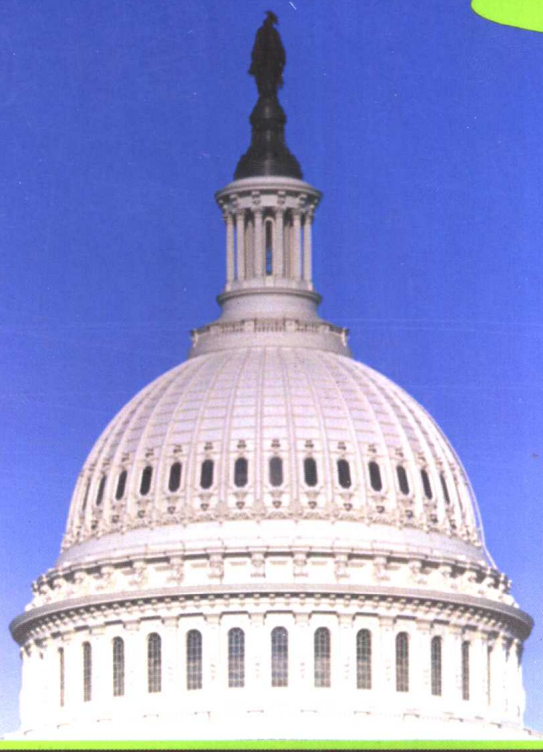
HZ BOOKS

软 件 工 程 技 术 丛 书

前沿论题系列



Java极限编程

Java Tools for Extreme Programming: Mastering Open Source Tools Including Ant, JUnit, and Cactus

（美） Richard Hightower
Nicholas Lesiecki 著

唐一丁 蔡永航 译

机械工业出版社
China Machine Press

软 件 工 程 技 术 丛 书

前沿论题系列

Java极限编程

Java Tools for Extreme Programming:

Mastering Open Source Tools Including Ant, JUnit, and Cactus

(美) Richard Hightower
Nicholas Lesiecki 著

唐一丁 蔡永航 译

机械工业出版社
China Machine Press

本书通过介绍一系列开源Java开发工具和生动的实例，描述如何用极限编程（eXtreme Programming）理论指导具体的开发行为，从而使Java开发更加灵活、高效、低成本。

本书主要讲述J2EE和极限编程的基础，重点介绍如何掌握极限编程过程中最难的部分：测试、整合和部署。全书以基础开篇，接着辅以范例并对著名的宠物商店范例进行功能性剖析。在了解了各个工具的概念后，作者介绍了测试的实践，一步一步指导读者掌握极限编程开发的技术要点。

本书内容丰富、详实、生动，适合作为Java软件开发人员的参考书。

Richard Hightower，Nicholas Lesiecki：Java Tools for Extreme Programming: Mastering Open Source Tools Including Ant，JUnit，and Cactus (ISBN: 0-471-20708-X).

Authorized translation from the English language edition published by John Wiley & Sons, Inc.

本书版权登记号：图字：01-2003-1688

图书在版编目（CIP）数据

Java极限编程/（美）海涛尔（Hightower, R.）等著；唐一丁等译. -北京：机械工业出版社，2004.1

（软件工程技术丛书　前沿论题系列）

书名原文：Java Tools for Extreme Programming: Mastering Open Source Tools Including Ant，JUnit，and Cactus

ISBN 7-111-13104-5

Ⅰ.J…　Ⅱ.①海…②唐…　Ⅲ.Java语言－程序设计　Ⅳ.TP312

中国版本图书馆CIP数据核字（2003）第087857号

机械工业出版社（北京市西城区百万庄大街22号　邮政编码　100037）

责任编辑：李　英　　贾　梅

北京中加印刷有限公司印刷·新华书店北京发行所发行

2004年1月第1版第1次印刷

787mm×1092mm 1/16·27.25印张

印数：0 001 - 4 000册

定价：58.00元

凡购本书，如有倒页、脱页、缺页，由本社发行部调换

本社购书热线电话：（010）68326294

译者序

我们先来看看来自未来某本书中的前言：

曾经的互联网（规范定义#1997xxxx&am）是一个谁都可以使用（包括高智商的猫狗动物）、不需要付完全责任、自由使用的联系方式，人们用它来发布公司组织（合法和非法的）的广告和大范围招聘消息，组织全世界范围的技术团队来合作进行人类生产方式的优化研究。在当时，最终得到肯定的网络安全人员可以成为两种人，一种是当时的黑客，一种是当时的电子警察（有意思的是，他们是现今采用的软件安全代理的祖先，当时由人类实体来担任），当时的黑客因为某些软件技术核心的不成熟或者根据系统设计的思路，修改和恶意复制资料，以此表达对对方的不满和对对方的重视程度，正如人类近万年的战争史，那是一个以折磨弱势群体为目标的社会，当时的他们只证明了每个灵长类将被其他更高智商生物取代的事实，却不能尽早认识到地球资源对于灵长生物的有限性，而我们现在的地外资源信息中心在太阳系以外仍在继续尝试和努力。现在的地球数字互联正如你现在身上的所有设备，每个都有一个生命ID，截至目前你可以在Galobal Digital Device Library 查询到每天全球有数以兆亿的数字设备在和近1000 亿具有活性细胞的高等生物与人类一起建设和生活。（笔者摘录于2002年《计算机是未来趋势？》）

Java及其近乎遍及全球的应用是在开放源代码领域孕育和成长起来的，我们再也不需要苦苦等待软件公司产品的发布速度和升级周期，在开源领域，每天都在诞生有关Java令人激动的新技术。Never wait!Now!Java!

极限编程：交流、简易、反馈和勇气。让我们的小型团队合作更加轻松而有效。我们可以持续整合，我们可以进行配对合作、迭代系统、极限建模、过程分离、阶段冻结、质量保证、里程监督、角色轮换、小版本发布、策略规划、系统重构等等。

正如Roy W. Miller所言："坦率地说，XP（或者任何其他灵活方法）根本就不重要。它能够产生的结果才是关键。如果XP这样的灵活方式可以帮助你更快地开发更好的软件而少受痛苦，那么它值得考虑。"

本书的精髓可以堪称经典，对于作者的很多软件思想和缜密的设计，译者是在第三次校稿时才颔首称是，同时也对作者深厚的设计功力表示由衷的赞叹。此书虽小，但对开源软件领域的主题思想把握得恰到好处。实例从简单见深入，又由复杂演绎简单，最终简单和复杂相结合，形成天然的结合，宛如大自然的巧夺天工。

此外，在开发工具的基础上，再为大家介绍一些与开源的软件工程相关的项目。你可以在http://www.tigris.org站点（本身是http://www.collab.net/developers的子项目）上找到以下项目：Subversion、Scarab、ArgoUML、GEF、Eyebrowse、Anzu、Style、Axion、binarycloud、RapidSVN。另外，作为一个与时俱进的软件开发者，不应该长时间闭门造车，也许每天习惯性地访问http://sourceforge.net站点，或者参与一些Jini项目应该比抱怨技术发

展太快有效得多！相信你已经有了明智的选择——软件技术不断在迎接新的曙光！

本书中的最新范例可以从http://www.rickhightower.com/CaseStudies 下载获得；书中提到的其他工具（Ant、JUnit、Cactus、HttpUnit、JMeter、JUnitPerf、JXUnit、MockObjects等）都可以在http://jakarta.apache.org、http://sourceforge.net下载最新版本；PetStore最新的源代码可以从Sun网站http://java.sun.com/搜索获得。

本书在我的同伴蔡永航的大力协助下才得以与大家见面，这与他辛勤地彻夜排错和努力地审查代码是分不开的，他已成功地通过了SCJP考试，但其本人却是工商管理科班出身，这使我有机会向他请教诸多经济学方面的问题；另外，华章的贾梅的不断鼓励和耐心指导为本书的早日面世打下了坚实的基础，在此致以崇高的敬意；另外，南京大学的钱海波的协助大大加速了本书译作的完成；南京新东方的王璇在语言上不吝赐教，南京的曹晓钢多次为本书的加速亮绿灯等等。还要感谢母校南京工业大学的颜艳燕老师在学习和心理上的一贯引导，使我能带着一颗平常心看待各类问题；感谢崔北亮——使我顺利走上现在的道路；还有我在Mars Studio的伙伴们，你们是我坚实的后盾！

由于水平有限，译作中若有值得商榷之处，欢迎联系本人tangyiding@yahoo.com.cn，或到本人Blog（http://www.086net.com）留下宝贵意见，本人悉心听取。

感谢原书作者Richard Hightower！也感谢你选择此书！

祝大家好运！

唐一丁

2003年3月31日

致　谢

首先我需要感谢我的妻子Kiley Hightower，在如此紧凑的写作进度中，她给我非常大的支持。我在咖啡店与我的笔记本度过了很多个夜晚，每次都直到咖啡店关门我才离开那儿。

我真诚感谢在出版社与我一起工作的Tim Ryan。他负责安排合著者、编写的参与者、技术编辑的工作，并且将作者与团队之间的合作事务安排得井井有条。Tim在本书编写的过程中不断地为本书提出建议以及询问相关的问题，这很大程度上引导了我的写作。Tim相当能干和敬业。再次感谢你！

所有荣誉应属于本书的合著者Nicholas Lesiecki！当我将本书的大纲交给Tim的时候，Wiley出版社对这项工作很感兴趣，并且将它放进即将完成的书籍列表中。我答应了一个相当有挑战性的日程表，甚至开始考虑是否需要休假来完成它，但是最后决定还是不这么做了。因为此时Nick帮助了我，使得我步步接近目标，同时还分担了许多工作，并且最终成为了本书的合著者。感谢Nick。

万分感谢Scott Fauerbach！Scott为宠物商店应用程序的基础版本做了详细的检查工作，且将其大大简化。最初这个应用程序相当抽象，并且在理解上有一些困难。Scott将其分解并将其变得尽可能的简单。Scott与我曾经在一些公司的团队中共事过——我们合作得很愉快。Scott在宠物商店应用程序商业层中的会话bean案例编写上表现相当出色。感谢Scott。

Erik Hatcher在Ant和JUnit的使用上有丰富的经验。我向他请教过许多次。当你在使用Ant方面遭遇困难的时候，身边有一个Ant的主题FAQ的编者将对你产生莫大的帮助。感谢Erik所做的一切支持工作。

我还要感谢Andy Barton允许我免费存放本书的主题站点，感谢Ron Tapia和Chris Lechner对站点进行维护。还要感谢Ron对程序进行调试，并将程序从SQL Server转换到Hypersonic SQL。

感谢Jon Thomas与Paul Visan对案例编写付出的努力。我还要感谢所有参与本书编写的人，Paul Visan、Erik Hatcher、Jon Thomas和Douglas Albright。

感谢Mary Hightower和Richard Hightower（我的母亲和父亲），我的三年级老师Wooten女士和Whitney Hightower的理解。

最后，但不表示不重要，我要感谢本书的编辑Avery Regier和Tiffany Taylor。Avery是一名技术编辑，在我看来他所做的工作已经超出了他的工作职责。他对我提出了许多不错的建议和看法，并且这些建议和看法与书中的内容完美融合在了一起。感谢Avery。Tiffany是一名版面编辑，她找出了每一个错误的制表符并且阅读了所有的句子——目前书中仍有错误的话，这个责任应该由我承担，而不是她。让我们赞许她一丝不苟的工作。感谢你，Tiffany。

作者简介

Rick Hightower是eBlox的开发主管，在那里他领导实施新式的软件开发过程，例如极限编程（Extreme Programming），同时他还负责有关编码标准的工作和软件开发的指导工作。

Rick是一名拥有十年以上软件开发经验的程序员。他在框架与工具、强制过程（enforcing processes）方面，在使用J2EE、XML、UML、CORBA、JDBC、SQL、Oracle技术开发企业级应用程序方面都具有相当丰富的经验。

他以前是Intel公司企业架构实验室里的一名Java架构高级软件工程师。他在那儿曾领导一个开发团队，设计和实现三层客户端-服务器端(C/S)应用程序；还在他所在的部门中引入了O-O CASE工具。后来又使用Java、COM、CORBA、中间件技术创建了一些框架。Rick还创造了ICBeans，并且拥有此技术的专利权，因此他还受到了Intel公司的表彰。

Rick曾经协助参与过一些Java书籍的编写，并且为Java Developer's Journal撰写文章。他还教授开发企业级JavaBeans、JDBC、CORBA、Applet（作为CORBA客户端）等相关课程。

Nicholas Lesiecki身为技术团队负责人供职于eBlox，在那里他负责团队开发和架构公司的垂直应用程序框架。在互联网信息爆炸革命时还是初生牛犊的他已经成为了一名Java大师。他以Java是美国第五大编程语言而自豪（根据Brainbench.com的标准）。在Jakarta中的Cactus项目中他表现十分活跃，并且习惯于通过测试和不厌其烦的重构来改进代码结构。

Nick居住在Tucson Arizona市中心，在那里天空中的晚霞洗涤着他的思想和大脑中的各种奇思怪想。他与他的妻子Suzanne和他们的“孩子”——一只爱德加狗和一只可爱的猫居住在一起。这三者想必也都做出了相当大的贡献，因为他们无私奉献着本应与Nick共同度过的时光。Nick深爱着他的妻子。

其他参与者

Doug “Chakka” Albright是一名软件开发人员，他编写了Ant标记参考。

Jon Thomas 是位于Tucson AZ的HealthTrio的高级系统工程师。他在就职于HealthTrio之前曾在eBlox、SimplySay和GroupSystems.com公司开发过产品，在开发过程中同时使用Java和VB。来自洛杉矶的他，在1995年“逃离”了隐藏着黄金的美国西南部。他与他的家人居住在Tucson，在那里他每天都在编写软件，还有祈祷，并且与明尼苏达的维京人共同欢庆节日。Jon编写了本书Ant的API参考。除此之外，Jon还编写了宠物商店应用程序范例的案例学习，此案例对应用程序进行转化以使其应用XML/XSL技术作为表现层。

Erik Hatcher拥有超过十年的软件开发经验，最近主要是用J2EE进行开发工作。他是Sun认证程序员，倾心于Apache软件组织的Jakarta项目，并且是jGuru中Ant FAQ的维护者。Erik目前是Virginia大学的Darden企业管理研究所的高级Java架构师。他编写了关于如何使用HttpUnit的案例部分。除此之外，当我在使用Ant遇到障碍难题时，他提供了必要的技术

帮助。

Paul Visan是eBlox的技术领导者，并对J2EE开发十分热衷。他来自罗马尼亚，在那儿他取得了很大的成就。他对宠物商店应用程序进行了转换以使其使用Struts，并且对应用程序的JSP和Struts版本进行了HttpUnit测试。除此之外，他还编写了案例以解释怎样将宠物商店的JSP部分转化为Struts。

Scott Fauerbach是一名首席软件工程师，他专攻Java领域的用户界面和服务器端的开发工具以及软件框架的工作。当Java还处于beta版本时Scott便已是一名专业的Java程序员，并开发了一整套名为Widgets的轻量级GUI组件，该套组件被包含在Visual Café的第一版中。

Scott目前在FourthStage工作，致力于使用Java、SOAP与JSP构建下一代B2B（business-to-business）软件。Scott参与了宠物商店应用程序的编写，并且编写案例研究把该应用程序的业务层移到几个企业会话bean上。

Ron Tapia是eBlox的系统工程师，他帮助我设置整个网站并且在Linux上调试运行此应用程序。

前　言

本书讲述了在极限编程的实践中如何使用开源工具实现自动测试和持续整合的技术。

让我们把这句话展开来说。自动测试和持续整合是极限编程软件开发方法中12个核心实践中的两个。极限编程是一种轻量级的软件开发过程，其关键价值是反馈、交流、简单和勇气。在第1章中我们将会说明完整的极限编程的实践过程；目前能说的是极限编程是由“通用的开发实践”和“正确的方法”所组成的。

在通用的开发实践中包含测试和不断整合的工作。几乎没有哪个软件开发商在开发中放弃这些步骤，毕竟一个系统需要经过整合才能发布，并需要通过测试确保用户能够接受开发出来的产品。由于互联网业的萧条，使得大量不遵守这些实践原则的软件开发商退出了这一行业。但是还是有许多软件开发商要么抗拒这些做法，要么即使他们承认必须这样做，却抱怨“忙都忙不过来了，哪有时间做这些事情。”本书正是解释和阐述了软件工具的妙用，以帮助读者将这些有价值的开发实践应用于软件开发中。

为什么要在工具的使用上花费大量的时间

极限编程是一个以人为中心的开发哲学；所以如果我们将注意力集中在工具上是极具讽刺意味的。极限编程方法认为软件编写过程中的关键挑战是来自于人的挑战——例如将人聚集在一起工作，帮助程序员学习和控制情感。它的四个关键价值（交流，反馈，简单，勇气）都以人为本。至今为止，大多数关于极限编程的书籍将重心放在人身上：这些书都在概括和传播极限编程的哲学及思想体系（Kent Beck认为《Extreme Programming Explained》可以作为极限编程的宣言），同时这些书中还说明编写软件的感觉。通过这些努力，Kent Beck与极限编程的发明者们遵从他们自己的哲学：先解决最重要的问题。但是，现有的书并没有覆盖所有实践极限编程方法的技术细节，这正是这本书产生的原因。

我们将解释在Java环境下（尤其是在J2EE环境下，尽管大多数工具在任何环境下使用起来都差不多）怎样实现持续整合和自动测试。我们将着眼于某些技术细节并提供有关范例以演示工具的使用。我们还将为大家讲述如何使用JUnit、Cactus、HttpUnit、JUnitPerf和JMeter编写自动测试，如何使用Ant（与前面所列的工具结合使用）来实现持续整合。

谁应该阅读这本书

尽管本书的观点来自极限编程，但你却没有必要通过实践极限编程才能从本书中获益。任何一个希望在自动化测试和持续整合方面得到帮助的人都能从这些工具和范例的实践中受益。如果你对极限编程一无所知，你应该参考第1章以便理解前言的其余部分，这样你才能够了解本书中某些做法的意图所在。实际上，前言部分介绍了自动测试和持续整合对于所有程序员的

价值。本书假定你是一名有一定经验的Java程序员。因为书中讲述了应用程序的测试和J2EE平台下的整合工具，本书还希望读者通晓J2EE技术和拥有开发经验。那些对J2EE应用程序开发不感兴趣的人也能从中找到大量有价值的材料，因为任何一个Java（JMeter与HttpUnit甚至不仅仅局限于Java项目）软件项目都能够应用书中的大部分工具。如果程序员不熟悉J2EE但又想将这些技术和工具应用于J2EE应用程序中的话，那我推荐你阅读一本全面介绍J2EE的书籍——《Developing Java Enterprise Applications，2nd edition》，作者为Stephen Asbury与Scott Weiner。

为什么要开源

我们很难对开源运动在软件领域的发展视而不见。当然，开源此时是一个“强意词”，但是开放源代码的开发工具较传统的工具提供了更加引人注目的优势——尤其在极限编程的开发中。我们将优势归结为两点：第一，开源工具很实用。第二，开源与极限编程有着相近的哲学思想。

开源工具的优势：

价格合理。开源软件几乎都遵守免费原则；本书中的所有工具我们都能通过Internet免费获得。免费软件意味着你或者你的公司不必破费就能使用它们，这总是有好处的，但是在这里不是主要的。主要的好处是你采用的这些工具不会受到你的老板或管理者的阻拦，因为他们担心这些软件只是一时流行。比如说，一旦你下载了JUnit，你喜欢上了它并且在团队内极力推广——加速软件开发和改进了质量——绝对没有人会阻止你的行为。如果在开始采用极限编程之前，你必须支付7500美元购买相关部署软件，这一定会遭到质疑的，不是吗？

工具是高质量的。程序员们每天都使用开源的开发工具。因为改进工具意味着直接改进他们的工作环境，所以开源开发工具不断接受改进和进行bug的修复。这项工作快速且持续不断地进行着。

工具符合标准。本书中涉及的工具，尤其是JUnit和Ant，在它们所处的领域都是标准。无数的开源项目使用Ant，而JUnit（一些工具基于它）的作者为Kent Beck（极限编程之父）与Erich Gamma（《Design Patterns: Elements of Reusable Object-Oriented Software》的主要作者之一）。

极限编程与开源软件的配合

极限编程与开源的发展是紧密联系的。两者都鼓励开放，给出一种合作开发的方式——如果你愿意，也可以采用这样的方法参与其中。两者的哲学都承认人类的弱点——世界上不存在完美的代码，并赞同帮助他人查找和修复错误的人应该得到大家的感谢。所有的开源代码都是共享的（这正是极限编程强调的）。许多开源项目使用自动测试，并从中获益。使用自动测试对开源项目来说尤其重要，因为代码的来源广泛，而且必须进行整合。两个系统都要求进行小型且增量式的软件发布。当然，两者的哲学同样重视代码——开源以阅读代码是愉快的、受教育的和有帮助的这一条件为前提。

它们两者的共通点全部罗列出来的话要花上不少功夫。通过使用开源工具（以反馈、协

助以及代码的形式将意见转达给开源社团）即等于实现了开源的一些价值或做法，这些价值和做法正可以将极限编程发扬光大。

> **阅读源代码**
>
> 如果你在寻找关于工具的更多信息，最好的地方就是它们的源代码。除了工具包含的Javadoc文档（另一份唾手可及的参考资料）以外，源代码便是工具行为最权威的发言人。开源软件存在的原因之一（除了人们喜欢免费这一因素之外）就是程序员可以深入研究软件开发者的作品。通过仔细阅读源代码，你能深刻了解程序是怎样实现这些工作的，并且领略编程本身的艺术魅力。如果你在使用工具时不幸遇到bug，那么阅读源代码就能帮助你确定bug到底在何处。

自动测试：概述

极限编程注重测试，将测试作为软件开发流程的核心。在《Extreme Programming Installed》后记中，Dan Rawsthorne说："极限编程之所以成功，在于它以产品的品质为中心，而不是以最后发布的软件产品为中心。"通过不断对软件进行测试，验证软件是否正常工作和软件是否符合客户的需求。自动测试确保测试是连续进行的。如果没有测试，一个团队只能靠猜测来判断软件是否符合需求。在缺少自动测试的情况下极限编程不能执行，开发也不可能完成任务。所有的软件项目都需要使潜在的用户满意并且没有错误。

测试和重构

重构（将已有的代码变得更加简单、明了，或添加功能）是另一个极限编程核心实践。重构是不能脱离测试进行的。如果你不采用极限编程，那么你不可能坚持重构。尽管大多数稳定或者难以改动的项目只需要偶尔的修改，但为了能够正确地修改系统，程序员必须修改现有的设计。此时自动测试将发挥其应有的作用。

面向对象程序设计（在某种程度上也包括其他的程序设计风格）将接口与实现分离。理论上，这意味着你可以改变类或方法的内部逻辑，而不必同时修改与之相关的代码。已经有很多书讨论了这种强大的抽象技术。但是，如果在实践中，程序员由于害怕干扰与接口互动的代码而不敢改变底层逻辑，那么将接口与实现分离实在没有任何意义。广泛的测试（频繁地运行测试）可以确保系统按照正常的方式运行，并且允许自由修改内部的逻辑。任何因变动所导致的问题都会被测试捕捉到。如果测试的时候设计A与设计B产生同样的结果，那么这两个设计便可以互换。有了测试，程序员便能放心对代码进行重构，因为测试可以保证代码的正确运行。

自动测试的种类

在极限编程中单元测试（unit test）是最常被提及的测试，但这只是众多测试中的一种。

单元测试需要与整合测试（integration test）、功能测试（functional test），及一些其他的辅助测试（auxiliary test）——如性能测试（performance test）及回归测试（regression test）等等共同合作，以确保整个系统的完全运行。

1. 单元测试：JUnit

单元测试是极限编程实践中的第一线测试（同时可能是最关键的测试）。编写一个单元测试，就是取出一个“单元”的程序代码，并测试任何可能产生的错误。单元测试，通常会检验类中所有公共接口中的方法。优秀的单元测试，无需测试类中所有行为组合，也无需测试某些极其简单的方法（如Get与Set方法）。更确切的说，单元测试提供了一个重要的常识，即确保单元代码的行为是我们预期的。有了这项工作作为保证，公共接口才变得有意义。这种方式可以使改变单元中的行为更加容易，同时也提供了一个单元行为的简便（并且是可以验证的）的向导。并且程序员能通过对测试的讨论发现类或方法的使用目的。

在极限编程中，单元测试是每天编码周期的一部分。理想的方式是先写测试再写程序代码，并且在具体实现中将测试作为向导。本书两位作者都是以这种方式设计程序的。同时我们发现，如果没有单元测试的向导与纠正，我们几乎无法开展工作。当完成一个单元测试，开发团队便将测试加入项目的测试套件中。这个测试套件每天都要执行好几次，并且所有的测试必须都要通过，单元测试100%的通过率比以下替换办法更合理：那就是有一部分关键的代码不工作。（如果这些程序代码不是关键的，那它为什么会在项目中出现？）

要建立一个高品质的系统，必须确保每一个类的正常运行，因为只有这样才能保证整个系统的运行无误。单元测试同时使程序代码架构变得简明易懂。如果一个软件程序员在同一段程序代码中的不同地方编写三次测试，人的惰性会促使他将这些代码移至另一个相同的地方。

JUnit是一种轻量级的测试框架。JUnit的作者Kent Beck和Erich Gamma依据SUnit设计出JUnit，SUnit是一个应用于Smalltalk并且取得成功又广为大家欢迎的测试框架。由于这个框架非常简单，因此其本身不断地被采用和扩充。本书介绍的所有测试工具都可以与JUnit互动或扩充自JUnit框架（除了JMeter，因为它是一个GUI工具）。

2. 整合/容器内的测试：Cactus

假设单元测试涵盖对象X，那么相关的对象Y与Z怎么处理呢？这三者共同组成子系统A。单元测试被假设是具有独立性的。一个好的单元测试，是无论系统是多么的混乱，它仅仅确认它所测试的类的功能是否与期望达成一致。有许多论文讨论到如何避免单元测试依赖性的方法。（这些论文可以在http：//www.junit.org 网站中找到，这些论文的核心思想是提供一个与测试类相关联对象的模拟实现。）不管如何，单元测试还是应该尽可能的独立。

在《Extreme Programming Installed》一书中，Jeffries等人对于只出现于类间合作中的错误做了一个有意义的观察。他们说：“以我们的实践经验告诉我们，这类错误很少发生。据我们猜测，或许是由于我们先编写了有关测试，所以避免了这类错误的产生。”他们也承认，“当显示出这种错误信息时，这样的问题确实很难解决。”一个好的单元测试，确实应该捕捉到大多数的错误。而系统整体的行为，则是由验收测试（acceptance testing）（也称为功能测试）来检验。但是，好的单元测试也应该检查子系统的行为。整合测试

（integration testing）则负责单元测试与验收测试之间的灰色地带，整合测试提供的是一种完整的测试，它测试所有程序的合作和准确定位介于期望结果与实际结果之间微妙的差异。整合测试一般都没法达到百分之百（例如，可能是相关的类尚未完成）；但成功率一般还是相当高的（介于80%至90%之间）。

整合测试中有一个重要的变化形态，就是容器内测试（in-container test）。在J2EE开发模型中规定组件必须存在于容器中。这些组件依赖容器所提供的服务。因此组件与这些服务的交互必须有相应的检查。尽管有些互动关系可以成功地模拟出来，但是模拟J2EE容器提供的所有服务是非常耗时间的，同时对于此类行为的检查也不是那么完美。由于容器的实现不同，所以有些服务，例如由部署描述符指定的行为就很难被测试到。

Cactus框架提供了存取J2EE Web容器的能力（也可以存取其他类型的容器，如EJB容器）。因为它允许测试容器内的程序代码，Cactus在某种程度上减轻了开发人员进行扩展的负担，以及降低了实现模拟行为的难度（也可以使用真正的服务来代替）。由于是在一个由Cactus所模拟的环境中执行程序，并且这个环境十分贴近真实的运行环境，因此采取这样的方法提供了一个可评估的反馈。当单一的对象只是与容器的服务互动时，容器内测试就犹如快速直接的单元测试。

3. 验收/功能测试：HttpUnit

功能测试则确保整个系统是按照预期的行为运行的，这个测试也称为验收测试。因为这个测试是客户用来检验系统是否已经完成（换句话说，一个Web站点是否完成，取决于其是否可以让用户登录，显示产品，以及线上订货）。有时功能测试会令人怯步（因为功能测试不像单元测试，能立即对生产力有所帮助）。但功能测试对于衡量进度，及找出缺陷都非常重要，缺陷通常来自过去未通过的测试，或者是未完成的功能模块。因为验收是客户的职责，所以验收测试是由客户编写的（由程序员实现）。单元测试是由开发团队负责，例如测试Foo类是否能够正确运行。而验收测试则是由客户负责（客户可能不知道Foo类），他们测试整个系统是否能够正确工作。

验收测试并不非常依赖特定的实现，例如，在重构期间，开发团队可能决定不再需要SubCategory对象，此时SubCategory便无法正常运行。因此他们修改整合测试（如果需要的话）以符合新的系统结构。但是功能测试没有改变，因为用户仍然需要目录搜索功能。

功能测试并不需要总是100%地运行，但应该在软件发布之前进行。功能测试通常验证特定的故事（顾客需求的特性在极限编程中的表述）。同样地，它们也可以在开发周期中追踪进度。每一个运行的测试代表了一种完成的特性。

很遗憾但可以理解的是，没有人已经编写了一个通用的验收测试工具。JUnit只能测试任意的Java类，但验收测试的工具必须是为特定应用程序定制的。对一个以数字为主的程序，验收测试只要确认输入、输出即可。但对于一个资料输入的应用程序，却需要使用带有图形界面的资料输入和输出工具。

我们选择HttpUnit来做功能测试，HttpUnit是用作测试的API，它能够以程序的方式调用Web资源，并且检验回应的结果。它的框架与JUnit结合，可以显示一个HTML网页的内在

结构，让你很容易检查其结构化的元素（show_product.jsp的响应的是一个含有产品价格的表格）。这种方式很符合本书的要求，因为J2EE最重视的就是Web组件。验收测试J2EE中具体的组件可能并不需要特定的框架，因为特定的框架包含低级的逻辑表达。

4. 性能测试：JUnitPerf及JMeter

除了基本的测试功能外，还有许多种测试：如平行测试（parallel test，测试新系统是否与旧系统的执行结果一致）、性能测试（performance test）、验证测试（validation test，对于不正确的输入，系统给出相应的回应）等等。所有这些测试当中，性能测试可能是应用最广泛的。毕竟，即使是功能最强大的系统，如果使用者不愿意使用，也不值一毛钱。客户端的应用程序性能比较差，这是一个问题，服务器端应用程序慢慢吞吞，更是一种危机。J2EE的应用程序一般是在服务器端，每分钟处理成百上千个交易。因此即使只是一小部分的代码没有效率，也可能拖垮整个系统。因此在这样的环境中，运行性能测试的重要性更胜于功能测试（甚至是单元测试）。

我们没有提供有关研究性能的工具（可以解决性能问题的关键工具）。而我们提供的是可以在早期发现性能问题的测试工具。JUnitPerf可以执行单元的性能测试——它可以装饰（decorate）现有的JUnit测试，当执行时间超过期望的时间时，表示这个性能测试失败。这种测试支持重构，它能评估可能存在问题的程序是否达到指定要求。JMeter提供强大的性能测试——当执行远程服务器请求的时候（Web、EJB、数据库等），它可以测量系统的响应时间，并以图形显示出来。使用JMeter，客户可以编写验收测试。例如，在Web服务器中，当150个使用者同时访问时，可以保持3秒钟以内的响应时间。

持续整合：概述

持续整合（continuous integration）是极限编程中另一项实践，受益于使用优秀的工具。基本上，持续整合意味着每天构建数次目前已完成的系统（包括通过所有的测试套件）。只要你持续地进行整合操作，可以说你的系统随时可以走出大门（至少已完成的部分都是没问题的）。这个过程必须实现适当的自动化（一个简单的指令，或许是从一个计时器发出便能构建及测试整个系统），否则一定没有人愿意做这样的事情。有许多合理的理由使我们花费时间和精力来建立持续整合的机制，例如：客户和开发团队可以同时了解实际的进度，减少系统整合的bug并且频繁地运行测试。

可视化进度

从实现持续性整合中得到的某些“微妙”效果将超过它所实际带来的利益。这将肯定团队开发工作的成果。完成构建并且通过测试系统，代表所有开发人员付出努力的结晶。身为系统开发人员，你意识到只要自己提交已完成的代码，自己便会融入开发的流程中。你不会莽撞的朝你自己的方向走，因为你知道你的代码将融合到整个系统中。任何人可以随时看到系统工作的状态，当通过更多的验收测试，并且系统展现出新的功能时，程序员感受到进度朝目标方向前进，进而从最终的成果中得到信心（如果完成的系统毫无生气，也

没通过验收测试，这意味着这个系统存在重大的麻烦，这一般都是因为系统太久没有进行整合所造成的）。

减少整合的痛苦

如果X类并不和Y类一致，尽快知道这个问题是很有意义的。X类和Y类在发生冲突之前的时间越长，当发现X和Y冲突问题的那一天，作者的记忆早已在这个问题上模糊。持续整合能在很短时间内保证矛盾的双方相遇。它不仅仅是立即解决这些问题（这个缺陷一般是由于矛盾的变动导致的），而且它强制开发人员进行有关检查以找出产生问题的原因："Ah，明白了，Sandra已经重构了CapriciousDictator类，因此我就没有理由为OppressiveRegime再增加extraEnforcement方法了。"这样，一个"纯净"的系统诞生了。

经常运行测试

单元测试是完整的持续整合系统的一部分。所有的单元测试百分之百通过之后，并且功能和整合测试运行过后完整的构建任务才完成。这不仅仅是产生系统，同时也证实系统能正确地运行。如果不经常运行测试，测试将失去价值。如果你经常进行构建运行工作并且以同样的频率运行测试，那么将对系统性能提供稳定的升级。

重新启动

我的第一个大型软件项目是比较混乱的……但却进行了持续整合的工作。你可能想知道为什么。我们的团队，由一群没有经验的Java开发人员组成，无法在任何机器上得到一个稳定的构建系统。我们只好在不可能的最后期限的压力下连接一大堆项目的附属文件和库文件。最后我们只能将当前在整合服务器上的所有代码通过一个基于外壳的建造脚本来完成编译，因为没有人有能力编译各自机器上的本地文件。为了看到某些部分在工作，我们将其上传到整合服务器中。这样，我们5到6个程序员同时工作，每次需要五分钟来运行我们的脚本文件（包括重新启动Web服务器）。如果开发人员希望去整合文件就需要在重新启动Web服务器之前得到许可："没有人反对我重起Web服务吧？！"结果是混乱的——但如果有人做了一些改变而导致冲突，我们将瞬间知道。

持续整合和J2EE

基于J2EE的开发模型，在整合和部署阶段，应用程序经过了具有重要意义的自定义过程。对一个完全的J2EE应用程序进行整合和打包需要深入了解不同的档案格式(JAR、WAR和 EAR)，部署描述符和应用程序部署方式，组件嵌入已有容器等相关知识。由于J2EE程序的构建过程十分复杂，我们需要一个自动化构建工具。

Ant辅助支援工作

与其他极限编程实践所不同的是，实现持续整合主要是通过技术手段。一旦拥有了一个

自动化过程，应用它将会是相当简单和极具竞争力的，开发人员在喘息之间就能将任何妖魔鬼怪玩弄股掌之间。你所需要做的仅仅是借助工具自动化进行循环构建工作来开始你的持续整合。因此，本书将涉及Ant，一个脱颖而出的Java构建自动化标准。

Ant允许开发人员编写XML形式的测试构建脚本来调用Java类实现相关工作。此工具是跨平台的（针对基于Java的工具的严格标准）并易于扩展和修改。Ant执行构建工具的所有基本任务：编译、档案的建立、classpath（类路径）的管理等等。它也支持测试和自动生成报告，还支持很多有用的任务，例如可以和源代码控制进行交互，FTP操作和对属性文件的编辑工作。这个预定的构建任务“兵器库”工具允许开发人员方便地将复杂的构建过程自动化。经过Ant的些许工作，就能够从代码控制中取出、构建、定制符合所需环境（例如产品相对整合）的Java应用程序，测试并且通过一个简单命令将Java应用程序部署到远程服务器上。

本书的结构

本书分为三个部分：

第一部分：技术介绍和关键概念。这一部分以极限编程（Extreme Programming）方法学概览开始——这是一个简短的初学者课程或者看作是实践者的复习。本书的该部分概览可以使你明白本书涉及到的实践（自动化测试和持续整合）如何融入到极限编程的逻辑图中。第2章解释了J2EE的构建和部署模型，特别介绍了需要一个自动化构建工具来帮助完成这一过程。在这部分结尾部分，还为大家介绍我们在本书中将构建，测试和重构多次的一个范例应用程序。

第二部分：掌握相关工具。第二部分的每一个章节都为开发人员预备了一个特殊工具的介绍。第4~6章则涵盖了通过Ant来实现持续整合，其余每一章节都介绍一个自动化测试工具。这个指南只用范例代码来举例说明如何运行工具来构造、测试、部署和重构你的代码。在第二部分结尾，你就能综合使用这些工具开展工作了。

第三部分：API（应用程序接口）和标记参考（Tag Reference）。该参考部分深入剖析了本书覆盖范围的API的详细信息。所有的类，变量和需要使用到有关API的方法（或标记及其属性，标记的介绍主要在第12章），这些内容以标准Javadoc风格和相关的代码范例一起呈现给你。

在站点上你能找到的内容

所有的配置脚本、构建脚本、应用程序和本书中涉及的其他源代码将在线提供于http://www.wiley.com/compbooks/hightower。同时还有范例源代码和安装指令勘误表（对于发布紧凑的开源API尤其有必要）和其他后继信息。这个系列的其他书籍的信息你可以在http://www.wiley.com/compbooks获得。

软件工程技术丛书书目

丛书编号	英文书名	中文书名	作　者
1	Object-Oriented and Classical Software Engineering, 5E	面向对象与传统软件工程(原书第5版)	Stephen R. Schach
1	Object-Oriented Software Engineering	面向对象的软件工程	Ian Sommerville
1	Software Engineering: A Practitioner's Approach, 5E	软件工程:实践者的研究方法(原书第5版)	Roger S. Pressman
1	Software Engineering, 6E	软件工程(原书第6版)	Ian Sommerville
1	Software Engineering with Java	软件工程:Java语言实现	Stephen R. Schach
1	Project-Based Software Engineering:An Object-Oriented Approach	基于项目的软件工程:面向对象研究方法	Evelyn Stiller
1.1.1	Software Process Improvement: Practical Guidelines for Business Success	软件过程改进	Sami Zahran
1.1.1	Making Process Improvement Work	软件过程改进简明实践	Neil S. Potter
1.1.2	The Road to the Unified Software Development Process	统一软件开发过程之路	Ivar Jacobson
1.1.2	The Unified Software Development Process	统一软件开发过程	Jacobson/Booch/Rumbaugh
1.1.2	The Rational Unified Process: An Introduction , 2E	Rational统一过程引论(原书第2版)	Philippe Kruchten
1.1.2	UML and The Unified Process: Practical Object-Oriented Analysis & Design	UML和统一过程: 实用面向对象的分析和设计	Jim Arlow
1.1.3	Managing Global Software Projects: How toLead Geographically Distributed Teams,Manage Processes and Use Quality Models	全球化软件项目管理	Gopalaswamy Ramesh
1.1.3	Software Project Management: A Unified Framework	软件项目管理:一个统一的框架	Walker Royce
1.1.3	How to Run Successful Projects III: The Silver Bullet	成功的软件项目管理:银弹方案(原书第3版)	Fergus O'Connell
1.1.3	Successful Software Development, 2E	成功的软件开发(原书第2版)	Scott E. Donaldson
1.1.3	Six Sigma Software Development	6σ软件开发	Christine B. Tayntor
1.1.3	IT Project Management: On Track from Start to Finish	IT项目管理: 从开始到结束的历程	Joseph Phillips
1.1.3	Successful IT Project Delivery: Learning the lessons of Project Failure	IT项目成功交付的秘诀	David Yardley
1.1.3	Software Project Management, 3E	软件项目管理(原书第3版)	Bob Hughes
1.1.3	Architecture-Centric Software Project Management	软件项目管理实用指南: 以体系结构为中心	Daniel J. Paulish
1.1.4	Handbook of Software Quality Assurance, 3E	软件质量保证(原书第3版)	Gordon G. Schulmeyer
1.1.4	Software Reliability Engineering	软件可靠性工程	John Musa
1.1.4	Implementing ISO 9001:2000 The Journey from Conformance to Performance	ISO 9001:2000 实施指南	Tom Tairnubg
1.1.4	CMMI Distilled: A Practical Introduction to Integrated Process Improvement	CMMI精粹:集成化过程改进实用导论	Dennis M. Ahern
1.1.4	CMM Implementation Guide	CMM实施与软件过程改进	Kim Caputo
1.1.4	Implementing the Capability Maturity Model	CMM实施指南	James R.Persse
1.1.4	Object-Oriented Defect Management of Software	面向对象的软件缺限管理	Houman Younessi
1.1.4	Metrics and Models in Software Quality Engineering	软件质量工程:度量与模型	Stephen H. Kan
1.1.4	Performance Solutions: A Practical Guide to Creating Responsive, Scalable Software	软件性能工程	Connie U. Smith