TURING 图灵程序设计丛书

MANNING

Test Driven Practical TDD and Acceptance TDD for Java Developers

测试驱动开发的艺术

[芬] Lasse Koskela 著
李贝 译

人民邮电出版社
POSTS & TELECOM PRESS

TURING 图灵程序设计丛书

Test Driven Practical TDD and Acceptance TDD for Java Developers

测试驱动开发的艺术

[芬] Lasse Koskela 著
李贝 译

人民邮电出版社
北京

图书在版编目（CIP）数据

测试驱动开发的艺术 /（芬）科斯科拉（Koskela, L.）著；李贝译. -- 北京：人民邮电出版社，2010.11
（图灵程序设计丛书）
书名原文：Test Driven : Practical TDD and Acceptance TDD for Java Developers
ISBN 978-7-115-23836-8

Ⅰ. ①测… Ⅱ. ①科… ②李… Ⅲ. ①软件开发－英文 Ⅳ. ①TP311.52

中国版本图书馆CIP数据核字(2010)第189051号

内 容 提 要

本书介绍了一种更快更好的软件开发方法——测试驱动开发。全书共分三部分：第一部分讲述了TDD和ATDD的相关知识、基本概念、方法，为测试驱动开发打下基础；第二部分将测试驱动开发用于具体的实践，重点讲解了TDD的各种技术；第三部分着重介绍了验收测试驱动开发，包括Fit框架、实现验收测试的方法等，最后讲解了引入TDD的各种技巧。

本书浓缩了作者多年的开发经验，适合各类Java开发人员学习参考。

图灵程序设计丛书

测试驱动开发的艺术

◆ 著 [芬] Lasse Koskela
译 李 贝
责任编辑 朱 巍
执行编辑 李胜华

◆ 人民邮电出版社出版发行 北京市崇文区夕照寺街14号
邮编 100061 电子函件 315@ptpress.com.cn
网址 http://www.ptpress.com.cn
中国铁道出版社印刷厂印刷

◆ 开本：800×1000 1/16
印张：21.75
字数：512千字 2010年11月第1版
印数：1－3 000册 2010年11月北京第1次印刷
著作权合同登记号 图字：01-2009-5730号
ISBN 978-7-115-23836-8

定价：59.00元

读者服务热线：(010)51095186 印装质量热线：(010)67129223
反盗版热线：(010)67171154

版 权 声 明

前　言

七年前，正值全球 IT 产业繁荣时期，大大小小的软件公司都发了疯似地想赶上下一波 IPO，招聘市场火爆异常。我也在此时投身到繁荣的新媒体产业，开始了我的编程生涯。从此我没日没夜地鼓捣各种代码段，配置服务器，往生产系统里上传 PHP 脚本，似乎一切尽在掌握。

一个九月的雨夜，又是加班到很晚，突然间我的心脏似乎停止了跳动：哎呀！我刚才做了什么？我是否删掉了生产数据库里的所有数据？好像是的！看来我只有卷铺盖走人了。我怎么才能把数据找回来呢？刚才还以为只是测试用的数据库呢！这种悲剧怎么能发生在我头上？然而，确实发生了。

第二天我没有被炒鱿鱼，主要原因是，看来客户对我删掉的数据并不太在意。而且，看来别的人也都干过类似的蠢事——他们安慰我说：大家都可能犯错。我得到一个教训，那个该死的夜晚也标志着我开始追求一种负责任的、可靠的软件开发态度。

几年以后，我换了家国际性咨询公司工作，为其他大公司开发应用和后台系统。在短短几年的职业生涯里我学到了不少东西，这得归功于我以前趴在电脑前熬夜的努力，而新工作无疑是我磨炼实战技艺的好机会。我又一次认为我已经对于软件开发行当熟门熟路了。可是我又错了，显然我比自己想象的要知道得少。我几乎每天都能学到重要的新知识。

我最重大的发现改变了我对软件开发的认识，极限编程（XP）给了我全新的视角，让我知道什么才是正确的软件开发方法。在我看来，XP 把我过去行之有效的披荆斩棘式的编程方式与一种系统的、训练有素的工作方法结合在一起。XP 项目除了能让开发团队更接近客户之外，最打动我的就是测试驱动开发（TDD）了。我以前认为编程和单元测试是两个分离的活动，现在“编码之前先写测试”这样一个简单的理念完全颠覆了我的旧思想。

TDD 绝非闲庭信步那样轻松。我时刻提醒自己要先写测试，一开始能做到，可是只过了半个小时，我就忘了遵守，还没有测试就在修改代码。随着时光流逝，我越来越能够坚持测试先行的编程方法，甚至一整天都不会落入往日的陋习中。接着我会被一段代码难住，凭我的能力我无法征服它。再往后，我能理解应该怎么做，但我的手法还不够用。再后来，我不知道如何能四两拨千斤地巧妙解题，却又往往不愿意愚公移山般地用笨办法尝试。年复一年，我学会了越来越多的技巧，掌握了越来越多的工具，终于获得了现在的功力。

我写此书的目的是让诸君不必像我以前那样笨拙地克服种种困难，你们有此书在手可以轻松地前行。对我而言，学会了测试先行，深刻地影响了我工作的方法和对编程的认识，正如敏捷方法改变了我对软件开发的认识。我希望你们也能学会测试先行。

关 于 本 书

为了更好更快地开发软件，软件开发人员经过实践和思考，提出了测试驱动开发的方法。本书作者作为软件开发人员，希望降低其他人学习 TDD 的难度。鉴于刚接触 TDD 的开发人员最头疼的是一些技术性问题，本书主要采用了“手把手”的教学方式。不仅会通过动手做例子来解释 TDD，而且还将用几章的篇幅专门讲解如何为通常认为难于测试的技术来编写单元测试。只有动手实践才能收到最佳的学习效果，而本书可以在你的探索之旅中充当一部导航仪。

本书读者对象

本书面向各层次的 Java 程序员，特别是希望提高效率和编码质量的人。测试驱动开发会为你提供一个框架，让你能增量地构建软件，借以释放你的潜能。不管你是在开发导弹控制系统，还是在创造下一个 YouTube，采用 TDD 都不会让你失望。

本书的第二类读者不一定对 TDD 感兴趣，但作为 Java 程序员，他们想在测试方面寻求一些帮助。测试驱动开发主要是一种设计和开发技术，但编写单元测试却是 TDD 中的一项基本工作。本书同样能为这一类读者提供纯粹编写测试的帮助——我们介绍了为很多（所谓的）难于测试的技术，如访问数据的代码、并行程序，以及用户界面代码编写测试的方法。

不管你是想做好工作，还是想自己提升，本书都能助你一臂之力。

路线图

本书涵盖了很多背景和基础知识。全书分为三大部分，关注不同的主题。

第一部分从最基础的内容讲起，讨论测试驱动开发和验收测试驱动开发。

第 1 章先提出问题（即要面对的挑战），然后解释了 TDD 和 ATDD 给出的有效解决方案，包括测试先行编程、渐进设计、测试自动化和果断重构。

第 2 章开始动手实践，为深入理解 TDD 展示了一个例子：以测试驱动方法编写的模板引擎。这一章还将讨论如何管理测试和选择下一个测试。

第 3 章以第 2 章为起点，为模板引擎引入了一些设计变更，从一个 Spike 开始，然后再以可控、严格的方式来修改模板引擎。

第 4 章回头再讲述策略，从选择测试到通过测试。这一章会讨论一些基本的测试概念，如

夹具、测试替身，还会讨论基于状态和基于交互的测试有什么区别。在说明了如何创建可测试设计的几条准则之后，这一章还介绍了几个重要的测试模式以及如何以测试先行的方式测试遗留代码。

第二部分同样侧重于动手实践，逐一讲解了如何对那些以往认为不好测试的技术应用 TDD。学习完第二部分，你就明白过去的认识有多么可笑。

第 5 章先从 Web 开发讲起。学习如何对使用 Servlet 和 Spring Controller 生成的请求/响应式的 Web 层进行测试驱动开发，以及测试驱动由 JSP 和 Apache Velocity 模板构建的表现层。这一章最后还介绍了借助基于组件的框架对 Web 应用程序进行测试驱动的开发。

第 6 章探讨如何测试驱动 Web 组件背后的数据访问层。将分别介绍基于原始的 JDBC、Spring Framework 的 Jdbc Template API 和事实上的 ORM 工具 Hibernate 来测试驱动数据访问对象。另外，还将讨论在单元测试中处理数据库问题的方法和集成测试。最后，展示了一些有关文件系统的技巧。

第 7 章探索了新的未知领域：不确定行为。在简单介绍了与时间相关的几种选择后，首先转向多线程，探讨了几个能够测试的方面，还讨论了线程安全、阻塞操作、启动和停止线程及异步执行。这一章最后讲解 Java 5 中新增的 java、util、concurrent 中的同步对象。

第 8 章讨论界面，即 Java Swing 应用程序的界面。首先，也是从测试驱动的 UI 代码该测试哪些方面入手。然后，介绍三个有用的设计模式和两个开源工作（Jemmy 和 Abbot），它们可以简化对 Swing 组件进行单元测试的工作。这一章（也是第二部分）最后提供了一个例子——测试驱动定制 Swing 组件的界面和行为。

第三部分调整步调，从讨论具体的测试驱动对象和类，上升到基于验收测试驱动的测试先行的方式，构建完整的系统。

第 9 章讨论用于需求管理的用户故事和验收测试的基本概念，之后，解释 ATDD 的过程和对团队的需求，以及基于 ATDD 开发软件的好处。这一章最后讨论验收测试时应说明系统的哪些方面和一些工具。

第 10 章介绍流行的验收测试工具 Fit。首先讨论如何使用 Fit 与用户协作，即先以表列格式列出验收测试，再将其转换成 Fit 格式。然后讨论将表列测试与系统关联起来的技术，涉及三个 Fit 内置的标准夹具和 FitLibrary 提供的几个实用工具。最后，讨论如何在命令行中和以 Ant 构建方式来运行 Fit 测试。

第 11 章探讨如何实现与工具无关的验收测试。在介绍了几种将测试与系统连接起来的方法后，讨论了这些技术的优缺点。最后还与读者分享了提高验收测试效率和应对复杂性的技巧。

第 12 章讨论了如何成功地推行 TDD。开头探讨了实现长久改变的先决条件，事关我们自己和同事。接着关注反对意见：怎样识别和处理反对意见。最后给出了很多确保推行成功的建议。

考虑到单元测试在测试驱动开发中的重要性，本书附录还为读者准备了相关的三个工具的简明教程。附录 A 和附录 B 分别展示了 JUnit 单元测试框架 4.3 和 3.8 版本的语法。附录 C 也差不多，它展示了 EasyMock（用于生成测试替身的动态模拟对象框架）的语法。

能在自己喜爱的 IDE 中编写测试驱动代码的确很棒，但我们还希望把测试的部分也融入自动构建的过程中。这正是添加附录 D 的原因。附录 D 提供了 Java 开发人员使用的标准构建工具——Apdche Ant 中运行单元测试的简明教程。

代码约定

本书代码主要有 Java 代码、标记代码和输出结果。对于大段代码，我们将配以标题。而小段代码则直接与正文混排在一起。无论是在哪里，代码都以等宽字体印刷，以区别于正文。在第二部分，经常会引用代码清单中的标识符和元素，这些也都使用等宽字体。另外，特别长的代码清单中还会有一些编号的注解，以便于正文中引用说明。

下载代码

可以从 Manning 公司网站下载本书全部示例代码，地址为：http://www.manning.com/koskela。其中包括本书中出现和省略的代码。

下载的文件中包含 Maven 2 POM 文件及使用 Maven（http://maven.apache.org）编译和运行示例的说明。请注意，下载的文件中不包含依赖文件。在第一次运行 Maven 时，它会在能上网的情况下帮你下载必需的依赖。此后，即使断开网络连接也可以在本地运行示例了。

我们强烈建议读者安装一个 IDE。本书中的示例代码是 Eclipse 项目的格式，因此需下载并安装 Eclipse（http://www.eclipse.org）的最新或最高版本。当然，IntelliJ IDEA（http://www.jetbrains.com/idea）和 NetBeans（http://www.netbeans.org）也不错，只不过你得自己配置一下项目。

在线资料

第 12 章本来还有一个主题，讨论测试驱动企业级 JavaBean，但未包含在本书中。我们把这个有 40 多页篇幅的内容放在了网上，其中对开发人员给出了相关的详细建议。

网上这一章的内容涵盖用来封装业务逻辑的会话 bean、面向持久性的实体 bean、异步消息驱动 bean 和 Timer API。

虽然我们致力于介绍最新的 EJB 3.0 规范，但我们给出的建议实际上同时适用于 3.0 及 2.*x* AIP。这样做的原因是很多遗留系统仍在使用 EJB 2.*x* 规范。

可以在如下地址下载这一章：http://www.manning.com/kosketa。

深入学习

通过本书可以掌握与测试驱动开发有关的很多知识，但这些恐怕还不能满足你的需求。好在，Manning 提供了网上论坛，让读者可以与其出版图书的作者在线沟通。当然也包括本书作者在内。访问 http://www.manning-sandbox.com/forum.jspa?forumID=306，即可与本书作者 Lasse 交流。

关于测试驱动开发这种技术或方法，很难用一本书涵盖其方方面面。因为 TDD 还在发展，

而且针对不同领域编写测试往往有很大差别。书中总会有一些应该收录但没有收录的内容。为此，读者还应该知道如何寻求帮助。我们推荐专门讨论 TDD 及相关问题的 Yahoo!论坛 testdriven-development。如果有着急的问题不知问谁才好，就在这个邮件列表中提问吧！

如果 Yahoo！论坛还不能满足你的需求，你还可以订阅 http://www.testdriven.com，这是 TDD 的门户网站。这个网站会推荐一些相关的文章、博客和工具。当然，行业内有关敏捷方法的一些会议也经常会以 TDD 为主题，遇到这样的活动，不妨参加一下。

我们约好了，在 TDD 社区再见！

作者在线

买了这本书之后，读者就可访问 Manning 公司的论坛，发表对本书的评论，提出技术问题并获得作者和其他用户的帮助。要访问和订阅论坛，请访问 http://www.manning.com/koskela。相应页面会给出注册之后如何使用论坛，如何寻求帮助，以及论坛版块划分的信息。

Manning 提供这个论坛的初衷就是让分散的读者能有机会聚到一起，与作者进行互动交流。但这并不表示作者有义务在论坛中发言，作者对相应图书论坛的贡献都是自愿的（而且没有物质回报）。为此，我们建议读者在论坛中多提一些有挑战性的问题，免得他们失去动力。

在拿到本书时，上面提到的作者在线论坛就已经开通并可以访问了。

关于封面插图

本书封面上的插图名为 Franc Comtois，他是一位法国东北部小镇勃艮第省 Free 郡的居民。勃艮第以前一直是一个独立的地区，直到 17 世纪才并入法国版图。该地区有自己独特的传统和语言，这种名为“弗朗什-孔泰”的语言如今还在使用。

本书的插图取自一本 1796 年出版的法国游记——*Encyclopedie des Voyages*，作者为 J.G.St.Saveur。在当时，旅游业才刚刚盛行，像这本书这样的旅游向导非常流行，无论是观光客还是足不出户的“神游旅行者”，这本书都能为他们了解法国以及世界各地人们的生活提供帮助。

Encyclopedie des Voyages 一书中的插图生动展示了 200 年以前世界各地的城镇和地区之间文化的差异。相隔仅几十公里远的人们的服饰风格也不一样，通过服饰风格就能辨认出对方来自哪个地区。有了这本书，我们就可以体会到那个时代的人们之间的巨大文化差异，体会到那么多富有浓郁地方特色的服饰文化。如今，这一切都逐渐消失了。现在已经很难通过服饰来区分不同地区的居民了。也许我们用文化多样性换来了更加多样化的个人生活，也是更快节奏的科技生活。

当所有的计算机书籍都变得千篇一律的时候，Manning 出版社则通过体现两个世纪以前丰富多彩的各地生活的图片作为封面插图，以这样的方式赞美计算机行业中的独特性和主动性。

目　录

第一部分　TDD 入门

Part 1 第一部分

TDD 入门

这一部分是TDD（Test-Driven Development）入门，引领读者体验测试驱动的艺术。第1章来认识TDD和它的延伸概念——验收测试驱动开发（Acceptance TDD，ATDD），我们从最基本的知识入手，对这两种技术先有一个大概的了解。第2章进入实际动手环节，将通过修改和运行实际代码深入体验测试先行的好处。第3章将更进一步，向读者展示大规模重构的例子，借以看看设计将会有多么显著的变化。

几年来，通过向一批批的程序员讲授TDD，我体会到实践是最好的老师。在学习了第2章和第3章并亲手实现功能完善的模板引擎之后，就可以接触一些重量级的技术了。第4章介绍了运用TDD思想的一些技巧和方法，包括如何选择下一个测试并使其通过。此外，必要时还将讨论相关的设计原则和测试工具。

本部分内容

第1章 综 述

我能忍受暴力，但施暴的理由却让我无法容忍。

——奥斯卡·王尔德

测试驱动开发（Test-Driven Development，TDD[①]），用一句话讲，就是“写代码只为修复失败了的测试”。我们先写一个测试，然后再写代码让测试通过。当我们在当前结构中找出最佳设计时，由于有足够的测试做保障，我们可以放心地改动现有设计而不必担心破坏已完成的功能。使用这种开发方法，我们可以让设计更加优良，能编写出可测试的代码，同时还能避免在不切实际的假设基础上过度地设计系统。要得到这些好处，只需不断添加可执行测试，一步步地驱动设计，从而最终实现整个系统。

这本书就是讲如何实行这每个小步骤的。在后面的章节中，我们将会学到TDD的本质以及玄妙之处，学到如何利用TDD开发普通Java应用程序及企业级Java应用程序，以及如何用ATDD（Acceptance TDD，验收测试驱动开发，测试驱动开发核心理念的一个扩展）来驱动整个开发过程。在用测试驱动开发实现某个具体功能之前，我们将会首先编写功能测试或验收测试，从系统功能角度驱动开发过程。

TDD其实并不是一个新概念。很久以前，不少开发人员就认为编写测试不仅仅是为了验证系统正确性了。至今你还可以从他们那里听到这个故事：从那时候起，我们写代码之前都会先写测试……。而现在，这种开发方法有了自己的名字——TDD。本书的大部分内容都是关于“如何测试驱动”以及“测试驱动什么”的。这些知识可以被用到各种日常软件开发任务中。

不过，相对于主流开发方法，TDD仍然很新。就像今天的日常用品曾经都是十足的奢侈品一样，新的开发或设计方法，通常都是只有资深的开发人员才能拥有的高级货，而在很多年以后，等这些先驱证明了新方法确实有效，这种方法才会被广泛接受，成为开发必备技能。

我相信TDD会逐渐地变成主流开发方法，而且我认为，TDD已经开始变为主流开发方法了。希望本书能够推动这个过程。

---

① 缩写TDD有时也代表测试驱动设计（Test-Driven Design）。有人把TDD称为用测试先行编程（Test-First Programming），只不过叫法有区别而已。

在本书的开始部分，我们会先讨论现有的开发方法在开发软件过程中遇到的挑战。一旦对现存问题有了基本共识，我们将着手讨论如何用TDD或ATDD解决这些问题，同时也会学着使用一些支持此开发方法的工具。

1.1 挑战：用正确的方法解决正确的问题

开发软件的目的是为协助组织的经营运作。作为一个专业的软件开发人员，我们的主要任务是向客户交付一个能够真正帮助他们提高工作效率并减少运作成本的系统。

回顾我多年的专业软件开发经验，参阅几十年来软件开发的文献记载，再看看整个世界范围内的技术人员对于开发方法的争论，我们不难看出很多开发组织本应向他们的客户交付更好的软件。换言之，我们现在交付的软件可不怎么好用。就算这些软件能够正常的工作，它们也没有真正解决客户的问题。从本质上讲，我们写出的代码并没有满足客户的真正需求。

接下来，我们来看看为什么“糟糕的代码质量”和“不能满足客户日益变化的需求”会妨碍我们给客户交付足够好用的系统。

1.1.1 糟糕的代码质量

虽然软件开发行业已经发展了很多年，但是开发出软件的质量依旧存在问题。近些年市场越来越注重软件的及时上市，对软件产品的需求量越来越大，同时很多的新技术也不断涌现。在这种情形下，软件行业将不可避免地继续面临质量问题。

这些质量问题主要表现在两个方面：高缺陷率和低可维护性。

1. 在缺陷的泥潭中挣扎

缺陷会带来很多额外的开销，因为它会导致软件不稳定，行为不可预测，或者完全不能使用。缺陷减少了软件本身的价值，有时甚至使软件造成的破坏远大于创造出的价值。

测试可以解决这些问题——我们仔细地检查软件是否像期望的那样工作，同时也试着通过某种方式检测其是否稳定。测试在软件开发中的重要性是毋庸置疑的，但是传统的测试方法只会在整个软件开发完毕并且代码“冻结”后才进行，而且会耗费很长时间。这种测试方法有很大的改进空间。例如，在测试阶段修复一个缺陷的成本，通常是在编码阶段就修复这个缺陷成本的两倍或者更多。有缺陷的软件是不能交付的。我们在寻找并修复缺陷上用的时间越多，开销越大，我们开发软件的能力也越低。

软件缺陷通常是由低质量的代码引起的，要维护这些代码简直就是噩梦，而想对其进行进一步开发也会举步维艰，代价高昂。

2. 维护困难，开发缓慢

好的代码有很多优点：整体设计优秀，清楚地划分出了各部分之间的功能和责任，并且没有重复。而糟糕的代码可不是这样，成天和这种代码打交道简直就是噩梦。这些代码很难读懂，修改也很困难，因此工作效率会很低。改这里，还会莫名其妙地破坏掉那里的功能。重复的代码使修复缺陷也变得不容易，改完一处，还要找出其他所有重复的代码，挨个修正同样的缺陷才算完。

糟糕的代码带来的问题还远不止这些。

“我不想碰它，那任务永远不可能完成，如果做了，天知道会破坏掉其他什么功能。”这是软件业需要解决的一个很现实的问题。想修改或者添加新功能时，应该是基于现有系统继续开发，而不是重写，这就是可维护性的重要之处。有良好的可维护性，我们才能及时应对迅速变化的业务需求。如果代码的维护性差，那么我们行动和响应的速度会变慢，按时交付的压力会变大，这会促使开发人员写出质量更差的代码。如果想持续地交付软件，我们必须打破这种恶性循环。

这些问题已经够糟糕了，但是还有其他问题：我们交付的软件通常都没有完全满足客户的需求。

1.1.2　不能满足客户需求

没人愿意花钱当冤大头，但软件行业的客户就经常被逼着这么做。在软件开发前期，客户和开发人员互相交换规格文档，然后开发工作就此开始。十二个月以后客户拿到系统，才发现这并不是自己当初想要的。更不用说在当前激烈的商战下，客户的业务需求和十二个月以前常常大相径庭。

由于交付的系统经常不能真正满足客户需求，软件开发行业一直在尝试用新的方法开发软件。我们试着在各种规格文档上多花些功夫，但效果却适得其反。我们试着延长交付系统的时间，却导致更多的人参与进开发过程。生产软件是为了支持人工作，但更多的人却为了生产软件而工作。另外，在开发初期确定过多的细节会导致文档不可靠。由于细节上的假设环环相扣，规格文件中的错误能轻易拖垮整个项目。

软件行业的业绩不佳，但也不必因此沮丧，因为这些问题有办法解决。敏捷软件开发[①]——包括Extreme Programming（极限编程）和Scrum等方法——就是解决这些问题的良药。本书余下的内容会详细介绍敏捷开发方法的核心实践——测试驱动。

1.2　解决方案：测试驱动

我们遇到的问题可以分为两个方面：代码质量不高和不能满足客户需求。解决方法也存在两个方面：学会正确地构建系统，以及学会构建正确的系统。本书所介绍的测试驱动方法能很相似地应用在这两方面，差别只在于如何利用测试来构建出维护性高并能真正满足客户需求的软件。

在细节层面上，我们用TDD方法以测试驱动的方式编写代码。在较高的，即软件的特性和功能层面上，我们使用类似的ATDD方法以测试驱动的方式构建系统。图1-1从提高内部质量和外部质量角度上描述了这种方法组合。

从图1-1中可以看出，在这两个不同的层面上结合使用测试驱动，能保证软件的内部质量，同时能保证可见的外部质量。在下面的小节里，我们将会讨论TDD和ATDD是如何带来这些改进的。在深入技术细节之前，先看看测试驱动如何帮我们应付软件开发中的挑战吧。

---

① Craig Larman所著的*Agile & Iterative Development: A Manager's Guide*（Addison-Wesley, 2003）一书很好地介绍了敏捷开发方法。