◇ 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）

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

王 净 译

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)
克里斯 • 舍弗(Chris Schaefer)
克拉伦斯 • 厚(Clarence Ho) 著

王　净 译

清华大学出版社

北　京

Pro Spring 5, An In-Depth Guide to the Spring Framework and Its Tools, Fifth Edition

Iuliana Cosmina, Rob Harrop, Chris Schaefer, Clarence Ho
EISBN：978-1-4842-2807-4

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: An In-Depth Guide to the Spring Framework and Its Tools, 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：成凤进
责任印制：

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 刷 者：
装 订 者：
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：

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

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(IoC)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布的版本，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。

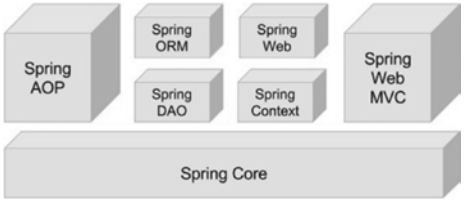

图 1-1　Spring Framework 1.x 版本概述

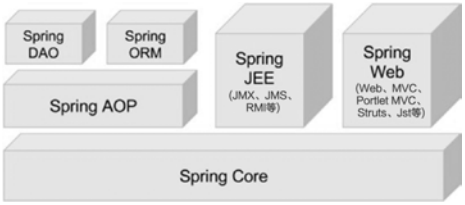

图 1-2　Spring Framework 2.x 版本概述

  - 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。

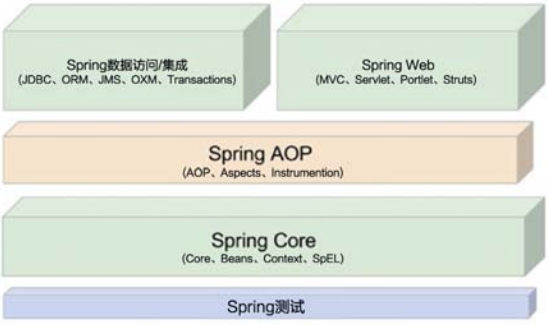

图 1-3　Spring Framework 3.0.x 版本概述

- 支持 Java 5 功能，例如泛型、可变参数以及其他改进。
- 对 Callables、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 4.0x：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。

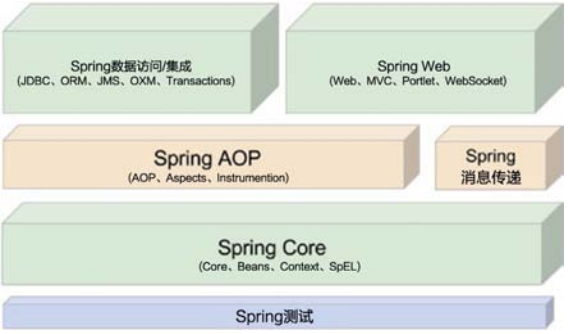

图 1-4　Spring Framework 4.0.x 版本概述

  - 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。

- 在核心容器(包含 ASM 5.1、CGLIB 3.2.4 以及 spring-core.jar 中的 Objenesis 2.4)和 MVC 方面有了相当大的改进。
- 添加了组合注解。
- Spring TestContext Framework 需要 JUnit 4.12 或更高版本。
- 支持新的库，包括 Hibernate ORM 5.2、Hibernate Validator 5.3、Tomcat 8.5 和 9.0、Jackson 2.8 等。

- Spring 5.x：这是一个主要版本。整个框架代码库都基于 Java 8，并且自 2016 年 7 月起与 Java 9 完全兼容[1]。
  - 支持 Portlet、Velocity、JasperReports、XMLBeans、JDO、Guava、Tiles 2 和 Hibernate 3。
  - 现在，XML 配置名称空间被流式传输到未版本化的模式；虽然特定版本的声明仍然被支持，但要针对最新的 XSD 架构进行验证。
  - 充分利用 Java 8 的强大功能，从而在性能上得到极大的改进。
  - Resource 抽象为防御 getFile 访问提供了 isFile 指示符。
  - Spring 提供的 Filter 实现完全支持 Servlet 3.1 签名。
  - 支持 Protobuf 3.0。
  - 支持 JMS 2.0+和 JPA 2.1+。
  - 引入了 Spring Web Flow，这是一个用于替代 Spring MVC 的项目，构建在反应式基础之上，这意味着它完全是异步和非阻塞的，主要用于事件循环执行模型，而非传统的每个请求执行模型都带有一个线程的大型线程池(基于 Project Reactor 构建[2])。
  - Web 和核心模块适用于反应式编程模型[3]。
  - Spring 测试模块有了很大的改进。现在支持 JUnit 5，引入了新的注解来支持 Jupiter 编程和扩展模型，例如@SpringJUnitConfig、@SpringJUnitWebConfig、@EnabledIf 和@DisabledIf。
  - 支持在 Spring TestContext Framework 中实现并行测试执行。

1.1.2 翻转控制或依赖注入

Spring 框架的核心是基于控制反转(Inversion of Control，IoC)的原理。IoC 是一种将组件依赖项的创建和管理外部化的技术。请参考一个示例，比如类 Foo 依赖于 Bar 类的一个实例来执行某种处理。传统上，Foo 通过使用 new 操作符来创建 Bar 实例或从某个工厂类中获取 Bar 实例。而如果使用 IoC 方法，Bar 的一个实例(或一个子类)在运行时由某个外部进程提供给 Foo。这种在运行时注入依赖项的行为促使 Martin Fowler 将 IoC 重命名为更具描述性的依赖注入(Dependency Injection，DI)。第 3 章将讨论 DI 所管理的依赖项的具体特性。

⚠ 正如将在第 3 章中看到的那样，当提到控制反转时，使用术语依赖注入始终是正确的。在 Spring 的上下文中，可以互换使用这些术语，而不会有任何意义上的损失。

Spring 的 DI 实现基于两个核心的 Java 概念：JavaBeans 和接口。当使用 Spring 作为 DI 提供程序时，可以通过不同的方式(例如，XML 文件、Java 配置类、代码中的注解或新的 Groovy bean 定义方法)在应用程序中灵活定义依赖项配置。JavaBean(POJO)提供了一种创建 Java 资源的标准机制，这些资源可以通过多种方式进行配置，例如构造函数和 setter 方法。在第 3 章中，将学习 Spring 如何使用 JavaBean 规范来形成其 DI 配置模型的核心；实际上，任何 Spring 管理的资源都被称为 *bean*。如果你对 JavaBean 不熟悉，请参阅第 3 章开头介绍的快速入门。

接口和 DI 是互利的技术。显而易见，将应用程序设计和编写为接口可以让应用程序更加灵活，但是将使用接口设计的应用程序连接在一起是非常复杂的，并且会给开发人员带来额外的编码负担。通过使用 DI，可以将基于接口的设计所需的代码量减少到几乎为零。同样，通过使用接口，可以充分利用 DI，因为 bean 可以利用任何接口实现来满足它们的依赖项。接口的使用还允许 Spring 利用 JDK 动态代理(Proxy 模式)为横切关注点提供诸如 AOP 的强大概念。

1 请记住，根据 http://openjdk.java.net/projects/jdk9/上提供的 Oracle 时间表，Java 9 已于 2017 年 9 月正式向公众发布。
2 Project Reactor 实现了 Reactive Streams API 规范，请参阅 https://projectreactor.io/。
3 反应式编程是一种涉及智能路由和事件消耗的微型架构，会导致非阻塞应用程序异步和事件驱动，并且需要少量线程在 JVM 中垂直扩展，而不是通过集群进行水平扩展。

在 DI 的上下文中，Spring 更像是容器，而不是框架——提供应用程序类的实例以及它们所需的所有依赖项，但它是以一种不受干扰的方式实现的。如果使用 Spring 实现 DI，那么仅依赖于类中的 JavaBeans 命名约定——没有特殊的类可供继承或者没有专门的命名方案可供遵循。如果有，在使用 DI 的应用程序中所做的唯一更改是在 JavaBean 上公开更多的属性，从而允许在运行时注入更多的依赖项。

1.1.3 依赖注入的演变

在过去的几年中，由于 Spring 和其他 DI 框架的普及，DI 在 Java 开发人员社区中获得了广泛的认可。同时，开发人员确信使用 DI 是应用程序开发的最佳实践，使用 DI 的好处也很好理解。

当 Java Community Process(JCP)在 2009 年采用 JSR-330(针对 Java 的依赖注入)时，DI 的流行得到了承认。SRR-330 已经成为正式的 Java 规范请求，正如所预料的那样，规范来源之一是 Rod Johnson——Spring 框架的创始人。在 JEE 6 中，JSR-330 成为整个技术栈的规范之一。与此同时，EJB 架构(从 3.0 版开始)也大幅改进；它采用 DI 模型以简化各种 Enterprise JavaBeans 应用程序的开发。

虽然在第 3 章将会对 DI 进行全面讨论，但关注一下使用 DI 的好处(相比于更传统的方法)还是非常值得的。

- **减少粘合代码**：DI 最大的优点之一是能够用最少的代码将应用程序的组件粘合在一起。通常所编写的代码是非常少的，所以创建依赖项时只需要创建一个对象的新实例即可。但是，当需要在 JNDI 存储库中查找依赖项时，或者无法直接调用依赖项时，粘合代码可能会变得非常复杂，就像调用远程资源一样。在这些情况下，DI 可以通过提供自动 JNDI 查找和远程资源的自动代理来真正简化粘合代码。
- **简化应用程序配置**：通过采用 DI，可以大大简化配置应用程序的过程。可以使用各种选项来配置那些可注入其他类的类。可以使用相同的技术向“注入器”表达依赖项需求，以注入适当的 bean 实例或属性。另外，DI 可以更容易地将一个依赖项的实现变换为另一个依赖项的实现。假设有一个对 PostgreSQL 数据库执行数据操作的 DAO 组件，并且想要将数据库升级到 Oracle。如果使用 DI，只需要简单地重新配置业务对象的适当依赖项，以便使用 Oracle 实现而不是 PostgreSQL 实现。
- **能够在单个存储库中管理常见依赖项**：如果使用传统方法对公共服务的依赖项(例如，数据源连接、事务和远程服务)进行管理，那么可以在需要的地方(依赖类中)创建依赖项的实例(或从某些工厂类中查找依赖项实例)。这样一来，就会导致依赖项遍布应用程序的不同类中，一旦更改这些类，可能就会出现问题。当使用 DI 时，关于这些公共依赖项的所有信息都包含在一个存储库中，从而使得依赖项的管理变得更加简单并且不易出错。
- **改进的可测试性**：当为 DI 设计类时，可以轻松替换依赖项。当测试应用程序时，这特别方便。假设有一个需要执行一些复杂处理的业务对象；其中使用 DAO 访问存储在关系数据库中的数据。当进行测试时，你可能对测试 DAO 并不感兴趣；而只是想要使用各种数据集来测试一下业务对象。在传统的方法中，业务对象负责获取 DAO 本身的实例，因此很难进行测试，因为无法轻松地将 DAO 实现替换为返回测试数据集的模拟实现。相反，需要确保测试数据库包含正确的数据，并使用完整的 DAO 实现进行测试。而如果使用 DI，可以创建返回测试数据集的 DAO 对象的模拟实现，然后将其传递给业务对象进行测试。此机制可以扩展用于测试应用程序的任何层，特别适用于测试可以创建 HttpServletRequest 和 HttpServletResponse 的模拟实现的 Web 组件。
- **培养良好的应用程序设计**：一般而言，DI 的设计意味着针对接口进行设计。在一个典型的面向注入的应用程序的设计中，所有主要组件都被定义为接口，然后使用 DI 容器创建这些接口的具体实现并连接在一起。在 DI 和基于 DI 的容器(例如 Spring)出现之前，这种设计是可能的，但通过使用 Spring，可以免费获得大量的 DI 功能，并且能够专注于构建自己的应用程序逻辑，而不是支持它的框架。

正如前面所介绍的那样，DI 为应用程序提供了很多好处，但它并非没有缺点。特别是对于那些不熟悉代码的人来说，DI 很难让他们知道某个特定依赖项的哪个实现与哪个对象连接。通常，只有当开发人员对 DI 没有任何经验时，才会出现上述问题。一旦经验丰富了并且遵循良好的 DI 编码实践(例如，将每个应用程序层中的所有可注入类放入同一个包)之后，开发人员就能轻松地知道哪个实现与哪个对象相连接。虽然在大多数情况下，优点是远远多于缺点的，但是在规划应用程序时应该考虑这一点。

1.1.4 除了依赖注入

虽然单独使用 Spring Core 及其先进的 DI 功能是非常有价值的，但 Spring 真正的价值在于其数不胜数的附加功能，所有这些功能都是使用 DI 原则进行精心设计和构建的。Spring 为应用程序的所有层提供功能，从用于数据访问权限的辅助应用程序编程接口(API)到高级 MVC 功能。Spring 中这些功能的优点在于，虽然 Spring 通常提供自己的方法，但也可以轻松地将它们与 Spring 中的其他工具集成在一起，从而使这些工具成为 Spring 家族的一流成员。

支持 Java 9

Java 8 带来了 Spring Framework 5 支持的许多令人兴奋的功能，其中最引人注目的是 lambda 表达式以及 Spring 的回调接口的方法引用。Spring 5 发布计划与 JDK 9 的最初发布计划保持一致，尽管 JDK 9 的发布期限已经推迟，但 Spring 5 已按计划发布。据估计，Spring 5.1 将完全支持 JDK 9。Spring 5 将利用 JDK 9 功能，比如紧凑字符串、ALPN 堆栈和新的 HTTP 客户端实现。尽管 Spring Framework 4.0 支持 Java 8，但兼容性仍然保持在 JDK 6 update 18。对于新开发的项目，建议使用更新版本的 Java，如 Java 7 或 Java 8。Spring 5 需要 Java 8+，因为 Spring 开发团队已经将 Java 8 语言级别应用于整个框架代码库，但从一开始，Spring 5 就是基于 JDK 9 构建的，以便为 JDK 9 的功能提供全面的支持。

使用 Spring 实现面向方面编程(AOP)

AOP 提供了在单个地方实现横切逻辑(即适用于应用程序多个部分的逻辑)的能力，并且可以自动在应用程序中应用该逻辑。Spring 对 AOP 的支持是为目标对象创建动态代理，并用配置好的通知来创建对象以执行横切逻辑。鉴于 JDK 动态代理的本质，目标对象必须实现一个接口来声明将应用 AOP 建议的方法。另一个流行的 AOP 库是 Eclipse AspectJ 项目[1]，它提供了更强大的功能，包括对象构建、类加载以及更强大的横切功能。然而，对于 Spring 和 AOP 开发人员来说真正的好消息是，从 2.0 版开始，Spring 提供了与 AspectJ 更紧密的集成。主要有以下几个亮点：

- 支持 AspectJ 风格的切入点表达式。
- 支持@AspectJ 注解样式，同时仍然使用 Spring AOP 进行织入(weaving)。
- 在 AspectJ 中实现 DI 方面的支持。
- 支持 Spring ApplicationContext 中的加载时织入。


从 Spring Framework 3.2 开始，可以使用 Java 配置启用@AspectJ 注解支持。

这两种 AOP 都有各自的位置，大多数情况下，Spring AOP 足以满足应用程序的横切需求。但是，对于更复杂的需求，AspectJ、Spring AOP 和 AspectJ 可以在同一个 Spring 应用程序中混合使用。

AOP 有许多应用程序。许多传统 AOP 示例中给出的典型功能大都是执行某种日志记录，但 AOP 的应用远不止日志记录应用程序。实际上，在 Spring 框架中，AOP 被用于多种用途，特别是在事务管理中。第 5 章将详细介绍 Spring AOP，其中将演示在 Spring 框架和自己的应用程序中 AOP 的典型用法，同时讨论 AOP 性能以及与传统技术相比 AOP 更适合的领域。

Spring 表达式语言

表达式语言(EL)是一种允许应用程序在运行时操纵 Java 对象的技术。然而，EL 的问题在于不同的技术提供了自己的 EL 实现和语法。例如，Java Server Pages(JSP)和 Java Server Faces(JSF)都有自己的 EL，它们的语法不同。为了解决这个问题，创建了统一表达式语言(UEL)。

由于 Spring 框架发展非常迅速，因此需要一种标准表达式语言，它可以在所有 Spring 框架模块以及其他 Spring 项目之间共享。所以，从 3.0 版开始，Spring 引入了 *Spring 表达式语言(Spring Expression Language*，SpEL*)*。SpEL 为评估表达式以及在运行时访问 Java 对象和 Spring bean 提供了强大功能。结果可以在应用程序中使用或注入其他 JavaBean 中。

Spring 中的验证

验证是任何类型的应用程序中都存在的另一个大问题。在理想情况下，无论数据操作请求来自前端、批处理作

1 www.eclipse.org/aspectj。

业还是从远程启动(例如，通过 Web 服务、RESTful Web 服务或远程过程调用[RPC])，包含业务数据的 JavaBeans 中的属性验证规则都应该以一致的方式应用。

为解决这些问题，Spring 通过 Validator 接口提供了一个内置的验证 API。该接口提供了一种简洁的机制，允许将验证逻辑封装到负责验证目标对象的类中。除目标对象外，validate 方法接收一个 Errors 对象，该对象用于收集可能发生的任何验证错误。

Spring 还提供了一个方便的实用程序类 ValidationUtils，它提供了便捷的方法来调用其他验证程序、检查常见问题(如空字符串)，并将错误报告给 Errors 对象。

在需求驱动下，JCP 还开发了 JSR-303(Bean Validation)，它提供了定义 bean 验证规则的标准方法。 例如，当将 @NotNull 注解应用于 bean 的属性时，它将强制该属性在能够保存到数据库中之前不包含空值。

从 3.0 版开始，Spring 为 JSR-303 提供了开箱即用支持。如果要使用其 API，只需要声明一个 LocalValidatorFactoryBean并将Validator接口注入到Spring管理的任何bean中即可。Spring负责解析底层实现。默认情况下，Spring将首先查找Hibernate Validator(hibernate.org/subprojects/validator)，这是一个流行的JSR-303实现。包括 Spring MVC在内的许多前端技术(例如JSF 2和Google Web Toolkit)也支持在用户界面中应用JSR-303验证。开发人员需要在用户界面和后端编写相同的验证逻辑的时代已经一去不复返了。更多内容，请参阅第10章。

⚠ 从 Spring Framework 4.0 版本开始，JSR-349(Bean Validation)的 1.1 版本得到支持。

在 Spring 中访问数据

数据访问和持久化似乎是 Java 世界中讨论最多的话题。Spring 为这些数据访问工具的选择提供了极好的集成。另外，Spring 使得普通的 JDBC 成为许多项目的一个可行的选择，其根据标准的 API 简化了封装的 API。Spring 的数据访问模块对 JDBC、Hibernate、JDO 和 JPA 提供了开箱即用支持。

⚠ 从 Spring Framework 4.0 版本开始，就已经删除了对 iBATIS 的支持。MyBatis-Spring 项目提供了与 Spring 的集成，读者可以在 http://mybatis.github.io/spring/上找到更多信息。

然而，在过去几年中，由于互联网和云计算的爆炸性增长，除了关系数据库，还开发了许多其他的“专用”数据库。本书的示例使用了基于键值对的用来处理大量数据的数据库(通常称为 NoSQL)、图形数据库以及文档数据库。为了帮助开发人员支持这些数据库并避免 Spring 数据访问模块复杂化，人们创建了一个名为 Spring Data[1]的单独项目。该项目被进一步划分为不同的类别，以便支持更具体的数据库访问需求。

⚠ 本书不包含 Spring 对非关系数据库的支持。如果对此主题感兴趣，则可以好好地研究一下前面提到的 Spring Data 项目。项目页面详细说明了它所支持的非关系数据库，并链接到这些数据库的主页。

Spring 对 JDBC 的支持使得在 JDBC 之上构建应用程序成为可能，即使创建更复杂的应用程序也是如此。对 Hibernate、JDO 和 JPA 的支持使得原本简单的 API 变得更加简单，从而减轻了开发人员的负担。当使用 Spring API 通过任何工具访问数据时，可以充分利用 Spring 出色的事务支持，第 9 章将对此进行全面讨论。

Spring 中最好的功能之一就是能够轻松地在应用程序中混合和匹配数据访问技术。例如，可以使用 Oracle 运行一个应用程序，同时使用 Hibernate 处理大部分数据访问逻辑。但如果想要使用某些 Oracle 特有的功能，那么可以非常容易地使用 Spring 的 JDBC API 实现数据访问层中的这一部分逻辑。

Spring 中的对象/XML 映射

大多数应用程序需要整合或向其他应用程序提供服务。一个常见的需求是定期或实时地与其他系统交换数据。在数据格式方面，XML 最常用。因此，经常需要将 JavaBean 转换为 XML 格式，反之亦然。Spring 支持许多常见的 Java-to-XML 映射框架，并且仍然不需要直接耦合到任何特定的实现。Spring 提供了用于编组(将 JavaBean 转换为 XML)以及解组(将 XML 转换为 Java 对象)到任何 Spring bean 的通用接口。Spring 支持 Java Architecture for XML Binding(JAXB)、Castor、XStream、JiBX 以及 XMLBeans 等通用库。在第 12 章中，当讨论远程访问 XML 格式的商业数据的 Spring 应用程序时，将介绍如何在自己的应用程序中使用 Spring 的对象/XML 映射(OXM)支持。

1 http://projects.spring.io/spring-data。

管理事务

Spring 为事务管理提供了一个优秀的抽象层，允许编程式和声明式事务控制。通过对事务使用 Spring 抽象层，可以更加简单地更改底层事务协议和资源管理器。可以从简单的、本地的且特定于资源的事务开始，逐步转向使用全局的多资源事务，而无须更改代码。第 9 章将详细介绍事务。

简化以及与 JEE 集成

随着诸如 Spring 之类的 DI 框架逐步被接受，许多开发人员选择通过使用 DI 框架来构建应用程序，以便支持 JEE 的 EJB 方法。因此，JCP 社区也意识到 EJB 的复杂性。从 EJB 规范的 3.0 版开始，API 被简化了，所以它现在包含 DI 的许多概念。

但是，对于构建在 EJB 上的应用程序，或者需要在 JEE 容器中部署基于 Spring 的应用程序并利用应用程序服务器的企业服务(例如，Java Transaction API 的事务管理器、数据源连接池和 JMS 连接工厂)的应用程序，Spring 还为这些技术提供简化的支持。针对 EJB，Spring 提供了一个简单的声明以执行 JNDI 查找并注入 Spring bean。反之，Spring 还提供了将 Spring bean 注入 EJB 中的简单注解。

对于存储在 JNDI 可访问位置的任何资源，Spring 允许删除复杂的查找代码，并将 JNDI 管理的资源作为依赖项在运行时注入其他对象中。这样做的另一个作用是将应用程序与 JNDI 分离，从而可以在未来更多地重用代码。

Web 层中的 MVC

几乎可以在任何环境中使用 Spring，包括从桌面到 Web，Spring 都提供了一组丰富的类来支持创建基于 Web 的应用程序。当使用 Spring 时，在选择如何实现 Web 前端时就有了最大的灵活性。为了开发 Web 应用程序，MVC 模式是最流行的做法。在最近的版本中，Spring 已经从简单的 Web 框架逐渐演变为完整的 MVC 实现。首先，Spring MVC 中的视图支持非常广泛。除了提供对 JSP 以及 Java Standard Tag Library(JSTL)的标准支持(主要由 Spring 标签库提供支持)，还可以充分利用对 Apache Velocity、FreeMarker、Apache Tiles、Thymeleaf 和 XSLT 的完全集成支持。另外，可以找到一组基本视图类，以便将 Microsoft Excel、PDF 和 JasperReports 输出添加到应用程序中。

大多数情况下，Spring MVC 足以满足 Web 应用程序开发需求。不过，Spring 还可以与其他流行的 Web 框架(如 Struts、JSF、Atmosphere、Google Web Toolkit(GWT)等)集成。

在过去几年里，Web 框架的技术发展迅速，用户需要更多的响应式和交互式体验，从而导致 Ajax 的崛起，成为开发富 Internet 应用程序(RIA)时广泛采用的技术。另一方面，用户也希望能够通过任何设备访问他们的应用程序，包括智能手机和平板电脑。这就需要支持 HTML5、JavaScript 和 CSS3 的 Web 框架。第 16 章将讨论如何使用 Spring MVC 开发 Web 应用程序。

WebSocket 支持

从 Spring Framework 4.0 开始就已经支持 JSR-356(用于 WebSocket 的 Java API)了。WebSocket 定义了一个用于在客户端和服务器之间创建持久连接的 API，通常在 Web 浏览器和服务器间实现。WebSocket 式的开发为高效的全双工通信打开了大门，为高度响应式应用程序实现了实时消息交换。对 WebSocket 支持的使用详见第 17 章。

远程支持

在 Java 中访问或公开远程组件从来就不是一件最简单的工作。通过使用 Spring，可以充分利用对各种远程技术的广泛支持来快速公开和访问远程服务。Spring 对各种远程访问机制提供支持，包括 Java Remote Method Invocation(RMI)、JAX-WS、Caucho Hessian and Burlap、JMS、Advanced Message Queuing Protocol(AMQP)和 REST。除了这些远程协议，Spring 还提供了基于标准 Java 序列化的基于 HTTP 的调用器。通过应用 Spring 的动态代理功能，可以将远程资源的代理作为依赖项注入某个类中，从而避免将应用程序与特定远程实现耦合，并且减少了为应用程序所需编写的代码量。第 12 章将讨论 Spring 的远程支持。

邮件支持

发送电子邮件是许多类型应用程序的典型需求，Spring 框架内对此进行了很好的处理。Spring 为发送电子邮件消息提供了一个简化的 API，可以很好地与 Spring DI 功能相匹配。Spring 支持标准的 JavaMail API。Spring 提供了在 DI 容器中创建原型消息的能力，并将其用作从应用程序发送的所有消息的基础。这样一来就可以轻松自定义邮件参数，例如主题和发件人地址。另外，为了自定义消息体，Spring 集成了模板引擎，如 Apache Velocity；从而允

许邮件内容在 Java 代码中外部化。

作业调度支持

大多数重要的应用程序都需要某种调度功能。无论是将更新发送给客户还是执行内务处理任务，能够在预定义时间运行代码的能力对于开发人员来说都是非常宝贵的工具。Spring 提供了可以满足大多数常见场景的调度支持。任务可以按固定时间间隔或通过使用 UNIX cron 表达式进行调度。另一方面，对于任务执行和调度，Spring 也与其他调度库进行了集成。例如，在应用程序服务器环境中，Spring 可以将执行委托给许多应用程序服务器所使用的 CommonJ 库。而对于作业调度，Spring 还支持包括 JDK Timer API 和 Quartz(一种常用的开源调度库)在内的库。第 11 章将详细介绍 Spring 中的调度支持。

动态脚本支持

从 JDK 6 开始，Java 就引入了动态语言支持，可以在 JVM 环境中执行用其他语言编写的脚本，比如 Groovy、JRuby 和 JavaScript。Spring 还支持在 Spring 驱动的应用程序中执行动态脚本，或者定义一个用动态脚本语言编写并注入其他 JavaBean 中的 Spring bean。Spring 支持的动态脚本语言包括 Groovy、JRuby 和 BeanShell。在第 14 章，将详细讨论 Spring 对于动态脚本的支持。

简化的异常处理

Spring 中真正有助于减少重复、样板代码量的一项功能是异常处理。在异常处理方面，Spring 思想核心认为在 Java 中检查型异常(checked exception)被过度使用，并且框架不应该强制捕捉那些不可能恢复的异常——这个观点完全正确。实际上，许多框架的设计目的是避免编写代码来处理检查型异常。但是，这些框架中有许多仍然坚持采用检查型异常的方法，从而人为减少了异常类层次结构的粒度。在使用 Spring 时你会注意到，由于使用未检查型异常(unchecked exception)给开发人员带来了极大便利，异常层次显著细化。在本书中，你将会看到一些例子，其中 Spring 异常处理机制可以减少必须编写的代码量，同时提高识别、分类和诊断应用程序内错误的能力。

1.2 Spring 项目

关于 Spring 项目最值得关注的事情之一是社区活动的水平以及 Spring 与其他项目(比如 CGLIB、Apache Geronimo 和 AspectJ)之间的交流。开源代码最受欢迎的优点之一是，即使项目明天结束，也会留下代码；但是当面对这些代码时，你可能并不希望使用一个与 Spring 大小相当的代码库来完成支持和改进。因此，了解 Spring 社区的建立和活跃程度是很有必要的。

1.2.1 Spring 的起源

如本章前面所述，Spring 的起源可以追溯到 *Expert One-to-One：J2EE Design and Development* 一书。在该书中，Rod Johnson 提出了自己的名为 Interface 21 Framework 的框架，该框架主要用于他自己的应用程序。正如你在今天所看到的，这个框架已经被发布到开源世界，形成 Spring 框架的基础。在经过早期测试版以及发布候选版阶段之后，Spring 得到了快速发展。第一个官方版本 1.0 于 2004 年 3 月发布。从那时起，Spring 发生了翻天覆地的变化，在编写本书时，Spring 框架的最新主要版本是 5.0。

1.2.2 Spring 社区

如果想要找到所需的任何开源项目，Spring 社区是最好的地方之一。邮件列表和论坛总是很活跃，不断有新的功能被开发出来。开发团队真正致力于使 Spring 成为所有 Java 应用程序框架中最成功的一个，主要体现在所复制代码的质量上。正如前面已经提到的，Spring 的发展也受益于其他的开源项目，这一事实在考虑 Spring 发行版与其他框架的依赖关系时非常重要。从用户的角度看，也许 Spring 的最佳特性之一就是随着版本一起发布的优秀文档和测试套件。文档提供了 Spring 的几乎所有功能，使得新用户可以轻松地选择使用该框架。Spring 提供的测试套件非常全面——开发团队对所有内容都进行了测试。如果他们发现一个 bug，那么首先会编写一个突出该 bug 的测试，然后让测试通过，从而修复这个 bug。当然，修复 bug 和创建新功能并不仅限于开发团队！如果想要提供自己的代码，

可通过官方 GitHub 存储库(http://github.com/spring-projects)提交针对任何 Spring 项目组合的请求。此外，可通过官方的 Spring JIRA(https://jira.spring.io/secure/Dashboard.jspa)创建和跟踪问题。这一切意味着什么呢？简言之，这意味着应该对 Spring 框架的质量充满信心，并且相信在可预见的未来，Spring 开发团队将继续改进这个已经非常优秀的框架。

1.2.3 Spring 工具套件

为简化在 Eclipse 中开发基于 Spring 的应用程序，Spring 创建了 Spring IDE 项目。不久之后，Rod Johnson 创立的 SpringSource 公司创建了一个名为 Spring Tool Suite(STS)的集成工具，该工具可从 https://spring.io/tools 下载。虽然该工具以前需要付费，但现在可免费使用。它将 Eclipse IDE、Spring IDE、Mylyn(基于任务的 Eclipse 开发环境)、Maven for Eclipse、AspectJ Development Tools 以及许多其他有用的 Eclipse 插件集成到一个包中。每个新版本都添加了更多功能，如 Groovy 脚本语言支持、图形化 Spring 配置编辑器、用于项目(比如 Spring Batch 和 Spring Integration)的可视化开发工具，以及对 Pivotal tc Server 应用程序服务器的支持。


SpringSource 已被 VMware 收购并且被并入 Pivotal Software。

除基于 Java 的套件外，Groovy/Grails 工具套件也具有相似的功能，但主要针对 Groovy 和 Grails 开发(http://spring.io/tools)。

1.2.4 Spring Security 项目

Spring Security 项目(http://projects.spring.io/spring-security)，以前被称为 Spring 的 Acegi Security System，是 Spring 产品组合中的另一个重要项目。Spring Security 为 Web 应用程序和方法级安全性提供全面的支持。它与 Spring 框架以及其他常用的身份验证机制(比如 HTTP 基本身份验证、基于表单的登录、X.509 证书以及单点登录(SSO)产品(例如 CA SiteMinder))紧密集成。它为应用程序资源提供基于角色的访问控制，在具有更复杂安全需求的应用程序(例如数据分隔)中，支持使用访问控制列表(ACL)。但 Spring Security 主要用于保护 Web 应用程序，相关内容将在第 16 章详细讨论。

1.2.5 Spring Boot

建立应用程序的基础是一项烦琐的工作。必须创建项目的配置文件，并且安装和配置其他工具(如应用程序服务器)。Spring Boot(http://projects.spring.io/spring-boot/)是一个 Spring 项目，可以轻松创建可运行的、独立的、生产级的、基于 Spring 的应用程序。Spring Boot 针对不同类型的 Spring 应用程序提供开箱即用的配置，这些配置都封装在*启动器(starter)*包中。例如，web-starter 包提供了一个预配置且易于自定义的 Web 应用程序上下文，并支持 Tomcat 7+、Jetty 8+和 Undertow 1.3 嵌入式 servlet 容器。

Spring Boot 还包含 Spring 应用程序所需的所有依赖项，同时考虑到版本之间的兼容性。在编写本书时，Spring Boot 的当前版本是 2.0.0.RELEASE。

第 4 章将详述 Spring Boot，作为 Spring 项目的替代配置，后续章节中的大多数项目都将使用 Spring Boot 运行，因为它使开发和测试更加实用和快速。

1.2.6 Spring 批处理和集成

不用多说，批处理作业的执行和集成是应用程序中的常见用例。为了满足该需求并为这些领域的开发人员提供便利，Spring 创建了 Spring Batch 和 Spring Integration 项目。Spring Batch 为批处理作业的实现提供了一个通用框架以及各种策略，减少了大量的样板代码。通过实现 Enterprise Integration Patterns(EIP)，Spring Integration 可使 Spring 应用程序与外部系统轻松集成。相关内容将在第 20 章详细讨论。

1.2.7 许多其他项目

前面已经介绍了 Spring 的核心模块以及 Spring 组合中的一些主要项目，但还有很多其他项目是为了满足社区的

不同需求而创建的，比如 Spring Boot、Spring XD、Android for Spring、Spring Mobile、Spring Social 和 Spring AMQP。其中一些项目将在第 20 章中进一步讨论。更多详细信息，请参阅 Spring by Pivotal 网站(www.spring.io/projects)。

1.3 Spring 的替代品

回顾前面关于开源项目数量的讨论，如果说 Spring 并不是唯一提供依赖注入功能或用于构建应用程序的完整端到端解决方案的框架，那么我想你不应该感到惊讶。实际上，相关的项目实在太多了。本着开放的精神，这里简要讨论一下其中的几个框架，但这些平台都不能提供与 Spring 完全一样的解决方案。

1.3.1 JBoss Seam 框架

Seam 框架(http://seamframework.org)由 Gavin King(Hibernate ORM 库的创建者)创建，是另一个完整的基于 DI 的框架。它支持 Web 应用程序前端开发(JSF)、业务逻辑层(EJB 3)和 JPA 持久化。正如你看到的，Seam 和 Spring 之间的主要区别在于 Seam 框架完全基于 JEE 标准构建。JBoss 还将 Seam 框架中的想法提供给 JCP，并成为 JSR-299(Java EE 平台的上下文和依赖注入)。

1.3.2 Google Guice

Google Guice(http://code.google.com/p/google-guice)是另一个流行的 DI 框架。在搜索引擎巨头 Google 的领导下，Guice 成了一个轻量级框架，专注于为应用程序配置管理提供 DI。它也是 JSR-330(用于 Java 的依赖注入)的参考实现。

1.3.3 PicoContainer

PicoContainer(http://picocontainer.com)是一个非常小的 DI 容器，允许在不引入除 PicoContainer 外的任何依赖项的情况下为应用程序使用 DI。由于 PicoContainer 只是一个 DI 容器，因此如果随着应用程序的增长需要引入另一个框架，比如 Spring，那么在这种情况下最好从一开始就使用 Spring。但是，如果只需要一个小型的 DI 容器，那么 PicoContainer 是一个不错的选择。由于 Spring 将 DI 容器与框架的其余部分分开打包，因此可以轻松使用 PicoContainer 并保持未来的灵活性。

1.3.4 JEE 7 容器[1]

如前所述，DI 的概念被 JCP 广泛采用并实现。当为应用程序服务器开发符合 JEE 7(JSR-342)的应用程序时，可以在所有层中使用标准 DI 技术。

1.4 小结

本章概述 Spring 框架，并讨论了所有主要功能，同时详细介绍了本书的相关章节。阅读本章后，你应该了解 Spring 能做些什么；接下来将学习它是如何做的。在下一章中，将讨论你需要知道的有关启动和运行基本 Spring 应用程序的所有信息，同时会展示如何获得 Spring 框架并讨论打包选项、测试套件和文档。另外，第 2 章将介绍一些基本的 Spring 代码，其中包括历史悠久的 Hello World 示例。

1 JEE 8 的发布日期被推迟至 2017 年底，请参阅 https://jcp.org/en/jsr/detail?id=366。