

◇ 了解Spring 5的新特性 ◇ 精通数据访问和事务处理 ◇ 创建微服务和其他Web服务
◇ 使用Spring Testing、Spring Remoting、WebSocket等新的功能性Web框架



Pro Spring 5, Fifth Edition

Spring 5 高级编程

(第5版)

尤莉安娜·科斯米纳(Iuliana Cosmina)
[美] 罗布·哈罗普(Rob Harrop) 著
克里斯·舍弗(Chris Schaefer)
克拉伦斯·厚(Clarence Ho) 译
王 净

Apress®



清华大学出版社

Spring 5 高级编程

(第 5 版)

[美] 尤莉安娜·科斯米纳(Iuliana Cosmina)
罗布·哈罗普(Rob Harrop) 著
克里斯·舍弗(Chris Schaefer)
克拉伦斯·厚(Clarence Ho)
王 净 译



清华大学出版社

北 京

Pro Spring 5, Fifth Edition

Iuliana Cosmina, Rob Harrop, Chris Schaefer, Clarence Ho

EISBN: 978-1-4842-2807-4

Original English language edition published by Apress Media. Copyright ©2017 by Apress Media. Simplified Chinese-Language edition copyright ©2019 by Tsinghua University Press. All rights reserved.

本书中文简体字版由 Apress 出版公司授权清华大学出版社出版。未经出版者书面许可，不得以任何方式复制或抄袭本书内容。

北京市版权局著作权合同登记号 图字：01-2018-2643

本书封面贴有清华大学出版社防伪标签，无标签者不得销售。

版权所有，侵权必究。侵权举报电话：010-62782989 13701121933

图书在版编目(CIP)数据

Spring 5高级编程：第5版 / (美)尤莉安娜·科斯米纳(Iuliana Cosmina) 等著；王净 译. —北京：清华大学出版社，2019
书名原文：Pro Spring 5, Fifth Edition

ISBN 978-7-302-51644-6

I. ①S… II. ①尤… ②王… III. JAVA 语言—程序设计—教材 IV. ①TP312.8

中国版本图书馆 CIP 数据核字(2018)第 257354 号

责任编辑：王 军

装帧设计：孔祥峰

责任校对：牛艳敏

责任印制：李红英

出版发行：清华大学出版社

网 址：<http://www.tup.com.cn>, <http://www.wqbook.com>

地 址：北京清华大学学研大厦 A 座 邮 编：100084

社 总 机：010-62770175 邮 购：010-62786544

投稿与读者服务：010-62776969, c-service@tup.tsinghua.edu.cn

质 量 反 馈：010-62772015, zhiliang@tup.tsinghua.edu.cn

印 装 者：三河市铭诚印务有限公司

经 销：全国新华书店

开 本：190mm×260mm 印 张：34.75 字 数：1279 千字

版 次：2019 年 1 月第 1 版 印 次：2019 年 1 月第 1 次印刷

定 价：128.00 元

产品编号：079785-01



译者序

Spring 是一种多层的 J2EE 应用程序框架,以 Rod Johnson 撰写的 *Expert One-on-One J2EE Design and Development* 一书的代码为基础发展而来。Spring 的核心是提供一种新的机制来管理业务对象及其依赖关系。例如,可以指定一个 DAO(数据访问对象)类依赖一个 DataSource 类。此外,还允许开发人员通过接口编程,使用 XML 文件来简单地定义其实现。Spring 拥有许多类来支持其他的框架(如 Hibernate 和 Struts),这使得集成变得易如反掌。

Spring 5 是自 2013 年 12 月的版本 4 发布之后, Spring 的第一个主发行版。Spring 项目的负责人 Juergen Hoeller 于 2016 年 7 月 28 日发布了第一个 Spring 5 里程碑版本。整个 Spring 5 代码库运行在 Java 8 上,因此 Spring 5 对环境的最低要求就是 Java 8。作为开发者的我们而言,已经能够借此享受到现代 Java 发行版中的所有新特性了。

《Spring 5 高级编程(第 5 版)》一书共 18 章,深入浅出地介绍了 Spring 5 所有的相关知识,其中重点介绍如何使用 Java 配置类、lambda 表达式、SpringBoot 以及反应式编程,此外,还与读者分享了一些实际的开发经验,包括远程处理、事务、Web 和表示层等。按照内容,大致可分为四大部分。第一部分(包括第 1~5 章)主要介绍 Spring 5 相关的基础知识,包括 Spring 框架的演变、翻转控制和依赖注入、Spring 配置以及 AOP。第二部分(第 6~12 章)主要介绍 Spring 的一些特殊功能,包括 JDBC 支持、使用 Hibernate、数据访问、事务管理、任务调度以及远程控制。第三部分(第 13~15 章)主要介绍 Spring 测试、脚本支持以及应用程序监控。第四部分(第 16~18 章)主要介绍 Spring 在网络方面的应用以及对 Spring 项目组合中的一些项目进行概述,特别是 Spring Batch、Spring Integration、Spring XD 以及 Spring 5 中增加的一些新功能。

本书适合有一定 Java 编程基础的读者阅读,适用于在 Java 平台下进行各类软件开发的开发人员、测试人员,尤其适用于企业级 Java 开发人员。既可以被刚开始学习 Spring 的读者当作学习指南,也可以被那些想深入了解 Spring 某方面功能的资深用户作为参考用书。

参与本书翻译的人有王净、田洪、范园芳、范桢、胡训强、纪红、彭洁、晏峰、余佳隽、张洁以及赵翊含,最终由王净负责统稿。此外,还要感谢我的家人,他们总是无怨无悔地支持我的一切工作,我为有这样的家庭而感到幸福。

译者在翻译过程中,尽量保持原书的特色,并对书中出现的术语和难词、难句进行了仔细推敲和研究。但毕竟有少量技术是译者在自己的研究领域中不曾遇到的,所以疏漏和争议之处在所难免,望广大读者提出宝贵意见。

最后,希望广大读者能多花些时间细细品味这本凝聚作者和译者大量心血的书籍,为将来的职业生涯奠定良好的基础。

王 净
作于广州



作者简介

Iuliana Cosmina 是一名 Spring 认证的 Web 应用程序开发人员,也是 Spring 认证的 Spring 专家(由 Pivotal 定义, Pivotal 是 Spring 框架、Spring Boot 以及其他工具的制造商)。她曾与 Apress 出版社合作出版了多本关于核心 Spring 认证和 Spring 认证 Web 开发的书籍。她是来自 Bearing Point Software 的一名软件架构师,也是 GitHub、Stack Overflow 等平台上活跃的编码者和软件贡献者。

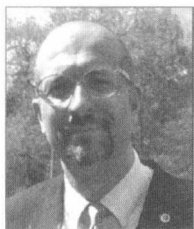
Rob Harrop 是一位软件顾问,致力于提供高性能、高度可扩展的企业级应用程序。他是一位经验丰富的架构师,对于理解和解决复杂的设计问题具有极高天赋。凭借扎实的 Java 和 .NET 开发功力,Harrop 已经成功在两种平台上部署不少项目。此外,他还在其他行业拥有丰富的经验,尤其是零售和政府领域。Harrop 共独自撰写或参与撰写了 5 本书,其中就包括本书(当然不是第 5 版),该书广受好评,包含大量关于 Spring 框架的资源。

Chris Schaefer 是 Pivotal Spring 项目的主要软件开发人员, Pivotal 是 Spring 框架、Spring Boot 以及其他 Spring 工具的制造商。

Clarence Ho 是中国香港软件咨询公司 SkywideSoft Technology Limited 的 Java 高级架构师。Clarence 在 IT 领域工作了 20 多年,一直担任许多内部应用程序开发项目的团队负责人,并为客户提供有关企业解决方案的咨询服务。



技术审校者简介



Massimo Nardone 在安全、Web/移动开发、云计算和 IT 架构方面拥有超过 23 年的经验，但他真正感兴趣的是安全和 Android。

他目前担任 Cargotec Oyj 的首席信息安全官(CISO)，并且是 ISACA 芬兰分会董事会成员。在他长期的职业生涯中，曾担任过以下职位：项目经理、软件工程师、研究工程师、首席安全架构师、信息安全经理、PCI/SCADA 审计师以及高级 IT 安全/云/ SCADA 架构师。此外，他还是赫尔辛基技术大学网络实验室的客座讲师和主管。

Massimo 拥有意大利萨勒诺大学的计算科学硕士学位，拥有 4 项国际专利(PKI、SIP、SAML 和代理领域)。除审校本书外，Massimo 还为多家出版公司审校了 40 多本 IT 书籍，并且是 *Pro Android Games*(Apress 出版社于 2015 年出版)一书的合著者。



致 谢

作为本书第 5 版的主要作者，我感到非常荣幸。一开始，我以为是要审校本书，直到后来，才意识到自己要主笔，激动不已。

在求学甚至以后的工作过程中，我读过 Apress 出版的许多书，不断提高自己的专业水平。本书是我与 Apress 合作出版的第三本书，能为培养下一代开发人员做出贡献是一件好事。

我十分感谢那些耐心倾听我抱怨失眠、工作太多、写作思路不畅的朋友。谢谢大家的支持，是你们让我在编写这本书时感到一些乐趣。

另外，我还要感谢我最喜爱的所有歌手，是他们的美妙音乐让我更轻松愉快地完成工作，尤其是 John Mayer；我之所以决心按时完成这本书，是为了可以去美国参加他的一场音乐会。这也是为什么我将书中的示例主题更改为这些歌手及其音乐的原因，这是对他们艺术和才能的致敬。

—Iuliana Cosmina



前言

本书涵盖 Spring 5 的所有内容，如果想要充分利用这一领先的企业级 Java 应用程序开发框架的强大功能，本书是最全面的 Spring 参考和实用指南。

本书第 5 版涵盖核心的 Spring 及其与其他领先的 Java 技术(比如 Hibernate、JPA 2、Tiles、Thymeleaf 和 WebSocket)的集成。本书的重点是介绍如何使用 Java 配置类、lambda 表达式、Spring Boot 以及反应式编程。同时，将与企业级应用程序开发人员分享一些见解和实际经验，包括远程处理、事务、Web 和表示层，等等。

通过本书，你可以学习如何完成以下事情：

- 使用控制反转(IOC)和依赖注入(DI)。
- 了解 Spring 5 中的新功能。
- 使用 Spring MVC 和 WebSocket 构建基于 Spring 的 Web 应用程序。
- 使用 Spring WebFlux 构建 Spring Web 反应式应用程序。
- 使用 JUnit 5 测试 Spring 应用程序。
- 使用新的 Java 8 lambda 语法。
- 使用 Spring Boot 达到更高的水平，以获取任何类型的 Spring 应用程序并立即运行。
- 在 Spring 应用程序中使用 Java 9 的新功能。

由于 Java 9 的发布日期不断推迟，因此 Spring 5 是基于 Java 8 发布的。书中介绍的与 Java 9 的互操作性都是基于早期的可访问版本完成的。

有一个与本书相关的多模块项目，使用 Gradle 4 进行配置。该项目位于 Apress 的官方存储库中，参见 <https://github.com/Apress/pro-spring-5>。只要在本机上安装了 Gradle，根据 README.adoc 文件中的说明即可在克隆后立即构建该项目。如果尚未安装 Gradle，那么可以使用 IntelliJ IDEA 下载并使用 Gradle Wrapper 来构建自己的项目 (https://docs.gradle.org/current/userguide/gradle_wrapper.html)。本书末尾的附录 A 介绍了项目结构、配置以及可用于开发和运行本书代码示例的开发工具的其他详细信息，这些工具可在 GitHub 上找到。

在编写本书的过程中，Spring 5 新的候选版本发布了，还发布了新版本的 IntelliJ IDEA，同时书中使用的 Gradle 以及其他技术的版本也更新了。因此，我们及时升级到新版本，以便提供最新信息，并使本书与官方文档保持同步。多位专家已经对本书的技术准确性进行了审查，但是如果发现任何不一致的地方，请发送电子邮件至 editorial@apress.com 并创建勘误项。

可以通过 www.apress.com/9781484228074 网页上的 Download Source Code 按钮访问本书的示例源代码。我们将对这些代码进行及时维护，以便与新技术保持同步，同时根据开发人员在学习 Spring 的过程中提出的建议不断进行丰富。

真心希望你喜欢用本书来学习 Spring，就像我们喜欢编写本书一样。

目 录

第1章 Spring 介绍.....	1	第3章 在 Spring 中引入 IoC 和 DI.....	23
1.1 什么是 Spring.....	1	3.1 控制反转和依赖注入.....	23
1.1.1 Spring 框架的演变.....	1	3.2 控制反转的类型.....	23
1.1.2 翻转控制或依赖注入.....	5	3.2.1 依赖抽取.....	24
1.1.3 依赖注入的演变.....	6	3.2.2 上下文依赖查找.....	24
1.1.4 除了依赖注入.....	7	3.2.3 构造函数依赖注入.....	25
1.2 Spring 项目.....	10	3.2.4 setter 依赖注入.....	25
1.2.1 Spring 的起源.....	10	3.2.5 注入与查找.....	25
1.2.2 Spring 社区.....	10	3.2.6 setter 注入与构造函数注入.....	26
1.2.3 Spring 工具套件.....	11	3.3 Spring 中的控制反转.....	28
1.2.4 Spring Security 项目.....	11	3.4 Spring 中的依赖注入.....	28
1.2.5 Spring Boot.....	11	3.4.1 bean 和 BeanFactory.....	28
1.2.6 Spring 批处理和集成.....	11	3.4.2 BeanFactory 实现.....	29
1.2.7 许多其他项目.....	11	3.4.3 ApplicationContext.....	30
1.3 Spring 的替代品.....	12	3.5 配置 ApplicationContext.....	30
1.3.1 JBoss Seam 框架.....	12	3.5.1 设置 Spring 配置选项.....	30
1.3.2 Google Guice.....	12	3.5.2 基本配置概述.....	30
1.3.3 PicoContainer.....	12	3.5.3 声明 Spring 组件.....	31
1.3.4 JEE 7 容器.....	12	3.5.4 使用方法注入.....	53
1.4 小结.....	12	3.5.5 了解 bean 命名.....	60
第2章 入门.....	13	3.5.6 了解 bean 实例化模式.....	66
2.1 获取 Spring 框架.....	13	3.6 解析依赖项.....	69
2.1.1 快速入门.....	13	3.7 自动装配 bean.....	71
2.1.2 在 GitHub 中查找 Spring.....	14	3.8 设置 bean 继承.....	77
2.1.3 使用正确的 JDK.....	14	3.9 小结.....	79
2.2 了解 Spring 打包.....	14	第4章 详述 Spring 配置和 Spring Boot.....	80
2.2.1 为自己的应用程序选择模块.....	15	4.1 Spring 对应用程序可移植性的影响.....	80
2.2.2 在 Maven 存储库上访问 Spring 模块.....	15	4.2 管理 bean 生命周期.....	81
2.2.3 使用 Gradle 访问 Spring 模块.....	16	4.3 挂钩到 bean 的创建.....	82
2.2.4 使用 Spring 文档.....	17	4.3.1 在创建 bean 时执行方法.....	82
2.2.5 将 Spring 放入 Hello World 中.....	17	4.3.2 实现 InitializingBean 接口.....	84
2.2.6 构建示例 Hello World 应用程序.....	17	4.3.3 使用 JSR-250 @PostConstruct 注解.....	86
2.2.7 用 Spring 重构.....	20	4.4 使用 @Bean 声明一个初始化方法.....	88
2.3 小结.....	22	4.5 挂钩到 bean 的销毁.....	89
		4.5.1 在 bean 被销毁时执行一个方法.....	89

4.5.2 实现 DisposableBean 接口	91	5.4.6 创建前置通知	141
4.5.3 使用 JSR-250 @PreDestroy 注解	92	5.4.7 通过使用前置通知保护方法访问	142
4.6 使用 @Bean 声明销毁方法	93	5.4.8 创建后置返回通知	145
4.7 了解解析的顺序	94	5.4.9 创建环绕通知	147
4.8 让 Spring 感知 bean	94	5.4.10 创建异常通知	148
4.8.1 使用 BeanNameAware 接口	95	5.4.11 选择通知类型	150
4.8.2 使用 ApplicationContextAware 接口	96	5.5 在 Spring 中使用顾问和切入点	150
4.9 使用 FactoryBean	97	5.5.1 Pointcut 接口	151
4.10 直接访问 FactoryBean	100	5.5.2 可用的切入点实现	152
4.11 使用 factory-bean 和 factory-method 属性	101	5.5.3 使用 DefaultPointcutAdvisor	152
4.12 JavaBean PropertyEditor	102	5.5.4 使用 StaticMethodMatcherPointcut	
4.12.1 使用内置的 PropertyEditor	102	创建静态切入点	153
4.12.2 创建自定义 PropertyEditor	106	5.5.5 使用 DyanmicMethodMatcherPointcut	
4.13 更多的 Spring ApplicationContext 配置	108	创建动态切入点	155
4.13.1 使用 MessageSource 进行国际化	108	5.5.6 使用简单名称匹配	157
4.13.2 在独立的应用程序中使用		5.5.7 用正则表达式创建切入点	158
MessageSource	110	5.5.8 使用 AspectJ 切入点表达式创建切入点	159
4.13.3 应用程序事件	111	5.5.9 创建注解匹配切入点	160
4.14 访问资源	112	5.5.10 便捷的 Advisor 实现	161
4.15 使用 Java 类进行配置	113	5.6 了解代理	161
4.15.1 Java 中的 ApplicationContext 配置	113	5.6.1 使用 JDK 动态代理	162
4.15.2 Spring 混合配置	119	5.6.2 使用 CGLIB 代理	162
4.15.3 Java 或 XML 配置?	121	5.6.3 比较代理性能	163
4.16 配置文件	121	5.6.4 选择要使用的代理	165
4.17 使用 Java 配置来配置 Spring 配置文件	123	5.7 切入点的高级使用	166
4.18 Environment 和 PropertySource 抽象	125	5.7.1 使用控制流切入点	166
4.19 使用 JSR-330 注解进行配置	128	5.7.2 使用组合切入点	168
4.20 使用 Groovy 进行配置	130	5.7.3 组合和切入点接口	170
4.21 Spring Boot	132	5.7.4 切入点小结	170
4.22 小结	135	5.8 引入入门	170
第 5 章 Spring AOP	136	5.8.1 引入的基础知识	171
5.1 AOP 概念	137	5.8.2 使用引入进行对象修改检测	172
5.2 AOP 的类型	137	5.8.3 引入小结	175
5.2.1 使用静态 AOP	137	5.9 AOP 的框架服务	175
5.2.2 使用动态 AOP	137	5.9.1 以声明的方式配置 AOP	175
5.2.3 选择 AOP 类型	138	5.9.2 使用 ProxyFactoryBean	176
5.3 Spring 中的 AOP	138	5.9.3 使用 aop 名称空间	180
5.3.1 AOP Alliance	138	5.10 使用 @AspectJ 样式注解	184
5.3.2 AOP 中的 Hello World 示例	138	5.11 AspectJ 集成	189
5.4 Spring AOP 架构	139	5.11.1 关于 AspectJ	189
5.4.1 Spring 中的连接点	139	5.11.2 使用单例切面	189
5.4.2 Spring 中的切面	140	5.12 小结	191
5.4.3 关于 ProxyFactory 类	140	第 6 章 Spring JDBC 支持	192
5.4.4 在 Spring 中创建通知	140	6.1 介绍 Lambda 表达式	192
5.4.5 通知的接口	141	6.2 示例代码的示例数据模型	193

6.3 研究 JDBC 基础结构.....	196	8.2 使用 JPA 执行数据库操作.....	259
6.4 Spring JDBC 基础结构.....	199	8.2.1 使用 Java 持久化查询语言来查询数据.....	260
6.5 数据库连接和数据源.....	200	8.2.2 查询非类型化结果.....	266
6.6 嵌入数据库支持.....	203	8.3 使用构造函数表达式查询自定义 结果类型.....	267
6.7 在 DAO 类中使用 DataSource.....	204	8.3.1 插入数据.....	269
6.8 异常处理.....	206	8.3.2 更新数据.....	271
6.9 JdbcTemplate 类.....	207	8.3.3 删除数据.....	272
6.9.1 在 DAO 类中初始化 JdbcTemplate.....	207	8.4 使用本地查询.....	273
6.9.2 通过 NamedParameterJdbcTemplate 使用命名参数.....	209	8.5 使用简单的本地查询.....	273
6.9.3 使用 RowMapper<T>检索域对象.....	210	8.6 使用 SQL ResultSet 映射进行本地查询.....	274
6.10 使用 ResultSetExtractor 检索嵌套域 对象.....	211	8.7 Spring Data JPA 介绍.....	278
6.11 建模 JDBC 操作的 Spring 类.....	213	8.7.1 添加 Spring Data JPA 库依赖项.....	279
6.12 使用 MappingSqlQuery<T>查询数据.....	215	8.7.2 使用 Spring Data JPA Repository 抽象 进行数据库操作.....	279
6.13 插入数据并检索生成的键.....	220	8.8 使用 JpaRepository.....	283
6.14 使用 BatchSqlUpdate 进行批处理操作.....	221	8.9 带有自定义查询的 Spring Data JPA.....	284
6.15 使用 SqlFunction 调用存储函数.....	225	8.10 通过使用 Hibernate Envers 保存 实体版本.....	293
6.16 Spring Data 项目: JDBC Extensions.....	226	8.10.1 为实体版本控制添加表.....	293
6.17 使用 JDBC 的注意事项.....	226	8.10.2 为实体版本控制配置 EntityManagerFactory.....	294
6.18 Spring Boot JDBC.....	227	8.10.3 启用实体版本控制和历史检索.....	296
6.19 小结.....	229	8.10.4 测试实体版本控制.....	297
第 7 章 在 Spring 中使用 Hibernate.....	230	8.11 Spring Boot JPA.....	298
7.1 示例代码的示例数据模型.....	230	8.12 使用 JPA 时的注意事项.....	302
7.2 配置 Hibernate 的 SessionFactory.....	232	8.13 小结.....	302
7.3 使用 Hibernate 注解的 ORM 映射.....	234	第 9 章 事务管理.....	303
7.3.1 简单的映射.....	235	9.1 研究 Spring 事务抽象层.....	303
7.3.2 一对多映射.....	238	9.2 PlatformTransactionManager 的实现.....	304
7.3.3 多对多映射.....	239	9.3 分析事务属性.....	305
7.4 Hibernate 会话接口.....	240	9.3.1 TransactionDefinition 接口.....	305
7.4.1 使用 Hibernate 查询语言查询数据.....	241	9.3.2 TransactionStatus 接口.....	306
7.4.2 使用延迟获取进行简单查询.....	241	9.4 示例代码的示例数据模型和基础结构.....	307
7.4.3 使用关联获取进行查询.....	243	9.4.1 创建一个带有依赖项的简单 Spring JPA 项目.....	307
7.5 插入数据.....	245	9.4.2 示例数据模型和通用类.....	308
7.6 更新数据.....	248	9.4.3 使用 AOP 配置进行事务管理.....	315
7.7 删除数据.....	249	9.5 使用编程式事务.....	316
7.8 配置 Hibernate 以便从实体生成表.....	250	9.6 使用 Spring 实现全局事务.....	318
7.9 注解方法或字段?.....	252	9.6.1 实现 JTA 示例的基础结构.....	318
7.10 使用 Hibernate 时的注意事项.....	254	9.6.2 使用 JTA 实现全局事务.....	319
7.11 小结.....	254	9.6.3 Spring Boot JTA.....	325
第 8 章 在 Spring 中使用 JPA 2 进行数据访问.....	255	9.6.4 使用 JTA 事务管理器的注意事项.....	328
8.1 JPA 2.1 介绍.....	255	9.7 小结.....	329
8.1.1 示例代码的示例数据模型.....	256		
8.1.2 配置 JPA 的 EntityManagerFactory.....	256		
8.1.3 使用 JPA 注解进行 ORM 映射.....	258		

第 10 章 使用类型转换和格式化进行验证.....	330	12.6.4 使用 Spring MVC 展示 REST 样式的 Web 服务.....	376
10.1 依赖项.....	330	12.7 配置 Castor XML.....	377
10.2 Spring 类型转换系统.....	331	12.7.1 实现 SingerController.....	378
10.3 使用PropertyEditors从字符串进行转换.....	331	12.7.2 配置 Spring Web 应用程序.....	380
10.4 Spring 类型转换介绍.....	333	12.7.3 使用 curl 测试 RESTful-WS.....	382
10.4.1 实现自定义转换器.....	333	12.7.4 使用 RestTemplate 访问 RESTful-WS.....	383
10.4.2 配置 ConversionService.....	334	12.7.5 使用 Spring Security 来保护 RESTful-WS.....	386
10.4.3 任意类型之间的转换.....	335	12.8 使用 Spring Boot 开发 RESTful-WS.....	389
10.5 Spring 中的字段格式化.....	338	12.9 在 Spring 中使用 AMQP.....	392
10.5.1 实现自定义格式化器.....	338	12.10 小结.....	397
10.5.2 配置 ConversionServiceFactoryBean.....	339		
10.6 Spring 中的验证.....	340	第 13 章 Spring 测试.....	398
10.6.1 使用 Spring Validator 接口.....	340	13.1 测试类别介绍.....	398
10.6.2 使用 JSR-349 Bean Validation.....	342	13.2 使用 Spring 测试注解.....	399
10.6.3 在 Spring 中配置 Bean Validation 支持.....	343	13.3 实施逻辑单元测试.....	400
10.6.4 创建自定义验证器.....	344	13.3.1 添加所需的依赖项.....	400
10.7 使用 AssertTrue 进行自定义验证.....	346	13.3.2 单元测试 Spring MVC 控制器.....	401
10.8 自定义验证的注意事项.....	347	13.4 实现集成测试.....	403
10.9 决定使用哪种验证 API.....	347	13.4.1 添加所需的依赖项.....	403
10.10 小结.....	347	13.4.2 配置用于服务层测试的配置文件.....	403
		13.4.3 Java 配置版本.....	404
第 11 章 任务调度.....	348	13.4.4 实施基础结构类.....	405
11.1 任务调度示例的依赖项.....	348	13.4.5 对服务层进行单元测试.....	408
11.2 Spring 中的任务调度.....	349	13.4.6 丢弃 DbUnit.....	410
11.2.1 Spring TaskScheduler 抽象介绍.....	349	13.5 实现前端单元测试.....	413
11.2.2 研究示例任务.....	350	13.6 小结.....	413
11.2.3 使用注解进行任务调度.....	355		
11.2.4 Spring 中异步任务的执行.....	357	第 14 章 Spring 中的脚本支持.....	414
11.3 Spring 中任务的执行.....	359	14.1 在 Java 中使用脚本支持.....	414
11.4 小结.....	360	14.2 Groovy 介绍.....	415
第 12 章 使用 Spring 远程处理.....	361	14.2.1 动态类型化.....	416
12.1 使用示例的数据模型.....	362	14.2.2 简化的语法.....	416
12.2 为 JPA 后端添加必需的依赖项.....	363	14.2.3 闭包.....	417
12.3 实现和配置 SingerService.....	364	14.3 与 Spring 一起使用 Groovy.....	418
12.3.1 实现 SingerService.....	364	14.3.1 开发 Singer 对象域.....	418
12.3.2 配置 SingerService.....	365	14.3.2 实现规则引擎.....	418
12.3.3 公开服务.....	367	14.3.3 将规则工厂实现为 Spring 可刷新 bean.....	420
12.3.4 调用服务.....	368	14.3.4 测试年龄分类规则.....	421
12.4 在 Spring 中使用 JMS.....	369	14.3.5 内联动态语言代码.....	423
12.4.1 在 Spring 中实现 JMS 监听器.....	371	14.4 小结.....	424
12.4.2 在 Spring 中发送 JMS 消息.....	372		
12.5 Spring Boot Artemis 启动器.....	373	第 15 章 应用程序监控.....	425
12.6 在 Spring 中使用 RESTful-WS.....	375	15.1 Spring 中的 JMX 支持.....	425
12.6.1 RESTful Web 服务介绍.....	375	15.2 将 Spring bean 导出为 JMX.....	425
12.6.2 为示例添加必需的依赖项.....	376	15.3 使用 Java VisualVM 进行 JMX 监控.....	426
12.6.3 设计 Singer RESTful Web 服务.....	376		

15.4	监视 Hibernate 统计信息	428	16.10.5	在歌手列表视图中启用 jqGrid	464
15.5	使用了 Spring Boot 的 JMX	429	16.10.6	在服务器端启用分页	466
15.6	小结	431	16.11	处理文件上传	468
第 16 章	Web 应用程序	432	16.11.1	配置文件上传支持	468
16.1	实现示例的服务层	433	16.11.2	修改视图以支持文件上传	469
16.1.1	对示例使用数据模型	433	16.11.3	修改控制器以支持文件上传	470
16.1.2	实现 DAO 层	435	16.12	用 Spring Security 保护 Web 应用程序	471
16.1.3	实现服务层	435	16.12.1	配置 Spring 安全性	471
16.2	配置 SingerService	436	16.12.2	将登录功能添加到应用程序中	473
16.3	MVC 和 Spring MVC 介绍	437	16.12.3	使用注解来保护控制器方法	475
16.3.1	MVC 介绍	438	16.13	使用 Spring Boot 创建 Spring Web 应用程序	475
16.3.2	Spring MVC 介绍	438	16.14	设置 DAO 层	476
16.3.3	Spring MVC WebApplicationContext 层次结构	439	16.14.1	设置服务层	477
16.3.4	Spring MVC 请求生命周期	439	16.14.2	设置 Web 层	478
16.3.5	Spring MVC 配置	440	16.14.3	设置 Spring 安全性	479
16.3.6	在 Spring MVC 中创建第一个视图	442	16.15	创建 Thymeleaf 视图	479
16.3.7	配置 DispatcherServlet	443	16.16	使用 Thymeleaf 扩展	482
16.3.8	实现 SingerController	444	16.17	小结	486
16.3.9	实现歌手列表视图	445	第 17 章	WebSocket	487
16.3.10	测试歌手列表视图	445	17.1	WebSocket 介绍	487
16.4	理解 Spring MVC 项目结构	445	17.2	与 Spring 一起使用 WebSocket	487
16.5	实现国际化(i18n)	446	17.3	使用 WebSocket API	488
16.5.1	在 DispatcherServlet 配置中配置国际化	446	17.4	使用 STOMP 发送消息	496
16.5.2	为国际化支持而修改歌手列表视图	448	17.5	小结	500
16.6	使用主题和模板	448	第 18 章	Spring 项目: 批处理、集成和 XD 等	501
16.7	使用 Apache Tiles 查看模板	450	18.1	Spring Batch	502
16.7.1	设计模板布局	450	18.2	JSR-352	507
16.7.2	实现页面布局组件	451	18.3	Spring Boot Batch	509
16.8	在 Spring MVC 中配置 Tiles	453	18.4	Spring Integration	512
16.9	实现歌手信息视图	454	18.5	Spring XD	516
16.9.1	将 URL 映射到视图	454	18.6	Spring 框架的五个最显著的功能	517
16.9.2	实现显示歌手视图	454	18.6.1	功能性 Web 框架	518
16.9.3	实现编辑歌手视图	456	18.6.2	Java 9 互操作性	526
16.9.4	实现添加歌手视图	459	18.6.3	JDK 模块化	526
16.9.5	启用 JSR-349(bean 验证)	460	18.6.4	使用 Java 9 和 Spring WebFlux 进行 反应式编程	528
16.10	使用 jQuery 和 jQuery UI	462	18.6.5	Spring 支持 JUnit 5 Jupiter	529
16.10.1	jQuery 和 jQuery UI 介绍	462	18.7	小结	536
16.10.2	在视图中使用 jQuery 和 jQuery UI	462	附录 A	设置开发环境	537
16.10.3	使用 CKEditor 进行富文本编辑	463			
16.10.4	使用 jqGrid 实现具有分页支持的 数据网格	464			

第 1 章

Spring 介绍

提到 Java 开发人员社区，我们会不自觉地想起 19 世纪 40 年代末的淘金者，他们在北美的河流上疯狂地淘金，寻找黄金碎片。作为 Java 开发人员，我们的“河流”充斥着开源项目，但像淘金者一样，找到一个有用的项目可能会既费时又费力。

对许多开源 Java 项目的常见抱怨是，它们仅仅是为了填补与最新技术或模式的差距而创建的。话虽如此，但许多高质量、可用的项目满足并解决了真实应用程序的实际需求，在阅读本书的过程中，就会遇到部分此类项目，从而更好地了解 Spring。Spring 的第一个版本于 2002 年 10 月发布，由一个带有易于配置和使用的控制反转(LoC)容器的小型内核组成。多年来，Spring 已经成为 Java Enterprise Edition(JEE)服务器的主要替代品，并且发展成为一个由许多不同项目组成的成熟技术，每个项目都有自己的目的，因此无论是想要构建微服务、应用程序还是经典的 ERP，Spring 都有一个项目可以满足需求。

在本书中，你将会看到许多使用了不同开源技术的应用程序，所有这些应用程序都整合在 Spring 框架下。在使用 Spring 时，应用程序开发人员可以使用各种开源工具，而无须编写大量代码，也不需要将应用程序与任何特定工具紧密耦合。

如章名所示，本章的主要内容是介绍 Spring 框架，而不是提供任何可靠的示例或说明。如果你已经熟悉 Spring，那么可以跳过本章并直接进入第 2 章。

1.1 什么是 Spring

如果想要解释 Spring，那么最难的部分就是对其进行分类。通常情况下，Spring 被描述为构建 Java 应用程序的轻量级框架，但这种描述带来了两个有趣的观点。

首先，与许多其他框架(比如仅限于 Web 应用程序的 Apache Struts)不同，可以使用 Spring 构建 Java 中的任何应用程序(例如，独立的应用程序、Web 应用程序或 JEE 应用程序)。

其次，该描述中的轻量级一词真正指的并不是类的数量或发布的大小，而是整体性定义 Spring 原则：最轻的影响。从某种意义上讲，Spring 是轻量级的，因为只需要对应用程序代码进行很少的更改(如果有的话)就可以获得 Spring Core 所带来的好处。如果想要在任何时候停止使用 Spring，那么你会发现可以很容易做到。

请注意，上述描述仅针对 Spring Core——许多额外的 Spring 组件(例如数据访问)需要更紧密地与 Spring 框架耦合。然而，这种耦合的好处是非常明显的，在本书中将介绍相关的技术来最大限度地减少这种耦合对应用程序的影响。

1.1.1 Spring 框架的演变

Spring 框架源自 Rod Johnson 编写的 Expert One-on-One: J2EE Design and Development 一书(Wrox 出版社，2002 年出版)。在过去十年中，Spring 框架在核心功能、相关项目以及社区支持方面发展迅猛。随着 Spring 框架的新主要版本的推出，有必要快速回顾一下 Spring 的每个里程碑版本所带来的重要特性，并最终发展到 Spring Framework 5.0。

- Spring 0.9: 这是该框架第一个公开发表的版本, 以 *Expert One-on-One: J2EE Design and Development* 一书为基础, 提供了 bean 配置基础、AOP 支持、JDBC 抽象框架、抽象事务支持等。该版本没有官方参考文档, 但可以在 SourceForge 上找到现有的源代码和文档¹。
- Spring 1.x: 这是发布的第一个带有官方参考文档的版本。它由图 1-1 所示的七个模块组成。
 - ◆ Spring Core: bean 容器以及支持的实用程序。
 - ◆ Spring Context: ApplicationContext、UI、验证、JNDI、Enterprise JavaBean(EJB)、远程处理和邮件支持。
 - ◆ Spring DAO: 事务基础结构、Java Database Connectivity(JDBC)和数据访问对象(DAO)支持。
 - ◆ Spring ORM: Hibernate、iBATIS 和 Java Data Object(JDO)支持。
 - ◆ Spring AOP: 符合 AOP 联盟的面向方面编程(AOP)实现。
 - ◆ Spring Web: 基本集成功能, 比如多部分功能、通过 servlet 侦听器进行上下文初始化以及面向 Web 的应用程序上下文。
 - ◆ Spring Web MVC: 基于 Web 的 Model-View-Controller(MVC)框架。
- Spring 2.x: 该版本由图 1-2 所示的六个模块组成。现在, Spring Context 模块包含在 Spring Core 中, 而在 Spring 2.x 版本中, 所有的 Spring Web 组件都由单个项目表示。

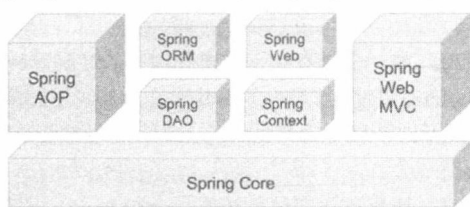


图 1-1 Spring Framework 1.x 版本概述

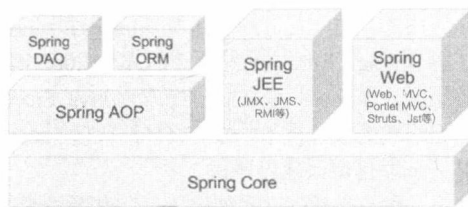


图 1-2 Spring Framework 2.x 版本概述

- ◆ 通过使用新的基于 XML Schema 的配置而不是 DTD 格式来简化 XML 配置。值得注意的改进方面包括 bean 定义、AOP 以及声明式事务。
- ◆ 用于 Web 和门户的新 bean 作用域(请求、会话和全局会话)。
- ◆ 支持 AOP 开发的@AspectJ 注解。
- ◆ Java Persistence API(JPA)抽象层。
- ◆ 完全支持异步 JMS 消息驱动的 POJO(用于普通的旧 Java 对象)。
- ◆ JDBC 简化包括在使用 Java 5+时的 SimpleJdbcTemplate。
- ◆ JDBC 命名参数支持(NamedParameterJdbcTemplate)。
- ◆ 针对 Spring MVC 的表单标签库。
- ◆ 对 Portlet MVC 框架的介绍。
- ◆ 动态语言支持。可以使用 JRuby、Groovy 以及 BeanShell 来编写 bean。
- ◆ JMX 中的通知支持以及可控的 MBean 注册。
- ◆ 为调度任务而引入的 TaskExecutor 抽象。
- ◆ Java 5 注解支持, 特别针对@Transactional、@Required 和@AspectJ。
- Spring 2.5.x: 该版本包含以下功能。
 - ◆ 名为@Autowired的新配置注解以及对 JSR-250 注解(@Resource、@PostConstruct 和@PreDestroy)的支持。
 - ◆ 新的构造型注解: @Component、@Repository、@Service 和@Controller。
 - ◆ 自动类路径扫描支持, 可以检测和连接带有构造型注解的类。
 - ◆ AOP 更新, 包括一个新的 bean 切入点元素以及 AspectJ 加载时织入(weaving)。
 - ◆ 完整的 WebSphere 事务管理支持。
 - ◆ 除了 Spring MVC @Controller 注解, 还添加了@RequestMapping、@RequestParam 和@ModelAttribute 注解, 从而支持通过注解配置进行请求处理。
 - ◆ 支持 Tiles 2。

¹ 可以从 SourceForge 站点下载 Spring 的早期版本(包括 0.9 版本): <https://sourceforge.net/projects/springframework/files/springframework/>。

- ◆ 支持 JSF 1.2。
- ◆ 支持 JAX-WS 2.0/2.1。
- ◆ 引入了 Spring TestContext Framework，提供注解驱动和集成测试支持，不受所用测试框架的影响。
- ◆ 能够将 Spring 应用程序上下文部署为 JCA 适配器。
- Spring 3.0.x: 这是基于 Java 5 的 Spring 的第一个版本，旨在充分利用 Java 5 的功能，如泛型、可变参数和其他语言改进。该版本引入了基于 Java 的 @Configuration 模型。目前已经对框架模块进行了修改，分别针对每个模块 JAR 使用一棵源代码树进行管理。图 1-3 中对此进行了抽象描述。



图 1-3 Spring Framework 3.0.x 版本概述

- ◆ 支持 Java 5 功能，例如泛型、可变参数以及其他改进。
- ◆ 对 Callable、Futures、ExecutorService 适配器和 ThreadFactory 集成提供很好的支持。
- ◆ 框架模块目前针对每个模块 JAR 都使用一棵源代码树进行分别管理。
- ◆ Spring Expression Language(SpEL)的引入。
- ◆ 核心 Java Config 功能和注解的集成。
- ◆ 通用型转换系统和字段格式化系统。
- ◆ 全面支持 REST。
- ◆ 新的 MVC XML 名称空间和其他注解，例如 Spring MVC 中的 @CookieValue 和 @RequestHeaders。
- ◆ 验证增强功能和 JSR-303(bean 验证)支持。
- ◆ 对 Java EE 6 的早期支持，包括 @Async/@Asynchronous 注解、JSR-303、JSF 2.0、JPA 2.0 等。
- ◆ 支持嵌入式数据库，例如 HSQL、H2 和 Derby。
- Spring 3.1.x: 该版本包含以下功能。
 - ◆ 新的缓冲抽象。
 - ◆ 可以用 XML 定义 bean 定义配置文件，同时也支持 @Profile 注解。
 - ◆ 针对统一属性管理的环境抽象。
 - ◆ 与常见 Spring XML 名称空间元素等价的注解，如 @ComponentScan、@EnableTransactionManagement、@EnableCaching、@EnableWebMvc、@EnableScheduling、@EnableAsync、@EnableAspectJAutoProxy、@EnableLoadTimeWeaving 和 @EnableSpringConfigured。
 - ◆ 支持 Hibernate 4。
 - ◆ Spring TestContext Framework 对 @Configuration 类和 bean 定义配置文件的支持。
 - ◆ 名称空间 c:简化了构造函数注入。
 - ◆ 支持 Servlet 3 中 Servlet 容器的基于代码的配置。
 - ◆ 能够在不使用 persistence.xml 的情况下启动 JPA EntityManagerFactory。
 - ◆ 将 Flash 和 RedirectAttributes 添加到 Spring MVC 中，从而允许通过使用 HTTP 会话重定向属性 URI 模板变量增强功能。
 - ◆ 能够使用 @Valid 来注解 Spring MVC @RequestBody 控制器方法参数。
 - ◆ 能够使用 @RequestPart 来注解 Spring MVC 控制器方法参数。

- Spring 3.2.x: 该版本包含以下功能。
 - ◆ 支持基于 Servlet 3 的异步请求处理。
 - ◆ 新的 Spring MVC 测试框架。
 - ◆ 新的 Spring MVC 注解@ControllerAdvice 和@MatrixVariable。
 - ◆ 支持 RestTemplate 和@RequestBody 参数中的泛型类型。
 - ◆ 支持 Jackson JSON 2。
 - ◆ 支持 Tiles 3。
 - ◆ 现在, @RequestBody或@RequestPart参数的后面可以跟一个Errors参数, 从而可以对验证错误进行处理。
 - ◆ 能够通过使用 MVC 名称空间和 Java Config 配置选项来排除 URL 模式。
 - ◆ 支持没有 Joda Time 的@DateTimeFormat。
 - ◆ 全局日期和时间格式化。
 - ◆ 跨框架的并发优化, 从而最小化锁定, 并改进了作用域/原型 bean 的并发创建。
 - ◆ 新的基于 Gradle 的构建系统。
 - ◆ 迁移到 GitHub(<https://github.com/SpringSource/spring-framework>)。
 - ◆ 在框架和第三方依赖中支持精简的 Java SE 7/OpenJDK 7。现在, CGLIB 和 ASM 已经成为 Spring 的一部分。除了 AspectJ 1.6, 其他版本都支持 AspectJ 1.7。
- Spring 4.0.x: 这是一个重要的 Spring 版本, 也是第一个完全支持 Java 8 的版本。虽然仍然可以使用较旧版本的 Java, 但 Java SE6 已经提出了最低版本要求。弃用的类和方法已被删除, 但模块组织几乎相同, 如图 1-4 所示。



图 1-4 Spring Framework 4.0.x 版本概述

- ◆ 通过新的 www.spring.io/guides 网站上的一系列入门指南提高了入门体验。
- ◆ 从先前的 Spring 3 版本中删除了弃用的软件包和方法。
- ◆ 支持 Java 8, 将最低 Java 版本提高到 6 update 18。
- ◆ Java EE 6 及以上版本现在被认为是 Spring Framework 4.0 的基准。
- ◆ Groovy bean 定义 DSL, 允许通过 Groovy 语法配置 bean 定义。
- ◆ 核心容器、测试和一般 Web 改进。
- ◆ WebSocket、SockJS 和 STOMP 消息。
- Spring 4.2.x: 该版本包含以下功能。
 - ◆ 核心改进(例如, 引入@AliasFor, 并修改现有的注解以使用它)。
 - ◆ 全面支持 Hibernate ORM 5.0。
 - ◆ JMS 和 Web 改进。
 - ◆ 对 WebSocket 消息传递的改进。
 - ◆ 测试改进, 最引人注目的是引入了@Commit 来替换@Rollback(false), 并引入了 AopTestUtils 实用工具类, 允许访问隐藏在 Spring 代理后面的底层对象。
- Spring 4.3.x: 该版本包含以下功能。
 - ◆ 完善了编程模型。