<<重构(Ruby版)>>

图书基本信息

书名：<<重构(Ruby版)>>

13位ISBN编号：9787111300786

10位ISBN编号：7111300785

出版时间：2010

出版时间：机械工业出版社

作者：Jay Fields,Shane Harvie,Martin Fowler

页数：294

译者：徐旭铭

<<重构(Ruby版)>>

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。

<<重构(Ruby版)>>

内容概要

本书是一本专门为职业Ruby程序员编写的重构指导。
它的目标是向你展示如何以一种既受到严格控制又高效的方式进行重构。
你将学习到不在代码中引入bug并能按部就班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信号；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。
本书是对Martin Fowler的经典权威著作《重构》的重大更新版，并以Ruby为例彻底重写——并非只是把代码从Java版中搬过来而已。
书中给出了一份详细的重构花名册，包含超过70条经过锤炼的Ruby重构技术，每一条都有详细指引、使用细节和范例代码。
其中很多重构技术都用到了Ruby专有的强大特性，你可以从华章网站下载到全部代码。
作者借用Fowler最初的想法，展示了如何以一种受控、高效以及迭代的方式进行重构，帮助你有条不紊地改善代码的质量而不会引入新的bug。
不论是编写还是维护Ruby代码，本书都将是不可或缺的重要参考。
本书内容　　理解重构的核心原则，以及进行重构的原因　　发现Ruby代码中的“坏味道”　　逐步将糟糕的设计转变成设计精良的代码　　构建测试以便保证重构正确进行　　理解重构中的难点以及克服的办法　　编写正确包装代码的方法　　在对象之间移动特性，将其放在最适合的地方　　将数据组织成更容易处理的形式　　简化条件表达式，更高效地利用多态　　创建更易于理解和使用的接口　　进行大型重构，这将可能影响整个软件系统数月乃至数年　　成功重构Ruby on Rails代码

作者简介

Jay Fields 是DRW Trading公司的一名软件程序员，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、印度和澳大利亚等国的敏捷公司中从事软件开发工作。
他现在位于芝加哥的DRW Trading

<<重构(Ruby版)>>

书籍目录

<<重构(Ruby版)>>

章节摘录

第1章重构初体验　　还记得在编写《重构》第1版的时候我曾经决定必须这样写这个开头。
传统意义上的技术书籍通常都会先进行一段基本介绍，例如历史和基本原理等。
可当有人在会议上这么干的时候，我总是会昏昏欲睡。
我会一直心不在焉、似听非听地看着演讲人，直到他给出一个具体的实例为止。
这时我才能回过神来，搞明白他究竟在说什么。
原理这种东西很容易说得太宽泛，让人无法理解怎么才能将它运用于实际。
有时候举个例子就能把话讲明白。
因此，在这本书中我决定先给出一个重构的实例。
几位审阅者觉得这么做颇不寻常，甚至可以说相当大胆。
但是我从来都没有后悔过。
在很多其他场合谈论重构的时候我也都会用到这个例子，我发觉用举例来开头其实是很不错的主意。
虽然例子中有些细节相当具体，但是你可以通过具体的实例来展现很多普遍的问题。
毫无例外，这本Ruby版也将会以一个例子来开头。
在这里我用的是和Java版一样的那个例子，虽然Jay已经把它变成Ruby版了。
多年来我一直在谈论这个例子，在这个过程中我又有了一些新的领悟，因此我在重新组织这段内容的时候也相应地做了很多修改。
就算你对这本书已经很熟悉了，我们也希望你能从中读到一点新的东西。
如果这是你第一次阅读本书，那么可以从头读起。
不过所有的介绍性例子都要面对一个很大的问题。
要是一上来就选择很大的程序进行讲解，光是描述它和解释重构的进行过程就已经很复杂了，无论是谁都会看晕。
（我不是没有试过，只是稍微复杂一点的例子就能轻易超过100页的篇幅）。
可要是选择的程序简单到足以理解的程度，重构看起来就会没什么价值了。
就像任何想要描述现实世界中有价值的技术的人一样，我发现自己进退维谷了。
老实说，我下面要给出的这样的小程序是不值得去重构的，但是如果我即将展示的代码是一个更大的系统的一部分，那么重构很快就会变得重要起来。
所以请你在看这个例子的时候，把它想象成属于一个比它大得多的系统。
1.1 起点　　这个示例程序非常简单。
它负责计算并打印出音像店里顾客的消费单据。
程序被告知客户租借了哪些影片，租期多长。
然后它会根据影片的租期计算出金额，并标识出影片的类型。
影片一共有三种类型：普通片、儿童片和新片。
除了计算金额以外，单据上还会计算出常客的租。

<<重构(Ruby版)>>

版权说明