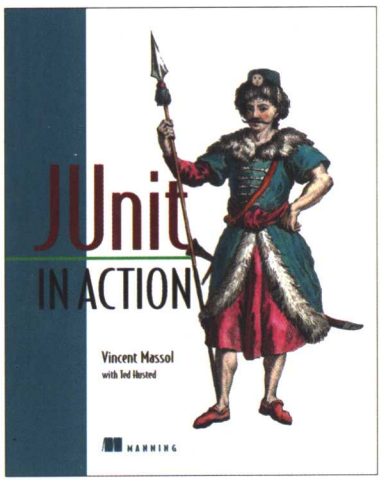
Broadview
WWW.BROADVIEW.COM.CN

Java 人的工具箱系列

JUnit 中文版

IN ACTION

[美] Vincent Massol 著
鲍志云 译



MANNING

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

Java 人的工具箱

JUnit IN ACTION 中文版

JUNIT IN ACTION

[美] Vincent Massol 著

鲍志云 译

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 软件开发中使用 JUnit 进行测试的原则、技巧与实践，深入阐述如何编写自动测试，把一段代码隔离开来测试有什么好处，如何判断何时需要进行整合测试，并对如何测试完整的 J2EE 应用进行了极具价值的讨论。本书富含开发实践当中的真实案例，以专家手笔讨论了实践中的测试技术，主要内容包括：用 mock objects 进行隔离测试；用 Cactus 进行容器内测试；用 Ant 和 Maven 进行自动构建；在 Eclipse 内进行测试；对 Java 应用程序、Filter、Servlet、EJB、JSP、数据库应用程序、Taglib 等进行单元测试。本书适合于在 Java 平台下进行各类软件开发的开发人员、测试人员、单元测试研习者以及编程爱好者阅读和学习，具有极高的参考价值。

1-93011-099-5 JUnit in Action by Vincent Massol, Ted Husted

版权贸易合同登记号　图字：01-2004-1020

图书在版编目（CIP）数据

JUnit in Action 中文版 /（美）马森（Massol,V.）著；鲍志云译. —北京：电子工业出版社，2005.1
（Java 人的工具箱）
书名原文：JUnit In Action
ISBN 7-121-00483-6

Ⅰ. J… Ⅱ. ①马… ②鲍… Ⅲ. JAVA 语言-程序设计 Ⅳ. TP312

中国版本图书馆 CIP 数据核字（2004）第 108939 号

责任编辑：周　筠　方　舟
技术协作：刘铁锋
责任校对：陈元玉
印　　刷：北京智力达印刷有限公司
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
经　　销：各地新华书店
开　　本：787×980　1/16　印张：24.25　　字数：480 千字
印　　次：2005 年 1 月第 1 次印刷
定　　价：39.00 元

凡购买电子工业出版社的图书，如有缺损问题，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。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。

译 序

本书简介

这是一本关于单元测试的书。单元测试是一种细粒度的测试，很多公司都已把单元测试作为开发流程的一部分，并明确要求开发者在提交功能模块的同时也要提交相应的测试用例，所以对大多数开发者而言，关于单元测试的知识已成了必修课。本书以最典型的单元测试框架 JUnit 为例讲述了单元测试的方法和最佳实践，简要介绍了 JUnit 的设计，并介绍了一些扩展自 JUnit 的单元测试工具。本书还介绍了很多典型的单元测试实例。这本书能帮助你用最高的效率获得在这个日益重视单元测试的世界中生存的技能。

为什么要单元测试

1．单元测试可以降低不确定性从而降低风险

给软件项目制定计划是件很难办的事。即便所有功能点都已确定，即便产品经理或者客户代表赌咒发誓说不会再有任何需求变更，开发者也依然难以确切地计算出开发所需的时间。这是因为开发者是人而不是机器，而人总会有犯错的时候。问题在于，一旦出现了 bug（这几乎是无可避免的），那么你在修复这个 bug 之前基本上无法知道这个 bug 是多久以前的哪次代码变动引入的。定位 bug 会耗去你半小时还是半个月，这取决于你的经验，取决你当时的生理和心理状态，还取决于运气。而这些都是很难量化计算的。所以，项目经理制定的日程表中无可避免地包含了大量的不确定性。

回忆一下，你的开发生涯中有多长时间是在和代码编辑器打交道，又有多长时间是在和调试器打交道呢？若你觉得看编辑器要比看调试器更顺眼一点，那么采纳单元测试不失为上策。从表面上看，为每个单元都编写测试代码增加了工作量。但是因为这些测试代码不仅为你织起了一张保护网，可以帮你迅速定位错误从而使你大幅减少对调试器的依赖，

还有助于优化设计，并且可以当文档用（都会在下文提到），所以增加的时间是物有所值的。想想吧，本来你可能要面对着调试器耗去的时间无法确定，从 20 分钟~20 小时都有可能，而且这段时间会让你极伤脑筋，调试过程可能充满了困惑与挫折，这段时间也会让项目经理极伤脑筋，因为他不能在日程表中写这个项目可能会在两个月到两年之间的哪一天交付。现在有了单元测试这种方法，可以把这段不确定的时间转化成编写并运行单元测试使之通过的相对可确定的时间（花费时间不会很多），而且编写单元测试并不是什么难事，也不会给人带来太多大悲大喜的情绪起伏（比如连续调试 10 个小时仍然没有一点点头绪，忽然在第 10 个小时零 1 分找到了 bug 所在），这非常有利于您的心脏健康☺ 所以，采纳单元测试是非常明智的。

2. 单元测试可以帮您优化设计

一般来说，写代码和测试代码的最好不要是同一个人。我记得有不少软件工程教材上都提到过这一点。但是在这个飞速发展的行业，没有什么东西是金科玉律。目前业界一般的做法是让开发者自己做单元测试。这是因为单元测试是一种白盒测试，粒度比较细，要求测试者对代码有较深层次的理解，而对代码的理解无人比得上开发者本人，所以让开发者自己来测是高效率的做法。当然，若贵公司采纳结对编程（pair programming）的做法，自然可以让结对的两个人一个开发一个做单元测试了。

既然自己写的代码要自己测，那么开发者当然不会自找麻烦地写出难以测试的代码出来。巧的是，容易测试的代码基本上可以和设计良好的代码划等号。因为一个测试用例其实也就是一个单元的最早用户。容易使用显然意味着良好的设计。

3. 单元测试用例可以当文档使用

不知道您有没有这样的经历？当不清楚一个 API 函数怎么使用，而去寻求文档的帮助时，往往会扫一眼这个函数的原型定义，然后跳过大段的英文说明直接去看文档中提供的样例程序，然后在自己的程序中照着样例依葫芦画瓢调用这个 API。那么，您有没有意识到，文档中的样例程序和对这个 API 函数的单元测试程序有异曲同工之妙呢？若有了对这个 API 的测试用例，那么同样可以依葫芦画瓢☺

本书导读

给单元测试做完广告之后，再回过来介绍一下本书主要内容吧。

若您主要用 Java 做开发工作，那么恭喜，这本书简直是为您量身定制的。本书介绍

了 JUnit 的使用，JUnit 与 Ant、Maven、Eclipse 的集成，Mock Objects 的使用（EasyMock 和 DynaMock 框架都有讲到），还有用 Cactus 进行容器内测试的方法；而且还举了很多实例，编写了对 servlet、filter、JSP、taglib、数据库应用及 EJB 的测试用例，基本上您依样学样就能知道如何进行单元测试了。本书还介绍了 JUnit 的设计，让您知其然且知其所以然。

若您主要用其他语言进行开发，那么本书中很多只和 Java 有关的内容对您就不适用了。但在可以从书店找到 CppUnit in Action、NUnit in Action 这类书籍之前，本书依然非常有参考价值。须知，几乎所有的 XUnit 单元测试工具都是参照 JUnit 写就的，那么想必若有人要以其他单元测试工具为蓝本来介绍单元测试，本书也会是他们的重要参考资料☺

在大多数公司，单元测试用例是由开发者自己编写的。但即便您不直接从事产品开发，而是从事产品质量保证（QA）方面的工作，那么本书对您也很有参考价值。因为市场上能购买到的现成测试工具和度量工具毕竟品种有限，在很多情况下您都需要“自己动手，丰衣足食”，为特定的项目打造特定的测试平台。现在“灰盒测试”挺流行，测试平台可能还会调用产品提供的接口。在这种情况下，JUnit 所采用的一些技术，以及本书所介绍的一些 JUnit 扩展所采用的技术，就很有参考价值了。更何况，若测试平台本身比较复杂，您可能还需要为测试平台编写单元测试呢。

最后，希望这个译本能带您在轻松愉快的心情下体验并掌握 JUnit。若译文有不妥之处，请来 mail 指正：wesley.bao@acm.org 。谢谢！

鲍志云

2004 年 9 月于南京

前 言

迄今为止，测试还是人类所能找到的能确保交付的软件正常运行的最好方法。本书是四年来在测试领域研究和实践的成果。实践来自我的IT咨询背景——我曾先后在Octo Technology公司和Pivolis公司供职；研究则来自我在晚上和周末所从事的开源（open source）开发工作。

自1982年开始（那还是我编程生涯的早期）我就对编写能帮助开发者写出更好代码并且提高开发速度的工具感兴趣了。这一兴趣引导我进入了软件咨询、质量改进等领域。近来，我在建立“持续构建（continuous-build）”平台，同时也在探索最佳开发实践。这两者都需要完善的test suite[1]来支持。测试同编码活动越接近，从代码获得反馈就越快，所以我对单元测试很感兴趣。单元测试与编码活动如此接近，以至于现在如同编写代码那样成了开发活动的一部分。

于是，我参与了一些同软件质量相关的开源软件项目。它们是：

- Cactus，用于对J2EE组件进行单元测试（http://jakarta.apache.org/cactus/）。
- Mock Objects，用于对任何代码进行单元测试（http://www.mockobjects.com/）。
- Gump，用于持续构建（http://jakarta.apache.org/gump/）。
- Maven，用于构建和持续构建（http://maven.apache.org/）。
- PatternTesting，用AOP来检查构架和设计规则，这是个概念验证性质的项目（http://patterntesting.sf.net/）[2]。

---

[1] 译注：有“测试集”、“测试包”、“测试套件”等译法，本书根据上下文或保留原文（比如JUnit中的类名）或译为“测试集”。

[2] 我很想在书中包括一个讲述用AOP框架来进行单元测试的章节，但是我还是没那么做。目前的AOP框架还不成熟，用它们来编写单元测试会导致冗余代码。我预测，专门的AOP/单元测试框架会在近年出现，那时我会在本书第2版中包含它们。想要了解如何用JUnit和AspectJ来测试EJB，可以参考我的blog中有关内容：http://blogs.codehaus.org/people/vmassol/archives/000138.htm。

参与了这些项目，*JUnit in Action* 这本书已是水到渠成。

没人想编写糟糕的代码。我们都希望写出可以正常工作的代码，我们都希望能以自己的代码为荣。但是，常常事与愿违。你是否经常听到这样的话："我们是想要写测试的，但在压力下没足够时间去写。""我们一开始是写了单元测试的，但两星期后我们就没动力继续下去了，再往后我们就放弃编写单元测试了。"

本书会教给你开开心心地编写高质量代码的工具和技能。本书手把手地教你如何高效地运用工具，避免常见的陷阱。本书将让你具备编写可以正常工作的代码的能力，还将帮你把单元测试引入日常开发活动，并建立起一套步骤，让你得以按部就班写出稳健的代码。

最重要的是，本书向你展示了如何控制你软件的熵[3]，而不是反过来被它控制。我想起了Lucretius在公元前94至公元前55年之间写就的*On the Nature of Things*中的一些章节（我就不列出拉丁文原文了）：

Lovely it is, when the winds are churning up the waves on the great sea, to gaze out from the land on the great efforts of someone else; not because it's an enjoyable pleasure that somebody is in difficulties, but because it's lovely to realize what troubles you are yourself spared.

这正是你意识到自己已经用精良的test suite"武装"起来后的感觉。你会看到，别人还在蹒跚挣扎；你会感到欣慰，因为你拥有测试，可以用来阻止任何人（包括你自己）破坏你的应用程序。

Vincent Massol
Richeville（离巴黎很近），法国

[3] 译注：即软件在多大程度上处于有序状态。

致 谢

本书历时一年才完成。我永远感谢我的爱妻，Marie Albane，还有我的孩子，Pierre-Olivier和Jean。在那一年中，他们容忍了我花费大半空余时间来写书，而这些时间本应用于陪伴他们。我不得不允诺，我不会再写另一本书——至少在一段时间内是如此……

感谢Ted Husted，他帮我改进版面、润色文字、重组章节，把书的第一部分组织得更通俗易懂，并替我修润引用之处。

若不是Kent Beck和Erich Gamma编写了JUnit，就不会有这本书的存在，感谢他们的创意。还要特别感谢Erich，他面临着交付Eclipse 2.1的压力，但还是答应阅读本书的原稿，并且对本书作了高度评价。

此外，若不是Tim Mackinnon和Steve Freeman发明了mock objects单元测试方法（这个主题占本书的很大比重），那么本书就不会是这个样子。感谢他们在伦敦Extreme Tuesday俱乐部就着啤酒给我讲mock objects，而我喝的是越桔汁！

如果没有审阅者的努力，本书的质量会大打折扣。感谢Mats Henricson、Bob McWhirter、Eric Hatcher、William Brogden、Brendan Humphreys、Robin Goldsmith、Scott Stirling、Shane Mingins和Dorothy Graham。我还要特别感谢Ilja Preuß、Kim Topley、Roger D. Cornejo和J. B. Rainsberger，他们给出了极为详尽的审阅意见并且提供了极棒的建议。

通过这第一本书，我走进了出版的世界。我对Manning的专业及他们对完美的追求印象至深。我一次次地觉得书已经完成了我可以休息一下了，但总又会有新一轮的校对工作来考验书稿。很感谢出版者Marjan Bace，虽然我一直在推迟交稿时间，但是他始终信赖我。责任编辑Marilyn Smith是效率的典范，总能在我递交样章几小时之后就返回给我改正过的样章和建议。文稿编辑Tiffany Taylor改正了大量的错误（经过Tiffany的狂风暴雨洗礼之后，我自己都几乎无法认出我写的章节）。Tony Roberts则承担了给书排版的重任，并帮我实现了很多排版上的要求。谢谢你，Tony。技术编辑Robert McGovern则漂亮地完成了给我找出技术错误的工作。

最后（但并非最次要），我还要感谢Octo Technology公司和Pivolis公司的CEO Francois Hisquin。在写作本书期间，我先后工作于这两家公司。在Francois的允许下，本书有一部分是在工作时间里写就的。

关于本书

本书是一本实例驱动的教你“怎样做”的书，主题是使用JUnit框架及其扩展对Java应用程序（包括J2EE应用程序）进行单元测试。本书所面向的读者包括软件构架师、开发者、测试团队成员、开发经理、使用极限编程方法的程序员以及各种敏捷方法的实践者。

本书讲述如何解决现实中测试方面的困难问题，比如如何对遗产应用进行单元测试、为真实对象编写真实测试、使用测试度量、测试自动化、隔离测试等等。

本书特色

有一些特色内容贯穿全书。

最佳实践

JUnit社群已经采纳了一些最佳实践。本书提到它们的时候会用标注框概要介绍这些最佳实践。

设计模式实战

JUnit 框架使用了一些著名的设计模式。当我们第一次讨论到一个使用了某种设计模式的组件的时候，我们会用标注框来定义模式并指出 JUnit 框架中哪里用到了它。

软件目录

在整本书中我们提及了如何使用JUnit的扩展和工具。为了提供方便，我们在本书最后的“参考文献”部分列出了所有这些软件包。在“参考文献”部分，我们还列出了提到过的其他书。

阅读指引

本书分成3个部分，第1部分是“JUnit精粹”，在这部分我们向你介绍了一般意义上的单元测试，并特地介绍了JUnit。第2部分“测试策略”则探讨了测试专业软件中复杂的对象的方法。第3部分“测试组件”探索了测试常见子系统的策略，这些子系统包括servlet、filter、JSP、数据库等等，甚至包括EJB。

第1部分：JUnit精粹

第1章带着你为一个简单的对象创建测试。在此过程中，我们介绍了单元测试的好处、理念和方法。随着测试变得越来越复杂，我们把JUnit作为创建更好的测试方案来展现。

第2章进一步深入研究了JUnit的类、生命周期和构架。为了把这些联系起来，我们描述了一些测试的例子，这些例子和你为你自己的类写的测试差不多。

第3章描述了一个复杂的test case[4]，以便向你展示JUnit对较大的组件如何工作。这个案例所分析的是很多应用中都有的组件：controller。我们介绍了案例代码，标出哪些代码是要测试的，然后展示如何测试它。我们一旦知道代码能按照我们的意愿正常工作，就开始为异常条件编写测试，以确保哪怕出了问题代码也能正常运行。

第4章讲述不同类型的软件测试以及它们在应用生命周期中扮演的角色，并讲述如何为可测试性进行设计（design for testability），如何实施测试先行的开发（test-first development）。

第5章探讨了把JUnit整合进开发环境的各种方式。我们提及了用Ant、Maven和Eclipse来自动化JUnit。

第2部分：测试策略

第6章描述了如何用stub来执行单元测试。这一章介绍了一个连接到Web服务器的示例应用并描述了如何用stub方法对调用远程URL的方法执行单元测试。

[4] 译注：常译作“测试用例”，本书为了与JUnit中类名保持一致，多数情况下保留原文不译。

第7章展示了mock objects方法，这是一种让你可以把代码从周围领域对象隔离出来测试的方法。该章接下来继续讲述那个示例应用（打开到Web服务器的HTTP连接），并展示如何为这个应用编写单元测试，着重指出mock objects和stub的不同。

第8章展示了另一种对J2EE组件很有用的单元测试方法：容器内测试（in-container testing）。本章讲述了如何用Cactus来在容器内运行单元测试。此外，我们分析了in-container方法和mock objects方法的优劣，以及何时该选哪种方法。

第3部分：测试组件

第9章展示了如何使用mock objects和in-container方法对servlet和filter执行测试，强调了这两种方法如何互为补充，并给出了何时选用哪种方法的策略。

第10章带我们走进单元测试JSP和taglib（标记库）的世界。这章展示了如何运用mock objects和in-container策略。

第11章则涉及了一个艰难但重要的主题：对通过JDBC访问数据库的应用程序进行单元测试。该章还展示了如何把数据库代码同数据库隔离开进行单元测试。

第12章则研究了如何使用mock objects、单纯的JUnit test case和Cactus来测试各种EJB。

代码

本书中的例子的源代码已贡献给Apache Software Foundation，可在SourceForge（http://sourceforge.net/projects/junitbook/）上找到。本书的Web页面（http://www.manning.com/massol）上也提供了到源代码的链接。附录A给出了源代码的组织细节和软件版本需求。

我们展示的Java代码中Java的关键字是用粗体显示的，这样可以提高代码的可读性。此外，当我们想要突出一段新代码中的改变时，改变也用粗体显示以吸引读者注意。在这种情况下，Java关键字就用非粗体的标准代码字体显示。代码中常常会出现数字和注释，这些数字指向紧跟在代码后面的讨论。

在本书中，我们用一种monotype字体来表示代码（JSP、Java和HTML）、Java方法、JSP tag的名称，以及大多数其他源代码标识符。

- 在正文中，引用方法之处可能不包含完整方法签名，因为方法也许具有不止一种形式。

- 在正文中，引用XML element或者JSP tag之处通常不包含括号或者属性。

参考资料

参考资料用下标或者在正文中标出。完整的出版信息和URL在本书最后的“参考文献”部分。正文中给出了Web站点URL并在索引处交叉引用。

作者在线交流

购买*JUnit in Action*一书也意味着你获得了免费访问Manning出版公司所运营的一个私有Web论坛的权利，在这个论坛上你可以对书作出评价，可以提出技术问题并从作者和其他读者处获得帮助。在http://www.manning.com/massol注册并访问论坛。这个页面提供了注册过后如何访问论坛的信息，也说明了你可以获得什么帮助，并且说明了论坛的规则。

Manning对读者的承诺是提供一个让读者之间以及读者和作者之间得以有效沟通的场所，但并不承诺作者的参与程度。作者对该论坛的参与是自愿的，并且没有物质回报。我们建议你向作者提一些有挑战性的问题，这样他才会有兴趣持续参与。

只要本书尚未绝版，“作者在线”论坛和以前的讨论的存档（都位于出版商的Web站点上）就会持续向读者开放。

关于作者

Vincent Massol是Jakarta Cactus框架的创建者，还是Maven、Gump、MockObjects开发团队的积极参与者。在4年中，他担任了几个大项目（大多数是J2EE构架）的技术构架师，之后他就和别人一起创立了Pivolis并担任CTO。Pivolis公司专注于把敏捷方法应用于软件外包。在白天，Vincent是一位顾问和演讲者；在晚上，他是开源软件的开发者。Vincent目前居住在法国巴黎，可以通过他的blog http://blogs.codehaus.org/people/vmassol/来联系他。

Ted Husted是Struts开发团队的积极参与者，JGuru Struts Forum的管理者，Structs in Action[5]的主要作者。作为顾问、演讲者、讲师，Ted和美国各地的多支Java开发团队合作过。Ted最近的开发项目自始至终使用了测试驱动的开发方法（test-driven development），可以通过http://sourceforge.net/projects/wqdata/上了解这个项目。Ted和他的夫人、两个孩子、四台电脑、一只老猫一起居住在纽约的Fairpot。

[5] *Structs in Action*，Ted Husted、Cedric Dumoulin、George Franciscus、David Winterfeldt，Greenwich，CT：Manning，2002。

关于书名

Manning的*in Action*丛书既包括了概览内容也包括了教你怎么做的实例，这种方式鼓励你理解记忆。认知科学告诉我们，在发现和探索的过程中记忆的效果是最好的。我们Manning的同事认为，探索是实践。我们相信，每当计算机科学家们创建一个新的应用，他们会实践新概念和新方法，看一下它们是否能让新的程序比上一个程序更好。“实例驱动”是*in Action*丛书的一个基本要素。该丛书中的每一本都鼓励读者实践新代码，探索新构想。我们Manning的同事相信，牢固的掌握来自探索和实践，而且最重要的是，来自同他人分享我们探索所得的过程。人们在实践中（in action）学习效果最好。

书名中的“in Action”还有一个更现实的来由：我们的读者都很忙。他们因工作需要或者为了解决问题而寻求书籍的帮助。他们需要的是让他们可以快速上手的书——也就是可以在实践中（in action）帮助他们的书。该丛书就是为这些“缺乏耐心”的读者设计的。你可以从任何一页开始读，并只阅读你解决问题所要用到的内容。

关于封面

*JUnit in Action*的封面是“Burco de Alpeo”，选自1799年初版于西班牙马德里的一本各地服装简编。那本书的扉页中提到：

Coleccion general de los Trages que usan actualmente todas las Nacionas del Mundo desubierto, dibujados y grabados con la mayor exactitud por R.M.V.A.R. Obra muy util y en special para los que tienen la del viajero universal

我们尽可能地照字面翻译：

搜集了已知世界各国的服装，由R.M.V.A.R进行高精度制图和印刷。本书对正准备进行世界旅行的人特别有用。

虽然我们对制图、刻版和手工着色的人一无所知，但是这幅画表明了他们的作品的精度，而这只是这本彩色简编中很多画中的一幅。这些画的多样性生动地表明了200年前世界各地的城镇和区域的独特个性。在那个时候，即便两个地方相隔几十公里，不同的服装风格就能区分出哪些人属于哪个地方。这本简编让我们真切感受到了那个时代（以及除了沟通极为方便的现代之外的任何一个历史时期）的隔离与距离。自那以后，服装风格已经改变了，那时各地如此丰富的多样风格，在今天也模糊同化了。现在已经常常很难靠衣着分辨出哪个人居住在哪个洲了。或许，客观地看，我们用文化和视觉上的多样性换取了个人生活的多样性，或者换取了更多彩也更有趣的精神与科技生活。