

<<JUnit实战>>

图书基本信息

书名：<<JUnit实战>>

13位ISBN编号：9787115274755

10位ISBN编号：7115274754

出版时间：2012-4

出版时间：人民邮电出版社

作者：塔凯文

页数：442

译者：王魁

版权说明：本站所提供下载的PDF图书仅提供预览和简介，请支持正版图书。

更多资源请访问：<http://www.tushu007.com>

<<JUnit实战>>

内容概要

本书从认识JUnit、不同的测试策略、JUnit与构建过程、JUnit扩展4个方面,由浅入深、由易到难地对JUnit展开了系统的讲解,包括探索JUnit的核心、软件测试原则、测试覆盖率与开发、使用stub进行粗粒度测试、使用mock objects进行测试、容器内测试、从Ant中运行JUnit测试、从Maven2中运行JUnit测试、持续集成工具、表示层的测试、Ajax测试、使用Cactus进行服务器端的Java测试、测试JSF应用程序、测试OSGi组件、测试数据库访问、测试基于JPA的应用程序、JUnit的其他用法等内容。

本书不仅仅介绍了使用JUnit框架测试项目必须掌握的核心概念,还指导读者如何使用JUnit框架编写自己的测试用例,并针对在编写代码的过程中如何使得代码可测试给出建议。本书还介绍了基础的软件开发原则,如测试驱动开发(TDD)等,以便指导用户如何使用不同的工具来测试典型Java EE应用程序的每一层。此外,本书也提供了几个附录,以帮助读者快速转换到最新版本的JUnit,并且能够轻松地集成自己喜欢的IDE。

本书适合于已具有一定Java编程基础的读者,以及在Java平台下进行各类软件开发的开发人员、测试人员。对于单元测试学习者和编程爱好者来说,本书则具有极高的学习参考价值。

<<JUnit实战>>

作者简介

作者：(美国)塔凯文(Tahchiev.P.) 译者：王魁

<<JUnit实战>>

书籍目录

第1部分 认识JUnit

第1章 JUnit起步

- 1.1 证实它能运行
- 1.2 从零开始
- 1.3 理解单元测试框架
- 1.4 JUnit的设计目标
- 1.5 安装JUnit
- 1.6 使用JUnit测试
- 1.7 小结

第2章 探索JUnit的核心

- 2.1 探索JUnit核心
- 2.2 运行参数化测试
- 2.3 JUnit的测试运行器
 - 2.3.1 测试运行器简介
 - 2.3.2 JUnitCore fa ?
ade
 - 2.3.3 自定义测试运行器
- 2.4 用Suite来组合测试
 - 2.4.1 组合一组测试类
 - 2.4.2 组合一组测试集
 - 2.4.3 Suite、IDE、Ant与Maven
- 2.5 小结

第3章 掌握JUnit

- 3.1 引入controller组件
 - 3.1.1 设计接口
 - 3.1.2 实现基类
- 3.2 让我们来测试它
 - 3.2.1 测试DefaultController
 - 3.2.2 添加一个处理器
 - 3.2.3 请求处理
 - 3.2.4 改进testProcessRequest
- 3.3 测试异常处理
 - 3.3.1 模拟异常条件
 - 3.3.2 测试异常
- 3.4 超时测试
- 3.5 引入Hamcrest匹配器
- 3.6 创建测试项目
- 3.7 小结

第4章 软件测试原则

- 4.1 单元测试的必要性
 - 4.1.1 带来更高的测试覆盖率
 - 4.1.2 提高团队效率
 - 4.1.3 监测衰退和减少调试
 - 4.1.4 自信地重构
 - 4.1.5 改进实现

<<JUnit实战>>

- 4.1.6 将预期的行为文档化
- 4.1.7 启用代码覆盖率以及其他指标
- 4.2 测试类型
 - 4.2.1 软件测试的4种类型
 - 4.2.2 单元测试的3种类型
- 4.3 黑盒测试与白盒测试
- 4.4 小结
- 第2部分 不同的测试策略
- 第5章 测试覆盖率与开发
 - 5.1 测量测试覆盖率
 - 5.1.1 测试覆盖率简介
 - 5.1.2 Cobertura简介
 - 5.1.3 生成测试覆盖率报告
 - 5.1.4 结合黑盒与白盒测试
 - 5.2 编写可测试的代码
 - 5.2.1 公共API是协议
 - 5.2.2 减少依赖关系
 - 5.2.3 创建简单的构造函数
 - 5.2.4 遵循最少知识原则
 - 5.2.5 避免隐藏的依赖关系与全局状态
 - 5.2.6 单态模式的优点和缺点
 - 5.2.7 优先使用通用方法
 - 5.2.8 组合优先于继承
 - 5.2.9 多态优先于条件语句
 - 5.3 测试驱动开发
 - 5.3.1 调整开发周期
 - 5.3.2 TDD的两个步骤
 - 5.4 在开发周期中的测试
 - 5.5 小结
- 第6章 使用stub进行粗粒度测试
 - 6.1 stub简介
 - 6.2 使用stub测试一个HTTP连接
 - 6.2.1 选择使用stub的方案
 - 6.2.2 使用Jetty作为嵌入式服务器
 - 6.3 使用stub替换Web服务器资源
 - 6.3.1 建立第一个stub测试
 - 6.3.2 针对故障情况进行测试
 - 6.3.3 回顾第一个stub测试
 - 6.4 替换连接
 - 6.4.1 创建自定义的URL协议处理器
 - 6.4.2 创建一个JDK的HttpURLConnection stub
 - 6.4.3 运行测试
 - 6.5 小结
- 第7章 使用mock objects进行测试
 - 7.1 mock objects简介
 - 7.2 使用mock objects进行单元测试
 - 7.3 使用mock objects来重构

<<JUnit实战>>

- 7.3.1 重构示例
- 7.4 替换一个HTTP连接
 - 7.4.1 定义mock objects
 - 7.4.2 测试一个简单的方法
 - 7.4.3 第一次尝试：简单的方法重构技巧
 - 7.4.4 第二个尝试：使用类工厂来重构
- 7.5 把mocks用作特洛伊木马
- 7.6 介绍mock框架
 - 7.6.1 使用EasyMock
 - 7.6.2 使用JMock
- 7.7 小结
- 第8章 容器内测试
 - 8.1 标准单元测试的局限性
 - 8.2 mock objects解决方案
 - 8.3 容器内测试
 - 8.3.1 实现策略
 - 8.3.2 容器内测试框架
 - 8.4 比较stub、mock objects和容器内测试
 - 8.4.1 stub的优点与缺点
 - 8.4.2 mock objects的优点和缺点
 - 8.4.3 容器内测试的优点与缺点
 - 8.4.4 容器内测试与容器外测试
 - 8.5 小结
- 第3部分 JUnit与构建过程
- 第9章 从Ant中运行JUnit测试
 - 9.1 生命中的一天
 - 9.2 从Ant中运行测试
 - 9.3 认识并安装Ant
 - 9.4 Ant的目标、项目、属性以及任务
 - 9.4.1 javac任务
 - 9.4.2 JUnit任务
 - 9.5 让Ant来执行任务
 - 9.6 使用Ivy进行依赖管理
 - 9.7 创建HTML报告
 - 9.8 批处理测试
 - 9.9 小结
- 第10章 从Maven2中运行JUnit测试
 - 10.1 Maven的功能
 - 10.1.1 约定优于配置
 - 10.1.2 强大的依赖管理
 - 10.1.3 Maven的构建生命周期
 - 10.1.4 基于插件的架构
 - 10.1.5 Maven项目对象模型
 - 10.2 建立一个Maven项目
 - 10.3 Maven插件的引入
 - 10.3.1 Maven的Compiler插件
 - 10.3.2 Maven的Surefire插件

<<JUnit实战>>

10.3.3 使用Maven生成HTML格式的JUnit报告

10.4 Maven的不足

10.5 小结

第11章 持续集成工具

11.1 尝试持续集成

11.1.1 持续集成测试

11.2 拯救用户的CruiseControl

11.2.1 开始使用CruiseControl

11.2.2 创建一个示例项目

11.2.3 解析CruiseControl配置文件

11.3 另一个持续集成工具--Hudson

11.3.1 Hudson简介

11.3.2 安装

11.3.3 配置Hudson

11.3.4 配置Hudson中的项目

11.4 持续集成的优势

11.5 小结

第4部分 JUnit扩展

第12章 表示层的测试

12.1 选择测试框架

12.2 HtmlUnit简介

12.2.1 一个实例

12.3 编写HtmlUnit测试

12.3.1 HTML断言

12.3.2 对特定的Web浏览器进行测试

12.3.3 测试多个Web浏览器

12.3.4 创建独立的测试

12.3.5 导航对象模型

12.3.6 通过特定的元素类型访问元素

12.3.7 通过名字与索引访问元素

12.3.8 通过引用访问元素

12.3.9 使用XPath

12.3.10 测试失败和异常

12.3.11 应用程序与网络导航

12.3.12 使用HtmlUnit测试表单

12.3.13 测试框架(frame)

12.3.14 测试JavaScript

12.3.15 测试CSS

12.3.16 SSL错误

12.4 集成HtmlUnit和Cactus

12.4.1 在Cactus中编写测试

12.5 Selenium简介

12.6 生成Selenium测试

12.6.1 一个实例

12.7 运行Selenium测试

12.7.1 管理Selenium服务器

12.7.2 使用JUnit 4运行Selenium测试

<<JUnit实战>>

- 12.8 编写Selenium测试
 - 12.8.1 针对特定的Web浏览器进行测试
 - 12.8.2 测试多个浏览器
 - 12.8.3 应用程序和网络导航
 - 12.8.4 通过引用访问元素
 - 12.8.5 通过异常使测试失败
 - 12.8.6 使用Selenium测试表单
 - 12.8.7 测试JavaScript告警
 - 12.8.8 捕获一个JUnit 3测试失败的截屏
 - 12.8.9 捕获一个JUnit 4测试失败的截屏
- 12.9 HtmlUnit与Selenium
- 12.10 小结
- 第13章 Ajax测试
 - 13.1 Ajax应用程序难以测试的原因
 - 13.1.1 传统的Web交互
 - 13.1.2 Ajax交互
 - 13.1.3 一个崭新的世界
 - 13.1.4 测试的挑战
 - 13.2 Ajax的测试模式
 - 13.2.1 功能测试
 - 13.2.2 客户端脚本单元测试
 - 13.2.3 服务测试
 - 13.3 功能测试
 - 13.3.1 使用Seleniun进行功能测试
 - 13.3.2 使用HtmlUnit进行功能测试
 - 13.4 JavaScript测试
 - 13.4.1 使用RhinoUnit测试JavaScript
 - 13.4.2 使用JsUnit测试JavaScript
 - 13.4.3 编写JsUnit测试用例
 - 13.4.4 编写JsUnit测试集
 - 13.4.5 手动运行JsUnit测试用例
 - 13.4.6 使用Ant自动运行JsUnit测试用例
 - 13.5 RhinoUnit与JsUnit
 - 13.6 使用JSLint检验最佳实践
 - 13.7 使用HttpClient测试服务
 - 13.7.1 调用XML服务
 - 13.7.2 验证XML响应
 - 13.7.3 验证JSON响应
 - 13.8 测试Google Web工具箱应用程序
 - 13.8.1 为GWT应用程序选择测试框架
 - 13.8.2 手动创建GWTTTestCase
 - 13.8.3 使用junitCreator创建GWTTTestCase
 - 13.8.4 运行测试用例
 - 13.8.5 安装和拆卸测试
 - 13.8.6 创建测试集
 - 13.8.7 运行测试集
 - 13.9 小结

<<JUnit实战>>

第14章 使用Cactus进行服务器端的Java测试

14.1 什么是Cactus ?

14.2 使用Cactus进行测试

14.2.1 你可以使用Cactus测试的Java组件

14.2.2 一般原则

14.2.3 Cactus如何工作

14.3 测试servlet和filters

14.3.1 介绍管理应用程序

14.3.2 使用Cactus编写servlet测试

14.4 测试JSP

14.4.1 回顾管理应用程序

14.4.2 什么是JSP单元测试 ?

14.4.3 单独使用Cactus对JSP进行单元测试

14.4.4 利用SQL结果数据执行JSP

14.5 测试EJB

14.6 什么是Cargo ?

14.7 使用Ant执行Cactus测试

14.7.1 用来准备文件的Cactus

14.8 使用Maven2x执行Cactus测试

14.8.1 Maven2 cactifywar MOJO

14.8.2 Maven2 cactifyear MOJO

14.9 从浏览器执行Cactus测试

14.10 小结

第15章 测试JSF应用程序

15.1 引入JSF

15.2 介绍示例应用程序

15.3 测试JSF应用程序时的典型问题

15.4 测试JSF应用程序的策略

15.4.1 黑盒方法

15.4.2 Mock object援救

15.5 使用JSUnit测试示例应用程序

15.5.1 从浏览器执行一个JSFUnit测试

15.5.2 使用JSFUnit测试Ajax

15.6 使用HtmlUnit与JSFUnit

15.7 JSF应用程序的性能测试

15.8 小结

第16章 测试OSGi组件

16.1 OSGi简介

16.2 第一个OSGi服务

16.2.1 示例应用程序

16.3 测试OSGi服务

16.3.1 Mock objects

16.4 引入JUnit4OSGi

16.5 小结

<<JUnit实战>>

第17章 测试数据库访问

- 17.1 数据库单元测试的阻抗不匹配
 - 17.1.1 单元测试必须执行隔离的代码
 - 17.1.2 单元测试必须易于编写和运行
 - 17.1.3 单元测试必须运行快速
- 17.2 DbUnit介绍
 - 17.2.1 示例应用程序
 - 17.2.2 设置DbUnit并运行示例应用程序
- 17.3 使用数据集来填充数据库
 - 17.3.1 剖析DatabaseOperation
- 17.4 用数据集断言数据库状态
 - 17.4.1 过滤数据集
 - 17.4.2 忽略数据列
- 17.5 使用ReplacementDataSet转换数据
 - 17.5.1 使用ReplacementDataSet处理不同的ID问题
 - 17.5.2 处理NULL值
- 17.6 从已有的数据库数据中创建数据集
- 17.7 高级技术
 - 17.7.1 DbUnit与模板设计模式
 - 17.7.2 通过自定义注释提高重用
 - 17.7.3 在数据集中使用表达式语言
- 17.8 数据库访问测试的最佳做法
 - 17.8.1 每个开发者使用一个数据库
 - 17.8.2 确保目标数据库被测试
 - 17.8.3 为加载和存储数据创建互补测试
 - 17.8.4 编写加载测试用例时, 应涵盖所有基本场景
 - 17.8.5 计划数据集的使用
 - 17.8.6 测试清理
- 17.9 小结

第18章 测试基于JPA的应用程序

- 18.1 测试多层应用程序
 - 18.1.1 示例应用程序
 - 18.1.2 多层、多种测试策略
- 18.2 JPA测试的方方面面
- 18.3 准备基础设施
- 18.4 测试JPA实体映射
 - 18.4.1 使用JPA ID生成器集成测试用例
- 18.5 测试基于JPA的DAO
- 18.6 测试外键名字
- 18.7 小结

第19章 JUnit的其他用法

- 19.1 介绍
 - 19.1.1 工具概述
 - 19.1.2 运行示例
- 19.2 透明地使用mock
 - 19.2.1 Unitils的EasyMock支持
 - 19.2.2 FEST-Mocks

<<JUnit实战>>

- 19.2.3 Mycila
- 19.3 DbUnit集成
- 19.4 使断言更简单
 - 19.4.1 JUnit-addons断言包
 - 19.4.2 Unitlis的ReflectionAssert
 - 19.4.3 FEST流畅断言模块
 - 19.4.4 Mylica继承断言
- 19.5 使用反射绕过封装
 - 19.5.1 内部替代物
 - 19.5.2 JUnit-addons
 - 19.5.3 FEST-Reflect
- 19.6 小结
- 附录A JUnit 3和JUnit 4之间的不同
 - A.1 全球的需求变化
 - A.1.1 JDK的要求
 - A.1.2 向后/向前兼容
 - A.2 API中的变化
 - A.2.1 包结构
 - A.2.2 构造函数
 - A.2.3 扩展TestCase
 - A.2.4 测试方法名称
 - A.3 注释与新增的静态导入
 - A.3.1 @Before与@After注释
 - A.3.2 @BeforeClass和@AfterClass注释
 - A.3.3 忽略测试的差异
 - A.3.4 静态导入
 - A.3.5 异常测试
 - A.3.6 超时测试
 - A.4 新的JUnit runners
 - A.4.1 测试运行器(Test runner)
 - A.4.2 测试集
 - A.4.3 参数化测试
 - A.5 新的断言和假设
 - A.5.1 Hamcrest断言
 - A.5.2 假设
 - A.5.3 新断言
 - A.5.4 断言错误
- 附录B 使用自定义的运行器和匹配器扩展JUnitAPI
 - B.1 介绍拦截器模式
 - B.2 实现自定义运行器
 - B.3 实现自定义匹配器
- 附录C 本书源代码
 - C.1 获取源代码
 - C.2 源代码概览
 - C.3 外部库
 - C.4 JAR版本
 - C.5 目录结构约定

<<JUnit实战>>

附录D JUnit IDE集成

D.1 JUnit与Eclipse的集成

D.1.1 安装Eclipse

D.1.2 从源代码创建Eclipse项目

D.1.3 从Eclipse运行JUnit测试

D.1.4 从Eclipse运行Ant脚本

D.2 引入JUnitMAX Eclipse插件

D.2.1 集成在你的开发周期中

D.2.2 执行顺序

D.2.3 恢复到上一个稳定版本

D.3 JUnit与NetBeans集成

D.3.1 安装NetBeans

D.3.2 从源代码中创建NetBeans项目

D.3.3 从NetBeans运行JUnit测试

D.3.4 从NetBeans运行Ant脚本

附录E 安装软件

E.1 安装HtmlUnit

E.1.1 标准配置

E.1.2 Eclipse的配置

E.2 使用HtmlUnit配置Cactus

E.3 安装Selenium

E.4 安装RhinoUnit

E.5 安装JsUnit

<<JUnit实战>>

章节摘录

版权页:第1部分 认识JUnit欢迎阅读《JUnit实战(第2版)》!

JUnit是一个由Kent Beck和ErichGamma1995年年底着手编写的框架。

自此以后,JUnit框架日益普及,现在已经成为单元测试Java应用程序的事实上的标准。

本书是第2版。

《JUnit实战》的第1版非常畅销,由Vincent Massol和Ted Husted于2003年编写,其内容是基于JUnit3.x版本的。

我们涵盖了JUnit最新的版本4.6,讨论了许多第1版尚未介绍的功能。

与此同时,我们关注其他一些有趣的测试代码的方法:mock objects,JUnit扩展、测试应用程序的不同层,等等。

这一部分从探索JUnit本身开始。

在本书后面的章节中,我们将专注于另一些工具和方法。

第1章 快速介绍了测试的概念。

你需要从基础知识入手逐步深入。

在本章的后半部分,我们会直接跳到代码内容,查看如何编写简单的测试、运行它并看到运行结果。

第2章 介绍了JUnit最核心的内容。

我们构建了一个稍大型的项目,并分析其代码。

我们不仅解释了JUnit的概念、Widget和内部构成,也为你展示了编写测试用例的最佳做法,并利用构建的项目对它们进行了说明。

第3章 重点介绍了测试。

我们描述了各种各样的测试以及它们所适用的情况。

我们还探讨了不同的平台(如开发、生产等),并展示了哪种测试和哪种情况能够在这些平台上最好地执行。

第1部分的最后一章致力于提升你的测试技巧。

我们告诉你如何衡量测试覆盖面以及如何提高测试覆盖面。

我们也解释了如何在编写测试之前生成可测试的代码,如何在动手编写代码之前编写测试。

媒体关注与评论

- “ 本书不仅针对JUnit、而且还针对常用的单元测试给出了权威的指导。
” ——Tyson S.Maxwell , Raytheon公司 “ 我向所有认真对待JUnit测试的人推荐这本书。
” ——Robert Hanson 《GW Tin Action》作者 “ 为单元测试打下一个稳固的基础，尤其是以Ant/Maven和Eclipse进行的单元测试。
” ——Doug Warren Java Web Services公司 “ 本书展示了如何想尽一切办法来进行测试。
” ——John Griffin 《Hibernate Search in Action》作者之一

<<JUnit实战>>

编辑推荐

《JUnit实战(第2版)》编辑推荐：JUnit是领先的Java单元测试框架，它的4.8版本在很大的程度上改善了Java的开发流程。

为了提高测试效率，JUnit针对新的应用程序类型（比如Ajax和基于HTML的表示层）以及应用程序框架（比如EJB和OSGi）进行了扩展。

塔凯文等编著的《JUnit实战（第2版）》经过彻底的修订，是目前针对Java应用程序进行单元测试的全新指南。

它提供了各种解决实际问题的技术。

例如，使用mocks进行隔离测试、为Java EE和数据库应用程序进行容器内测试以及测试自动化。

本书采用了实例驱动的编写风格。

并涵盖了JUnit 4.8的新功能，比如简化测试编写的新注释、改进的异常处理以及新的断言方法。

此外，你还会学到如何将JUnit与其他重要的开源框架、工具进行集成。

《JUnit实战(第2版)》适合于已具有一定Java编程基础的读者，以及在Java平台下进行各类软件开发的开发人员、测试人员。

对于单元测试学习者和编程爱好者来说，《JUnit实战(第2版)》则具有极高的学习参考价值。

版权说明

本站所提供下载的PDF图书仅提供预览和简介，请支持正版图书。

更多资源请访问:<http://www.tushu007.com>