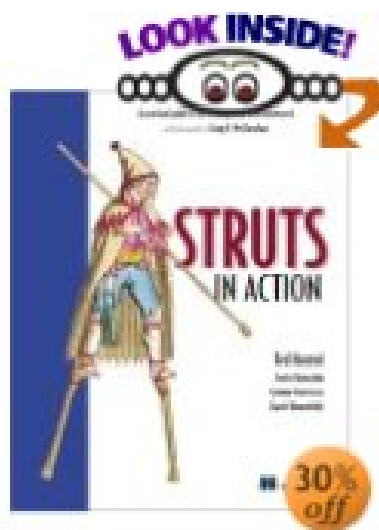


Struts 在行动

使用领先的Java框架构建Web应用

Ted Husted
Cedric Dumoulin
George Franciscus
David Winterfeldt
(著)
Eric Liu (译)



1	介绍	15
1.1	关于本书	16
1.1.1	谁创建了Struts?	16
1.2	什么是应用框架?	17
1.2.1	其它类型的框架	17
1.3	使用的技术	17
1.3.1	超文本传输协议 (HTTP).....	18
1.3.2	公共网关接口(CGI).....	19
1.3.3	Java servlet.....	19
1.3.4	JavaServer Pages	19
1.3.5	JSP标签	20
1.3.6	JavaBeans	21
1.3.7	Model 2.....	23
1.4	Struts开始于三万英尺	23
1.4.1	建立简单的Struts应用	24
1.4.2	跳到开始开发了	24
1.4.3.1	创建ActionForm	25
1.4.3.2	创建 RegisterAction.....	25
1.4.3.3	创建Struts 配置文件 (struts-config.xml)	27
1.4.3.4	创建页面	28
1.4.4	再看看	29
1.4.4.1	做了什么	29
1.4.4.2	它如何工作	29
1.4.4.3	什么还没做	32
1.5	小结	32

译者的话

年初为了构建一个需要快速开发的大型应用的时候，我找到了 Struts。这个框架使我得以从杂乱的思绪中解放出来，从而结合 EJB 构建了一个清晰的系统。从此以后，让我与 Struts 结下了不解之缘，从而也对开源项目有了新的认识。而不是仅仅限于现成的商业解决方案。

我接触的第一本书就是《Struts in Action》。这之前也只是从从 Apache 的 Struts 主页和相关链接和一些文章获取信息。当然这些信息显得杂乱。是《Struts in Action》才使我对这个强大的框架有了一个系统的理解和看法。从而用这些认识再来改进原来构建的系统。

年中，因为不开心，辞去了一家大型 IT 公司的技术总监职务，回家休养。一时闲下来，还觉得浑身不舒坦，甚至原来没显现的程序员职业病——“颈椎病”也发作了（真是奇怪！！）。不过不能闲着，后来想反正没事，国内缺乏中文的 Struts 资料和书籍，何不将此书翻译出来，供大家参考。国内还有好多不懂英语的程序员，也并不见得就不是很好的程序员。

于是开干！

以前倒是翻译过许多资料，但第一次翻译书籍，其中艰辛难以言表。尤其是有颈椎的折磨，每次总坐不到两三个小时。有时兴致所至忘记了休息，又只好去接受中医的“惨无人道”的按摩治疗和难咽的中药。

最近终于将此书初稿译完，我将逐章进行校对和发布到一些论坛上去。个人精力和水平毕竟有限，不妥之处，望各位读者能批评指正。

最后，感谢我的妻子女儿。我妻子在我赋闲的日子毫无怨言，默默支持我的工作。而我的女儿在我告诉她我忙不能给她讲故事的时候，也从没有纠缠过我，自己阅读去了。

Eric Liu

shqlau@hotmail.com

cn_fbi@yahoo.com.cn

QQ 28730869

13 October 2003

前言

此刻你手中的是几个Struts最重要的开发者的艰辛成果。Ted, Cedric, George, 和 David, 他们一起完成了一个卓越的工作, 阐释了Struts 的工作原理和如何实际使用它。如果你是一个新手, 本书将使你很快入将Struts应用到你的实际项目中。当然即使是最老道的Struts开发者也可以从中学到不少东西。

我从上世纪90年代末开始对Web框架发生兴趣。我准备写一种语言, 它可以使我能解决在进行高级开发时的某些问题--通过它可以使我从动态分配内存中的问题中解放出来。

一开始, 我真正想实现的是如何使一些Web应用开发者的的工作变得容易一点。但后来Struts那以置信的流行说明我并不是唯一在这些问题中挣扎的人—Struts满足了通常的需要。

当早期的JSP规范草案发布时 (0.91和0.92), 此文档中最迷人概念的是JSP使用了两种基本设计风格的理念。*Model 1* 设计主要表现在表单提交可以回复到创建该表单的Servlet或JSP页面。这种设计鼓励混合**表现**逻辑 (用来创建表单) 和**业务**逻辑 (用来验证表单输入和处理请求事务)。这种设计在那些开发者仅具单一技能 (比如页面设计者懂一些编程而JAVA开发者也知道一些HTML) 时经常使用。而且在时间第一的情况下也很有用 (原型必须在下一周一运行而我们还没有获得风险投资)。而经验告诉我们, *Model 1* 设计在后来会很难进行维护和升级。

相反, *Model 2* 设计是将表单提交给一个**控制器**组件。控制器组件调度相应的业务逻辑组件来完成请求的事务。业务逻辑组件和数据库交互, 并获取它在下次用户交互所需要的信息。控制器组件将响应页面的创建委派给表现组件, 表现组件只有一个目的, 就是创建响应。你可能想Model 2听起来好像更加复杂—也许甚至好想要完全扼杀简单应用。实际上, 基于Model 2 创建一个应用并不比用Model 1创建同一个应用花太多的时间。但主要的优点会很快显现出来。如果你创建了一个正确的应用, 对一层的主要修改, 应该基本不会影响到其它层 (如果有的话), 并且你可以马上在没受影响的层重用你的逻辑。

虽然, 所有这些对Web应用架构的聪明的研究正在进行, 但我的专业工作让我也对方向感兴趣。我正为一个专为长途运输行业提供信息服务的美国公司工作, 并且我们正在准备将业务扩展到欧洲。这样就产生了处理多语言和国际化问题的需要。我很快就受到了我用来实现基本MVC的单一的简单Servlet的折磨, 即它没有解决包含一个语言选择控件的问题。

对国际化问题的第一次努力是我尝试使用JSP1.1种的新的标记来创建“用户接口组件”—它最终导致了现在Struts一个标记`<bean:message>`。

紧跟着, 我加入到Sun Microsystems 并工作于Tomcat servlet 和 JSP container的创建 (我是Tomcat 4的基础 Catalina servlet 容器的主要架构者)。这个开发的大部分工作是在Apache的开源社区内进行, 作为Jakarta项目的一部分—本项目由1999年Sun 贡献了servlet 和 JSP 参考实现源代码给Apache起始。然而, 我并不满意于那时候Model 2-面向应用设计的状态, 所以我开始想怎样来解决这个问题。

虽然我已经有了一个漂亮的注意, 来解决剩下的问题, 但实际代码并没开始, 直到在2000年纪念日我和我的家人在Oregon海岸过周末的时候, 我将我的笔记本电脑也带了去, 并应起了我妻子的很大抱怨。后来成为ActionForm的最初最初版本就诞生在那个周末, 它组建开始解决大量有趣的问题。另外, 为表现组件和业务组件定义逻辑名称的理念—以及集中在一个文件中定义这些名称的理念—非常清楚地有益于解决在开发两层之间的协调和重叠问题,

以及实现层间改变相互影响的隔离目标。

通过我在Tomcat 的工作，逐渐认识到开放源代码开发的好处，这也是使Struts进入开源世界的自然选择。这个选择——以及Struts 处理一些最基本的Web应用设计问题的优雅方式——导致了它真正的令人惊讶的接受程度。成千上万的开发人员下载 Struts，克服学习曲线，通过 STRUTSUSER邮件列表提问和接收问题，并在世界各地成功地开发了许多基于Struts的应用。

当然，并不是我自己一个人能完成所有的事情。Ted, Cedric, David, 以及所有过去和现在的Struts 项目的参与者，连同George 和 Struts开发人员社区，共同使这个框架远远超过了我自己能够独立完成的状况。对它们，我表示最衷心的感谢。而对你，本书的读者，我希望你会发现Struts 是你另一件有用的武器，能够值得你去花时间修习其技术和API。

Enjoy!

Craig McClanahan

Portland, Oregon

序

到2000年，我们进入了Java时代。早期的宣传都已实现，并且出现了一些有趣的开发工具和库。我已经写了好几年的Web应用程序了。像许多开发者一样，我开始时使用JavaScript和Perl写一些简单的应用。这是一个强大的组合，但难以维护。接下来是ColdFusion，它更强大，但那时它对我的客户的钱包来说还是太昂贵了。我甚至尝试了FileMaker Pro，它很有趣，但是非常非常的专有化。

我的这些连续Web应用的主要客户是一个公共电视台。电视台的主要资金来源（现在仍然是）年度拍卖。本地的支持者提供货物或者服务，人们通过拍卖购买它们来支持电视台。当然，我们会很快在网站上递交这些高端东西的图像：艺术品，度假服务包，汽车，签名手稿，等等。

1998年，我们使用一个JavaScript和Perl写成的应用来接受这些高端物品的“预出价”。实际的出价在现场直播的电视拍卖会进行。此应用要做的就是设定起拍价。1999年，我们在销售时接受在线和电话出价。每年我们都在在线拍卖上使用不同的平台，因为每一年我们都发现已有的平台不能满足所有的需要。

因为我们已经很满意使用Apache HTTPD server，我花了一些时间来看看是否能使用初生的Jakarta 站点，在这里我发现了Struts。首先，我不能确保项目是不是还活动。但文档看起来似乎是，所以我订阅了邮件列表，看是不是真的有人在。一个示例应用已经包含在文档中。我从这个开始起步，试图决定这个框架是不是我想要的。于是我进入了“Struts应用之旅”，它描述了这个例子如何工作的，并逐屏幕进行了说明。我将这个经历递交到列表中，在这里其他一些订阅者慢慢纠正了我对某些问题的误解。我继续跟踪回应这个列表，紧握所能帮助其他人，也受到其他先行者的帮助。这个列表的访问稳定增长。到年底，Struts的架构师和领导开发者Craig McClanahan，正在寻找帮助编写1.0 release版本文档的人。2000年9月，我被推举为为一个Struts 提交者，最终我在2001年6月提交了 1.0。

接下来，我开始建立我的“More About Struts”网页。首先，首先，它仅是一个保持我所写的Struts 材料的地方。后来我开始添加一些连接到其它人发布的一些Struts“扩展”的链接，然后是逐渐开始出现的Struts文章。我的Struts资源页面增长很快，也更为许多人所知，所以我将它移到了Struts的主站。如今他已经是一个为Struts 世界的每个人所熟知的一个包含许多资源链接的页面。

Struts列表是一个用信息的宝藏，特别是因为Craig 自己也亲自来传授一些实现细节和架构理念。但是在列表文档发现最好的信息的确是个挑战。所以我开始编写一个“精华区”来只想那些最好的邮件链接，它最后成为一个大型的FAQ。2001年6月，JGuru 决定开辟一个 Struts 论坛和 FAQ，所以我们将主要的Struts FAQ 移到了 JGuru，我还继续管理它。

同时，出版商开始注意起Struts，索稿邮件也发到了我的邮箱。经过和其他提交者商量的结果，我们终于决定和Manning Publication合作。象 Apache一样，Manning 有一个长久的质量承诺。虽然我们想尽快地完成一本Struts 书籍，但是我们也想把它写成是尽可能最好的Struts书。

成果就是*Struts in Action*。它完全是一本“团队书籍”。David Winterfeldt, Struts Validator的创建者,非常乐意地撰写了Validator 一章。同样, Cedric Dumoulin, Tiles的创建者,也编写了Tiles 一章。George Franciscus 提供了关键的第一章,这有助于帮助新手上路。我们甚至让Craig写了前言(他说“宁肯进行编程”)。当然,其他 Struts 开发者和提交者在每个阶段都审阅书的手稿,我们也感谢他们提供了很多有用的意见。

嗯,关于拍卖呢?我们已经进入了使用Struts的第三个年头了。现在不是每年重写我们的应用了,而是每年进行一些改进。

Ted Husted
Fairport, New York

鸣谢

我们认识到许多人对使本书成为可能的支持和理解。我们希望那些我们最应该感谢的人——那些在像这样一个项目进行时站在我们一边的家人和朋友——已经知道我们是多么的感激他们的爱心和耐心。

当还有更多的人，他们并没有留下姓名，也为此书贡献了许多力量。

首先，是有许多志愿开发者进行了Struts的开发。数百个人的帮助才使Struts成为今天的样子。许许多多的人是通过邮件列表进行了宝贵的坦率的间接贡献。其他一些人则直接贡献了代码或者文档。比如对于 1.02 release，他们就包括Arun M. Thomas, Chris Assenza, Chris Audley, Craig R. McClanahan, David Geary, dIon Gillard, Don Clasen, Ed Burns, Eric Wu, Florent Carpentier, Jeff Hutchison, Jimmy Larsson, John Rousseau, John Ueltzhoeffer, Larry McCay, Luis Arias, Marius Barduta, Martin Cooper, Matthias Kerkhoff, Mike Schachter, Niall Pemberton, Oleg V Alexeev, Paul Runyan, Ralph Schaer, Rob Leland, Robert Hayden, Sean Kelly, Stanley Santiago, Wong Kok Kai。

好多 Struts开发人员也慷慨的评审了本书的手稿，并提供了很多有用的意见。Dan Malks就提供了本书第2章的很多注解。Richard Starr对第3章的彻底评论对我们定稿Hello World logon 例子帮助甚大。

我们也感谢那些对手稿提供反馈的其他开发者，如Martin Cooper, Vincent Masool, John Yu, Jon Skeet, Max Loukianov, James Holmes, Bill Wallace, Nathan Anderson, Cody Burleson, Darryl Thompson, James F. McGovern, Steve Wilkinson, 和Shawn Bayern。我们的技术责任编辑， Steve Wilkinson 和Stephen LeClair，更应该特别感谢，他们找出了书中尽可能的小错误。我们也非常感谢我们的组版编辑， Liz Welch，它修改了我们书中许多笔误。

本书也非常感谢我们的出版商， Marjan Bace和Manning 的编辑小组。Manning 并不是仅要我们的工作，他们要我们最好的工作。这耗费了许多的努力以及大量的时间，但读者将花费时间和金钱在此书上，他们应该值得我们向他们提供最好的东西。

关于此书

Struts 框架集合了几种相关的技术，使开发者可以创建易于构建、扩展和维护的基于标准的应用。Struts 已经是全世界开发人员不管是新手还是老手的框架选择。

Struts in Action 一步步介绍了Struts 框架。并且书中好包含展示此书介绍的最好实践技术的几个示例应用程序。本书旨在帮助那些需要关于如何使他们的应用运行在Struts下面的实际和实战技术的专业技术人员。

开发者用Struts构建Web应用一般在其应用的各部分使用几种相关的技术。一本包含这些全信息的书才可能会满足众多需求。为了能在一本书里面包含 Struts，我们试图在本书中包括HTML 标记语言， JSP页面语法，JavaBean 开发的习惯，或者类似技术的细节。嘉定读者已经熟知这些技术，以便能跟得上我们表述的例子。并且假定读者熟知URL,文档层次，web应用档案，其及其他创建并发布Web应用的相关概念。

我们也不包括基本的Java编程语言。关于 HTML，JSP，JavaBeans，和其他相关技术，已经有大量的信息。我们假定阅读此书的读者熟知 Java 语法，应用开发生命周期，以及面向对象设计概念。关于关系数据库的基础，加上JDBC技术，我们建议掌握，但不是必需。

那么我们的注意力将着眼于Web应用和Struts框架。

技术之间的关系已经说明—HTML，Java，数据库，其及其他技术—是本书的焦点，也是我们讨论最深入的范围。

然而，我对于那些不太精通Struts 依赖的相关技术的读者，书中也包括了基本的HTTP,Java servlet，JSP，核定指标签的介绍。

Roadmap

第1张总体介绍了Web应用的开发，特别介绍了Struts。我们会看到Struts 是如何编写和发布的，驱动应用的背后相关技术，以及Struts 的总体架构。在最后，我们开始创建我们的第一个Struts应用。

第2章探索Struts的架构。我们从Struts 架构的总揽开始，紧跟着会仔细看看控制六是如何在整个框架中流动的。本章最后以一个坦率的关于Struts的强项和弱点的讨论结论结尾。本章旨在给开发人员关于Struts的各个方面的真正情况的一个坚实基础。也帮助产品经理决定是否Struts会很好的适合他们的团队。

第3章开始开发一个简单的应用。像第1章的练习一样，这是也简单的登录程序，但是包括了Web应用的基本组成。其目的是给动手的开发人员在第2部分进入详细的细节之前一个总体印象。为了照顾实际，我们回过头来将该例子从Struts 1.02升级到Struts 1.1 release。

第4章探索Struts 框架的骨干—配置元素。我们也描述了配置web 开发描述符和Ant的build 文件，来帮助构建和部署你的应用。

第5章涉及了Struts ActionForm。这个关键的对象可以扮演应用中的许多角色：传输对象，防火墙， API，数据校验器，以及类型转换器。我们介绍了几个技术，来有助于你得益于调用form beans的双刃剑的Struts。

第6章讨论 Struts ActionForward。你可以从这章一探这个Web 应用中对狡猾的家伙的究竟。ActionForward我以帮助你清楚的定义你的应用的进入点，使你更容易看到你是否覆盖了你所有的基础部分。

第7章讲述的是Struts ActionMapping。这个映射是 Struts 控制器的基础。Action 类可以在一个重根据不同的任务配置成不同的形式，来达到重用。这里我们会解释如何使用Action Mapping 来控制应用的流程，并得益于每个Action 类。

第8章涉及 Struts Action 对象。这是Struts应用主要承担工作的部件—在这里通常开发者会花大量的时间。我们详细探讨了随Struts绑定的Action类，以及几个来自于 Scaffold 包的东西，以及从输入的Struts ActionForm组装业务组类的痛苦过程。

第9章涉及到了 Struts ActionServlet。这个控制器类是框架的“发言人”。它发号司令去让其它的对象完成繁重的工作。我们也会涉及一些新的方式来定制ActionServlet一边最好的迎合与你的应用或者某个特殊模块的需要。

第10章我们会探讨通常的Struts JSP 标签和 JSP页面。从用户的角度讲， web 页面就是应用，它代表了应用项要做的一切。使用Struts 的关键优点就是它可以表现内容和采集内容分离开来。在这一章，我们会详细探索 Struts JSP 标签，并简要介绍和 Struts 一起使用其他表现系统，比如 XLST 和 Velocity。大部分Struts应用都依赖 JSP 来创建动态页面，但框架本身可以和许多表现系统一起使用。

第11章我们涉及Tiles页面装配框架。象Tiles之类的动态模板系统，引入了一种新的编程模式到Web应用的表现层。一个tile 封装了一个标记块，和方法封装了Java代码块非常相似。用Tiles 构建Web页面给不受约束和混乱的HTML世界带来了一致性和灵活性。

第12章我们要讨论用户输入校验的重要内容。对Struts 核心的一个流行的扩展是 Struts Validator。这是一个非常强大的组件，它可以在相同的培植下提供基于客户端的校验和服务器的校验。我们会展示如何将校验集成到你的Struts 应用之中，使用预编写的校验器或者专门为应用编写的。

第13章涉及Struts 的国际化特征。Struts 从底层提供国际化特征。这一章国际化是如何以及在何处构架到Struts之中，你需要做的就是让它正确工作。基本主题是，今天你为一种语言开发的应用所必须做的，明天也可以加入另外的语言支持。

第14章探讨Struts的数据服务。这一章讲述如何使用一个helper 类来将 Struts Action 连接到不同的企业数据系统—包括数据库，搜索引擎，以及内容辛迪加服务。还提供了一个使用JDBC, Lucene, 以及RSS的例子。

第15章是我们的特征性应用， Artimus。这个企业级的应用将扫除所有的障碍，并在一个精简的、精彩的可重用包内展示了Struts 的关键和附加特征。认证，客户化，本地化，Scaffold, Tiles, 事务，校验器，还有许多，它是一个按顺序编排的 the best and brightest Struts 所提供的最好和最靓的特征集合。

第16章是Struts 1.1 升级指南。这一章我们仍然使用第15章的Artimus 应用，但用新的1.1 特征进行了翻新，包括 DynaForms, plug-in, 以及多重模块。如果你已经有了一个遗留的Struts 1.0 应用需要升级，这一章就是为你写的！

第17章展示了如何在Struts中使用Velocity 模板系统。我们用Velocity 模板修订了我们的登录程序，并且从两方面比较了 Velocity 模板和 JSP页面。

Code

本书的源代码由Apache 软件基金赠与。源代码现今已经是Struts 份发包的一部分，并且也可以从Manning的站点，www.manning.com/husted上获得。

本章前面部分的源代码主要由一些说明文字的片段组成。如果是要给出完整的代码，他们会被加上一个编号的程序清单；其间还有一些代码注释。

在展示源代码的时候，我们有时会使用 加黑 字体来强调特殊的部分。

在文本中，Courier 字体用来指示代码(JSP, Java, 和HTML) 和 Java 方法，JSP 标签名，以及其他源代码标识符：

- 文本中对方法的引用通常不包括方法体，因为可能存在对方法调用的不止一种形式
- 对JSP 标签的应用通常包括括号和前缀，但不包括标签接受的属性列表(<bean:write>).
- 本本中对XML元素的引用会包括括号，但不包括属性和关闭标签。
- 当一个Java 类或者标签库在某一节中首次出现时，整个包的标识符会出现在括号中，并设置为Courier 字体 (`java.util.Map`)，其他对该类的引用会设置为常规的类型。
- 当 JSP 和HTML一起散布在代码清单或者片断中时，我们将对HTML 元素使用大写形式而对JSP 元素使用小写字母。

参考

参考书目在一对方括号中给出。例如 [ASF, Artimus]。完整的出版细节和/或在本书末尾的“参考”一节给出。

作者在线

购买 *Struts in Action* 包含了对运行在Manning Publication上的私有Web论坛的访问权限，在那里你可以发表对本书的意见，提技术问题，从作者和其他用户处得到帮助。为了访问论坛并订阅它，可以在浏览其中访问www.manning.com/husted。这个页面提供了在你注册后如何访问论坛的信息，以及那些类型的信息是有效的，以及论坛的行为规则。

Manning义务是提供一个场所，在这里读者间，读者与作者间可以进行一个有意义的对话。但作者不参与大量的其他特殊事务，他们会主动贡献（免费）一些东西给论坛。我们建议读者提一些有挑战性的问题，免得他们失去兴趣！

作者在线论坛以及以前的一些讨论档案在本书付印的时候就可以从出版商的网站上访问。

关于作者

Ted Husted 是一个知名的Struts 权威，Struts 开发团队的活动成员，以及JGuru Struts 论坛的管理员。作为一个咨询师，Ted 为全美的很多专业Struts开发团队服务。Ted 也帮助管理Apache 的Jakarta 项目，该项目包含Struts 框架。Ted和他的妻子，两个孩子，四台计算机，以及一只老猫生活在纽约的Fairport。

Cedric Dumoulin 是 Struts 开发团队的活动成员，也是 Tiles 框架的作者。Cedric 目前是Lille大学的研究员。他也是一个领先的国际互联网银行公司的开发部工作。他生活在法国的 Lille。

George Franciscus 是Nexcel的负责人，该组织为许多行业提供技术和管理咨询，包括

电信，银行，人寿保险，财产和死亡保险。George 擅长于 Java, J2EE, Domino, 关系数据库，主机系统技术等。他有多伦多大学的计算机科学学士学位。George 和他的妻子和三个孩子生活在加拿大的多伦多。

David Winterfeldt 是一个Struts 贡献者，以及Commons Validator 包的作者。他是一个实现J2EE技术的大公司的高级开发人员。David 目前住在纽约市。

Craig McClanahan, Struts 框架的创立者，并为此书写了前言。Craig 是Tomcat 4 的主要架构师，以及Java Web Services Developer Pack的实现架构师。 he 现在是Sun的JavaServer Faces (JSR-127)的规范领导者和J2EE平台的Web层架构师。Craig, 作为 Struts的主要开发者，也许是提供了本书最重要的部分——我们的写作框架。

关于标题

通过结合介绍、总揽，以及如何操作的例子， *In Action* 系列书旨在帮助进行学习 **和** 铭记。根据认知科学的研究，人们记住的事情是他们自我动因进行探索的事情。

虽然Manning 中没有一个认是认知科学家，但我们确信要使学习成为持久的记忆必须经过各个阶段的探索，动手，以及对正在所学的进行重复。人们理解和记住新的事物，即使说他们掌握了它们，仅仅是要在主动地探索之后。人类的学习是主动的 (*in action*)。 *In Action* 系列书的本质是它是例子驱动的。他鼓励读者去尝试，动手编一些新的代码，并探索新的想法。

本书标题还有另一方面，也是最实际的，原因：我们的读者都是很忙碌的。他们用书来完成工作或者解决问题。他们需要一本书，允许他们很容易跳进跳出，并在他们需要的时候学习他们需要的部分。他们需要一本书来帮助他们行动 (*in action*)。这个系列的书就是针对这些读者。

关于封面

封面上的图是一个来自于波尔多荒野的牧羊人，“Berger des Landes de Bordeaux”。波尔多地区位于法国的西南部，有很多非常适合葡萄种植的阳光充足的小山，以及散布着羊群的开发和湿润的原野。踩在他的高跷上，牧羊人可以更好地在沼泽地中行进，并完成他的职责。

这个图来自于法国旅游手册 *Encyclopedie des Voyages*，由J. G. St. Saveur编写，出版于1796年。现在放松旅行是一个新的现象，像这样的旅游书非常流行，将旅游者和自助旅行者引向法国其他地区和国外的居住者。

Encyclopedie des Voyages 中多种多样的插图讲述了200年前的世界各地城镇的生动和独特的景象。可以通过穿着习惯来区分相隔数十英里的两个区域的人们区分各自属于哪一个地方。旅游指南给人的生活以一种隔离和距离的感觉，关于每一个历史时期如何同我们的过度兴奋的现在如何不同。

因为地区差异，服装风格会不同，并且非常丰富，但是随时间淡化。现在已经很难各个大陆的居住者了。也许，试着乐观地看待它，我们会对各式的个人生活得文化和视觉的多样性。或者更加多样和有趣的智慧和技术。

在 Manning 我们赞成书记涵盖的计算机业务的创新，主动。以及两个世纪前的丰富的区域生活多样性由旅游书中的图片给我们的生活带回来的乐趣。

第一部分

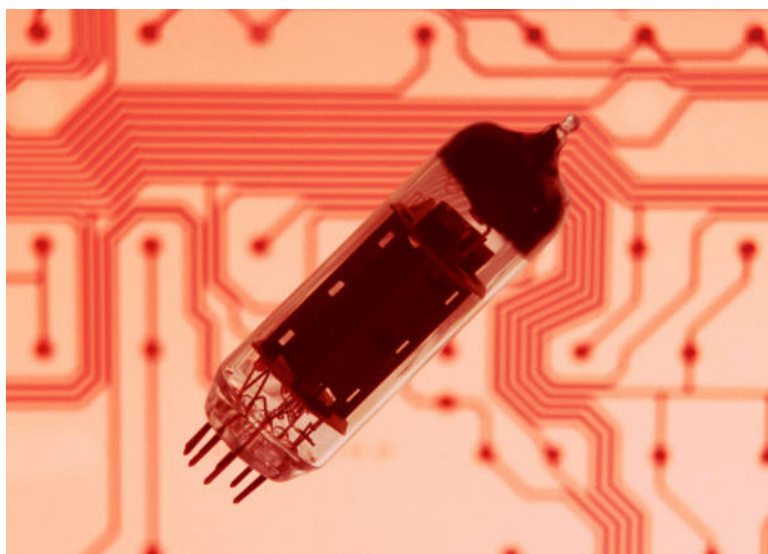
Struts 入门

第一部分是Struts 入门。我们介绍Java web 应用，检视了框架的结构，构建了两个简单的应用，并浏览了Struts 的配置组件。

1 介绍

本章包括

- 应用框架介绍
- 理解HTTP, CGI, servlet, 和 JSP
- 使用 Model 2 架构
- 构建简单的应用



The only stupid question is the one you never ask.

—佚名

1.1 关于本书

欢迎你阅读《Struts 在行动》。本书的目的是帮助Web应用开发者可以最好的使用Struts web 应用框架。

Struts 是一个开源软件，有助于开发者更加快速和容易地建立Web应用程序。Struts依靠绝大多数开发者已熟知的标准技术—比如JavaBeans, Java servlet, 以及 JavaServer Page (JSP)。通过基于标准的技术，“填空式”的软件开发方法，Struts 可以减轻在创建新项目时的令人抱怨的极费时间的工作。

1.1.1 谁创建了Struts?

Struts是Apache软件基金下Jakarta项目的一部分。除Struts之外，还有其他成功的开源产品，包括Tomcat, Ant, 和 Velocity。

开始的代码基础从2000年5月开始开发，直到2001年6月，1.0版本发布。有 30 多个开发者参与进来，并有数千人参与到讨论组中。Struts 代码基础由一个志愿的Commnitter团队来管理。到 2002年，Struts 小组共有9个志愿Commnitter。

Struts框架的主要架构设计和开发者是Craig R.McClanahan。Craig 也是Tomcat 4的主要架构师，以及Java Web Services Developer Pack的主要架构师和实现者。他现在是Sun的JavaServer Faces (JSR-127) 以及J2EE平台的Web层架构的规范领导。

Struts 在Apache 软件许可 [ASF, License]下对公众是免费的。使用此软件没有任何获得和再现成本。不象其他一些开源许可，Apache 软件许可对商业用途是友好的。你可以在你的商业项目中使用Struts,并自由分发Struts 库。你也可以将Struts 组件集成到你的框架中，就像他们是你自己编写的一样。详细情况，参见Apache Software License，www.apache.org/LICENSE。

1.1.2 为什么Struts 要开源?

现在许多非常好的Java程序和框架都是开源项目。许多开发人员为这些项目工作，同时又在象IBM, Sun Microsystems, 以及 Apple这样的公司从事其常规工作。这类软件的开发式协作有利于整个软件市场。今天，许多开源组件都集成到商业产品之中。公司可以向其客户出售其专业的文档，保证支持水平，以及其他有价值的售后服务。

当软件是自由的的时候，对市场来说它更容易得到支持。Struts 就是个典型例子。虽然它还是个很新的产品，也已经有很多文章和教程涉及到它，但却还没有什么象这样的书籍。

许多开发团队不喜欢使用不是自己内部开发的软件。开源组件提供了所有自行开发的软件的优点，但绝不将你锁定在一个只有你们团队才懂的专有解决方案上。

开源软件对所有人都是双赢的。

1.1.3 为什么叫Struts?

这个框架之所以叫“Struts”，是为了提醒我们记住那些支撑我们房屋，建筑，桥梁，甚至我们踩高跷时候的基础支撑。这也是一个解释Struts在开发Web应用程序中所扮演的角色的精彩描述。当建立一个物理建筑时，建筑工程师使用支柱为建筑的每一层提供支持。同样，软件工程师使用Struts为业务应用的每一层提供支持。

1.2 什么是应用框架？

框架（framework）是可重用的，半完成的应用程序，可以用来产生专门的定制程序[Johnson]。象人一样，软件应用的相似性比不同点要多。它们运行在相同的机器上，期望从相同的设备输入信息，输出到相同的显示设备，并且存储数据到相同的硬盘设备。工作在传统桌面应用的开发人员习惯于那些可以覆盖应用开发同一性的工具包和开发环境。构架在这些公共基础上的应用框架可以为开发人员提供可以为他们的产品提供可重用服务的基础架构。

框架向开发人员提供一系列具有以下特征的骨架组件：

- 已经知道他们在其他程序上工作的很好；
- 它们随时可以在下一个项目中使用；
- 他们可以被组织的其他团队使用；

框架是典型的**构建还是购买**命题。如果你自己构建它，在你完成时你就会理解它，但是在你被融入之前将花费多长时间？如果要购买，你必须得克服学习曲线，同样，在你可以用它工作之前得花多长时间？这里没有所谓正确答案，但许多观察者将会同意，象Struts这样的框架能提供比从头开始开发更显著的投资回报，特别是对于大型项目来说。

1.2.1 其它类型的框架

框架的概念不仅用于应用程序也可用于组件。通过此书，我们也介绍其他可以和Struts一起使用的框架。这些包括 Lucene 搜索引擎，Scaffold 工具包，Struts 验证器，以及Tiles 标签库。象应用框架一样，这些工具也提供了一些半完成的版本，可以用在用户的定制组件之中。

某些框架限制到专门的开发环境中。Struts 以及本书中涉及的组件都不是这样。你可以在很多环境中来开发Struts: Visual Age for Java, JBuilder, Eclipse, Emacs, 以及Textpad 。如果你可以用来开发Java, 你就可以用它来开发Struts。

译者注：目前很多大型公司也重视到它，它们的工具也提供相应的Struts开发支持。比如IBM WSAD，BEA WorkShop等。另外，一些公司专门提供可视化的Struts集成开发环境。

1.3 使用的技术

使用Struts的应用开发使用了大量的使能技术。这些技术并不是专门针对Struts，而是所有Java web应用都可以使用的。开发者使用Struts之类的框架是为了隐藏在诸如HTTP，CGI，以及JSP之类技术后面的繁琐的细节。作为一个Struts开发者，你并不需要知晓所有的

相关知识，但是这些基本技术的工作原理可能有助于你针对棘手问题设计出创造性的方案。如果你已经非常熟悉这些技术，你可以跳过这些章节到1.4节。

1.3.1 超文本传输协议 (HTTP)

当两个国家之间进行调解时，外交官们总是遵循一定的正式**协议**。外交协议设计来避免误解，以及防止谈判破裂。同样，当计算机间需要对话，他们也遵循一个正式的协议。这个协议定义数据是如何传输，以及他们到达后如何进行解码。Web应用程序就是使用HTTP协议在运行浏览器的计算机和运行的服务器的程序间传输数据。

很多服务器应用程序使用HTTP之外的其他协议。他们在计算机之间维护一个持久性的连接。应用服务器可以清楚的知道是谁连接上来，而且何时中断连接。因为他们知道每一个连接的状态，以及每一个使用它的人。这称之为状态协议。

相反，HTTP 是一个无状态协议。HTTP server 可以接受来自于各种客户的各种请求，并提供各种响应，即使是这个响应仅仅是说No。没有大量的协商和连接持久性，无状态协议可以处理大量的请求。这也是Internet可以扩展到很多计算机的原因。

HTTP成为通用标准的原因是其简单性。HTTP 请求看起来就像一个平常的文本文档。这使应用程序很容易创建HTTP请求。你甚至可以通过标准的程序如Telnet来手动传递一个HTTP请求。当HTTP 响应返回时，他也是一个开发者可以直接阅读的平面文本。

HTTP请求的第一行包含方法，其后是请求的来源地址和HTTP版本。HTTP请求头跟在首行后面，可以没有也可以有多个。HTTP 头向服务器提供额外的信息。可以包括浏览器的种类和版本，可接受的文档类型，浏览器的 cookies等等。7种请求方法中，GET 和 POST 是用得最多的。

一旦服务器接收到请求，他就要产生一个HTTP响应。响应的第一行称为状态行，包含了HTTP协议的版本，数字型状态，以及状态的简短描述。状态行后，服务器将返回一个HTTP响应头，类似于HTTP请求头。

如上所述，HTTP并不在请求间保持状态信息。服务器接受请求，发出响应，并且继续愉快地处理文本请求。

因为简单和效率，无状态协议不适合于需要跟踪用户状态的动态应用。

Cookies 和 URL 重写是两个在请求间跟踪用户状态的方式。cookie 是一种特殊的信息包，存储于用户的计算机中。URL 重写是在页面地址中存储一个特殊的标记，Java 服务器可以用它来跟踪用户。这两种方法都不是无缝的，是用哪一个都意味着在开发时都要进行额外的工作。对其本身来说，标准的HTTP web 服务器并不传输**动态内容**。它主要是使用请求来定位文件资源，并在响应中返回此资源。通常这里的文件使用Hypertext Markup Language (HTML) [W3C, HTML] 格式化，以使浏览器可以显示它们。HTML页面通常包含一些到其他页面的超文本连接，也可以显示其他一些内容比如图像和视频等等。用户点击连接将产生另一个请求，就开始一个新的处理过程。

标准web 服务器处理静态内容处理得很好，但处理动态内容时则需要额外的帮助手段了。

定义

静态内容直接来自于文本或数据文件，比如HTML 或者 JPEG 文件。这些文件可以随时改变，但通过浏览器请求时，却不能自动改变。

相反，动态内容是临时产生的，典型地，它是针对浏览器的个别请求的响应。

1.3.2 公共网关接口(CGI)

第一个普遍使用来产生动态内容的标准是公共网关接口Common Gateway Interface (CGI)。CGI使用标准的操作系统特征，比如环境变量和标准输入输出，在Web服务期间以及和主机系统间创建桥和网关。其他程序可以看到web server传递过来的请求，并创建一个定制响应。

当web 服务器接收到一个对CGI程序的请求，它便运行这个程序并提供它请求里面包含的信息。CGI程序运行，并将输出返回给Web server，web server则将输出响应给浏览器。CGI 定义了一套关于什么信息将作为环境变量传递，以及它希望怎样使用标准输入和输出的惯例。象 HTTP一样，CGI 是灵活和易于实现的，并且已经有大量现成的CGI 程序。

CGI 的主要缺点是它必须为每个请求运行一个程序。这是一个相对昂贵的处理方法，对大容量站点来说，每分钟有数千个请求，有可能使站点瘫痪。CGI 程序的另一个缺点是平台依赖性，一个平台上开发的程序不一定在另一个平台上能运行。

1.3.3 Java servlet

Sun公司的Java Servlet 平台直接解决了CGI 程序的两个主要缺点。

首先，servlet 比常规CGI程序提供更好的性能和资源利用。其次，一次编写，随处运行的JAVA特性意味着servlet在有JVM的操作系统间是轻便可移动的。

Servlet看起来好像是一个微小的web server。它接受请求并产生响应。但，和常规web servers不同，servlet API 是专门设计来帮助Java 开发人员创建动态应用的。

Servlet 本身是编译成字节码的Java 类，就像其他Java对象一样。Servlet 访问HTTP特定服务的API，但仍然有另外一个Java 对象运行于程序之中，并管理所有的Java资产。

为了使常规web servers能访问servlet，servlet 被安插在一个容器之中。Servlet容器连接到Web服务器。每个servlet 都可以宣称它可以处理何种样式的URL。当符合所注册样式的请求到达，web server 将请求传递给容器，容器则调用响应的servlet。

但不像CGI程序，并不是针对每个请求创建一个新的servlet。一旦容器实例化一个servlet，它就仅为每个新的请求创建一个新的线程。Java 线程可比使用CGI程序的服务器处理开销小多了。

一旦servlet 被创建，使用它处理额外的请求仅带来很小的额外开销。Servlet 开发人员可以使用init() 方法保持对昂贵资源的引用，比如到数据库或者EJB Home 接口的连接，以便它们可以在不同的请求间进行共享。获得这些资源要耗费数秒时间，这比大多数冲浪者愿意等的时间要长些。

Servlet的另一个好处是，它是多线程的， servlet 开发人员必须特别注意确保它们的servlet是线程安全的。学习servlet 编程，我们推荐*Java Servlets by Example*，作者Alan R. Williamson [Williamson]。

1.3.4 JavaServer Pages

虽然servlets 对CGI 程序来说前进了一大部，但它也不是万能药。为产生响应，开发人员不得不使用大量的println 语句来生成HTML。像这样的代码：

```
OUT.PRINTLN("<P>ONE LINE OF HTML.</P>");
```

```
OUT.PRINTLN("<P>ANOTHER LINE OF HTML.</P>");
```

在产生HTTP 响应的Servlet中是很普遍的。也有一些库有助于你产生HTML，随着应用越来越复杂，Java 开发人员将不再扮演HTML 页面设计的角色。

同时，大多数项目经理更喜欢将团队分成不同的小组。它们喜欢HTML 设计人员处理表现层的工作，而Java 工程师则专注于业务逻辑。单独使用servlet鼓励混合标记和业务逻辑，很难区分团队人员的专业工作。

为解决这个问题，Sun提出了一个结合脚本和模板技术到一个组件中的服务器页面技术。为创建JSP页面，开发者按同样创建HTML页面的方式创建页面，使用相同的HTML语法。为将动态内容引入页面，开发人员可以将脚本元素置入页面之中。脚本元素是一些标记，封装了可以被JSP识别的逻辑。你可以在JSP页面中很容易的识别出脚本元素，他们被封装在一对`<%` 和 `%>`标记中。例如，要显示页面的最后修改日期，开发人员可以将以下代码放入页面中：

```
<B>THIS PAGE WAS ACCESSED AT <%= NEW DATE() %></B>
```

有三种不同的脚本元素：

表达式，脚本小程序和宣称。如表1.1所示：

表格 1.1 JSP 脚本元素

元素	目的
表达式	JAVA代码，封装在 <code><%=</code> 和 <code>%></code> 之中，用来计算JAVA语句的值，并将结果插入SERVLET的输出之中
脚本程序	JAVA代码，封装在 <code><%</code> 和 <code>%></code> 之中，常用来创建动态内容
宣称	JAVA代码，封装在 <code><%!</code> 和 <code>%></code> 之中，常用来创建动态内容

为区分JSP 页面，程序将文件存为扩展名.jsp。当一个客户请求JSP 页面时，容器将页面翻译成Java servlet源代码文件，并将它编译成Java 类文件--就象你写的servlet文件一样。在运行时，容器也能检测JSP文件和相应类的最后更新时间。如果，JSP 文件自上次编译以来修改了，容器将重新翻译和编译JSP文件。

项目经理现在可以将表现层分派给HTML开发人员，将业务逻辑工作分派给JAVA开发人员。重要的是记住， JSP 页面事实上是一个servlet。你可以在servlet做的，也可以在JSP 中做。

1.3.5 JSP标签

脚本元素仅是两种产生动态内容的方式之一。Scriptlet 是快捷、简单、强大的手段但要求开发者在HTML中混合Java代码。经验告诉我们，混合业务逻辑到JSP页面中将导致难以维护的应用和最小的可重用性。一个可选的方案是使用JSP 标签（tags）。JSP 标签可以和HTML标记混合使用，就象它们是原生HTML 标记一样。一个 JSP 标签可以代表许多Java 语句，但是所有的开发者都需要了解如何在页面中插入标记。源代码隐藏在Java类文件之中。

为在其他页面中使用同一代码，只需要在该页面中重新插入相同的标签。如果标签代表的代码改变了，所有的标签都将使用更新的版本。而使用标签的 JSP 页面并不需要进行修订。

JSP 标记比 scriptlet 提供了更好的可重用性，也更易被页面设计者使用，因为它们看起来很像 HTML 标记。

有大量的现成的 JSP 标签库 (tags libraries) 可用，他们完成很多有用的功能。其中就有新的 JSP 标准标签库 (JSTL)。这是一个新的标准，提供丰富的可重用的 JSP 标签库。

关于 JSTL 的详细情况，我们高度推荐《*JSTL in Action*》，作者 Shawn Bayern [Bayern]。Struts 可以很好的和 JSTL 以及其他公开标签库一起工作，甚至是你自己写的标签库。

关于 JSP 的详细内容，我们强烈推荐《*Web Development with JavaServer Pages*》，作者 Duane K. Fields, Mark A. Kolb, 和 Shawn Bayern [Fields]。

JSP 是 Struts 开发者工具箱的一部分。大多数 Struts 开发者使用 JSP 和定制标记来创建应用的动态内容。

1.3.6 JavaBeans

JavaBeans 是一种 Java 类，它遵从一定的设计模式，使他们易于和其他开发工具和组件一起使用。

定义

JAVABEAN 是一种 JAVA 语言写成的可重用组件。为写成 JAVABEAN，类必须是具体的和公共的，并且具有无参数的构造器。JAVABEANS 通过提供符合一致性设计模式的公共方法将内部域暴露称为属性。众所周知，属性名称符合这种模式，其他 JAVA 类可以通过自省机制发现和操作这些 JAVABEAN 属性。

JSPS 和 ASPs

MICROSOFT 和 SUN 都提供它们各自品牌的服务器页面。SUN 提供 JAVASERVER PAGES (JSP) 而 MICROSOFT 提供 ACTIVE SERVER PAGES (ASP)。JSP 和 ASP 都设计来时开发者能从后端系统产生动态页面。

虽然表面看起来相似，ASP 和 JSP 仍有一些不同之处：

JSP 是平台独立性的，一次编写，随处运行；

开发者通过 JAVA COMMUNITY PROCESS (JCP) 指引方向；

JSP 开发者可以通过定制标记扩展 JSP 标记集；

JAVABEANS 和 ENTERPRISE JAVABEANS (EJB) 可以和 JSP 一起使用，增强可重用性和减小维护。

JSP 可以存取其他一些 JAVA 库，包括 JAVA 数据库 CONNECTIVITY (JDBC)，JAVA MAIL，JAVA MESSAGE SERVICE (JMS)，以及 JNDI。

JSP 编译成二进制类文件，不需要在每次请求时进行解释；

JSP 有广泛的支持，包括工具，容器和服务；

该你做的

整本书都在写要编写你自己的JSP 标签，这里是这个过程的一个快速总揽：

- 1 编写一个累，通过实现实现 `DOSTART()` 或者 `DOEND()` 方法来实现 `JAVAX.SERVLET.JSP`。
- `TAGEXT.TAGSUPPORT` 或者 `JAVAX.SERVLET.JSP.TAGEXT.BODYTAGSUPPORT` 接口。这些方法获得一个 `JSPWRITER` 对象，你可以用它来输出你需要的HTML内容。
- 2 创建一个标签库描述文件(TLD)来将你的新建的类，映射到一个标签名称。
- 3 在你的WEB应用描述符(WEB.XML)中定义你的 `<TAGLIB>` 元素。通过在JSP页面的顶部放置下面的语句：[%@TAGLIB URI="/TAGS/APP.TLD PREFIX="APP" %](#)
来告诉JSP页面泥浆使用你自己的标签库。
- 4 这个语句导入将在本页中使用的标签库，并分配它一个前缀。关于更多细节，请参考JSP 标签库技术页面。

JavaBean 设计模式提供两种类型的方式来访问bean的内部状态：访问器 (*accessor*) 用来读JavaBean的状态，修改器 (*mutator*) 用来改变 JavaBean的状态。

Mutator 通常以小写的 *set* 前缀开始，后跟属性名。属性名的第一个字母必须大写。返回值通常是void，因为mutators 仅仅改变属性的值，而不返回它们。简单属性的mutator 在其方法体中可能只有一个参数，可以是各种类型。Mutator 也可根据其前缀称为设置器 *setters*。例如，对Double类型的属性weight的mutator方法体可能是：

```
public void setWeight(Double weight)
```

相似的设计模式也用于访问器方法的创建。Accessor通常以小写的 *get* 为前缀，后跟属性名。属性名的第一个字母必须大写。返回值必须匹配相应的修改器方法的参数。简单属性的Accessor在其方法体中不能接受参数。同样，访问器accessor 也经常称为获取器 *getter*。属性weight 的访问器方法体可能是：

```
PUBLIC DOUBLE GETWEIGHT()
```

如果访问器返回一个逻辑值，这种情况下有个变体模式。不使用小写的 *get*，逻辑属性的访问器可以使用小写的 *is* 前缀，后跟属性名。属性名的首字母必须大写。返回值肯定是逻辑值，不管是 `boolean` 还是 `Boolean`。逻辑访问器在其方法体中不能接受参数。On属性的逻辑访问器 的方法体可能是：

```
PUBLIC BOOLEAN ISON()
```

在使用JavaBean时，规范的方法体扮演了极为重要的角色。其他组件可以使用Java 的反射API 通过查找前缀为 *set*, *is*, 或者 *get* 的方法来发现JavaBean的属性。如果一个组件在一个JavaBean中发现一个这样的方法，它就知道这个方法可以用来访问或者改变JavaBean的属性。

Sun 引入JavaBeans是为了工作于GUI组件，但它们已经用在 Java 开发的各个方面，包括Web应用。Sun 的工程师开发了 JSP 标签的扩展类时，他被设计来可以和JavaBeans一起

工作。一个页面的动态数据可以当作一个JavaBean来传递,并且 JSP 标记可以随后使用bean的属性来定制页面的输出。

1.3.7 Model 2

Servlet/JSP 规范的 0.92 版描述了一个应用中使用 servlet 和 JSP 的架构。在其后的规范中, **Model 2** 这个叫法消失了,但它已经在 Java web 开发人员中非常通用了。

根据 Model 2, servlet 处理数据存取和导航流, JSP 处理表现。Model 2 使 Java 工程师和 HTML 设计者分别工作于它们所擅长和负责的部分。Model 2 应用的一部分发生改变并不强求其他部分也跟着发生改变。HTML 开发人员可以改变程序的外观和感觉,并不需要改变后端 servlet 的工作方式。

Struts 框架是基于 Model 2 的架构。它提供一个控制器 controller servlet 来处理导航流和一些特殊类来帮助数据存取。随框架也提供一个充实的标签库,以使 Struts 易于和 JSP 一起使用。

1.4 Struts开始于三万英尺

抓紧你的帽子!

既然我们已经讲了一些基本知识,我们现在可以进行一个Struts的飞速之旅了。在我们打开框架吃到一些果子之前,我们先了解一个大概模样。

Struts 使用 Model 2 架构。Struts 的 *ActionServlet* 控制导航流。其他 Struts 类,比如 *Action*, 用来访问业务逻辑类。当 *ActionServlet* 从容器接收到一个请求,它使用 URI (或者路径“path”) 来决定那个 *Action* 将用来处理请求。一个 *Action* 可以校验输入,并且访问业务层以从数据库或其他数据服务中检索信息。

为校验输入或者使用输入来更新数据库, *Action* 需要知道什么指被提交上来。并不是强制每个 *Action* 从请求中抓取这些值,而是由 *ActionServlet* 将输入绑定到 *JavaBean* 中。输入 bean 是 Struts *ActionForm* 类的子类。*ActionServlet* 通过查找请求的路径可以决定使用哪个 *ActionForm*, *Action* 也是通过同样的方法选取的。*ActionForm* 扩展 `org.apache.struts.action.ActionForm` 类。每个都必须以 HTTP 响应进行应答。通常, Struts

Action 并不自行加工响应信息,而是将请求转发到其他资源,比如 JSP 页面。Struts 提供一个 *ActionForward* 类,用来将一个页面的路径存储为逻辑名称。当完成业务逻辑后, *Action* 选择并向 *Servlet* 返回一个 *ActionForward*。*Servlet* 然后使用存储在 *ActionForward* 对象中的路径来调用页面完成响应。

Struts 将这些细节都绑定在一个 *ActionMapping* 对象中。每个 *ActionMapping* 相对于一个特定的路径。当某个路径被请求时, *Servlet* 就查询 *ActionMapping* 对象。*ActionMapping* 对象告诉 *Servlet*, 哪个 *Actions*, *ActionForms*, 和 *ActionForwards* 将被使用。

所有这些细节,关于 *Action*, *ActionForm*, *ActionForward*, *ActionMapping*, 以及其他一些东西,都在 *struts-config.xml* 文件中定义。*ActionServlet* 在启动时读取这个配置文件,并创建一个配置对象数据库。在运行时, Struts 应用根据文件创建的配置对象,而不是文件本身。图 1.1 显示了这些组件是如何一起工作的。

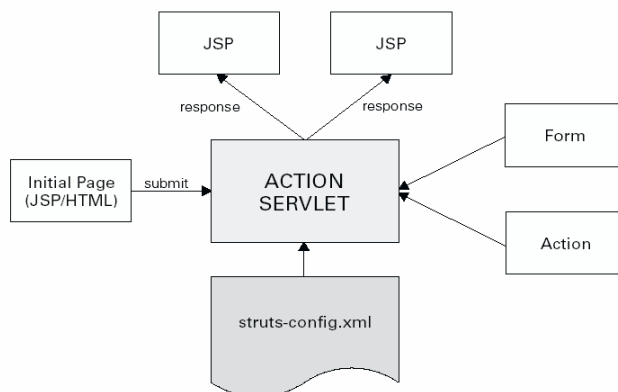


图 1-1 Struts 组件

不管你相信与否，你已经知道了足够的东西，可以组装一个简单的Struts应用了。它并不需要很多工作，但它显示了Struts实际是如何工作的。

1.4.1 建立简单的Struts应用

开发者进行开发，我们大多数则从例子中学习最好的方法。虽然我们花了几乎一页不到的篇幅来讲Struts是如何工作的，我们也需要建立一些什么，以便你可以看到如何完成这个工作。完成了这章，我们将建立一个非常简单但全功能的web 应用。这个程序将用来注册用户和用户密码。一旦你完成它，你将接触到部署你的Web应用所需要的所有Struts组件。

1.4.2 跳到开始开发了

当我们渴望要创建点什么时，接下来再设置环境并遇到障碍的时候就可能让所有人都觉得受挫。根据这章所述，你需要的是一个 Java Development Kit (JDK)，一个web 容器（比如Tomcat），以及一个简单的文本编辑器。如果你还没有一个Java 开发环境和web 容器，下面是你该做的：

下载并安装JDK 1.4.

下载并安装 Tomcat 4 .

校验Tomcat 是否工作正常。

1.4.3 落到实处

我们的第一个Struts程序将是一个用户注册程序。用户将看到一个注册屏幕，包含3个字段：用户名，密码和密码确认。成功的注册要求两次密码相符。如果注册成功，控制将转向一个页面，显示注册成功`successful!`。如果两次输入密码不同，控制流将转向一个显示失败的页面。

这个简单的练习将展示以下内容：

- 创建HTML 表单；
- 从HTML 表单获取输入；
- 处理输入（业务逻辑）；
- 根据动态输入改变控制流；

为完成这个程序，你需要建立：

- 一个ActionForm
- 一个Action
- struts-config.xml 文件
- 三个页面

就这些！

1.4.3.1 创建ActionForm

ActionForm 是一个JavaBean，扩展了 org.apache.struts.ActionForm类。这个对象捕获通过请求传送的输入。当浏览器提交一个表单，它在请求中为每个表单中的字段创建一个参数。ActionForm 针对每个HTML表单中的字段具有一个对应的属性。ActionServlet 匹配请求中的参数和ActionForm中的属性。当匹配好后，ActionServlet 为属性调用setter方法，并将请求中的值传入。在我们的练习中，表单中的username 字段需要一个 setUsername(String)方法。password 字段需要setPassword1(String) 和 setPassword2(String)方法。这些方法负责组装隐藏在RegisterForm JavaBean中的实例变量。RegisterForm 的源代码显示在清单1中。

```
PACKAGE APP;

IMPORT ORG.APACHE.STRUTS.ACTION.*;

PUBLIC CLASS REGISTERFORM EXTENDS ACTIONFORM {

    PROTECTED STRING USERNAME;

    PROTECTED STRING PASSWORD1;

    PROTECTED STRING PASSWORD2;

    PUBLIC STRING GETUSERNAME () {RETURN THIS.USERNAME;};

    PUBLIC STRING GETPASSWORD1 () {RETURN THIS.PASSWORD1;};

    PUBLIC STRING GETPASSWORD2 () {RETURN THIS.PASSWORD2;};

    PUBLIC VOID SETUSERNAME (STRING USERNAME) {THIS.USERNAME = USERNAME;};

    PUBLIC VOID SETPASSWORD1 (STRING PASSWORD) {THIS.PASSWORD1 = PASSWORD;};

    PUBLIC VOID SETPASSWORD2 (STRING PASSWORD) {THIS.PASSWORD2 = PASSWORD;};

}
```

代码清单 1.1 RegistrationForm

创建一个文件，取名为RegisterForm.java，内容示于代码清单1.1。存储在 <Base Directory>/webapps/register/WEB-INF/classes/app下。默认情况下，<Base Directory> 可能是 C:/PROGRAM FILES/APACHE TOMCAT 4.0。对于其他容器，使用其classes目录的路径来部署我们的Register 程序。

1.4.3.2 创建 RegisterAction

Action 一个 Java 类，扩展了 org.apache.struts.Action。ActionServlet 组装 ActionForm，然后将其传递给Action。Action 通常负责输入校验，存取业务信息，以及决定向Servlet返回哪个ActionForward。

现在，创建一个文件，命名为RegisterAction.java，其内容为代码清单1.2的内容：

```
package app;
```

```

import org.apache.struts.action.*;
import javax.servlet.http.*;
import java.io.*;

public class RegisterAction extends Action {
    public ActionForward perform (ActionMapping mapping,
    ActionForm form,
    HttpServletRequest req,
    HttpServletResponse res) {
        // ①Cast the form to the RegisterForm
        RegisterForm rf = (RegisterForm) form;
        String username = rf.getUsername();
        String password1 = rf.getPassword1();
        String password2 = rf.getPassword2();
        // ②Apply business logic
        if (password1.equals(password2)) {
            try {
                // ③Return ActionForward for success
                UserDirectory.getInstance().setUser(username,password1);
                return mapping.findForward("success");
            } catch (UserDirectoryException e) {
                return mapping.findForward("failure");
            }
        }
        // ④Return ActionForward for failure
        return mapping.findForward("failure");
    }
}

```

代码清单 1.2 RegisterAction.Java

将文件存放在<Base Directory>/webapps/register/WEB-INF/classes/app 目录下。

虽然很简单，但是我们的RegisterAction 却做了Action 的典型事情。在①，输入ActionForm 被转换为RegisterForm。我们就可以获取username, password1, 和 password2的内容。如果两次密码匹配②，我们将用户添加到 *UserDirectory* 中③，并返回与*success* 对应的ActionForward 。*UserDirectory* 是一个 helper 类，它记录usernames 和 passwords 到一个标准的属性文件之中。否则，返回与*failure* 对应的ActionForward 。当我们在下一步创建struts-config 文件时，我们将标识代表success和 failure的ActionForward 对象。

注：*STRUTS 1.1*提供另外一个进入方法，名为EXECUTE。这个方法提供更好的意外处理，但不和*STRUTS 1.0* 的PERFORM方法一样。在这里我们将使用PERFORM 方法，以使我们的例子可以运行在两个版本之下。

1.4.3.3 创建Struts 配置文件 (struts-config.xml)

struts-config.xml 文件包含了ActionServlet 需要用来处理对应用请求的详细信息。为了练习，我们创建一个空壳的struts-config.xml 文件。你需要做的是填入一些细节。

文件存储在<BaseDirectory>/webapps/register/WEB-INF/目录下，需要改变的是：首先，添加/register 到<action>元素的 path 属性。ActionServlet 使用Web容器转发给它的URI来选择正确的Action 类。URI 和ActionMapping 的path 属性匹配。这里，请求给出的路径必须在去除前缀和后缀后和/register 匹配。前缀或后缀通常是/do/ 或者.do。 我们的练习中，将后缀设置为.do。当URI 具有一个.do 扩展名，容器就知道将请求转发给ActionServlet。Struts会自动去除 扩展名，所以我们在配置时不必加上它们。

下一步添加

registerForm

到<action> 元素的 name 属性。<action> 元素使用name 属性来识别哪个ActionForm 将被创建，并将提交的表单组装给他。

然后，添加

app.RegisterAction

到<action> 元素的 type 属性。ActionServlet 使用这个属性来识别将用来处理请求的Action 类。

接下来，在<forward> 元素下，添加

success

到 name 属性，并且

/success.html

到 path 属性。最后，再在另一个<forward>下添加

failure

到 name 属性，

/failure.html

到 path 属性。

这些元素将创建ActionForward 对象,我们将用它来选择程序的控制流。<forward> 元素定义了RegisterAction中使用的逻辑名称之间的关联。

Struts-config.xml 源代码见代码1.3。

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE struts-config PUBLIC
"-//Apache Software Foundation//DTD Struts Configuration 1.0//EN"
"http://jakarta.apache.org/struts/dtds/struts-config_1_0.dtd">
<struts-config>
<form-beans>
<form-bean name="registerForm" type="app.RegisterForm"/>
</form-beans>
<action-mappings>
<action path="/register"
type="app.RegisterAction"
name="registerForm">
```

```

<forward name="success" path="/success.html"/>
<forward name="failure" path="/failure.html"/>
</action>
</action-mappings>
</struts-config>

```

代码清单 1.3 Struts-Config.XML

Struts框架将struts-config.xml 文件视为部署描述符使用。它使我们可以创建和改变 ActionMapping 和路径的关联而不用重新编译java类。我们也可以改变页面之间的连接，而不改变JSP模板。

1.4.3.4 创建页面

最后的步骤是创建success.html, failure.html, 以及register.jsp 页面。

3个文件的源代码如下,见代码清单1. 4,1. 5,1.6。存放在<Base Directory>/webapps/register 目录下。

```

<HTML>
<HEAD>
<TITLE>SUCCESS</TITLE>
</HEAD>
<BODY>
Registration succeeded!
<P><A href="register.jsp">try another?</A></P>
</BODY>
</HTML>

```

代码清单 1.4 Success HTML

```

<HTML>
<HEAD>
<TITLE>FAILURE</TITLE>
</HEAD>
<BODY>
Registration failed!
<P><A href="register.jsp">try again?</A></P>
</BODY>
</HTML>

```

代码清单 1.5 Failure.html

```

<%@ taglib uri="/WEB-INF/struts-form.tld" prefix="form" %>

```

```
<form:form action="register.do">
UserName:<form:text property="username"/><br>
enter password:<form:password property="password1"/><br>
re-enter password:<form:password property="password2"/><br>
<form:submit value="Register"/>
</form:form>
```

代码清单 1.6 Register.jsp

这时，所有构建一个简单Struts应用的工作都做完了。
现在，试一下运行如何。

如果，Tomcat 没有运行，启动它。

在浏览器中输入以下地址：

<http://localhost:8080/register/register.jsp>

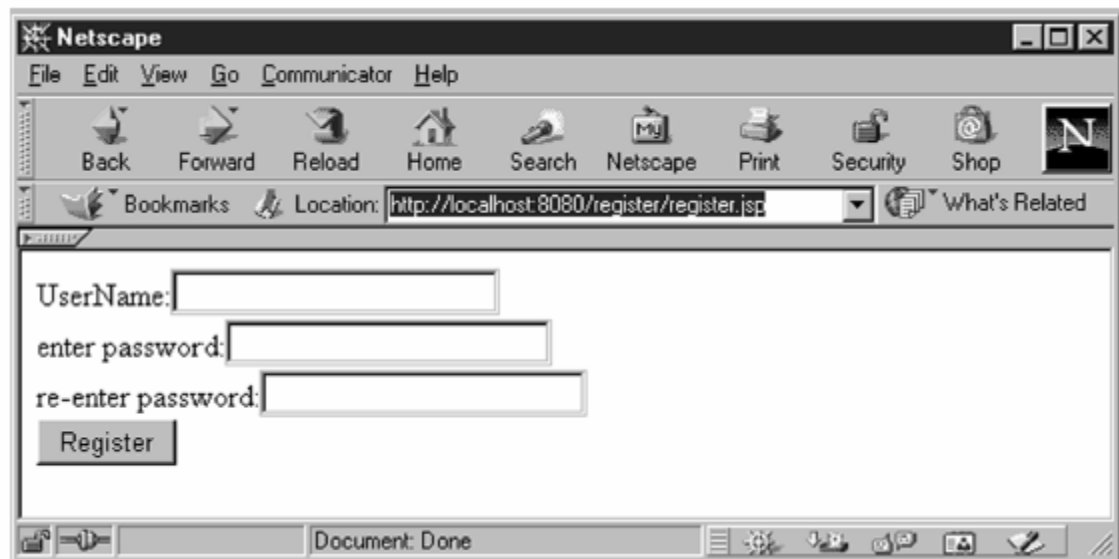


图 1-2 Registration 程序主界面

1.4.4 再看看

让我们再回头看看，我们做了什么，它如何工作的，以及我们还需要做些什么。

1.4.4.1 做了什么

建立Register应用我们实际上完成了以下内容：

- RegisterForm ActionForm
- RegisterAction Action
- 3个页面

1.4.4.2 它如何工作

当你知识浏览器到地址<http://localhost:8080/register/register.jsp>，Tomcat按通常情况加工这个页面。输入username 和password，点击Register 提交页面。浏览器在请求中post表单的内容。容器检查请求将送到哪一个注册的路径去。然后请求被转发到ActionServlet ，并由 RegisterAction来处理。在返回成功或失败之前，RegisterAction 校验输入的有效性。最后 servlet将控制根据返回的ActionForward转发到响应页面。图1.3 是程序结构。

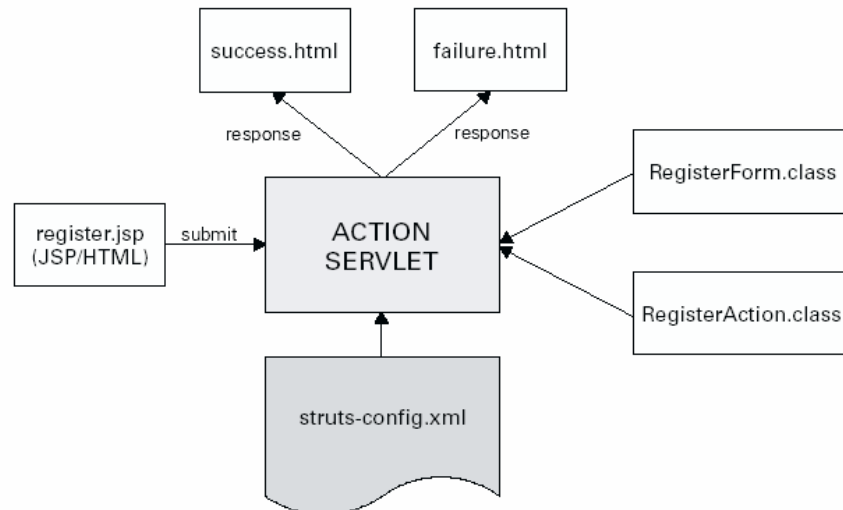


图 1-3 Registration 程序结构

再看看代码1.6中的Register.jsp，可以看到表单是提交给URI /register。但是如果你观察正提交的页面，会发现是提交给register.do。Struts 表单标记自动加上.do 前缀。当我们设置程序骨架时，我们要求所有匹配*.do的请求都传递给ActionServlet。

当接收到一个请求，ActionServlet 做的第一件事情就是查找ActionMapping来匹配请求的路径。ActionMapping是Struts根据 struts-config.xml 文件创建的JavaBean。我们给出了XML 的具体文件，但运行时，Struts 引用的是对象，而不是XML 文档。

从代码1.3中可以看到，我们使用这个元素创建了一个到path /register的映射：

```
<action
  path="/register"
  type="app.RegisterAction"
  name="registerForm"
  input="/register.jsp">
```

然后，ActionServlet 检查是否有name 属性和这个映射相关：

```
<action
  path="/register"
  type=" app.RegisterAction"
  name="registerForm"
  input="/register.jsp">
```

这里 /register 映射通过registerForm的名称标识了一个form bean。ActionServlet 使用这

个名字属性来查找相应的ActionFormBean 对象。由Form Bean标识的类型（type）用来创建ActionForm 对象：

```
<form-beans>
<form-bean
name="registerForm"
type="RegisterForm"/>
</form-beans>
<action-mappings>
<action
path="/register"
type=" app.RegisterAction"
name="registerForm"
input="/register.jsp">
<forward
name="success"
path="/success.html"/>
<forward
name="failure"
path="/failure.html"/>
</action>
</action-mappings>
```

这里，servlet将使用RegisterForm 类:

```
<form-beans>
<form-bean
name="registerForm"
type="app.RegisterForm"/>
</form-beans>
```

一旦RegisterForm 被实例化， ActionServlet 就试图为请求中的输入域调用RegisterForm 的setter方法。在例子中，它们是setUsername, setPassword1, 和 setPassword2。如果某个setter方法不存在，该参数就会被忽略。

ActionMapping 对象的type 属性是ActionServlet 用来实例化ActionForm的类名。这里，将使用你创建的RegisterAction 对象。RegisterAction对象的perform 方法被调用，并传递一个到在前面一步中创建和组装的RegisterForm的引用：

```
<action
path="/register"
type="app.RegisterAction"
name="registerForm"
input="/register.jsp">
```

```
<forward
name="success"
path="/success.html"/>
<forward
name="failure"
path="/failure.html"/>
</action>
```

依赖于perform 方法的执行结果，将返回两个ActionForward之一。findForward() 方法使用一个String 参数来查找 与name属性相匹配的forward 对象。而path 属性则由ActionServlet 用来决定用哪个页面来完成响应：

```
<forward
name="success"
path="/success.html"/>
<forward
name="failure"
path="/failure.html"/>
```

1.4.4.3 什么还没做

为了尽快入门，我们省略了一些内容。我们没有编译源代码，而依赖于这个程序绑定的早就编译了的类文件。我们想给你一个机会来开始开发Struts程序，而没有心烦的日常事务，比如 Ant 构建文件。

在第3章，我们开发另外一个程序来展示框架的其他特征。在那里，我们也介绍Ant和一个编辑器jEdit的入门。我们业开始深入介绍Struts组件。

本书的第2部分非常详细的讨论了框架组件。在第4部分，我们将它们和起来开发一个实际的应用Artimus。

1.5 小结

在本章，我们介绍了Struts应用框架。并介绍了基本知识，关于HTTP, CGI,, Java servlet, JSP,以及JavaBeans。我们也说明了Model 2 应用架构，以及他如何用来结合运用servlets 和 JSPs 在同一个应用之中。

本章末尾，我们快速建立第一个简单的Struts应用。现在你已经有关于Strtus Web应用程序象什么的初步印象，下一章我们将更深入的讨论Strtuts框架的理论和具体实践。