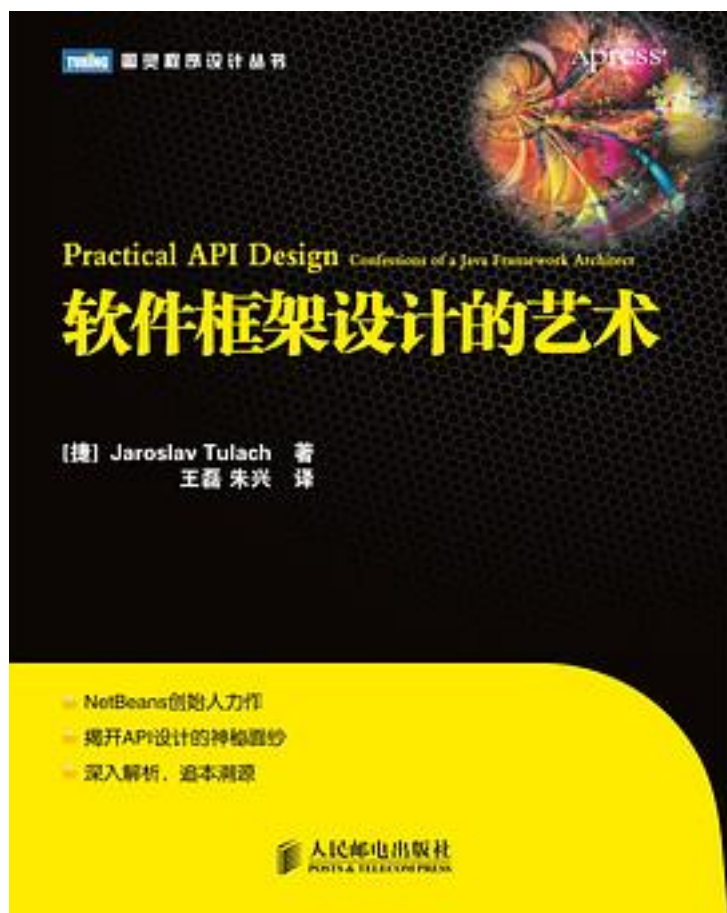


软件框架设计的艺术



[软件框架设计的艺术_下载链接1](#)

著者:[捷] Jaroslav Tulach

出版者:人民邮电出版社

出版时间:2011-3

装帧:平装

isbn:9787115248497

本书帮助你解决API设计方面的问题，共分3个部分，分别指出学习API设计是需要进行科学的训练的、Java语言在设计方面的理论及设计和维护API时的常见情况，并提供了各种技巧来解决相应的问题。

本书作者是NetBeans的创始人，也是NetBeans项目最初的架构师。相信在API设计中遇到问题时，本书将不可或缺。

本书适用于软件设计人员阅读。

作者介绍:

Jaroslav Tulach

NetBeans的创始人，也是NetBeans项目最初的架构师。有着丰富的项目开发经验，一直致力于如何提高开发人员的设计技巧，从而保证了NetBeans项目的成功。

目录: 第一部分 理论与理由

第1章 软件开发的艺术 4

1.1 理性主义，经验主义以及无绪 4

1.2 软件的演变过程 6

1.3 大型软件 8

1.4 漂亮，真理和优雅 9

1.5 更好的无绪 12

第2章 设计API的动力之源 14

2.1 分布式开发 14

2.2 模块化应用程序 16

2.3 交流互通才是一切 20

2.4 经验主义编程方式 22

2.5 开发第一个版本通常比较容易 24

第3章 评价API好坏的标准 26

3.1 方法和字段签名 26

3.2 文件及其内容 27

3.3 环境变量和命令行选项 29

3.4 文本信息也是API 30

3.5 协议 32

3.6 行为 35

3.7 国际化支持和信息国际化 35

3.8 API的广泛定义 37

3.9 如何检查API的质量 37

3.9.1 可理解性 37

3.9.2 一致性 38

3.9.3 可见性 39

3.9.4 简单的任务应该有简单的方案 40

3.9.5 保护投资 40

第4章 不断变化的目标 42

4.1 第一个版本远非完美 42

4.2 向后兼容 43

4.2.1 源代码兼容 43

4.2.2 二进制兼容 44

4.2.3 功能兼容——阿米巴变形虫效应 50

4.3 面向用例的重要性 52

4.4 API设计评审 55

4.5 一个API的生命周期 56

4.6 逐步改善 60

第二部分 设计实战

第5章 只公开你要公开的内容 67

5.1 方法优于字段 68

- 5.2 工厂方法优于构造函数 70
- 5.3 让所有内容都不可更改 71
- 5.4 避免滥用setter方法 72
- 5.5 尽可能通过友元的方式来公开功能 73
- 5.6 赋予对象创建者更多权利 77
- 5.7 避免暴露深层次继承 82
- 第6章 面向接口而非实现进行编程 85
 - 6.1 移除方法或者字段 87
 - 6.2 移除或者添加一个类或者接口 88
 - 6.3 向现有的继承体系中添加一个接口或者类 88
 - 6.4 添加方法或者字段 88
 - 6.5 Java中接口和类的区别 90
 - 6.6 弱点背后的优点 91
 - 6.7 添加方法的另一种方案 92
 - 6.8 抽象类有没有用呢 94
 - 6.9 要为增加参数做好准备 95
 - 6.10 接口VS.类 97
- 第7章 模块化架构 98
 - 7.1 模块化设计的类型 100
 - 7.2 组件定位和交互 103
 - 7.3 编写扩展点 116
 - 7.4 循环依赖的必要性 117
 - 7.5 满城尽是Lookup 121
 - 7.6 Lookup的滥用 126
- 第8章 设计API时要区分其目标用户群 129
 - 8.1 C和Java语言中如何定义API和SPI 129
 - 8.2 API演进不同于SPI演进 131
 - 8.3 java.io.Writer这个类从JDK 1.4到JDK 5的演进 131
 - 8.4 合理分解API 143
- 第9章 牢记可测试性 147
 - 9.1 API设计和测试 148
 - 9.2 规范的光环正在褪去 151
 - 9.3 好工具让API设计更简单 153
 - 9.4 兼容性测试套件 155
- 第10章 与其他API协作 158
 - 10.1 谨慎使用第三方API 158
 - 10.2 只暴露抽象内容 162
 - 10.3 强化API的一致性 164
 - 10.4 代理和组合 168
 - 10.5 避免API的误用 176
 - 10.6 不要滥用JavaBeans那种监听器机制 180
- 第11章 API具体运行时的一些内容 184
 - 11.1 不要冒险 186
 - 11.2 可靠性与无绪 189
 - 11.3 同步和死锁 191
 - 11.3.1 描述线程模型 192
 - 11.3.2 Java Monitors中的陷阱 193
 - 11.3.3 触发死锁的条件 196
 - 11.3.4 测试死锁 201
 - 11.3.5 对条件竞争进行测试 204
 - 11.3.6 分析随机故障 206
 - 11.3.7 日志的高级用途 208
 - 11.3.8 使用日志记录程序控制流程 210
 - 11.4 循环调用的问题 215

- 11.5 内存管理 218
- 第12章 声明式编程 223
 - 12.1 让对象不可变 225
 - 12.2 不可变的行为 229
 - 12.3 文档兼容性 230
- 第三部分 日常生活
- 第13章 极端的意见有害无益 236
 - 13.1 API必须是漂亮的 237
 - 13.2 API必须是正确的 237
 - 13.3 API应该尽量简单 240
 - 13.4 API必须是高性能的 242
 - 13.5 API必须绝对兼容 242
 - 13.6 API必须是对称的 245
- 第14章 API设计中的矛盾之处 247
 - 14.1 API设计中的自相矛盾 248
 - 14.2 背后隐藏的工作 251
 - 14.3 不要害怕发布一个稳定的API 252
 - 14.4 降低维护费用 255
- 第15章 改进API 258
 - 15.1 让有问题的类库重新焕发活力 259
 - 15.2 自觉地升级与无意识地被迫升级 265
 - 15.3 可选的行为 268
 - 15.4 相似API的桥接和共存 274
- 第16章 团队协作 286
 - 16.1 在提交代码时进行代码评审 286
 - 16.2 说服开发人员为他们的API提供文档 290
 - 16.3 尽职尽责的监控者 292
 - 16.4 接受API的补丁 297
- 第17章 利用竞赛游戏来提升API设计技巧 300
 - 17.1 概述 300
 - 17.2 第一天 301
 - 17.2.1 非public类带来的问题 304
 - 17.2.2 不可变性带来的问题 304
 - 17.2.3 遗漏实现的问题 308
 - 17.2.4 返回结果可能不正确的问题 309
 - 17.2.5 第一天的解决方案 310
 - 17.3 第二天 313
 - 17.3.1 我想修正犯下的错误 316
 - 17.3.2 第二天的解决方案 317
 - 17.4 第三天：评判日 320
 - 17.5 也来玩下这个游戏吧 327
- 第18章 可扩展Visitor模式的案例 328
 - 18.1 抽象类 331
 - 18.2 为改进做好准备 333
 - 18.3 默认的遍历 334
 - 18.4 清楚地定义每个版本 337
 - 18.5 单向改进 339
 - 18.6 使用接口时的数据结构 340
 - 18.7 针对用户和开发商的Visitor模式 341
 - 18.8 三重调度 343
 - 18.9 Visitor模式的圆满结局 345
 - 18.10 语法小技巧 346
- 第19章 消亡的过程 348
 - 19.1 明确版本的重要性 349

- 19.2 模块依赖的重要性 349
- 19.3 被移除的部分需要永久保留吗 352
- 19.4 分解庞大的API 352
- 第20章 未来 356
 - 20.1 原则性内容 357
 - 20.2 无绪长存 358
 - 20.3 API设计方法论 360
 - 20.4 编程语言的演变 361
 - 20.5 教育的作用 363
 - 20.6 共享 365
- 参考书目 366
- • • • • [\(收起\)](#)

[软件框架设计的艺术_下载链接1_](#)

标签

架构

设计

框架

API

编程

软件开发

计算机

Java

评论

总的来说框架可以这样看,一来是为了规范团队开发的,相同的命名规范,相同的接口,相同

的实现方式,二来是为了分离部分底层逻辑的,直接使用更加抽象的思维进行开发,忽略底层的io,数据库,类库加载之类的操作,基本上可以说是为了实现敏捷开发,三来么可以说是留人之策,把你培训成代码工人,你想走也无处可去,毕竟依靠框架做项目的开发人员待遇都不会很高的,除非你转去做架构之类的,但是这些东西你在使用框架的时候是根本接触不到的,做久了员工也就不怎么想跳了,留住50%的老员工绝对没有什么问题,四来就是新人上手快,即使这娃再怎么笨,框架良好的封装,最小化的用户接口也能让他基本上不怎么犯错,即便不了解深层的原理也可以实现很多功能,这样一来降低了成本(人力资源成本),毕竟新人的薪水都不怎么高的说,除非是天才型的

是本好书，值得慢慢看，但我只花一天看完，眼睛又涨又难受，shit

以『无绪』作为API的终极目标。

本书原著带有Java社区偏见、内容冗长，翻译还很晦涩。但对于初学编程关心API设计，有志于成为架构师的人，本书有用。
对于其他人嘛——本书治疗失眠效果好，建议睡前看。
我看这书能把我大脑从亢奋状态迅速变成想睡状态，合上书就睡着了。第二天早起精神好，吃嘛嘛香。

#等我写java了一定再回来看这本书=.=

#到底是原文，还是翻译的问题，部分章节真啰嗦。

不好看

更正下...翻译很水...

这书不错

其实是API设计的艺术。作者是NetBeans的创始人。本书是他多年经验的分享，非常精彩。在声明式编程里，作者也表示了对函数式语言的推崇，比如函数式语言可以通过不可变的数据结构来避免死锁。让我对函数式语言的好奇心又增加了，有机会要看看。

有闪光点，也有不靠谱的长篇累牍。不少示例代码过于随意。

这本书很多地方还是有些超出我的知识范围。希望以后能重新读一遍。这本书真的很好，而且我也花了太多的时间去读。平均下来估计一个小时不到30页。

写得有点乱，不够深入浅出，不是很易懂。条理也有些不清。或许是自己功力还不够吧

这书写的太催眠了，以至于距离第一次翻这书已经过去了好几年

这本书主要教程序员怎么设计 API。作者表达中包含很多隐喻以及举例多与 Netbean 相关，加之翻译的原因，很多细节没看明白，后面有机会再翻阅一遍。

以为是讲框架设计的，其实是讲API设计的

更加偏向于API的设计

这其实是一本教你如何在开发中与人保持原则和平衡的书，它描述了代码作为服务提供者的原则和策略。
由于内容绝大部分都来自开发一线，因此推荐大家在读本书之前扎实一下JDK相关的基础和核心特性。
结构上，一些来自特别语境的细节对主体框架带来了干扰，需要读者在上下文中保持更多的前提，在一定程度上降低了本书的易读性

太跟NetBeans相关了，很多地方没太读懂。

不太适合初学者阅读

[软件框架设计的艺术 下载链接1](#)

书评

从书名Practical就看出来这是一本和实践相结合的书，另外作者是Netbeans的创始人，这里的Practical是他十多年开发Netbeans的经验总结，推荐！另外推荐Effective Java作者Joshua Bloch的Presentation: How to Design a Good API & Why it Matters。 <http://www.infoq.com/prese...>

先不论作者的文章如何，就译者的认真和努力，就值得看，对于原作者的一些内容，译者都增加了自己的说明，书整体阅读非常流畅，在看过的翻译的书，这个算得上是上乘之作了。抱歉，你的评论太短了 好吧，其实我想称赞的译者，如果有一些java基础或设计模式的基础，看这本书...

不知有多少人和我一样，对自己日常使用的开发框架和IDE的作者充满敬意，对它们的开发过程充满好奇。如果你也使用过NetBeans，曾把它当作日常IDE，那么你应该会对《软件框架设计的艺术》感兴趣，因为其中包含了NetBeans创始人Jaroslav Tulach在设计NetBeans过程中总结出来的经验...

本书的作者是NetBeans的创始人，而我和NetBeans结缘始于2011年ACM的东北赛区竞赛。当时竞赛方东北大学给我们提供的是Ubuntu + Eclipse + NetBeans，其中Eclipse没有挂CDT，而我当时不知到NetBeans已经支持C++了，所以看见这个配置非常纠结，差点就有用Make的冲动了。可以说这...

早在英文版还没有翻译的时候就关注这本书，对于书中所立足的角度实在是非常的难能可贵的，软件开发领域一直存在着很多所谓下意识的，凭感觉的，不可捉摸的工作内容

，就像api的设计，要把这样的问题阐释清楚实在不是易事。
是原作者对于探索问题的热忱才使得那些模糊难以描述的...

学习就好比打仗，如果《XX之美》相当于纸上谈兵，《XX架构模式》相当于冲锋陷阵，那么这本书就是战后修整。能够规范你的设计，让你少走弯路，让你不至于迷失在模式和新技术中。 ===== 抱歉，你的评论太短了
=====

英文书名是：Practical API Design: Confessions of a Java Framework Architect。Practical API Design这个才是真正的书名。

以扯淡为主，轻松好看，不要指望是一本很有含量的书，就象闲侃，你不要要求那么多，牛B的人跟你闲侃，不要想从中得到诸多专业的知识
字数不够，好吧，总结下：这本书是闲谈某个软件开发的架构的一些问题，相当于论坛帖子集合 够了吗？

正如该书“阅读指南”中所说，该书比较啰嗦。不过，老外有此风格的不是一个、两个，忍了。
全书的主题紧紧围绕”无绪“的概念进行，所谓”无绪“，就是指某些事情并不需要对背后的原理、规则有深刻的理解，就可以使用。典型的，不懂得汽车的原理，但我们照样开车，而且开得还...

正如本书作者在序言中问到“仅仅是又多了一本设计书吗？”作者相信本书的存在“自有其必要性”，原因在于本书探讨的设计领域是如此的卓尔不群，却又是Java程序员在开发中必须要面对的问题，那就是框架的设计，API的设计。
我自认为对面向对象设计的掌握已经深入骨髓，对设计...

买之前先到豆瓣来看看，发现有位“胖子”同学的评论说翻译的好。
于是下决心买了。不过…… 1. 排版问题。书到手打开一看，晕，满页满页的黑块。你388页的书卖75，就不能把版面好好整整么？至于这样省纸啊？ 2.翻译问题。不能说译者不认真，但我个人感觉是译者因为...

大部分的篇幅在扯淡，感觉整个书的主题是如何设计前后兼容的API。大部分在讲一些

设计API的小技巧，基本没有涉及到如何设计完整的软件框架。没看过的没必要花时间看了，还不如看看这个：Java API Design Checklist，<http://dongxi.net/b14gn>。

[软件框架设计的艺术 下载链接1](#)