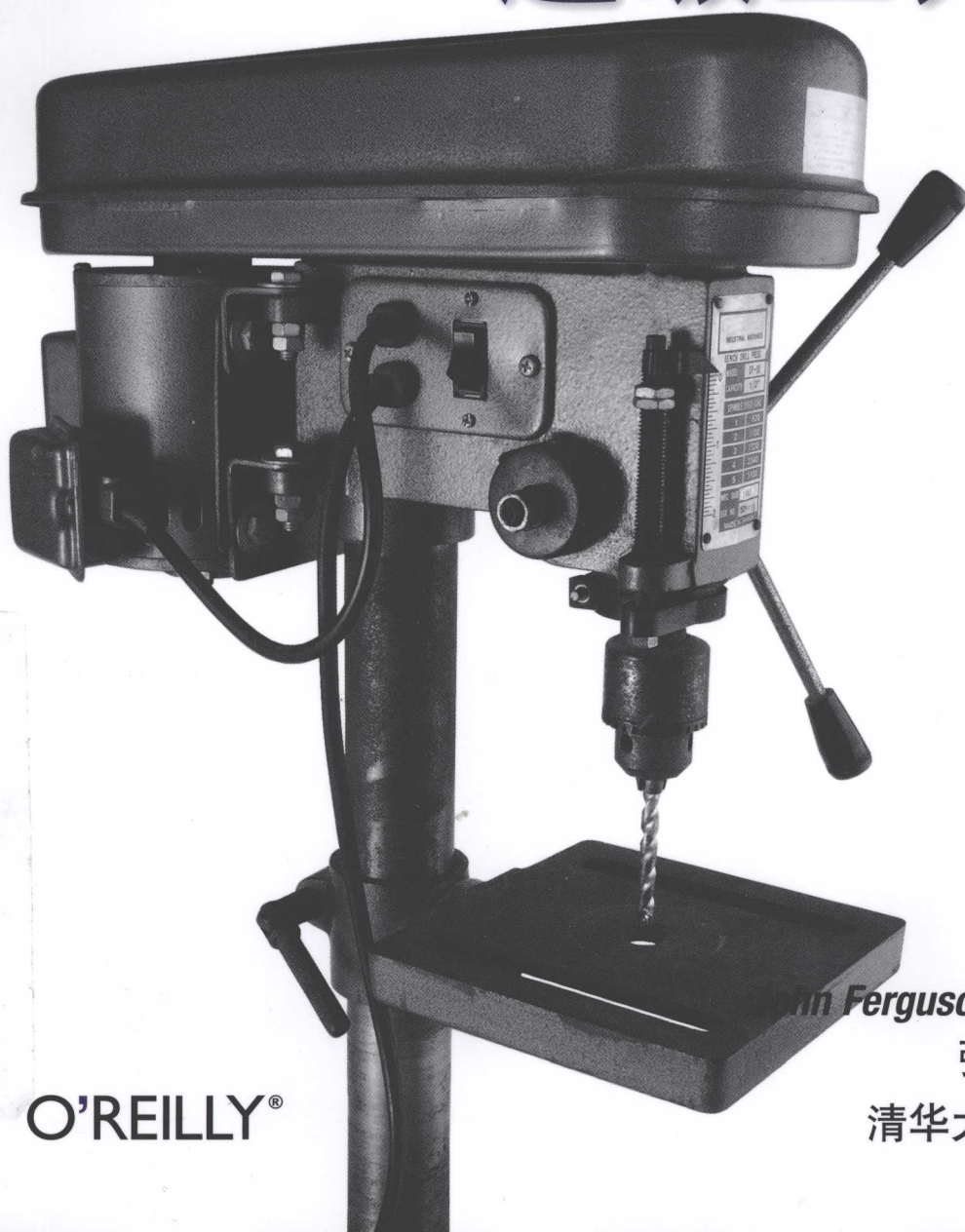


JAVA POWER TOOLS

JAVA开发 超级工具集



O'REILLY®

John Ferguson Smart 著

张文波 等译

清华大学出版社

JAVA 开发 超级工具集

John Ferguson Smart 著

张文波 等译

O'REILLY®

Beijing • Cambridge • Farnham • Köln • Sebastopol • Taipei • Tokyo

O'Reilly Media, Inc. 授权清华大学出版社出版

清华大学出版社

Copyright ©2007 by O'Reilly Media, Inc.

Authorized Simplified Chinese translation edition, by O'Reilly Media, Inc., is published by Tsinghua University Press, 2009. Authorized translation of the original English edition, 2007 O'Reilly Media, Inc., the owner of all rights to publish and sell the same.

All rights reserved including the rights of reproduction in whole or in part in any form.

本书之英文原版由 O'Reilly Media, Inc. 于 2007 年出版。

本中文简体翻译版由 O'Reilly Media, Inc. 授权清华大学出版社于 2009 年出版。此翻译版的出版和销售得到出版权和销售权的所有者——O'Reilly Media, Inc. 的许可。

版权所有, 未经书面许可, 本书的任何部分和全部不得以任何形式复制。

北京市版权局著作权合同登记

图字: 01-2009-5138 号

本书封面贴有清华大学出版社防伪标签, 无标签者不得销售。

版权所有, 侵权必究。侵权举报电话: 010-62782989 13701121933

图书在版编目 (CIP) 数据

Java 开发超级工具集 / (美) 斯马特 (Smart, J. F.) 著; 张文波等译. —北京: 清华大学出版社, 2009. 11

书名原文: Java Power Tools

ISBN 978-7-302-20971-3

I. J… II. ①斯… ②张… III. JAVA 语言—程序设计 IV. TP312

中国版本图书馆 CIP 数据核字 (2009) 第 164525 号

责任编辑: 龙啟铭

封面设计: Mike Kohnke, 张 健

责任校对: 徐俊伟

责任印制: 孟凡玉

出版发行: 清华大学出版社

<http://www.tup.com.cn>

社 总 机: 010-62770175

投稿与读者服务: 010-62776969, c-service@tup.tsinghua.edu.cn

质 量 反 馈: 010-62772015, zhiliang@tup.tsinghua.edu.cn

地 址: 北京清华大学学研大厦 A 座

邮 编: 100084

邮 购: 010-62786544

印 刷 者: 清华大学印刷厂

装 订 者: 三河市新茂装订有限公司

经 销: 全国新华书店

开 本: 178×233 印 张: 49.25 字 数: 1227 千字

版 次: 2009 年 11 月第 1 版 印 次: 2009 年 11 月第 1 次印刷

印 数: 1~3000 册

定 价: 99.00 元 (册)

本书如存在文字不清、漏印、缺页、倒页、脱页等印装质量问题, 请与清华大学出版社出版部联系调换。联系电话: (010)62770177 转 3103 产品编号: 028960-01

O'Reilly Media 公司简介

为了满足读者对网络 and 软件技术知识的迫切需求，世界著名计算机图书出版机构 O'Reilly Media 公司授权清华大学出版社，翻译出版一批该公司久负盛名的英文经典技术专著。

O'Reilly Media 公司是世界上在 UNIX、X、Internet 和其他开放系统图书领域具有领导地位的出版公司，同时也是联机出版的先锋。

从最畅销的 *The Whole Internet User's Guide & Catalog* (纽约公共图书馆评为 20 世纪最重要的 50 本书之一) 到 GNN (最早的 Internet 门户和商业网站)，再到 WebSite (第一个桌面 PC 的 Web 服务器软件)，O'Reilly Media 公司一直处于 Internet 发展的最前沿。

许多书店的反馈表明，O'Reilly Media 公司是最稳定的计算机图书出版商——每本书都一版再版。与大多数计算机图书出版商相比，O'Reilly Media 公司具有深厚的计算机专业背景，这使其形成了非常不同于其他出版商的出版方针。O'Reilly Media 公司所有的编辑人员以前都是程序员，或者是顶尖级的技术专家。它还有许多固定的作者群体——他们本身是相关领域的技术专家、咨询专家，而现在编写著作，O'Reilly Media 公司依靠他们及时地推出图书。因为 O'Reilly Media 公司紧密地与计算机业界联系着，所以它知道市场的真正需要。

译者序

有人说，人与动物的根本区别就是会制造和使用工具。亚里士多德有这样一句名言，“给我一个支点，我就可以翘动地球”。理论上这是可行的。当然，仅仅有支点是远远不够的，还需要使用杠杆，杠杆就是工具，由此可以看到，工具的重要作用。其实，工具无处不在。我们吃饭，需要工具（筷子，碗等）；我们工作，需要工具（电脑，鼠标等）；我们出行，需要工具（自行车，公交车等），就是简单的钉几页纸，也需要钉书器或曲别针之类的小工具，画一条直线、画一个圆，会需要板尺和圆规……可以说，工具在生活中随时随地为我们服务，提供帮助。当然，没有这些工具，人们也可以生活或工作，比如，画直线，画圆，徒手完全可以，但总不会有使用板尺画的线条那么笔直，也不会有使用圆规画的圆圈那么圆；出行走路也可以，但总不如坐车的速度快。

软件工具的作用也是如此，从构建程序到调试，从技术文档的编写到单元测试以及集成、负载和性能测试，从质量度量到问题管理，从版本控制到持续集成，有关Java软件开发的各个方面都有相当多的辅助工具，可以提供帮助，让开发人员提高效率。

简单举个例子，设计良好的项目，需要良好的文档来支撑，向最终用户解释如何使用项目。虽然有人认为，源代码本身提供了足够的技术文档。但只查看源代码将很难理解应用程序，代码行不能很好地说明业务逻辑和设计决定。有一些工具可以帮助用户完善、丰富文档流程并将其自动化，比如Javadoc、SchemaSpy和Doxygen，可生成美观、准确、便于使用的最新在线文档，包括图形化的数据库模型和UML图，这类自动生成的文档足以帮助新开发人员理解产品架构。通过将文档生成流程完全集成到软件开发生命周期中，可以保证整个开发团队永远有最新技术文档可用。

本书讲解了大量软件开发工具，都是作者从浩如烟海的大量工具中精挑细选出来的，也许你正在烦恼苦思冥想的难题，使用本书介绍的一个看起来不起眼的小工具就可以迎刃而解。本书最主要的作用是，书中介绍的小工具，可以帮助开发人员提高效率，更出色地完成任务。

本书主要由张文波翻译，参加翻译的还有韦笑、王雷、李志云、李晓春、陈安华、孙宏、赵成璧、侯佳宜、许伟、戴文雅、于樊鹏、刘朋、王嘉佳、李腾、邓卫、邓凡平、陈磊、李建锋、刘延军、魏宇、赵远锋、樊旭平、程烨尔、唐玮、周京平、徐冬、冯哲、李绯、李强、赵东辉、吕巧珍、宋雁、何晓刚、马丽娟、段涛、吴江华、孙燕、周刚、张乐华、高强、王红亮、周峰、谢晖、李琳、孙向阳、李波、程云建、许晓哲、朱珂、李元园、曹锋、赵志鹏、冯佳、黄志雄、李健、林彩娥、孙蕾、杨金奎、张百涛、李展、张文波、赵楠、周文培、连祥宇、徐增辉、邢博特、刘子瑛、刘欣、黄虹、侯文茹、董云峰、胡平、隋杨等人。最后，祝广大读者从本书中挖掘出更多的宝藏。

译者

目录

序	1
前言	3
导言	15

第一部分 构建工具

第 1 章 用 Ant 设置项目	23
1.1 构建过程中的 Ant	23
1.2 安装 Ant	23
1.3 Ant 概述	25
1.4 在 Ant 中编译 Java 代码	32
1.5 使用属性自定义构建脚本	34
1.6 在 Ant 中运行单元测试	37
1.7 用 Javadoc 生成文档	52
1.8 应用程序打包	53
1.9 部署应用程序	57
1.10 引导构建脚本	59
1.11 用 Maven 任务在 Ant 中使用 Maven 依赖	60

1.12	在 Eclipse 中使用 Ant.....	64
1.13	在 NetBeans 中使用 Ant.....	64
1.14	用 XMLTask 操作 XML	65
1.15	小结	70
第 2 章 用 Maven 2 设置项目		71
2.1	Maven 和开发构建过程	71
2.2	Maven 和 Ant	72
2.3	安装 Maven	72
2.4	声明式构建和 Maven 项目对象模型	74
2.5	理解 Maven 2 的生命周期	85
2.6	Maven 目录结构	86
2.7	根据环境配置 Maven	87
2.8	Maven 2 中的依赖管理	89
2.9	用 MvnRepository 查找依赖	97
2.10	项目继承和聚合	98
2.11	使用原型 (Archetype) 创建项目模板	101
2.12	编译代码	104
2.13	测试代码	105
2.14	打包和部署应用程序	108
2.15	使用 Cargo 部署应用程序	110
2.16	在 Eclipse 中使用 Maven	113
2.17	在 NetBeans 中使用 Maven	115
2.18	使用插件定制构建过程	115
2.19	用 Archiva 设置企业仓库	123
2.20	使用 Artifactory 设置企业仓库	135
2.21	在 Maven 中使用 Ant.....	145
2.22	高级原型	150
2.23	使用组件	154

第二部分 版本控制工具

第 3 章 用 CVS 设置版本控制	163
3.1 CVS 概述	163
3.2 设置 CVS 仓库	163
3.3 在 CVS 中创建新项目	164
3.4 检出项目	166
3.5 处理文件 —— 更新和提交	167
3.6 解决仓库锁定问题	170
3.7 使用关键字替换	171
3.8 处理二进制文件	172
3.9 CVS 标记	173
3.10 在 CVS 中创建分支	174
3.11 从分支中合并更改	176
3.12 查看更改历史	176
3.13 还原更改	178
3.14 在 Windows 中使用 CVS	180
 第 4 章 用 Subversion 设置版本控制	 181
4.1 Subversion 概述	181
4.2 安装 Subversion	184
4.3 Subversion 仓库类型	185
4.4 设置 Subversion 仓库	186
4.5 设置新的 Subversion 项目	188
4.6 检出工作副本	189
4.7 将现有文件导入到 Subversion	191
4.8 理解 Subversion 仓库的 URL	192
4.9 使用文件	193
4.10 查看当前状态: Status 命令	197
4.11 解决冲突	199
4.12 使用标记、分支和合并	200
4.13 回滚到以前的修订版本	204
4.14 对二进制文件使用文件锁定	205
4.15 打破和窃取锁定	206
4.16 用 svn:needs-lock 属性设置锁定文件为只读	208

4.17	使用属性	208
4.18	Subversion 中的更改历史记录: 日志和 Blame 命令	211
4.19	用 svnserve 设置 Subversion 服务器	212
4.20	设置安全的 svnserve 服务器	215
4.21	设置支持 WebDAV/DeltaV 的 Subversion 服务器	216
4.22	设置安全的 WebDAV/DeltaV 服务器	221
4.23	用钩子脚本定制 Subversion	221
4.24	将 Subversion 安装为 Windows 服务	223
4.25	备份和还原 Subversion 仓库	224
4.26	在 Eclipse 中使用 Subversion	225
4.27	在 NetBeans 中使用 Subversion	233
4.28	在 Windows 中使用 Subversion	239
4.29	缺陷跟踪和变更控制	245
4.30	在 Ant 中使用 Subversion	246
4.31	小结	249

第三部分 持续集成

第 5 章	用 Continuum 设置持续集成服务器	255
5.1	Continuum 概述	255
5.2	安装 Continuum 服务器	255
5.3	手工启动和停止服务器	258
5.4	检查服务器状态	260
5.5	以 Verbose 模式运行 Continuum 服务器	260
5.6	添加项目组	260
5.7	添加 Maven 项目	260
5.8	添加 Ant 项目	263
5.9	添加外壳脚本项目	263
5.10	管理项目构建	264
5.11	管理用户	266
5.12	设置通知方法	267
5.13	配置和制订构建计划	267
5.14	调试构建	269

5.15	配置 Continuum 邮件服务器	270
5.16	配置 Continuum 网站端口	271
5.17	用 Continuum 自动生成 Maven 网站	272
5.18	配置手工构建任务	273
5.19	小结	275
第 6 章 用 CruiseControl 设置持续集成服务器		276
6.1	CruiseControl 概述	276
6.2	安装 CruiseControl	277
6.3	配置 Ant 项目	278
6.4	用 Publisher（发布器）通知开发人员	283
6.5	在 CruiseControl 中设置 Maven 2 项目	289
6.6	CruiseControl 操作面板	290
6.7	第三方工具	290
6.8	小结	292
第 7 章 LuntBuild ——基于 Web 的持续集成服务器		294
7.1	LuntBuild 概述	294
7.2	安装 LuntBuild	294
7.3	配置 LuntBuild 服务器	295
7.4	添加项目	298
7.5	为版本编号使用项目变量	304
7.6	构建结果诊断	305
7.7	与 Eclipse 一起使用 LuntBuild	308
7.8	在 Luntbuild 中使用 Cobertura 报告测试覆盖	309
7.9	将 Luntbuild 与 Maven 集成	317
7.10	小结	321
第 8 章 用 Hudson 持续集成		323
8.1	Hudson 概述	323
8.2	安装 Hudson	323
8.3	管理 Hudson 的主目录	324
8.4	安装升级	325
8.5	配置 Hudson	325

8.6	添加新构建任务	327
8.7	组织作业	332
8.8	监控构建	333
8.9	查看和提升特定构建	334
8.10	管理用户	334
8.11	认证与安全	336
8.12	查看更改	337
8.13	Hudson 插件	338
8.14	记录测试结果	338
8.15	记录代码度量	339
8.16	报告代码覆盖	340
第 9 章 用 Openfire 设置即时消息平台		343
9.1	开发项目中的即时消息软件	343
9.2	安装 Openfire	343
9.3	在 Openfire 上设置用户账户	344
9.4	认证外部数据库中的用户	345
9.5	针对 POP3 服务器认证用户	347
9.6	用群聊天召开虚拟团队会议	347
9.7	用 Openfire 插件扩展功能	348
9.8	与 Continuum 一起使用 Openfire	348
9.9	与 CruiseControl 一起使用 Openfire	350
9.10	与 Luntbuild 一起使用 Openfire	351
9.11	使用 Smack API 从 Java 应用程序中发送 Jabber 消息	351
9.12	用 Smack API 检测用户是否在线	353
9.13	使用 Smack API 接收消息	353

第四部分 单元测试

第 10 章 用 JUnit 测试代码		357
10.1	JUnit 3.8 和 JUnit 4	357
10.2	用 JUnit 4 进行单元测试	358
10.3	设置和优化单元测试用例	359

10.4	用 Timeout 进行简单性能测试	361
10.5	轻松检查异常	361
10.6	使用带参数的测试	362
10.7	使用 assertThat 和 Hamcrest 库	364
10.8	JUnit 4 的理论机制	366
10.9	与 Maven 2 一起使用 JUnit 4	368
10.10	与 Ant 一起使用 JUnit 4	369
10.11	在 Ant 中有选择地运行 JUnit 4 测试	371
10.12	集成测试	373
10.13	在 Eclipse 中使用 JUnit 4	374
第 11 章	用 TestNG 进行下一代测试	377
11.1	TestNG 概述	377
11.2	用 TestNG 创建简单单元测试	377
11.3	定义 TestNG 测试套件	379
11.4	Eclipse 的 TestNG 插件	380
11.5	在 Ant 中使用 TestNG	382
11.6	与 Maven 2 一起使用 TestNG	386
11.7	管理测试生命周期	387
11.8	使用测试组	391
11.9	管理依赖	393
11.10	并行测试	395
11.11	测试参数和数据驱动的测试	396
11.12	检查异常	397
11.13	处理部分失败	397
11.14	重新运行失败的测试	398
第 12 章	用 Cobertura 最大化测试覆盖	399
12.1	测试覆盖	399
12.2	从 Ant 中运行 Cobertura	400
12.3	检查 TestNG 测试的代码覆盖	402
12.4	理解 Cobertura 报告	404
12.5	实施高水平代码覆盖	406
12.6	在 Maven 中生成 Cobertura 报告	408

12.7	将覆盖测试集成到 Maven 构建过程中	409
12.8	Eclipse 中的代码覆盖	411
12.9	小结	413

第五部分 集成、功能、负载和性能测试

第 13 章 用 StrutsTestCase 测试 Struts 应用程序 419

13.1	概述	419
13.2	测试 Struts 应用程序	419
13.3	StrutsTestCase 概述	420
13.4	使用 StrutsTestCase 进行模拟测试	421
13.5	测试 Struts 错误处理	425
13.6	定制测试环境	425
13.7	一级性能测试	426
13.8	小结	426

第 14 章 用 DbUnit 进行数据库集成测试 427

14.1	引言	427
14.2	概述	427
14.3	DbUnit 的结构	429
14.4	示例应用程序	433
14.5	准备数据库	434
14.6	验证数据库	440
14.7	替换值	445
14.8	其他数据集格式	450
14.9	处理自定义数据类型	453
14.10	其他应用	457

第 15 章 用 JUnitPerf 进行性能测试 465

15.1	JUnitPerf 概述	465
15.2	用 TimedTest 测量性能	465
15.3	用 LoadTest 模拟负载	468
15.4	对非线程安全的测试进行负载测试	469

15.5	在 Ant 中分离性能测试和单元测试	470
15.6	在 Maven 中分离性能测试和单元测试	471

第 16 章 用 JMeter 进行负载和性能测试 472

16.1	概述	472
16.2	安装 JMeter	472
16.3	测试简单的 web 应用程序	473
16.4	组织测试用例	478
16.5	记录和显示测试结果	481
16.6	使用 JMeter 代理服务器记录测试用例	484
16.7	使用变量进行测试	486
16.8	在多台计算机上进行测试	488

第 17 章 用 SoapUI 测试 Web 服务 490

17.0	概述	490
17.1	SoapUI 概述	490
17.2	安装 SoapUI	492
17.3	安装本地 web 服务	492
17.4	用 SoapUI 测试 web 服务	494
17.5	用 SoapUI 进行负载测试	500
17.6	从命令行运行 SoapUI	502
17.7	从 Ant 中运行 SoapUI	505
17.8	从 Maven 中运行 SoapUI	506
17.9	持续测试	506
17.10	小结	508

第 18 章 用 Sun JDK 工具监视和

分析 Java 应用程序的性能 509

18.1	Sun JDK 性能分析和监视工具	509
18.2	用 jConsole 连接并监视 Java 应用程序	509
18.3	用 jConsole 监视远程 Tomcat 应用程序	511
18.4	用 JDK 工具检测和识别内存泄露	513
18.5	用堆转储、jmap 和 jhat 诊断内存泄露	518

第 19 章 在 Eclipse 中分析 Java 应用程序的性能	523
19.1 在集成开发环境中分析应用程序性能	523
19.2 Eclipse 测试和性能工具平台	523
19.3 安装 TPTP	524
19.4 TPTP 和 Java 6	525
19.5 使用 TPTP 进行基本性能分析	526
19.6 用基本内存分析结果分析内存使用	530
19.7 分析执行时间	532
19.8 显示代码覆盖统计	533
19.9 使用过滤器优化结果	533
19.10 分析 web 应用程序的性能	536
19.11 小结	537
第 20 章 测试用户界面	538
20.1 概述	538
20.2 用 Selenium 测试 web 应用程序	538
20.3 用 FEST 测试 Swing 图形用户界面	564
20.4 小结	572

第六部分 质量度量工具

第 21 章 用 Checkstyle 检测和实施编码标准	577
21.1 用 Checkstyle 实施编码标准	577
21.2 在 Eclipse 中使用 Checkstyle	579
21.3 在 Eclipse 中定制 Checkstyle 规则	582
21.4 使用 XML 配置文件定制 Checkstyle 规则	584
21.5 定制 Checkstyle: 可舍弃和使用的常见规则	586
21.6 用 Checkstyle 定义源代码文件头规则	589
21.7 禁用 Checkstyle 测试	590
21.8 与 Ant 一起使用 Checkstyle	590
21.9 与 Maven 一起使用 Checkstyle	591

第 22 章 用 PMD 预先检测错误	594
22.1 PMD 和静态代码分析	594
22.2 在 Eclipse 中使用 PMD	594
22.3 在 Eclipse 中配置 PMD 规则	596
22.4 PMD 规则集	597
22.5 编写 PMD 规则集	600
22.6 在 Eclipse 中生成 PMD 报告	602
22.7 禁用 PMD 规则	602
22.8 用 CPD 检测剪切和粘贴	603
22.9 在 Ant 中使用 PMD	604
22.10 在 Maven 中使用 PMD	606
 第 23 章 用 FindBugs 预先检测错误	 609
23.1 FindBugs: 专业的程序错误检测工具	609
23.2 在 Eclipse 中使用 FindBugs	610
23.3 用 FindBugs 过滤器有选择地禁用规则	612
23.4 使用 FindBugs 注释	613
23.5 在 Ant 中使用 FindBugs	615
23.6 在 Maven 中使用 FindBugs	617
23.7 小结	619
 第 24 章 检查结果——用 Jupiter 进行半自动化代码评审 ...	 620
24.1 Jupiter 概述——用于 Eclipse 的代码评审工具	620
24.2 在 Eclipse 中安装 Jupiter	621
24.3 理解 Jupiter 的代码评审流程	622
24.4 个人代码审查	622
24.5 配置	623
24.6 设置默认配置值	627
24.7 单独评审	628
24.8 团队评审	631
24.9 返工阶段	633
24.10 Jupiter 的后台处理	635
24.11 小结	635

第 25 章 用 Mylyn 突出工作重点	637
25.1 Mylyn 概述	637
25.2 安装 Mylyn	638
25.3 跟踪任务和问题	638
25.4 与任务仓库交互	641
25.5 用上下文管理将工作重点集中在任务上	645
25.6 使用 Eclipse 更改集	647
25.7 与其他开发人员共享上下文	649
25.8 小结	650
第 26 章 监视构建统计信息	651
26.1 概述	651
26.2 QALab	651
26.3 用 StatSCM 度量源代码管理	658
26.4 在 Ant 中用 StatSVN 提供统计信息	659

第七部分 问题管理工具

第 27 章 Bugzilla	665
27.1 Bugzilla 概述	665
27.2 安装 Bugzilla	665
27.3 设置 Bugzilla 环境	669
27.4 管理用户账户	670
27.5 使用用户组限制访问	672
27.6 配置产品	673
27.7 用里程碑跟踪进度	675
27.8 用分类管理产品组	675
27.9 搜索程序错误	676
27.10 创建新程序错误	678
27.11 Bugzilla 程序错误的生命周期	678
27.12 安排通知 (Whining)	681
27.13 在 Bugzilla 中定制字段	682
27.14 小结	683