

<<精通Spring>>

图书基本信息

书名：<<精通Spring>>

13位ISBN编号：9787115150295

10位ISBN编号：711515029X

出版时间：2006-10

出版时间：人民邮电出版社

作者：孟劼

页数：593

字数：939000

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>

<<精通Spring>>

内容概要

本书由浅入深、循序渐进地介绍了开源应用框架Spring的开发思想和实践。全书共分为20章，内容涵盖了Spring的基础知识体系、获取和部署方法、背后的先进核心理念、相关的部分设计模式和J2EE核心模式、核心组件Bean工厂的使用、Spring风格的面向方面编程、各种J2EE企业级应用框架和解决方案(包括持久数据访问、事务管理、EJB、邮件服务、日程安排、Web MVC框架、表现层视图技术、JMX远程方案等)、Spring和单元测试等内容。

此外本书也剖析了Spring宠物店示例，最后还讨论了宠物店和Martin Fowler企业架构模式(POEAA)的关系。

本书适用初中级读者学习，对高级用户也有很好的参考价值。

书籍目录

第1章 Spring基础知识	11.1 Spring的来历和目标	11.2 Spring概览	21.3 Spring的应用场景	31.4 Spring和J2EE
51.4.1 经典J2EE体系架构	51.4.2 轻量级容器	61.4.3 Spring和J2EE的关系	61.5 小结	6第2章 开始Spring之旅
72.1 获取Spring	72.1.1 两大开源网站简介	72.1.2 获取Spring发布包和源代码	72.2 Spring的标准发布包和相关软件包	92.2.1 Spring标准发布包
92.2.2 Spring相关软件包	102.2.3 Spring自带范例	122.3 部署Spring	132.4 Spring应用初体验	142.4.1 Hello World遇到的问题
142.4.2 使用Spring对Hello World进行重构	202.4.3 实现依赖注入(DI)	222.4.4 重构总结	242.4.5 简单的Web应用程序Hello World	242.5 小结
26第3章 贯穿Spring应用的核心理念	273.1 轻量级容器再探	273.1.1 容器和轻量级容器	273.1.2 需要容器的理由	283.1.3 轻量级容器的优势
283.2 控制反转(IoC)	283.2.1 一个简单的例子	293.2.2 引入问题	313.2.3 使用控制反转(Inversion of Control)模式	323.2.4 总结
333.3 依赖注入	343.3.1 依赖注入的几种形式	353.3.2 使用Spring进行设值方法和构造函数注入	353.3.3 设值方法注入和构造函数注入的使用场合	363.4 面向方面编程(Asspect Oriented Programming)
373.4.1 AOP的历史来源	373.4.2 为什么需要AOP	383.4.3 AOP的重要概念	393.4.4 理解“横切”	413.4.5 AOP的实现策略
433.4.6 认识AspectJ	443.4.7 安装AspectJ的运行环境	443.4.8 用J2SE动态代理实现AOP	473.4.9 用AspectJ实现AOP	513.4.10 深入理解AspectJ
543.5 单元测试(Unit Test)	583.5.1 单元测试和JUnit框架	593.5.2 JUnit框架的3个核心概念	603.5.3 准备测试	603.5.4 添加测试代码并进行测试
623.5.5 取得更有意义的出错信息	633.5.6 捕捉错误信息	643.5.7 单元测试技巧	653.6 测试驱动开发(Test Driven Development)	673.6.1 传统测试的问题
673.6.2 测试驱动开发的应对策略	683.6.3 一个简单的示例	683.6.4 总结测试驱动开发的优势	733.7 小结	74第4章 模式: Spring前行的路标
754.1 模式(Pattern)入门	754.1.1 什么是模式	754.1.2 不用模式产生的问题	764.1.3 通过实例理解模式本质	794.1.4 小结
814.2 工厂模式(Design Pattern: Factory Method)的精髓	814.2.1 引入问题	824.2.2 解决方法	834.2.3 工厂模式和依赖倒置的关系	854.3 单例模式(Design Pattern: Singleton)
884.3.1 单例模式的实现	884.3.2 单例注册表	894.4 模板模式和策略模式(Design Pattern: Template Method And Strategy)	914.4.1 模板模式	924.4.2 策略模式
954.5 代理模式(Design Pattern: Proxy)	1004.5.1 第一个代理模式的例子	1004.5.2 虚拟代理(Virtual Proxy)	1034.6 数据访问对象模式(J2EE Pattern: Data Access Object)	1054.7 模型视图控制器模式(Architecture/Framework Pattern: Model View Controller)
1134.7.1 Model 1和Model 2简介	1144.7.2 一个Model 2框架示例	1154.8 框架与模式的关系	1204.9 Spring和工厂模式	1214.9.1 Spring工厂体系的另类视图
1214.9.2 Spring工厂核心原理	1224.10 Spring和单例模式	1234.11 Spring的模板以及策略模式	1254.11.1 Spring模板模式	1254.11.2 Spring策略模式
1264.12 Spring和代理模式	1284.13 Spring和数据访问对象模式	1294.14 Spring和MVC模式	131第5章 Spring核心Bean工厂装配指南	1335.1 核心Bean工厂
1335.1.1 初识Bean工厂	1335.1.2 拥有一份Bean的基础定义	1355.1.3 使用构造函数创建Bean	1365.1.4 使用静态工厂方法创建Bean	1365.1.5 使用实例工厂方法创建Bean
1375.1.6 Bean的标识符	1395.1.7 是否使用Singleton	1395.2 装配各种应用协作对象: Bean	1415.2.1 设置Bean的属性和协作者	1415.2.2 Bean的依赖决议
1465.2.3 构造子参数决议	1475.2.4 自动装配(Autowiring)	1505.2.5 依赖检查	1525.3 Bean属性和构造函数参数的细节	1535.3.1 使用value元素设定属性
1535.3.2 使用null元素设定空值	1545.3.3 使用collection(集合)元素定义集合	1545.3.4 通过嵌套Bean元素来定义内部Bean(innerBean)	1575.3.5 使用idref元素预检错误	1585.3.6 使用ref元素设定依赖
1585.3.7 value与ref的简写式	1615.3.8 使用depends-on属性强制依赖	1625.4 方法也可以被注入	1625.4.1 使用Lookup方法注入	1625.4.2 替换任意方法
1645.5 定制Bean的一些原生属性	1655.5.1 生命周期接口	1655.5.2 让Bean“知道”自己的身份	1675.5.3 什么是FactoryBean	1695.6 父和子Bean的定义
1705.7 使用后处理器(Post-Processor)	1735.7.1 使用BeanPostProcessor定制Bean	1735.7.2 使用BeanFactoryPostProcessor定制Bean工厂	1755.7.3 使用自定义的PropertyEditor	177第6章 ApplicationContext和更多特性
1786.1 ApplicationContext简介	1786.2 ApplicationContext的扩展功能	1786.2.1 资源国际化	1786.2.2	

事件传播 1816.2.3 底层资源和ApplicationContext 1846.3 在ApplicationContext中定制Bean的属性和行为 1886.4 使用自定义的PropertyEditors 1896.5 创建Web应用上下文 1926.6 更多的轻松装配方式 1936.6.1 从属性表达式来设置Bean的属性或者构造函数参数 1936.6.2 从一个字段值来设置Bean的属性或者构造函数参数 1956.6.3 调用另外一个方法并且使用其返回值(也可以没有返回值) 1956.6.4 引入其他Bean的定义文件 1966.7 Bean包装器(Wrapper)和属性编辑器(PropertyEditor) 1976.7.1 BeanWrapper简介 1976.7.2 JavaBean属性的设置、获取及嵌套 1976.7.3 属性编辑器 2006.8 Spring验证机制初探 2026.9 小结 203第7章 Spring面向方面编程基础 2047.1 Spring AOP基础概念 2047.1.1 AOP联盟简介 2047.1.2 Spring AOP功能简介 2057.1.3 Spring AOP概念重整 2067.2 Spring AOP编程起步 2077.2.1 陈旧案例重现 2077.2.2 抽离关注点 2087.2.3 划分切入点 2107.2.4 结合Spring AOP和IoC : 声明配置 2127.3 Spring AOP代理和代理工厂 2137.3.1 Spring代理工厂简介 2147.3.2 使用ProxyFactory 2147.3.3 使用ProxyFactoryBean 2157.3.4 使用ProxyFactoryBean暴露的属性 2157.3.5 选择合适的AOP代理 2187.4 Spring AOP建议(Advice) 2197.4.1 前置建议(Before Advice) 2197.4.2 返回后(后置)建议(After Returning Advice) 2247.4.3 环绕拦截建议(Interception Around Advice) 2307.4.4 抛出建议(Throws Advice) 2347.4.5 引介建议(Introduction Advice) 2377.5 Spring AOP切入点(Pointcut)和顾问(Advisor) 2377.5.1 未使用切入点存在的问题 2377.5.2 引入切入点和顾问 2377.5.3 初识切入点背后的ClassFilter和MethodMatcher接口 2397.5.4 透过MethodMatcher接口了解静态和动态切入点 2397.5.5 静态切入点和顾问DefaultPointcutAdvisor 2427.5.6 静态切入点和顾问StaticMethodMatcherPointcut 2427.5.7 静态切入点和顾问NameMatchMethodPointcut 2457.5.8 使用正则表达式切入点 2467.5.9 使用便利的切入点顾问(PointcutAdvisor) 2477.5.10 使用动态切入点(Dynamic Pointcut)和自定义顾问(Advisor) 249第8章 面向方面编程(AOP)高级应用 2548.1 Spring AOP引介(Introduction)概述 2548.2 使用混入(Mixin)实现引介(Introduction)建议 2568.2.1 引入目标对象和待织入功能接口 2568.2.2 给予关键引介建议 2578.2.3 声明配置引介建议并测试 2598.3 使用自动代理(Auto Proxy) 2628.4 Spring AOP企业级服务 2648.4.1 Spring AOP企业级服务菜单 2648.4.2 使用TransactionProxyFactoryBean提供声明式事务管理 2658.4.3 更为强大灵活的事务管理声明方式 2708.4.4 使用EJB代理 2728.5 小结 272第9章 J2EE的持久化数据访问方案 2739.1 传统高效的JDBC方案 2739.2 基于SQL语句映射的Ibatis框架 2759.2.1 Ibatis简介 2759.2.2 Ibatis和高级ORM工具的比较 2759.2.3 Ibatis的应用场合 2769.2.4 Ibatis示例快照 2769.3 流行的ORM利器Hibernate 2789.3.1 Hibernate简介 2789.3.2 Hibernate示例快照 2799.4 再探数据访问对象(DAO) 2829.4.1 传统的JDBC和DAO 2829.4.2 Ibatis和DAO 2829.4.3 Hibernate和DAO 2869.5 ORM方案的应用场合 2869.6 宠物店背后的Ibatis : SQL Map基本概念 2869.7 宠物店背后的Ibatis : XML配置文件 2879.7.1 SQL Map的XML配置文件 2879.7.2 使用元素引入属性文件 2889.7.3 使用元素配置SqlMap 2899.7.4 使用元素指定别名 2899.7.5 使用元素配置事务策略 2899.7.6 使用元素配置数据源 2909.7.7 使用元素引入SQL Map映射文件 2929.7.8 先行准备主控测试和子测试套件 2939.8 宠物店背后的Ibatis : XML映射文件 2949.8.1 SQL Map的XML映射文件 2949.8.2 通过Mapped Statement使用SQL 2969.8.3 使用元素指定SQL语句 2969.8.4 SQL语句特殊规则 2979.8.5 内联参数(Inline Parameter)简介 2989.8.6 使用parameterClass指定参数类 3009.8.7 基本类型输入参数 3009.8.8 使用resultClass指定结果类 3019.8.9 使用resultMap配置映射 3029.8.10 Ibatis事务处理 3039.8.11 小结 3049.9 宠物店背后的Ibatis : XML映射文件高级特性 3059.9.1 Ibatis自动生成的主键 3059.9.2 简单的动态SQL元素 3069.9.3 缓存Mapped Statement结果集 3079.9.4 缓存类型 3089.9.5 动态Mapped Statement 3109.9.6 运行自动测试 3129.10 用Hibernate创建Spring宠物店的简易进货系统 3139.10.1 Spring宠物店持久层渗入 3139.10.2 实现Hibernate进货DAO : 实体持久对象 3149.10.3 实现Hibernate进货DAO : 映射和基本配置 3159.10.4 实现Hibernate进货DAO : 工具类和DAO实现 3179.10.5 实现Hibernate进货DAO : 测试验收 3189.11 小结 320第10章 Spring和J2EE持久化数据访问 32110.1 Spring对数据访问对象(DAO)的支持 32110.1.1 通用的数据访问异常 32110.1.2 通用的DAO抽象支持类 32110.1.3 向业务对象注射DAO 32210.2 Spring对JDBC的支持 32410.2.1 JdbcTemplate和数据源概述 32410.2.2 使用JdbcTemplate和数据源 32510.2.3 对象化RDBMS操作

<<精通Spring>>

概述 32710.2.4 对象化RDBMS操作：使用MappingSqlQuery 32810.2.5 对象化RDBMS操作：使用SqlUpdate 33110.3 Spring对IBatis的支持 33210.3.1 标准JavaBean实体和映射 33210.3.2 衔接IBatis配置和DAO实现 33310.3.3 关键整合点：Spring配置文件 33410.3.4 添加声明式事务管理 33510.4 Spring对Hibernate的支持 33610.4.1 在Spring上下文中配置SessionFactory 33610.4.2 重建Hibernate进货DAO伪实现 33810.4.3 TDD实现规划测试案例 33910.4.4 TDD实现完善基础设施 34110.4.5 添加HibernateTemplate和HibernateCallback实现 34110.4.6 声明式管理Hibernate本地事务 34310.4.7 声明式管理Hibernate分布式事务 345第11章 传统的企业JavaBean(EJB) 34711.1 什么是EJB 34711.2 什么是会话Bean(Session Bean) 34811.2.1 无状态会话Bean 34911.2.2 有状态会话Bean 34911.2.3 何时使用会话Bean 34911.3 实体Bean(Entity Bean) 35011.3.1 EntityBean和Session Bean的异同 35011.3.2 容器管理的持久性(CMP) 35111.3.3 何时需要EntityBean 35211.4 消息驱动Bean 35211.4.1 Message-DrivenBean和Session以及Entity Bean的异同 35311.4.2 何时使用Message-DrivenBean 35311.5 定义客户端访问接口 35311.5.1 远程客户端(Remote Clients) 35411.5.2 本地客户端(Local Clients) 35411.5.3 Local接口和CMR(Container-Managed Relationships) 35511.5.4 远程还是本地访问的选用 35511.5.5 方法参数和访问方式 35511.6 企业Bean的内容和命名约定 35611.7 EJB和JBoss开发实践 35711.7.1 JBoss的安装与启动 35711.7.2 在JBoss下开发部署第一个HelloWorld EJB 36011.7.3 无状态会话Bean(Stateless Session Bean)：开发和部署 36411.7.4 无状态会话Bean：效果和生命周期 36911.7.5 有状态会话Bean(Stateful Session Bean)：开发和部署 37011.7.6 有状态会话Bean(Stateful Session Bean)：效果和生命周期 37611.8 小结 377第12章 Spring和EJB 37812.1 概述 37812.2 Spring对EJB提供的支持 37812.3 Spring的EJB抽象访问层 37912.3.1 深入BeanFactoryLocator接口 38012.3.2 BeanFactoryLocator背后资源消耗的考量 38212.4 使用Spring构建本地无状态会话Bean 38312.4.1 提供业务契约：公共服务接口 38412.4.2 提供本地和本地Home接口 38412.4.3 提供可测试的业务委托：POJO实现类 38512.4.4 提供Spring介入的企业Bean实现主类 38512.4.5 给出EJB组件部署描述符 38612.4.6 给出Spring和特定EJB容器配置，发布EJB 38712.4.7 创建Web客户端，配置Web应用 38812.4.8 解析Spring配置：SLSB代理工厂Bean 38912.4.9 创建Servlet测试SLSB，发布Web客户包 39012.5 使用Spring构建本地有状态会话Bean 39112.5.1 SFSB解决方案 39112.5.2 提供业务契约：公共服务接口 39212.5.3 提供本地和本地Home接口 39212.5.4 提供可测试的业务委托：POJO实现类 39312.5.5 提供Spring介入的企业Bean实现主类 39412.5.6 提供定制化的EJB异常 39612.5.7 给出部署表述符和Spring配置文件 39612.5.8 创建Web客户端并配置Web应用 39712.5.9 解析Spring配置：使用JndiObjectFactoryBean自动查找JNDI 39812.5.10 创建Servlet测试SFSB并发布Web客户包 39912.6 Spring眼中的EJB 40012.6.1 EJB的缺点 40012.6.2 Spring提倡的J2EE轻量架构理念：IoC/DI、AOP、装配和粘合 400第13章 Spring和邮件服务 40213.1 Spring邮件抽象层 40213.2 使用Spring邮件抽象发送简单文本邮件 40413.2.1 定义抽象父类模板 40413.2.2 实现邮件发送子类 40513.2.3 更灵活的实现 40613.2.4 特定功能的子类实现 40913.3 发送HTML MIME邮件的设计实现 40913.3.1 分析设计和初步实现 41013.3.2 定义MailMessage实体 41113.3.3 功能实现和配置 41313.4 发送HTML/纯文本MIME邮件，使用接口回调 41513.4.1 回调的契机 41513.4.2 功能实现和配置 41513.5 小结 418第14章 Spring和排程(scheduling)技术 41914.1 排程(Scheduling)原理 41914.2 什么是Timer 41914.2.1 执行一次 42014.2.2 周期执行 42114.3 Spring对Timer的支持 42214.3.1 用Spring来控制Timer 42214.3.2 使用MethodInvokingTimerTaskFactoryBean工厂类 42414.4 什么是Quartz 42614.4.1 Quartz的核心概念 42614.4.2 使用SimpleTrigger排程 42614.4.3 使用CronTrigger排程 43014.5 Spring对Quartz的支持 43314.5.1 使用Spring支持Quartz SimpleTrigger排程 43314.5.2 使用Spring支持Quartz CronTrigger排程 43514.6 Quartz的持久化 43814.6.1 Quartz持久机制的核心概念 43814.6.2 使用Spring做一个JDBC有状态Quartz Job 43814.6.3 JDBC持久的有状态和无状态job示例运行效果解析 44114.7 小结 442第15章 J2EE中流行的Web MVC框架 44315.1 请求驱动的Web MVC框架简介 44315.2 请求驱动的Web MVC框架在Struts中的实现 44415.2.1 Struts的简介和基本Web配置 44415.2.2 Struts的Action和ActionForm 44515.2.3 Struts的核心原理 44715.3 请求驱动的Web MVC框架在WebWork 2中的实现 44915.3.1 WebWork 2简介 44915.3.2 WebWork 2实

现Web登录的页面流程 45015.3.3 WebWork 2开发实践：配置Web应用 45115.3.4 WebWork 2开发实践：配置验证、实现Action并绑定验证 45215.3.5 WebWork 2开发实践：实现视图 45415.4 事件驱动的Web MVC框架 45515.4.1 简介 45515.4.2 Tapestry概述 45515.4.3 搭建Tapestry示例运行环境 45615.4.4 Tapestry开发实践：配置Web应用 45715.4.5 Tapestry开发实践：创建HTML视图模板 45815.4.6 Tapestry开发实践：创建HTML组件规范文件 46015.4.7 Tapestry开发实践：创建页面类 46115.4.8 无需组件规范的HTML视图模板 46215.5 小结 464第16章 请求驱动的Spring Web MVC框架16.1 Spring Web MVC框架简介 46516.2 分发器 46616.3 处理器映射 46716.4 处理器拦截器 46816.5 控制器简介 47016.5.1 简单控制器 47016.5.2 控制器抽象基类 47116.5.3 命令控制器抽象基类 47216.5.4 表单控制器简介 47316.6 普通表单控制器 47316.6.1 普通表单控制器：视图流转 47316.6.2 普通表单控制器：基本配置 47516.6.3 普通表单控制器：控制器和处理流程 47616.6.4 普通表单控制器：表单、领域模型和视图 47916.6.5 普通表单控制器：验证器和错误采集 48216.7 向导式表单控制器 48216.7.1 向导式表单控制器：视图流转 48216.7.2 向导式表单控制器：基本配置 48516.7.3 向导式表单控制器：控制器、验证器和处理流程 48616.7.4 向导式表单控制器：表单、领域模型 48916.8 模型、视图和视图解析(View Resolving) 49016.8.1 返回视图名称 49016.8.2 保存模型属性 49016.8.3 视图解析器和视图抽象 49116.9 小结 492第17章 Spring和视图技术及Web框架的集成 49317.1 Spring集成JSP和JSTL 49317.1.1 JSP和JSTL简介 49317.1.2 视图解析器 49317.1.3 Spring的专有标签 49717.2 Spring集成XSLT 49917.2.1 XSLT简介 49917.2.2 示例开发：需求和部署配置 49917.2.3 示例开发：配置Web应用 50017.2.4 示例开发：把模型数据转换成XML文档 50017.2.5 示例开发：使用XSLT进行文档转换 50317.3 Spring集成PDF/Excel 50417.3.1 简介 50417.3.2 示例开发：部署、配置Web应用和模型数据采集 50417.3.3 示例开发：把模型数据转换成PDF和Excel视图 50517.4 Spring集成Struts 50817.4.1 集成概述和手法 50817.4.2 Spring应用上下文装载的两种方式 50917.4.3 使用Spring的ActionSupport类整合Struts 51017.4.4 使用Spring的DelegatingRequestProcessor覆写Struts的RequestProcessor 51217.4.5 使用Spring代理委托管理Struts Action 51417.4.6 使用Spring拦截Struts Action，替换宠物店的安全策略 51517.5 Spring集成WebWork2 51917.6 Spring集成Tapestry4 52217.6.1 Tapestry4背后的IoC容器：HiveMind 52217.6.2 部署环境和逻辑分层 52217.6.3 分层重构：引入表现层的业务注入，转移业务逻辑 52317.6.4 分层重构：引入业务层，实现业务逻辑 52417.6.5 分层重构：引入工具层，向Tapestry应用注入Spring上下文 52417.6.6 集成配置：Web应用配置、修改组件规范文件 52517.6.7 集成配置：通过HiveMind和工具类，衔接Spring 52617.7 小结 528第18章 Spring和单元测试 52918.1 模仿对象 52918.2 Spring Mock简介 52918.2.1 Spring Mock Web简介 53018.2.2 扩展JUnit框架的测试基类 53018.3 Spring Mock快速指南 53118.3.1 Web组件的单元测试：搭建测试环境 53118.3.2 Web组件的单元测试：视图转发 53318.3.3 Web组件的单元测试：会话状态 53318.3.4 Web组件的单元测试：简单逻辑 53418.3.5 事务性单元测试：使用Spring Mock事务基类搭建测试环境 53518.4 EasyMock快速指南 53818.4.1 EasyMock简介 53818.4.2 组件单元测试：搭建测试环境 53918.4.3 组件单元测试：模拟业务接口和领域对象的交互 54018.4.4 组件单元测试：模拟具体类和DAO的交互 54118.5 结合使用Spring Mock和EasyMock 54118.5.1 Web组件单元测试：模拟控制器和业务接口、领域对象的交互 54218.5.2 Web组件单元测试：重定向测试 545第19章 Spring和JMX技术 54719.1 JMX概述 54719.2 JMX的体系分层架构 54719.2.1 核心组件 54819.2.2 设备层(Instrumentation Level) 54919.2.3 代理层(Agent Level) 55019.2.4 分布服务层(Distributed Service Level) 55219.2.5 附加管理协议API 55219.3 JMX编程实践 55319.3.1 标准MBean编程：MBean接口和实现类 55319.3.2 标准MBean编程：JMX Agent 55419.3.3 标准MBean编程：手工和可视化注册MBean 55519.3.4 通知模型编程 55719.3.5 动态MBean和辅助元数据类型编程 56119.3.6 JMX连接器(Connectors)编程：基本流程 56219.3.7 JMX Connectors编程：准备MBean套件和监听器 56319.3.8 JMX Connectors编程：创建服务器 56519.3.9 JMX Connectors编程：创建客户端 56719.4 Spring对JMX的集成支持 56819.4.1 Spring集成JMX概述 56819.4.2 视Spring Bean为JMX MBean：简单的JavaBean配置 56919.4.3 视Spring Bean为JMX MBean：客户代码 57119.4.4 视Spring Bean为JMX MBean：自动侦测并且注册MBean 57219.4.5 视Spring Bean为JMX MBean

<<精通Spring>>

: ObjectName命名策略 57419.4.6 Spring的JSR-160 Connector支持: 服务器端配置和代码 57619.4.7
Spring的JSR-160 Connector支持: 客户端配置和代码 57819.4.8 Sping集成JMX技术展望 579第20章
从Spring宠物店看企业应用架构模式 58020.1 企业应用架构模式(POEAA)简介 58020.1.1 什么是
企业应用架构模式 58020.1.2 架构和分层 58020.1.3 企业应用 58020.2 宠物店和分层 58120.3
宠物店和领域逻辑 58220.3.1 如何组织领域逻辑 58220.3.2 事务脚本 58320.3.3 领域模型的基本
概念 58520.3.4 领域模型的持久 58820.3.5 服务层 58920.4 宠物店和数据源架构模式
59120.5 宠物店和Web表现模式 59220.5.1 模板视图 59220.5.2 转换视图 59220.5.3 两步视图
592

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>