HZ BOOKS
PEARSON

开发人员专业技术丛书

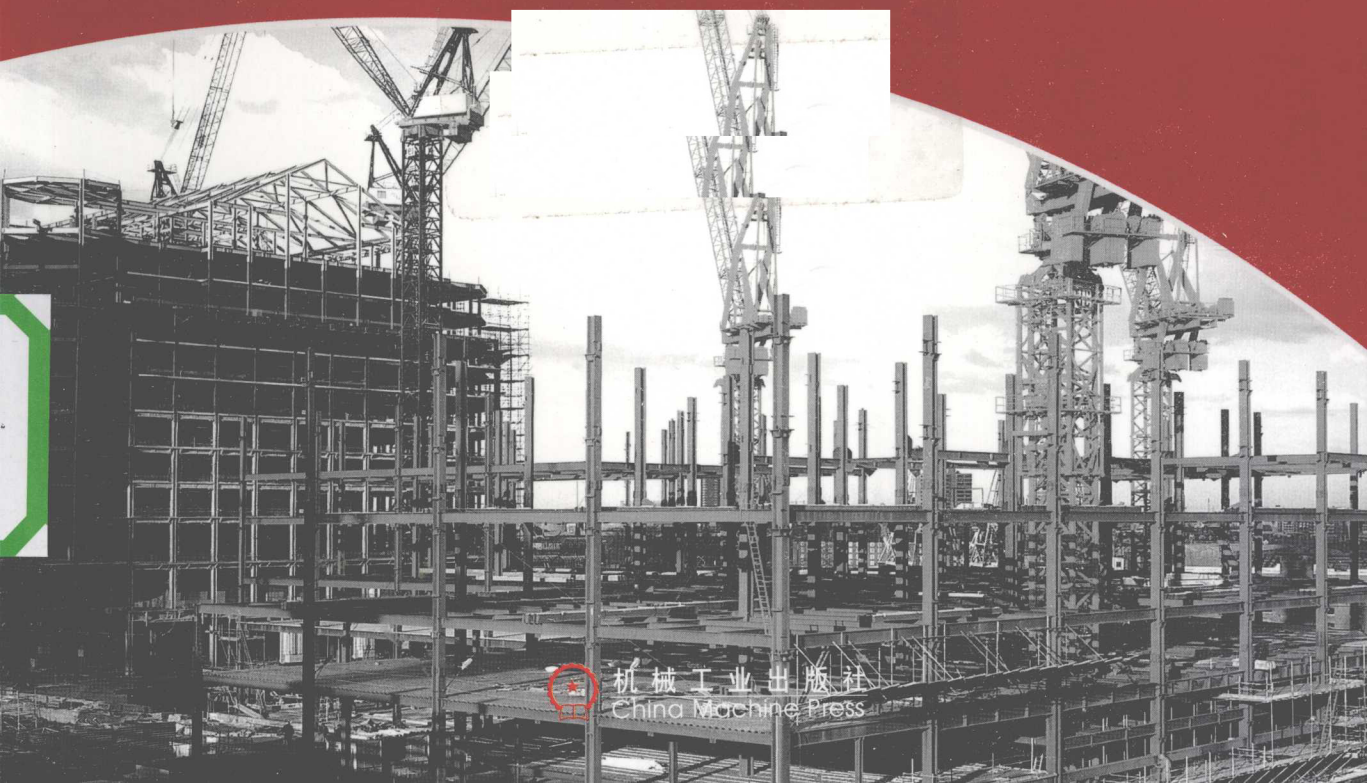
经典图书新版 以Ruby语言为实例

Refactoring: Ruby Edition

重构（Ruby版）

Jay Fields
（美） Shane Harvie 等著
Martin Fowler

徐旭铭 译



机械工业出版社
China Machine Press

开发人员专业技术丛书

Refactoring: Ruby Edition

重构（Ruby版）

Jay Fields
（美）Shane Harvie 等著
Martin Fowler
徐旭铭 译

机械工业出版社
China Machine Press

本书是一本专门为 Ruby 程序员编写的重构指导。它的目标是向读者展示如何以一种既受到严格控制又高效的方式进行重构。读者将学习到不会在代码中引入 bug 并能按部就班地改进结构的重构方式。本书的主要内容：第 1 章指出重构是什么；第 2 章讨论进行重构的理由；第 3 章指出需要进行重构的标志；第 4 章讨论测试在重构中扮演的重要角色；第 5 章 ~ 第 12 章介绍了重构花名册，它包含了在重构领域里到目前为止的成果。当需要进行某项任务时，这份花名册可以手把手地提醒我们安全的做法。

本书的目标读者是专业的 Ruby 程序员、资深设计师和架构师。

本书版权登记号：图字：01-2009-7725

图书在版编目(CIP)数据

重构：Ruby 版/(美)菲尔德斯(Fields, J.)等著；徐旭铭译. —北京：机械工业出版社，2010.4
(开发人员专业技术丛书)
书名原文：Refactoring: Ruby Edition

ISBN 978-7-111-30078-6

Ⅰ. 重…　Ⅱ. ①菲…　②徐…　Ⅲ. 计算机网络 - 程序设计　Ⅳ. TP393.09

中国版本图书馆 CIP 数据核字(2010)第 043719 号

机械工业出版社（北京市西城区百万庄大街 22 号　邮政编码　100037）
责任编辑：秦　健

北京京北印刷有限公司印刷
2010 年 4 月第 1 版第 1 次印刷
186mm × 240mm · 19.5 印张
标准书号：ISBN 978-7-111-30078-6
定价：49.00 元

凡购本书，如有缺页、倒页、脱页，由本社发行部调换
客服热线：(010) 88378991；88361066
购书热线：(010) 68326294；88379649；68995259
投稿热线：(010) 88379604
读者信箱：hzjsj@hzbook.com

译 者 序

本书英文版的出版距离第一本《重构》出版已经有十年了，即使是中文版也有六年之久。侯捷和熊节的译本质量很高，只是当时我还在大学里懵懵懂懂，读了几页就扔掉了，知道它在讲什么，但是不知道要怎么用。另一本让我有这种感觉的则是《设计模式》。这样的书并不难读，作者能以非常浅显的方式把问题讲解出来，但是要真正读懂其中的内容，却不是一朝一夕可以做到的。正如书中最后提到的，本书中所讲解的只是基本技术，它们彼此之间是相对孤立的，要做到融会贯通还需要大量的实践经验。正所谓“学而不思则罔，思而不学则殆”。

工作以后渐渐接触到一些大型系统，有些代码甚至比我的年龄还要长，当时看起来聪明的设计已经不太适合现在的要求了，这时再读它们的感觉自然大不一样。前辈们总结了多年的经验，让我们可以不必摸着石头过河。好的软件不是写出来的，而是改出来的。设计模式能让你保持在大方向上不会脱轨，而重构则能让你在面对变化时不至于手足无措。

这本书是为 Ruby 而写，所有的示例都替换成了 Ruby。作为一门灵活的动态语言，Ruby 很多时候都不需要像在 Java 里重构那样束手束脚。另外，很多现代的 IDE 都已经具备了基本的重构能力，可以让你避免一些烦琐的复制粘贴。因此在第 1 版中有些显得相当繁复的步骤在 Ruby 里变得十分简练，这就让我们可以更加关注重构的本质，不会陷入语言的细枝末节。因此，虽然在这本书里针对十年来重构技术的进步进行了相应的更新，并且添加了一些针对 Ruby 的内容，但是页数并没有超出第 1 版。

前面说到，《重构》第 1 版的中译本水平很高，在翻译本书的过程中，我自然倍感压力，不敢砸了前人的招牌。在此感谢陈冀康老师给我这个成为本书译者的机会。由于水平所限，错误和不妥之处在所难免。恳请专家和同行批评指正，也希望使用本书的读者给予批评和建议，不胜感激。

徐旭铭

2010 年初

序

我还能清晰地记得自己当初学习面向对象编程的情形。在学习的过程中，我总是有一种莫名的紧张——总觉得少了点什么。有些新概念看起来那么简单、那么熟悉，就好像在告诉你其实它们还有更深的内涵等待你发掘。这实在是种让人挠心的感觉啊。

在阅读设计模式的文献时候我总是饶有兴致，但是遗憾的是读完以后却得不到什么启迪。我去和不同的程序员交谈，浏览各种网站，阅读大量的书籍和伪代码，可是仍然觉得自己忽略了什么重要的东西。虽然我能理解面向对象工具的工作方式，但是却没办法通过一种我觉得是正确的方法去使用它们。

最终，我拿起了本书的第 1 版。

软件并非灵光一闪就能创造出来的东西。人们总是把注意力放在如何组织开发过程上，而忘记了其实软件开发本身就是一个过程。说得再准确一点，正如《重构》所教诲的，它是一系列细小的决策和行为，而这些决策和行为，正是由渴望创造出色产品的欲望和价值观所驱使的。

只要理解了软件开发是一项持续的活动而非静态的事件，就能帮助我们记住代码也应该是有生命的。好的代码是容易修改的代码。而坏的代码则可以通过不断地改进变得易于修改。在容易修改的代码上工作是非常愉悦的。而在难以改动的代码上工作则令人崩溃。如果不进行重构，你做的修改越多，要面对的压力就会越大。

因此要成为一名软件程序员，光知道什么是好代码还不够，更重要的是要知道怎样让代码变好。软件可不是从石头里蹦出来的，它是我们在键盘上一点一点敲出来的。而重构这本书则教我如何出色地完成这一过程。它教会了我怎样坐下来循序渐进地写出漂亮的代码。

当我第一次读到《重构》的时候，我身在一个小团队里，它的任务是要帮助更大的团队写出更好的软件。在开会和代码复审的时候，我都会随身带着这本书，既当矛又当盾。我非常热爱自己的工作和（更重要的是）软件开发的技艺，我相信和我一起工作的很多程序员在看到我夹着这本书冲向他们的小隔间的时候，一定都非常惧怕我的眼神。其实在这些会议上我并没有经常引用书中的内容，我带着它只是因为它能提醒我们：只要我们记住我们的工作应该是完美的，并且通过各种步骤让它变得完美，那么它就一定可以变得更好。

总结过去的经历，我逐渐意识到我们过去使用的语言和工具其实都是在和我们作对。原本这本书里的技术是为了 Smalltalk 开发而产生的，而让重构大放异彩的却是在动态环境里，因此

我们决定要让它们在 Ruby 里浴火重生。作为一名 Ruby 的坚定支持者，看到这样一本能产生深远意义的书采用 Ruby 来作为主要语言进行描述时，是多么激动人心啊。

本书作为新一代 Ruby 程序员的指路明灯，不但能教会他们创建更好更灵活的软件，（我希望）更可以让他们和我一样爱上软件开发这门技艺。

Chad Fowler
Co-Director, Ruby Central, Inc.
CTO, InfoEther, Inc.

前　言

差不多十年前，我（Martin）曾经和 Kent Beck 一起做过一个项目。这个项目的名字叫 C3，它后来成为极限编程诞生的标志性项目，并帮助我们看清了敏捷软件运动的方向。

我们从那个项目里学到了很多东西，不过真正震撼到我的是 Kent 那种有条不紊、持续不断改进系统设计的风格。一直以来我对编写干净的代码都抱有极大的热情，坚信花时间去清理有问题的代码，以便让团队能更快捷地开发功能是非常有价值的事情。而 Kent 向我介绍了一种很多顶尖 Smalltalk 程序员使用的技术，它能让我的工作效率成倍提升。这是一种他们称之为重构的技术，我很快就变得想要在任何场合下都把它介绍给别人。但是市面上没有任何出版物或是类似的资源可以让我指引人们去自己学习这项技术。既然 Kent 和其他 Smalltalk 程序员都没意愿要写一本，所以我就决定自己动手了。

结果我的那本《重构》大受欢迎，在重构成为主流技术的过程中看起来还扮演了相当重要的角色。随着近年来 Ruby 的兴起，给这本书写一本 Ruby 版是很有意义的，为此我拉来了 Jay 和 Shane。

什么是重构

重构是改变软件系统的过程，它不会改变代码的外部行为，但是可以改善其内部结构。它清理代码的严谨方式能把引入 bug 的风险降至最低。基本上当你进行重构的时候，就意味着代码的设计在完成时会得到改善。

很多人觉得“代码的设计在完成时会得到改善”这种说法相当古怪。多年来大多数人都相信设计第一、编码第二的原则。而随着时间的推移，不断修改代码以及系统的完整性后，原本设计的结构也会慢慢变得模糊。代码逐渐从一项工程活动沦落为敲敲打打的修补工作。

重构与此正好相反。有了重构，你可以把一个糟糕甚至混乱的设计，逐渐转变成设计良好的代码。每一个步骤都非常简单，甚至有点过分简单了。比如把一个实例变量从一个类移到另一个类，从一个方法里抽出一些代码单独放到一个方法里去，以及在层次体系之间移动一些代码等。但是这些小改动累积起来却能够彻底改进设计。这和通常认为的软件衰败论的观念是完全相反的。

在重构的时候你会发现工作的重心发生了变化。设计不再是最先进行，而是在开发过程中不断进行的。你会从构建系统中学习到如何改进设计。这种交互能让程序的设计随着开发工作的进行一直保持在较好的水准上。

本书的内容

本书是一本专门为职业 Ruby 程序员编写的重构指导。我们的目标是要向你展示怎样以一种受到严格控制同时又高效的方式来进行重构。你会学习到不在代码里引入 bug 并能按部就班地改进结构的重构方式。

通常一本书都是以介绍来开头的。虽然我对此并无异议，但是我发现要用泛泛而谈或是抽象的定义来介绍重构并不容易。所以我们还是先举个例子吧。第 1 章会给出一个包含了常见设计错误的小程序，然后将它重构成一个比较能让人接受的面向对象程序。我们会看到重构过程以及好几种重构技术的应用。如果你想要理解重构究竟是什么，那就绝对不能错过这一章。

在第 2 章里，我们讨论了重构中一些普遍的原则、定义，以及进行重构的理由。我们还会讲到重构里存在的一些问题。在第 3 章里，Kent Beck 会讲解如何在代码里寻找坏味道，以及如何通过重构来清理它们。测试在重构中扮演着非常重要的角色，因此第 4 章会讨论如何通过一个简单的测试框架将测试织入代码中去。

第 5 章～第 12 章覆盖了重构的花名册，这也是本书的精华所在。这并非一份完整的花名册，只能说是花名册的一部分。它包含了我们在重构这个领域里到目前为止的成果。当我们需要进行某项任务，比如说用多态替换条件逻辑的时候，这份花名册可以手把手地提醒我们安全的做法。我们希望你经常回来翻阅这些章节。

Ruby 中的重构

在编写《重构》第 1 版的时候，我选择了 Java 来展示这项技术，这主要是因为当时 Java 是一门非常热门的语言。其实绝大多数重构技术并不局限于任何语言，所以很多人读过第 1 版以后将重构带到了 Java 以外的世界。

不过使用最顺手的语言来学习重构显然是非常有帮助的。现在有很多人都在学习 Ruby 语言，而重构又是 Ruby 文化中的一个核心部分，所以我们觉得有必要为 Ruby 爱好者提供一种学习重构的方法——特别是对那些没有使用大括号语言背景[⊖]的人。

Jay 和 Shane 承担了这项工作，从头到尾审阅本书的第 1 版，并把它改成 Ruby 的风格。他们仔细阅读了原文，去掉所有 Java 风格的东西，然后用 Ruby 的方式重新组织上下文。他们都是资深的 Ruby 程序员，同时又有深厚的 Java 和 C#背景，因此这项工作由他们来做真是再适合不过了。

他们还添加了一些 Ruby 特有的重构技术。虽然我们在前面说过：绝大多数情况下，重构对任何面向对象语言来说都是一样的，但是仍然会有一些特别的东西是某些语言专有的情况。

谁该阅读本书

本书的目标读者是专业程序员，也就是那些靠写软件吃饭的人。书中的例子和讨论包含了

⊖ 此处指 Java/C/C++ 系的语言。——译者注

大量需要阅读和理解的代码。

虽然重构的焦点是代码，但是它对系统的设计也会产生巨大的影响。因此资深设计师和架构师也有必要理解重构的理念并且在项目中使用它们。重构最好是由有威望的资深开发人员来进行。这样的开发人员能够更好地理解重构背后的思想，并将它们应用于具体的工作场景。

如果不想通读，这里列出了本书的大纲。

- **如果你想知道什么是重构**，请阅读第 1 章；其中的例子应该能让你了解它的流程。
- **如果你想知道为什么要进行重构**，请阅读前两章。它们会告诉你什么是重构，以及为什么应该进行重构。
- **如果你想知道应该在哪里进行重构**，请阅读第 3 章。它会告诉你哪些是提示你需要进行重构的标志。
- **如果你想要实际进行重构**，请完整地阅读前 4 章。然后浏览花名册，只要了解它的大概内容即可，你不用理解所有的细节。当你真的需要进行重构的时候，再去仔细阅读相关的内容也不迟。花名册是一份参考目录，所以用不着一口气读完。

我们在编写本书时假设你从来没有遇到过重构，也没有读过本书的第 1 版，因此你可以把本书当做是对这个主题的完整介绍。你可以凭自己的语言喜好，选择本书或是原书第 1 版。

我有第 1 版——还需要再买这本吗

大概用不着。如果你对第 1 版很熟悉，那么本书不会给你提供多少新内容。虽然在使用这些重构技术的时候需要转换到 Ruby 语言上来，但要是你和我们一样，这应该不算什么太困难的事情。

但是基于以下两点原因，我们觉得第 1 版的拥有者可以考虑买下这本 Ruby 版。第一个原因是你不太熟悉 Java 语言，而又觉得这种陌生让你在阅读第 1 版的时候觉得很吃力。如果这样，我们希望这本 Ruby 版会稍微简单一点。其次就是如果你正在领导一个 Ruby 团队，而团队里有人不适应第 1 版里的 Java，在这种情况下，这本专为 Ruby 编写的书可能更能帮助你们理解重构。

站在前人的肩膀上

有时候别人会称我（Martin）为“重构之父”。这时我总是会婉拒这个称谓，虽然我的书确实帮助重构流行了起来，但是它绝对不是我的发明创造。我的工作主要都是构建在那些来自 Smalltalk 社区的顶尖高手之上的。

重构社区里的两位顶尖程序员是 Ward Cunningham 和 Kent Beck。他们很早就将重构视为开发过程中的一个核心组成部分，并将它们运用到自己的工作中。特别是和 Kent 的合作真正让我领略到了重构的重要性，这也直接激发了我编写本书的灵感。

Ralph Johnson 在伊利诺伊大学香槟分校领导的一支团队以其对对象技术做出的一系列实际贡献而久负盛名。Ralph 很早就是重构的拥护者，他的几个学生也一直致力于研究这个课题。

Bill Opdyke 在他的博士论文里第一次详细论述了重构技术。John Brant 和 Don Roberts 则开发出了全世界第一款自动化重构工具：Smalltalk 重构浏览器。

自从我的书出版以来，很多人发展出了各种重构技术。工具领域更是出现了爆炸性的成长。任何专业的 IDE 现在都得有一个“重构”菜单才像样，而很多人现在也已经把重构当成是他们开发工具里必不可少的一个部分了。这里我们要着重说明的是，虽然工具会让重构变得更简单，但是就算没有工具你也一样可以有效地进行重构。

Ruby 版制作花絮

人们总是想知道一本书是怎么制作出来的，特别是多人合著的时候。

Martin 在 1997 年初就开始了《重构》第 1 版的编写工作。他记录下编程时进行重构的情况，这样一来这些笔记就可以提醒他如何高效地进行某种重构（这也是本书的主干）。这本书自 1999 年出版以来，销售成绩一直都很稳定——每年都能卖出一万五千多本。

2006 年的时候 Jay 找到了 Martin，说是想写一本 Ruby 版。Jay 找了不少人来帮忙，其中 Shane 的贡献最大，并很快就成为了主要作者之一。Martin 的写作重心当时在别的项目上，因此他在这一版里没有太多地参与，不过我们还是把他的名字留在了封面上，毕竟第一稿是由他提供的，其中很多内容都在这一版里保留了下来。

作者简介

Jay Fields 是 DRW Trading 的一名软件程序员，也是一位研讨会的常客。Jay 对发现和成熟化创新式解决方案总是抱有激情。Jay 的个人网站是 www. jayfields. com。

Shane Harvie 在美国、印度和澳大利亚等地的敏捷环境里开发了很多软件。他现在位于芝加哥的 DRW Trading 公司工作，博客网址是 www. shaneharvie. com。

Martin Fowler 是 ThoughtWorks 的首席科学家，他是一位作家、演说家和咨询师，在软件开发活动中经常发表演讲。他关注的是企业软件设计——诸如什么能产生良好的设计，而良好的设计又需要哪些实践等。他是面向对象技术、重构、模式、敏捷方法论、领域建模、统一建模语言（UML）和极限编程的先行者。过去十年来他一直在为 ThoughtWorks 工作，这是一家非常出色的系统开发和咨询公司。

致　谢

《重构》永远都是我（Jay）最喜欢的一本书。它让我大开眼界，由此我才成为一名更好的程序员。虽然我非常喜欢第1版，但是我还是希望能为Ruby读者降低一些门槛。因此在2006年下半年的时候我萌发了要写一本Ruby版的念头。于是我打电话给Martin想听听他的意见，令我惊喜的是Martin非常支持这个想法。遗憾的是，这个项目由于种种原因一拖就是好几个月。有时候朋友甚至会说："我们干嘛不这个周末就坐下来开始写呢?"

到2007年1月，我们终于开始着手进行这个项目了。尽管大多数内容都可以直接拿来重复利用，但是这本书还是耗费了我们几个人相当大的精力才完成。它耗费的时间远超过一个周末，如果没有来自大家的无私贡献是绝对不可能完成的。

Ali Aghareza编写了好几个小节以及绝大部分的图片。

John Hume和Stephen Chu共同编写了几个小节。

虽然这本书的内容和第1版非常相似，但是由于得到了以下专家的审阅和建议，它的质量已不可同日而语，他们是：Brian Guthrie、Chris Stevenson、Clinton Begin、Dave Astels、Dave Hoover、George Malamidis、Justin Gehtland、Ola Bini、Ricky Lui、Ryan Plate和Steve Deobald。我肯定还漏掉了一些名字，在此表示歉意并致以感谢。

Stuart Halloway也参与了本书的审阅并建议我们加入新的内容。我相信正是他的推动才让这本书变得更有价值。

——Jay和Shane

在此我要感谢Jay和Shane为这本Ruby版所做的工作。在所有印有我名字的书中，这本书绝对是最轻松的，我要做的只是放手让他们去干就行了——要是写书都像这么简单就好了！

——Martin

Martin Fowler
Melrose，Massachusetts
fowler@acm.org
http://www.martinfowler.com
http://www.refactoring.com

Jay Fields
New York，New York
jay@jayfields.com
http://www.jayfields.com

Shane Harvie
Melbourne，Australia
shane@shaneharvie.com
http://www.shaneharvie.com

目　录