



TURING

让Spring之父Rod Johnson拍案叫绝的原创经典

涵盖Spring 3.0

王福强 著

Spring

揭秘



 人民邮电出版社
POSTS & TELECOM PRESS

TURING

王福强 著

Spring

揭秘



人民邮电出版社
北京

图书在版编目 (CIP) 数据

Spring 揭秘 / 王福强著. —北京: 人民邮电出版社,
2009.9

ISBN 978-7-115-20942-9

I. S… II. 王… III. JAVA 语言—程序设计 IV. TP312

中国版本图书馆CIP数据核字 (2009) 第088998号

内 容 提 要

本书以幽默生动的语言、辅以有趣的故事和典故, 循循善诱地阐述了 Spring 框架的方方面面。针对 Spring 框架的主要功能以及开发者们遇到最多的问题, 首先介绍问题的相关背景, 然后逐条进行深度剖析, 最后通过分析来引入 Spring 框架可以提供的最佳解决方案。虽言 Spring, 却不局限于 Spring, 本书向读者展现了更宽广的软件开发的世界!

本书非常适合 Java 开发人员阅读和参考。

Spring揭秘

- ◆ 著 王福强
责任编辑 傅志红
执行编辑 杨 爽
- ◆ 人民邮电出版社出版发行 北京市崇文区夕照寺街14号
邮编 100061 电子函件 315@ptpress.com.cn
网址 <http://www.ptpress.com.cn>
北京艺辉印刷有限公司印刷
- ◆ 开本: 800×1000 1/16
印张: 42.5
字数: 1197千字 2009年9月第1版
印数: 1-3 500册 2009年9月北京第1次印刷

ISBN 978-7-115-20942-9/TP

定价: 99.00元

读者服务热线: (010)51095186 印装质量热线: (010)67129223

反盗版热线: (010)67171154

推荐序（一）

在Spring作为关键词在每一位技术人员简历里面出现的今天,我们可以深刻地体会到Spring的春天真的来了。

Spring经过几年的实战和演变,已经不再是一个单纯的体系框架,而代表着轻量级Java开发的规范。在短短几年的发展中, Spring给Java企业级开发带来了无比强大的推动力,让轻量级开发技术飞速发展。在Spring中,简单实现涵盖了一切元素,让我们将Java这门语言的所有特性发挥得淋漓尽致。通过Spring,你不仅可以学到如何应对每个领域的最佳实践,而且可以深入理解Java企业级开发中各层面的体系结构。因而,这简单的背后隐藏着巨大的秘密。

本书是揭开秘密的一把钥匙。作者借用大量的生活化比喻将Spring各环节的技术概念清晰、简单地展现在读者面前,让众多技术概念不再生硬难懂。同时,本书借助对Spring各项功能的解读,将读者带入Java企业级开发的方方面面,不仅具体讲述了如何应对领域问题,而且还将背后的领域知识关联起来,让我们了解其因果关系和各种实践的差异和不同。更值得称赞的是书中的实例场景非常实用,可见作者的代码精心提取于实战项目,让读者在深入理解概念应用的同时,可以参考实际的使用方法去解决自己项目中同类型的问题,为可行性提供了有力的证明。

通过阅读,相信读者获得的不仅仅是对Spring的深入理解,而是一种系统化知识的演练,一种开放构架思维的突破,一种解决问题的思路方法。如果你也想将Spring作为关键词放入你的简历之中,那么我相信你需要了解的不仅是如何使用Spring这样简单的问题,还需要知道它背后众多的“秘密”。

Yanger

Spring中文论坛创始人,
现任SpringTag.com运营总监

推荐序（二）

回顾昔日传统的IT世界里经常有这样两个场景。

场景一：大厂垄断。名牌大厂通过一定时间对标准、技术、产品和解决方案的把持，慢慢把小玩家挤出局，做成一家独大的局面。长此以往，对于标准和技术的审视、重构和优化几乎成为象牙塔里的说道，纯粹意义上的技术革新极难付诸实施。

场景二：杂草场景。最初标准不完善，发展线路不清晰，然后诸多小玩家入场，肆意推崇各类技术点上的终极解决之道，彼此间常漠视或势同水火。结果经年下来，造成众多IT孤岛，需要多层封装转化才可打通，或者干脆残留诸多极难考证的IT化石。

Java的发展历程很精彩，完全可以就此写本抓人眼球的书。最初有Sun开门立派，独辟蹊径在1998年推出J2EE规范对抗当时如日中天的微软，甫一推出，市场便震惊于其对于企业级解决方案思考的广度和深度。适逢.com风潮，对于J2EE的追捧自然而至，各大厂商也纷纷入围圈地企图瓜分霸权。

慢慢地，Java世界的发展步调有点前述场景一的味道，EJB的厚重迂腐，大厂方案中的平台方案绑定，高耦合性以及高端价位，还有其他众多同质解决方案的无序并存，对于各类中小企业乃至高端企业应用都有不堪承受之重，而这些都促使以Apache、JBoss等各路开源方案为代表的草根力量积极质疑反思。

2002年Rod Johnson的*Expert One-on-One J2EE Design and Development*一书的出版让很多开发人员心有戚戚焉。作为轻量级企业解决方案的提出者，他不局限于坐而论道，而是迅速通过Spring这个开源框架将其思想付诸实施并接受公众的批评和反馈，利用IOC、AOP这些思考基石，全面支持JavaEE要求的所有必要功能，凭借自身不断增强的功能和迅速吸纳集成市场其他专项解决问题的能力，Spring已慢慢成为Java企业开发事实上的标准。

也许其中的意义并不在于草根力量迫使大厂淡出Java世界的霸权争夺，而是让人们看到了一种开放的力量有效地指导技术发展方向，积极容纳各种前沿产品并和谐发展。

说完Java和Spring，谈回这本书和本书作者阿福。

有这样一个段子我印象很深刻，有个出名的油画鉴赏家，他一个不太为人知的秘密就是，他常去看还未装裱的作品的侧面，通过画布侧面油漆的厚度和颜色的变化去解读作者花在这个作品的心血和历程。这里做为本书见证人之一，我也分享下本书的侧面像。

我初识阿福于2004年，那时，他很痴迷于跆拳道，嘴里喊着“阿达”，目视虚空一目标点，然后腿上比划各种上抬角度。但让人惊讶的是，就这个毕业还没满一年的人，写出的Java代码却难得地考究和优雅。渐渐发现，那时的他对各类Java世界活跃元素的追踪热情也丝毫不亚于跆拳道。最初的Spring 1.0刚推出没多久就被他应用到一个日本大型信贷项目的批作业中，去解决各类强耦合问题和事务管理，之后就见他在诸多让一般人抓狂的复杂金融业务系统和场景中，慢慢试验、体会和推广Spring。2006年，他和我们常念叨，市面上真没一本值得推敲的Spring中文书，也没有见一本有份量的一线设计开发人员悉心奉上的心路思考过程。没多久他就辞职闭关，潜心写作该书。只可惜出版业的反应还

是慢了一些。

对于读者来说，须知万物溯其源，学东西不是为学而学，最初是有个场景、问题，然后是解决方案的提出、选择、权衡、实施、反思和改进。对于Java的学习和应用一直有个很有意义的话题就是，如能掌握精髓，自可海阔鱼跃天高鸟飞，只因心中已修得真经。可修行路上若方法不当，无人指导在真实复杂场景中的作战思路和心得，纵使看了规范，看了API，看了手册，看了诸多Hello World和样板代码，还是不解其意，只有硬头先抄，盼经年累月恍然大悟。于是常有人叹修行路上苦无机缘，总是要白走几年。这就是古人常说刻舟求剑总不得，授人鱼不如授人以渔一意。

在本书中，阿福通过多年复杂业务场景下的技术和业务上的淬火打磨，把Spring对于企业方案思考的由来、底层实现和与真实世界业务系统的结合悉心架构，娓娓道来，实乃近年国内同类书中惊鸿一现之作。

也藉此希望中国IT业可以有越来越多的一线人员奉献出更多嘎嘎独造可藏之于名山的作品。

——李昕焰，大连尚嘉信息咨询有限公司总经理

致 谢

每一本书的诞生都伴随着一篇被认为是“例行公事”的致谢文字，但当你真的完成一本书的创作之后，就会发现，即使是“例行公事”地去撰写一篇致谢，也同样是饱含着感激之情的。正因为有了身边那么多人的支持，作者才能够历尽艰辛，战胜写作期间的彷徨和孤苦，将最终的书稿呈现到大家的面前。

我首先要感谢我的父母，有了他们的理解和支持，我才能走到今天。试想一下，年届而立，却依然没有成家，在这种情况下为了写书居然还辞去工作，我让父母承受了怎样的压力？我很庆幸，父母不仅自己承受着这样的压力，而且还在背后默默地支持我，并时时给予我鼓励和宽慰。

其次，我要感谢我之前工作单位的领导、同事和朋友们，感谢他们在精神和实际生活中给予我的无私帮助。Daniel（王昕宇）、Dino（李昕焰）及徐敬琪和曲静夫妻俩在我写书期间不但时不时地拉我出去“腐败”，给予了我精神上的极大支持，而且还不遗余力地帮我出谋划策。还有李永刚、郭勇胜、梁贵、阿九（史荣九）等更是适时地在精神上给予我支持。真心感谢所有这些关心我、支持我的朋友们。当然，我还要感谢张万创，是他无私地把自己的托管主机奉献出来，我才能使用www.spring21.cn为大家提前奉上本书的部分样章。

古人云，千里马常有，而伯乐不常有。我很幸运，自己尚未展现千里马之才，就遇上了刘江主编这个伯乐。当本书尚在写作期间时，刘江主编就主动联系我商讨出版事宜，这让我免去了许多后顾之忧，可以全身心地投入写作。所以，我应该感谢刘主编的知遇之恩，谢谢刘主编。

我还要真心地感谢本书的策划编辑杨福川，他在本书的出版过程中可谓尽职尽责，从初期的选题评估和申报，到后期的编辑和修改，始终不遗余力。对于出版过程中的相关事宜，我们更是合作无间，当大家手上捧着本书的时候，不要忘记福川兄所付出的努力。

最后，我要感谢国内的技术社区，尤其是JavaEye，从Robbin到社区中的各位兄弟，你们对本书的好评和建议给予了我源源不断的前进动力，最终推动我完成本书的编写。

应该说，这样一篇简短的致谢并不足以表达我内心的感激之情。虽然不能在这里一一罗列出那些给予我无私帮助的亲朋好友的名字，但希望你们知道，对你们的感激之情无时无刻不存我心。

前 言

早期的J2EE平台推出的EJB规范是为了简化分布式应用的开发，并在J2EE平台上为各种企业级服务提供最佳实践。相对于更早以前的诸如CORBA等分布式架构来说，EJB确实使分布式架构得以简化。在当时特定的历史条件下，基于EJB架构的企业级应用的确有其先进性。但是，随着时间的推移，基于EJB架构的企业级应用逐渐暴露出各种形式的不足。于是开发者们开始探索一种新的技术，希望这种技术能弥补EJB的缺陷，一场技术革命就此拉开了序幕。

实际上，我们可以从两个方面来看待EJB被“革命”的原因。首先，分布式架构更适用于那些大型的企业级应用，这些应用往往对安全性、可扩展性的要求更高。对于这类大型企业级应用来说，为了获得更重要的功能特性，分布式架构在性能等方面的缺陷是可以容忍的。然而，对于大多数基于J2EE平台的中小型企业级应用来说，分布式架构在性能方面的损失则是完全不可接受的，分布式架构的引入反而会增加开发和维护的难度，甚至导致整个项目失败。颇有讽刺意味的是，恰恰是开发者们早期对EJB架构的过度推崇造成了如今这种难以收拾的局面。EJB本身笨重的编程模型被应用到不合适的场景中，给积重难返的EJB压上了“最后一根稻草”。

其次，EJB规范本身可看作是为J2EE平台上的各种企业级服务提供的一种最佳实践，但是这一最佳实践实际上并未达到预期效果。实体Bean（Entity Bean）被证明是一种失败的实践，其他种种现象也从各个侧面反映了EJB的缺陷，如整个应用需要绑定到应用服务器、编程模型复杂而笨重、应用本身不易测试等。

随着EJB自身所暴露的缺陷越来越多，开发者们不再盲目崇拜EJB，而开始对它感到失望。在这种情况下，一场革命在所难免，而燃起这场革命的星星之火的就是Spring。Spring框架提倡一切从实际出发，使用基于POJO的轻量级编程模型推进整个应用的快速开发。通过使用IoC、AOP等先进的技术理念，结合对原有J2EE平台各种企业级服务的抽象和适度集成，Spring框架为Java平台上的企业级开发注入了新鲜的血液。

从Spring框架身上，纵观，你几乎可以看到整个基于Java平台的软件开发的演变历史；横看，你可以跟踪和捕捉当前业界最先进的理念和软件开发模式。如果是初次踏上Java平台这片领地，沿着Spring框架所展示的路线，你将迅速地领略到整个Java平台的强大魅力；如果你已经在Java平台“征战”多年，Spring框架一定能使你回忆起昔日所经历的“酸甜苦辣”，并更加深切地感受到Spring框架给我们带来的价值。

本书的主要目的就是和读者朋友一起以轻松愉快的心情饱览整个Spring框架的“无限风光”，让你能够在愉快的旅途之后以全新的状态投入到现实的Java企业级应用开发工作中去。

关于本书

促使我下决心写下这本书的原因很多，但主要可以归结为两点。一个是因为自身在经年的软件开

发过程中的所见所闻，当历尽千辛走到Spring阳光下的时候，再回头看看，会发现自己和他人走过的路是多么地泥泞和曲折，而这些现在是完全可以避免的。可是，就算是在2007的项目中，我却依然发现大部分人还是在过去的沼泽中挣扎，所以我想通过这本书传达一种信息，希望人们能够尽快的跳出那片早就应该远离的沼泽。另一个原因则是希望通过对Spring框架的介绍，为大家传达一种业界内先进的理念和开发模式，为国内软件开发过程的改进和开发质量的提高，尽自己的一份力。

帮助你了解如何使用Spring框架所提供的各种方案来解决开发过程中所遇到的实际问题固然重要，但从Spring框架中汲取更多的“营养”才是我们应该进一步关注的，知其然，知其所以然，然后我们才可以在Java开发领域内游刃有余，Spring框架中所展现的开发理念以及方法学等，都应该成为我们日常开发过程中不可或缺的伙伴。所以，本书除了会像其他Spring书籍一样，对Spring框架提供的各种功能和特性进行详尽的讲解，中间还将穿插Spring实现中牵扯的设计模式和最佳实践的分析，以期最大限度地挖掘Spring这座宝藏。

技术是发展的，但思想却是延续的，当我们借助Spring之船飞渡波澜不惊的水面亦或闯过惊涛骇浪之后，不要忘记，或许某天我们会踏上另一条更好的船舶，但Spring之船所带给我们的那些理念和思想，却应该始终陪伴着我们，激励我们也好，鞭策我们亦善……

作者在线

如果在阅读本书的过程中遇到任何疑问或发现任何问题，可通过我的个人网站<http://www.spring21.cn/>与我取得联系。我会悉心为你解答有关本书的任何问题，虽然这是无偿的，但是我很乐意这样做，因为这正是当初写作本书的初衷。

目 录

第一部分 掀起 Spring 的盖头来	
第 1 章 Spring 框架的由来2	
1.1 Spring之崛起.....2	
1.2 Spring框架概述.....3	
1.3 Spring大观园.....5	
1.4 小结.....8	
第二部分 Spring 的 IoC 容器	
第 2 章 IoC 的基本概念10	
2.1 我们的理念是：让别人为你服务.....10	
2.2 手语，呼喊，还是心有灵犀.....13	
2.2.1 构造方法注入.....13	
2.2.2 setter方法注入.....13	
2.2.3 接口注入.....14	
2.2.4 三种注入方式的比较.....15	
2.3 IoC的附加值.....15	
2.4 小结.....17	
第 3 章 掌管大局的 IoC Service Provider18	
3.1 IoC Service Provider的职责.....18	
3.2 运筹帷幄的秘密——IoC Service Provider 如何管理对象间的依赖关系.....19	
3.2.1 直接编码方式.....19	
3.2.2 配置文件方式.....20	
3.2.3 元数据方式.....21	
3.3 小结.....21	
第 4 章 Spring 的 IoC 容器之 BeanFactory22	
4.1 拥有BeanFactory之后的生活.....24	
4.2 BeanFactory的对象注册与依赖绑 定方式.....26	
4.2.1 直接编码方式.....26	
4.2.2 外部配置文件方式.....28	
4.2.3 注解方式.....31	
4.3 BeanFactory的XML之旅.....33	
4.3.1 <beans>和<bean>.....33	
4.3.2 孤孤单单一个人.....35	
4.3.3 Help Me, Help You.....36	
4.3.4 继承？我也会！.....50	
4.3.5 bean的scope.....51	
4.3.6 工厂方法与FactoryBean.....56	
4.3.7 偷梁换柱之术.....61	
4.4 容器背后的秘密.....66	
4.4.1 “战略性观望”.....66	
4.4.2 插手“容器的启动”.....67	
4.4.3 了解bean的一生.....74	
4.5 小结.....85	
第 5 章 Spring IoC 容器 ApplicationContext86	
5.1 统一资源加载策略.....86	
5.1.1 Spring中的Resource.....87	
5.1.2 ResourceLoader, “更广义 的URL”.....88	
5.1.3 ApplicationContext与 ResourceLoader.....91	
5.2 国际化信息支持 (I18n MessageSource).....97	
5.2.1 Java SE提供的国际化支持.....97	
5.2.2 MessageSource与 ApplicationContext.....98	

5.3 容器内部事件发布	102	第 8 章 Spring AOP 概述及其实现机制	135
5.3.1 自定义事件发布	102	8.1 Spring AOP概述	135
5.3.2 Spring的容器内事件发布类结构分析	105	8.2 Spring AOP的实现机制	136
5.3.3 Spring容器内事件发布的应用	107	8.2.1 设计模式之代理模式	136
5.4 多配置模块加载的简化	109	8.2.2 动态代理	139
5.5 小结	110	8.2.3 动态字节码生成	141
第 6 章 Spring IoC 容器之扩展篇	111	8.3 小结	142
6.1 Spring 2.5的基于注解的依赖注入	111	第 9 章 Spring AOP 一世	143
6.1.1 注解版的自动绑定 (@Autowired)	111	9.1 Spring AOP中的Joinpoint	143
6.1.2 @Autowired之外的选择——使用JSR250标注依赖注入关系	115	9.2 Spring AOP中的Pointcut	144
6.1.3 将革命进行得更彻底一些 (class-path-scanning功能介绍)	116	9.2.1 常见的Pointcut	146
6.2 Spring 3.0展望	119	9.2.2 扩展Pointcut (Customize Pointcut)	151
6.3 小结	120	9.2.3 IoC容器中的Pointcut	152
第三部分 Spring AOP 框架		9.3 Spring AOP中的Advice	153
第 7 章 一起来看 AOP	122	9.3.1 per-class类型的Advice	153
7.1 AOP的尴尬	124	9.3.2 per-instance类型的Advice	159
7.2 AOP走向现实	125	9.4 Spring AOP中的Aspect	163
7.2.1 静态AOP时代	125	9.4.1 PointcutAdvisor家族	164
7.2.2 动态AOP时代	126	9.4.2 IntroductionAdvisor分支	167
7.3 Java平台上的AOP实现机制	126	9.4.3 Ordered的作用	168
7.3.1 动态代理	126	9.5 Spring AOP的织入	170
7.3.2 动态字节码增强	126	9.5.1 如何与ProxyFactory打交道	170
7.3.3 Java代码生成	127	9.5.2 看清ProxyFactory的本质	175
7.3.4 自定义类加载器	127	9.5.3 容器中的织入器——ProxyFactoryBean	179
7.3.5 AOL扩展	127	9.5.4 加快织入的自动化进程	185
7.4 AOP国家的公民	128	9.6 TargetSource	190
7.4.1 Joinpoint	128	9.6.1 可用的TargetSource实现类	191
7.4.2 Pointcut	130	9.6.2 自定义TargetSource	195
7.4.3 Advice	131	9.7 小结	197
7.4.4 Aspect	133	第 10 章 Spring AOP 二世	198
7.4.5 织入和织入器	133	10.1 @AspectJ形式的Spring AOP	198
7.4.6 目标对象	133	10.1.1 @AspectJ形式AOP使用之先睹为快	199
7.5 小结	134	10.1.2 @AspectJ形式的Pointcut	201
		10.1.3 @AspectJ形式的Advice	211
		10.1.4 @AspectJ中的Aspect更多话题	220

10.2 基于Schema的AOP.....	223	14.2.1 基于操作对象的查询.....	303
10.2.1 基于Schema的AOP配置 概览.....	223	14.2.2 基于操作对象的更新.....	310
10.2.2 向基于Schema的AOP迁移.....	225	14.2.3 基于操作对象的存储 过程调用.....	313
10.2.3 @AspectJ到“基于Schema 的AOP”迁移.....	227	14.3 小结.....	316
10.3 小结.....	235	第 15 章 Spring 对各种 ORM 的集成	317
第 11 章 AOP 应用案例	237	15.1 Spring对Hibernate的集成.....	318
11.1 异常处理.....	237	15.1.1 旧日“冬眠”时光.....	318
11.1.1 Java异常处理.....	237	15.1.2 “春天”里的“冬眠”.....	321
11.1.2 Fault Barrier.....	238	15.2 Spring对iBATIS的集成.....	329
11.2 安全检查.....	239	15.2.1 iBATIS实践之“前生”篇.....	329
11.3 缓存.....	240	15.2.2 iBATIS实践之“今世”篇.....	331
11.4 小结.....	240	15.3 Spring中对其他ORM方案的集成 概述.....	337
第 12 章 Spring AOP 之扩展篇	241	15.3.1 Spring对JDO的集成.....	337
12.1 有关公开当前调用的代理对象 的探讨.....	241	15.3.2 Spring对TopLink的集成.....	340
12.1.1 问题的现象.....	241	15.3.3 Spring对JPA的集成.....	341
12.1.2 原因的分析.....	242	15.4 小结.....	344
12.1.3 解决方案.....	243	第 16 章 Spring 数据访问之扩展篇	345
12.2 小结.....	245	16.1 活用模板方法模式及Callback.....	345
第四部分 使用 Spring 访问数据		16.1.1 FTPClientTemplate.....	345
第 13 章 统一的数据访问异常层次体系	249	16.1.2 HttpClientTemplate.....	349
13.1 DAO模式的背景.....	249	16.2 数据访问中的多数据源.....	350
13.2 梦想照进现实.....	251	16.2.1 “主权独立”的多数据源.....	350
13.3 发现问题, 解决问题.....	252	16.2.2 “合纵连横”的多数据源.....	352
13.4 不重新发明轮子.....	254	16.2.3 结束语.....	354
13.5 小结.....	257	16.3 Spring 3.0展望.....	356
第 14 章 JDBC API 的最佳实践	258	16.4 小结.....	356
14.1 基于Template的JDBC使用方式.....	258	第五部分 事务管理	
14.1.1 JDBC的尴尬.....	258	第 17 章 有关事务的楔子	358
14.1.2 JdbcTemplate的诞生.....	261	17.1 认识事务本身.....	358
14.1.3 JdbcTemplate和它的 兄弟们.....	274	17.2 初识事务家族成员.....	360
14.1.4 Spring中的DataSource.....	296	17.3 小结.....	362
14.1.5 JdbcDaoSupport.....	301	第 18 章 群雄逐鹿下的 Java 事务管理	363
14.2 基于操作对象的JDBC使用方式.....	302	18.1 Java平台的局部事务支持.....	363
		18.2 Java平台的分布式事务支持.....	365
		18.2.1 基于JTA的分布式事务管理.....	366

18.2.2	基于JCA的分布式事务管理	367
18.3	继续前行之前的反思	367
18.4	小结	369
第 19 章	Spring 事务王国的架构	370
19.1	统一中原的过程	371
19.2	和平年代	376
19.2.1	TransactionDefinition	376
19.2.2	TransactionStatus	382
19.2.3	PlatformTransactionManager	382
19.3	小结	392
第 20 章	使用 Spring 进行事务管理	393
20.1	编程式事务管理	393
20.1.1	直接使用PlatformTransactionManager进行编程式事务管理	393
20.1.2	使用TransactionTemplate进行编程式事务管理	394
20.1.3	编程创建基于Savepoint的嵌套事务	396
20.2	声明式事务管理	397
20.2.1	引子	397
20.2.2	XML元数据驱动的声明式事务	399
20.2.3	注解元数据驱动的声明式事务	410
20.3	小结	413
第 21 章	Spring 事务管理之扩展篇	414
21.1	理解并活用ThreadLocal	414
21.1.1	理解ThreadLocal的存在背景	414
21.1.2	理解ThreadLocal的实现	415
21.1.3	ThreadLocal的应用场景	416
21.1.4	使用ThreadLocal管理多数据源切换的条件	417
21.2	谈Strategy模式在开发过程中的应用	420
21.3	Spring与JTA背后的奥秘	423
21.4	小结	427

第六部分 Spring 的 Web MVC 框架

第 22 章	迈向 Spring MVC 的旅程	430
22.1	Servlet独行天下的时代	430
22.2	繁盛一时的JSP时代	433
22.3	Servlet与JSP的联盟	436
22.4	数英雄人物, 还看今朝	438
22.5	小结	440
第 23 章	Spring MVC 初体验	441
23.1	鸟瞰Spring MVC	442
23.2	实践出真知	446
23.2.1	Spring MVC应用的物理结构	447
23.2.2	按部就班地开始工作	451
23.3	小结	459
第 24 章	近距离接触 Spring MVC 主要角色	460
24.1	忙碌的协调人HandlerMapping	460
24.1.1	可用的HandlerMapping	461
24.1.2	HandlerMapping执行序列(Chain Of HandlerMapping)	463
24.2	我们的亲密伙伴Controller	464
24.2.1	AbstractController	465
24.2.2	MultiActionController	468
24.2.3	SimpleFormController	476
24.2.4	AbstractWizardFormController	496
24.2.5	其他可用的Controller实现	503
24.3	ModelAndView	505
24.3.1	ModelAndView中的视图信息	505
24.3.2	ModelAndView中的模型数据	506
24.4	视图定位器ViewResolver	506
24.4.1	可用的ViewResolver实现类	507
24.4.2	ViewResolver查找序列(Chain Of ViewResolver)	511

24.5 各司其职的View.....	511	25.6.3 切换主题的ThemeChange- Interceptor.....	555
24.5.1 View实现原理回顾.....	512	25.7 小结.....	556
24.5.2 可用的View实现类.....	515	第 26 章 Spring MVC 中基于注解的 Controller.....	557
24.5.3 自定义View实现.....	521	26.1 初识基于注解的Controller.....	557
24.6 小结.....	523	26.2 基于注解的Controller原型分析.....	558
第 25 章 认识更多 Spring MVC 家族 成员.....	524	26.2.1 自定义用于基于注解的Contro- ller的HandlerMapping.....	558
25.1 文件上传与MultipartResolver.....	525	26.2.2 自定义用于基于注解的Contro- ller的HandlerAdaptor.....	560
25.1.1 使用MultipartResolver 进行文件上传的简单分析.....	526	26.3 近看基于注解的Controller.....	563
25.1.2 文件上传实践.....	527	26.3.1 声明基于注解的Controller.....	563
25.2 Handler与HandlerAdaptor.....	530	26.3.2 请求参数到方法参数的绑定.....	569
25.2.1 问题的起源.....	530	26.3.3 使用@ModelAttribute访 问模型数据.....	572
25.2.2 深入了解Handler.....	531	26.3.4 通过@SessionAttribute 管理Session数据.....	574
25.2.3 近看HandlerAdaptor的 奥秘.....	533	26.4 小结.....	576
25.2.4 告知Handler与Handler- Adaptor的存在.....	535	第 27 章 Spring MVC 之扩展篇.....	577
25.3 框架内处理流程拦截与Handler- Interceptor.....	536	27.1 Spring MVC也Convention Over Configuration.....	577
25.3.1 可用的Handler- Interceptor实现.....	537	27.1.1 Convention Over Configuration 简介.....	577
25.3.2 自定义实现Handler- Interceptor.....	538	27.1.2 Spring MVC中的Convention Over Configuration.....	578
25.3.3 HandlerInterceptor 寻根.....	540	27.2 Spring 3.0展望.....	581
25.3.4 HandlerInterceptor 之外的选择.....	541	27.3 小结.....	582
25.4 框架内的异常处理与Handler- ExceptionHandler.....	544	第七部分 Spring 框架对 J2EE 服务的 集成和支持	
25.5 国际化视图与LocalResolver.....	548	第 28 章 Spring 框架内的 JNDI 支持.....	584
25.5.1 可用的LocaleResolver.....	549	28.1 JNDI简单回顾.....	584
25.5.2 LocaleResolver的足迹.....	550	28.2 Spring框架内JNDI访问的基石—— JndiTemplate.....	585
25.5.3 Locale的变更与 LocaleChangeHandler.....	551	28.3 JNDI对象的依赖注入—— JndiObjectFactoryBean.....	587
25.6 主题 (Theme) 与ThemeResolver.....	552	28.4 小结.....	588
25.6.1 提供主题资源的 ThemeSource.....	552		
25.6.2 管理主题的 ThemeResolver.....	554		

第 29 章 Spring 框架对 JMS 的集成	589
29.1 说说JMS的身世	589
29.2 使用JMS API进行应用开发的传统 套路	590
29.3 Spring改进后的JMS实战格斗术	592
29.3.1 消息发送和同步接收	592
29.3.2 异步消息接收	601
29.3.3 JMS相关异常处理	607
29.3.4 框架内的事务管理支持	608
29.4 小结	609
第 30 章 使用 Spring 发送 E-mail	610
30.1 思甜前, 先忆苦	610
30.2 Spring的E-mail抽象层分析	612
30.2.1 直接创建邮件消息并发送	614
30.2.2 使用MimeMessage- Preparator发送邮件	615
30.3 Spring的E-mail支持在实际开发中的 应用	616
30.4 小结	622
第 31 章 Spring 中的任务调度和 线程池支持	623
31.1 Spring与Quartz	623
31.1.1 初识Quartz	623
31.1.2 融入Spring大家庭的Quartz	626
31.2 Spring对JDK Timer的集成	631
31.2.1 JDK Timer小记	631
31.2.2 Spring集成后的JDK Timer	632
31.3 Executor的孪生兄弟 TaskExecutor	634
31.3.1 可用的TaskExecutor	635
31.3.2 TaskExecutor使用实例	637
31.4 小结	639
第 32 章 Spring 框架对 J2EE 服务的 集成之扩展篇	640
32.1 MailMonitor的延伸	640
32.2 Spring 3.0展望	642
32.3 小结	642
第 33 章 Spring 远程方案	643
33.1 从“对面交谈”到“千里传声”	643
33.2 Spring Remoting架构分析	645
33.2.1 Spring Remoting之远程访问 异常体系	645
33.2.2 统一风格的远程服务公开与 访问方式	646
33.3 Spring Remoting提供的远程服务 支持	648
33.3.1 基于RMI的Remoting方案	648
33.3.2 基于HTTP的轻量级 Remoting方案	651
33.3.3 基于Web服务的远程方案	655
33.3.4 基于JMS的远程方案	658
33.4 扩展Spring Remoting	660
33.5 Spring Remoting之扩展篇	663
33.5.1 拉开JMX演出的序幕	663
33.5.2 Spring 3.0展望	664
参考文献	665

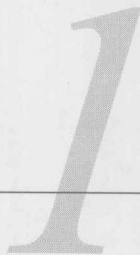
Part 1

第一部分

掀起 Spring 的盖头来

本部分内容

■ 第 1 章 Spring 框架的由来



本章内容

- Spring之崛起
- Spring框架概述
- Spring大观园

J2EE作为一种企业级应用开发平台，其优异表现是我们有目共睹的。但纵然是最为强大的军队，如果没有一个好的指挥官，不知道如何发挥这支军队的强大战斗力，那这支军队也不可能取得太多辉煌的战果。在J2EE平台的一些早期实践中，就出现了对J2EE平台所提供的各项服务的滥用，将基于J2EE平台的企业级开发带入了窘境。

Spring是于2003年兴起的一个轻量级的Java开发框架，由Rod Johnson在其著作*Expert One-On-One J2EE Development and Design*中阐述的部分理念和原型衍生而来。它的最初目的主要是为了简化Java EE的企业级应用开发，相对于过去EJB^①时代重量级的企业应用开发而言，Spring框架的出现为曾经阴霾的天空带来了灿烂的阳光。

1.1 Spring 之崛起

在中世纪的欧洲，当重装骑兵所向披靡时，哪国的军队中如果没有一支重装骑兵真的会让人笑话的，按照电影《大腕》里的一句话说“你都不好意思跟人打招呼”。应该说，在当时的历史/军事环境下，重装骑兵在军队中确实发挥了不可或缺的作用。有时候，一次关键时刻的重装骑兵冲锋就可以奠定战局的胜利。但是，时过境迁，历史的车轮一直在向前缓缓行进，重装骑兵头上的光环也随之渐趋黯淡，其缺点开始显露无遗。

- 重装骑兵代价高昂。一名重装骑兵的装备花费几乎能够武装一小队轻步兵，对于财力不够雄厚的国家来说，维持一支常备的重装骑兵队伍绝非易事。实际上，对于财力雄厚的大国（相当于IT界的IBM、微软）来说，为了减轻财政上的压力，通常也是将这部分花销尽量摊派给贵族。
- 兵种自身限制太多。沉重的盔甲以及一整套装备使得重装骑兵的机动性和灵活性大打折扣，在正式投入战斗之前，重装骑兵需要很长时间的列装和部署，对于瞬息万变的战场形势来说，某些情况下，这等同于自杀。
- 发挥作用的场景有限。纵使各翼军队能够掩护重装骑兵完成部署，但如果战场地形不适合重装骑兵冲锋，那也就无法让他们大显身手，前期的准备或者战斗掩护就更是得不偿失。

当新的战术和兵种能够更加高效地完成作战任务时，依然主打重装骑兵的军队能取得什么样的战

① 此后提到的EJB通常都是指1.x和2.x版本的EJB，因为EJB3现在已经受Spring框架的影响，也主打基于POJO的轻量级应用解决方案了。