Broadview
www.broadview.com.cn
PEARSON

Jolt大奖精选丛书

持续集成

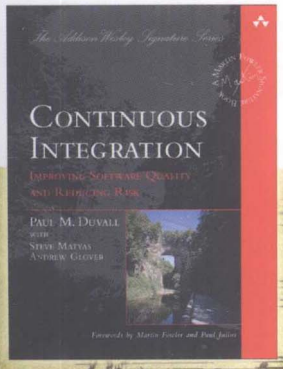
软件质量改进和风险降低之道

Continuous Integration

Improving Software Quality and Reducing Risk

[美] Paul M. Duvall
Steve Matyas
Andrew Glover 著

王海鹏 译



電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
http://www.phei.com.cn

Jolt 大奖精选

持续集成

软件质量改进和风险降低之道

Continuous Integration

Improving Software Quality and Reducing Risk

Paul M. Duvall
[美] Steve Matyas 著
Andrew Glover
王海鹏 译

电子工业出版社
Publishing House of Electronics Industry
北京 BEIJING

内 容 简 介

Jolt 大奖素有“软件业之奥斯卡”的美称，本丛书精选自 Jolt 历届获奖图书，以植根于开发实践中的独到工程思想与杰出方法论为主要甄选方向。本书全面深入地讨论持续集成的各个方面，介绍了一种增加项目可见性、降低项目失败风险的有效实践。此外，还介绍了测试驱动、代码审查、数据库集成、信息反馈等实践和工具。全书列举了持续集成系统的优缺点，以及如何使用持续集成系统、什么时候使用等，可操作性极强。

本书荣获 2008 年 Jolt 世界图书大奖，适合软件开发人员及团队阅读，还可作为软件工程方面的教材。

版权贸易合同登记号：图字：01-2011-6134

图书在版编目（CIP）数据

持续集成：软件质量改进和风险降低之道 /（美）杜瓦尔（Duvall, P.M.），（美）迈耶斯（Matyas, S.），（美）格洛弗（Glover,A.）著；王海鹏译. —北京：电子工业出版社，2012.6
（Jolt 大奖精选丛书）
书名原文：Continuous Integration: Improving Software Quality and Reducing Risk
ISBN 978-7-121-14869-9

Ⅰ. ①持… Ⅱ. ①杜… ②迈… ③格… ④王… Ⅲ.①软件质量－质量管理 Ⅳ. ①TP311.5

中国版本图书馆 CIP 数据核字（2011）第 214922 号

策划编辑：张春雨 符隆美
责任编辑：董 英
印 刷：北京东光印刷厂
装 订：三河市皇庄路通装订厂
出版发行：电子工业出版社
北京市海淀区万寿路 173 信箱 邮编 100036
开 本：787×980 1/16 印张：17 字数：435 千字
印 次：2012 年 6 月第 1 次印刷
印 数：4000 册 定价：59.00 元

凡所购买电子工业出版社图书有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系，联系及邮购电话：（010）88254888。

质量投诉请发邮件至 zlts@phei.com.cn，盗版侵权举报请发邮件至 dbqq@phei.com.cn。

服务热线：（010）88258888。

经久不息的回荡

今时的读书人，不复有无书可读之苦，却时有品种繁多而无从择优之惑，甚而专业度颇高的技术书领域，亦日趋遭逢乱花迷眼的境地。此时，若得觅权威书评，抑或有公信力的排行榜，可按图索骥，大大增加选中好书的命中率。然而，如此良助，不可多得，纵观中外也唯见一枝独秀——素有“软件业奥斯卡”之美誉的 Jolt 奖！

震撼世界者为谁

在计算设备已经成为企业生产和日常生活之必备工具的今天，专业和大众用户对于软件的功能、性能和用户体验的要求都在不断提高。在这样的背景下，如何能够发挥出软件开发的最高效率和最大效能，已经是摆在每一个从业者面前的重大课题，而这也正是 Jolt 大奖横空出世的初衷及坚持数年的宗旨。

Jolt 大奖历时 20 余年，在图书及软件业知名度极高，广受推崇。奖如其名，为引领计算机科学与工程发展主流，Jolt 坚持将每年的奖项只颁给那些给整个 IT 业界带来震撼结果的图书、工具、产品及理念等，因一流的眼光及超高的专业度而得以闻名遐迩，声名远播。

除图书外，Jolt 针对软件产品设有诸多奖项分类，如配置管理、协作工具、数据库引擎/数据库工具、设计工具/建模、开发环境、企业工具、库/框架、移动开发工具等。但图书历来是 Jolt 大奖中最受瞩目且传播最广的一个奖项分支。Jolt 曾设有通用类图书、技术类图书等分类，每个分类又设有“卓越奖”（Jolt Award，一般为一个）和“生产力奖”（Productivity Award，一般为 2 或 3 个）。

获奖技术图书一经公布，即打上经典烙印，可谓一举“震撼全世界”（赞助商Jolt可乐的广告词）。

作为计算机技术图书的厚爱者，我们总在追问——是谁在震撼世界，是谁在照亮明天？Jolt大奖恰似摆在眼前的橱窗，让我们可以近距离观看潮流在舞蹈，倾听震撼在轰鸣！

朝花夕拾为哪般

Jolt像是一年一度的承诺，在茫茫书海中为我们淘砺出一批批经得起岁月冲刷的杰作，头顶桂冠的佳作也因而得以一批批引进中国，为国人开阔了眼界，滋补了技术养分。然而，或因技术差距造就的生不逢时、水土不服，或因翻译、制作的不如人意，抑或是疏于宣传等诸多原因，这些经典著作在国内出版后，尽管不乏如获至宝的拥趸，却仍不为诸多人所知，从而与大量本应从中获益的读者擦肩而过。既然这生生错失的遗憾本不该发生，则更不应延续。为此，我们邀国外出版同行、国内技术专家一道，踏上朝花夕拾之路，竭力为广大读者筛选出历久弥新、震撼依旧的Jolt图书精品。

Jolt获奖图书皆由业界专家一致评出，并得到软件从业人员的高度认可，虽然这些书今天读来，不再能看到上世纪史诗时代那般日新月异的理论突破，以及依赖于高深繁复的科学研究所取得的系统化成果，更多是在日复一日的开发实践中总结和提炼出来的工程思想和方法论。重新选材之所以有所弃取，从Jolt多年来的评奖规律中可窥端倪——

一万小时真理见

凡是在工程思想领域取得革命性、颠覆性突破的图书，就被归于“震撼”获奖分类。比如，从基于过程的程序设计模型过渡到面向对象的全新模型，就是软件开发思想上的一次带来巨大震撼的革命；再比如，打破传统的瀑布模型而转向持续集成的软件交付模型，这也是一场业界的重大思想转变。像这样的重大思想突破，可以说是数年甚至数十年一遇的，而荣获Jolt大奖的图书中更为常见的，则是基于最佳实践的“生产效率”获奖者。获得此类殊荣的图书，都是作者们从平凡的、重复的，甚至用一般人的眼光看来不怎么起眼的日常开

发实践中，以独具的慧眼、过人的耐心和大胆的创新，闯开一条不平常道路的心血与经验总结。

这些图书所涉及的主题，都是普通的软件开发人员每天要面对的工作——代码阅读、撰写测试用例、修复软件问题……但就是这样貌似平淡无奇的工作，是否能每一天、每一个项目都做好，着实拉开了软件开发人员素质的差距，也决定了软件企业开发出来的产品和服务的质量。我们中国有一句古话，叫做熟能生巧；某位著名企业家也说过一句家喻户晓的名言："把简单的事千百万次地做好，就是不简单的。"这些朴素而实际的真理，同样也是本套丛书最能彰显的所谓程序员精神。它建立在脚踏实地的实践基础之上，也充满了对于自由和创新的向往。

名作可堪比名曲

就不因岁月流逝而褪色来说，与这些 Jolt 名作相媲美者，只有那些百年响彻、震撼古今的经典名曲。希望本丛书带给大家的每部著作，也如百听不厌的乐曲，掩卷良久方余音绕梁，真知存心。仔细想来，软件开发与古典音乐岂非有异曲同工之妙？既是人类心智索问精确科学的探究，亦是寻觅美学享受的追求。工程是艺术的根基，而艺术是工程的极致。衷心地希望各位读者能够认真阅读本丛书的本本珍品，并切实地用于自己的日常工作中，在充分享受大师魅力的同时，为中国的软件事业谱写更多、更震撼的乐章。

电子工业出版社博文视点

二〇一二年春

译者序

软件项目开发有两大难题：一是确定软件的需求，即确定目标；二是确定目前离目标还有多远，即确定剩余的工作量。第二个问题就是项目缺少可见性的问题，这对“人月神话”做出了“巨大贡献”。当一个项目经理或一名开发者说已经完成了80%的任务，您必须保持审慎的乐观。因为剩下的20%可能还需要80%的时间，甚至永远也不能完成。您可能迟迟不能拿到可以部署的软件，对此所有的人都无能为力，只能表示深深的遗憾。这确实让专业软件开发者的声誉蒙羞。但是对于大型软件开发这样的复杂工作，我们的经验确实显得有些不够。

早在自顶向下的结构化设计时代，Harlan Mills就指出，“自顶向下”的含义就在于项目消除了集成困难。如何将诸多知识工作者的工作有效地组织起来，形成一个概念完整的产品，这是越来越多的人们所面对的挑战。软件项目，特别是大型软件项目的高失败率，引起了很多智者的思考。

Grady Booch说，成功的项目有两个特点：一是具有良好的架构愿景，二是采用迭代增量式的开发过程。越来越多的人认识到，项目的早期就应该验证架构的可行性，得到系统的“可行走的骨架”。持续集成是成功项目不可或缺的实践。

成功的项目，还离不开信任：客户与开发者相互信任，管理层和开发者相互信任，开发者之间相互信任。里根说：“信任，但要检查。”我们也可以反过来说，因为可以随时检查，所以我们信任。因此，采用持续集成策略之后，项目中的信任大为增强。

项目在经过较短的启动阶段之后，就一直可以提交能工作的软件。开发者可以更早听到用户的反馈意见。客户可以根据业务环境和约束条件灵活地决定下一步的开发重点。系统的架构和功能随着业务的发展而演进。

持续集成也模糊了开发和运营之间的界限。根据具体项目的要求，可以缩

短交付间隔，实现持续交付。这对于许多业务来说，是极有价值的。

《持续集成：软件质量改进和风险降低之道》这本书向我们介绍了一种增加项目可见性、降低项目失败风险的有效实践。许多软件开发的资深人士认定，这种方法非常不错。我们不必把宝全部押在最后那一次“大爆炸”式的集成上，而是采用“早集成、常集成”的策略。这样做可以减少缺陷引入和缺陷发现之间的时间，提高开发效率，降低风险。您对项目完成百分比的报告将有更大的信心，而且任何时候，您都可以得到一个可以部署的软件。虽然功能可能还没有全部实现，但它是可用的！

这本书向我们揭示了这样一个道理：如果一件事很难，而您又必须做，不妨经常去做，每次做一点点。其实这也是古老的“分而治之”思想的一种应用。正所谓“滴水穿石，跬步千里”。

敏捷软件开发的许多实践都是互相关联的。持续集成在与其他实践结合时，才能将它的效用发挥到极致。这本书除了介绍持续集成的基本原则和工具之外，也介绍了测试驱动、代码审查、数据库集成、信息反馈等实践和工具。人（思想）、过程和自动化工具的完美结合，将形成一个和谐的开发生态环境。如果您一直在追求效率更高的软件项目管理方法，我相信这本书一定能给您带来一些启发和灵感。

一本好书使您改变。它将改变您的思想，您看待问题的角度和方式，最终，它将改变您的行为。然而，所有具有重要意义的改变都不会在一夜之间发生。改变随时都在发生，但按照您的意志去领导变革却很难。如果您相信这种变革必须发生，不妨朝着这个方向去努力，经常改变，每次改变一点点。

软件业中没有银弹，不可能有某种东西在短时间内让您的开发效率提高 10 倍。但是我们也很容易发现不同人和不同团队之间的开发效率相差巨大，不止 10 倍。那些软件高手和明星团队就像职业围棋选手，他们高得惊人的效率是多年用心改进实践的结果。

参加本书翻译工作的人员除封面署名的外还有：王海燕、李国安、周建鸣、

范俊、张海洲、谢伟奇、林冀、钱立强、甘莉萍。在这本书的翻译过程中，我学到了很多，因此郑重地向大家推荐它。如果这本书对于您改进软件开发实践有所帮助，我将十分高兴。

王海鹏

辛卯年夏末于上海

Martin Fowler[1]序

在软件行业发展的初期，软件项目中最棘手、最紧张的时刻就是集成。单独能工作的一些模块被组装在一起，系统整体却常常失败，而且很难找到失败的原因。但在最近几年里，集成基本上已不再是项目中的痛苦之源，而是变成了“小事一桩”。

这种转变的关键在于更为频繁地进行集成。人们曾经认为日构建（daily build）是一个较难达到的目标。但是今天我接触到的项目每天都集成许多次。很奇怪，如果您遇到很痛苦的事情，更频繁地去做这件事似乎是比较好的建议。

关于持续集成，一件有趣的事情就是人们常常会对它产生的影响感到吃惊。我们经常发现人们认为它的好处不大，但它却给项目带来了完全不同的感觉。项目的可见性变得好了很多，因为问题能够更快地检测出来。引入缺陷和发现缺陷之间的时间间隔变短，就更容易发现缺陷，您可以很容易地看见改变了什么，以方便找到问题的根源。当它与良好的测试程序配合时，可以大大减少缺陷的数量。结果是，开发者在调试上花的时间减少了，在增加功能上花的时间更多了，他们相信自己是在一个坚实的基础上开发软件。

当然，光说您应该更频繁地集成是不够的。在这个简单的词语后面有一些原则和实践，正是这些原则和实践使得持续集成变成现实。您可以找到一些建议，它们散布在一些书籍中和 Internet 上（我很自豪，我也在这方面提供过一些内容），但是您必须亲自花力气去寻找。

所以我很高兴看到 Paul 把这些信息收集起来，成为一本完整的书。对于希望执行这些最佳实践的人来说，这是一本参考手册。和许多简单的实践一样，细节之中包含着许多令人烦恼的东西。这些年来，我们已经对这些细节有了许多了解，并学会了如何进行处理。这本书汇集了这些经验，为持续集成奠定了坚实的基础，就像持续集成为软件开发奠定了坚实的基础一样。

---

[1] Martin Fowler 是 ThoughtWorks 公司的首席科学家，在面向对象分析、UML 模式、软件开发方法学等方面都是世界顶级专家，他是著名畅销书《分析模式》、《UML 精粹》和《重构》的作者。

Paul Julius序

我一直希望有人能够抽出时间来写这本书，早写比晚写好。私下里说，我总希望这个人就是我。但是我很高兴 Paul、Steve 和 Andy 最后能够一起完成这本完整的、经过深思熟虑的专著。

我一直投入在持续集成之中，做那些似乎永远也做不完的事情。2001 年 3 月，我和朋友一起创建了开放源代码项目 CruiseControl，并成为项目的管理者。我的日常工作是在 ThoughtWorks 提供咨询，利用持续集成（Continuous Integration，CI）的原则和工具帮助客户设计、构建和部署测试解决方案。

在 CruiseControl 的邮件列表中，2003 年活动开始多了起来。我有机会读到数千个不同的 CI 故事。软件开发者们遇到的问题各不相同，非常复杂。开发者们完成所有这些工作的理由对我来说越来越清楚了。CI 的好处，如快速反馈、快速部署和可重复的自动化测试，要远大于实现 CI 的麻烦。但是，在创建这类环境时却很容易忽视这一点。当我们第一次发布 CruiseControl 时，我从来没想到过人们会通过一些有趣的方式，利用 CI 来改进他们的软件开发过程。

2000 年，我在一个大型的 J2EE 应用程序开发项目中工作，这个项目用到了 J2EE 规范中提供的所有功能。这个应用程序本身就已够让人吃惊，更不必说它的构建了。我所谓的构建，是指编译、测试、打包并执行功能测试。当时 Ant 还处在初期，还没有成为 Java 应用程序构建工具的事实标准。我们使用了一套组合的 shell 脚本来编译所有的东西并执行单元测试，使用另一套 shell 脚本将所有的东西变成可部署的包。最后，我们通过一些手工的步骤来部署 JAR 包并执行功能测试。不用说，这个过程变得费时费力，而且经常容易出错。

从那时起我就希望创建一个可重复的构建过程，只要按“一个按钮”就可以（当时 Martin Fowler 的热门话题之一）。Ant 解决了跨平台的构建脚本的问题。我还想要另外一些东西，能够处理那些烦琐的步骤：部署、功能测试及报告结果。那时，我研究了已有的解决方案，但没有找到我想要的。在那个项目中，我从未找到我所希望的东西。那个应用程序成功地完成了开发并投入使用，但我知道事情还可以做得更好。

在那个项目和下一个项目之间的时间里，我找到了答案。Martin Fowler和 Matt Foemmel 刚刚发布了他们关于 CI 的第一篇文章。很幸运，我遇到了另一些 ThoughtWorks 的同事，他们正致力于把 Fowler/Foemmel 系统变成可复用的解决方案。我很兴奋，太兴奋了！我知道这就是在上个项目中一直萦绕在我心中的问题的答案。在几周之后，我们已经准备好了所有的东西，并在几个已有的项目中开始使用。我甚至拜访了一个愿意进行 Beta 测试的地方，在一个真正的目标企业中安装了 CruiseControl 的前身。不久之后，我们开放了源代码。对我来说，再也不会回头了。

作为 ThoughtWorks 的一名顾问，我遇到了一些极为复杂的企业级部署结构。我们的客户根据业界的宣传资料承诺的优势，经常希望得到快速的修复。就像所有的技术一样，关于它可以怎样轻易地改变您的企业，实际上存在着一些误导。如果说我从多年的顾问工作中学到了什么，那就是没有什么事情像看起来那么简单。

我想告诉客户如何实际地应用 CI 的原则。我想强调从开发的“韵律”转变到真正享受到那些好处的重要性。如果开发者每个月只签入（check in）一次，不关注自动化的测试，或者并不急于修复失败的构建，那么要完全享受 CI 的好处就很成问题了。

这是否意味着 IT 经理们应该先完成向这些实践的转变，再来碰 CI 呢？不是的。实际上，应用 CI 的实践可以是促成这些改变的最快推进器。我发现，安装像 CruiseControl 这样的 CI 工具会使软件团队变得积极主动，而不是消极被动。这些改变不会在一夜中发生，您必须有正确的预期——包括那些相关的 IT 经理们。通过对底层原理的深入理解，即使是最复杂的环境也可以变得易于理解，易于测试，而且易于很快地投入生产使用。

本书的作者们为您整理好了比赛场地。我发现这本书既全面又深入。这本书深入地讨论了 CI 最重要的方面，它将帮助读者做出理智的决定。书中的各种主题介绍了今天在 CI 领域中运用的各种方法，帮助读者衡量需要进行的折中。最后，我很高兴看到 CI 社区中有这么多的工作被规范化，成为下一步创新的基础。由于这一点，我极力推荐这本书。它是一个重要的资源，利用这些 CI 魔法，可以使得企业级应用程序的复杂配置变得有意思。

前言

在我刚刚参加工作的时候，我看到杂志上有一张整页的广告，展示了键盘上的一个键，类似 Enter 键，上面标着“Integrate（集成）”（参见图 1）。键下面的文字是“假如一切如此容易。”我已记不清楚这个广告是谁为了什么而做的，但它打动了我的心。在软件开发方面，我曾想，这当然永远不会实现，因为在我们的项目里，我们会花几天的时间在“集成地狱”中挣扎，在接近项目里程碑的时候尝试将大量软件组件拼凑起来。但是我喜欢这个想法，所以我剪下了这张广告，把它贴在我的墙上。对我来说，它代表了我成为一名高效率的软件开发者的主要目标之一：将重复的、容易出错的过程自动化。而且，它包含我的梦想，即将软件集成变成项目中的“小事一桩”（Martin Fowler 曾这样说）——只是自然发生的事情。持续集成（CI）可以帮助您将项目中的集成变成小事一桩。

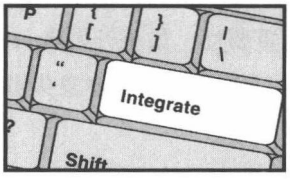


图 1 集成

这本书讲什么

请考虑软件项目中的一些典型的开发过程：编译代码、通过数据库定义数据并操作数据、进行测试、复查代码，最后部署软件。另外，团队肯定需要就软件的状态进行沟通。请想象一下，如果您可以按一个键就完成这些过程。

这本书向您展示了如何创建一个虚拟的集成按钮，将许多软件开过过程都自动化。而且，我们介绍了如何持续地按下这个按钮，从而减少创建可部署的应用程序时的风险，如较晚才发现缺陷、低品质的代码等。在创建 CI 系统时，许多过程都被自动化，在每次修改开发的软件时，都执行这些过程。

什么是持续集成

集成软件的过程不是新问题。在一个人开发的项目中，依赖外部系统又比较少的话，软件集成不会成为太大的问题，但是随着项目复杂度的增加（即使只增加一个人），就会对集成和确保软件组件能够一起工作提出更多的要求——要早集成，常集成。等到项目快结束时才来集成会导致各种各样的软件品质问题，解决这些问题代价很大，常常会导致项目延期。CI 以较小增量的方式迅速地解决这些风险。

在 Martin Fowler 的热门文章“Continuous Integration”[1]（持续集成）中，他将 CI 描述为：

……一种软件开发实践，即团队的成员经常集成他们的工作，通常每个成员每天至少集成一次——这导致每天发生多次集成。每次集成都通过自动化的构建（包括测试）来验证，从而尽快地检测出集成错误。许多团队发现，这个过程会大大减少集成问题，让团队能够更快地开发出一致的软件。

根据我的经验，这意味着：

- 所有开发者都先在他们自己的工作站上执行私有构建[2]，然后再将他们的代码提交到版本控制库中，从而确保他们的变更不会导致集成构建失败。
- 开发者每天至少向版本控制库提交一次代码。
- 集成构建每天在一台独立的计算机上进行多次。
- 每次构建都必须 100%通过测试。
- 生成可以进行功能测试的产品（如 WAR、配件、可执行程序等）。
- 修复失败的构建是优先级最高的事情。
- 某些开发者复查构建生成的报告，如编码标准报告和依赖分析报告，寻找可以改进的地方。

---

[1] 参见 www.martinfowler.com/articles/continuousIntegration.html。

[2] 私有（系统）构建和集成构建模式在 Stephen P. Berczuk 和 Brad Appleton 写的 *Software Configuration Management Patterns* 一书中有介绍。

本书讨论了 CI 中自动化的方面，因为您会从自动化重复的、容易出错的过程中得到许多好处。但是，正如 Fowler 所指出的，CI 就是经常集成工作的过程，这不一定需要自动化的过程。我们相信，既然已经有许多工具让 CI 成为自动化的过程，那么使用 CI 服务器来自动化 CI 实践就是一种有效的方式。不管怎样，也许手工集成的方式（利用自动化的构建）很适合您的团队。

快速反馈

持续集成增加了您获得反馈信息的机会。这样，您每天都能多次了解项目的状态。CI 可以用来减少引入缺陷和修复缺陷之间的时间间隔，从而改进软件的总体品质。

开发团队不应该相信因为 CI 系统自动化了，就可以避免集成问题。如果团队只使用自动化的工具来编译源代码，就更是如此。有些人把编译称为“构建”，实际上不是的（参见第 1 章）。有效的 CI 实践包含的东西比工具多得多。它包括本书中介绍的实践，如经常向版本控制库提交代码，立即修复失败的构建，以及使用独立的集成构建计算机等。

CI 的实践支持快速反馈。如果应用了有效的 CI 实践，您可以每天多次了解到正在开发的软件的健康状况。而且，CI 与重构、测试驱动开发等实践配合得挺好，因为这些实践的中心思想都是进行小的变更。从本质上来说，CI 提供了一张安全网，确保变更能够与软件的其他部分一起工作。从更高的层面上讲，CI 增加了团队整体的信心，减少了项目所需的人工，因为它通常是一个无人值守的过程，在软件发生变更时执行。

关于“持续”的注释

我们在本书中使用“持续”这个术语，但是这种用法从技术上来说是不对的。“持续”意味着某事一旦启动就不会停止。这意味着集成过程一直在执行，但是即使对于最密集的 CI 环境来说，也不是这样的。所以，我们在这本书中讲述的更像是“经常集成”。

谁应该读这本书

在我的经验中，把软件开发当成工作的人和把它当成职业的人之间有着显著的差别。本书是为那些把软件开发当成职业，并发现自己在做一些重复的过程的人（或者我们将帮助您意识到您正频繁地这样做）。我们描述 CI 的实践和好处，让您知道如何应用这些实践，这样您就可以将时间和专业知识用在更重要、更有挑战性的问题上。

这本书介绍了与 CI 相关的主要话题，包括如何利用持续反馈、测试、部署、审查和数据库集成来实现 CI。不论您在软件开发中承担哪种角色，您都可以将 CI 纳入到自己的软件开发过程之中。如果您是一位软件开发专家，希望变得更有效率（在您拥有的时间里做更多的事情或得到更可靠的结果），您会从本书中学到很多东西。

开发者

如果您已注意到自己宁愿为用户开发软件而不是与软件集成问题搏斗，那么这本书将帮助您消除原来的大部分“痛苦”。这本书不会要求您花更多的时间来集成，它是讲如何让大部分的软件集成工作变成“小事一桩”，让您能够关注最喜欢做的事情：开发软件。这本书中的许多实践和例子向您展示了如何实现一个有效的 CI 系统。

构建/配置/发布管理

如果您的工作是让能工作的软件发布出去，您会发现这本书特别有意思，因为我们在书中展示，通过在每次对版本控制库进行变更时执行一些过程，您可以生成一致的、能工作的软件。可能许多人在管理构建的同时还承担项目中的其他角色，如开发。CI 将替您完成一些“思考”，不必等到开发生命周期的末尾，它每天都能多次生成可测试的软件。

测试者

CI 为软件开发提供了快速的反馈信息，消除了过去即使进行了“修复”

之后，缺陷还会再次出现的痛苦。在实现了 CI 的项目中，测试者通常会提高满意度，并提高对他们的角色的兴趣，因为送来测试的软件更为频繁，每次要测试的范围也较小。在开发生命周期中使用 CI 系统之后，您的测试是一直进行的，而不是像通常那样有时忙得要死，有时又闲着没事。以前测试者要么工作到很晚，要么又没有东西可测试。

经理

对于团队一致地、重复地交付工作软件的能力，如果您希望有更大的信心，那么这本书将给您带来巨大的冲击。您可以更有效地管理时间、费用和品质，因为您决策的基础是能工作的软件、真实的反馈信息和测量指标数据，而不是项目进度计划上的任务项。

本书的组织结构

本书分为两个部分。第一部分介绍了 CI，从头开始讨论了 CI 的概念和实践。第一部分针对的是那些不熟悉 CI 的核心实践的读者。但是，如果没有第二部分，这些 CI 的实践是不完整的。第二部分自然地将核心概念扩展为 CI 系统执行的有效过程，包括测试、审查、部署和反馈等。

第一部分　CI 的背景知识：原则与实践

第 1 章，启程，让您通过一些例子了解如何利用 CI 服务器来持续构建软件。

第 2 章，引入持续集成，让您熟悉常见的实践方法，知道如何开始 CI。

第 3 章，利用 CI 减少风险，通过场景式的例子说明 CI 可以缓解的主要风险。

第 4 章，针对每次变更构建软件，探讨了利用自动化的构建，在每次变更时集成软件。