

<<Spring技术内幕 (第2版) >>

图书基本信息

书名：<<Spring技术内幕 (第2版) >>

13位ISBN编号：9787111365709

10位ISBN编号：7111365704

出版时间：2012-2

出版时间：机械工业出版社

作者：计文柯

页数：416

版权说明：本站所提供下载的PDF图书仅提供预览和简介，请支持正版图书。

更多资源请访问：<http://www.tushu007.com>

前言

为什么要写这本书 本书探讨了Spring框架的设计原理、架构和运行机制。作为在Java领域最为成功的开源软件之一，Spring在Java EE开发中，使用者众多。本书以Spring的源代码为依托，结合Spring的设计思路，从内部实现的角度，对Spring的实现进行了翔实的分析，希望能够通过这种分析，使读者在开发者的层面掌握Spring，为开发Spring应用提供更扎实的框架基础。

“忽如一夜春风来”，伴随着2002年Rod Johnson的《Expert One-on-One J2EE Design and Development》一书的出版而正式发布的Spring框架（也就是当年的interface21），经过这几年的发展，已经逐渐成熟起来。

“吹面不寒杨柳风”，Spring带来的崭新开发理念，也早已伴随着它的广泛应用而“飞入寻常百姓家”。

与此同时，随着Spring的不断成熟和完善，开源社区的成长，以及Spring开发团队的不懈努力，以Spring为核心的一系列开源软件产品也越来越丰富，现已发展成为一个包括软件构建、开发、运行、部署整个软件生命周期的产品族群。

Spring不但改变了Java EE应用的开发和服务模式，向纯商业软件发出了有力的挑战，同时也成为Java软件生态链中不可或缺的重要组成部分。

它所具备的那种平易近人、内涵丰富的品质，对我们这些软件爱好者来说，实在是一个不可多得的学习范本。

简化Java企业应用的开发是Spring框架的目标。

其轻量级的开发思想，为开发者提供便利的出发点（for the developer，to the developer and by the developer——这是Rod Johnson在一次演讲中的开场白），以及具有活力的开源社区，所有的这些，都为使用Java开发企业应用和Web应用带来了福音，这些都是Spring吸引我们的地方。

在Java企业应用中，与我们熟悉的企业应用服务器一样，Spring也希望能够集成管理企业应用资源，以及为应用开发提供平台支持。

在这一点上，Spring与UNIX和Windows等传统意义上的操作系统，在计算系统中起到的作用是类似的。

不同点在于，传统操作系统关心的是存储、计算、通信、外围设备等这些物理资源的管理，并在管理这些资源的基础上，为应用程序提供统一的平台和服务接口；而Spring关心的是，如何为开发者集中管理在Java企业应用和Web应用中涉及的数据持久化、事务处理、消息中间件、分布式计算等抽象资源，并在此基础上，为应用提供了一个基于POJO的开发环境。

尽管二者面向的资源、管理的对象、支持的应用，以及使用的场景不同，但它们在整個计算系统中的定位，却有着可以类比和相互参考之处。

所以，笔者根据对传统操作系统的认识方法，粗浅地把Spring框架划分为核心、组件和应用三个基本的层次，通过这三个层次中一些主要特性来剖析Spring的工作原理和运作机制。

同时，也用这样的认识逻辑来组织本书中要阐述的内容。

在这样的层次划分中，首先看到的是对IoC容器和AOP这两个核心模块的工作原理的分析，它们都是Spring平台实现的核心部分；同时，它们也是Spring的其他模块实现的基础。

虽然，对大多数开发者而言，都只是在此基础上进行相关的配置和外部功能的使用，但是，深入理解这两个核心模块的工作原理和运作机制，对于我们更好地应用Spring进行开发是至关重要的。

因为，从Spring要简化Java EE开发的出发点来看，它是通过对POJO开发提供支持来实现的。

具体地说，Spring通过为应用基于POJO的开发模式提供支持，从而使应用开发和复杂的Java EE服务实现解耦，并由此通过提高单元测试覆盖率（也就是应用系统的可测试性）来有效地提高整个Spring应用的开发质量。

在这样的开发场景下，需要把为POJO提供支持的各种Java EE服务支持抽象到Spring应用平台中去，并将其封装起来。

具体来说，这一系列的封装工作，在Spring及其应用实现中，离不开IoC容器和AOP这两个核心模块的

<<Spring技术内幕 (第2版)>>

支持，它们在很大程度上体现了Spring作为应用开发平台的核心价值。

它们的实现是Rod Johnson在他的另外一本著作《Expert One-on-One J2EE Development without EJB》中所提到“Without EJB设计思想”的具体体现，同时，也深刻地体现了Spring背后的设计理念。

其次，在IoC容器和AOP这两个核心模块的支持下，Spring为了简化Java EE的开发，为应用开发提供了许多现成的用户态的系统组件，比如事务处理、Web MVC、JDBC、O/R映射、远端调用等，通过这些系统组件，为企业应用服务的实现提供驱动支持。

这些由Spring或其生态系统（其本身、子项目或者社区）提供的，类似于驱动模块般的系统组件是开发应用时经常会用到的Java EE服务抽象。

通过使用Spring提供的这些类似于驱动组件的中间产品，通过这一层Java EE服务的抽象，从而让用户可以通过使用简单的开发接口或应用模板，不但能够很方便地使用各种Java EE服务，还可以灵活地选取提供这些服务的各种不同的具体实现方案。

让应用可以在各种第三方开源软件或者商业产品中自由选择，充分体现了Spring作为应用平台的开放性。

Spring作为一个开源项目，它本身就是一个开放的生态系统。

对于和Spring相关的一些项目，可以把它们看做在某个领域的用户应用，因为它们是和Spring实现紧密相关的，或者它们本身就作为Spring框架的应用案例，体现了许多使用Spring的技巧。

这些内容都是我们开发应用时的理想参考，并且会随着技术的发展而不断丰富，比如Spring DM、Spring FLEX、ACEGI安全性框架，以及Pet Clinic应用实例等。

一方面，可以把这些实现作为应用的一个基本方案加以裁剪，以满足特定领域的需求；另一方面，通过剖析这些应用，可以为应用开发提供很好的参考和借鉴，提高应用开发的效率。

从更深层次的技术层面上来看，因为Spring是一个基于Java语言的应用平台，如果我们能够对Spring的运行环境Java计算模型（比如JVM的实现原理）有一些了解，将会加深我们对Spring实现原理的理解。

反射机制、代理类、字节码技术等这些JVM特性，都是在Spring实现中会涉及的一些Java计算环境的底层技术。

一般的应用开发人员可能不会直接从事与JVM底层实现相关的工作，但是，这些计算环境的底层知识对深入理解Spring是不可缺少的。

说了这么多，很多读者可能已经有些迫不及待了，只有对Spring的设计和实现身临其境地接触才是真实的，这里太多的文字已经成为一种累赘。

本书将带领你到Spring核心设计这个茂密而又充满生机的源代码丛林中去一探究竟。

在这里，你会惊奇地发现：这个过程就像是阅读优美的散文一样，是与开源软件开发者及开发者社区之间的一种畅快淋漓的交流，让人如痴如醉。

第1版与第2版的区别 本书是第2版，在写作过程中吸收了读者对上一版内容的许多意见和建议，比如着重增加了对Spring宏观框架和设计方面的阐述，加强了对Spring各种特性应用场景方面的描述，并结合了深入具体的源代码实现。

希望通过这些改进，给读者一个从应用到设计再到实现的完整理解，弥补第1版中深度有余，内容层次不够丰富，分析手法单一等诸多不足。

较第1版而言，第2版的改动主要体现在以下几个方面，希望读者能够在阅读中体会。

在内容阐述方式上，对每一章的内容进行了调整和重新编排，基本按照“使用场景”、“设计和实现过程”、“源码实现”这样的逻辑来重新组织大部分内容。希望通过这样的组织方式，能够使读者以由表及里，由配置应用到设计实现，从抽象到具体的方式来了解Spring的各个模块，从而丰富对Spring各个层次的认识。

在基于实现源码分析的基础上，增加了许多对Spring设计的分析，这些设计分析主要包括：在各个Spring模块中，核心类的继承关系、主要接口设计、主要功能特性实现的对象交互关系等。在描述这部分内容时，大多以UML类图和时序图的方式给出，从而帮助读者对Spring的设计有一个直观的了解，而不至于一下子就深入到Spring源代码实现中去，导致只见树木不见森林，另一方面也可

以改善对Spring源代码解读的学习曲线。

同时,在设计分析的过程中,尽可能地对在Spring设计中使用到的一些典型设计模式进行提示,通过这种方式使读者可以体会到各种设计模式在Spring设计中的灵活运用;结合Spring的设计和实现为设计模式的运用提供一系列绝佳的实际案例,从而提高读者对软件设计的理解和设计模式的实际运用能力。

在具体内容的呈现上,对上一版的内容进行了一些调整,这些调整包括:增加了第1章,对Spring项目的概要情况进行了简要阐述;同时把第1版中的一些内容,比如源代码环境的准备、Spring发布包的构建、Spring IDE的基本使用,以及Pet Clinic应用实例的分析等内容,放到了附录部分进行阐述。除此之外,在这一版中,根据Spring项目的自身发展情况增加了一些新的内容,比如对Spring DM和Spring FLEX这两个模块的分析。

通过对这些Spring模块的分析,一方面可以了解Spring的发展历程,丰富视野;另一方面,也可以看到Spring与时俱进的旺盛生命力。

读者对象 学习Java语言和Java EE技术的中高级读者 Spring是利用Java语言实现的,其很多特性的设计和实现都极其优秀,非常具有研究和参考价值。

对这部分读者来说,不仅可以从本书中了解Spring的实现原理,还能通过Spring的设计原理和源代码实现,掌握大量的Java设计方法、设计模式、编码技巧和Java EE开发技术。

Spring应用开发人员 如果要利用Spring进行高级应用开发,抑或是相关的优化和扩展工作,仅仅掌握Spring的配置和基本使用是远远不够的,必须要对Spring框架的设计原理、架构和运作机制有一定的了解。

对这部分读者而言,本书将带领他们全面了解Spring的设计和实现,从而加深对Spring框架的理解,提高开发水平。

同时,本书可以作为他们定制和扩展Spring框架的参考资料。

开源软件爱好者 Spring是开源软件中的佼佼者,它在实现的过程中吸收了很多开源领域的优秀思想,同时也有很多值得学习的创新。

尤为值得一提的是,本书分析Spring设计和实现的方式也许值得所有开源软件爱好者进行学习和借鉴。

通过阅读本书,这部分读者不仅能领略到开源软件的优秀思想,还可以掌握分析开源软件源代码的方法和技巧,从而进一步提高使用开源软件的效率和质量。

平台开发人员和架构师 Spring的设计思想和体系结构、详细设计和源码实现都是非常优秀的,是平台开发人员和架构师们不可多得的参考资料。

如何阅读本书 本书主要内容分为三个部分,分别阐述了Spring的核心、组件和应用三个方面。

在展开这三个部分的内容之前,第1章对Spring的项目情况和整体架构进行了简要的介绍,这一章就像一个热身活动,为本书的主要内容做铺垫,如果您已经很熟悉Spring的使用,这一章可以自行跳过,直接进入下面三个主体部分的内容。

第一部分详细分析了IoC容器和AOP的实现,这部分内容是理解Spring平台的基础,适合对Spring的运行机理有深入了解需求的读者阅读。

在对AOP实现模块的分析中涉及的一些JVM底层技术,也是读者需要具备的背景知识。

第二部分深入阐述了基于Spring IoC容器和AOP的Java EE组件在Spring中的实现。

在这部分内容中可以看到,每一个组件实现的内容基本上都是相对独立的,读者可以结合自己的需求选读。

如果对Spring Web MVC的实现感兴趣,可以阅读第4章;如果对Spring提供的数据库操作的实现机制感兴趣,可以阅读第5章;如果对Spring中提供的统一事务处理的实现感兴趣,可以阅读第6章;如果对Spring提供的各种不同的远端调用实现感兴趣,可以阅读第7章。

第三部分讲述了一些基于Spring的典型应用的实现。

如果读者对在Spring应用中如何满足应用资源的安全性需求方面的内容感兴趣,可以阅读第8章,本章对为Spring应用提供安全服务的ACEGI框架的实现进行了分析,在深入了解这部分内容的基础上,读

<<Spring技术内幕 (第2版)>>

者可以根据自己的应用需求定制自己的安全系统。

第9章分析了Spring DM的设计和实现，通过Spring DM，可以将Spring应用便利地架构到OSGi的框架上去。

第10章分析了Spring Flex的设计和实现，为使用Adobe Flex作为应用前端架构的Spring应用提供参考。

阅读本书时，建议读者在自己的计算机中建立一个源代码阅读环境，这样一方面可以追踪最新的源代码实现，另一方面，可以在阅读的过程中进行各种方式的索引和动手验证，加深对开源软件开发方式的体会。

关于如何建立Spring的源代码环境，进行Spring项目的构建，通过IDE阅读源代码的基本方法等，感兴趣的读者可以参考本书附录中的内容。

在附录A、B、C中，对如何建立Spring项目环境进行了简要介绍，这部分内容包括如何获取Spring项目的源代码，如何构建Spring的发布包，如何使用Spring IDE工具等。

这些知识不但适用于建立Spring的源代码研究环境，还适用于其他的Java开源项目，有一定的普遍性和参考价值。

对于不同的Java开源项目，其使用的源代码管理工具、代码仓库的位置、权限配置会有所不同，但是，整个源代码的获取过程与获取Spring源代码的过程是类似的，整个构建过程也与Spring的构建方式大体相似，是非常值得我们参考的。

在附录D中，对伴随Spring项目的应用实例Pet Clinic进行了分析，这个应用实例为Spring应用开发提供了一个现实的使用案例，虽然简单，却相对完整。

这个应用实例本身也是Spring团队的作品，是Spring项目发布的一部分，其中为我们更好地使用Spring提供参考。

勘误和支持 由于作者对Spring的认知水平有限，再加上写作时的疏漏，书中还存在许多需要改进的地方。

在此，欢迎读者朋友们指出书中存在的问题，并提出指导性意见，不甚感谢。

如果大家有任何与本书相关的内容需要与我探讨，也可以加入本书微信群q.weibo.com/943166，我会及时给予回复。

最后，衷心地希望本书能给大家带来帮助，并祝大家阅读愉快！

致谢 感谢互联网，感谢开源软件，感谢Java，感谢Spring，感谢我们的社区，让我体验到如此美妙的开放氛围，体会到开源软件如此独特的魅力！

好了，不多说了，笔者真诚地希望通过本书为你打开一个小小的入口，曲径通幽，通过这个入口，让我们一起在由开源软件和互联网构成的美丽风景中快乐地旅行！

计文柯 (Wenke J)

<<Spring技术内幕 (第2版) >>

内容概要

本书是国内唯一一本系统分析Spring源代码的著作，也是Spring领域的问题之作，由业界拥有10余年开发经验的资深Java专家亲自执笔，Java开发者社区和Spring开发者社区联袂推荐。

本书第1版不仅在内容上获得了读者的广泛好评，而且在销量上也摘取了同类书的桂冠，曾经一度掀起Java类图书的销售热潮。

第2版不仅继承了第1版在内容组织和写作方式上的优点，而且还根据广大读者的反馈改进了若干细节上的不足。

更为重要的是，结合Spring的最新版本对过时的内容进行了更新，并增加了大量新内容，使本书更趋近于完美。

本书从源代码的角度对Spring的内核和各个主要功能模块的架构、设计和实现原理进行了深入剖析。

你不仅能从本书中参透Spring框架的优秀架构和设计思想，还能从Spring优雅的实现源码中一窥Java语言的精髓。

本书在开篇之前对Spring的设计理念和整体架构进行了全面的介绍，能让读者从宏观上厘清Spring各个功能模块之间的关系；第一部分详细分析了Spring的核心：IoC容器和AOP的实现，能帮助读者了解Spring的运行机制；第二部分深入阐述了各种基于IoC容器和AOP的Java

EE组件在Spring中的实现原理；第三部分讲述了ACEGI安全框架、DM模块以及Flex模块等基于Spring的典型应用的设计与实现。

无论你是Java程序员、Spring开发者，还是平台开发人员、系统架构师，抑或是对开源软件源代码着迷的代码狂人，都能从本书中受益。

作者简介

计文柯：资深软件开发专家和项目经理，拥有10余年业界经验，对Spring等开源软件的应用和实现原理有深入研究和独到见解。产品研发和项目管理经验丰富，曾就职于华为、摩托罗拉等知名企业和硅谷移动互联网创业公司，在软件工程和项目管理方面积累了大量最佳实践。现与同伴一起创立并运营深圳云果科技，专注于云计算解决方案的研究与实施。

书籍目录

前言

第1章 Spring的设计理念和整体架构

- 1.1 Spring的各个子项目
- 1.2 Spring的设计目标
- 1.3 Spring的整体架构
- 1.4 Spring的应用场景
- 1.5 小结

第一部分 Spring核心实现篇

第2章 Spring Framework的核心：IoC容器的实现

- 2.1 Spring IoC容器概述
 - 2.1.1 IoC容器和依赖反转模式
 - 2.1.2 Spring IoC的应用场景
- 2.2 IoC容器系列的设计与实现：BeanFactory和ApplicationContext
 - 2.2.1 Spring的IoC容器系列
 - 2.2.2 Spring IoC容器的设计
- 2.3 IC容器的初始化过程
 - 2.3.1 BeanDefinition的Resource定位
 - 2.3.2 BeanDefinition的载入和解析
 - 2.3.3 BeanDefinition在IoC容器中的注册
- 2.4 IoC容器的依赖注入
- 2.5 容器其他相关特性的设计与实现
 - 2.5.1 ApplicationContext和Bean的初始化及销毁
 - 2.5.2 lazy-init属性和预实例化
 - 2.5.3 FactoryBean的实现
 - 2.5.4 BeanPostProcessor的实现
 - 2.5.5 autowiring (自动依赖装配) 的实现
 - 2.5.6 Bean的依赖检查
 - 2.5.7 Bean对IoC容器的感知
- 2.6 小结

第3章 Spring AOP的实现

- 3.1 Spring AOP概述
 - 3.1.1 AOP概念回顾
 - 3.1.2 Advice通知
 - 3.1.3 Pointcut切点
 - 3.1.4 Advisor通知器
- 3.2 Spring AOP的设计与实现
 - 3.2.1 JVM的动态代理特性
 - 3.2.2 Spring AOP的设计分析
 - 3.2.3 Spring AOP的应用场景
- 3.3 建立AopProxy代理对象
 - 3.3.1 设计原理
 - 3.3.2 配置ProxyFactoryBean
 - 3.3.3 ProxyFactoryBean生成AopProxy代理对象
 - 3.3.4 JDK生成AopProxy代理对象
 - 3.3.5 CGLIB生成AopProxy代理对象

<<Spring技术内幕 (第2版) >>

- 3.4 Spring AOP拦截器调用的实现
 - 3.4.1 设计原理
 - 3.4.2 JdkDynamicAopProxy的invoke拦截
 - 3.4.3 Cglib2AopProxy的intercept拦截
 - 3.4.4 目标对象方法的调用
 - 3.4.5 AOP拦截器链的调用
 - 3.4.6 配置通知器
 - 3.4.7 Advice通知的实现
 - 3.4.8 ProxyFactory实现AOP
- 3.5 Spring AOP的高级特性
- 3.6 小结

第二部分 Spring组件实现篇

第4章 Spring MVC与Web环境

- 4.1 Spring MVC概述
- 4.2 Web环境中的Spring MVC
- 4.3 上下文在Web容器中的启动
 - 4.3.1 IoC容器启动的基本过程
 - 4.3.2 Web容器中的上下文设计
 - 4.3.3 ContextLoader的设计与实现
- 4.4 Spring MVC的设计与实现
 - 4.4.1 Spring MVC的应用场景
 - 4.4.2 Spring MVC设计概览
 - 4.4.3 DispatcherServlet的启动和初始化
 - 4.4.4 MVC处理HTTP分发请求
- 4.5 Spring MVC视图的呈现
 - 4.5.1 DispatcherServlet视图呈现的设计
 - 4.5.2 JSP视图的实现
 - 4.5.3 ExcelView的实现
 - 4.5.4 PDF视图的实现
- 4.6 小结

第5章 数据库操作组件的实现

- 5.1 Spring JDBC的设计与实现
 - 5.1.1 应用场景
 - 5.1.2 设计概要
- 5.2 Spring JDBC中模板类的设计与实现
 - 5.2.1 设计原理
 - 5.2.2 JdbcTemplate的基本使用
 - 5.2.3 JdbcTemplate的execute实现
 - 5.2.4 JdbcTemplate的query实现
 - 5.2.5 使用数据库Connection
- 5.3 Spring JDBC中RDBMS操作对象的实现
 - 5.3.1 SqlQuery的实现
 - 5.3.2 SqlUpdate的实现
 - 5.3.3 SqlFunction
- 5.4 Spring ORM的设计与实现
 - 5.4.1 应用场景
 - 5.4.2 设计概要

<<Spring技术内幕 (第2版) >>

- 5.5 Spring驱动Hibernate的设计与实现
 - 5.5.1 设计原理
 - 5.5.2 Hibernate的SessionFactory
 - 5.5.3 HibernateTemplate的实现
 - 5.5.4 Session的管理
- 5.6 Spring驱动iBatis的设计与实现
 - 5.6.1 设计原理
 - 5.6.2 创建SqlMapClient
 - 5.6.3 SqlMapClientTemplate的实现
- 5.7 小结
- 第6章 Spring事务处理的实现
 - 6.1 Spring与事务处理
 - 6.2 Spring事务处理的设计概览
 - 6.3 Spring事务处理的应用场景
 - 6.4 Spring声明式事务处理
 - 6.4.1 设计原理与基本过程
 - 6.4.2 实现分析
 - 6.5 Spring事务处理的设计与实现
 - 6.5.1 Spring事务处理的编程式使用
 - 6.5.2 事务的创建
 - 6.5.3 事务的挂起
 - 6.5.4 事务的提交
 - 6.5.5 事务的回滚
 - 6.6 Spring事务处理器的设计与实现
 - 6.6.1 Spring事务处理的应用场景
 - 6.6.2 DataSourceTransactionManager的实现
 - 6.6.3 HibernateTransactionManager的实现
 - 6.7 小结
- 第7章 Spring远端调用的实现
 - 7.1 Spring远端调用的应用场景
 - 7.2 Spring远端调用的设计概览
 - 7.3 Spring远端调用的实现
 - 7.3.1 Spring HTTP调用器的实现
 - 7.3.2 Spring Hession/Burlap的实现原理
 - 7.3.3 Spring RMI的实现
 - 7.4 小结
- 第三部分 Spring应用实现篇
 - 第8章 安全框架ACEGI的设计与实现
 - 8.1 Spring ACEGI安全框架概述
 - 8.1.1 概述
 - 8.1.2 设计原理与基本实现过程
 - 8.1.3 ACEGI的Bean配置
 - 8.2 配置Spring ACEGI
 - 8.3 ACEGI的Web过滤器实现
 - 8.4 ACEGI验证器的实现
 - 8.4.1 AuthenticationManager的authenticate
 - 8.4.2 DaoAuthProvider的实现

<<Spring技术内幕 (第2版) >>

- 8.4.3 读取数据库用户信息
- 8.4.4 完成用户信息的对比验证
- 8.5 ACEGI授权器的实现
 - 8.5.1 与Web环境的接口FilterSecurityInterceptor
 - 8.5.2 授权器的实现
 - 8.5.3 投票器的实现
- 8.6 小结
- 第9章 Spring DM模块的设计与实现
 - 9.1 Spring DM模块的应用场景
 - 9.2 Spring DM的应用过程
 - 9.3 Spring DM设计与实现
 - 9.4 小结
- 第10章 Spring Flex的设计与实现
 - 10.1 Spring Flex模块的应用场景
 - 10.2 Spring Flex的应用过程
 - 10.3 Spring Flex的设计与实现
 - 10.4 小结
- 附录A Spring项目的源代码环境
- 附录B 构建Spring项目的发布包
- 附录C 使用Spring IDE
- 附录D Spring Pet Clinic应用实例

章节摘录

我们如何才能既让开发变得容易,又能享受到JavaEE提供的各种服务呢? Spring的目标就是通过自己的努力,让用户体会到这种简单之中的强大。同时,作为应用框架, Spring不想作为另外一种复杂开发模型的替代,也就是说不想用另一种复杂性去替代现有的复杂性,那是换汤不换药,并不能解决问题.这就意味着需要有新的突破。要解决这个问题,需要降低应用的负载和框架的侵入性, Spring是怎样做到这一点的呢?

Spring为我们提供的解决方案就是IoC容器和AOP支持。作为依赖反转模式的具体实现, IoC容器很好地降低了框架的侵入性,也可以认为依赖反转模式是Spring体现出来的核心模式。这些核心模式是软件架构设计中非常重要的因素,我们常常看到的MVC模式就是这样的核心模式。使用好这些核心模式,就像我们在Web应用中使用MVC模式一样,可以获得非常大的便利。

Spring核心的模式实现,是为应用提供IoC容器和AOP框架,从而在企业应用开发中引入新的核心模式,并使用户的开发方式发生很大的变化,具体来说,就是使用POJO来完成开发,在简化用户开发的同时,依然能够使用强大的服务,能够实现复杂的企业应用的开发需求。

……

<<Spring技术内幕 (第2版) >>

媒体关注与评论

正如当初所预料的，本书的第1版大获成功，不仅获得了良好的口碑，而且也取得了喜人的销售成绩，堪称Spring图书领域的里程碑著作。在改版之前，作者收集了大量读者对第1版的反馈，不仅改进了第1版中存在的不足，而且还在组织结构和写作方式进行了优化。更为重要的是，第2版增加了大量新的内容，使本书的内容更丰富、更深入。相信第2版会比第1版更值得期待。强烈推荐！

——专业Spring开发者社区 本书第一版热销的事实有力地证明了这的确是一本值得所有Spring开发者反复研读的书。它以源代码分析为手段，对Spring的架构原理和设计思想进行了全面地剖析，不仅能让我们更深入、更彻底地认识Spring，领略Spring的架构之美和设计之美，更重要的是，它将全面提升我们的Spring开发技能。

——Spring中文用户组

<<Spring技术内幕 (第2版) >>

编辑推荐

畅销书全新升级，第1版广获好评，摘取Spring类图书销量桂冠，掀起Java类图书销售热潮 系
系统解读Spring最新版本源代码，从宏观和微观两个角度深入剖析Spring架构设计与实现原理 资
深Java专家亲自执笔，Java开发者社区和Spring开发者社区联袂推荐

<<Spring技术内幕 (第2版) >>

版权说明

本站所提供下载的PDF图书仅提供预览和简介, 请支持正版图书。

更多资源请访问:<http://www.tushu007.com>