二个 action 的效果是相同的。如果这两个 action 都是更改了 form bean 的属性，那么第一个 action 做出的修改将会丢失，因为 Struts 框架会在第二个 action 被调用时调用 form bean 中的 reset()方法。

19. 如果你想继续，这里有一些很流行但是无证问题的列表

- 我该如何捕捉二进制或格式化的值，比如日期或电话号码？
- 我可以创建动态 ActionForward 吗？
- 我该怎么使用我自己的（ActionForm，ActionForward，ActionMapping，ActionServlet）类？

How to Help FAQ

1. 深入

每个志愿者项目都从涉及到的人员中获取力量。我们邀请你加入我们，可多可少任你算则。在这个项目中成员假设的角色和责任是基于表现的。每个人的输入都会影响！

这里有一位开发人员的建议来如何深入。它特别谈到了 Tomcat，但是想法可以用于到任意的 Apache 项目。

- Contributing 贡献 -- Craig R. McClanahan

这里有发送给 Jakarta Turbine 邮件列表的另外一个意见，它是关于开源产品和专利产品之间如何提升用户群体的进程和对比的。

- 理解开源 – Cameron Riley

写给 ASF 的开发人员，革命的原则提供了集体如何工作，还有我们和分层团队工作不同的深入理解。

革命的原则 – James Duncan Davidson

使用产品是一个很重要的作用。我们需要用户来报告问题，贡献补丁，建议特性等。你的反馈会帮助技术的发展。

2. 加入邮件列表

有多种参与方式。不管你如何选择来参与，我们建议你加入我们的邮件列表。

在你加入 ASF 的邮件列表之前，请先确保阅读邮件列表指南。如果你已经阅读并明白这些指南上的内容后，我们欢迎你加入 Struts 邮件列表。

在提交新的问题之前，请先咨询**邮件列表归档**和 Eric Raymond 所写的非常出色的如何聪明地提问。

要确保在提交之前，邮件客户端要关闭 HTML 功能。

3. 我的公司怎么来帮助支持 Apache Struts?

Apache Struts 是一个志愿者产品。我们的用户都是志愿者，它们捐赠时间和力量来支持这个产品。如果你想支持 Apache Struts 并成为我们用户中的一员，那么你需要深入了解并成为一名志愿者。

使用 Apache Struts 产品的开发团队有一个挑战是捐赠一名队员的每周一个下午的时间（如果你可以备用资源还可以更多）。你的团队成员浏览过 JIRA 的问题而不要补丁或单元测试，并添加补丁或测试。（请注意我们不在 JavaDoc 文档中使用@author 标签）如果你的补丁中包含有@author 标签，那么我们要求请去除它。

如果 Apache Struts 产品不能做你想要的事情了，那么**你**就可以提出和加强补丁了。如果 Apache Struts 产品没有按你想的来发布，那么你就可以提出测试和修复发布。（比如 Craig McClanahan 对 Tomcat 所做的那样）

如果 Struts 可以做你想做的事情，可以将你的 war 文件保存在 FAQ 和 how-to 中帮助其它人，之后我们可以做成文档的一部分。邮件列表非常活跃，可移动的归档也很多。我们通常可以让志愿者来减少最佳的文章做成相关文档并和他人分享。

我们不会用 Struts 来赚钱，但是想成为我们用户的人可以捐赠时间或精力作为代表金钱。

4. 我该如何创建补丁？

补丁是机器可以读取的脚本，并可以自动创建对文本文件的改变，包括源代码和文档。补丁格式则是需要人来读取的。开发人员之间交流补丁并在应用到主资源库之前来讨论其中的改变。

影响源代码或文档的最佳方式是提供补丁。之后，Apache Struts 的开发人员可以审查补丁并决定是否在主资源库中来应用它。

要创建补丁，首先你要从主资源库中检出一份源代码或文档的拷贝。之后你就可以对这份拷贝做出修改，使用简单的 Subversion 命令来创建补丁，比如：

```
svn diff Main.java patchfile.txt
```

之后，在 JIRA 的问题中创建其改变的内容，然后附加补丁文件。

一些 Apache 的项目要求你提交补丁到邮件列表。我们则建议你创建一个 JIRA 问题，之后附加补丁到这个问题中。要完成这个操作，你必须先创建问题，之后修改报告并添加补丁。我们知道这种方法有一点笨拙，但是它能让我们不会丢失东西，帮助保证你的补丁会被考虑到。

NetBeans 社区在创建补丁的主题中也有一个帮助部分。

5. 我该如何报告缺陷或建议功能?

为 Apache Struts 产品跟踪缺陷报告和增强建议是通过 Apache Struts JIRA 功能来处理的。请从列表中选择合适的 Apache Struts 产品，之后选择你认为该报告相关的组件。你会自动被所报告的缺陷或增强状态的电子邮件通知。请在报告提交前确保阅读过如何有效报告 Bug。

如果你不能编写补丁来解决问题，那么说明这个问题的单元测试也是可以的。（当然，单元测试证明你的补丁能正确表示问题就行。）

如果这个缺陷或功能已经在跟踪了，你可以为问题投票，这会获得它的更多关注。每个用户一次可以投六票。

如果这个问题附加了补丁，你可以应用到你本地的 Struts 中，并且报告这个补丁是否起作用。从开发人员处获得关于这个补丁的反馈是很有帮助的。如果你已经尝试了补丁并发现它有用，不要犹豫，在这个问题上添加一个“对我有用”的标记。

建议的功能也在 JIRA 问题跟踪器上来维护。

6. 我该如何为 Struts 贡献源代码?

一个很好的起点是**查看开放问题的列表**并在问题跟踪器中添加功能建议。如果你看到一个问题需要补丁，而这个补丁你可以编写，那么你可以随时附加补丁。如果你看到一个问题需要单元测试来证明它已经被修复了，那么可以随时附加测试用例。如果一些人已经发布了问题的补丁，你可以看看解决的结果，将补丁添加到你本地 Struts 的代码中去。之后让大家知道是否对你偶用，如果真的有用，那么就会这个问题和补丁投票。

如果没有关于你的症状的未解决问题，那么另外一个很好的起点就是为已经存在的功能（尽管它们现在工作良好）**贡献单元测试**。

你可以为代码或文档上传一个建议补丁，在问题跟踪器中创建功能建议。**在创建之后**，你可以上传一个包含补丁的文件。

我们目前进行单元测试的方法对于练习方法级别的东西非常管用，但是不能真正解决动态变化的情形--很多特定 Struts 自定义标签的执行。你可以尝试假定 JSP 容器会做什么，但是很多可靠的测试会真正执行真实容器中的标签。对于那个目的，我们可以使用 Cactus 测试框架，重复执行基于 JUnit 的测试要保证在转换环境后没有什么不好的事情发生。现在，有一些很小的动态测试；理想情况下，我们将对每个标记都进行测试，覆盖每个标签属性值的合理组合（是的，这是一个很艰巨的任务--如果我们做到这一点，测试代码的最终行数无疑将超过框架本身的代码行数）。

7. 我该如何贡献文档?

Struts 2 的文档使用 Atlassian Confluence wiki 软件来维护，并且会自动输出成 HTML 来在网站上观看。要帮助 Struts 2 的文档，你必须在 cwki.apache.org/confluence 创建一个账户，并且必须向 ASF 提交一个贡献者许可协议。

帮助文档的其它方式是在需要修改的页面留下意见。如果你有 cwiki Confluence 账户，你可以在 Struts 2 Wiki 上创建页面而不需要提交 CLA（贡献者许可协议）。

Struts 1 的文档在资源库下的 `xdocs` 文件夹下作为子项目来维护。要为子项目创建文档，改变自文档 trunk 目录并运行 `maven site`。Maven 会构建编译站点并存储输出在

target/docs/文件夹下。可以参考 Struts 维护网站来获取详细信息。

从程序角度来说，在 Struts 1 文档和软件之间的补丁的唯一不同是文档保存在 XML 格式下而不是 Java 源代码。否则，所有的就属于同一指令了。

如果你想来帮助我们的问题，那么在 XML 源码下提供补丁和新的页面是很重要的。否则，其它志愿者会帮助你来做这些事情，但可能永远都无法完成（这样不好但确实如此）。

如果你提交了新的材料，决定在哪里放置这些和其余的文档有关的东西是很重要的。而且，其他人会在添加之前弄清楚，其他人也可能是你自己。

如果你做出了新的贡献，比如新的 HOW-TO，要提交我们使用的 XML 格式的文档。这会很容易在官方文档中加入你的贡献，也可以作为 Struts 发布包的一部分。

入门的窍门是下载每天的构建并创建子项目站点。之后尝试在 xdocs/下添加你自己的 XML 页面来看看构建是否成功。如果没成功，会报告出有问题的元素在哪里，就像报告错误的程序表达式在哪儿是一样的。如果成功了，那么你的页面会在 target/site/下可用。

你也可以在 Apache Struts wiki 中来发布文档。

8. 下一个发布包是什么时候？

这里有一个关于发布的事实：

Apache 产品的发布是建立在优点的基础上，而不是关于严格的时间表。志愿者捐献他们可以从事产品开发的时间。但是所有的志愿者都有自己的工作和生活，那才是要优先考虑的。因为 Struts 没有对个人工作付费，我们无法制定对于日期上的承诺。

Apache 采用的发布底线是非常严格的。我们不会参考日程表来折中我们软件的质量（之后附带一些准备好的东西）。当所有都准备好时就准备发布。

这听起来可能有些轻率，但这是真的。产品质量的交付，先进的软件不是随便一个人就能预言的。如果谁这么做了，那就在骗你。所以，我们不会这么做。

我们要做的是我们开发的软件的所有发布是在开发完成之后。这种方式你可以判断一下开发的进程有多快，而且开发出的东西是否迎合了你的需求。如果你现在需要一个功能，那么你就可以每天来构建，或者使用自己的补丁。没有内部的代码库，私有开发列表，秘密聊天室或电话会议。你所看到的就是我们所做的。如果你看到了开发列表，那么你就可以知道开发人员知道的所有事情了。实际就是这样的。

那么，你要告诉开发团队什么事情呢？如果你可以基于每天构建来发布应用程序，那么就可以考虑那种选择。尽管我们没有发布更新，你仍然可以发布你自己的应用程序，而且你也可以访问所有最新的补丁或增强。（就像我们在大厅工作。）如果你仅可以基于 Struts 的发布包来发布你的应用程序，那么你应该将应用程序基于 Struts 的发布包，并且要留意下一次的发布。这种方式至少可以预先警示和准备。

9. 我该如何帮助下一个发布？

- 很重要的是，下载最新的构建包或开发包并用来测试你自己的应用程序。在问题跟踪器上报告任何问题或怀疑的问题。我们更快地解决这些问题，那么 beta 或候选发布版就会在完成之前发布。（我们该怎么知道什么时候完成？--当我们解决所有问题的时候=:O）我们更快地发现问题，那么就更快地完成。）
- **贡献单元测试**。离发布之日越近，我们就越担心会出现一些问题。我们做的测试越多，那么我们对应用补丁就越有自信。证明了那些问题确实是缺陷的测试我们是很

欢迎的。但是我们也渴望对所有功能都有测试，即便那些功能运行良好=:O)。

- 在问题跟踪器上**查看问题列表**。如果你可以回复什么问题，那么就回复。如果有发布的补丁，可以在你的应用程序中来测试，并报告结果，如果补丁有效请对它进行投票。
- **确认问题分类和状态**。新手常常将功能建议或帮助问题作为“bug”。这会使得我们在下一个 beta 发布之前（很显然）需要应用的修复列表变得很庞大，这会使得找到某些问题非常困难。如果问题看起来分类不正确，请做出最近的判断并改变它。如果一个页签看起来和另外一个重复，那么就要更改。每个对页签的修改都会自动发到开发列表中，开发人员都会看到。
- 在问题跟踪器上对你感觉应该先处理的问题**进行投票**。如果在你要投票的问题上没有包含补丁，那么你可以自己尝试来编写补丁。（这才是好的做法，补丁才是唯一可以投票的东西）很多开发人员都向 Struts 贡献了代码和文档。你也可以=:O)

在用户邮件列表里回答问题。代码编写人员和志愿者相比，时间上有很大的限制。如果开发人员在邮件列表中来互相支持，那么代码编写人员就会有更多时间用在下一次的发布上。

10. 我该如何来帮助做出决定？

阿帕奇软件基金会的一个指导原则是“他们做工作，并做出决定”。这句话其实是一个双关语。一个项目会通过投票（非常少）来做出决定，但是真正的决定是当志愿者真正开始工作时做出的。除非有人自愿来做工作，其它决定都是没有意思的。

在 ASF 的项目中，比如 Apache Struts，志愿者持续给项目贡献并被邀请成为“代码提交者”。在适当的时候，代码提交人员被邀请加入项目管理委员会（Project Management Committee，PMC）。ASF 的一个目标是对于所有代码提交人员都必须在项目管理委员会中。

“持续”的意思是个人在项目中至少活跃六个月以上。贡献应该是补丁（代码或问题）和发布邮件列表。补丁必须能够被资源库接受。发布的邮件应该是有用的，友好的，协作的。预期提交人员中的最重要的品质是友好的风范并促进信誉。

PMC 成员注意到 Struts 开发人员满足了我们的条件，我们其中之一会对内部的 PMC 邮件列表进行投票。（这通常当一些人在厌倦作为志愿者编写补丁的时候！）内部的邮件列表很少使用，也不会用来作为开发讨论。如果 PMC 投票通过，我们会向开发人员私下发送邀请，给他一个接受的机会或直接拒绝。如果候选人可以接受，那么 PMC 会宣布新的开发列表中的成员。

要了解更多关于做出决定的信息，请参考“How the ASF Works”和 Apache Struts Charter。要了解项目的基础信息，请参考 Struts 1 wiki 中的“项目维护和资源”。

Struts 是如何工作的？

[从 Kickstart FAQ 而来引申]

“Jave Servlet 被设计用来处理由 Web 浏览器发出的请求。服务器页面被设计来创建动态的 Web 页面，它们可以将广告牌站点转变为可交互的应用程序。Struts 使用了一个特殊的 Servlet 作为总机来路由来自 Web 浏览的请求到合适的服务器页面。这使得企业级 Web 应用程序非常容易创建和维护。”

这是一个非常好的高层次的描述，但是我们要详细描述 Struts 框架的机制和依赖。

- 开发 Web 应用程序有一个部署描述符（`WEB-INF/web.xml`），这是必须要编写的。

这个文件描述了 Web 应用程序的配置，包括欢迎页面（当没有特定请求时展示的目录下的文件），映射的 Servlet（路径或扩展名），还有那些 Servlet 需要的参数。

在 `web.xml` 文件中，配置框架的 ActionServlet 作为处理所有给定映射（通常使用扩展名`.do`）请求的 Servlet。ActionServlet 就是开篇提及的“总机”。

而且在 `web.xml` 中，配置 `ActionServlet` 来为 Struts 自身使用一个或多个配置文件。

而现在，我们可以说在服务器上使用`/myapp` 作为路径安装 Web 应用程序，并使用最简单的配置。

如果你需要更多关于部署描述符的信息，请从 Sun 公司的 Java 网站阅读有关 Servlet 规范的信息。

- 在框架配置文件中，使用应用程序的控制器组件来关联路径，就是我们熟知的 `Action` 类（比如”login”==>LoginAction 类）。这就告诉 `ActionServlet` 对于收到的请求 `http://myhost/myapp/login.do`，应该调用控制器组件 `LoginAction`。

  注意 URL 中的扩展名`.do`。这个扩展名会让容器（比如 Tomcat）调用 `ActionServlet`，会将“login”这个部分作为你想做的事情。参考配置信息，之后执行 `LoginAction`。

- 对于每个 `Action`，也在框架中使用结果页面的名字配置了，这就可以作为 action 的结果来展示。作为 action 的结果，可以有多于一个的视图（通常情况下，至少有两个：一个是“success”，另外一个是“failure”）。

  `Action`（编写的控制器组件）是基于这些*逻辑*结果映射名的。它会使用如“success”，“failure”，“ready”，“ok”，“UserError”等这样的信息回报给 `ActionServlet`。Struts 框架（通过编写的配置信息）知道如何转发到合适的*特定*页面。这也增加了视图层重新配置的优点，只要简单编辑 XML 配置文件即可。

  在这一点上，Struts 框架知道如何派发到控制器组件中，并展示控制处理程序的结果显示些什么。应用程序的“模型”部分是完全由你决定的，并从控制器组件中来调用。

- 也可能在 Struts 框架配置文件的 action（或一组 action）来关联 JavaBean。JavaBean 作为表单或展示数据的资源库，可以在视图层和控制层之间交互。

  这些 Bean 是自动在控制器组件（比如 `LoginAction` 类）和关联这个控制器的任意视图页面中可见的。

  这些 Bean 也可以利用框架的帮助来进行验证，并确保用户在表单中编写了对系统良好的数据。它们可以通过 session 来运输，允许表单去跨多个视图页面和控制器中的 Action。

  注意：你必须为视图层（*到达*客户端的）使用某种形式的服务器端技术（JSP，Velocity，XSLT）来看到这些数据（普通 HTML 将不会起作用）。在服务器端工作的框架，客户端视图也必须在那里组合。

  客户端提供的数据通过普通的表单提交（POST/GET）方法，框架在调用控制器组件之前 Bean 中更新数据。

- 在 Web 应用程序中，用户将会看到页面代表的视图层。它们可以是 JSP 页面，Velocity 模板，XSLT 页面和其它可以在框架中使用的技术。

  尽管普通 HTML 文件可以在应用程序中使用，但它们不能利用动态特性的所有优点。

  在 Struts JSP 标签库的示例时候，一些其它包可以用来使框架更容易和你喜欢的展示技术使用。对于 Velocity 模板，有 Velocity 为 Struts 准备的视图工具。如果

你想在应用程序中使用 XSLT，你可以在 stxx 和 StrutsCX 之间选择。

这些包使得标准的 Struts 框架元素看起来就像原始展示技术的一部分。Struts 也很容易混合和匹配。如果需要，你可以使用在相同的应用程序中使用 JSP，Velocity 模板和 XSLT。

因为 Struts 依赖于标准的 Servlet 技术，你应该可以在 Struts 中使用任意的 Java 展示技术。

- 尽管 Struts 框架的关注点是控制器，展示层也是应用程序很重要的一部分。Struts 的 JSP 标签库包含了一些通用的和 Struts 特定的标签来帮助你在视图中使用动态数据。

  自定义的 JSP 标签是 Struts 代码中的一个很好的想法。这是教育性的告诉我们要注意 1.1b3 版的 Struts 核心 Java 代码约有 28000 行，而对于标签库（包含 tiles）的代码就达到了 41000 行。

  这些标签帮助你将视图层和控制层粘贴在一起而不需要在 JSP 中嵌入大量的 Java 代码。这会使页面看起来更像是 XML 风格的，对于 Web 设计人员来说也很容易处理普通文本格式的 JSP。它也帮助我们在控制器和视图之间最小化依赖。

  自定义标签也可用来创建表单，逻辑跳转到其它页面，调用 web 应用中的其它 action。

  标签也可以帮助你来进行国际化，错误消息处理等。

  所有的这些能力依赖于提供给 Struts 配置文件的方式。

记得这里描述的机制仅当 `ActionServlet` 处理请求时有效是很重要的。

因为仅当请求提交时可以让容器（比如 Tomcat，WebSphere 等）调用 `ActionServlet`，你必须保证依赖于 Struts 的任意页面可以通过请求映射到 `ActionServlet`（比如扩展名为 `.do`）来完成。

HOWTO

如何访问数据库

很多开发人员认为访问数据库是应用程序“业务端”的部分。很多时候，我们不会为访问数据库而去访问数据库。我们使用数据库作为大型业务事务的一部分。那么我们来看看从 Struts 框架访问业务逻辑。

在 Web/持久层和业务类（包含那些访问数据库的类）之间使用 Action 作为一个瘦的适配器是很好的。

所以第一个设计业务 API 使用普通的 Java 类。最佳的做法就是使用采用了普通 Java 类，并返回 Javabean 或 Javabean 集合的对象。之后 Action 调用这些对象并传递回结果给 Web/持久层。

通用的方法是为应用程序的每个业务事务或用例创建 Action 类。简单的“CRUD”应用程序可能会有 CreateAction，RetrieveAction，UpdateAction 和 DeleteAction。为了完成每个事务，Action 可以调用任何需要的业务 API 类。

理想情况下，所有数据库访问代码应该封装在业务 API 类的后面，那样框架不知道你使用的什么样的持久层（即使是有持久层）。它只是传递一个键或查询字符串来获取 bean 或 bean 的集合。这可以让你在其它环境中使用相同的业务 API 类，而且在 Web 环境外基于业务 API 使用单元测试。

Struts 的邮件阅读器示例程序中展示了这是怎么完成的。邮件阅读器使用了 DAO（Data Access Object，数据访问对象）模式来从（Struts）控制层中分隔出持久层。邮件阅读器定义了 DAO 接口，那么 Action 就可以调用了，之后定义使用存储在主内存中的数据库的实现类。其它实现也可以定义和使用，而不需要修改任何的 Struts 类。

开始入门，最简单的方式是在 Action 和应用程序用例之间设置 1 对 1 的关系。每个用例可以使用一个或多个业务 API 的调用，但是从用户角度来说，每个用例都是独立的事务。

当你积累出经验，你会发现组合 Action 类的方法，就是使用 DispatchAction。使用单独的“框架”Action 来调用所有的业务类也是可以的，正如在沙箱子项目中，Scaffold ProcessAction 完成的那样。

使用很少的 Action 需要对 Struts 和 MVC 框架运作模式的深入理解。不要犹豫刚开始在创建更多的 Action 类时所犯的错误。配置文件可以使得之后重构 Action 非常容易，因为你可以修改 Action 的类型而不需要修改应用程序中的任何东西。

使用数据源

当你使用 DAO 方式，所有的数据库访问细节就隐藏在业务接口之后了。业务类的实现处理了所有的细节。比如使用来 `DataSource` 连接池式连接数据库。

作为一项规则，你通常应该使用连接池来访问数据库。`DataSource` 接口是当今实现连接池的最佳方式。很多容器和数据库系统都捆绑了数据源的实现，你可以直接使用。很多时候，数据源也可以通过 JNDI 方式获得。JNDI 方式非常简单，对于业务类访问数据源而不用担心谁创建了它。

持久层框架

有很多有用的并且成熟的持久层框架可用。在使用原生 JDBC 或“你自己的”解决方案之前，你应该很小心地审查这些包。这里有一个在 Struts 用户列表中经常提到的包的列表：

- Hibernate
- iBatis（现在是 MyBatis，译者注）
- Object Relational Bridge
- Torque/Peers

要获取详细信息，可以参考 SourceForge 上的 Struts 社区资源。

也可以参考

http://www.mail-archive.com/struts-user@jakarta.apache.org/msg24621.html
http://www.mail-archive.com/struts-user@jakarta.apache.org/msg24709.html
http://www.mail-archive.com/struts-user@jakarta.apache.org/msg24626.html
http://www.mail-archive.com/struts-user@jakarta.apache.org/msg24331.html
http://www.mail-archive.com/struts-user@jakarta.apache.org/msg24102.html
http://www.mail-archive.com/struts-user@jakarta.apache.org/msg23501.html
http://www.mail-archive.com/struts-user@jakarta.apache.org/msg23455.html
http://www.mail-archive.com/struts-user@jakarta.apache.org/msg23375.html

http://www.mail-archive.com/struts-user@jakarta.apache.org/msg23321.html
http://www.mail-archive.com/struts-user@jakarta.apache.org/msg23098.html
http://www.mail-archive.com/struts-user@jakarta.apache.org/msg22713.html
http://www.mail-archive.com/struts-user@jakarta.apache.org/msg21974.html
http://www.mail-archive.com/struts-user@jakarta.apache.org/msg21026.html
http://www.mail-archive.com/struts-user@jakarta.apache.org/msg19338.html
http://www.mail-archive.com/struts-user@jakarta.apache.org/msg18323.html
http://www.mail-archive.com/struts-user@jakarta.apache.org/msg14975.html
http://www.mail-archive.com/struts-user@jakarta.apache.org/msg14914.html
http://www.mail-archive.com/struts-user@jakarta.apache.org/msg14435.html
http://www.mail-archive.com/struts-user@jakarta.apache.org/msg01562.html
转化/数据传输
http://www.mail-archive.com/struts-user@jakarta.apache.org/msg24480.html
http://www.mail-archive.com/struts-user@jakarta.apache.org/msg23623.html
http://www.mail-archive.com/struts-user@jakarta.apache.org/msg10195.html
http://www.mail-archive.com/struts-user@jakarta.apache.org/msg10205.html

呈现动态结果集

很多查询的结果会映射到你使用的 ActionForm 上，所以你可以呈现结果集作为 ActionForm 的集合。但是有时结果集中的列不是 ActionForm 的属性，甚至是事先知道的。

很高兴的是，Struts 的 JSP 标签不关心你使用的是什么类型的 bean。你可以直接输出结果集。但是结果集保留了到数据库的连接，并直接传递“所有的”到 JSP 中，使其变得很凌乱。那么开发人员该做什么呢？

从 1.1 版开始，简单的选择是使用 ResultSetDynaClass 来转换结果集到 DynaBean 的 ArrayList 中。Struts 自定义标签可以使用 DynaBean 的属性，这和使用传统 JavaBean 的属性是一样方便的。（参考 4.3.1 节来获取详细信息。）

因为这些类都在 BeanUtils 组件的 JAR 中，你现在已经在用了，只是需要实现转换规则（参考 ResultSetDynaClass）

如何创建 Action Form

这里有一个简单的登录表单来展示 Struts Action 框架是如何比直接使用 HTML 和标准 JSP 更容易处理表单的。参考下面名为 logon.jsp 的页面（基于 Struts 的邮件阅读器示例程序）：

```
<%@ page language="java" %>
<%@ taglib uri="/WEB-INF/struts-html.tld" prefix="html" %>
<%@ taglib uri="/WEB-INF/struts-bean.tld" prefix="bean" %>
<html:html>
<head>
<title>
   <bean:message key="logon.title"/>
</title>
</head>
```

```
<body bgcolor="white">
<html:errors/>
<html:form action="/logon" focus="username">
<table border="0" width="100%">
   <tr>
      <th class="right"><bean:message key="prompt.username"/></th>
      <td class="left"><html:text  property="username" size="16"/></td>
   </tr>
   <tr>
      <th class="right"><bean:message key="prompt.password"/></th>
      <td class="left"><html:password property="password" size="16"/></td>
   </tr>
   <tr>
      <td class="right">
         <html:submit>
            <bean:message key="button.submit"/>
         </html:submit>
      </td>
      <td class="right">
         <html:reset>
            <bean:message key="button.reset"/>
         </html:reset>
      </td>
   </tr>
</table>
</html:form>
</body>
</html:html>
```

下面的项目说明基于这个示例的 form 处理的关键特性：

- taglib 指令告诉 JSP 页面编译器去哪里为 Struts 的 JSP 标签找标签库的描述符。在这里，我们使用 bean 作为前缀来从 strtus-bean 库中标识标签，而“html”作为前缀来标识从 struts-html 标签库中获取的标签。任何想用的前缀都是可以使用的。
- 这个页面使用了一些 message 标签来查看从 MessageResources 对象包含的应用程序的所有资源中获得的国际化消息。这个页面要运行起来，下面的消息的键必须在资源中定义：
  - logon.title-logon 页面的标题
  - prompt.username-提示字符串“Username”
  - prompt.password-提示字符串“Password”
  - button.submit-按钮标签“Submit”
  - button.reset 按钮标签“Reset”

  当用户登录时，应用程序可以在用户 session 中存储一个 Locale 对象。这个 Locale 可以以在合适的语言来选择消息。实现给定用户选择来更换语言这是很容易的--仅更改存储的 Locale 对象即可。那么所有的消息都会自动更换。

- **errors** 标记展示了由业务逻辑组件存储的任意消息，如果没有存储错误，那么就不显示东西。这个标签会在下面来介绍。
- **form** 标签呈现了 HTML 的<form>元素，这是基于规定的属性的。它也使用 ActionForm bean[org.apache.struts.action.ActionForm]关联这个表单中的所有字段。这个标签在 Struts 配置中会寻找/logon action 映射。logon 映射告诉标签 form bean 存储在 session 上下文中，键是 logonForm。开发人员提供 ActionForm bean 的 Java 实现，是 Struts 类 ActionForm（请参考第四章相关内容）的子类。这个 bean 用来为所有的和 bean 的属性名相匹配的输入字段提供初始值。如果合适的 bean 没有发现，那么就会自动创建一个新的，通过 action 映射中的配置，使用指定的 Java 类名。
- form bean 也可以在标签中提供 name 和 type 属性来指定。但通常的做法是，所有的都在 Struts 配置文件（请参考第四章相关内容）中来指定。
- **text** 标签呈现了 HTML 的<input>元素，其 type 是“text”。这种情况下，占据浏览器屏幕的字符位置的数量也可以指定了。当页面执行时，就是当前对应 bean 的 username 属性的值（也就是说，那个值是 getUsername 方法返回的值）
- **password** 标签的用法也是类似的。不同的是浏览器会使用型号字符来代替输入的真实值，因为类型是密码。
- **submit** 和 **reset** 标签在页面的下部生成对应的按钮。每个按钮的文本标签使用消息标签来创建，由于这样来提示，这些值都可以国际化。

传输数据

在 ActionForm 和业务对象之间传递数据，很多开发人员使用 Commons 组件 BeanUtil 的方法。从 ActionForm 中传递数据到业务对象中，你可以使用如下的语句：

```
PropertyUtils.copyProperties(actionForm, businessObject);
```

要从业务对象往 ActionForm 中传递数据，那么可以调转参数

```
PropertyUtils.copyProperties(businessObject, actionForm);
```

关于使用这个技术想更多信息，可以参考 Commons 的 BeanUtils 文档，还有 Struts 的邮件阅读器示例应用程序。

除了 BeanUtils，也有其它的工具可以很容易地来使用业务对象和 ActionForm。要了解更多关于 POJO ActionForm 的信息，可以参考 FormDef 扩展和 Struts Live 工具包。

Multipart Form

处理 multipart 表单也很容易。很显然，当你创建至少有一个“file”类型的 input 的表单时，就创建 multipart 表单。创建 multipart 表单的第一步是使用 struts-html 标签库来创建显示页面：

```
<%@page language="java">
<%@taglib uri="/WEB-INF/struts-html.tld" prefix="html">
<html:form action="uploadAction.do" enctype="multipart/form-data">
    Please Input Text: <html:text property="myText">
    Please Input The File You Wish to Upload: <html:file property="myFile">
   <html:submit />
</html:form>
```

下一步是创建 ActionForm bean：

```
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import org.apache.struts.action.ActionForm;
import org.apache.struts.action.ActionMapping;
import org.apache.struts.upload.FormFile;
public class UploadForm extends ActionForm {
    protected String myText;
    protected FormFile myFile;
    public void setMyText(String text) {
        myText = text;
    }
    public String getMyText() {
        return myText;
    }
    public void setMyFile(FormFile file) {
        myFile = file;
    }
    public FormFile getMyFile() {
        return myFile;
    }
}
```

参考 FormFile 的 Javadoc 文档来看看其中处理上传文件的方法。同时也看看 ActionServlet 和 ActionMapping 的 Javadoc 文档来看看你可以指定的各种参数去改变文件是如何上传的。基本上来说，在 action 类的 `execute` 方法中，你可以调用 `((UploadForm) form).getMyFile()` 来检索 FormFile 并做你想要做的工作。

如何构建应用程序

本文档概述了一个可能的开发步骤序列，可以用它来创建 Struts 应用程序。它并不是每一个参考开发活动的一个完整描述。很多细节上的文档在引用之处都用链接来导向了。

对于那些时间/耐心很少的人来说，你可以花几分钟从 Maven 的 archetype 特性中来获得一个 Struts 应用程序的构建和运行。

注意-本文档的编写时间（2006-10-16），这个示例的版本还没有发布。一旦发布以后，这个说明可以移除，archetype 也可以运行正常。

```
$ cd ~/projects
$ mvn archetype:create                                \
      -DarchetypeGroupId=org.apache.struts            \
      -DarchetypeArtifactId=struts-archetype-blank    \
      -DarchetypeVersion=1.3.5                        \
      -DgroupId=com.example                           \
      -DpackageName=com.example.projectname           \
      -DartifactId=my-webapp
$ cd my-webapp
$ mvn install
$ mvn jetty:run
```

注意事项

1. 需求开发和设计已经超出了本文档的范围。
2. 要获得安装框架的帮助，可以参考 6.2 章节内容。
3. 除了下面讨论的，还有很多其它方法来使用 Struts 框架开发，也有很多其它可用的特性。
4. 本文档关注于表单/数据的应用程序，也适用于其它类型的应用程序。
5. 这份材料最初是为 Struts 1.1（beta 2）版本所写，1.3.5 发布包中也可以使用。

概述

1. 用 JSP 文件实现数据项表单。
2. 实现一个或多个 `ActionForm` 子类来缓冲 JSP 和 Action 之间的数据。
3. 创建 XML 文档来定义应用程序的验证规则。
4. 实现一个或多个 `Action` 子类来响应提交请求。
5. 创建 `struts-config.xml` 文件来关联 form 和 action。
6. 创建或更新 `web.xml` 来引用 `ActionServlet`。
7. 并行任务
    1. 构建
    2. 单元测试
    3. 部署

细节

1. 用 JSP 文件实现数据项表单。
    1. 使用来自 `html` 标签库的元素来定义表单元素。
    2. 使用来自 `bean` 标签库的 `message` 和其它元素来定义表单的标签和其它静态文本。
        1. 创建和维护展示的文本元素的属性文件。（更多信息可以参考 0.6 章节内容）
    3. 使用 `property` 属性来链接表单字段和 `ActionForm` 实例的变量。
2. 实现一个或多个 `ActionForm` 子类来缓冲 JSP 和 Action 之间的数据。
    1. 创建和相关 JSP 表单中的属性名对应的 get/set 方法对。比如：

```
<html:text property="city" />
```
需要
`getCity()`和 `setCity(String c)`

2. 当需要时，创建 `reset` 方法并将 `ActionForm` 中的字段设置为它们的默认值。很多 ActionForm 也不需要这样做。

3. 创建 XML 文档来定义应用程序的验证规则。
4. 实现一个或多个 `Action` 子类来响应提交请求。
    1. 如果你想一个类来处理多于一个事件，那么就让子类继承 DispatchAction 或 LookupDispatchAction。（例如：一个 Action 来处理“插入”，“更新”和“删除”事件，使用不同的“替代”execute 方法来实现）
    2. 在应用程序的 Action 类中使用 `execute` 方法（或它的替代）来和负责数据库交互的对象进行接口连接。比如 EJB 等。
    3. 使用 `execute` 方法（或它的替代）的返回值来直接到合适的下一个页面的用户界面上。
5. 创建 `struts-config.xml` 文件来关联 form 和 action。
6. 创建或更新 `web.xml` 来引用 `ActionServlet`。（可以阅读 5.4 节获取详细内容）
7. 并行任务
    1. 构建
        1. 使用 Ant。它可以编译，创建 WAR 文件，执行 XSLT 转换，执行单元测试，和版本控制系统交互，清理等。（更多信息请参考 Ant 网站）
        2. 增量创建和使用构建脚本，比如创建需要复制，编译的文件等。
    2. 单元测试
        1. 使用 JUnit 对普通 JavaBean 进行单元测试。（更多信息请参考 JUnit 网站）
        2. 使用 Cactus 对 JSP，标签库和常规的 Servlet 进行单元测试。（更多信息请参考 Cactus 网站）
        3. 使用 StrutsTestCase 来对 Action Servelet 进行单元测试。（更多信息请参考 StrutsTestCase 网站）
        4. 将所有单元测试添加到构建脚本中，并作为一个独立的目标。这个目标应该使用 `junit` 标签来执行每个子测试用例。（更多信息可以参考）
    3. 部署
        1. 构建脚本应该创建包含所有开发文件和构建框架的文件的 WAR 文件。

Struts 校验器指南

Struts 校验器从 Struts 0.5 发布日期开始就以某种形式存在了。最初它是打包作为开发人员发布包的内容。后来，核心代码移入了 Jakarta 的 Commons 中，而 Struts 特定的扩展成为 Struts 1.1 的一部分。

为了广大使用 Struts 校验器的开发人员的便利，本文档首先概要说明了从 Struts 1.1 版开始加入的核心功能，之后介绍新加入的功能。

一旦你配置了校验器插件，那么它可以加载校验器资源，你只需扩展 `org.apache.struts.validator.action.ValidatorForm` 即可，而不用扩展 `org.apache.struts.action.ActionForm` 了。之后，当 `validate` 方法被调用时，从 Struts 配置中获取的 action 的 name 属性就用来为当前的 form 加载校验信息。在校验器配置中的 form 元素的 `name` 属性应该和 action 元素的 `name` 属性相匹配。

另外一种方法是使用 action 映射属性。这种情况下，你需要扩展 ValidatorActionForm 而不是 ValidatorForm 了。ValidatorActionForm 从 Struts 的配置中使用了 action 元素的 path 属性，它应该在校验器配置中匹配 form 元素的 name 属性。

那么分离的action映射可以对多页面form的每个页面来定义，而校验form可以和action相关联而不是页面数字（比如在多页面 form 的校验器示例中展示的那样）。

国际化

每个校验器 form 和校验器配置文件中的 FormSet 元素所分组的。FormSet 有语言，国家和与 java.util.Locale 类相对应的变种属性。如果这些属性没有被指定，那么 FormSet 就被设置为默认的语言环境。FormSet 也可有和它相关的常量。和 FormSet 同级的可以有全局元素，它也可以有常量和执行验证的校验器 action。

注意：你必须在国际化 FormSet 之前声明一个默认的没有国际化的 FormSet。这就允许校验器在没有发现语言环境设置时回到默认的版本。

对于插件式的校验器的默认错误消息可以使用 msg 元素来覆盖。为了对表面校验替换使用 msg 属性来生成错误消息，如果字段的 name 属性的名字匹配上了校验器的 name 属性，从字段中来的 msg 属性会被使用。

错误消息的参数可以使用 arg 元素和位置属性来设置。如果 arg 元素的 name 属性没有设置，它会成为构建不同错误消息的默认 arg 值。如果 name 属性设置了，你可以对指定的插件校验器来指定参数，之后这会用于构建错误消息。

```
<field property="lastName" depends="required,mask">
    <msg name="mask" key="registrationForm.lastname.maskmsg"/>
    <arg position="0" key="registrationForm.lastname.displayname"/>
    <var>
        <var-name>mask</var-name>
        <var-value>^[a-zA-Z]*$</var-value>
    </var>
</field>
```

默认情况下，arg 元素会在消息资源中来查找 key 属性。如果 resource 属性设置为 false，它会直接传递值而不会从消息资源中检索值。

注意从 Struts 1.1 版开始，你必须在任一模块中明确地定义消息资源来使用校验器，由于访问顶层资源的问题。这会仅仅影响使用模块的应用程序。

```
<field property="integer" depends="required,integer,intRange">
    <arg position="0" key="typeForm.integer.displayname"/>
    <arg position="1" name="intRange" key="${var:min}" resource="false"/>
    <arg position="2" name="intRange" key="${var:max}" resource="false"/>
    <var>
        <var-name>min</var-name><var-value>10</var-value>
    </var>
    <var>
        <var-name>max</var-name><var-value>20</var-value>
    </var>
</field>
```

校验的标准构建

校验器附带了一组预定义的校验机制，如下：

- **required** -强制性字段验证。没有变量

```
<field property="name" depends="required">
      <arg position="0" key="customer.name"/>
</field>
```

- **requiredif**-字段依赖性校验器

废弃了，使用 validadwhen

- **validwhen**-基于另外一个字段来检查字段的校验器

参考后续章节-“使用 validwhen 设计复杂校验”

- **minlength**-验证输入数据不小于指定的最小值长度。需要 minlength 变量。

```
<field property="name" depends="required,minlength">
   <arg position="0" key="customer.name"/>
   <arg position="1" name="minlength" key="${var:minlength}"
      resource="false"/>
   <var>
      <var-name>minlength</var-name><var-value>3</var-value>
   </var>
</field>
```

- **maxlength**-验证输入数据不小于指定的最大值长度。需要 maxlength 变量。

```
<field property="name" depends="required,maxlength">
   <arg position="0" key="customer.name"/>
   <arg position="1" name="maxlength" key="${var:maxlength}"
      resource="false"/>
   <var>
      <var-name>maxlength</var-name><var-value>30</var-value>
   </var>
</field>
```

- **mask**-根据正则表达式进行格式校验。需要 mask 变量来指定正则表达式。从 1.1 版开始，正则表达式必须以^开头并以$结束（参考下面的示例）

```
<field property="name" depends="required,mask">
   <msg name="mask" key="registrationForm.lastname.maskmsg"/>
   <arg position="0" key="registration.name"/>
   <var>
      <var-name>mask</var-name><var-value>^[a-zA-Z]*$</var-value>
   </var>
</field>
```

- **byte**-验证字段可以转换成 Byte 类型

```
<field property="age" depends="byte">
   <arg position="0" key="employee.age"/>
</field>
```

- **short**-验证字段可以转换成 Short 类型

```
<field property="productnumber" depends="short">
   <arg position="0" key="order.prodno"/>
</field>
```

- **integer**-验证字段可以转换成 Integer 类型

```
<field property="ordernumber" depends="integer">
   <arg position="0" key="order.number"/>
</field>
```

- **long**-验证字段可以转换成 Long 类型

```
<field property="ordernumber" depends="long">
   <arg position="0" key="order.number"/>
</field>
```

- **float**-验证字段可以转换成 Float 类型

```
<field property="amount" depends="float">
   <arg position="0" key="sale.amount"/>
</field>
```

- **double**-验证字段可以转换成 Double 类型

```
<field property="amount" depends="double">
   <arg position="0" key="sale.amount"/>
</field>
```

- **date**-验证字段可以转换成 Date 类型。校验器使用 java.text.SimpleDateFormat 来解析日期，可选使用 datePattern 或 datePatternStrict 变量。如果没有指定模式，那么就假设使用默认的短日期格式。使用 datePatternStrict 和 datePattern 变量的不同是 datePatternStrict 额外地检查输入数据是否和指定模式（比如 1/1/2004 格式就不能匹配到 MM/dd/yyyy 格式）的长度一致。

```
<field property="saledate" depends="required,date">
   <arg position="0" key="myForm.saledate"/>
   <var>
      <var-name>datePattern</var-name>
      <var-value>MM/dd/yyyy</var-value>
   </var>
</field>
```

还有，

```
<field property="saledate" depends="required,date">
    <arg position="0" key="sale.orderdate"/>
    <var>
        <var-name>datePatternStrict</var-name>
        <var-value>MM/dd/yyyy</var-value>
    </var>
</field>
```

- **range**-验证数字范围

已经废弃了，使用 intRange，longRange，floatRange 或 doubleRange。

- **intRange**-验证 integer 字段的指定范围。需要 min 和 max 变量来指定范围。 这个校验器依赖 integer 校验器，它也必须在字段的 depends 属性中。

```
<field property="age" depends="required,integer,intRange">
    <arg position="0" key="employee.age"/>
    <arg position="1" name="intRange" key="${var:min}"
        resource="false"/>
    <arg position="2" name="intRange" key="${var:max}"
        resource="false"/>
    <var>
        <var-name>min</var-name><var-value>18</var-value>
    </var>
    <var>
        <var-name>max</var-name><var-value>65</var-value>
    </var>
</field>
```

- **longRange**-验证 long 字段的指定范围。需要 min 和 max 变量来指定范围。 这个校验器依赖 long 校验器，它也必须在字段的 depends 属性中。

```
<field property="age" depends="required,long,longRange">
    <arg position="0" key="employee.age"/>
    <arg position="1" name="longRange" key="${var:min}"
        resource="false"/>
    <arg position="2" name="longRange" key="${var:max}"
        resource="false"/>
    <var>
        <var-name>min</var-name><var-value>18</var-value>
    </var>
    <var>
        <var-name>max</var-name><var-value>65</var-value>
    </var>
</field>
```

- **floatRange**-验证 float 字段的指定范围。需要 min 和 max 变量来指定范围。 这个校验器依赖 float 校验器，它也必须在字段的 depends 属性中。

```
<field property="ordervalue" depends="required,float,floatRange">
    <arg position="0" key="order.value"/>
    <arg position="1" name="floatRange" key="${var:min}"
        resource="false"/>
    <arg position="2" name="floatRange" key="${var:max}"
        resource="false"/>
    <var>
        <var-name>min</var-name><var-value>100</var-value>
    </var>
    <var>
        <var-name>max</var-name><var-value>4.99</var-value>
    </var>
</field>
```

- **doubleRange**-验证 double 字段的指定范围。需要 min 和 max 变量来指定范围。 这个校验器依赖 double 校验器，它也必须在字段的 depends 属性中。

```
<field property="ordervalue" depends="required,double,doubleRange">
    <arg position="0" key="employee.age"/>
    <arg position="1" name="doubleRange" key="${var:min}"
        resource="false"/>
    <arg position="2" name="doubleRange" key="${var:max}"
        resource="false"/>
    <var>
        <var-name>min</var-name><var-value>100</var-value>
    </var>
    <var>
        <var-name>max</var-name><var-value>4.99</var-value>
    </var>
</field>
```

- **creditCard**-校验信用卡数字格式

```
<field property="name" depends="required, creditCard">
    <arg position="0" key="customer.cardnumber"/>
</field>
```

- **email**-校验电子邮件地址格式

```
<field property="customeremail" depends="email">
    <arg position="0" key="customer.email"/>
</field>
```

- **url**-校验 url 格式。有四个可选变量（allowallschemes，allow2slashes，nofragments 和 schemes），可以用来配置这些校验器。
  - **allowallschemes** 指定了是否所有的模式都允许。合法值有 true 或 false（默认是 false）。如果设置为 true，那么 schemes 变量就会忽略了。

- **allow2slashes** 指定了是否允许两个'/'字符。合法值有 `true` 或 `false`（默认是 `false`）。
- **nofragments** 指定了是否允许 fragements。合法值有 `true` 或 `false`（默认是 `false`，也就是说 fragements 是允许的）。
- **schemes**-用来指定逗号分隔的合法模式列表。如果不指定，那么默认使用的就是 `http`，`https` 和 `ftp`。

```
<field property="custUrl" depends="url">
    <arg position="0" key="customer.url"/>
</field>
<field property="custUrl" depends="url">
    <arg position="0" key="customer.url"/>
    <var>
        <var-name>nofragments</var-name>
        <var-value>true</var-value>
    </var>
    <var>
        <var-name>schemes</var-name>
        <var-value>http,https,telnet,file</var-value>
    </var>
</field>
```

常量/变量

全局常量可以在全局标记中，而且 FormSet/Locale 常量可以在 formset 标签中来创建。常量当前仅可以在字段的 property 属性，字段的 var 元素 value 属性，字段的 msg 有元素 key 属性，字段的 arg 元素的 key 属性中来替换。字段的变量在 arg 元素中来替代（比如${var:min}）。替换的顺序是 FormSet/Locale 常量首先替换，全局变量第二替换，对 arg 元素的变量最后替换。

```
<global>
    <constant>
        <constant-name>zip</constant-name>
        <constant-value>^\d{5}(-\d{4})?$</constant-value>
    </constant>
</global>
<field property="zip" depends="required,mask">
    <arg position="0" key="registrationForm.zippostal.displayname"/>
    <var>
        <var-name>mask</var-name>
        <var-value>${zip}</var-value>
    </var>
</field>
```

在字段下的 var 元素可以用来存储被插件校验器使用的变量。这些诶变量可以通过字段的 getVar(String key)方法来访问。

```
<field property="integer" depends="required,integer,intRange">
    <arg position="0" key="typeForm.integer.displayname"/>
    <arg position="1" name="intRange" key="${var:min}" resource="false"/>
    <arg position="2" name="intRange" key="${var:max}" resource="false"/>
    <var>
        <var-name>min</var-name>
        <var-value>10</var-value>
    </var>
    <var>
        <var-name>max</var-name>
        <var-value>20</var-value>
    </var>
</field>
```

使用 validwhen 来设计复杂校验

[从 Struts 1.2.0 版开始]在校验设计中的一个常用需求是验证基于另外一个字段（比如，你让用户输入两次密码，为了保证输入值的一致的）的一个字段。此外，也有表单中的一些字段可能仅仅需要其它字段有特定的值。validwhen 校验器就是设计来处理这些情况的。

validwhen 校验器使用单独的 var 字段，称作 test。这个 var 的值是一个布尔值变量，它必须是 true 来使验证成功。这个值允许的表达式有：

- *this*表示，包含了当前字段测试的值。
- 其它表单中使用字段名引用的字段，比如 customerAge。
- null 表示，将基于 null 或空字符串的匹配。
- 单引号或双引号的字符串值。
- 十进制，十六进制或八进制格式的整数值。

引用索引字段也允许下列表达式：

- 表单中的索引字段由明确的整数来引用，比如 childLastName[2]。
- 表单中的索引字段由隐藏的整数来引用，比如 childLastName[]，会在数组中使用相同的索引作为测试字段的索引。
- 表单中的索引字段的属性由明确或隐藏的整数来引用，比如 child[].lastName，会在数组中使用相同的索引作为测试字段的索引。

这种方式如何工作的示例，考虑表单中有 sendNewsletter 和 emailAddress 字段。如果 sendNewsletter 字段是非空的，emailAddress 字段就是需要的。那么你可以使用 validwhen 这样来编码：

```
<field property="emailAddress" depends="validwhen">
    <arg position="0" key="userinfo.emailAddress.label"/>
    <var>
        <var-name>test</var-name>
        <var-value>((sendNewsletter == null) or (*this* != null))</var-value>
    </var>
</field>
```

这可以读作：如果 sendNewsletter 或这个字段值不是 null 时，这个字段是合法的。

这里有一个使用了索引字段的略微复杂的示例。假设表单的一些行允许用户输入他们想订购的零件编号和数量。Bean 类 orderLine 的数组用称为 coderLines 的属性来持有每个条目。如果你想来验证每行的零件数目都有数量的输入，那么你可以这么来做：

```
<field property="quantity" indexedListProperty="orderLines"
    depends="validwhen">
    <arg position="0" key="orderform.quantity.label"/>
    <var>
        <var-name>test</var-name>
        <var-value>((orderLines[].partNumber == null) or (*this* !=
null))</var-value>
    </var>
</field>
```

这可以读作：如果对应的 partNumber 字段是 null，或这个字段不是 null，那么这个字段就是合法的。

最后一个示例，设想表单中用户必须输入英寸格式的身高，如果他们的身高低于 60 英寸，那么就不能验证通过 nbaPointGuard 作为事业。

```
<field property="nbaPointGuard" depends="validwhen">
    <arg position="0" key="careers.nbaPointGuard.label"/>
    <var>
        <var-name>test</var-name>
        <var-value>((heightInInches >= 60) or (*this* == null))</var-value>
    </var>
</field>
```

这种语法的一些注释：

- 所有的比较必须以括号来结束
- 两项之间仅可以使用 and 或 or 来连接。
- 如果比较的两项可以转换成整数，那么就是数字的比较，否则是字符串的比较。

插件式的校验器

按照惯例，应用程序使用的校验器可以通过名为“validator-rules.xml”的文件来加载，而且校验表单（或“校验规则”）可以分别（也就是说，在“validations.xml”文件中）来配置。这种方法分开了校验器，你可以在另外一个应用程序中重用来自每个应用中指定的验证规则。

这个校验器捆绑了一些准备好可以使用的校验器。捆绑的校验器包括：required，mask，byte，short，int，long，float，double，date（没有本地化支持）和数字范围。

在默认创建情况下，“mask”校验器依赖于“required”。那就是说“required”必须在“mask”运行之前完全成功。“required”和“mask”校验器是部分构建于框架内部的。如果字段是 null 或长度为零，那么任何不是“required”的字段会忽略其它校验器。无论怎么样，“required”和“mask”的实现仍然可以通过配置文件实现插件式管理，就像其它的校

验器一样。

如果使用了 Javascript 标签，那么客户端的 Javascript 生成器会寻找校验器中 javascript 属性的值并且生成一个提供验证表单方法的对象。要获得更多关于 Javascript 校验器标签如何使用的详情，可以参考 html 标签库 API 参考文档。

“mask”校验器允许你使用正则表达式来校验字段。它使用了来自 Apache Jakarta 的正则表达式包。

使用的主类是 `org.apache.regexp.RE`。

下面是默认 validator-rules.xml 中的校验器配置示例。

```
<validator name="required"
        classname="org.apache.struts.validator.FieldChecks"
        method="validateRequired" methodParams="java.lang.Object,
              org.apache.commons.validator.ValidatorAction,
              org.apache.commons.validator.Field,
              org.apache.struts.action.ActionMessages,
              org.apache.commons.validator.Validator,
              javax.servlet.http.HttpServletRequest"
        msg="errors.required"/>
<validator name="mask"
        classname="org.apache.struts.validator.FieldChecks"
        method="validateMask" methodParams="java.lang.Object,
              org.apache.commons.validator.ValidatorAction,
              org.apache.commons.validator.Field,
              org.apache.struts.action.ActionMessages,
              org.apache.commons.validator.Validator,
              javax.servlet.http.HttpServletRequest"
        depends="" msg="errors.invalid"/>
```

创建插件式校验器

`methodParams` 属性使用了逗号分隔的类名列表。`method` 属性需要遵守上述列表的前面名，这个列表可以有下列任意组合来组成：

- `java.lang.Object`-Bean 校验器会执行。
- `org.apache.commons.validator.ValidatorAction`-当前的 ValidatorAction 会执行。
- `org.apache.commons.validator.Field`-Field 对象会被校验。
- `org.apache.struts.action.ActionMessages`-如果校验失败，错误对象会添加一个 ActionMessage。
- `javax.servlet.http.HttpServletRequest`-当前请求对象。
- `javax.servlet.ServletContext`-应用程序的 ServletContext。
- `org.apache.commons.validator.Validator`-当前的 org.apache.commons.validator.Validator 实例。
- `java.util.Locale`-当前用户的本地化环境

多页面表单

field 元素有一个可选的 page 属性。它可以设置为整数。页面上所有的任意字段小于或等于当前页面会的验证在服务器端执行。页面上所有的任意字段等于当前页面的验证会在客

户端的 Javascript 中生成。multi-part 表单期望 page 属性被设置好。

```
<html:hidden property="page" value="1"/>
```

比较两个字段

这里有一个示例，它说明了如何比较两个字段来判断它们的值是否相同。这种情况的好的示例是当你验证用户修改他们密码时会有一个主密码字段和确认密码字段。

```
<validator name="twofields" classname="com.mysite.StrutsValidator"
      method="validateTwoFields" msg="errors.twofields"/>
<field property="password" depends="required,twofields">
   <arg position="0" key="typeForm.password.displayname"/>
   <var>
      <var-name>secondProperty</var-name>
      <var-value>password2</var-value>
   </var>
</field>
```

```
public static boolean validateTwoFields(Object bean,ValidatorAction va,
   Field field, ActionErrors errors, HttpServletRequest request,
   ServletContext application) {
   String value = ValidatorUtils.getValueAsString(bean,
      field.getProperty());
   String sProperty2 = field.getVarValue("secondProperty");
   String value2 = ValidatorUtils.getValueAsString(bean, sProperty2);
   if (!GenericValidator.isBlankOrNull(value)) {
     try {
       if (!value.equals(value2)) {
          errors.add(field.getKey(),Resources.getActionError(
                application,request,va,field));
          return false;
       }
     } catch (Exception e) {
          errors.add(field.getKey(),Resources.getActionError(application,
                request,va,field));
          return false;
     }
   }
   return true;
}
```

已知问题

自从 Struts 校验器依赖于 Commons 的 Validator 组件，问题报告和增强请求可以基于任

意产品来列出了。

- Commons Validator 问题跟踪器（JIRA）

条件必填字段

你可以定义如“如果字段 X 是非空的，并且字段 Y 等于“male”，那么才验证这个字段”的逻辑。这么来做的推荐方式是使用 validwhen 校验器，上面已经介绍过了，这从 Struts 1.2.0 开始就可以使用了。requiredif 校验器，是从 Struts 1.1 开始加入的，在开始使用 validwhen 之后就被废弃了，而且 requiredif 会在今后的发布包中被移除。而如果你使用 requiredif，这里有一个简单的教程。

我们假设你有一个医疗信息，并有三字段，sex，pregnancyTest 和 testResult。如果 sex 是“f”或“F”，那么 pregnancyTest 是必填的。如果 pregnancyTest 非空，那么 testResult 就是必填的。那么在校验器配置中的配置项就会是这样：

```
<form name="medicalStatusForm">
<fieldproperty="pregnancyTest" depends="requiredif">
    <arg position="0" key="medicalStatusForm.pregnancyTest.label"/>
    <var>
        <var-name>field[0]</var-name><var-value>sex</var-value>
    </var>
    <var>
        <var-name>fieldTest[0]</var-name><var-value>EQUAL</var-value>
    </var>
    <var>
        <var-name>fieldValue[0]</var-name><var-value>F</var-value>
    </var>
    <var>
        <var-name>field[1]</var-name><var-value>sex</var-value>
    </var>
    <var>
        <var-name>fieldTest[1]</var-name><var-value>EQUAL</var-value>
    </var>
    <var>
        <var-name>fieldValue[1]</var-name><var-value>f</var-value>
    </var>
    <var>
        <var-name>fieldJoin</var-name><var-value>OR</var-value>
    </var>
</field>
<field property="testResult" depends="requiredif">
    <arg position="0" key="medicalStatusForm.testResult.label"/>
    <var>
        <var-name>field[0]</var-name><var-value>pregnancyTest</var-value>
    </var>
```

```
    <var>
        <var-name>fieldTest[0]</var-name><var-value>NOTNULL</var-value>
    </var>
</field>
</form>
```

这里有一个更为复杂的示例，使用了索引属性。

如果在 Struts 配置中有这样的内容

```
<form-bean name="dependentlistForm"
    type="org.apache.struts.webapp.validator.forms.ValidatorForm">
    <form-property name="dependents"
        type="org.apache.struts.webapp.validator.Dependent[]" size="10"/>
    <form-property name="insureDependents" type="java.lang.Boolean"
        initial="false"/>
</form-bean>
```

而 dependent 是一个包含属性 lastName，firstName，dob 和 coverageType 的 bean。

你可以定义如下的校验：

```
<form name="dependentlistForm">
<field property="firstName" indexedListProperty="dependents"
    depends="requiredif">
    <arg position="0" key="dependentlistForm.firstName.label"/>
    <var>
        <var-name>field[0]</var-name><var-value>lastName</var-value>
    </var>
    <var>
        <var-name>fieldIndexed[0]</var-name><var-value>true</var-value>
    </var>
    <var>
        <var-name>fieldTest[0]</var-name><var-value>NOTNULL</var-value>
    </var>
</field>
<field property="dob" indexedListProperty="dependents"
    depends="requiredif,date">
    <arg position="0" key="dependentlistForm.dob.label"/>
    <var>
        <var-name>field[0]</var-name><var-value>lastName</var-value>
    </var>
    <var>
        <var-name>fieldIndexed[0]</var-name><var-value>true</var-value>
    </var>
    <var>
        <var-name>fieldTest[0]</var-name><var-value>NOTNULL</var-value>
    </var>
</field>
```

```
<field property="coverageType" indexedListProperty="dependents"
    depends="requiredif">
    <arg position="0" key="dependentlistForm.coverageType.label"/>
    <var>
        <var-name>field[0]</var-name><var-value>lastName</var-value>
    </var>
    <var>
        <var-name>fieldIndexed[0]</var-name><var-value>true</var-value>
    </var>
    <var>
        <var-name>fieldTest[0]</var-name><var-value>NOTNULL</var-value>
    </var>
    <var>
        <var-name>field[1]</var-name>
        <var-value>insureDependents</var-value>
    </var>
    <var>
        <var-name>fieldTest[1]</var-name><var-value>EQUAL</var-value>
    </var>
    <var>
        <var-name>fieldValue[1]</var-name><var-value>true</var-value>
    </var>
    <var>
        <var-name>fieldJoin</var-name><var-value>AND</var-value>
    </var>
</field>
</form>
```

这可以读作：如果 lastName 字段是非空的，那么 firstName 字段也是必填的。因为 fieldIndexed 是 true，它就表示 lastName 必须和 firstName 有相同索引字段的属性。dob 也是如此，如果非空，期望对日期进行验证。

如果对于相同索引的 bean 非空，而且非索引字段 insureDependents 是 true，那么 lastName 也是必填的。

你也可以使用[n]语法，字段索引可以是任意数字，唯一的限制是它们全部必须是 AND 或 OR，不能混用。

不支持的 JavaScript 校验

[从 Struts 1.2.0 开始]你可以强制客户端的 Javascript 校验来检查所有限制，而不是在第一个错误发生后停止。可以在校验器插件中设置一个新的属性，`stopOnFirstError`，并设置为 false。

这里有一个可以在 Struts 配置文件中使用的配置段：

```
<plug-in className="org.apache.struts.validator.ValidatorPlugIn">
   <set-property property="pathnames"
      value="/WEB-INF/validator-rules.xml,/WEB-INF/validations.xml"/>
   <set-property property="stopOnFirstError" value="false"/>
</plug-in>
```

校验器 API 指南

简洁的 Struts 校验器 API 指南可以帮助你入门。（请参考 org.apache.struts.validator 包的 Javadoc 文档）

校验器参考资源

- 使用校验器检查表单
- **Struts 校验器：校验两个字段匹配**
- **DynaForm 和校验器**
- 校验用户输入

IDE HOWTO

- 安装 Eclipse IDE
- 安装 Netbeans IDE（4.0 版）

External FAQ and HOWTO

- Apache Struts Wiki
- jGuru FAQ
- Struts Tips
- Struts University
- Struts Validator:Validating Two Fields Match
- Unit Test with Struts TestCase

附录

A. 开发路线图

本文概述了一些我们期望未来 Struts 1 发布的更新。

本文的目的仅在于讨论。所有代码库的发布更更新受限于 Apache Struts PMC 的投票。

JIRA 查询

Apache Struts 项目使用 JIRA 问题跟踪器来管理问题报告和增强建议。为了您的方便，这里有一些畅通的 JIRA 查询：

- 开放的问题（Struts 1）
- 开放的问题（Struts 2）
- 开放的问题（沙箱）

Struts 1.x

Struts 1.x 系列的发布聚焦于已有功能的重构并持续关注向后兼容性。

新功能可以加入到 1.x 系列中，但只有向后兼容先前的版本的可以保留下来。Struts 框架的 API 已经涉及到严格的废弃/替换/移除协议。鼓励开发人员停留在当前保持“最佳可用”的发布状态来观察废弃的警告。特性在移除前是废弃的，而废弃的特性会在后续的发布中被移除。

在整个 1.x 系列中，对于单元测试的覆盖面还要继续加强和扩大。可能的情况下，Bug 报告应该包含失败的测试用例。新功能的建议应该包含工作测试套件。（鼓励通过测试驱动开发模式创建的新特性。）

增强的请求正如它们被建议的那样，**被记录在 JIRA 中。JIRA 中的增强建议列表并不暗示这是“计划的”**，而只是一些人的建议而已，这个想法（至今）也并没有排除。

未来发布的里程碑提供了对于活跃的建议或对当前发布没有准备的可能开发的增强建议。如果开发人员的报告没有给定指定的里程碑标签，那么**它可能永远也不会实现**。当开发人员（包含非提交人员）确实开发了一种增强特性，它们应该重新标记一个特定的发布里程碑，比如“1.4.1”或“1.4.2”。

如果一个增强特性没有标记为特定的目的，你自己可以随时启动它。很多我们最好的特性是由开发人员贡献的，你也一样。如果你开发了一种增强特性，附加一个你开发增强特性的记录之后尽快发布一个补丁。如果增强特性的开发没有成功，发布一个说明增强特性中有什么问题，看看其它开发人员是否可以来探究解决。

开发里程碑

这里是我们已有的里程碑，还有可能的方向。

1.0.x 的发布（完成）

框架内部的重要重构来提供对模块和新的“config”系列对象的支持。将 Struts 的 Tiles 和 Validator 绑定到主要发布包中。初始的 Struts EL 发布作为一个可选包。

1.1.x 的发布（完成）

分割了内部类库到它们在 Jakarta Commons 项目的独立项目中。添加了对模块和插件的支持。

1.2.x 的发布（完成）

添加了通配符映射，内部项目结构调整，移除废弃内容，其它小的增强特性的支持。

1.3.x 的发布（待定）

进化增强产品基础，基于现有的特性或代码库。

- 将发布分离成独立的 JAR 组件。
- 支持 Maven 2 构建。
- 将 action 移到“Struts Chain”请求处理器中。
- 增强所有配置信息来扩展配置环境，就如已经完成的 Tiles 定义那样。

B. 各种容器上的安装

B1. Bluestone Universal Business Server 7.2

- 需要 UBS 7.2 版来运行 WAR 文件应用程序。UBS 7.2 发布版在这里。如果你使用 7.2.1 版，你需要下载 WAR 文件补丁，在 Bluestone 的产品增强部分网站，参考这里。
- 在安装正确版本和/或 UBS 7.2 补丁之后，你需要修改 apserver.txt 文件来指出 WAR 文件应用程序的正确目录。找到和如下描述相似的部分来修改：

```
[SaServletEngine.class]
session_affinity=1
type=1
program=/SaServletEngine.class
file_path=f:\webapps
host=localhost:20000
```

- 使用由“file_path”变量指定的目录，或修改它指向你自己的 webapp 目录。复制“struts-documention.war”和“struts-example.war”文件到 webapp 目录中，之后开启运行 UBS（如果需要的话请阅读 UBS 的发布文档来获取如何启动的相关信息）。Webapp 应用现在可以使用如下的 URL 来访问了：
  http://localhost/<PLUGIN>/SaServletEngine.class/struts-example/
  http://localhost/<PLUGIN>/SaServletEngine.class/struts-documentation/
- 注意：“<PLUGIN>”代表了为特定 web 服务器使用的插件。对于 Windows 上的 Apache 来说，就可能是“cgi-bin/SaCGI.exe”，而对于 Windows 上的 IIS 来说，就可能是“script/SaCGI.exe”或“script/ISAPI.dll”。参考 UBS 的文档来获取更多信息。

B2. iPlanet Web Server 4.2

从 Tomcat（支持 Web 应用和 WAR）移植基于 Struts 的应用程序到 iWS 4.2（不支持 Web 应用和 WAR）会有一些问题。

在 Iws 5.0 路线图中提及的，Web 应用和 WAR 会在 iWS 5.0 中支持。

类路径问题

这是相当简单的。因为没有类路径指明要在`$SERVER_ROOT/config/jvm12.conf` 中设置 `WEB-INF/lib` 和 `WEB-INF/classes` 的概念。

上下文相对路径

所有 URL 应该从文档根部开始都可见。在我的情况下，创建一个从`$DOCROOT/myapp` 到 `webapps/myapp` 的符号连接。

扩展映射

配置文件$SERVER_ROOT/config/rules.properties 和 web.xml 有相同的机制。

在 rules.properties 文件中有如下内容，可以转发以"do"结尾的所有 url 到逻辑名为 action 的 Servlet 中。

```
####
@.*[.]do$=action
####
```

B3. iPortal Application Server 1.3

基于 Windows 2000 测试

重要须知

当前，iPAS 1.3 已经完全支持 JSP 1.1/1.2 规范了。

具体来说，指定在 JSP 1.1 勘误表和 JSP 1.2 建议最终草案中“问题 7”的自定义标签参数的自动类型转换还没有实现。

既然这样，使用了 Struts 标签库标签的 JSP 页面中的参数需要转换（比如布尔值），将不能在 JRun 下编译。这包含了 Struts 示例程序。尝试运行示例应用程序会导致和下面抛出异常相似的问题：

```
/struts-example/index.jsp:
Compilation failed [IT_Builder:1000]
at com.iona.j2ee.builder.JavaBuilder.build(JavaBuilder.java:84)
at com.iona.j2ee.builder.JspBuilder.build(JspBuilder.java:51)
at com.iona.j2ee.builder.WarBuilder.build(WarBuilder.java:111)
at com.iona.j2ee.builder.EarBuilder.build(EarBuilder.java:99)
at com.iona.j2ee.builder.EarBuilder.main(EarBuilder.java:223)
at iportal.build.main(build.java:14)
ocale(boolean) in org.apache.struts.taglib.html.HtmlTag
cannot be applied to (java.lang.String)
_x0.setLocale("true");
^
1 error
```

（获取更多详细信息，可以参考 http://www.mail-archive.com/struts-user@jakarta.apache.org/msg01860.html）

下面的介绍描述了如何在 Ipas 1.3 下安装 Struts 示例程序。随后的一部分描述了 Struts 示例程序是如何修补与 Struts 一起工作的。

下面的介绍假设如下内容：

- iPortal Application Server 1.3 已经安装了。
- Struts 和 XML 解析器库已经在类路径下了。

安装 Struts 示例应用程序

- 点击菜单项[Start iPAS Services]来开启 iPAS 服务。
- 点击菜单项[iPortal Application Server]开启 iPortal Application Server。
- 开启命令行界面。更改$INSTALLDIR\IONA 来运行 setenvs.bat 文件。
- 创建名为 jars 的目录。

现在，开启 EARSCO 工具。输入 java iportal.earsco 并按提示做：

- 下一步。
- 输入 struts-example 的应用程序名称之后点击下一步。
- 在第三部点击复选框并输入 struts-example 的 WAR 名称。之后点击下一步。
- 点击完成。

现在你必须复制 struts-example 的 war 文件内容到 EARSCO 目录结构中，按如下操作：

在$INSTALLDIR\IONA\jars\struts-examples\src\struts-example.war 下，复制内容到如下目录：etc，lib，src 和 web。

- 在根目录 $INSTALLDIR\jakarta-tomcat-3.2.1\webapps\struts-example\WEB-INF 中复制所有文件到 earsco 目录 $INSTALLDIR\IONA\jars\struts-examples\src\struts-example.war\etc 下。不要复制 classes 或 lib 目录。
- 复制目录 $INSTALLDIR\jakarta-tomcat-3.2.1\webapps\struts-example\WEB-INF\lib 到 earsco 目录 $INSTALLDIR\IONA\jars\struts-examples\src\struts-example.war\lib 下
- 复制目录 $INSTALLDIR\jakarta-tomcat-3.2.1\webapps\struts-example\WEB-INF\classes 到 earsco 目录 $INSTALLDIR\IONA\jars\struts-examples\src\struts-example.war\src 下。
- 复制目录 $INSTALLDIR\jakarta-tomcat-3.2.1\webapps\struts-example 到 earsco 目录 $INSTALLDIR\IONA\jars\struts-examples\src\struts-example.war\web 下
- 下一步，修改 $INSTALLDIR\IONA\jars\struts-examples\etc 目录中的 application.xml 文件：

```
<application>
<!-- Add display name -->
<display-name>Struts Example</display-name>
.......
```

- 最后更新$INSTALLDIR\IONA\jars\struts-examples directory 目录下的 cc.xml 文件，如下操作：

```
<configuration>
    <web-app>
        <context-root>struts-example</context-root>
    </web-app>
</configuration>
```

现在，就可以准备编译和部署 struts-example 了。

编译$INSTALLDIR\IONA\jars\struts-examples 目录下的源码，输入

```
java iportal.build
```

之后，输入

```
java iportal.deploy
```

部署的第一个名字会被部署配置程序提示并要求提供 struts-example.ear 文件和 cc.xml 文件的两个位置。一旦两个元素都已经指定了，继续执行，直到完成按钮点击。EAR 文件应该就成功部署了。

通过在浏览器中使用如下的 URL 来测试示例应用：

```
http://hostname:9000/struts-example/index.jsp
```

struts-documentation.war 可以通过相同的过程来安装。

修补 struts 示例应用程序

在这部分的开头已经提示了，Struts 应用示例程序在没有修改时是不能在 iPAS 1.3 下运行的。下列修改是需要的：

- index.jsp，logon.jsp：修改<html:html locale="true">成<html:html locale=<%=true%>>。
- registration.jsp，subscription.jsp：修改所有 filter="true"的实例到 filter=<%=true%>。

B4. Jetty Java HTTP Servlet Server

Jetty 是一个小型的纯 Java 实现的开源 HTTP 服务器，它支持 Servlet 2.3 和 JSP 1.2 的规范。Jetty 可以从 http://www.mortbay.com/jetty 来下载。

当放置到 Jetty 的 webapps 目录下时，Struts 的 WAR 文件直接在箱外运行。一个额外需要的步骤是为每个加到 WAR 文件在 Jetty 服务器配置文件中添加一个配置项，来映射合适的请求路径到 Struts 的 Web 应用程序中（使用“<Call name="addWebApplication">...”）。

比如，如果你复制了 Struts 二进制发布包的 WAR 文件到称为"%JETTY_HOME%/webapps/struts"的"%JETTY_HOME%/webapps"的子目录下，那么你将不得不做：

```
- %JETTY_HOME%/webapps/struts/struts-documentation.war
- %JETTY_HOME%/webapps/struts/struts-example.war
- %JETTY_HOME%/webapps/struts/struts-exercise-taglib.war
- %JETTY_HOME%/webapps/struts/struts-upload.war
- %JETTY_HOME%/webapps/struts/struts-blank.war
```

想使用 demo.xml 配置文件来运行 Jetty，那么就需要添加如下的代码段到 demo.xml 中，在监听器声明之后的任意位置上添加。

```
<!-- 为 Struts 的 Jetty 配置开始-->
<Call name="addWebApplication">
<Arg>/struts/struts-documentation/*</Arg>
<Arg><SystemProperty name="jetty.home"
      default="."/>/webapps/struts/struts-documentation.war</Arg>
<Arg><SystemProperty name="jetty.home"
       default="."/>/etc/webdefault.xml</Arg>
```

```
<Arg type="boolean">false</Arg>
<!-- 如果是 true，展开 war 到临时目录下-->
</Call>
<Call name="addWebApplication">
<Arg>/struts/struts-example/*</Arg>
<Arg><SystemProperty name="jetty.home"
      default="."/>/webapps/struts/struts-example.war</Arg>
<Arg><SystemProperty name="jetty.home"
      default="."/>/etc/webdefault.xml</Arg>
<Arg type="boolean">true</Arg>
<!-- 如果是 true，展开 war 到临时目录下-->
</Call>
<Call name="addWebApplication">
<Arg>/struts/struts-exercise-taglib/*</Arg>
<Arg><SystemProperty name="jetty.home"
      default="."/>/webapps/struts/struts-exercise-taglib.war</Arg>
<Arg><SystemProperty name="jetty.home"
      default="."/>/etc/webdefault.xml</Arg>
<Arg type="boolean">false</Arg>
<!-- 如果是 true，展开 war 到临时目录下-->
</Call>
<Call name="addWebApplication">
<Arg>/struts/struts-upload/*</Arg>
<Arg><SystemProperty name="jetty.home"
       default="."/>/webapps/struts/struts-upload.war</Arg>
 <Arg><SystemProperty name="jetty.home"
       default="."/>/etc/webdefault.xml</Arg>
 <Arg type="boolean">true</Arg>
<!-- 如果是 true，展开 war 到临时目录下-->
</Call>
<Call name="addWebApplication">
<Arg>/struts/struts-blank/*</Arg>
<Arg><SystemProperty name="jetty.home"
      default="."/>/webapps/struts/struts-blank.war</Arg>
 <Arg><SystemProperty name="jetty.home"
       default="."/>/etc/webdefault.xml</Arg>
 <Arg type="boolean">true</Arg>
<!-- 如果是 true，展开 war 到临时目录下-->
</Call>
 <!-- 为 Struts 的 Jetty 配置结束-->
```

B5. JRUN 3.0 SP2A, VERSION 3.02A.11614

在微软 Windows 2000，IIS 5.0 下测试

重要须知

当前，JRun 没有完全兼容 JSP 1.1/1.2 规范。

特别要指出，JSP 1.1 勘误表和 JSP 1.2 建议最终草案里的“问题 7”的自定义标签参数的自动类型转换还没有实现。

由于这个原因，JSP 页面使用 Struts 标签库，其参数需要转换（比如布尔值）则在 JRun 下不能编译。这包含了 Struts 的示例应用程序。要运行示例应用程序会导致抛出和如下相似的异常：

```
/struts-example/index.jsp:
javax.servlet.ServletException: Compilation error
occurred:
allaire.jrun.scripting.DefaultCFE:
Errors reported by compiler:
c:/JRun/servers/default/Struts
Example/WEB-INF/jsp/jrun__index2ejspa.java:41:1:41:27:
Error: No match was found for method
"setLocale(java.lang.String)"
```

（要获得更多详细内容，可以参考：http://www.mail-archive.com/struts-user@jakarta.apache.org/msg01860.html）

下面的介绍描述了如何在 JRun 下安装 Struts 示例应用程序。后续的章节描述了 Struts 示例应用程序如何和 Struts 进行修补工作。

下面的介绍假设了如下内容：

- JRun 已经安装了并选择和 Web 服务器整合。
- $APP_SERVER_NAME 是用来运行应用程序的应用服务器的名称。（当 JRun 第一次安装的时候，它会创建一个应用服务器称为 JRun Default Server）
- $APP_SERVER_DIR 是用来存放$APP_SERVER_NAME 运行应用程序的目录。对于 JRun Default Server，目录就是$JRUN_HOME/servers/default，这里的$JRUN_HOME 就是 JRun 的安装路径。

安装 Struts 示例应用程序

- 登录到 JRun 管理控制台。
- 在左边的面板上，选择$APP_SERVER_NAME。展示当前服务器状态的页面会出现在右边。
- 在右边的面板上，点击 WAR 部署链接。包含当前部署的 Web 应用程序的列表就会出现。
- 在右边的面板上，点击部署应用程序，按如下步骤完成 Web 应用程序信息表单：
  - Servlet WAR 文件或目录：输入 struts-example.war 文件的完整路径或点击浏览去选择路径。
  - JRun 服务器名称：$APP_SERVER_NAME
  - 应用程序名称：Struts Example
  - 应用程序主机：所有主机

    - 应用程序 URL：/struts-example
    - 应用程序部署目录：默认是$APP_SERVER_NAME/Struts Example。（或者是为应用程序指定的名称）
- 表单填写完成后，点击部署按钮
- 如果部署成功，点击左边面板的$APP_SERVER_NAME 来重新启动应用服务器。展示当前服务器状态的页面会出现在右边。点击重新启动服务器按钮来重新启动应用服务器。
- 在浏览器中使用如下的 URL 来测试示例应用程序：
  http://hostname/struts-example/index.jsp
  struts-documentation.war 也可以使用相同的步骤来安装。

安装未打包的 Web 应用程序

上述的步骤是为*.war 文件来部署应用程序使用的。
对于未打包的应用程序来说，配置包含了如下的步骤：

- 从 JRun 管理控制台，选择$APP_SERVER_NAME（在左边面板）并点击 WAR 部署（在右边的面板）。
- JRun 服务器名称:$APP_SERVER_NAME
    - 应用程序名称：myApplication
    - 应用程序主机：所有主机
    - 应用程序 URL：/myApplication
    - 应用程序部署目录：默认是$APP_SERVER_NAME/myApplication
- 点击创建来提交表单
- 一旦 Web 应用程序创建完成，就可以在 $APP_SERVER_NAME/myApplication/WEB-INF 目录下来安装和配置 Struts 组件（struts.jar，struts*.tld 等）
- 安装其余的应用程序组件：必须的.class 文件，JSP 文件，.properties 文件等。
- 配置请求 URI（比如*.do）的扩展映射到 action servlet，展开左边面板的 $APP_SERVER_NAME，展开 Web 应用程序分支并点击 myApplication。右边面板会展示 myApplication 的出配置选项。点击 Servlet URL 映射，存在的映射列表就会展示出来。点击编辑按钮并创建如下的配置项：
    - 虚拟路径/扩展：*.do
    - Servlet 调用：action
- 点击更新按钮保存修改。
- 重新启动应用服务器。
- 应用程序应该就可以通过浏览器来访问了。

JRun 应用服务器每次在对 Web 应用程序做出如下修改时，都需要重新启动：

- 修改了.class 或.jar 文件
- 修改了.properties 文件
- 修改了.xml 文件

给 Struts 示例应用程序打补丁

在这些介绍的开头已经提到过了，Struts 示例应用程序在 JRun 下不修改是无法运行的。需要进行下面的修改：

- index.jsp，logon.jsp：修改<html:html locale="true">到<html:html locale=<%=true%>>
- logon.jsp：修改<html:html redisplay="true">到<html:html redisplay=<%=true%>>
- registration.jsp，subscription.jsp：修改所有的实例 filter="true"到 filter=<%=true%>

B6. Novell ExteNd Application Server 4.0

使用 ExteNd 工作台来部署 WAR 到 Novell ExteNd Application Server 上：

- 开启 Novell ExteNd 应用服务器。
- 使用 ExteNd 工作台，创建新的项目：
  1. 文件>新项目
  2. 选择 *Deploy-Only* 并点击 OK
  3. 在存档文件下选择合适的 Struts WAR 文件
  4. 标明文件类型（比如 WAR 1.2）
  5. 给项目一个名字（比如 struts-example）
  6. 给项目一个位置（比如 D:\Struts）
  7. 点击 Next
  8. 审查这些材料保证所有都对
  9. 点击 Finish
- 建立部署计划和服务器简介：
  1. 右键单击刚创建的项目图标并选择*部署计划*
  2. 当询问创建新的部署计划时选择 OK
  3. 保存部署计划
  4. 创建服务器简介：项目>部署设置
  5. 点击服务器简介页签
  6. 选择服务器简介并点击 OK
- 部署项目：
  1. 项目>部署文件

使用 SilverCmd 从命令行部署 WAR：

- 开启 SilverStream 应用服务器
- 为 struts-example.war 应用程序创建 XML 部署计划
- 调用 struts-example-depl-plan.xml 文件。你可以使用如下内容：

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE warJarOptions PUBLIC
"-//Silverstream Software, Inc.//DTD J2EE WAR Deployment Plan//EN"
"deploy_war.dtd">
<warJarOptions>
    <warJar>
        <warJarName>struts-example.war</warJarName>
        <isEnabled>true</isEnabled>
```

```
        <urls>
            <el>struts-example</el>
        </urls>
    </warJar>
</warJarOptions>
```

- 为 struts-documentation.war 应用程序创建 XML 部署计划
- 调用 struts-documentation-depl-plan.xml 文件。你可以使用如下内容：

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE warJarOptions PUBLIC
"-//Silverstream Software, Inc.//DTD J2EE WAR Deployment Plan//EN"
"deploy_war.dtd">
<warJarOptions>
    <warJar>
        <warJarName>struts-documentation.war</warJarName>
        <isEnabled>true</isEnabled>
        <urls>
            <el>struts-documentation</el>
        </urls>
    </warJar>
</warJarOptions>
```

- 运行下面的“SilverCmd DeployWAR”命令来部署应用程序。你可以修改“localhost”到任意你想部署使用的服务器。你也可以修改“Silvermaster”到任意你想部署的数据库上。

```
SilverCmd DeployWar localhost Silvermaster struts-example.war
-f struts-example-depl-plan.xml
SilverCmd DeployWar localhost Silvermaster struts-documentation.war
-f struts-documentation-depl-plan.xml
```

B7. Orion Application Server

在下面的步骤中，$ORION_HOME 引用了安装过的 Orion 的目录位置，而$STRUTS_HOME 则是解压缩 Struts 二进制发布包的目录位置。

- 修改文件$ORION_HOME/config/application.xml 并定义两个新的应用程序，可以通过添加下面的声明，直接跟着 web-module 指令后的默认 Web 应用程序：

```
<web-module id="strutsDoc"
    path="$STRUTS_HOME/webapps/struts-documentation.war"/>
<web-module id="strutsExample"
    path="$STRUTS_HOME/webapps/struts-example.war"/>
```

- 修改文件$ORION_HOME/config/default-web-site.xml（或任意 Orion 站点的配置文件）来包含下面的声明，在<default-web-app>声明之后：

```
<web-app application="default" name="strutsDoc"
    root="/struts-documentation"/>
<web-app application="default" name="strutsExample"
    root="/struts-example"/>
```

- 在启动 Orion 之后，你应该可以访问这些应用程序了（假设你没有修改默认的端口号，默认的是 80）：

```
http://localhost/struts-documentation
http://localhost/struts-example
```

- Orion 的版本至少在 1.0.3 时还有关于 ServletContext.getResource()调用的 bug，这会阻止 Struts 示例应用程序在箱外工作。当你访问示例程序时，这个 manifests 本身作为 JSP 错误，显示下面的信息：javax.servlet.jsp.JspException: Missing resources attributeorg.apache.struts.action.MESSAGE，在错误跟踪记录之后。在 ORION_HOME/log/global-application.log 的日志文件中也会有初始化的错误消息，要解决这个问题，你可以采取下面的做法：
  - 在$STRUTS_HOME/webapps 目录中，在这里你会注意到 Orion 自动展开了每个 Web 应用程序到解压后的目录结构。
  - 在$STRUTS_HOME/webapps/struts-example/WEB-INF 目录下，复制文件 struts-config.xml 到一级目录下（也就是到 $STRUTS_HOME/webapps/struts-example 下）
  - 修改$STRUTS_HOME/webapps/struts-example/WEB-INF/web.xml 文件，修改“config”初始参数的值（对 action servlet）/WEB-INF/struts-config.xml 为/action.xml。
  - 重新启动 Orion，那么就可以访问到示例应用程序了。
  - 注意，这个解决方法有一点安全相关的副作用：struts-config.xml 文件可以通过远程客户端通过下面的 URL 来检索到：http://localhost/struts-example/struts-config.xml。因此，你应该保证这个文件中没有存储敏感的信息（比如数据库密码）。

B8. SliverStream Application Server 3.7.1 及更高版本

- 启动 SliverStream 应用服务器。
- 对 struts-example.war 应用程序创建 XML 部署计划。调用 struts-example-depl-plan.xml 文件，你可以使用下面的内容

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE warJarOptions PUBLIC
"-//SilverStream Software, Inc.//DTD J2EE WAR Deployment Plan//EN"
"deploy_war.dtd">
<warJarOptions>
    <warJar>
        <warJarName>struts-example.war</warJarName>
        <isEnabled>true</isEnabled>
        <urls><el>struts-example</el></urls>
    </warJar>
</warJarOptions>
```

对 struts-documentation.war 应用程序创建部署计划。调用 struts-documentation-depl-plan.xml 文件，你可以使用下面的内容：

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE warJarOptions PUBLIC
```

```
"-//SilverStream Software, Inc.//DTD J2EE WAR Deployment Plan//EN"
"deploy_war.dtd">
<warJarOptions>
    <warJar>
        <warJarName>struts-documentation.war</warJarName>
        <isEnabled>true</isEnabled>
        <urls><el>struts-documentation</el></urls>
    </warJar>
</warJarOptions>
```

运行下面的“SilverCmd DeployWAR”命令来部署应用程序。你可以修改“localhost”到任意你想部署使用的服务器。你也可以修改“Silvermaster”到任意你想部署的数据库上。

- SilverCmd DeployWar localhost Silvermaster struts-example.war -f struts-example-depl-plan.xml
- SilverCmd DeployWar localhost Silvermaster struts-documentation.war -f struts-documentation-depl-plan.xml

B9. Tomcat 3.2.1 和 Apache

注意 Tomcat 4 的说明和 Tomcat 3 会有不同，但是 Tomcat 4 的 Web 连接器仍然在开发中。Tomcat 3.2.1 之前的版本不推荐和 Struts 一起使用。

- 这些说明假设你已经成功根据 Tomcat 文档整合了 Tomcat 和 Apache。
- 复制“struts-documentation.war”和“struts-example.war”到 $TOMCAT_HOME/webapps 目录下。
- 如果 Tomcat 已经运行了，要重新启动它。
- Tomcat 会生成文件$TOMCAT_HOME/conf/tomcat-apache.conf，Apache 会使用它。这个文件在你每次启动 Tomcat 时都会生成，所以复制这个文档到一个安全的位置（比如 Apache 配置目录；在 Unix 系统下通常是/usr/local/apache/conf）。
- 如果你运行的是 Tomcat 3.1，Tomcat 不会对新的应用程序生成配置项。那么就要加入下面的代码到已经保存的 tomcat-apache.conf 文件中，使用 Tomcat 的目录路径来替换$TOMCAT_HOME

```
Alias /struts-documentation
"$TOMCAT_HOME/webapps/struts-documentation
<Directory "$TOMCAT_HOME/webapps/struts-documentation>
    Options Indexes FollowSymLinks
</Directory>
ApJServMount /struts-documentation/servlet
/struts-documentation
<Location "/struts-documentation/WEB-INF/">
    AllowOverride None
    deny from all
</Location>
Alias /struts-example
"$TOMCAT_HOME/webapps/struts-example"
<Directory "$TOMCAT_HOME/webapps/struts-example>
```

```
    Options Indexes FollowSymLinks
</Directory>
ApJServMount /struts-example/servlet /struts-example
<Location "/struts-example/WEB-INF/">
    AllowOverride None
    deny from all
</Location>
```

- 上面生成的文件不知道定义在 web.xml 文件中关于扩展映射的事情，所以要进入到控制器 Servlet 的“*.do”URI 是不会被识别的。要修改这个问题，添加下面的代码到刚才保存过的“tomcat-apache.conf”中，在对应的.jsp 扩展之后：`AddHandler jserv-servlet .do`
- 保证保存过的“tomcat-apache.conf”在 Apache 的“httpd.conf”配置文件中引用了。典型的使用方法是在“httpd.conf”文件底部添加下面的代码：`Include /usr/local/apache/conf/tomcat-apache.conf`
- 为了识别“index.jsp”作为 Web 应用程序的默认页面，在“httpd.conf”中寻找“DirectoryIndex”指令。如果有一个了，那么添加“index.jsp”到列表的末尾，那么就会是这样了：`DirectoryIndex index.html index.jsp`。如果没有这个配置项，那么就添加一个：`DirectoryIndex index.jsp`。
- 重启 Apache，使得可以识别新的应用程序。那么现在就可以在浏览器中使用如下的地址来访问应用程序了：

```
http://localhost/struts-documentation
http://localhost/struts-example
```

B10. Weblogic 5.1（service pack 8）

- 获取并安装 Xerces XML 解析器(问题会使用 Sun 的参考实现来报告)。放置 xerces.jar 在 WebLogic 的系统路径下。
- 获取并解压 Struts 的二进制发布包（这个过程假设它已经提取到了 `c:\jakarta-struts` 下)。
- 在 weblogic.properties 中为每个你想配置的 Struts Web 应用程序添加配置项。比如，要使用 struts-example 应用程序，就要在 weblogic.properties 中添加如下内容：

```
weblogic.httpd.webApp.strutsexample=
c:/jakarta-struts/webapps/struts-example.war
```

- 你需要在 WebLogic 的类路径下包含 struts.jar 和应用程序使用的任意类文件，这会自动完成（除非部署解压的 Web 文件-参考下面的介绍)。
- 启动 WebLogic 服务器并在浏览器中输出 Struts 应用程序的地址。比如，要连接在步骤 3 中添加的示例应用：`http://localhost:7001/strutsexample`。
- 这个示例应用程序依赖于 Struts 指定的资源文件 ApplicationResources.properties 在类路径上展示。而 WebLogic 仅从文档中提取*.class 文件，那么这个文件就不会被发现，那么在第一次就会导致需要和 javax.servlet.ServletException:runtime failure in custom tag ‘message’相似的问题。步骤 6&7 会执行应用程序，还有其它依赖于 ApplicationResources.properties 的。
- 从*.war 文件中提取 ApplicationResources.properties，并且手动将它复制到 WebLogic 为这个应用程序创建的_tmp_war_directory 的各自包下。再次引用 struts-example

应用程序，就会是这样：c:\jakarta-struts\webapps\WEB-INF_tmp_war_strutsexample。

- 重新启动 WebLogic。现在你就可以运行应用程序了：http://localhost:7001/strutsexample

上面的步骤应该跟在部署*.war 文件之后。对于解压的 Web 应用程序，配置包含了添加 struts.jar 和/WEB-INF/classes 到 WebLogic 的类路径下。出于这个原因，我们建议部署应用程序用 war 文件到 WebLogic 下。而相同的示例应用程序可以通过修改 weblogic.properties（假设 war 文件已经提取到目录 webapps/struts-example 下了）文件的格式来提取并成功部署：

```
weblogic.httpd.webApp.strutsexample=
c:/jakarta-struts/webapps/struts-example/
```

并使用更新后的 WebLogic 类路径来启动 WebLogic。比如：

```
c:\jdk1.3\bin\java -ms16m -mx64m
-classpath c:\weblogic\lib\weblogic510sp8boot.jar;
```

额外的推荐

- Servlet 和 JSP 重载应该关闭。首先，可能付出性能的损失。根据数量不同的 JSP，请求的数量和配置检查间隔，服务器会减缓。第二，JSP 和 Servlet 的重载，我们就打开了各种 Weblogic 类加载器问题的大门，那就很难去诊断了，很难去处理和失去 HTTP 会话。
- 将 sessionid 设置成 JSESSIONID，如果 cookie 用来跟踪 session 和 jsessionid，如果 session 是基于 URL 的。（对于基于 URL 的 session，还有一些其它问题，是因为 BEA 对 session id（J2EE 不兼容）作为查询参数的编码方式引起的。特别是<bean:include>不会对基于 URL 的 session 起作用。而使用正确的 session 名称会解决一些问题。）
- 在 weblogic.properties 中配置最大 J2EE 兼容和/或最大性能的 JSP-Servlet 注册。

JSP-Servlet 支持一些初始化参数，可以自定义并获得最佳性能，最大的兼容或（下面会来说明）很容易调试：

```
weblogic.httpd.initArgs.*.jsp=\
pageCheckSeconds=-1,\
setEmptyStrings=false,\
compileCommand=./_jspCompiler_.cmd,\
workingDir=/weblogic/myserver/tmp_classfiles,\
keepgenerated=true,\
debug=true,\
verbose=true
```

在上面的示例中，批处理文件（_jspCompiler_.cmd）调用了 jikes 结果大大减少了启动时间。（jikes 是比 javac 快三倍的）批处理文件包含仅一行代码：

```
@jikes -g -nowarn %*
```

当所有测试都完成并且速度是主要关注点的时候下一个配置就可以使用了…

```
weblogic.httpd.initArgs.*.jsp=\
pageCheckSeconds=-1,\
setEmptyStrings=false,\
```

```
compileCommand=./_jspCompiler_.cmd,\
workingDir=/weblogic/myserver/tmp_classfiles,\
keepgenerated=false,\
debug=false,\
verbose=false
```

要和

```
@jikes -O -nowarn %*
```

来一起使用

Weblogic 支持 `web.xml` 中通过`<context-params>`相似的设置。（请阅读 BEA 站点的文档来获取详细信息）

对于其它问题，可以从 Struts 开发人员邮件列表中参考 WLS 5.1 SP8 的更多修改。

B11. WebSphere Application Server 3.5 FixPack 2

- 在下面的步骤中，`$WAS_HOME` 引用了安装 WebSphere 应用服务器的目录，而 `$STRUTS_HOME` 是解压的 Struts 二进制发布包的目录。
- WebSphere 在 3.5.2 版之前不支持 JSP 1.1 和 Servlet 2.2，这就是使得没有大量代码修改来运行 Struts 功能是非常复杂的。请在要使用 Struts 之前升级到 3.5.2（3.5 和补丁包 2）。可以参考 http://www.ibm.com/software/webservers/appserv/efix.html 来下载和安装 WebSphere 3.5 补丁包 2 的说明。
- 警告：Struts 在 WebSphere 的箱外不能使用。这个修改会在 WebSphere 3.5.3 中（在本文档编写时没有发布）修改这个问题。但是，使用 WebSphere 3.5.2 就可以使用 Struts 了。
- 确保 WebSphere 应用服务器已经启动了。在 Windows NT/2000 下，是“IBM WS AdminServer”服务。
- 打开 WebSphere 管理控制台。
- 打开之后，在任务工具栏选项中选择“转换 War 文件”，或者从控制台->任务菜单中来选择。这会打开“转换 War 文件”的向导对话框。
- 选择运行 Web 应用程序的 Servlet 主机，就会转换 War 文件了，比如“默认的 Servlet 引擎”，展开树控制的节点，点击 Next 按钮。
- 选择虚拟主机并关联 Web 应用程序，比如“默认主机”。点击 Next 按钮。
- 点击浏览按钮，选择`$STRUTS_HOME/webapps/struts-example.war`。点击 Next 按钮。
- 为 Web 应用程序选择目标目录，比如`$WAS_HOME/hosts/default_host`。点击 Next 按钮。
- 输入“Web 应用程序的 Web 路径”，比如 struts-example 和“Web 应用程序名称”，比如 struts-example。点击 Finish 按钮。
- 经过长时间的听过，你应该得到一个消息框，并且有文本命令“转换 war 文件”完全成功的提示。点击 Ok。
- 你需要添加 jaxp.jar 和 jaxp 兼容解析器，比如 JAXP 1.0.1 中的 parser.jar 到 struts-example 的 Web 应用程序 Servlet 目录下，比如 `$WAS_HOME/AppServer/hosts/default_host/struts-example/servlets`。
- 在这一点上，如果 WAS 3.5.2 正确实现了 Servlet 2.2，那么所有就都好了。然而，

WAS 3.5.2 在调用了 ServletContext.getResource(String) 或 ServletContext.getResourceAsStream(String) 时会返回 null。这个 manifests 本身作为异常出现在应用服务器的输出日志里，比如 default_server_stdout.log。

- 警告：不要被 Web 应用程序从管理控制器中成功启动的事实所愚弄。事实上并没有。管理控制台在撒谎。如果你试图访问 webapp，比如 http://localhost/struts-example/，就会失败。
- 因为这一点，你需要给 Struts 源代码打补丁。在三个地方调用了 getResourceAsStream:

  ResourceTag.doStartTag()

  ActionServlet.initMapping()

  PropertyMessageResources.loadLocale(String)

当然，ActionServlet 也很重要

- 将源代码从

```
// Acquire an input stream to our configuration resource
InputStream input =
    getServletContext().getResourceAsStream(config);
```

改为

```
// Acquire an input stream to our configuration resource
InputStream input = new
    java.io.FileInputStream(getServletContext().getRealPath(config
));
```

- 如果必须，对其它类也进行相似的处理。
- 重新编译 ActionServlet 并复制 .class 文件到 $WAS_HOME/AppServer/hosts/default_host/struts-example/servlets/ org/apache/struts/action/ActionServlet.class 下。
- WAS 3.5.2 的类加载器的另外一个 bug 是 Class.getResource()不能从 jar 中加载资源，所以你必须复制 $STRUTS_HOME/lib/struts-config_1_0.dtd 到 $WAS_HOME/AppServer/hosts/default_host/struts-example/servlets/org/apache/struts/resources/struts-config_1_0.dtd 中，或者连接到因特网，从 Jakarta 网站来获取 dtd。
- 在管理控制台中启动 webapp。
- 在浏览器中通过下面的 URL 来测试加载示例应用程序：http://localhost/struts-example/。

B12. WebSphere Application Server 3.5 and the Example Application

服务器：Windows 2000 服务器和 WebSphere 3.5.3 高级版

1. 启动管理服务器
2. 启动管理控制台
3. 使用转换 War 文件任务从 Struts-b1 发布包中转换 struts-example.war 文件。
4. 转换到默认服务器下，默认服务器引擎和标准的安装目录

（c:\websphere\appserver\hosts\default_host）。
5. 在 servlet 目录下创建 WEB-INF 目录，并复制 struts-config.xml，database.xml 和 web.xml 到里面（保持 WEB-INF 和所有的 TLD 在里面，WEB-INF 目录必须可见）。
6. 复制 jaxp 1.0.1（不是 1.1.1）的 jaxp.jar 和 parser.jar 到 struts-example 应用程序的 servlet 目录下。
7. 在 serlvet 目录下，使用 WinZip 打开 struts.jar。提取三个 DTD（struts-config_1_0.dtd，web-app_2_2.dtd 和 web-app_2_3.dtd）到 servlet 目录下，保证使用的文件夹名称（所以文件会提取到 servlets/org/apache/struts/resources 下）
8. 在默认服务器/默认 Servlet 引擎下的管理控制台点击 struts-example 并点击屏幕右侧的高级页签。
9. 在默认错误页面位置，输入/ErrorReporter 并点击 Apply。
10. 通过管理控制台开启默认服务器。你应该就可以在 default_host_stdout.log 文件中看到 ActionServlet 打出的完整没有异常的消息了。
11. 通过浏览器使用 http://localhost/struts-example/index.jsp 来访问应用程序。
12. 如果返回“Application not Available”，那么就回到管理控制台，右键点击 struts-example 并选择重启 WebApp。
13. 报告成功后，回去重新尝试 URL，那么就会正常显示了。

无论出于原因，一些安装不像 XML 文件那样引用 PUBLIC DTD。如果在 default_host_stdout.log 文件中看到在 DTD 注册时关于非法公共 URL 引用的错误，或者你的页面报出“cannot find //logon or //saveRegistration（比如 action 映射）”，那么就这么来做：
1. 停止默认服务器。
2. 到 servlets\WEB-INF 下编辑 web.xml 和 struts_config.xml。
3. 在 DOCTYPE 声明中，改变 PUBLIC 到 SYSTEM 并从 web.xml 中移除“-//Sun Microsystems, Inc.//DTD Web Application 2.2//EN”，从 struts-config.xml 中移除“-//Apache Software Foundation//DTD Struts Configuration 1.0//EN”
4. 保存上述设置并回到步骤 10。

仅仅作为故障排除指南

如果你得到了如“Cannot find ActionMapping 等”或“Cannot find key org.apache.struts.MESSAGE”的错误，那么应用程序仍然像在 Richard 发现的 struts-config 问题一样。上面的步骤应该是正确的。如果你得到了//logon 的 404 错误或者其它什么东西没有发现，那么在 XML 配置中还是有问题，并且不能正常初始化 Action Servlet。跟着上面关于 DTD 问题的步骤，那就应该可以了。

最后一点想法，我们没有进行大量的测试但是我不相信需要做出其它 Struts 源代码上的调整。所有关于如何在 WebSphere 上使用的工作是 WebSphere 配置的问题（我不认为还有其它的）。

如果修改 DTD 到 SYSTEM，仅在使用转换 War 下来做。Ant 似乎并不喜欢这个做法！☺