

Java 开发专家

精通

Spring

——深入Java EE开发核心技术

Spring 2.5 是迄今为止最完美的 Java EE 架构级框架

全面深入、多维度演绎 Spring 2.5 的各个方面

本书蕴含作者多年 Java EE 研发实践及经验



罗时飞

飞思科技产品研发中心

编著

监制



电子工业出版社

PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
<http://www.phei.com.cn>

精通 Spring

——深入Java EE开发核心技术

罗时飞

飞思科技产品研发中心

编著
监制

内 容 简 介

本书是关于 Spring 2.5 的权威教程,是 Java/Java EE 开发者必备的参考书。本书详尽、系统地介绍了 Java EE 的基础知识、Spring 2.5 的各种功能,以及 Spring 2.5 的高级使用技巧和最佳实践。全书共分为 5 篇:第 1 篇为综述,主要围绕 Java EE 5、Spring 展开;第 2 篇介绍 Spring 2.5 核心技术,主要围绕 Spring 元框架进行阐述;第 3 篇介绍 DAO 层集成技术,主要围绕 JDBC、Hibernate 和 JPA 等持久化技术展开论述,针对 Spring 使能应用的事务管理和集成测试,也进行了相关介绍;第 4 篇介绍 Java EE 服务及技术的集成,主要围绕企业应用中使用的各种 Java EE 服务及技术展开论述;第 5 篇介绍 Spring 2.5 高级特性,主要从忘却的 Spring 高级话题和 Spring 最佳实践角度给出论述;附录 A 完整地介绍了 Spring 2.5 支持的各种命名空间及其中的所有元素。全书理论与实践并重,通过大量的实例帮助读者尽快掌握 Spring 2.5 的使用技巧,从而提高本书的参考、阅读价值。

本书适合作为 Java/Java EE 开发者、系统分析师和架构师的参考书,同时,本书非常适合于高校相关专业业的学生,以及对 Java/Java EE 开发有兴趣的各类开发者。

未经许可,不得以任何方式复制或抄袭本书的部分或全部内容。

版权所有,侵权必究。

图书在版编目(CIP)数据

精通 Spring:深入 Java EE 开发核心技术 / 罗时飞编著.北京:电子工业出版社,2008.10

(Java 开发专家)

ISBN 978-7-121-07298-7

I. 精… II. 罗… III. Java 语言—程序设计 IV. TP312

中国版本图书馆 CIP 数据核字(2006)第 131476 号

责任编辑:杨 鹂

印 刷:北京智力达印刷有限公司

装 订:北京中新伟业印刷有限公司

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

开 本:787×1092 1/16 印张:33.75 字数:864 千字

印 次:2008 年 10 月第 1 次印刷

印 数:5 000 册 定 价:59.00 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 zlt@phei.com.cn, 盗版侵权举报请发邮件至 dbqq@phei.com.cn。

服务热线:(010) 88258888。

热烈祝贺“开发专家之 Sun ONE”系列被
评为电子工业出版社“最佳品牌奖”

“开发专家之 Sun ONE”全新提升为“Java 开发专家”系列
——源自精品 成就理想

Preface

出版说明

★ 从“开发专家之 Sun ONE”到“Java 开发专家”

“开发专家之 Sun ONE”系列丛书从诞生之日至今，已经四岁了。在这个系列里面，我们一直努力体现着这么一个理念：用一种较为敏锐的视角来跟踪 IT 技术的发展轨迹，并把可能为广大程序员所希望获得的知识，用图书出版的方式奉献给大家。

在这个系列中，我们陆续出版了约 30 种图书，有《Java 与模式》、《JSP 应用开发详解（第二版）》、《精通 EJB（第三版）》、《Tomcat 与 Java Web 应用开发详解》、《精通 Struts：基于 MVC 的 Java Web 设计与开发》、《JBoss 管理与开发核心技术（第三版）》、《精通 Spring》、《精通 Hibernate：Java 对象持久化技术详解》等一大批读者朋友耳熟能详的作品。很多作品都是在国内没有同类图书的情况下出版的。在这几年的出版工作中，我们时刻感受着市场的风险，也时刻收获着无数读者给我们的认可。

在这个系列中，凝聚了大量资深技术专家的心血。有大家都熟知的阎宏、刘晓华、孙卫琴、罗时飞等，还有一些正在不断腾跃的开发高手。这些非常优秀的国内原创作者们一直都在支持着“开发专家之 Sun ONE”系列的出版工作，在这里，我们要向他们说声：谢谢。

桃李不言，下自成蹊。由于这些年“开发专家之 Sun ONE”在“两个效益”中的杰出表现，电子工业出版社授予这个系列“最佳品牌奖”。

时代不断前进，技术不断变革。为了顺应 Java 领域的技术发展态势，为了赋予这个经典的图书系列更强的生命力，我们将“开发专家之 Sun ONE”升级为“Java 开发专家”。我们将继承原有的出版理念，紧密跟踪技术热点和发展趋势，会聚更多优秀作者，全力奉献更经典的作品。

★ 规划你的 Java 开发之路

喜马拉雅山脉的最高峰不断地在温室效应中降低，而 Java 世界的颠峰永远都在技术人员的追求中不断升高。每个人都有不同的路，每个人都有不同的行路方式，不过，往往“到了山顶才发现，错误的路和正确的路就差那么几步！”

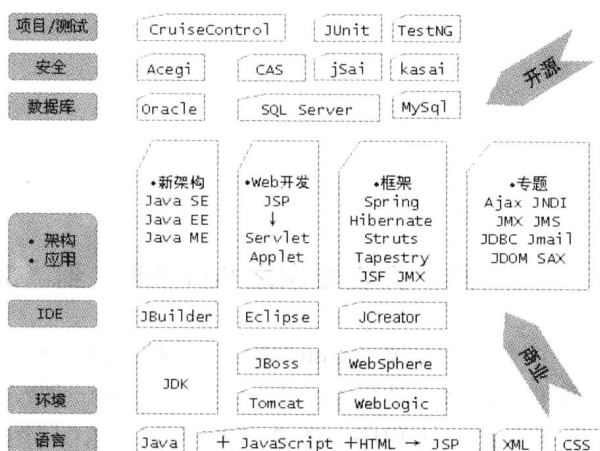
身处 Java 洪流中的程序员最累（不过大家都说 Java 程序员薪水最高，呵呵），我们简单

整理了一下 Java 领域的相关技术、工具、架构，如下图所示。这个框图中的每一个英文单词（或缩写）都可以写成一本书。Java 领域还有一个特点，那就是商业产品和开源产品层出不穷，潮流不断。相比于其他领域，如.NET，Java 开发更是体现了这句谚语：条条大路通罗马。

罗马只有一个，大路却有多条。看上去，似乎到罗马很容易，反正路多嘛。不过，路多却容易迷失方向。当你在 Java 领域中摸爬滚打几年后，发现自己在无数条道路上走了很久，却不知道罗马何日才能到达，甚至连罗马的方向都不知道，这时你肯定会很失落。

很遗憾，在这个简短的出版说明文章里面，我们无法告诉你每一条连贯的、不费周折的通往罗马的道路该如何走。或许，通过“Java 开发专家”系列中的某本书，你可以找到属于你的正确道路。在一般情况下，我们不会就某一项很窄的话题来单独写一本书，我们还是希望通过我们的一些专业和智慧，尽力把一些相关技术整合起来，用较为简明的方式表达出来，最后由你来选择。

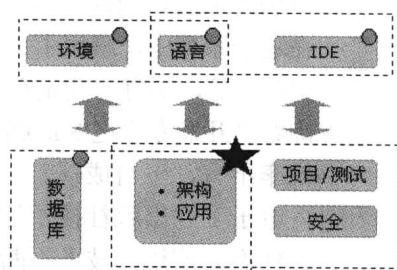
这里有句话与大家共勉：少走弯路，就是捷径！



★ “Java 开发专家”的奉献

犹如在上面那个框图中展现的那样，我们希望在各个层面、各个方向上都能给读者奉献出优秀的图书作品，全面体现技术与应用的结合。从宏观上看，我们会从语言、IDE、环境、数据库、架构与应用、安全、项目与测试等方面进行选择，选出一些读者迫切需要的技术来先行规划。

“Java 开发专家”虽然新禧初绽，但因其源自盛放的“开发专家之 Sun ONE”系列而根基稳健，两个系列会有一段很长的并行时间，我们会用一种优化的方式来保证读者的顺利选择。无论哪一个系列，必定都有大家喜欢的图书。



在技术上，有着持久化的方法，在学习上，也需要有持久化的精神。

从“开发专家之 Sun ONE”到“Java 开发专家”，希望可以带给你持久化的动力。

联系方式

咨询电话：(010) 68134545 88254160

电子邮件：support@fecit.com.cn

服务网址：http://www.fecit.com.cn http://www.fecit.net

通用网址：计算机图书、飞思、飞思教育、飞思科技、FECIT

舞动 Spring 2.5

对于 Java EE 社区而言，刚过去的 5 年是不平凡的，而其中的主角正是 Spring，它颠覆了传统的 J2EE 开发模型。当然，同 Spring 并肩作战的还有 Tapestry、Hibernate 等，它们都是敏捷 POJO 开发模型的推崇者，它们的成功推动了 Java EE 5 平台规范顺利推出，这一版本的平台规范向敏捷开发吹起了冲锋号。

现在，Spring 已经发展到 2.5 版本。在这之前，陆陆续续出现了 1.0、1.1、1.2、2.0 等版本。其中，控制反转容器和 AOP 技术实现始终是 Spring 框架的元框架(meta-framework)，而 DAO 层集成技术、Java EE 服务及技术、Web 层支持等都是构建在这一元框架基础之上的。这些内容统一构成了 Spring (<http://www.springframework.org>)。大量的企业应用、产品已经离不开 Spring 的身影，而我们再也不能够将它看成单纯的框架。如今，Spring 已经成为了“平台”的代名词。所有官方 Spring 子项目都架构在 Spring 框架之上，比如 Spring Security、Spring Batch、Spring Web Flow、Spring LDAP、Spring Web Services 等。这些子项目已经囊括了企业应用开发的各种技术栈。

展望未来，相信接下来的 5 年会更加精彩。OSGi 已经将触角伸向了 Java EE 平台，借助 OSGi 能够解决现有 Java EE 平台缺乏真正模块化、动态能力的弊端。特别地，Spring DM 子项目 (<http://www.springframework.org/osgi>) 已经针对 OSGi 提供了全面的抽象和集成支持。更有甚者，基于 OSGi 和 Spring DM 的 Java EE 应用服务器已经被成功研发出来，即 SpringSource dm Server。SpringSource dm Server 支持的开发和部署模型非常广泛，这其中包括原生 OSGi Bundle 的支持。毫无疑问，无论是 Spring DM，还是 SpringSource dm Server，这一切都离不开 Spring 本身的发展。

此时此刻，作者要感谢出版社、感谢读者的广大支持，没有你们的支持，本书的写作和出版不会这么顺利，再次真诚地谢谢你们。最后，不得不提的是，感谢妻子王女士一直以来对我写作的支持。

在《精通 Spring 2.0》（第 2 版）基础上，本次（第 3 版）的写作内容全面提升到 Spring 2.5。除了跟进 Spring 2.5 外，全书的组织、写作手法也是本版工作的重中之重，比如更加突出实战、代码组织更人性化、更好的阅读体验等。另外，由于 Spring 2.5 及 Java EE 5 涉及的知识面非常广，内容很广泛，本书错误之处在所难免，希望读者、同行批评指正。

祝您踏上愉快的阅读旅程！

罗时飞
2008 年于广州

网站介绍

关于 www.open-v.com

我们专注于 Java EE 平台、敏捷方法及开源技术咨询工作。图书创作能够将我们的咨询经验带给整个社区。有特色的技术培训能够缩短开发者、开发团队采纳敏捷技术、方法的周期。有价值的咨询服务也是我们的重点，我们非常注重咨询的效果，从而真正为企业带来实实在在的价值。

毫不夸张地说, Spring 2.5 是一套有关 Java EE API 的百科全书, 它针对各种 Java EE API 的使用都提供了一流的、一致的抽象和集成工作, 从而统一 Java EE API 暴露给开发者的客户视图。开发者都知道, Java EE API 的使用非常烦琐, 许多与业务无关的技术细节需要开发者悉心打理。稍有不慎, 各种 Java EE 问题随之而来, 而 Spring 2.5 正是为解决 Java EE 编程模型中的这些问题而出现的。

为完成各种 Java EE API 的集成工作, Spring 开发团队提供了 Spring 元框架, 即控制反转容器 (IoC) 和 AOP 技术实现。所有的 Java EE API 集成工作都是在这元框架基础上构建的。从目前来看, Spring 2.5 主要提供了三方面的 Java EE API 集成: DAO 层集成技术; Java EE 服务及技术; Web 层支持。

本书正是围绕 Spring 2.5 中的上述各项内容而准备的。

本书特点

时隔两年后,《精通 Spring 2.0》(第二版)成功写作完成,并出版发行。同《精通 Spring》(第一版)相比,本次改进、新增的内容非常多,下面总结了本书的特点。

- 全面跟进 Spring 2.5。
- 尽量将 Spring 最实用的、动人的一面展现给读者。
- 在写作过程中,理论与实践知识并重。事实上, Spring 2.5 为那些打算涉足 Java EE 开发领域的开发者创造了条件,因为 Spring 降低了 Java EE 平台技术的学习曲线。一旦开发者初步熟悉 Spring 后,再深入到各 Java EE API 也是不错的选择。本书在介绍 Java EE API 集成工作前,对它们的背景和基础知识进行了详尽阐述。与此同时,各章内容采用的示例都是单独的自成一体的经典 Eclipse 项目。
- 在代码示例的选材上,力求经典和权威。Spring 2.5 内置了展示 Spring 特性的各种示例,比如, petclinic、jpetstore、imagedb, 本书在结合它们阐述各 Spring 知识点的过程中,不时修改和扩展,甚至新增了基于不同技术栈的示例实现,比如,实现了 Hibernate 版本的 imagedb 等。这些示例的升值空间很大,因为 Spring 开发团队在不断完善它们,它们也展现了 Spring 的最新特性。现在,开发者可以一劳永逸地享受

到这些示例带来的快乐。

- 无论知识体系，还是写作风格，各章内容统一、自成一体，开发者阅读起来非常舒服。
- 作者尽量将自身架构和开发大型 Java EE/Spring 使能项目的经验、进行 Java EE 咨询期间获得的 Spring 高级技巧和最佳实践体现在书中。
- 不断改进图书内容。

服 务 网 站

针对书中展示的各种 Java 代码、Ant build.xml 和其他脚本，我们特别提供了相关网站 (<http://www.open-v.com>)。同时，为保证图书同 Spring 2.5 最新发布版的同步，我们会时常更新图书中的源代码，并公布到这一网站中，欢迎广大读者下载使用。

编 著 者

联系方式

咨询电话：(010) 68134545 88254160

电子邮件：support@fecit.com.cn

服务网址：<http://www.fecit.com.cn> <http://www.fecit.net>

通用网址：计算机图书、飞思、飞思教育、飞思科技、FECIT

第 1 篇 综述

第 1 章	Java EE 5	3
1.1	Java EE 5 引入的新特性.....	3
1.2	进入 EJB 3.0 时代	6
1.3	Java EE 开发模型的局限性	8
1.4	小结	10
第 2 章	步入 Spring 2.5.....	11
2.1	挑战 Java EE 5 开发模型	11
2.1.1	轻量级开发模型	12
2.1.2	倡导敏捷开发	15
2.1.3	Spring 2.5 的架构价值	16
2.2	有所为和有所不为	18
2.2.1	Spring 2.5 提供的功能	18
2.2.2	排除在外	20
2.3	Spring 2.5 时代的到来	21
2.4	小结	22
第 3 章	获得 Spring 2.5 发布版和源码..	23
3.1	获得 Spring 2.5 持续发布版.....	23
3.2	获得持续更新的 Spring 2.5 项目源码.....	25
3.3	小结	26

第 4 章	启动 Spring 2.5 使能项目	27
4.1	开发平台的搭建.....	27
4.1.1	JDK 的安装及设置	27
4.1.2	选用 Eclipse IDE 和 WTP.....	28
4.1.3	借助插件调试 Web 应用	28
4.1.4	获取及安装 Spring IDE	30
4.2	Spring IDE 的使用	30
4.3	小结	34

第 2 篇 Spring 2.5 核心技术

第 5 章	控制反转容器	37
5.1	有关 DI 容器的背景知识.....	37
5.2	BeanFactory 和 ApplicationContext	38
5.3	宿主 DI 容器配置元数据的 不同方式	39
5.3.1	基于 XML 的 DI 容器 配置元数据	39
5.3.2	基于注解的 DI 容器 配置元数据	42
5.4	基于泛型访问 DI 容器	43
5.5	支持的不同依赖注入类型	44

5.5.1	设值注入	44
5.5.2	构建器注入	45
5.5.3	属性注入	47
5.5.4	方法注入	48
5.6	Autowiring 策略	52
5.6.1	autowire 属性	52
5.6.2	<bean/>元素的 dependency-check 属性	54
5.6.3	@Required 注解	55
5.6.4	@Autowired 注解	56
5.6.5	细粒度控制 Autowiring 策略	58
5.6.6	借用<context:annotation- config/>元素	61
5.7	善待 depends-on 属性	61
5.8	抽象和子 Bean 定义	63
5.9	别名 (Alias)	64
5.10	外在化应用参数的配置	65
5.10.1	<context:property- placeholder/>元素	65
5.10.2	<context:property-override/> 元素	66
5.11	受管 Bean 的作用范围	67
5.11.1	单例和原型	67
5.11.2	仅仅适合于 Web 环境的 三种作用范围	69
5.12	在 Web 应用中使用 DI 容器	72
5.12.1	往 Web 应用中加载 DI 容器	73
5.12.2	复合多个配置文件	74
5.12.3	于 Web 应用中操控 DI 容器	75
5.12.4	国际化和本地化 消息资源	76
5.13	探索<util/>命名空间	78
5.13.1	<util:constant/>元素	78
5.13.2	<util:property-path/>元素	79
5.13.3	<util:properties/>元素	80
5.13.4	<util:list/>元素	81

5.13.5	<util:map/>元素	82
5.13.6	<util:set/>元素	84
5.14	使用<p/>命名空间	85
5.15	操控资源	86
5.15.1	内置的 Resource 继承链	86
5.15.2	借助 DI 容器访问 各种资源	88
5.15.3	妙用 classpath*前缀	89
5.16	回调接口集合及其触发 顺序	89
5.16.1	BeanNameAware 回调 接口	89
5.16.2	BeanClassLoaderAware 回调接口	90
5.16.3	BeanFactoryAware 回调接口	90
5.16.4	ResourceLoaderAware 回调接口	91
5.16.5	ApplicationEventPublisherAware 回调接口	91
5.16.6	MessageSourceAware 回调接口	91
5.16.7	ApplicationContextAware 回调接口	92
5.16.8	@PostConstruct 注解	92
5.16.9	InitializingBean 回调接口	92
5.16.10	<bean/>元素的 init-method 属性	92
5.16.11	@PreDestroy 注解	93
5.16.12	DisposableBean 回调接口	93
5.16.13	<bean/>元素的 destroy- method 属性	93
5.17	小结	94
第 6 章	面向切面编程	95
6.1	AOP 背景知识	96
6.2	AspectJ 6 介绍	97
6.2.1	AspectJ 的安装及使用	98
6.2.2	Before 装备	104
6.2.3	AfterReturning 装备	106
6.2.4	AfterThrowing 装备	108

6.2.5	After 装备	110
6.2.6	Around 装备	110
6.2.7	引入 (Introduction)	112
6.3	Spring AOP 的基本概念	115
6.4	Spring AOP 的老式用法	118
6.4.1	Before 装备	119
6.4.2	基于 ProxyFactoryBean 的 手工代理	121
6.4.3	AfterReturning 装备	123
6.4.4	AfterThrowing 装备	125
6.4.5	Around 装备	126
6.4.6	Introduction 引入	129
6.4.7	使用自动代理特性	129
6.4.8	切换代理机制	131
6.4.9	基于 ProxyFactory 的 编程代理	132
6.5	基于 @AspectJ 的 Spring AOP	132
6.5.1	声明切面、pointcut 和 装备	133
6.5.2	各种装备的使用	136
6.5.3	切换代理机制	139
6.5.4	控制各装备的触发顺序	139
6.5.5	pointcut 表达语言	140
6.6	基于 <aop:config/> 元素的 AOP	144
6.6.1	声明切面、pointcut 和装备 ..	145
6.6.2	各种装备的使用	146
6.6.3	<aop:advisor/> 元素	150
6.6.4	切换代理机制	150
6.7	借用 AspectJ 6 进行领域 对象的 DI 操作	151
6.7.1	直接使用 AnnotationBean- ConfigurerAspect 切面	151
6.7.2	@Configurable 注解	154
6.7.3	借助 aop.xml 控制启用的 特定切面	156
6.7.4	<aop:include/> 元素	156
6.7.5	<context:spring- configured/> 元素	157

6.7.6	借用 <context:load-time- weaver/> 元素	157
6.8	小结	158

第 3 篇 DAO 层集成技术

第 7 章	DAO 抽象支持	163
7.1	背景	163
7.2	DAO 集成支持	165
7.2.1	DataAccessException 异常体系	166
7.2.2	DaoSupport 继承链	167
7.2.3	DataAccessUtils 实用类	168
7.3	小结	169
第 8 章	JDBC 集成	171
8.1	背景知识及示例	171
8.2	Spring 对 JDBC 提供的 支持	176
8.3	运行 JDBC 版 PetClinic 实例	177
8.4	JdbcTemplate 及相应的 支持类	178
8.4.1	JdbcTemplate 核心类	179
8.4.2	JdbcDaoSupport 支持类	187
8.5	NamedParameterJdbcTemplate 及 相应的支持类	188
8.5.1	NamedParameterJdbcTemplate 模板类	188
8.5.2	NamedParameterJdbcDao- Support 支持类	190
8.6	SimpleJdbcTemplate 及相应的 支持类	191
8.6.1	SimpleJdbcTemplate 模板类	191
8.6.2	SimpleJdbcDaoSupport 支持类	192
8.6.3	SimpleJdbcInsert 辅助类	193
8.6.4	基于 JDBC 的 PetClinic 综合示例分析	195
8.7	内置的 DataSource 继承链	197

8.7.1	用于测试目的的 DriverManagerDataSource.....	198
8.7.2	用于测试目的的 SimpleDriverDataSource.....	198
8.7.3	用于测试目的 Single- ConnectionDataSource	198
8.7.4	Apache DBCP 数据源	199
8.7.5	Java EE 容器内置的 数据源	199
8.7.6	LazyConnectionDataSource- Proxy 数据源	199
8.7.7	TransactionAwareDataSource- Proxy 数据源	200
8.7.8	UserCredentialsDataSource- Adapter 数据源	201
8.7.9	IsolationLevelDataSource- Adapter 数据源	202
8.7.10	WebSphereDataSource- Adapter 数据源	202
8.7.11	IsolationLevelDataSource- Router 数据源	202
8.8	将 JDBC 操作建模成 Java 对象	203
8.8.1	SqlUpdate 辅助类	204
8.8.2	UpdatableSqlQuery 辅助类	205
8.8.3	MappingSqlQuery 辅助类	206
8.8.4	SqlFunction 辅助类	206
8.9	与存储过程交互	207
8.9.1	JdbcTemplate 针对存储 过程提供的支持	208
8.9.2	StoredProcedure 辅助类	209
8.9.3	SimpleJdbcCall 辅助类	209
8.10	处理大批量数据	210
8.10.1	JdbcTemplate 内置的 batchUpdate()方法	210
8.10.2	SimpleJdbcTemplate 内置的 batchUpdate()方法	213
8.10.3	BatchSqlUpdate 辅助类	214
8.11	基于 JDBC 的 LOB 集成支持	216

8.11.1	运行及分析 imagedb 示例应用	216
8.11.2	NativeJdbcExtractor 继承链	217
8.11.3	操作 LOB 字段	218
8.12	如何获得和生成主键	221
8.12.1	KeyHolder 及 GeneratedKey- Holder 实现者	221
8.12.2	DataFieldMax Value- Incrementer 继承链	222
8.13	对行集的支持	224
8.13.1	JdbcTemplate 内置的 queryForRowSet()方法 集合	224
8.13.2	NamedParameterJdbcTemplate 内置的 queryForRowSet() 方法集合	224
8.14	JDBC 最佳实践	225
8.15	小结	226
第 9 章	事务集成	227
9.1	背景知识及示例	228
9.2	Spring 对事务提供的支持	230
9.3	Spring 眼中的事务 管理策略	231
9.3.1	事务定义	231
9.3.2	各种 PlatformTransaction- Manager 实现	236
9.4	编程式事务	238
9.4.1	TransactionTemplate 及 相关回调接口	238
9.4.2	使用@Transactional 注解和 <tx:annotation-driven/>元素	241
9.4.3	拥抱 EJB 3.0 引入的 @TransactionAttribute 注解	245
9.5	声明式事务	245
9.5.1	TransactionProxyFactoryBean 辅助类	245
9.5.2	<tx:advice/>元素	247
9.6	在 AspectJ 6 应用中使用 @Transactional	248

9.6.1	直接使用 Annotation-TransactionAspect 切面	248	10.4.4	AbstractJUnit4Spring-ContextTests 支持类	277
9.6.2	借用<tx:annotation-driven/>元素	251	10.4.5	AbstractTransactionalJUnit4-SpringContextTests 支持类 ...	277
9.6.3	借用<context:load-time-weaver/>元素	251	10.5	集成测试最佳实践	279
9.7	事务集成高级特性	252	10.6	小结	280
9.7.1	Java EE 应用服务器的事务集成	252	第 11 章	Hibernate 集成	281
9.7.2	<tx:jta-transaction-manager/>元素	253	11.1	背景知识及示例	281
9.7.3	选择合适的事务策略	254	11.2	Hibernate Tools 介绍	289
9.8	小结	254	11.2.1	Ant 支持	289
第 10 章	单元和集成测试	257	11.2.2	Eclipse 支持	291
10.1	背景知识及示例	257	11.3	Spring 对 Hibernate 提供的支持	293
10.2	Spring 对集成测试的支持	259	11.4	运行 Hibernate 版 PetClinic 实例	293
10.2.1	ReflectionTestUtils 实用类	259	11.5	基于 Hibernate 集成的 CRUD 操作	294
10.2.2	运行 PetClinic 中的集成测试类	260	11.5.1	HibernateTemplate 模板类	294
10.3	遗留 JUnit 3.8 集成测试支持 ...	261	11.5.2	HibernateCallback 回调接口	299
10.3.1	AbstractSingleSpringContext-Tests 支持类	261	11.5.3	关于 SessionFactory.getCurrentSession() 方法的使用	300
10.3.2	AbstractDependencyInjection-SpringContextTests 支持类	263	11.6	LocalSessionFactoryBean	301
10.3.3	AbstractTransactionalSpring-ContextTests 支持类	265	11.7	AnnotationSessionFactoryBean	303
10.3.4	AbstractTransactionalDataSourceSpringContextTests 支持类	267	11.8	事务管理支持	306
10.3.5	AbstractAnnotationAware-TransactionalTests 支持类 ...	270	11.9	基于 Hibernate 的 LOB 处理	308
10.4	新引入的 TestContext 集成测试框架	272	11.10	为 imagedb 示例启用 JTA 事务	311
10.4.1	面向开发者的支持类	272	11.11	集成测试支持	314
10.4.2	AbstractJUnit38Spring-ContextTests 支持类	274	11.11.1	分析 Hibernate 版 PetClinic 实例的集成测试工作	314
10.4.3	AbstractTransactionalJUnit38-SpringContextTests 支持类 ...	275	11.11.2	混合使用 JDBC 和 Hibernate	315
			11.12	小结	320

第 12 章	Java 持久化 API 集成	321
12.1	背景知识及示例	321
12.2	Spring 对 JPA 提供的支持	324
12.3	基于 JPA 集成的 CRUD 操作	325
12.3.1	JpaTemplate 模板类	325
12.3.2	JpaCallbck 回调接口	326
12.3.3	@PersistenceContext 注解	327
12.4	AbstractEntityManager-FactoryBean 继承链	329
12.4.1	LocalEntityManager-FactoryBean 辅助类	329
12.4.2	LocalContainerEntity-ManagerFactoryBean 辅助类	331
12.4.3	DataSourceLookup 继承链	333
12.5	事务管理支持	335
12.6	装载期织入 (LTW)	337
12.6.1	ReflectiveLoadTimeWeaver 实现类	338
12.6.2	InstrumentationLoad-TimeWeaver 实现类	339
12.6.3	LoadTimeWeaver 继承链在 JPA 集成中的应用	340
12.7	SharedEntityManagerBean	341
12.8	集成测试支持	341
12.8.1	AbstractJpaTests 支持类	341
12.8.2	AbstractAspectjJpaTests 支持类	342
12.8.3	混合使用 JDBC 和 JPA	344
12.9	小结	348

第 4 篇 集成 Java EE 服务及技术

第 13 章	JNDI 集成	351
13.1	背景知识及示例	351
13.2	Spring 对 JNDI 提供的支持	354
13.3	JndiObjectFactoryBean	354
13.4	<jee:jndi-lookup/>元素	358

13.5	JndiTemplate 和 JndiCallback 的使用	359
13.6	小结	360
第 14 章	EJB 3.0 集成	361
14.1	背景知识及示例	361
14.2	Spring 对开发 EJB 3.0 组件提供的支持	363
14.3	Spring 对访问 EJB 3.0 组件提供的支持	365
14.3.1	借助 JndiObjectFactory-Bean 辅助类	365
14.3.2	org.springframework.ejb.access 包	366
14.3.3	<jee:remote-slsb/>元素和 <jee:local-slsb/>元素	367
14.4	关于遗留 EJB 2.x 支持	368
14.5	小结	368
第 15 章	线程池和任务调度集成	369
15.1	Spring 提供的线程池支持	369
15.1.1	SyncTaskExecutor 执行器	371
15.1.2	SimpleAsyncTaskExecutor 执行器	371
15.1.3	ThreadPoolTaskExecutor 和 ConcurrentTaskExecutor 执行器	371
15.1.4	TimerTaskExecutor 执行器	372
15.1.5	SimpleThreadPool-TaskExecutor 执行器	372
15.1.6	commonj.WorkManager-TaskExecutor 执行器	372
15.1.7	jca.work.WorkManager-TaskExecutor 继承链	373
15.2	Spring 提供的任务调度支持	373
15.2.1	针对 java.util.Timer 的任务调度支持	373
15.2.2	针对 Quartz 的任务调度支持	375

15.2.3	针对 java.util.concurrent 的 任务调度支持.....	380
15.2.4	针对 CommonJ 的任务 调度支持.....	381
15.3	小结.....	382
第 16 章	Java 消息服务集成.....	383
16.1	背景知识及示例.....	384
16.2	Spring 对 JMS 消息 提供的支持.....	386
16.3	借助 JmsTemplate 发送 JMS 消息.....	386
16.4	同步和异步消费 JMS 消息.....	393
16.4.1	借助 JmsTemplate 同步 接收 JMS 消息.....	393
16.4.2	AbstractMessageListener- Container 容器.....	394
16.4.3	<jms:listener-container/> 元素.....	400
16.5	JMS 事务管理.....	400
16.6	小结.....	403
第 17 章	JavaMail 集成.....	405
17.1	背景知识及示例.....	405
17.2	Spring 对 JavaMail 提供的支持.....	407
17.3	发送简单邮件.....	408
17.4	发送含有附件的邮件.....	411
17.5	发送含有 HTML 和 内嵌资源的邮件.....	413
17.6	小结.....	414
第 18 章	远程服务集成.....	415
18.1	远程服务背景知识及示例.....	416
18.2	Spring 对远程服务 提供的支持.....	419
18.3	RMI/RMI-IIOP 集成.....	421
18.4	Hessian 和 Burlap 集成.....	423
18.4.1	DispatcherServlet 和 HttpRequestHandlerServlet 辅助类.....	424

18.4.2	宿主在 Sun JDK 6.0 内置的 HTTP 服务器中.....	425
18.5	HTTP Invoker 支持.....	426
18.6	Web 服务集成.....	427
18.6.1	JAX-RPC 集成.....	428
18.6.2	JAX-WS 集成.....	429
18.7	基于 JMS 的远程服务.....	430
18.8	小结.....	432
第 19 章	Java 管理扩展集成.....	433
19.1	背景知识及示例.....	434
19.2	Spring 对 JMX 提供的支持.....	437
19.3	自动注册 MBean 组件.....	438
19.3.1	关于 MBeanExporter 的 autodetectMode 和 registrationBehavior 属性.....	439
19.3.2	Hibernate 暴露的 StatisticsService MBean.....	440
19.4	将 POJO 导出成 MBean 组件.....	442
19.5	控制 MBean 组件的 管理接口.....	444
19.5.1	AbstractConfigurable- MBeanInfoAssembler 继承链.....	445
19.5.2	基于注解的 Metadata- MBeanInfoAssembler.....	447
19.5.3	<context:mbean-export/>和 <context:mbean-server/> 元素.....	449
19.5.4	面向异步处理的 Lifecycle 接口.....	451
19.6	控制 MBean 组件的 ObjectName.....	453
19.6.1	KeyNamingStrategy 实现类.....	453
19.6.2	IdentityNamingStrategy 实现类.....	454
19.6.3	MetadataNamingStrategy 实现类.....	454
19.7	发送与接收 JMX 通知.....	455

19.8	通过应用访问 MBean 组件.....	458
19.9	小结.....	459
第 20 章	Java EE 连接器架构集成.....	461
20.1	背景知识及示例.....	461
20.2	Spring 对 JCA 提供的支持.....	463
20.3	CciTemplate 及相关回调接口.....	463
20.4	将 JCA 操作建模成 Java 对象.....	466
20.5	事务管理.....	468
20.6	宿主在 JCA 适配器中的 DI 容器.....	469
20.6.1	将 SpringContextResource-Adapter 部署到 RAR 中.....	470
20.6.2	ResourceAdapterFactory-Bean 辅助类.....	471
20.7	小结.....	472

第 5 篇 Spring 2.5 高级特性

第 21 章	忘却的 Spring 高级话题.....	475
21.1	分发和监听事件.....	475
21.2	AOP 拦截器链.....	477
21.3	DataSourceUtils、Session-FactoryUtils、EntityManager-FactoryUtils.....	478
21.4	Web 层集成支持.....	480
21.5	<context:component-scan/> 元素.....	481
21.6	如何优雅地销毁 DI 容器.....	481
21.6.1	Web 应用类型.....	482
21.6.2	EJB 应用类型.....	483
21.6.3	Java SE 应用、集成测试类型.....	484
21.7	DI 容器的分层管理.....	485
21.8	脚本集成.....	488
21.9	小结.....	490

第 22 章	Spring 最佳实践.....	491
22.1	注重分层架构设计.....	491
22.2	合理采纳注解技术.....	492
22.3	日志管理策略.....	492
22.4	善待 Java EE 容器内置的类装载器.....	495
22.5	逐步采纳 Spring 2.5.....	496
22.6	小结.....	498

附录 A	基于 XML Schema 的权威配置指南.....	499
A.1	XML 配置文件.....	499
A.2	<beans/>命名空间.....	501
A.2.1	<import/>元素.....	501
A.2.2	<alias/>元素.....	501
A.2.3	<bean/>元素.....	502
A.2.4	<beans/>元素.....	505
A.3	<util/>命名空间.....	506
A.3.1	<util:constant/>元素.....	506
A.3.2	<util:property-path/>元素.....	506
A.3.3	<util:properties/>元素.....	507
A.3.4	<util:list/>元素.....	507
A.3.5	<util:map/>元素.....	507
A.3.6	<util:set/>元素.....	508
A.4	<context/>命名空间.....	508
A.4.1	<context:property-placeholder/>元素.....	508
A.4.2	<context:property-override/>元素.....	509
A.4.3	<context:annotation-config/>元素.....	509
A.4.4	<context:component-scan/>元素.....	509
A.4.5	<context:load-time-weaver/>元素.....	509
A.4.6	<context:spring-configured/>元素.....	510
A.4.7	<context:mbean-export/>元素.....	510
A.4.8	<context:mbean-server/>元素.....	510