Broadview
WWW.BROADVIEW.COM.CN

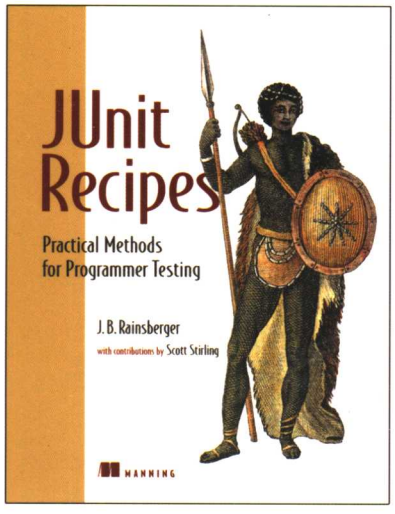
Java 人的工具箱系列

MANNING

JUnit Recipes 中文版

——程序员实用测试技巧

Practical Methods for Programmer Testing



[加] J.B.Rainsberger
Scott Stirling 著

陈浩 王耀伟 李笑 译

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

JUnit Recipes 中文版

—— 程序员实用测试技巧

JUnit Recipes Practical Methods for Programmer Testing

[加] J. B. Rainsberger 著
Scott Stirling

陈 浩
王耀伟 译
李 笑

電子工業出版社
Publishing House of Electronics Industry
北京 • BEIJING

内 容 简 介

本书主要介绍了在 Java 开发中使用 JUnit 进行单元测试的各种方法、原则、技巧与实践。本书出自开发一线专家之手，本着实用的原则，涵盖各类 Java 开发中应用 JUnit 的实用技巧，内容丰富、全面深入。无论是需要应用 JUnit 进行单元测试的一线 Java 开发人员，还是 JUnit 入门、进阶者，对他们来说，本书都是一本不可多得的实用指南。

本书介绍了大量的 JUnit 实用测试技巧，从如何命名测试类到测试复杂的 J2EE 应用程序（包括 servlets、 JSP、 EJB 和 JMS 组件等）。它告诉您如何在不同情况下优化自己的代码。每个测试技巧都依照固定的格式进行介绍：首先提出问题及其背景知识，然后探讨具体的解决方案。因此，本书的技巧被编写成了一百多个相互独立的短文，每个问题都针对一个特定的 JUnit 使用问题，方便您在遇到具体的问题时，随时查阅。

版权贸易合同登记号：图字：01-2004-4695

图书在版编目（CIP）数据

JUnit Recipes 中文版：程序员实用测试技巧 /（加）雷恩斯伯格（Rainsberger,J.B.），（加）斯特灵（Stirling,S.）著；陈浩，王耀伟，李笑译. —北京：电子工业出版社，2006.9
书名原文：JUnit Recipes：Practical Methods for Programmer Testing
ISBN 7-121-03099-3

Ⅰ.J... Ⅱ.①雷...②斯...③陈...④王...⑤李... Ⅲ. JAVA 语言－程序设计 Ⅳ.TP312

中国版本图书馆 CIP 数据核字（2006）第 097934 号

责任编辑：周 筠
印 刷：北京东光印刷厂
装 订：三河市金马印装有限公司
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：787×1092 1/16 印张：40.5 字数：880 千字
印 次：2006 年 9 月第 1 次印刷
定 价：69.00 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，联系电话：（010）68279077；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。

译者序

随着极限编程（Extreme Programming，XP）和测试驱动开发（Test-Driven Development，TDD)盛行，如何有效地组织自动化测试已经成为软件工程中的一个重要课题。自动化测试的书籍已经不少（比如《JUNIT IN ACTION 中文版》，《单元测试之道 Java 版——使用 JUnit 》等），但截至目前，市场上多数的 JUnit 书籍都侧重于介绍 JUnit 的功能和讲授 JUnit 的使用方法，很少深入地探讨如何养成良好的 JUnit 编写习惯。

JUnit 是一个优秀的 Java 代码测试框架，但有的时候，有些代码却很难找到有效的测试方案。这时候，这本书可以帮助你。本书的作者根据自己丰富的 JUnit 使用经验，为您提供了大量优化测试方案的准则，比如如何组织测试数据，如何测试昂贵的外部资源（如数据库等）。通过对这本书的学习，可以帮助您养成良好的 JUnit 代码编写习惯。养成良好的编写习惯是非常重要的，它不仅可以帮助你写出更好的代码，还可以使你的工作更高效，有助于提高客户的满意度。

这本书介绍了大量的 JUnit 实用测试技巧，从如何命名测试类到测试复杂的 J2EE 应用（包括 servlets、JSP、EJB 和 JMS 组件等），它告诉您如何在不同情况下优化自己的代码。每个测试技巧都依照固定的格式进行介绍：首先提出问题及其背景知识，然后探讨具体的解决方案。因此，本书的技巧实际上被编写成了一百多个相互独立的单元，每个单元都面向一个特定的 JUnit 使用问题，您可以在遇到具体问题的时候，随时查阅它。

本书的原名为 *JUnit Recipes*，而 Recipes 的原意就是菜谱的意思。这本书就像是一部 JUnit 的烹饪大全，它描述了每个菜肴的属性（问题所归属的领域），组成部分（由背景部分介绍，这部分还为您介绍了在解决方案中用到的各种工具），还有具体的烹饪方法（即诀窍部分，这部分介绍了如何使用背景部分介绍的工具解决实际问题）。同时，作者还详细讨论了所选择方案的优缺点，与类似方案所做的横向比较，有助您加深对所推荐方案的理解。

在布局上，书中的技巧也由浅入深，首先介绍解决基本测试问题的通用方法，然后在这些方法的基础上，逐渐为您引入高级的测试技巧。本书还对测试新项目和遗留项目分别进行了讨论。对于打算在新项目上使用 JUnit 的程序员，本书介绍了如何设计易于测试的代码，以避免产生许多遗留代码的问题。对于要在已有项目上使用 JUnit 的程序员，本书

专门为那些最难测试的遗留组件提供了有效的测试方案，比如 EJB 和 JDBC 等。

最后，我要感谢在翻译这本书时给予我帮助的王愚、刘宇辉以及 Sun 中国研究院的同事们，他们在翻译过程中为我提供了大量的帮助。我还要感谢博文视点的工作人员，是他们的积极努力才保证了这本书的出版。另外，由于书中的翻译难免有不当之处，希望广大读者提供宝贵的意见，给予批评指正。

王耀伟

2006 年 7 月于星城国际大厦

前言

Bones: 我不知道这种配方是否曾被用作镇定剂。

Kirk: 要多久才能准备好？

Bones: 马上。

Kirk: 很好。这药需要多长时间才能起效？

Bones: 三到四秒钟。

Kirk: 你怎么测试它的？

Bones: 还没有测试过。

Spock: 这不必要，船长。

Bones: 这很简单，不会出错的。

Kirk: 现在，任何事情都会出错。我要你马上测试它！

Scotty: 有志愿者吗？

Bones: 会有的。

Scotty: 那么我来做志愿者。（他费力地打开一瓶威士忌。）这是为了镇痛。

Spock: 但这是无痛的。

Scotty: （假笑）好吧，你很快会警告我的，Mr. Spock，说吧。（Scotty 用力地吸入含安定药的烟，但没有任何效果。）

Kirk: 这药不起效。

Spock: 确实是。很奇怪。

Kirk: 这是我们最后的机会了。

Spock: 船长，您似乎不明白。它没有起效，但它一定会起效的。不会错的，船长，它应该会起效的。

Kirk: 一个科学的论断……

Spock: 但如果镇定剂不起效，它显然是不可能起效的，那么我们必须开始对我们的思维模式做一个根本的改变。

摘自"Spectre of the Gun", *Star Trek* original series
Episode No: 056, Air Date: 10.25.1968, Stardate: 4385.3

你手上拿的这本书，是为我们这些在日常工作中使用 JUnit 的人编写的一本优秀的技巧手册。这不是一本关于 TDD 的书，也不是 JUnit 的基本教材。实际上，这本书是为那些在专业的实战环境中使用 JUnit 的人编写的一套 JUnit 技巧——既包括简单的，也包括复杂的。你是否想知道如何去测试一个小服务程序？一个 XSLT 脚本？再或者一个实体 bean？你是否关心如何去命名和组织你的测试实例类？你是否曾费劲地去测试数据库，或者去组织大量的测试数据？这本书就是关于这些问题的技巧，同时它还涉及很多其他的测试难题。这些技巧编写得很好，易于理解，并且很注重实际。每项技巧都以实例的形式给出，注明了要解决的问题，问题的环境，以及解决问题的各种技巧。

我第一次遇到 J.B.是 2003 年在新奥尔良的 *XP Agile Universe* 大会上。他是 *FitFest* 运动的热心参与者。在 *FitFest* 实验室，一有机会他就编写测试和代码。在这些会议的许多即兴讨论和交谈中，他也是一个敢于直言的积极参与者。他的专业知识和技能给我留下了深刻的印象，因此我决定研究一下他其他的著作。这没有让我失望，而让我明白了 J.B.确实精通于他的业务。或者，像我的一个亲密伙伴说的那样：“J.B.确实知道很多技巧”。

当我第一次知道 J.B.在写这本书的时候，我对这本书充满了期待；他已经成功地实现了超越。没有任何一本书能像他的这本书一样，在一本书内收录这么多关于 JUnit 和单元测试的智慧、知识和实践建议。读这本书让我完全相信 J.B.确实了解 JUnit，以及它所有的拓展子系统，以及 JUnit 环境的方方面面。我很确信这本书将成为我书架上最容易拿到的几本书之一，这样我就可以在急需的时候迅速拜读其中的内容。

Robert C. Martin
Object Mentor 公司创始人

序言

如果你曾经见过我，不论是在网上还是当面见到，你都可能已经听过我要讲的这个故事。

我曾在多伦多的IBM实验室里参与过一个大型项目。那是2000年的中期，2000年的千年狂欢已经结束了很久，而我已经在一个预计以一个月完成的组件上花费了将近三个月的时间。源源不断的麻烦来自于我们的测试团队，而每个修正都只是在不断扩张的大麻烦上加上另一个补丁。大约就是那个时候，我读了由 Ron Jeffries, Ann Anderson 和 Chet Hendrickson 编写的 *Extreme Programming Installed* 的草稿。通过互联网的帮助，这份草稿将我带到了 www.junit.org，在这里我遇到了一个用来测试 Java 代码的神奇工具，那就是JUnit。没过几分钟我就意识到，它一定会在事业上帮助我。

之后不久，我冲进了我的经理的办公室，告诉他我不可能在产品交付之前及时为现有的代码打包并修复其他的缺陷。代码已经变得太过复杂，因此很难理解。我无法想象在修复已有缺陷的同时，将会引入多少新的缺陷。我们还是没有得到及时的答复。“让我回家”我告诉他，“我回去后会使用 JUnit 测试它。这样会成功的。”他答应了，当事情变成这个样子的时候，他还能有什么选择呢？

虽然我还不知道如何有效地使用JUnit，但在JUnit的帮助下，我只用九天的时间就重写了原来要花三个月的代码。一开始花费500个小时写出来并且无法运行的代码，在大约100个小时内被全部重写，其中包括一个含有超过125个测试的测试套件。这让我充满了成就感——我迷上了JUnit。

从那时起我就加入了JUnit社区，我不仅成为“测试驱动开发”模式的参与者，还在JUnit雅虎社区回答问题。几年之后，我将我的JUnit技巧提高到可以保证以大约25小时的时间，完成类似上述的组件。在编码中节约95%的时间是很有意义的；虽然无以证明，但我还是将其中大部分的原因归功于JUnit。

在2001年的时候我遇到了一些关于JUnit文档的抱怨。很显然，我们没有一个合适的参考指南。我决定写一本《JUnit：新手指南》在网上发布，这个指南平均每个月吸引超过 1000 个读者。那个指南是这本书的起源，虽然那个时候我还不知道这一点。这本书的不少内容都取自我对 JUnit 邮件列表上一些问题的回答。但更多的内容来自于在多

个项目中总结的、来之不易的 JUnit 使用经验；我并不是想展示一些关于使用 JUnit 的理想化观点。目前已经发布了太多的观点和演示——但这里没有。这本书介绍的是使用的 JUnit 技巧，因为这为我的项目带来了成功。因此有必要声明两件事情：我展示的大多是我自己的观点，这来自于我的经验；另外这本书不仅仅提供了解决问题的办法。这本书提供的是建议，而不是规则。

到这本书出版并到了你手上的时候，情况可能已经变化。有些技巧可能已经变得陈旧。我对此无能为力——人们每天都在发现使用 JUnit 的更好的办法。虽然其中一些技巧可能会过时，但其中的观念（技巧背后的动机）永远都不会过时。测试独立性是很重要的，小的测试更有效，将实现与接口分离会使测试变得简单，将你的代码与框架脱钩可以使测试成为可能。请全面学习这些不断重复的主题。它们是本书最有价值的部分，因为它们会在我们停止编写 Java 软件之后的很长时期内继续帮助你，只要提到的情况继续发生。如果你发现它们对你有所帮助，那么我作为作者和 JUnit 社区成员的工作就算成功了。

致谢

在 2002 年的后期，我确定了 2003 年的两个主要目标：更多地参与 XP/Agile Universe 大会，写一本书。虽然这本书六个月之后才能上架，我还是很高兴地宣告，现在我已经实现了这两个目标。一个人在没有帮助的情况下很难获得成功，因此我想借这个机会，感谢那些帮助我编写这本书的人。

首先，我想感谢 Manning 出版社的工作人员，他们在 2003 年三月份联系我，请我写一本关于我最喜欢的主题：JUnit 的书。*JUnit in Action* 的作者，Vincent Massol，非常热心地为我提出建议，而所有那些与我合作的 Manning 出版社每一个员工都非常支持我的工作。Jackie Carter 作了非常杰出的工作，他不仅为这本书做了复审和编辑，还在写作过程中给予我莫大支持。如果团队中有像 Jackie Carter 这样的伙伴，一个第一次写书的人也可以把书写得很好！我还想感谢出版商 Marjan Bace，他不仅帮助我保证了书的质量，还耐心帮助我确定书的标题。Marjan 在实现自己既定的目标时毫不松懈，而与他一起工作非常辛苦，这是在需要完成优秀、困难的工作时可以理解的辛劳。除了 Jackie 和 Marjan，我还想感谢 Susan Capparelle，Clay Andres, Lianna Wlasiuk，Leslie Haimes 和 Mary Piergies，他们为我提供了珍贵的原始资料，并帮助我校验了手稿，设计了封面，并制作了本书的最终拷贝。Alistair Cockburn 完成了一个非常成功的项目，不论他的团队是否愿意以同样的方式运作其他的项目，在本书这个案例中，他们确实做得非常成功！

整个社区的成员都为这本书作出了各种贡献——这本书还没有出版，它的雅虎社区成员已经达到了 100 人。我常常对互联网聚集网民并让他们彼此协作的能力感到惊讶。在这里无法做一个完整的列表，但我在这里谨向如下协助者表示衷心的感谢：Vladimir Bossicard，Simon，Chappell，Roger Cornejo，Ward Cunningham，Mark Eames，Sven Gorts，Paul Holser，Dave Hoover，Ramnivas Laddad，Jason Menard，Rick Mugridge，Jonathan Oddy，Kay Pentecost，Paolo Perrotta，Bret Pettichord，Ilja Preuß，Michael Rabbior，Neil Swingler，Edmund Schweppe 和 Chad Woolley。他们帮助我完成了各种实例，复查了我的手稿，对其中的技巧提出了建议，讨论了书中的一些观点，并且搜索到了相关参考书目。我还能再向他们要求什么呢？

一些协助者不属于这个社区，因此我想单独感谢他们。首先是 Scott Stirling，他参与了《使用测试数据进行测试》和《JUnit 结果》两章的写作。此外 Scott 还大量参与了这本

书内容轮廓的早期设计工作，保证了我们能在书中介绍大量的基本概念。我很希望 Scott 有更多的时间来参与！

Eric Armstrong 在修改这个手稿的早期版本时，比其他任何复审人员所做的工作都要多。如果你想写一本书，你可以首先想办法让 Eric 感兴趣，这样的结果将会比没有他好得多。当 Eric 不再有时间的时候，Robert Wenner 参与进来，并继续这项工作。没有他们的深入而详细的注解，这本书不会像现在这样完美。Eric 和 Robert 完成以后，George Latkiewicz 又对这本书做了一遍复审，他们仔细审查了书中的小毛病，修改了那些会使读者愤怒、让作者难堪的不合时宜的陈述。George 的杰出工作使我轻松了很多。

Mike Bowler 不仅回答了我关于 HtmlUnit 和 GSBase 的所有问题，还为我提供了一个非常急需的宣传媒介。他向我指出了 J2EE 测试中最常见的问题，告诉了我更重要的是哪些技巧。与 Mike 的合作一直都很愉快，我会在任何时候抓住机会与他合作。

极限编程和敏捷软件发展协会提供了这本书的编写机会。其中，他们不仅有 Kent Beck 负责 xUnit 框架，还向我发出邀请让我参加他们讨论，使我有机会向他们学习，逐渐成长为一个程序员。他们对我的悉心教导和建议，我在这里表示衷心感谢。我希望我能用这本书向他们的支持和帮助作出一些回报。

特别地，我想感谢 Bob 叔叔（或者 Robert C. Martin，如果他更喜欢这个称呼的话），感谢他答应为这本书撰写前言。我不想随便冠以“行为楷模”这样的头衔，但 Bob 确实是我的一个楷模。只是我希望能从他那里获得必要的认可，以避开他那切实而严厉的批评。像敏捷编程协会的许多其他人一样，Bob 专注于解决问题，而不是去发表批评；但如果你犯了错误，那么他会毫不客气；哪怕是他自己犯了错，他也会毫不犹豫地指出。Bob 很让人尊敬，当他讲话时，我都洗耳恭听——虽然有时让人难堪。谢谢你，Bob！

当我是一个年轻孩子的时候，我一直看不上所有的作品，直到我遇到了 Bruce Adlam 和 Caroline Schaillee 这样的老师。这本书写得好的部分，大部分是他们的功劳；有瑕疵的部分，应该由我承当所有的责任。Nick Nolfi 也应该得到尊重，他给了我一些有趣的编程问题让我解决，并培养了我编写代码的乐趣。我还应该向学校电脑实验室里那些可怜的 ICON 电脑道歉，它们不得不承受我高强度的持续使用。

我的妻子 Sarah，在我宣布放弃全职工作的相对安全保障去写这本书的时候，没有责怪我，这才使这本书的诞生成为可能。没有她的长期支持和鼓励，我肯定没有办法写完它。如果有一天她要写自己的第一本书，我一定会承担起我的责任给她支持。

最后，我要感谢我的母亲 Joan Skillen，她不仅为抚养我长大而付出很多艰辛，还为支持我追求理想而付出了巨大的牺牲。

关于此书

单元测试之外

作为测试驱动开发模式的实践者，我们都倾向于将 JUnit 当作对象测试工具。因为大多数的 JUnit 协会都与测试驱动开发协会有关系，因此这似乎很合理。但是，有可能你并不是一个测试驱动发展模式的实践者。你现在的项目也许正在使用 JUnit，但只是为了给现有的代码编写测试，或者站在比对象更高的层次上为系统编写测试。我们非常希望让你明白，JUnit 同样适用于编写其他类型的测试。

这其中包括集成测试，集成测试仍然属于程序员的测试范围。这种测试更关注多个类的协作，而不是单个对象在特定阶段的行为特征。如果要保证你的类以期望的方式“相互交谈”，这种测试很有帮助。集成测试存在一系列与对象测试不同的问题。集成测试一般比对象测试更脆弱，因为它们有更复杂的测试实体：在调用打算测试的行为之前，你首先必须保证你正确地设置了大量的对象。甚至，你有可能需要测试系统与外部资源的集成：比如数据库、基于网络的服务或者文件系统。在所有的级别（包括对象测试，集成测试和端到端测试），只要涉及到诸如此类的低速资源，或者难以获取的外部资源，我们都会对如何编写高效的测试进行详细讨论。因为这种问题主要出现在 J2EE 应用中，本书的第 2 部分“测试 J2EE”提供了大量的、编写涉及这种资源的测试的诀窍，你可以在各种情况下使用它们。

还有客户测试，它的目的是向客户或者终端用户证明：他们需要的功能在系统中是存在的。这种测试一般执行缓慢，会涉及几乎整个系统。除了需要让客户编写测试外，客户测试的最大挑战就是，编写代码的时候必须保证，系统用户接口的稍微改动，不会危害到整个系统。如何解决这种问题超出了本书的范围，但我们已经尽量提出了一些建议。

端到端测试以端到端的方式，完整地测试整个系统，因此这种测试往往是最难的。它们一般很难有效地自动化，因此许多项目都将精力放在对象测试的自动化上，而将端到端测试留作手动测试。这种情况下，JUnit 也可以帮助你，尤其是在结合了行之有效的技巧和功能丰富的插件的时候。在这种时候，我们主要讨论用户接口级别的测试。如果你在编写 web 应用程序，那么 HtmlUnit（http://htmlunit.sourceforge.net）可能是你工具箱里最重要的工具。它不仅提供了 HTTP 客户端用来模拟网页浏览器，还提供了自定义的断言层次，允

许你对应用程序返回的网页内容进行分析。我们在第 13 章“测试 J2EE 应用”中提供了使用 HtmlUnit 工具的诀窍，同时在第 3 部分“更多的测试技巧”中介绍了其他一些专门的 JUnit 测试包。

最后我们要说的是，没有任何一本书可以介绍使用 JUnit 的所有可以想到的方法。因此，除了本书提到的技巧外，你还可以使用 JUnit 做更多的事情。Kent Beck 曾经说过，JUnit 的目标是创建一个每个人都需要的框架，而不需要堆积只有部分人需要的功能。他希望我们这样看待 JUnit：“JUnit 很不错，但如果我能在这里添加少许的功能，它就会变得完美。”已经出现了大量的开源项目为 JUnit 提供自定义的拓展，而我们提供的一些技巧也许可以帮助你去实现自定义的 JUnit 项目。这种项目越多，JUnit 就会变得越热闹。

本书的结构

本书的第 1 部分介绍的技巧，是使用 JUnit 编写测试的基本技巧。如果我们作者的工作到位的话，那么你使用 JUnit 编写的每个测试，其实都可以分解为这些基本技巧的组合。第 1 章给出了 JUnit 的总体介绍，包括使用它的原因，使用的目的，如何安装和如何编写测试。同时我们还介绍了对象测试的基本概念。如何编写高效的对象测试是本书的重要主题，我们认为，这个主题主要与如何使用在第 2 章“码元测试”中介绍的技巧来编写测试有关。如果这很容易做到的话，那么本书的其余 15 章就没有存在的必要了；但实际上，除了最简单的项目，实际的问题很快就需要求助于其他部分，因此在第 1 部分的其余章节，我们提供了其他一些技巧，用来对付项目中比较复杂的测试问题。

第 3 章“组织和编译 JUnit 测试”介绍了如何组织你的的测试源代码，如何创建测试。我们不仅介绍了在文件系统中存放测试源代码的技巧，还提供了让测试类和产品类进行相互通信的技巧。另外，我们提供了一些创建测试的实例，其中既有来自 IDE 内部的例子，也有来自自动化测试创建过程的例子。

第 4 章“管理 Test Suites”给出的是将测试收集到“Test Suite”中的技巧。Test Suite 被当作一个执行的测试集合的组。典型情况下，我们的目标是永远都执行所有测试，但恰恰有时候你不能在你的项目中这样做。我们已经提供一些技巧，描述了好几个将测试收集到自定义 Suite 的技巧。

第 5 章“使用测试数据进行测试”介绍了在测试内管理数据的技巧。所有的方法都没有好坏之分，但我们倾向于将测试数据硬编码到测试中。我们倾向这种办法是因为，它支持以文档方式进行测试的概念——如果测试逻辑和测试数据相互分离，测试就更难读。另一方面，如果一个测试需要 100 份信息，那么将它们都直接硬编码到测试中也会导致测试代码阅读困难。我们需要用不同的策略来维护测试中使用的数据，我们已经在本章中将许多策略当作技巧与你分享。

第 6 章“运行 JUnit 测试”讨论了一些执行测试的策略。一般我们喜欢在所有情况下

都运行所有的测试；但如果你想只运行一个测试，或者忽略一些测试，或者甚至在各自的虚拟机中运行每个的测试，那么我们的技巧可以帮助你。

第 7 章“汇报 JUnit 结果”介绍了几个自定义 JUnit 报告测试结果的技巧。如果你需要比“370 个成功，2 个失败，1 个错误”更多的信息，那么本章的技巧可以帮助你以期望的格式，获取你需要的其他信息。本章提供了用 Ant 报告测试结果的技巧，同时深入探讨了 JUnit 框架本身。

我们用第 8 章“为 JUnit 排除疑难”总结了第 1 部分。本章的技巧介绍了在使用 JUnit 时，如何解决你可能遇到的困难。其中许多问题对第一次使用 JUnit 的人来讲都是通用问题，其中包括了配置错误及排除办法。也有些问题是在你开始用 JUnit 解决复杂系统（比如那些基于 J2EE 的系统）的问题时遇到的。

第 2 部分从第 9 章“测试与 XML”开始。在 J2EE 应用中，不使用 XML 文档，你就没有办法玩得转。因此，我们认为从 XML 测试技巧开始是符合逻辑的，而本章主要围绕 XMLUnit 和 Xpath 这两个主题展开。

下面是第 10 章“测试与 JDBC”，它介绍了测试数据库的技巧。数据库是最昂贵的外部资源，大多数 JUnit 新手在开始编写测试的时候都会遇到。本章发起了关于模拟对象的讨论，该讨论在本书的其余部分将会不断地被重新提起。本章我们探讨了如何从 JDBC 分离持久服务，然后对每个服务进行单独测试。我们还介绍了如何减少需要 JDBC 客户端的代码量，以便你在实时数据库上编写并执行少量测试。我们还提到了 DbUnit，这个工具可以在你确实要测试数据库的时候，帮助你维护测试数据。

第 11 章“测试 EJB”是目前书中最复杂的一章。EJB 的复杂性导致了对其进行有效的测试同样异常困难。再次地，我们既向可测试设计的方向进行演化，也在容器中测试遗留 EJB。我们审查了 MockEJB，它提供了对模拟对象的支持，尤其是对 EJB 的支持的一个包，但多数是它的模式环境，即一个 JNDI 路径的模拟实现。由于 JNDI 到处散布于 J2EE 应用中，模拟环境在测试与 J2EE 的集成的时候，被证明非常的方便。最后，我们在本章添加了一些 JMS 技巧，因为 JMS 的最常见用法就是用在消息驱动的 JavaBean 中。

在最后部分完成的时候，我们将注意力转向了前面的部分。第 12 章“测试 Web 组件”描述了如何测试小服务程序、JSP、快速模板和 Web 资源过滤器。像我们的 EJB 部分一样，我们讨论了从 Servlet 框架分离应用逻辑的办法，以及 ServletUnit 如何以测试遗留网站提供的虚拟容器环境。我们的技巧包括，如何使用 HtmlUnit 以分离其余应用的方式验证动态网页的内容，我们认为这是对测试行为可怜的低估。这同时也是我们倾向于使用在 JSP 上使用快速网页模板的原因：在独立模式中使用快速引擎很简单，虽然我们这样讲，但实际上，测试中并没有可用的独立 JSP 引擎。

第 13 章“测试 J2EE 应用”更多的讨论了端到端问题。我们介绍了如何使用 HtmlUnit

以端到端的方式测试 Web 应用。本章大量的技巧都主要介绍如何将看上去是端到端的问题 当作组件问题进行处理，因为这样我们就可以独立地测试它们。越是能独立地进行测试，测试就越容易，同时你也就能在编写和执行测试时获得更多的好处。

第 3 部分从 14 章开始，它简要地介绍了测试和设计模式。我们很容易在整本书到处讨论如何测试设计模式中的所有类，但因为这不是本书的主题，因此我们只选择了几个几乎在每个项目中都会遇到的重要模式实例：Singleton，Observer/Observable，工厂和模板方法。我们认为，一旦你使用过几次本书中技巧，并理解了它们的基本原则，那么你就会发现：在考虑测试一个轻量级的设计、一个适配器或者一个复合组件的时候，你基本不会再遇到困难。

我们用下面的两章来讨论两个 JUnit 的常见扩展。第 15 章介绍了关于开源项目 GSBase（http://gsbase.sourceforge.net）技巧。这个产品的一些功能可以简化 JUnit 测试的编写，另外还有一些工具可以在你的项目中直接使用。第 16 章介绍了另外一个开源项目 JUnit-addons（http://junit-addons.sourceforge.net），它不仅有测试功能，还有一个具备更开放的拓展构架的 JUnit Test Runner 替代品。这个构架使监视和分析测试的运行更加容易，这是 JUnit 的 Test Runner 本身并不直接支持的。我们强调这两个项目，不仅因为我们在自己的工作中大量地使用它们，还因为两个项目之间已经有了一些合并。合并产生的项目必定会成为 JUnit 的一个标准插件。

还有一些技巧没有放在其他章，因此将它们收录到第 17 章“拾遗”中。在这里谈到了一些测试技巧，包括基于文件的应用、Test Case 语法和测试私有方法。关于它们我们思考了很久，其实可以将这些技巧放在本书的其余部分，但有时我们没有这样做。

我们收录了前面几个技巧的完整解决方案，并将它们放在附录 A“完整方案”中。一个完整的解决方案需要几百行代码，而我们又不想影响你阅读技巧的其余部分，于是就将这些方案放在了本书的最后部分。这样你就可以阅读完整的解决方案，如果你明白了其中所有的细节，就可以在自己的项目中使用它们。

在附录 B 中，我们给出了少量的关于程序测试的文章，它们略微超出了 JUnit 的范围，但可以为本书其他部分提出的建议设置相应的环境。这个附录也可以很容易地拓展为一整本书，但我们只选择了几个最重要的主题在技巧之外进行拓展。

最后我们提供了一个参阅列表——包括一系列的书、文章和我们在工作中认为比较重要的 Web 站点。我们强烈推荐你使用它们对 Java 测试、Java 编程和其他工作进行进一步的学习。

编码约定

因为这是一本关于编程的书，我们应该花一些时间来描述我们的编码约定。下面是我

们编写的关于代码实例约定的一个简明列表。

我们一般不用缩写。这个约定仅有的几个例外是：用“e”在捕捉异常的代码段表示一个异常对象，用类名称前缀“Impl”表示某种“实现”。一般来讲，缩写只是为了简化打字员的工作，而把困难留给其他所有的人。我们决定在打字上多花一点时间，好让你更容易理解我们的代码。对我们自己来讲，这不过是做好作为作者和程序员的本职工作。

我们不指定任何特定的前缀或者后缀作为全局、类级别或者实例级别的标识符。有些项目喜欢使用下划线开头；有些喜欢使用下划线结尾；还有一些项目喜欢使用伪匈牙利命名法中用于实例范围的前缀标识符，比如用“m_”代表“member”。我们觉得这种表示法与因子分解相互抵触，因此不使用它们。我们希望能够按照的需求拆分这些代码。

我们尽可能少地使用遗弃代码。遗弃代码的目的就是避免再使用它们。因为希望这本书被很多人读，我们不想鼓励别人去使用本来希望不再被使用的方法。如果我们使用遗弃方法作为例子，那说明截至出版的时候，我们还无法找到一个可用的替代方案。我们希望这些代码能维持尽可能久。

有时候我们“按需”引入声明，也就是引入像“`import java.sql.*`”之类的声明。有人认为按需引入是有害的，应该被禁止，但我们认为优秀设计的一个特征就是仅依赖几个包，也就是说松耦合和很少的引入声明。如果你想从一个特定的包中引入很多的类——这迫使你按需引入——那就是内聚包的一个特征。

我们有时候使用“public（公有）”域。请不要晕倒。经过对 Java 程序员众所周知的糟糕代码行为的深思熟虑——包括在Java之前的一些思考——Ward Cunningham 已经审查了这个问题，并建议说，公有域不仅有充分的理由，并且还应该被优先采用。这样的话，须维护的代码就更少，可以减少一些潜在的错误。我们将公有域限定在值对象范围，其中包括用来表示我们准备在网页模板中使用的 Presentation 对象，比如 JSP 或者快速宏指令。

注意

Ward 关于公有域的说法——当我们就使用公有域的问题询问 Ward 时，他说，“我们之所以将程序中这么多东西告诉编译器，其中一个原因就是为了让编译器从我们的角度理解我们编写的代码。但我们常常不得不问：是理解帮助我们完成工作的吗？如果不是的话，那么就应该换一种做法。在过去十年，在自动测试方面我们已经学到了足够多的东西，足以用来永久修改声明带来的回报。

我们更喜欢使用和拓展已有的代码。这就是说，如果需要添加一项功能，而我们知道有人早就创建了它，那么就引入它，而不是像过去那样重造相同的轮子。在示例项目中，有大量的代码都来自 Jakarta Commons Collections 项目。当我们开始编写自己的工具时，决定使用已有的材料，并在必要的时候扩充它。我们不想打击那些屈服于“非独创”综合

症的程序员，这毛病在程序员中很流行。

在某种情况下，一行代码或者命令必须写在一行中，没有回车换行，否则它就不能正常工作。有时候这种行可能过长，无法放在一页纸的宽度之内；在这种情况下我们使用一个箭头(⇛)来表示：你必须将下面的代码写入到同一行中。

通用参考资源

我们在整本书中推荐两部作品，均出自 Martin Fowler 之手——*Refactoring: Improving the Design of Existing Code* 和 *Patterns of Enterprise Application Architecture*。绝大多数的引用都以脚注形式给出，或者在直接描述引用的代码，但因为我们如此频繁的引用这两部作品，我们使用一个速记符号。如果你看到像[Refactoring, 230]这样的引用，应该知道指的是 *Refactoring* 一书的第 230 页；如果你看到[PEAA, 412]，应该知道指的是 *Patterns* 一书的第 412 页。请查阅本书尾部的阅读列表，以获取我们推荐为补充读物的书籍和文章列表。

让示例代码活跃起来

在编写这本书的过程中——尤其是在分析示例代码的过程中，我们提取了一些打算在未来的项目中使用的小型引擎和工具。并以“Diasparsoft Toolkit”为名称发布了这个项目，你可以从 www.diasparsoftware.com/toolkit 免费在线获取。本书使用的示例代码可以从出版商的网站 www.manning.com/rainsberger 下载。

作者在线

购买《JUnit 技巧》的读者，可以自由地访问一个由 Manning 出版社维护的内部网络论坛。在那里，你可以对本书作出评论，询问技术问题，还可以从作者或者其他读者那里获取帮助。如果要访问论坛并订阅它的内容，请使用浏览器访问页面：www.manning.com/rainsberger。这个页面介绍了注册后如何访问论坛，可以获取那些服务，以及论坛的相关行为准则。

Manning 出版社对读者作出承诺提供一个交流的场所，供读者之间或者读者与作者之间进行交流。但这并不是要求作者承诺以特定的工作量来参与这个论坛，因为作者对在线论坛的付出是自愿的（也是免费的）。我们建议你向作者提出一些有趣的问题，以避免他对论坛失去兴趣！

只要本书仍在出版，作者的在线论坛，以及作者以前的谈话记录就可以随时访问。