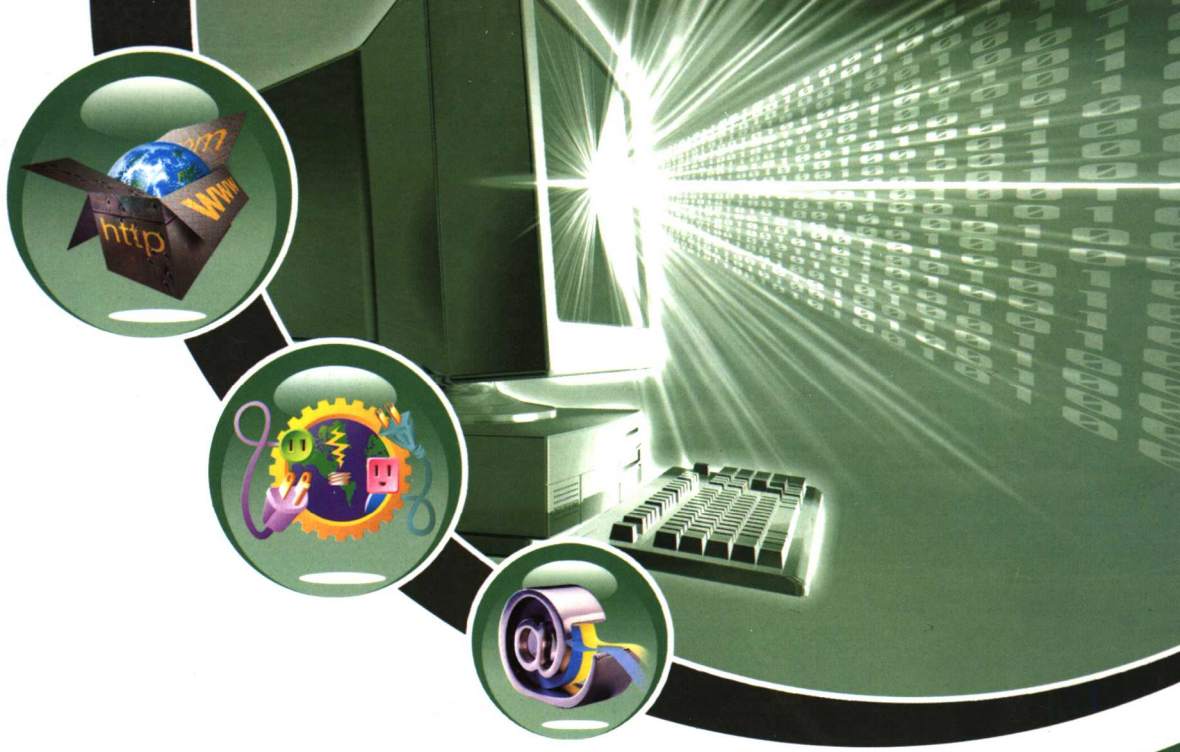


孙更新　宾　晟　周　峰　编著

Struts框架结构的Java Web开发技术

基础与实践教程

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
http://www.phei.com.cn

Struts框架结构的Java Web开发技术基础与实践教程

孙更新　宾　晟　周　峰　编著

電子工業出版社
Publishing House of Electronics Industry
北京 · BEIJING

内 容 简 介

本书讲解了Struts基础知识和Tiles、Validator框架以及如何构建插件等内容，最后还介绍了Hibernate框架和Spring框架的基本知识，并在此基础上介绍了在Struts+Hibernate+Spring开发模式下进行Web应用开发的基本方法。最后通过两个实际开发实例，使读者能够结合实际，快速、高效、灵活地设计出专业的基于Struts框架的企业级Web应用。本书采用理论与实例结合、相互渗透、逐步引导的方法，通过实例剖析技术的具体应用，使读者能十分容易入门并逐步精通。

本书可作为初学者的入门教程，更适用于有一定Web编程基础的读者，通过本书的学习读者可以迅速提高自己的编程水平，达到实际商业开发的要求。

图书在版编目（CIP）数据

Struts框架结构的Java Web开发技术基础与实践教程/孙更新，宾晟，周峰编著.—北京：电子工业出版社，2008.1

ISBN 978-7-121-05289-7

Ⅰ.S… Ⅱ.①孙…②宾…③周… Ⅲ.①软件工具—程序设计—教材②Java语言—程序设计—教材 Ⅳ.TP311.56 TP312

中国版本图书馆CIP数据核字（2007）第169579号

责任编辑：李红玉 戴 新
印 刷：北京天竺颖华印刷厂
装 订：三河市金马印装有限公司
出版发行：电子工业出版社
北京市海淀区万寿路173信箱 邮编：100036
北京市海淀区翠微东里甲2号 邮编：100036
开 本：787×1092 1/16 印张：26.75 字数：680千字
印 次：2008年1月第1次印刷
定 价：40.00元

凡所购买电子工业出版社图书有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系，联系及邮购电话：（010）88254888。
质量投诉请发邮件至zlts@phei.com.cn，盗版侵权举报请发邮件至dbqq@phei.com.cn。
服务热线：（010）88258888。

前　言

随着Web技术的发展，在Java/JSP Web领域中出现了大量的开发框架，开发框架的出现大大提高了跨平台语言Java的开发效率。

Jakarta Struts框架是Apache软件组织提供的一个开放源代码的项目，它为Java Web应用提供了MVC模式的具体实现，尤其适用于开发大型企业级应用程序。Struts为Web应用提供了一个通用的框架，使得开发人员可以把主要精力集中在实际业务逻辑的开发上，并且非常适合于团队开发，符合现代软件开发的各项要求。

随着Struts的广泛应用，市面上关于Struts的图书也层出不穷，但是普遍存在两方面的问题：一个是过于偏重理论知识的介绍，使读者阅读之后还是不能够进行实际开发；另一个是大量地堆砌代码，没有进行深入的分析，使读者囫囵吞枣，毫无收获。

本书在总结这两方面问题的基础上，提供了对Struts框架中几乎所有知识点的介绍，不仅有理论知识的介绍，还把我们多年积累的实际商业开发中的技巧和经验融入其中，使读者能够掌握实际项目中所需要的基本知识。

本书结构

本书内容可以归纳为4大部分，共12个章节。本书章节安排如下：

- 基础部分：包括第1章至第6章的内容，介绍Struts框架的基础知识。
- 高级部分：包括第7章和第8章的内容，介绍Struts框架中的高级应用技术。
- 框架部分：包括第9章和第10章的内容，介绍Struts框架与Hibernate框架以及Spring框架的结合使用，详细讲解目前常用的SSH开发模式。
- 开发部分：包括第11章和第12章的内容，通过两个具体的商业化实例分析Struts框架以及Hibernate在实际开发中的应用技巧。

本书特色

- 详尽的实例

本书从第2章至第12章附有大量的实例，通过这些实例介绍知识点。每个实例都是作者精心选择的，并且已经测试成功。这些实例可以帮助读者更容易地理解知识点。

- 附有完整的应用系统

本书第11章和第12章分别介绍了两个完整的应用系统。这两个应用系统通过数个知识点分别介绍，这也是对本书其他章节所介绍知识点的应用。

• 代码注重可读性和实用性

本书代码都经过作者测试，保证代码的正确性。代码都加了详细的注释，简单易懂。本书没有把介绍重点放在语法，而是重点介绍实例或者系统的实现方法。

本书的读者对象

本书是一本Struts开发实践的技术书籍，不仅介绍了Struts框架最基本的知识，还介绍了Validator、Tiles以及SSH方面的知识。本书适合以下读者：

• 本书内容由浅入深，Struts初学者可以轻松地学习本书内容，对于Struts初学者是非常合适的。

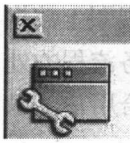
• 本书介绍了Struts开发中的常用方法和技巧，对于从事Java Web应用开发的读者是合适的。

本书以实例介绍知识点，介绍重点放在实例或系统的实现方法上。这避免了枯燥的理论介绍，对于通过实践学习知识的读者有很大的帮助。阅读完本书后，即使是初学者，也可以创建一个基于Struts框架的Web应用系统。

以下人员对本书的编写提出过宝贵意见并参与了本书的部分资料的搜集工作，他们是王寿苹、孙宁、王荣芳、李德路、李岩、周科峰、陈勇、高云、韩亚男，同时感谢电子工业出版社及美迪亚电子信息有限公司的各位老师，谢谢你们的帮助和指导。

尽管我们尽了最大努力，但由于时间仓促，加之水平有限，本书难免有疏漏之处，欢迎各界专家和读者朋友批评指正。

目　录



第1章

Struts基础知识

课前导读

随着MVC（Model-View-Controller）设计模式在Web开发中的广泛应用，出现了很多为建立基于MVC模式的Java Web应用程序而提供的开源框架，Struts就是其中应用最广泛的一种，Struts是MVC的一种实现，它很好地结合了JSP、Servlet、JavaBean、Taglib等技术。在本章中，我们将重点给读者介绍基于Struts框架的MVC设计模式的实现机制和原理，并在此基础上简单介绍Struts框架的结构体系和工作原理。

重点提示

在本章我们将重点学习：

- MVC设计模式
- Struts开发框架
- Struts工作原理

1.1 MVC设计模式

MVC是一种目前广泛流行的软件设计模式，早在20世纪70年代，IBM就推出了Sanfronscisico项目计划，其实就是MVC设计模式的研究。随着Web应用程序的广泛应用，应用于Web应用程序快速开发的MVC框架不断出现，成为新的开发热点。

MVC的英文全称为Model-View-Controller，即把一个应用程序的输入层、业务处理层、控制流程按照View、Model、Controller的方式进行分离，这样一个应用程序就被划分成相对独立而又协同工作的3个层，即视图层、模型层、控制层。

在MVC设计模式中，它的模型、视图、控制器分别担负不同的任务，图1-1显示了这3个模块各自的功能以及它们的相互关系。

在MVC设计模式中，客户端Web浏览器会提交各种不同的用户请求，这些请求由控制器进行处理，控制器根据事件的类型来改变模型或视图，视图也可以接受模型发出的数据更新通知，依据数据更新的结果调整视图效果，并呈现在用户面前。而模型也可以通过视图所获得的用户提交的数据进行具体业务逻辑的处理。

1.1.1 视图（View）

视图是用户看到并与之交互的界面，对于Web应用来说，可以概括为HTML界面，但有可能为XHTML、XML和Applet。随着应用的复杂性和规模性的增加，界面的处理也变得具有

挑战性。一个应用可能有很多不同的视图，MVC设计模式对于视图的处理仅限于视图上数据的采集和处理，以及用户的请求，而不包括在视图上进行的业务流程的处理。业务流程的处理交予模型（Model）处理。比如一个订单的视图只接受来自模型的数据并显示给用户，以及将用户界面的输入数据和请求传递给控制器和模型。

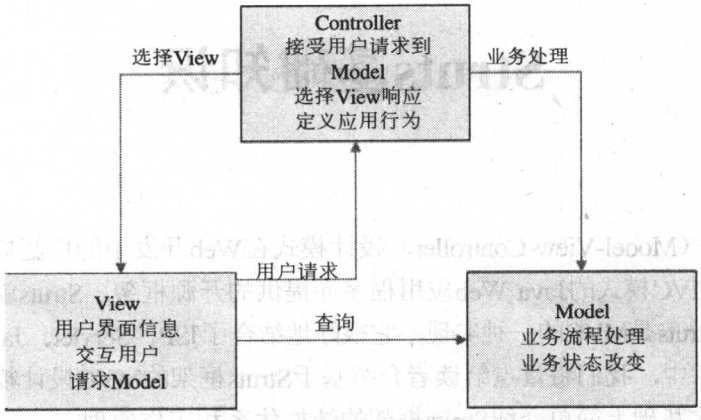


图1-1 MVC设计模式

1.1.2 模型（Model）

模型是业务流程、状态的处理以及业务规则的制定。业务流程的处理过程对其他层来说是黑箱操作，模型接受视图请求的数据，并返回最终的处理结果。业务模型的设计可以说是MVC最主要的核心。目前流行的EJB模型就是一个典型的应用例子，它从应用技术实现的角度对模型做了进一步的划分，以便充分利用现有的组件，但它不能作为应用设计模型的框架。它仅仅告诉你按这种模型设计就可以利用某些技术组件，从而减少了技术上的困难。对一个开发者来说，就可以专注于业务模型的设计。MVC设计模式告诉我们，把应用的模型按一定的规则抽取出来，抽取的层次很重要，这也是判断开发人员是否优秀的依据。抽象与具体不能隔得太远，也不能太近。MVC并没有提供模型的设计方法，而只告诉你应该组织管理这些模型，以便于模型的重构和提高重用性。

除业务模型外还有一个很重要的模型那就是数据模型。数据模型主要指实体对象的数据存储（持久化）。比如将一张订单保存到数据库，从数据库获取订单。我们可以将这个模型单独列出，所有有关数据库的操作都限制在该模型中。

1.1.3 控制器（Controller）

控制器可以理解为从用户接收请求，将模型与视图匹配在一起，共同完成用户的请求。控制器并不做任何的数据处理。例如，用户单击一个连接，控制器接受请求后，并不处理业务信息，它只是把用户的信息传递给模型，告诉模型做什么，并选择符合要求的视图返回给用户。因此，一个模型可能对应多个视图，一个视图可能对应多个模型。

综上所述，MVC模式的出现不仅实现了功能模块和显示模块的分离，同时它还提高了应用系统的可维护性、可扩展性、可移植性和组件的可复用性。

1.1.4 MVC的适用范围

大部分使用过程语言（例如ASP、PHP）开发出来的Web应用，初始的开发模板基本都是混合层的数据编程。例如，直接向数据库发送请求并用HTML显示，开发速度往往比较快，但由于数据页面的分离不是很直接，因而很难体现出业务模型的样子或者模型的重用性。产品设计弹性力度很小，很难满足用户的变化性需求。

MVC要求对应用分层，虽然要花费额外的工作，但产品的结构清晰，产品的应用通过模型可以得到更好的体现。

首先，最重要的是应该有多个视图对应一个模型的能力。在目前用户需求迅速变化的情况下，可能会有使用多种方式访问应用的要求。例如，订单模型可能有本系统的订单，也有网上订单，或者其他系统的订单，但对于订单的处理都是一样的，也就是说订单的处理是一致的。按MVC设计模式，一个订单模型以及多个视图即可解决问题。这样减少了代码的复制，即减少了代码的维护量，一旦模型发生改变，也易于维护。

其次，由于一个应用被分离为3层，因此有时改变其中的一层就能满足应用的改变。一个应用的业务流程或者业务规则的改变只需改动MVC的模型层。

最后，分层还有利于软件工程化管理。由于不同的层各司其职，每一层不同的应用具有某些相同的特征，有利于通过工程化、工具化产生管理程序代码。

MVC的设计实现并不十分容易，虽然理解起来比较容易，但对开发人员的要求比较高。而且MVC只是一种基本的设计思想，还需要详细的设计规划。

经验表明，MVC由于将应用分为3层，意味着代码文件增多，因此，对于文件的管理需要费点心思。

综上所述，MVC需要精心地计划，使得你会认真考虑应用的额外复杂性，把这些想法融进到架构中，增加应用的可拓展性。如果能把握这一点，MVC模式会使得你的应用更加强壮、更加有弹性、更加个性化。但在构建MVC框架时会花费一定的工作量，所以MVC不适合小型应用程序的开发。

1.1.5 JSP Model 1和JSP Model 2

尽管MVC设计模式很早就提出了，但在Web项目的开发中引入MVC却是步履维艰。主要原因：一是在早期的Web项目的开发中，程序语言和HTML的分离一直难以实现。CGI程序以字符串输出的形式动态地生成HTML内容。后来随着脚本语言的出现，前面的方式又被倒了过来，改成将脚本语言书写的程序嵌入在HTML内容中。这两种方式有一个相同的不足之处，即它们总是无法将程序语言和HTML分离。二是脚本语言的功能相对较弱，缺乏支持MVC设计模式的一些必要的技术基础。

直到基于J2EE的JSP Model 2问世该问题才得以改观。它用JSP技术实现视图的功能，用Servlet技术实现控制器的功能，用JavaBean技术实现模型的功能。

JSP规范中提出了两种用JSP技术建立应用程序的方式。这两种方式在术语中分别称做JSP Model 1和JSP Model 2，它们的本质区别在于处理批量请求的位置不同。在Model 1体系结构中，如图1-2所示，JSP页面独自响应请求并将处理结果返回客户端。这里仍然存在表达与内容的分离，因为所有的数据存取都是通过JavaBean来完成的。尽管Model 1体系结构十分

适合简单应用的需要，但它却不能满足复杂的大型应用程序的实现。不加选择地随意运用Model 1，会导致JSP页面内被嵌入大量的脚本片段或Java代码，特别是当需要处理的请求量很大时，情况更为严重。尽管这对于Java程序员来说可能不是什么大问题，但如果JSP页面是由网页设计人员开发并维护的，而这通常是开发大型项目的规范，就确实是个问题了。从根本上讲，将导致角色定义不清和职责分配不明，给项目管理带来不必要的麻烦。

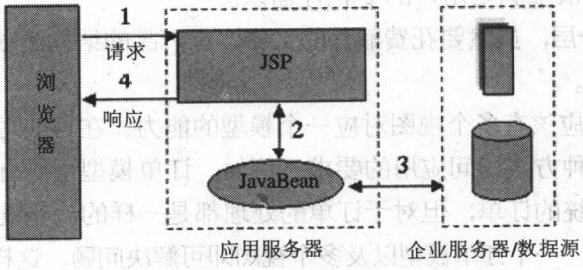


图1-2 JSP Model 1体系结构

Model 2体系结构如图1-3所示，是一种把JSP与Servlet联合使用从而实现动态内容服务的方法。它吸取了两种技术各自的突出优点，用JSP生成表达层的内容，让Servlet完成深层次的处理任务。在这里，Servlet充当控制者的角色，负责管理对请求的处理，创建JSP页面需要使用的JavaBean和对象，同时根据用户的动作决定把哪个JSP页面传给请求者。特别要注意，在JSP页面内没有逻辑代码，它仅负责检索原来由Servlet创建的对象或JavaBean，从Servlet中提取动态内容插入静态模板。在我们看来，这是一种有代表性的方法，它清晰地分离了表达和内容，明确了角色的定义以及开发者与网页设计者的分工。事实上，项目越复杂，使用Model 2体系结构的好处就越大。

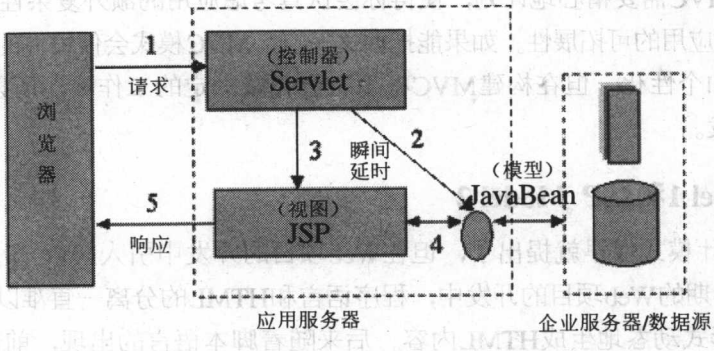


图1-3 JSP Model 2体系结构

某些开发者觉得JSP Model 2仍不能满足需要，于是Craig R. McClanahan于2000年5月提交了一个Web Framework给Java Community，这就是后来的Struts框架的雏形。

2001年7月，Struts 1.0正式发布。该项目也成为了Apache Jakarta的子项目之一。

Struts本质上就是在JSP Model 2的基础上实现的一个MVC架构。它只有一个中心控制器，并采用XML定制转向的URL，采用Action来处理逻辑。

1.2 Struts简介

Struts是一个基于Sun J2EE平台的MVC框架，最早是作为Apache Jakarta项目的组成部分问世运做的，主要采用Servlet和JSP技术来实现。由于Struts能充分满足应用开发的需求，而且简单易用，在过去的几年中颇受关注。Struts把Servlet、JSP、自定义标签和信息资源（Message Resources）整合到一个统一的框架中，开发人员利用其进行开发时不用再自己编码实现全套MVC模式，极大地节省了时间，所以说Struts是一个非常不错的应用框架。

Struts这个名字来源于在建筑和旧式飞机中使用的支撑金属架，可以到http://jakarta.apache.org/Struts官方网站免费下载该软件。

1.2.1 Struts体系结构

对于已经习惯了使用传统的JSP设计模式实现Web应用程序的开发者来说，在开始接触Struts时经常会出现一种疑惑：几乎所有的Struts专业书籍和文档中总是使用Struts框架（Framework），那么到底什么是框架？

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

框架的体系提供了一套明确机制，从而让开发人员很容易地扩展和控制整个框架开发的结构。框架基本上都包含如下几个逻辑模块：

控制器（Controller）：控制整个框架中各个组件的协调工作。

业务逻辑层（Business Logic）：对框架本身来说，这里仅仅只是概念和几个提供服务的基础组件，要真正地实现与客户的业务逻辑接轨，还需要开发人员在框架上再次扩展。

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

Struts框架有其自己的控制器（Controller），同时整合了其他的一些技术来实现模型（Model）和视图（View），那么我们就分别从模型、视图、控制器来看看Struts的体系结构。

• 视图

视图就是一组JSP文件。在这些JSP文件中没有业务逻辑，也没有模型信息，只有标签，这些标签可以是标准的JSP标签或客户化标签。

此外，通常把Struts框架中的ActionForm Bean也划分到视图模块中。ActionForm Bean也是一种JavaBean，在运行时，该Bean用于验证HTML表单数据以及将其属性重新设置为默认值。Struts框架利用ActionForm Bean来进行视图和控制器之间表单数据的传递。

• 模型

模型表示应用程序的状态和业务逻辑，Struts框架没有提供特定的模型组件，在大型分布式应用开发中，业务逻辑通常由JavaBean或EJB组件来实现。

• 控制器

控制器是Struts框架的中枢，由ActionServlet类和Action类来实现。其中ActionServlet在MVC模型中扮演中央控制器的角色，接收所有客户端的请求，并把请求委派到指定的Action类。Action类负责调用模型的方法，更新模型的状态，并帮助控制应用程序的流程。一旦Action类完成处理，ActionServlet将根据Action返回的键值来决定调用什么视图来显示Action类的处理结果。此外，除了Action以外，对于业务逻辑的操作还需要由ActionMapping、ActionForward这几个组件协助完成，它们用来指定不同业务逻辑或流程的运行方向。

• 配置文件Struts-config.xml

上面提到一个用户请求是通过ActionServlet来处理和转发的。那么，它如何决定把用户请求转发给哪个Action对象呢？这就需要描述用户请求路径和Action映射关系的配置信息。这些配置信息被存储在特定的XML文件Struts-config.xml中。这些信息在系统启动的时候被读入内存，供Struts在运行期间使用。

1.2.2 Struts工作原理

图1-4显示了Struts框架的体系结构响应客户请求时，各个组成部分的工作原理。

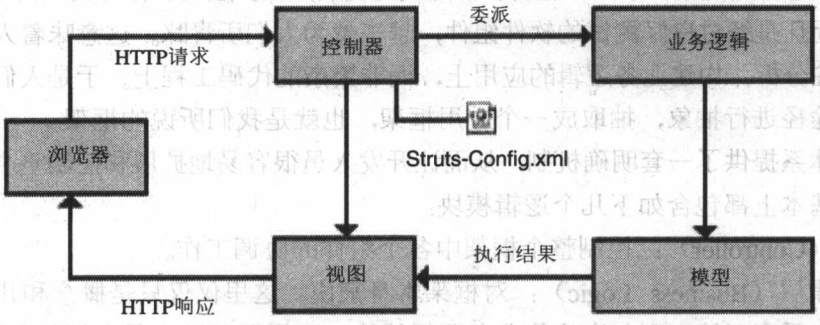


图1-4 Struts工作原理

对于基于Struts框架的Web应用程序，当客户端浏览器发出请求时，请求会被控制器截获，并调用在Web应用启动时就加载并初始化的ActionServlet，然后通过ActionServlet从Struts-config.xml文件中读取预先设置的配置信息，并且把它们存放到各个配置对象中，例如Action的映射信息存放在ActionMapping对象中。

控制器根据配置文件中的信息，或者选择合适的Action对象，或者直接选择合适的View对象返回给浏览器。Action对象本身没有任何逻辑功能，它只是控制器和模型之间的接口，控制器通过Action调用匹配的模型对象，模型对象根据执行结果，选择合适的View对象返回给浏览器。

通过上述分析，我们可以看到在Struts框架的整个工作过程中，控制器是控制整个程序执行流程的调度者，在Struts中基本的控制器组件是ActionServlet类，但是ActionServlet类不包含任何控制信息，程序的所有调度信息都需要在Struts-config.xml配置文件中设置。

1.2.3 Struts核心类

Struts框架中需要几个内部核心类来完成其基本的工作流程，如表1-1所示。

表1-1 Struts内部核心类

核心类名称	功能
ActionServlet	接收并转发所有请求
Action	包含和调用事务逻辑
ActionForm	显示模块数据
ActionMapping	帮助控制器将请求映射到操作
ActionForward	用来指示操作转移的对象
ActionError(s)	用来存储和回收错误

• ActionServlet类

ActionServlet类继承自javax.servlet.http.HttpServlet类，其在Struts框架中扮演的角色是中心控制器。它提供一个中心位置来处理全部的客户请求，然后再由ActionServlet把请求转发给其他组件。ActionServlet主要负责将HTTP的客户请求信息组装后，根据配置文件的指定描述，转发到适当的处理器。此外，Struts框架中只允许一个应用中配置一个ActionServlet，但是可以将ActionServlet扩展。

ActionServlet类对应的类名为org.apche.struts.action.ActionServlet，它本质上和一个普通的Servlet没有区别，你完全可以把它当做一个Servlet来看待，只是在其中完成的功能不同罢了。ActionServlet主要完成如下功能：

1. 从配置文件web.xml中读取初始化参数。
2. 将一个来自客户端的URI映射到一个相应的Action类或直接转向一个输出页面。

• Action类

ActionServlet把客户端提交的全部请求都委托到RequestProcessor对象。RequestProcessor使用Struts-config.xml文件检查请求URI，找出所对应的Action标识符。

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

• ActionMapping类

上文讲解过一个客户请求是如何被控制器转发和处理的，但是，控制器如何知道什么样的信息转发到什么样的Action类呢？这就需要一些与动作和请求信息相对应的映射配置说明。

在Struts框架中，这些映射配置信息是被存储在特定的配置文件Struts-config.xml中的，这些配置信息在系统启动的时候被读入内存，供Struts框架在运行期间使用。在内存中，每一个位于Struts-config.xml文件中的<action>元素都与ActionMapping类的一个实例对应。

• ActionForward类

ActionForward对象是配置对象。这些配置对象拥有唯一的标识以允许它们按照有意义的名称如“success”，“failure”等来检索。ActionForward对象封装了向前进的URL路径且被请求处理器用于识别目标视图。ActionForward对象建立自位于Struts-config.xml文件中的<forward>元素。

• ActionForm类

在上面讲解ActionServlet、Action和ActionMapping的时候，我们都提到了ActionForm。一个应用系统中消息转移的非持久性数据存储，是由ActionForm负责完成的。ActionServlet使用ActionForm来保存请求的参数，ActionForm中的属性名称与HTTP请求参数中的名称相对应，控制器将请求参数传递到ActionForm的实例，然后将这个实例传送到Action类。

• ActionError（s）类

Struts框架提供了两个类来进行错误处理：ActionErrors和ActionError，它们都扩展org.apache.struts.action。

ActionErrors类用来保存ActionError对象的集合，其中每一个ActionError对象代表一个独立的错误信息。每个ActionError都包含关键字，能够映射到资源文件中存储的错误信息，而这个资源文件是在ActionServlet初始化参数中指定的。

ActionError类定义了一组重载的构造器来创建错误信息，第一个构造器方法使用一个字符串作为参数，另一个使用java.text.MessageFormat类，可在消息中指定替换字符串。

ActionError类从不独立进行错误处理，它们总是被存储在ActionErrors对象中。ActionErrors对象保存ActionError类的集合以及它们特定的属性值，我们可以使用自己定义的属性值，或是使用ActionErrors.GLOBAL_ERROR。

1.2.4 Struts执行流程

在基于Struts框架的Web应用中，当Web应用启动时会加载并初始化ActionServlet。ActionServlet从struts-config.xml配置文件中读取配置信息，并把它们存放到各种配置对象中。这个初始化过程是在服务器启动时自动完成的，在初始化完成之后，当ActionServlet接收到一个客户请求时，将执行如下流程：

1. 检索和用户请求匹配的ActionMapping实例，如果存在，表示找到ActionMapping所对应的ActionForm对象，如果不存在，则返回用户请求路径无效。

2. 如果ActionForm实例不存在，就创建一个ActionForm对象，把客户提交表单数据保存到该ActionForm对象中。

3. 根据配置信息决定是否需要表单验证，如果需要验证，就调用ActionForm中的validate( )方法。

4. 如果ActionForm的validate( )方法返回null或返回一个不包含ActionMessage的ActionError对象，表示表单验证成功。

5. ActionServlet根据ActionMapping实例包含的映射信息决定将请求转发给哪个Action。如果相应的Action实例不存在，就先创建这个实例，然后调用Action的execute( )方法。

6. Action的execute( )方法返回一个ActionForward对象，ActionServlet再把客户请求转发给ActionForward对象所指向的JSP组件。

7. ActionForward对象指向的JSP组件生成动态网页，返回给客户端浏览器。

在流程4中，如果ActionForm的validate( )方法返回一个包含一个或多个ActionMessage的ActionErrors对象，则表示表单验证失败，此时ActionServlet将直接把请求转发给包含客户提交表单的JSP组件。在这种情况下，不会再创建Action对象和调用Action的execute( )方法。

1.3 各种MVC框架比较

基于Web的MVC框架在J2EE的世界内已是空前繁荣，几乎每隔一两个星期就会有新的MVC框架发布。目前比较好的MVC，老牌的有Struts、WebWork，新兴的有Spring、Tapestry、JSF等。这些大多是著名团队的作品另外还有一些边缘团队的作品也相当出色，如Dinamica、VRaptor等。这些框架都提供了较好的层次分隔能力，在实现良好的MVC分隔的基础上，通过提供一些现成的辅助类库，可以促进生产效率的提高。

1.3.1 Spring框架

Spring是一个开源框架，是为了解决企业应用程序开发复杂性而创建的。框架的主要优势之一就是其分层架构，分层架构允许用户选择使用哪一个组件，同时为J2EE应用程序开发提供集成的框架。

Spring实际上是“Expert One-on-One J2EE Design and Development”（中文译名为《J2EE设计开发编程指南》）一书中所阐述的设计思想的具体实现。在该书中，作者Rod Johnson倡导J2EE实用主义的设计思想，并随书提供了一个初步的开发框架实现（interface21开发包）。而Spring正是这一思想的更全面和更具体的体现。Rod Johnson在interface21开发包的基础之上，进行了进一步的改造和扩充，使其发展为一个更加开放、清晰、全面、高效的开发框架。

Spring使得使用基本的JavaBean来完成以前只能由EJB完成的事情变得可能。而且，Spring的用途不仅限于服务器端的开发。从简单性、可测试性和松耦合的角度而言，任何Java应用都可以从Spring中受益。

Spring框架是一个分层架构，由7个定义良好的模块组成。各模块构建在核心容器之上，核心容器定义了创建、配置和管理JavaBean的方式，如图1-5所示。

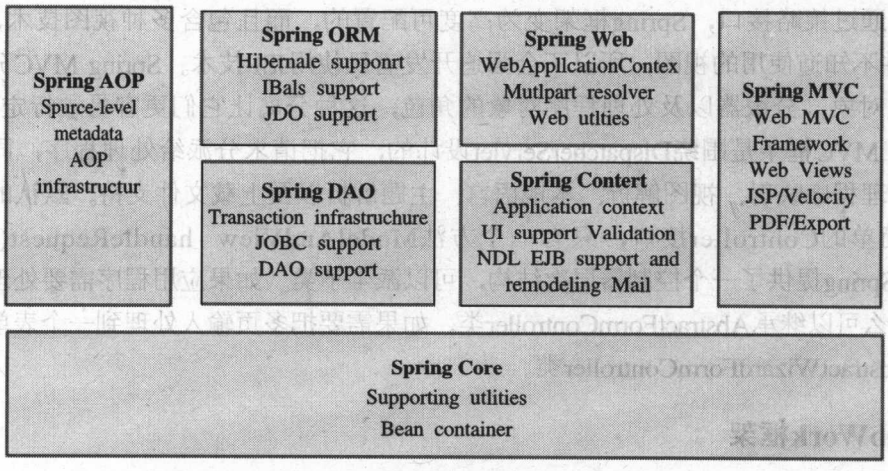


图1-5 Spring核心模块

组成Spring框架的每个模块（或组件）都可以单独存在，或者与其他一个或多个模块联合实现。每个模块的功能如下：