CD-ROM

轻松掌握 Struts2

■ 郝玉龙　迟健男　编著

清华大学出版社　●　北京交通大学出版社

轻松掌握 Struts2

郝玉龙　迟健男　编著

清华大学出版社
北京交通大学出版社
·北京·

内 容 简 介

本书对企业级 Java EE 开发框架 Struts2 进行了系统讲解。Struts2 是一个设计精巧的框架，在企业开发领域已经得到广泛应用。为使读者更方便理解框架，本书先通过一个简单的示例对框架进行介绍，使读者有一个整体的感性认识，然后按照自下而上的方式分别对 Action 组件、拦截器、标记库、结果视图类型、类型转换、输入校验、异常处理、国际化、Ajax 支持和与 Spring、Hibernate 集成等 10 个专题对框架进行深入介绍，使读者对框架的各个功能特性都有深入的理解。在各个专题的讲解中，对框架底层的实现机制进行了深入剖析，加深读者对框架的理解，同时对实际应用中的开发技巧和方法通过具体示例进行详尽演练，务必达到使读者不仅能够在实际开发中灵活运用 Struts2 框架，而且对 Struts2 框架的设计思想和设计模式有透彻领会，从而切实提高自身能力水平。

本书适用于对 Java EE 编程有一定了解，希望尽快掌握 Struts2 编程技术的开发人员，也适合希望提高 Java EE 应用系统架构设计水平的中高级开发人员参考。本书也可作为 Struts2 编程技术的培训教材。

图书在版编目（CIP）数据

轻松掌握 Struts2 / 郝玉龙，迟健男编著. —北京：清华大学出版社；北京交通大学出版社，2010.7
ISBN 978-7-5121-0134-0

Ⅰ. ①轻…　Ⅱ. ①郝…　②迟…　Ⅲ. ①软件工具-程序设计　Ⅳ. ①TP311.56

中国版本图书馆 CIP 数据核字（2010）第 104548 号

责任编辑：谭文芳
出版发行：清 华 大 学 出 版 社　邮编：100084　电话：010-62776969　http://www.tup.com.cn
　　北京交通大学出版社　邮编：100044　电话：010-51686414　http://press.bjtu.edu.cn
印 刷 者：北京鑫海金澳胶印有限公司
经　　销：全国新华书店
开　　本：203×280　印张：16　字数：457 千字　附光盘 1 张
版　　次：2010 年 7 月第 1 版　2010 年 7 月第 1 次印刷
书　　号：ISBN 978-7-5121-0134-0/TP・594
印　　数：1～4 000 册　定价：35.00 元（含光盘）

前　言

随着网络的日益普及以及社会信息化程度的提高，越来越多的软件开发人员需要开发 Web 应用程序。Struts2 作为目前最受欢迎的企业级 Java EE Web 应用程序开发框架越来越受到欢迎，并且取得了许多成功的案例，已经成为 Java EE 开发人员必须掌握的技能之一。

本书起源

在写作本书之前笔者也曾犹豫再三，原因有三。一是 Java EE 编程技术日新月异，Struts2 框架的发展更是如此，就在写作本书的过程中，Struts2 框架的版本就经历了不小的变化。针对一门迅速发展的技术写作就好比对疾驰的火车拍照，难以反映出 Strut2 框架的现状和全貌。二是 Struts2 是一个设计精巧的开发框架，它的各个部分之间是紧密联系的，因此如何对 Struts2 框架展开描述，以便读者能够循序渐进地学习 Struts2 框架一直是困扰笔者的一个问题，提笔时总有一种“老虎吃天，无从下口”的感觉。三是感觉自身水平有限，难免存在疏漏，贻误读者，心中总是惶惶然。

但回顾自己学习 Struts2 框架过程中走过的历程，经历了被种种错误资料误导的痛苦，被 Google 搜索出的浩如烟海的数据淹没的窒息，对某个具体问题经过尝试实验解开后的喜悦，所以也想把这些感受拿出与你分享。

本书写作的目的在于，通过对 Struts2 框架较为系统地介绍，使读者能够全面认识 Struts2 框架，不但掌握运用 Struts2 框架的技巧，更能领会 Struts2 框架设计思想的精妙，从而提高 Java EE 应用的架构设计水平。

本书不是一本手册书或操作指南，它不会面面俱到讲述 Struts2 框架的各个知识点，而是始终以较高的视点带领读者俯览整个框架，从繁复纷杂中描出框架的主线，理解框架的设计，品味框架的精妙。

本书主要内容

本书各章节的内容安排如下。

第 1 章，了解 Struts2 框架的来源、Struts2 框架的设计思想及框架在 Web 应用中的地位，对框架有个全面清晰的认识。

第 2 章，创建一个简单的 Struts2 Web 应用，了解 Struts2 Web 应用的基本组成和配置方法，对 Struts2 获得感性的认识，并为后续各章深入学习 Struts2 框架打下基础。

第 3 章，深入讲解框架的 Action 组件。Action 组件是对客户端请求进行处理的组件，也是开发人员花费精力最多的地方。在这一章中，将学习 Action 组件的定义、配置，Action 组件如何响应外部请求、如何访问外部资源等。

第 4 章，深入讲解拦截器。拦截器是框架中设计最精妙的部分，也是最能体现 Struts2 框架设计思想的组件。Struts2 框架的许多核心功能，都是通过拦截器来实现的。在一定程度上，理解了拦截器，也就理解了 Struts2 框架。这一章将学习拦截器的定义、工作原理，Struts2 框架中的拦截器实现，以及如何自定义开发拦截器组件。

第 5 章，学习框架的标记库。标记库是为了帮助开发人员实现用户界面开发而提供的一个工具。这一章将系统介绍 Struts2 框架的数据存储机制和访问语言。

第 6 章，学习框架的视图组件。视图是用来展示信息的组件。与 Struts1 不同的是，Struts2 框架的视图将不再限定为 JSP，而是可以包含 freemarker、velocity、Stream 等多种视图类型，它使得 Web 应用的表现形式更加丰富。这一章将深入学习 Struts2 框架的内置视图类型，以及如何利用它们实现多样 Web 应用。

第 7 章，学习框架的类型转换机制。类型转换功能是 Web 应用开发必须要解决的问题，这是由于 HTTP 协议仅能传输文本文件的局限所造成的。这一章将深入学习 Struts2 框架的类型转换的实现原理，内置的类型转换器以及在 Struts2 框架下的类型转换编程方法。

第 8 章，学习框架的输入校验机制。与类型转换功能一样，输入校验也是 Web 应用开发必须要解决的问题。Struts2 框架提供了一种声明式的输入校验机制。这一章将深入学习 Struts2 框架的输入校验机制的实现原理，内置校验器、Struts2 框架下的输入校验规则配置以及如何开发自定义校验器。

第 9 章，学习 Struts2 框架的声明式异常处理机制。

第 10 章，学习 Struts2 框架的国际化支持特性，掌握如何在 Struts2 框架下开发国际化应用。

第 11 章，学习 Struts2 框架对 Ajax 的支持，以及如何开发 Struts2 框架下的 Ajax 应用。

第 12 章，学习 Struts2 框架如何与 Spring、Hibernate 集成，以开发复杂的 Web 应用。

本书特色

与市场上的同类图书相比，本书具有以下特色。

（1）思路清晰，结构严谨。书中内容遵循由浅入深、循序渐进的原则，先对 Struts2 框架进行一个总体介绍，然后按照不同专题对框架的各个方面系统讲解。在各个专题的讲解过程中，注意前后呼应，相互印证，逐步加深对框架的理解。

（2）语言通俗，讲解透彻。在对 Struts2 框架的介绍过程中，采用简洁易懂的语言进行讲解，短小精悍的示例进行演练，使学习过程变得轻松愉快。

（3）理论实践并重。不但注重实际操作的演示，更加注重对框架设计精髓的讲解，使读者不但能够在日后的工作中能够熟练运用框架，更能够提高自身的架构设计水平，获得全面的提升进步。

适用读者

本书适用于对 Java EE 编程有一定了解，希望尽快掌握 Struts2 编程技术的开发人员，也适合希望提高 Java EE 应用系统架构设计水平的中高级开发人员参考。本书还可作为 Struts2 编程技术的培训教材。

使用与反馈

为方便广大读者使用本书学习 Struts2 编程，本书附带的光盘中包含了所有例程的源代码，以及关于如何使用这些源代码的详细指导和说明。

由于作者水平有限，加之编写时间仓促，书中难免出现错误和不足。对于书中的任何问题，请发 E-mail 至：haoyulongsd@163.com。

致谢

在本书的编写过程中，得到众多同事的指导和帮助。感谢陈启槐教授对本书提出的建设性意见！感

谢李博、李冰、孙阳、丰硕，他们参与了本书的部分修改工作，并对本书的内容组织提供了建设性的意见。感谢本书的编辑，北京交通大学出版社的谭文芳老师，给予我充足的写作时间，使我能够从容完成本书的编写。还要特别感谢我的妻子季平，在我多次想要放弃时，正是她的鼓励，给了我继续写下去的动力，使我能够克服各种困难，确保本书能够顺利完成。

郝玉龙

2010 年 4 月

目　录

第 1 章　认识一下 Struts2

引言

在深入学习 Struts2 之前，我们首先回答什么是 Struts2？ Struts2 是一个用来开发企业级动态 Java EE Web 应用程序的框架。但你可能马上会产生疑问：什么是企业级动态 Java EE Web 应用呢？什么是框架？Struts2 又是一个怎样的框架呢？别着急，听我慢慢道来。

1.1　什么是 Web 应用

Struts2 是一个用来开发企业级动态 Java EE Web 应用程序的框架。因此首先要弄清什么是 Web 应用。

顾名思义，Web 应用是运行在 Web 上的应用程序。但是反过来，运行在 Web 上的应用程序都是 Web 应用吗？答案是否定的。这里所说的 Web 应用是指运行在网络上，以浏览器作为通用客户端的应用程序，在许多地方又被称为 B/S（Browser/Server，浏览器-服务器）模式的应用。当使用 IE 或者 FireFox 在网易、新浪等门户网站上冲浪时，使用的就是 Web 应用。

除此之外，还存在一种所谓的 C/S（Client/Server，客户-服务器）模式的应用。例如，常用的聊天工具 QQ 或 MSN、下载工具迅雷等就是典型的 C/S 模式的应用。

区别 Web 应用与 C/S 模式的应用很简单，从使用角度上，Web 应用使用标准的浏览器作为客户端，由于常见的浏览器，如 IE 等已经成为操作系统的一部分，因此 Web 应用在使用时无需专门安装，而 C/S 模式的应用需要专门安装客户端软件。你一定还记得安装 QQ 软件的经历吧？从技术的角度看，Web 应用是基于标准的应用层协议 HTTP 的，而 C/S 模式的应用是基于网络层协议 TCP/UDP 的。回顾一下网络课上学习的知识，应用层协议是定义在网络层协议之上的，正是由于 Web 应用采用的是一种更高层次的标准化通信模式，使 Web 应用能够获得更快的普及和推广。

现在我们对什么是 Web 应用有了了解，那么什么是动态 Web 应用呢？这就需要回过头去探究一下 Web 的起源及其发展历程。

1989 年，为了使欧洲各国的核物理学家能通过计算机网络及时沟通传递信息进行合作研究，英国人蒂姆·伯纳斯-李发明了 Web，它通过 HTTP 协议，把网络上不同计算机内的信息有机地结合在一起。Web 使得分布在全世界各地物理实验室、研究所的最新信息——数据、图像资料——可供大家共享。1991 年，Web 被推广到互联网上，并迅速得到推广，蒂姆·伯纳斯-李也因此获得了 Web 之父的美誉。

从 20 世纪 90 年代到现在近二十年的时间，在各种商业利益需求的驱动和各国软件开发人员的共同努力下，互联网经历了一个飞速的发展时期，Web 应用的发展也经历了一个由静态到动态的不断发展演变的过程。

最初的 Web 应用都是静态应用。所有 Web 页面都是内容固定的静态页面，Web 应用其实就是个静态网页的集合。用户请求一个静态页面资源，服务器将返回这个资源。举一个形象的比喻，使用静态 Web 应用就像是读书，书中每一页的内容是固定的，用户只能选择去读哪一页，而不允许修改其中的内容。静态 Web 应用比纸质图书优越的地方在于允许读者在任何网络联通的位置都可以访问，而且信息的检索

更加方便而已。

随着 Web 应用的普及推广，静态 Web 应用渐渐无法满足人们的需要。首先，用户与 Web 应用的交互方式太局限，Web 应用只能接收用户的资源请求，返回静态的网页。用户无法向服务器提交除服务器资源请求以外的信息。另外，随着 Web 应用规模的扩大，海量的静态网页信息也难以维护。举一个简单的例子，如果中国移动以静态网页的方式为用户提供每月的通话记录，那么中国移动目前有 4 亿多用户，则需要 4 亿多个静态网页来完成这一艰巨的任务。试想这每月数以亿计的静态网页需要多少软件工程师来开发维护，服务器的存储空间又如何来承载？

于是就产生了动态 Web 应用。区别于静态 Web 应用，所谓的动态 Web 应用有以下特点： 用户向服务器提交的信息不再是单一的静态资源请求，还包括一系列特定于用户的请求参数信息；服务器向用户客户端返回的不再是事先已经存在服务器上的静态资源，而是根据特定用户的参数信息动态生成的响应信息。

从上面的叙述可以看出，在动态 Web 应用中，客户端与服务器的交互不再“单纯”，变得越来越复杂起来。这也就使得 Web 应用的开发变得越来越困难。

1.2 开发 Web 应用为什么需要框架

我们现在清楚了什么是 Web 应用，那么在 Web 应用开发中为什么需要框架呢？这主要由以下三个因素导致的：企业应用开发要求的不断提升、 Web 应用模型的先天不足及 Java EE 身后的巨大空白。

1.2.1 企业应用开发的特点

需要反复强调的是，本书探讨研究的 Struts2 是一个开发企业级 Web 应用的框架。这里所说的企业级应用程序，并不是特指为企业开发的应用软件，而是泛指那些为大型组织部门创建的应用程序，利用这些程序，可以在一定程度上为上述单位的日常工作和业务活动提供支撑。因此，企业级应用是严格区别于学校老师布置的作为作业的小程序，或者为讨女朋友开心而开发的小游戏。对于后者，由于软件规模较小，不需要考虑经费投入和技术风险，完全可以天马行空，任意发挥，无需考虑采用什么框架。但企业级 Web 应用开发是一项复杂的工程，从软件功能要求到开发过程管理，都有着严格的规范。

企业级 Web 应用在功能要求上有以下特点。

（1）功能复杂。除了满足特定应用领域的业务功能要求外，企业应用由于其特殊的运行环境和管理要求，还具有一些复杂的特殊要求。例如，为确保企业应用安全，企业应用要求具有严格的权限管理和日志审计；为实现企业信息的高度共享，还要求企业应用与现有的信息系统如企业邮件服务器、银行系统等实现对接；为满足企业生产国际化的需求，要求企业应用实现国际化的用户界面；等等。而这些重要的功能要素恰恰是开发一般的应用系统所忽略的。

（2）较高的稳定性。企业应用的运行与企业单位的业务工作息息相关，因此企业应用的一个显著特点就是较高的稳定性能要求，必须确保企业应用长时间连续正常运行，在遇到异常或错误时能够正确处理。这就要求应用开发在设计分析阶段就建立起良好的软件架构。

（3）良好的扩展性。企业应用开发的基础和起点就是用户需求，用户需求分析是一个由模糊到清晰的渐进过程，而且在开发过程中，用户的需求由于各种因素的影响几乎不可避免的发生变化。另外，许多企业应用开发项目本身就是分多期进行的。企业应用的需求不断变化，这就要求应用设计中采用灵活的可扩展的框架，以便适应不断变化的需求。

企业级 Web 应用在软件开发管理上具有以下特点。

（1）时间紧，任务重。有过软件开发企业工作经历的读者都会亲身感受过，一旦接手一个企业应用开发项目，所面临的最大困难就是复杂的功能要求和紧迫的开发周期。这就要求企业充分利用已有的开发工作积累（包括可重用的开发框架，开发组件）来尽可能的提高开发工作效率。

（2）规模化开发。企业应用开发一般采用多人参与、分工协作的开发模式。这就要求应用开发在架构上适应目前的规模化开发模式，以确保较高的工作效率和工程化的管理模式。

1.2.2　Web 应用模型的先天不足

1.2.1 节讲了企业级 Web 应用开发的高标准严要求，但回过头来看一下 Web 应用模型，却存在着技术体制上的“先天不足”。在 1.1 节讲过，不管是静态 Web 应用还是动态 Web 应用，它们采用一致的工作模式：应用由服务器端和客户端两部分组成，服务器端和客户端通过 HTTP 协议采用“请求－应答”模式进行通信。

从上面的描述可以看出 HTTP 协议在 Web 应用模型中的重要地位。作为一种简单的通信协议，HTTP 协议在促进 Web 应用得到快速普及上功不可没。但“成也萧何，败也萧何”，由于 HTTP 协议最初是针对静态 Web 应用而设计的，随着开发复杂的动态 Web 应用的不断深入，HTTP 协议渐渐表现得力不从心。主要原因有以下两点。

1．HTTP 协议是一种无状态的协议

HTTP 协议是一种非连接的通信协议，它采用“请求－应答”模式的通信方式。浏览器作为客户端向服务器发起请求，服务器根据客户端的请求，返回一个静态页面或者动态生成的页面作为应答，此时浏览器与客户机之间的一次通信过程结束，二者之间的连接将中断，服务器对客户端的状态信息将不做任何保留。如果客户端浏览器需要发起新的资源请求，则对不起，一切重头再来，需要客户端按照前面的步骤重新启动新一轮的“请求－应答”通信过程。

HTTP 协议的无状态特性主要源于最初满足静态 Web 应用的通信需求。在静态 Web 应用中，客户端浏览器向 HTTP 服务器发起请求，这个请求通常为代表服务器上特定资源的 URL，HTTP 服务器根据客户端请求将相应的资源作为响应发回给客户端，则信息传输过程完成。在上面的的过程中，无论对于客户端还是服务器，都没有必要记录这个过程，因为每一次请求和响应都是相对独立的，一个 URL 对应着唯一的超文本，而 HTTP 服务器也绝对公平公正，不管你是张三，还是李四，它都会根据接收到的 URL 请求返回相同的超文本。正是因为这样的唯一性，使得记录用户的行为状态变得毫无意义，所以，HTTP 协议被设计为无状态的连接协议符合它本身的需求。反之，如果将 HTTP 设置为保持状态的基于连接的方式，除了增加协议的复杂性外，每一个客户连接都将占用服务器大量的资源，导致服务器性能随着用户数目的增多而急剧恶化。

但随着动态 Web 应用的出现，HTTP 无状态的特性却逐渐成为一种障碍。因为在动态 Web 应用中，服务器与浏览器的交互变得越来越复杂，服务器必须记住从同一客户发起的各种请求，才能够进行正确响应，以简单的购物车为例，服务器端维护购物车页面必须知道客户在以前历次操作中的具体内容。于是，为了支持动态 Web 应用中客户端与服务器之间的交互，开发人员不得不采用各种复杂的会话管理技术如 Cookie 或者 Session 来实现客户端与服务器之间的交互状态的维护管理。（关于利用 Cookie 或者 Session 来实现交互状态的维护管理详细内容请参看《Java EE 编程技术》。）

2．HTTP 仅能传输文本

让我们仔细研究一下 HTTP 协议的名称，它的全称是 Hyper Text Transfer Protocol，即超文本传输协

议。顾名思义，它仅被设计用来传输超文本，即 HTML 网页，因此通过 HTTP 传输的内容都是采用文本的形式。在服务器端，根据服务器 Web 组件的实现方式，采用文本形式表示的请求参数将被转换为对应的数据类型，如在 Servlet 中，客户端的各种参数需要转换成 Java 对象，同样，服务器端产生的动态响应信息又必须先转换为文本的形式并通过 HTTP 协议传输到客户端。这种信息的转化过程很烦琐，同时又很容易出错，但却是必不可少的，已成为广大 Web 应用开发人员挥之不去的梦魇。

1.2.3 Java EE 身后的空白

Java EE 是一个基于 Java 编程语言的企业 Web 应用开发解决方案。在 Java EE Web 应用解决方案中，提出了两种 Web 组件：Servlet 和 JSP。它们都是运行在服务器端的 Java 程序，不同的是，JSP 更侧重于应用表现层的实现，而 Servlet 更侧重于业务逻辑层的控制。由于 JSP 在运行之前也会被服务器编译成 Servlet，因此它们其实是同一种组件。下面以 Servlet 为例来讲解 Java EE 解决方案是如何实现动态 Web 应用开发的。

我们可以回顾一下 Servlet 的工作原理。Servlet 运行在服务器上，接收到客户端的请求，则创建一个新的线程来处理请求，并返回生成的动态结果页面。Servlet 处理客户的请求的方法为 Service 方法，下面来看一下 Service 方法的定义：

```
public void service(ServletRequest req,ServletResponse res) throws ServletException,
java.io.IOException
```

从上面的定义可以看出，Java EE 服务器将客户端的请求和服务器的响应分别用对象 ServletRequest 和 ServletResponse 进行了封装，在 Servlet 中可以通过直接调用上述对象的方法如 getParmerter()来获取 HTTP 通信过程中的请求信息或设置响应的内容。

另外，为弥补 HTTP 的无状态性的特征，解决 Servlet 对同一客户的不同请求的会话跟踪问题，Java EE 服务器还提供了 HttPSession 对象来实现 Servlet 组件的会话管理。

遗憾的是，Java EE 解决方案就为我们提供了这么多，它虽然规范了企业 Web 应用的开发模式和实现标准，但在它的身后，留下了巨大的空白。让我们仔细来看一下这片空白中都包含哪些内容。

（1）类型转换。前面已经提到，由于 HTTP 协议仅能够传输二进制文本，客户端的各种参数需要转换成 Java 对象，同样，服务器端产生的动态响应信息又必须先转换为文本的形式并通过 HTTP 协议传输到客户端。这种复杂乏味的信息转化工作不得不由程序员手工完成。

（2）输入校验。所谓“病从口入”，作为 Web 应用中的信息输入源头，客户端的输入信息必须经过严格的校验来确认其合法性。这种校验工作是保证应用程序安全的重要前提。在 Java EE 中，它也必须由程序员手工来完成。

（3）架构定义。在一个复杂的 Web 应用中，按照逻辑关系，Web 应用可以分为表现层，逻辑层和数据层等，其中的每一层都可以进一步划分为更多的层次。随着 Web 应用规模的扩大，上述不同层次间的接口定义变得越来越关键。如果层次间的接口定义得合理明晰，无论 Web 应用怎么变化，整个应用的整体结构是不变的，某一层的变化也不会传递到应用中的其他层面，则从根本上确保 Web 应用具有良好的稳定性和扩展性。反之，如果定义不合理，Web 应用的任何一处的变化都会引起整个应用架构的“震荡”，你就会发现自己像一头陷入沼泽的大象，越努力挣脱，结果却陷得越深。Java EE Web 解决方案只是从技术的角度定义了 Web 应用的一种标准的解决方案，它仅仅规定了 Web 应用组件模型的工作接口，而不会也不可能提出一种标准的 Web 应用架构模式。

（4）国际化与本地化实现。国际化与本地化是企业应用系统的基本要求，开发人员必须实现这一基本特性，确保应用的适应特性。

看到上面这些，你可能对开发一个优秀的 Web 应用系统产生恐惧。不过没关系，“框架”来了。

1.3　什么是框架

框架是软件工程中一个比较抽象的概念，可以从以下两个角度来理解。

从软件设计的角度，框架是一个可复用的软件架构解决方案，它规定了应用的体系结构，阐明了软件体系结构中各层次间以及层次内部各组件间的依赖关系、责任分配和控制流程，表现为一组接口、抽象类以及其实例之间协作的方法。框架的使用将大大降低应用系统的设计难度，确保系统设计的质量。

从软件实现的角度，框架是软件快速实现的基础平台，它包含一组可重用的组件，它使得某一领域内的软件的基础功能和通用流程的实现更加高效便捷，将使得开发人员可以专注于特定业务逻辑，从而大大提高软件的开发效率。

从上面的定义可以看出，框架都具有一定的适用范围，都是针对特定应用领域软件开发的。因此，作为程序开发人员要时刻牢记：没有可以适合任何领域的万能框架，就像没有普遍存在的真理一样。

采用框架技术进行软件开发的主要优点如下。

（1）确保开发质量。框架对整个软件的体系进行了合理的规划设计，对一些常用的功能提供了基础实现，尤其是一些开源的架构，如 Struts2 等，经历了全世界范围内的众多开发人员的共同努力以及数以万计的软件开发项目的实践，从根本上确保了软件的稳定性和扩展性。

（2）提高开发效率。框架的最大优点就是重用。重用包括两个层次。一是应用分析设计上的重用，框架已经设计架构好了应用的整体架构以及各层之间的接口关系，这就使得开发人员可以专注于业务领域的设计分析。二是代码上的重用，框架中提供了一些通用功能组件的实现，将在很大程度上减少开发人员工作量。更重要的是，通过框架，实现了对应用开发中底层 API 的封装，降低了开发难度，大大提高了开发效率。

（3）降低软件开发维护费用。框架将软件开发合理地划分成几个工作层面，有利于在一个项目内多人协同工作。框架的使用，使得新加入项目的人员可以尽快了解项目的整体架构、技术规范和编码，这样就省去了对新加入人员必须开展培训带来的开支。开源框架是不断更新维护的，这对软件后期的维护也是一个有利的支撑。

在 1.1 节讲过，不管是静态 Web 应用还是动态 Web 应用，它们采用一致的工作模式：应用由服务器端和客户端组成，服务器端和客户端通过 HTTP 协议采用“请求-应答”模式进行通信。正是 Web 应用这种连续一致的工作模式使得 Web 应用框架的存在成为可能。目前业界比较流行的 Web 应用框架有 Struts2、Spring、Tapestry、JSF 等。各种不同的框架各擅其长，不过要说起影响力，自然要首推 Struts2。本书也将深入学习 Struts2 框架。

1.4　认识一下 Struts2 框架

好了，前面说了那么多，主要是介绍 Web 应用和框架等 Struts2 相关的背景知识，现在该转入正题，让我们好好认识一下 Struts2。

1.4.1　Struts2 的前世今生

虽然“英雄不问出处”，但搞清楚 Struts2 框架产生的来龙去脉对于学习 Struts2 还是很有帮助的。

首先要澄清一个概念，Struts2 并不是 Struts 框架的 2.0 版本，而是在 Struts 框架和 Webwork 框

架基础上发展起来的一个全新的框架。作为全世界第一个发布的成熟的 MVC 框架，Struts 框架仍然在 Apache 旗下继续推进，截止到本书写作时的最新版本为 1.3.10 版本。而 Struts2 框架的最新版本为 2.1.8.1。

从某种程度上来讲，Struts2 没有继承 Struts 的血统，而是继承 WebWork 的血统，它的大部分设计思想和理念都是与 Webwork 一脉相承的。或者说，WebWork 衍生出了 Struts2，而不是 Struts 衍生了 Struts2。但为什么称其为 Struts2 而不称为 WebWork2，我认为在很大程度上是因为 Struts 框架影响太大的缘故。因为 Struts2 是 WebWork 的升级，而不是一个全新的框架，而且吸收了 Struts 的一些优势，因此稳定性、性能等各方面都有很好的保证，是一个非常值得期待的框架。

所以，现在的 Struts 框架在不断进化的过程中产生了两个分支：一个是原来老的 Struts 框架，通常称为 Struts 1.0，仍然采用原来的架构体系和设计理念；另一个就是 Struts 2.0 框架，它继承了 Webwork 的设计思想和理念，并且吸收了 Struts 框架的一些优势。从这个意义上来说，只有 Struts2 代表了 Struts 框架的未来发展。Struts 框架的这种演变也正印证了所谓的“天下大势，合久必分”的道理。在 Java EE 开源的指导思想下，不同的设计思想和理念在实践中不断交融本来也是再正常不过的事了。

1.4.2 Struts2 框架的基本思想

Struts2 框架的基本思想是采用 MVC 设计模式，即将应用设计成模型（Model）、视图（View）和控制（Control）三个部分。为了更好地领会 Struts2 框架的基本设计思想，我们有必要详细讲解一下 MVC 设计模式。

提起 MVC，在软件开发设计领域可是鼎鼎大名。MVC 其实不是一个新鲜事物，它最早由Trygve Reenskaug在 1974 年提出，是为程序语言Smalltalk发明的一种软件设计模式。令人欣慰的是，它一经提出，就风靡至今。

我们知道，软件设计的核心就是对现实世界中的复杂系统建模的过程。世界是运动的，软件建模就是对运动之中的系统的本质的把握。MVC 模式巧妙地将系统分成了模型、视图和控制三个部分，三者之间各自具有独立的功能，彼此之间又存在着清晰的关联。

视图是用户看到并与之交互的界面。视图的职能是实现系统与用户的交互，包括将系统中的数据显示给用户以及接收用户的输入信息等。对老式的 Web 应用程序来说，视图就是由 HTML 元素组成的界面，在新式的 Web 应用程序中，HTML 依旧在视图中扮演着重要的角色，但一些新的技术已层出不穷，它们包括 Macromedia Flash 和 XHTML，XML/XSL，WML 等一些标识语言以及 Web Services 等。

模型是数据和业务规则的集合。所谓的业务规则是对数据处理的算法，这种业务规则往往是特定与专门的应用系统的，因此也是开发人员工作量比较集中的地方。在一个 Web 应用系统中，模型既包含存储在后台数据库系统中的数据，还包括对这些数据进行处理的算法、业务规则的程序组件。被模型返回的数据是与显示无关的，数据表现任务是由视图来完成的。

有了模型和视图组件，但是如何将二者关联起来呢？这就需要控制器组件出场了。控制器的作用就像一个前台，它接收用户的请求，然后协调视图和模型组件完成用户请求处理。

下面来看看在 Java EE Web 解决方案中，是如何来实现 MVC 模式的。如图 1-1 所示，视图由 JSP 来承担，控制器由 Servlet 来实现，而模型由 JavaBean 和后台的数据库来完成。首先由浏览器向控制器 Servlet 发起请求，控制器根据请求创建相应的模型 JavaBean，模型 JavaBean 访问后台的数据完成指定的业务逻辑操作并返回一个结果，控制器 Servlet 根据模型返回的结果将请求转发到一个视图 JSP，JSP 在处理过程中要获取模型中的数据，最后 JSP 生成的动态页面返回到浏览器，一个完整的 Web 应用处理流程到此结束。

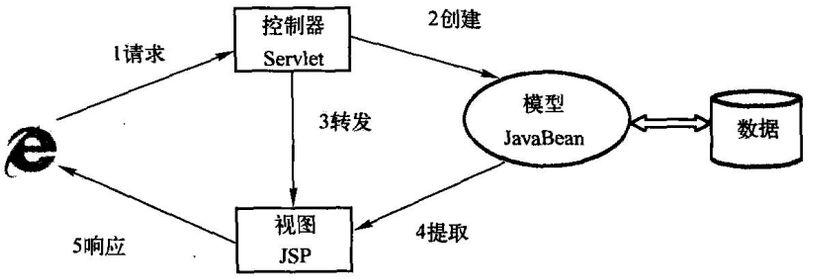

图 1-1　Java EE 中的 MVC 实现

MVC 设计模式，将应用系统的职能进行了合理的划分，通过将应用划分成模型、视图和控制三部分，大大降低了系统设计实现的难度。由于运用 MVC 的应用程序的三个部件是相互独立，改变其中一个不会影响其他两个，所以依据这种设计思想能构造良好的、松耦合的组件，确保整个应用系统架构设计的稳定性和扩展性。

软件设计的根本任务，就是对软件中各层次组件之间的合理的职责划分和接口设计。软件设计的终极目标，就是追求所谓的“高内聚，低耦合”。高内聚就是使相同或相近的功能尽可能使用相同的组件来实现，以提高软件的重用度；低耦合就是使不同组件间尽可能少地产生依赖关系。MVC 设计模式正是对“高内聚，低耦合”这一内涵的坚决贯彻，这也是它历经 20 余年仍然长盛不衰的根本原因。

说明：俗话说“金无足赤，人无完人”，在了解 MVC 设计模式优越性的同时，我们也要清醒地认识到 MVC 模式的不足。首先，将一个系统合理地划分为 MVC 三部分并不是一件容易的事情，开发一个 MVC 架构的系统，将不得不花费相当可观的时间去考虑如何将 MVC 运用到系统设计中，同时由于模型和视图要严格地分离，这样也给程序设计实现带来了一定的困难。另外，由于 MVC 将一个应用程序分成了三个部件，这就意味着开发人员需要编写更多的程序代码。

因此 MVC 并不适合小型甚至中等规模的应用程序，这样会带来额外的工作量，增加应用的复杂性。但对于开发存在大量用户界面，并且逻辑复杂的大型 Web 应用，MVC 将会使软件在健壮性、代码重用和结构方面上一个新的台阶。尽管在最初构建 MVC 框架时会花费一定的工作量，但从长远的角度来看，它会大大提高后期软件开发的效率和系统的扩展性和可维护性。

1.4.3　Struts2 框架的 MVC 实现

前途是光明的，道路是曲折的。尽管 Java EE Web 解决方案中的 MVC 流程实现过程清晰明了，JSP、Servlet 和 JavaBean 分别作为视图、控制和模型各司其职，但是要在 Java EE Web 应用中实现 MVC 模式，还有许多具体细致的工作需要去做，其根本原因在于 MVC 三者之间存在着复杂的数据交换和逻辑关联。

首先来看 MVC 之间的数据交换。视图要向控制提交请求信息。控制在创建模型的同时，将请求数据流导向模型。模型完成数据处理的过程也是数据自我更新的过程。当控制通知视图更新时，视图又要从模型提取数据，这时数据又从模型流回视图。也就是数据流经历了视图—模型—视图的循环过程。在数据的流动过程中，数据的类型需要在二进制文本和 Java 类型间转换，出于安全的目的，数据还要进行必要的校验等，这些职能在设计时如何来分配不是一件容易的事情。

再看一下 MVC 之间的逻辑关联。为了正确处理客户端发起的请求，控制器需要在视图与模型之间建立起正确的关联。模型对数据的处理可能产生不同的结果，则对应不同的视图，控制器需要建立模型和多个结果视图之间的关联。随着应用规模的扩大，这种复杂的关联关系变得更加眼花缭乱。更为致命的是，事情往往不是一帆风顺的，在各个环节都可能出现异常，则导致不同的行为发生，应用程序变得愈发难以控制。如何优雅地解决试图、模型和控制之间的关联关系，使得应用程序更加容易控制显然是个