

《Java应用架构设计》

图书基本信息

书名：《Java应用架构设计》

13位ISBN编号：9787111437680

10位ISBN编号：7111437683

出版时间：2013-9-1

出版社：机械工业出版社

作者：克内恩席尔德 (Kirk Knoernschild)

页数：251

译者：张卫滨

版权说明：本站所提供下载的PDF图书仅提供预览和简介以及在线试读，请支持正版图书。

更多资源请访问：www.tushu000.com

内容概要

编辑推荐

全球资深Java技术专家的力作，系统、全面地讲解如何将模块化设计思想引入开发中，涵盖18个有助于实现模块化软件架构的模式

媒体推荐

“基础永远不会过时。在本书中，Kirk介绍了如何立足基础，以低成本有效地构建高质量的软件密集型系统。你会发现这本书写得很好、很及时并且全是务实的理念。”

——Grady Booch，IBM院士

“与GoF的《设计模式》一样，Kirk的这本书应该成为每一位企业级开发人员和架构师的必备品，对Paremus的每一位工程师来说，这本书是必备读物。”

——Richard Nicholson，OSGi联盟主席、Paremus CEO

“通过写这本书，Kirk为软件社区做出了重要的贡献：他分享了自己关于模块化的大量真知灼见，这些内容新手能理解、在计算机课堂上能讲授并且有经验的程序员也可参考。我希望本书能够有广泛的读者。”

——Glyn Normington，Eclipse Virgo项目的领导者

“我们的行业需要开始思考模块化这个词——因而需要这本书！”

——Chris Chedgey，Structure 101创始人兼CEO

“在本书中，Kirk为我们提供了在现实世界中进行模块化软件开发所需要的设计模式。尽管模块化确实有助于管理复杂性和创建更容易维护的软件，但是天下没有免费的午餐。如果你想获得模块化所提供的收益，那么购买本书吧。”

——Patrick Paulin，Modular Mind咨询师和培训师

“Krik巧妙地记录了使用OSGi和Eclipse运行时技术的最佳实践。为了更好地理解如何创建优秀的软件，每一位高级Java开发人员都需要阅读本书。”

——Mike Milinkovich，Eclipse基金会执行总监

《Java应用架构设计》

作者简介

Kirk Knoernschild资深软件开发专家，系统架构师，软件开发项目经理，敏捷教练，软件开发咨询师。精通Java、OSGi、敏捷方法、系统架构等，拥有丰富的软件开发经验，担任过软件开发团队中的多个职位。同时他还是一位积极的开源社区贡献者，发表了大量经典文章，广受读者好评。曾受邀参与国际大型软件开发会议并发表演讲。著有《Java Design: Objects, UML, and Process》，合著有《No Fluff Just Stuff 2006 Anthology》。他曾培训和指导过成千上万的软件开发人员，讨论的主题涵盖Java/J2EE、建模、软件架构与设计、基于组件的开发、面向服务架构以及软件过程等。

张卫滨 资深软件开发专家，精通Java语言，对Java开源框架有较深入的研究。目前主要从事企业级软件的开发，拥有丰富的软件开发经验。熟悉Spring、Hibernate以及Eclipse等开源产品，对Web前端技术有一定的研究，熟悉相关技术以及dojo、jQuery、ExtJS等框架。技术社区的积极实践者，曾参与技术新闻和文章的翻译工作，已出版译著有《Spring实战（第3版）》。

书籍目录

本书赞誉

译者序

序

序

前言

第一部分 模块化的理由

第1章 模块定义

1.1 定义模块

1.1.1 可部署

1.1.2 可管理

1.1.3 可测试

1.1.4 原生可重用

1.1.5 可组合

1.1.6 无状态

1.2 软件模块的简洁定义

1.3 结论

第2章 模块化的两个方面

2.1 运行时模型

2.2 开发模型

2.2.1 编程模型

2.2.2 设计范式

2.3 模块化现状

2.4 结论

第3章 架构与模块化

3.1 定义架构

3.2 关于软件架构的一个故事

3.2.1 象牙塔

3.2.2 乌龟和塔

3.3 架构的目标

3.3.1 悖论

3.3.2 消除架构

3.4 模块化：被忽视的部分

3.5 回答我们的问题

3.6 结论

3.7 参考文献

第4章 征服复杂性

4.1 企业级复杂性

4.2 技术债

4.3 设计腐化

4.3.1 干扰可维护性

4.3.2 阻止可扩展性

4.3.3 抑制可重用性

4.3.4 限制可测试性

4.3.5 妨碍集成

4.3.6 阻碍理解

4.4 循环依赖

4.4.1 循环类型

- 4.4.2 悄然引入的循环
- 4.4.3 管理循环
- 4.4.4 循环总是不好的吗
- 4.5 结合点、模块和SOLID
- 4.6 管理复杂性
- 4.7 模块化的益处
- 4.8 结论
- 4.9 参考文献
- 第5章 实现重用
- 5.1 可用重用悖论
- 5.2 关于重用的免责声明
- 5.2.1 粒度
- 5.2.2 重量级
- 5.3 重用还是可用
- 5.4 模块化权衡
- 5.5 模块化设计
- 5.6 结论
- 5.7 参考文献
- 第6章 模块化与SOA
- 6.1 重新审视“自上而下”
- 6.2 粒度——架构师的强大对手
- 6.2.1 现实世界的一个例子
- 6.2.2 提升一个等级
- 6.2.3 另一个维度
- 6.2.4 全景图
- 6.2.5 服务样例
- 6.3 另一个视图
- 6.4 结论
- 第7章 参考实现
- 7.1 为什么不用OSGi
- 7.2 这个练习的背景：构建系统
- 7.3 初始版本
- 7.4 第一次重构
- 7.5 第二次重构
- 7.6 第三次重构
- 7.7 第四次重构
- 7.7.1 关于OSGi的好处
- 7.7.2 小结并准备下一次重构
- 7.8 第五次重构
- 7.9 第六次重构
- 7.10 第七次重构
- 7.11 事后剖析
- 7.11.1 关于模块测试
- 7.11.2 关于管理模块依赖
- 7.11.3 关于模块重用
- 7.11.4 关于构建
- 7.11.5 关于面向对象
- 7.12 结论
- 7.13 参考文献

第二部分 模式

第8章 基本模式

8.1 管理关系

8.1.1 表述

8.1.2 描述

8.1.3 多种实现

8.1.4 影响

8.1.5 样例

8.1.6 小结

8.2 模块重用

8.2.1 表述

8.2.2 描述

8.2.3 多种实现

8.2.4 效果

8.2.5 样例

8.2.6 小结

8.3 模块内聚

8.3.1 表述

8.3.2 描述

8.3.3 多种实现

8.3.4 效果

8.3.5 样例

8.3.6 小结

第9章 依赖模式

9.1 非循环关系

9.1.1 表述

9.1.2 描述

9.1.3 多种实现

9.1.4 效果

9.1.5 样例

9.1.6 小结

9.2 等级化模块

9.2.1 表述

9.2.2 描述

9.2.3 多种实现

9.2.4 效果

9.2.5 样例

9.2.6 小结

9.3 物理分层

9.3.1 表述

9.3.2 描述

9.3.3 多种实现

9.3.4 效果

9.3.5 样例

9.3.6 小结

9.4 容器独立

9.4.1 表述

9.4.2 描述

9.4.3 多种实现

- 9.4.4 效果
- 9.4.5 样例
- 9.4.6 小结
- 9.5 独立部署
 - 9.5.1 表述
 - 9.5.2 描述
 - 9.5.3 多种实现
 - 9.5.4 效果
 - 9.5.5 样例
 - 9.5.6 小结
- 9.6 参考文献
- 第10章 可用性模式
 - 10.1 发布接口
 - 10.1.1 表述
 - 10.1.2 描述
 - 10.1.3 多种实现
 - 10.1.4 效果
 - 10.1.5 样例
 - 10.1.6 小结
 - 10.2 外部配置
 - 10.2.1 表述
 - 10.2.2 描述
 - 10.2.3 多种实现
 - 10.2.4 效果
 - 10.2.5 样例
 - 10.2.6 小结
 - 10.3 默认实现
 - 10.3.1 表述
 - 10.3.2 描述
 - 10.3.3 多种实现
 - 10.3.4 效果
 - 10.3.5 样例
 - 10.3.6 小结
 - 10.4 模块门面
 - 10.4.1 表述
 - 10.4.2 描述
 - 10.4.3 多种实现
 - 10.4.4 效果
 - 10.4.5 样例
 - 10.4.6 小结
- 第11章 扩展性模式
 - 11.1 抽象化模块
 - 11.1.1 表述
 - 11.1.2 描述
 - 11.1.3 多种实现
 - 11.1.4 效果
 - 11.1.5 样例
 - 11.1.6 小结
 - 11.2 实现工厂

11.2.1 表述

11.2.2 描述

11.2.3 多种实现

11.2.4 效果

11.2.5 样例

11.2.6 小结

11.3 分离抽象

11.3.1 表述

11.3.2 描述

11.3.3 多种实现

11.3.4 效果

11.3.5 样例

11.3.6 小结

11.4 参考文献

第12章 通用模式

12.1 就近异常

12.1.1 表述

12.1.2 描述

12.1.3 多种实现

12.1.4 效果

12.1.5 样例

12.1.6 小结

12.2 等级化构建

12.2.1 表述

12.2.2 描述

12.2.3 多种实现

12.2.4 效果

12.2.5 样例

12.2.6 小结

12.3 测试模块

12.3.1 表述

12.3.2 描述

12.3.3 多种实现

12.3.4 效果

12.3.5 样例

12.3.6 小结 201

第三部分 模块化架构模式与OSGi

第13章 OSGi简介

13.1 一点历史

13.2 OSGi所能带来的收益

13.2.1 模块化开发

13.2.2 管理依赖

13.2.3 模块平台

13.2.4 版本化的bundle

13.2.5 动态（重）部署

13.2.6 环境相关的控制

13.3 深入理解OSGi

13.4 OSGi bundle

13.4.1 bundle状态

- 13.4.2 OSGi Service
- 13.5 OSGi运行时管理
- 13.6 重新查看模块化的两个方面
- 13.7 OSGi与模式
 - 13.7.1 管理依赖
 - 13.7.2 动态性
 - 13.7.3 Blueprint规范
- 第14章 贷款样例与OSGi
 - 14.1 起步
 - 14.2 清单文件
 - 14.3 Service
 - 14.3.1 Blueprint服务
 - 14.3.2 贷款样例配置
 - 14.3.3 OSGi Service声明
 - 14.4 安装与执行
 - 14.5 结论
- 第15章 OSGi与Scala
 - 15.1 起步
 - 15.2 Scala代码
 - 15.3 Scala bean配置
 - 15.4 Scala Service配置
 - 15.5 构建Scala模块
 - 15.6 安装与执行
 - 15.7 结论
- 第16章 OSGi与Groovy
 - 16.1 起步
 - 16.2 Groovy代码
 - 16.3 Groovy bean配置
 - 16.4 Groovy Service配置
 - 16.5 构建Groovy模块
 - 16.6 安装与执行
 - 16.7 结论
- 第17章 OSGi的未来
 - 17.1 将OSGi作为推动者
 - 17.2 颠覆性
 - 17.3 生态系统的威力
 - 17.3.1 生态系统与模块化的两个方面
 - 17.3.2 基于组件的开发（CBD）不是已经成功了吗
 - 17.4 生态系统
 - 17.5 结论
- 附录A 类设计的SOLID原则

精彩短评

1、浙江图书馆

2、拿到书之后，一个周就看完了，之后的一个周又详细看了一遍之前标注的章节，收获颇丰。书中提及的模式并不高深，却可以产生很多共鸣，尤其是之前尝试去做过模块化的童鞋，如果经历过模块化方案的纠结之后，再回头看这本书会感觉有很多共鸣，理解也会更加深刻。虽然书中的例子是以Java为例，但是我相信这些思想并不局限于JAVA，而是语言无关的，书中也谈及了OSGi，但是不要认为是OSGi的强关联，相反，OSGi只是一种模块化环境，让你的模块化做得更彻底极致。还是先体会下模块化的思想哈

3、啰嗦累赘，不痛不痒

4、一般吧，还不适合我

5、和标题不符，没有具体讲osgi的知识，反而更多的是设计相关的知识。而且都是较为简单的设计知识。

6、本来想了解osgi的全篇的理论 差评

7、我好像在阅读历史，一些陈旧的理论。固然有用，但对我而言，显得并不是那么适合了。就此书而言，至少对于中文版出版的时间，可谓生不逢时。

精彩书评

1、书中关于软件架构的核心思想是：模块化，分层，等级化，消除循环依赖等。前面章节，反复强调分层设计，层中等级化调用依赖。输入依赖优于输出依赖。最后的章节，作者希望结合OSGi，介绍和讲解前面论述的内容，OSGi实质内容不多，不能期望从中得到OSGi的详细知识。借用java具体技术语言，对于.net阵营的开发者，其中的软件架构观点是相通的，一致的。书中的理论，在其他经典著作中已有论述，对于涉猎过其他书籍的读者来说，多数内容是似曾相识的。作者着重强调的内容，是系统架构中普遍的共识问题，新意不多，经验之谈。该书通过他人博文中推荐得知，推为经典。通读之后，余不以为然，明辨之，慎言之。

章节试读

1、《Java应用架构设计》的笔记-第16页

最大化灵活度，管理复杂度

2、《Java应用架构设计》的笔记-第41页

粗粒度的模块更容易使用，而细粒度的模块具备更高的可重用性。

3、《Java应用架构设计》的笔记-第28页

严重依赖的一个原因就是没有正确的使用抽象，没有进行抽象的地方很难进行扩展。

4、《Java应用架构设计》的笔记-第40页

Web框架、ORM框架以及安全框架，还有很多不能叫上名的框架。但是大多数这样的框架都是水平的，而不是垂直的。也就是说，他们处理的问题与特定基础设施相关并会带来冗长的代码，而不会处理业务问题。我想要明确关注的是垂直方向的重用，因为这是我们长期以来一直没有得到的灵丹妙药。

5、《Java应用架构设计》的笔记-第34页

伴随着灵活性会带来复杂性。这就产生了一个问题：“最合适使用SOLID原则的地方究竟在哪里？”。

6、《Java应用架构设计》的笔记-第39页

最大化重用会使得可用复杂化。

7、《Java应用架构设计》的笔记-第27页

就像财务债一样，技术债也需要支付利息，不过它的形式是因为匆忙和脏乱的设计选择，我们在将来的开发中要付出额外的努力。我们可以选择继续支付利息，或者可以用更好的设计重构匆忙和脏乱的设计以偿还本金。尽管偿还本金的操作需要成本，但是会降低将来要支付的利息。

8、《Java应用架构设计》的笔记-第83页

具备大量输入依赖的模块应该是很稳定的

9、《Java应用架构设计》的笔记-第14页

架构不仅是一些技术理念，它也是一个社会性的结构。通过架构的社会性方面，我们可以拟合架构师和开发人员之间的分歧。

10、《Java应用架构设计》的笔记-第41页

重量级指的是模块依赖其环境的程度。重量级的模块依赖其操作环境，而轻量级的模块会避免这些依赖。轻量级的模块更易重用，但是重量级的模块更易使用。

11、《Java应用架构设计》的笔记-第39页

最大化重用会使得可用复杂化。

12、《Java应用架构设计》的笔记-第4页

模块成熟度模型：

- 1.未管理的/混乱的
- 2.管理依赖
- 3.适当隔离
- 4.修改代码库以最小化耦合
- 5.面向服务的架构

13、《Java应用架构设计》的笔记-第11页

架构就是一系列重要的决策，这些决策涉及软件系统的组织，组成系统的结构化元素及其接口的选择，元素之间协作时特定的行为，结构化元素和行为元素形成更大子系统的组合方式 + 架构风格

14、《Java应用架构设计》的笔记-第2页

软件模块是可部署的、可管理的、原生可重用的、可组合的、无状态的软件单元，它为用户提供了简洁的接口。

15、《Java应用架构设计》的笔记-第64页

接口要接近使用他们的类，远离实现他们的类

16、《Java应用架构设计》的笔记-第59页

如果真得有分层的系统，那么我可以将每层拆分为一个独立的模块，而且上层的模块依赖较低层次的模块，但是不能相反。

17、《Java应用架构设计》的笔记-第25页

随着系统的演化，它的复杂性会增加，除非你做一些工作对其进行维护和缩减。

18、《Java应用架构设计》的笔记-第50页

如果担心所创建的服务过于粗粒度并且无法最大化其重用潜能，那么我可以将服务行为拆分为粒度更小且更易服用的模块。然后，可以将模块组合为服务并且在服务间对模块进行重用。所带来的结果就是不同的实体具备不同等级的粒度，进而为组合，使用和重用软件实体带来巨大的灵活性。

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:www.tushu000.com