

拥有近10年从业经验的资深软件开发工程师和测试工程师经验结晶，系统且实用，为开发高质量Android应用提供全方位指导

从测试和调试两个维度深度探索Android系统，为Android应用的自动化测试和调试提供原理性的解决方案



施懿民◎著

Android Application Testing and Debugging

Android应用测试 与调试实战

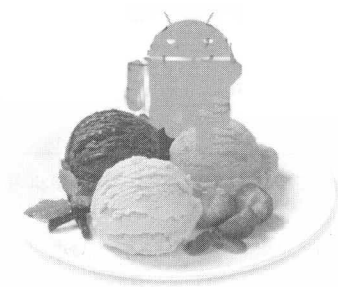


机械工业出版社
China Machine Press

Android Application Testing and Debugging

Android应用测试 与调试实战

施懿民◎著



机械工业出版社
China Machine Press

图书在版编目 (CIP) 数据

Android 应用测试与调试实战 / 施懿民著. —北京: 机械工业出版社, 2014.3
(移动开发)

ISBN 978-7-111-46018-3

I. A… II. 施… III. 移动终端—应用程序—程序设计 IV. TN929.53

中国版本图书馆 CIP 数据核字 (2014) 第 041414 号

本书是 Android 应用测试与调试领域最为系统、深入且极具实践指导意义的著作, 由拥有近 10 年从业经验的资深软件开发工程师和调试技术专家撰写, 旨在为广大程序员开发高质量的 Android 应用提供全方位指导。它从 Android 应用自动化测试工程师和开发工程师的需求出发, 从测试和调试两个维度, 针对采用 Java、HTML 5、C++&NDK 三种 Android 应用开发方式所需要的测试和调试技术、方法进行了细致而深入的讲解, 为 Android 应用的自动化测试和调试提供原理性的解决方案。

全书一共 16 章, 分为两大部分: 第一部分为自动化测试篇 (第 1~11 章), 详细讲解了进行 Android 自动化测试需要掌握的各种技术、工具和方法, 包括 Android 自动化测试基础、Android 应用的白盒自动化测试和黑盒自动化测试的技术和原理、Android 服务组件和内容组件的测试、HTML 5 应用和 NDK 应用的测试, 以及 Android 应用的兼容性测试和持续集成自动化测试; 第二部分为调试技术篇 (第 12~16 章), 详细讲解了 Android 应用调试所需要的各种工具的使用、操作日志的分析、内存日志的分析, 以及多线程应用 HTML 5 应用和 NDK 应用的调试方法和技巧。

Android 应用测试与调试实战

施懿民 著

出版发行: 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: 姜 影

印 刷: 藁城市京瑞印刷有限公司

开 本: 186mm×240mm 1/16

书 号: ISBN 978-7-111-46018-3

版 次: 2014 年 4 月第 1 版第 1 次印刷

印 张: 27.75

定 价: 79.00 元

凡购本书, 如有缺页、倒页、脱页, 由本社发行部调换

客服热线: (010) 88378991 88361066

购书热线: (010) 68326294 88379649 68995259

投稿热线: (010) 88379604

读者信箱: hzsj@hzbook.com

版权所有·侵权必究

封底无防伪标均为盗版

本书法律顾问: 北京大成律师事务所 韩光 / 邹晓东



前言

为什么要写这本书

几年前看过微软工程师 Adam Nathan 写的一本书《.NET and COM: The Complete Interoperability Guide》，对作者说过的一句话印象极其深刻：学习一门新技术的最佳方法就是写一本书。对于这个说法，笔者深以为然，在实际工作中，迫于项目交付的压力，很多时候只会使用一门技术中自己熟悉的部分，对于其他部分，甚至自己熟悉的那一部分，也只是知其然而不知其所以然，似懂非懂。而当要写一本书介绍这门技术的时候，就不得不去上下求索，这时不仅要横向了解该门技术的各个细节，还要纵向了解该门技术的发展思路以及各部分的来龙去脉，才能向读者解释它。因此可以说，写作本书的过程也是笔者系统学习 Android 操作系统的过程。

而笔者之所以从测试和调试的角度来介绍 Android 操作系统，是因为从这两个角度入手能够很好地从横向和纵向两个方向学习 Android 系统。从测试的角度来说，由于一般测试人员对 Android 应用进行集成测试和系统测试，开发人员负责单元测试工作，这就要求测试人员，特别是自动化测试人员对 Android 应用所涉及的技术有一个广度的认知，对 Android 的各种技术都要知其然。从调试的角度来说，却又可以从源码级别知各项技术的所以然。比如，要调试应用的功能错误，就必须对应用的源码有一个完整的理解，只有这样才能知道应该在什么地方设置断点，在什么时候让调试器捕捉异常；而要调试应用的内存泄露问题，就必须对 Android 系统采用的垃圾回收机制有一个通透的理解，只有这样才能从根源上发现并解决内存方面的问题。另外，反观市面上的技术书籍，介绍自动化测试和调试技术的书实在少。但在软件开发过程中，不仅测试是必不可少的环节，而且程序员实际上只花 20% 的时间写代码并完成编译过程，剩下 80% 的时间都是花在调试和修改代码上，这种情况也坚定了笔者写作本书的信心。

最后，由于笔者能力有限，同时为了及时出版本书，不得不放弃一些如多核设备上并行程序的调试、NDK 程序的验尸调试等内容，希望感兴趣的读者可以完成这方面的分享。

读者对象

- Android 自动化测试工程师
- Android 开发工程师
- 无编程基础的 Android 测试工程师

如何阅读本书

虽然本书讲解的是自动化测试和程序调试相关技术，但有些测试工程师的行业经验更丰富些，而编程基础则相对薄弱。因此本书分为两大部分：

第一部分为自动化测试篇（第 1 ~ 11 章），这一部分列举 Android 自动化测试中可以使用的几种测试技术，尽可能详细地介绍了 Android 白盒、黑盒自动化测试所用到的技术及其原理。由于 Android 应用可以使用 Java 语言配合 SDK，也可以用 HTML 5 技术，还可以用 C/C++ 语言配合 NDK 技术编写，所以这部分尽量涵盖了针对这三种技术编写的应用所采用到的测试技术。这些内容适合 Android 自动化测试工程师和对自动化测试感兴趣的手工测试工程师阅读。这一部分除了第 11 章需要有 C/C++ 编程经验之外，其他章节无需编程基础即可阅读。另外每个章节都是独立的，读者可以根据自己的实际需要分开来阅读。

第二部分为调试技术篇（第 12 ~ 16 章），第 12 章讲解的是通用的调试技术，这部分对于 Android 自动化测试工程师和 Android 开发工程师都是必要的知识，这一章节无需编程知识即可掌握。而第 13 章之后，主要涉及的是性能方面的调试技术，其中涉及一些 Android 系统内部的实现细节，这些技术更适合具备一定开发经验的自动化测试和开发工程师阅读。

勘误和支持

由于作者的水平有限，加之编写时间仓促，书中难免会出现一些错误或者不准确的地方，恳请读者批评指正。为此，作者特意在 Google+ 上创建一个在线支持与应急方案的社区 <https://plus.google.com/u/0/communities/112928495323595574856>。你可以将书中的错误发布在 Bug 勘误表页面中，同时如果你遇到任何问题，也可以访问 Q&A 页面，作者将尽量在线上提供满意的解答。书中的全部源文件除可以从华章网站^①下载外，还可以从这个社区和 github 上 <https://github.com/shiyimin/androidtestdebug> 下载，作者也会将相应的功能更新及时发布出来。如果你有更多的宝贵意见，也欢迎发送邮件至邮箱 shiyimin.aaron@gmail.com，期待能够得到你们的真挚反馈。

① 参见华章网站 www.hzbook.com。——编辑注

致谢

首先要感谢支付宝的陈晔（新浪微博：[@Monkey 陳曄曄](#)），在本书写作时，他参与了大部分章节的技术审阅工作，帮助完善了书中的细节。

感谢阿里集团的梁剑钊（新浪微博：[@liangjz](#)）推荐的玄黎（新浪微博：[@浪头](#)）、李子乐（新浪微博：[@子乐_淘宝太禅](#)）参与本书技术审阅工作。

感谢机械工业出版社华章公司的编辑杨福川老师，在这一年多的时间中始终支持我的写作，他的鼓励和帮助引导我顺利完成全部书稿。

谨以此书献给我最亲爱的家人，以及众多热爱 Android 测试的朋友们！

施懿民

目 录

前言

第 1 章 Android 自动化测试初探 1

- 1.1 快速入门 1
- 1.2 待测示例程序 2
- 1.3 第一个 Android 应用测试工程 6
- 1.4 搭建自动化开发环境 12
 - 1.4.1 安装 Eclipse 和 ADT
开发包 12
 - 1.4.2 创建模拟器 13
 - 1.4.3 启动模拟器 21
 - 1.4.4 连接模拟器 23
 - 1.4.5 连接手机 24
- 1.5 本章小结 29

第 2 章 Android 自动化测试基础 30

- 2.1 Java 编程基础 30
- 2.2 JUnit 简介 36
 - 2.2.1 添加测试异常情况的
测试用例 41
 - 2.2.2 测试集合 43
 - 2.2.3 测试准备与扫尾函数 45
 - 2.2.4 自动化测试用例编写
注意事项 47
- 2.3 Android 应用程序基础 47

- 2.3.1 Android 权限系统 47
- 2.3.2 应用的组成与激活 51
- 2.3.3 清单文件 54
- 2.3.4 Android 应用程序的单
UI 线程模型 56
- 2.4 本章小结 57

第 3 章 Android 界面自动化 白盒测试 58

- 3.1 Instrumentation 测试框架 58
 - 3.1.1 Android 仪表盘测试工程 58
 - 3.1.2 仪表盘技术 60
 - 3.1.3 Instrumentation.
ActivityMonitor 嵌套类 63
- 3.2 使用仪表盘技术编写测试用例 64
 - 3.2.1 ActivityInstrumentationTest-
Case2 测试用例 66
 - 3.2.2 sendKeys 和
sendRepeatedKeys 函数 70
 - 3.2.3 执行仪表盘测试用例 72
 - 3.2.4 仪表盘测试技术的限制 74
- 3.3 使用 robotium 编写集成测试
用例 77
 - 3.3.1 为待测程序添加
robotium 用例 77

3.3.2 测试第三方应用	80	6.1.1 依赖注入	144
3.3.3 robotium 关键源码解释	84	6.1.2 服务定位器	146
3.4 Android 自动化测试在多种 屏幕下的注意事项	87	6.2 内容供应组件	147
3.5 本章小结	90	6.2.1 统一资源标识符	150
第 4 章 Android 界面自动化 黑盒测试	91	6.2.2 MIME 类型	152
4.1 monkey 工具	91	6.2.3 内容供应组件的虚拟表 视图	152
4.1.1 运行 monkey	93	6.3 内容供应组件示例	154
4.1.2 monkey 命令选项参考	97	6.4 测试内容供应组件	159
4.1.3 monkey 脚本	98	6.5 本章小结	163
4.1.4 monkey 服务器	105	第 7 章 测试 Android HTML 5 应用	164
4.2 编写 monkeyrunner 用例	109	7.1 构建 Android HTML 5 应用	164
4.2.1 为待测程序录制和 回放用例	110	7.1.1 WebView 应用	164
4.2.2 运行 monkeyrunner	110	7.1.2 使用视口适配 Android 设备的多种分辨率	170
4.2.3 手工编写 monkeyrunner 代码	111	7.1.3 使用 CSS 适配多种 分辨率	175
4.2.4 编写 monkeyrunner 插件	114	7.1.4 使用 Chrome 浏览器模拟 移动设备浏览器	176
4.3 本章小结	118	7.2 使用 QUnit 测试 HTML 5 网页	177
第 5 章 测试 Android 服务组件	119	7.2.1 QUnit 基础	177
5.1 JUnit 的模拟对象技术	119	7.2.2 QUnit 中的断言	179
5.2 测试服务对象	128	7.2.3 测试回调函数	181
5.2.1 服务对象简介	128	7.2.4 测试 WebView 应用	182
5.2.2 在应用中添加服务	130	7.3 本章小结	185
5.2.3 测试服务对象	136	第 8 章 使用 Selenium 测试 HTML 5 浏览器应用	186
5.3 本章小结	140	8.1 Selenium 组成部分	186
第 6 章 测试 Android 内容供应 组件	142	8.2 安装 Selenium IDE	187
6.1 控制反转	142		

8.3 Selenium IDE 界面	188	第 10 章 Android 其他测试	232
8.3.1 菜单栏	188	10.1 Android 兼容性测试	232
8.3.2 工具栏	189	10.1.1 运行 Android 兼容性 测试用例集合	232
8.4 使用 Selenium	189	10.1.2 兼容性测试计划说明	237
8.4.1 使用 Selenium IDE 录制 测试用例	189	10.1.3 添加一个新的测试 计划	238
8.4.2 运行 Selenium 测试用例	194	10.1.4 添加一个新的测试 用例	239
8.4.3 等待操作完成	199	10.1.5 调查 CTS 测试失败	241
8.4.4 Selenium WebDriver 命令	200	10.2 Android 脚本编程环境	243
8.5 数据驱动测试	206	10.2.1 Android 脚本环境简介	243
8.6 Selenium 编程技巧	208	10.2.2 安装 SL4A	243
8.6.1 在测试代码中硬编码 测试数据	208	10.2.3 为 SL4A 安装脚本引擎	244
8.6.2 重构 Selenium IDE 生成 的代码	209	10.2.4 编写 SL4A 脚本程序	246
8.7 本章小结	212	10.2.5 在 PC 上调试脚本程序	250
第 9 章 Android NDK 测试	213	10.3 国际化测试	251
9.1 安装 NDK	213	10.4 模拟来电中断测试	254
9.2 NDK 的基本用法	214	10.5 本章小结	255
9.3 编译和部署 NDK 示例程序	214	第 11 章 持续集成自动化测试	257
9.4 Java 与 C/C++ 之间的交互	217	11.1 在 Ant 中集成 Android 自动化测试	257
9.4.1 Makefiles	222	11.1.1 Ant 使用简介	257
9.4.2 动态模块和静态模块	222	11.1.2 Android 应用编译过程	262
9.5 在 Android 设备上执行 NDK 单元测试	223	11.1.3 使用 Ant 编译 Android 工程	263
9.6 unittest++ 使用基础	228	11.2 在 Maven 中集成 Android 自动化测试	268
9.6.1 添加新测试用例	228	11.2.1 使用 Android Maven Archetypes 创建新 Android 工程	268
9.6.2 测试用例集合	229	11.2.2 Android Maven 工程 介绍	270
9.6.3 验证宏	229		
9.6.4 数组相关的验证宏	230		
9.6.5 设置超时	230		
9.7 本章小结	231		

11.2.3	与设备交互	271
11.2.4	与模拟器交互	272
11.2.5	集成自动化测试	274
11.3	收集代码覆盖率	276
11.4	本章小结	280

第 12 章 Android 功能调试工具 ... 281

12.1	使用 Eclipse 调试 Android 应用	281
12.1.1	Eclipse 调试技巧	282
12.1.2	使用 JDB 调试	294
12.1.3	设置 Java 远程调试	296
12.1.4	调试器原理简介	301
12.2	查看 Android 的 logcat 日志	302
12.2.1	过滤 logcat 日志	303
12.2.2	查看其他 logcat 内存日志	304
12.3	Android 调试桥接	304
12.3.1	adb 命令参考	306
12.3.2	执行 Android shell 命令	309
12.3.3	dumpsys	312
12.4	调试 Android 设备上的程序	317
12.4.1	调试命令行程序	317
12.4.2	调试 Android 应用	318
12.4.3	调试 Maven Android 插件启动的应用	321
12.5	本章小结	322

第 13 章 Android 性能测试之 分析操作日志 ... 323

13.1	使用 Traceview 分析操作日志	326
13.1.1	记录应用操作日志	326

13.1.2	Traceview 界面说明	328
13.1.3	使用 Traceview 分析并优化性能瓶颈	329
13.2	使用 DDMS	334
13.2.1	使用 DDMS	335
13.2.2	DDMS 与调试器交互的原理	336
13.2.3	三种启动操作日志记录功能的方法	338
13.3	使用 dmtracedump 分析函数调用树	339
13.4	本章小结	341

第 14 章 分析 Android 内存问题 ... 343

14.1	Android 内存管理原理	343
14.1.1	垃圾内存回收算法	343
14.1.2	GC 发现对象引用的方法	351
14.1.3	Android 内存管理源码分析	352
14.1.4	Logcat 中的 GC 信息	361
14.2	调查内存泄露工具	362
14.2.1	Shallow size 和 Retained size	362
14.2.2	支配树	363
14.3	分析 Android 内存泄露实例	364
14.3.1	在 DDMS 中检查示例问题程序的内存情况	366
14.3.2	使用 MAT 分析内存泄露	368
14.3.3	弱引用	372
14.3.4	MAT 的其他界面使用方法	373

14.3.5	对象查询语言 OQL (Object Query Language).....	376
14.3.6	使用 jHat 分析内存文件.....	381
14.4	显示图片.....	382
14.4.1	Android 应用加载大图片的最佳实践.....	386
14.4.2	跟踪对象创建.....	388
14.5	频繁创建小对象的问题.....	390
14.6	Finalizer 的问题.....	393
14.7	本章小结.....	394

第 15 章 调试多线程和 HTML 5

应用	395
15.1 调试应用无响应问题	395
15.2 Android 中的多线程	397
15.3 调试线程死锁	400
15.3.1 资源争用问题	400
15.3.2 线程同步机制	405
15.3.3 解决线程死锁问题	406
15.4 StrictMode	410

15.4.1	在应用中启用 StrictMode.....	413
15.4.2	暂时禁用 StrictMode.....	415
15.5	调试 Android 上的浏览器应用.....	416
15.5.1	在 Android 系统自带的浏览器上调试.....	416
15.5.2	在 Chrome 浏览器上调试.....	418
15.6	本章小结.....	422

第 16 章 调试 NDK 程序..... 423

16.1	使用 Eclipse 调试 Android NDK 程序.....	423
16.2	在命令行中调试 NDK 程序.....	426
16.3	Android 的 C/C++ 调试器的工作原理.....	431
16.3.1	调试符号.....	433
16.3.2	源码.....	433
16.3.3	多线程调试的问题.....	433
16.4	本章小结.....	434

第 1 章

Android 自动化测试初探

本章的目的是让有经验的测试开发工程师迅速上手 Android 自动化测试代码的编写流程。因为大部分 Android 程序都是基于 SDK 开发，所以本章只介绍为基于 Android SDK 的应用编写自动化测试代码，想了解测试 Android NDK 应用的方法可阅读第 9 章。

1.1 快速入门

Android 开发环境的安装，由于种种原因，变得非常难于实现，这在一定程度上给 Android 开发的初学者带来了很大的不便。为了帮助读者将主要精力放在学习 Android 自动化测试的开发技术上，而不是将时间浪费在环境的准备和安装上，本书配套资源提供了一个 VirtualBox 虚拟机，方便读者学习和尝试本书里讲解的各种技术。本书后续章节的示例代码和示例命令，如果没有特殊说明，均使用该虚拟机演示。使用该虚拟机可参照以下步骤：

- 1) 下载并安装最新版本的 VirtualBox: <https://www.virtualbox.org/wiki/Downloads>。
- 2) 打开 VirtualBox，依次单击菜单栏的“控制”和“注册”菜单项，导入“AndroidBook.vbox”虚拟机。
- 3) 在 VirtualBox 中选择刚刚注册的虚拟机，单击工具栏里的“启动”按钮启动它。
- 4) 在虚拟机上安装的是编写本书时最新的 Ubuntu 12.04 操作系统，超级用户的用户名和密码均为“student”。
- 5) 在虚拟机中已经准备好了 Eclipse、Android 开发环境和使用真机设备的相关配置。在虚拟机中单击“Dash”图标（或者同时按下键盘的 Alt 和 F2），并输入“gnome-terminal”打开终端程序，如图 1-1 所示。



图 1-1 在附带虚拟机中打开终端程序

6) 在终端中输入下列命令启动 Eclipse:

```
$ ~/eclipse/eclipse
```

注意

在 Windows 中, 如果当前登录用户的用户名是中文, 当在资源管理器中双击 VirtualBox 的 .msi 安装包尝试安装时, VirtualBox 安装程序会弹出“系统找不到指定的路径”的错误消息框, 如图 1-2 所示。

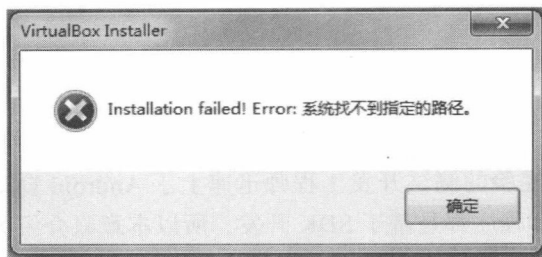


图 1-2 “系统找不到指定的路径”的错误消息框

这是 VirtualBox 安装程序的一个缺陷, 因为 MSI 安装程序运行时会解压一些临时文件到登录用户的临时文件夹中, 而 VirtualBox 安装程序对包含中文的路径支持得不是很好, 就会弹出这样的对话框, 所以这里建议使用英文名的管理员用户安装。

1.2 待测示例程序

本章示例所采用的待测程序是一个简单的 Android 应用, 模拟数据库程序的增删改查功能。程序的主界面是一个书籍列表界面, 按作者名列出了每个作者的著作书名。在列表中单击书名可以查看书籍的详细信息, 在详细信息界面上单击“编辑”按钮可以编辑书籍的信息, 完成后单击“保存”按钮即可保存更改并返回到列表界面。列表界面上有“删除”和“添加”按钮, 向列表中添加一本新书籍的操作与“编辑”类似; 在从列表中删除一本书时, 需要先单击“删除”按钮, 然后再单击要删除的书籍。待测示例程序的主界面如图 1-3 所示。

待测程序的源代码可以在配套资源(或虚拟机的 /home/student/samplecode 中)的“chapter1\cn.hzbook.android.test.chapter1”文件夹中找到, 按照下面的步骤在 Eclipse 中导入该工程:

- 1) 启动 Eclipse。
- 2) 依次单击 Eclipse 菜单栏中的“File”、“Import...”菜单项。
- 3) 在新弹出的“Import”对话框中选择“Existing Projects into Workspace”列表项, 然后单击“Next”按钮进入下一步, 如图 1-4 所示。

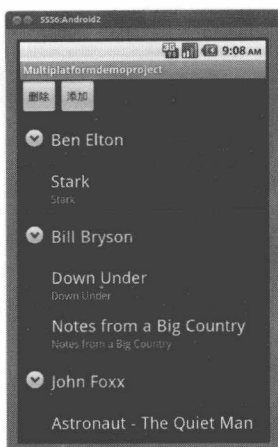


图 1-3 待测示例程序的主界面截图

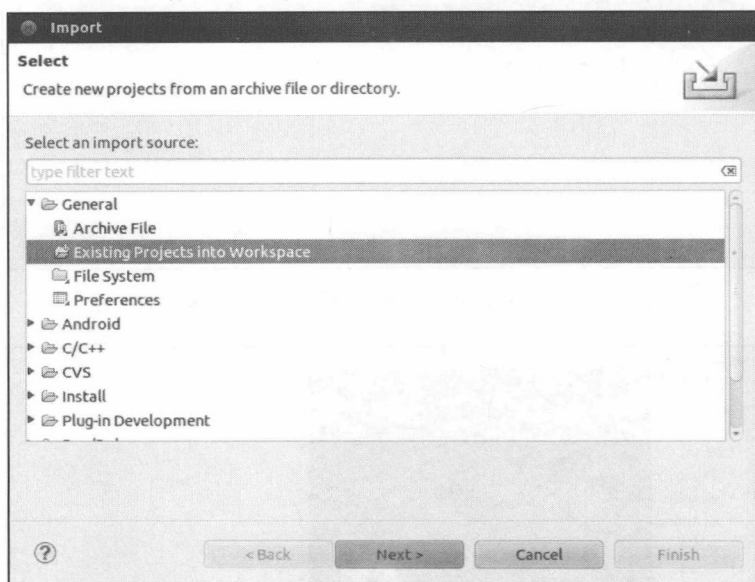


图 1-4 在 Eclipse 中导入待测示例应用工程

4) 在“Import Projects”界面中勾选“Select root directory”项，并单击旁边的“Browse...”按钮，填入配套资源中待测应用源文件的根目录。完成后应该可以在“Projects”列表框中看到要导入的工程名：“cn.hzbook.android.test.chapter1”。勾选“Copy projects into workspace”复选框，以便将应用的源代码复制到本地硬盘。最后单击“Finish”完成导入操作。如图 1-5 所示。

5) 完成导入后，用右键单击刚导入的工程文件，并依次选择“Run As”，“Android

Application”运行应用。如图 1-6 所示。

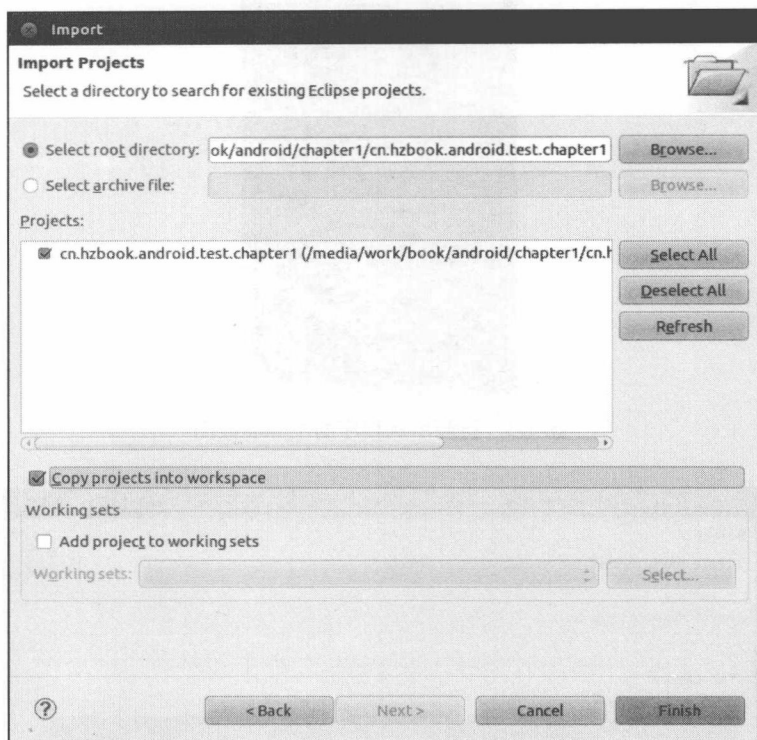


图 1-5 在 Eclipse 中导入工程向导中选择待测示例工程

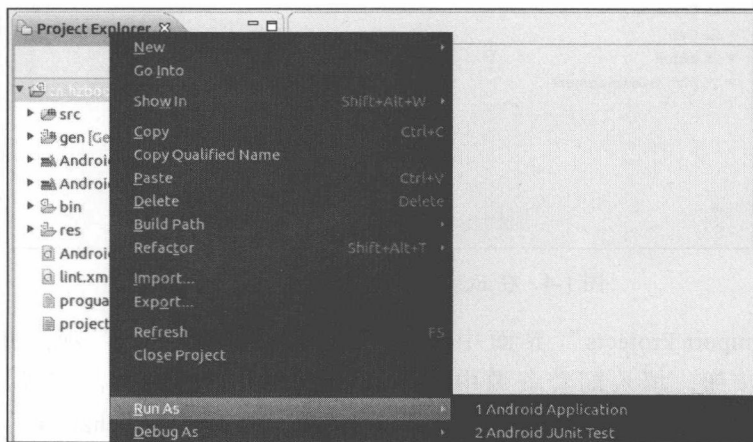


图 1-6 在 Eclipse 中运行示例待测应用

6) 这时 Eclipse 会自动启动 Android 模拟器并打开应用。此时打开 Eclipse 下方的“Console”窗口并选择“Android”下拉框，应该可以看到类似图 1-7 的输出。

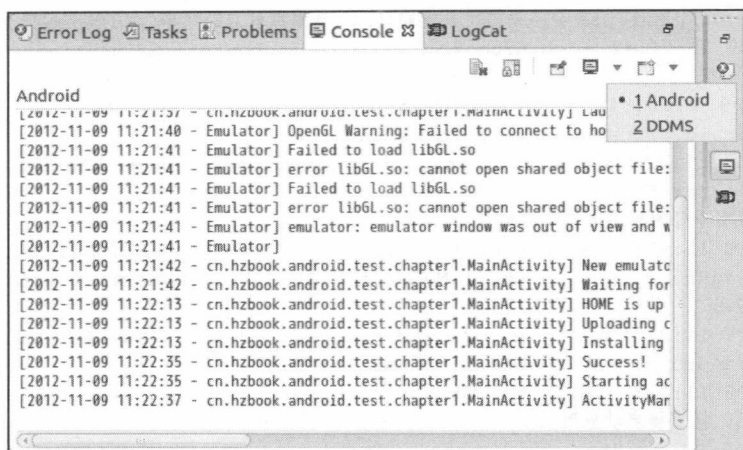


图 1-7 在 Eclipse 中启动应用的过程输出

图 1-7 的输出中详细显示了 Eclipse 从启动模拟器到运行应用的完整过程。在测试过程中，会经常用到输出内容来排查错误，下面结合注释解释输出内容：

```
# 在这里 Eclipse 接受到启动 Android 应用的命令
[2012-11-09 11:21:37 - cn.hzbook.android.test.chapter1.MainActivity] Android
Launch!

#Eclipse 首先确定用来与模拟器进行通信（调试程序）的 adb 程序是否运行，如果没有运行就会启动它
#adb 程序的用法会在后面调试章节中讲解
[2012-11-09 11:21:37 - cn.hzbook.android.test.chapter1.MainActivity] adb is
running normally.

# 找到应用的主 Activity 并启动它
[2012-11-09 11:21:37 - cn.hzbook.android.test.chapter1.MainActivity] Performing
cn.hzbook.android.test.chapter1.MainActivity activity launch

# 自动选择最合适的模拟器，这里因为示例应用最低要求 Android 2.2 版本，而系统中正好有
# Android 2.2 版本的模拟器，所以直接做出最佳选择
[2012-11-09 11:21:37 - cn.hzbook.android.test.chapter1.MainActivity] Automatic
Target Mode: launching new emulator with compatible AVD 'Android2'

[2012-11-09 11:21:37 - cn.hzbook.android.test.chapter1.MainActivity] Launching
a new emulator with Virtual Device 'Android2'

# 这些错误与 OpenGL 有关，可以忽略它们
[2012-11-09 11:21:40 - Emulator] OpenGL Warning: Failed to connect to host.
Make sure 3D acceleration is enabled for this VM.

[2012-11-09 11:21:41 - Emulator] Failed to load libGL.so
[2012-11-09 11:21:41 - Emulator] error libGL.so: cannot open shared object file:
No such file or directory
[2012-11-09 11:21:41 - Emulator] Failed to load libGL.so
[2012-11-09 11:21:41 - Emulator] error libGL.so: cannot open shared object file:
No such file or directory

# 模拟器会自动将自身窗口调整到屏幕的最佳位置
[2012-11-09 11:21:41 - Emulator] emulator: emulator window was out of view and was
recentered
[2012-11-09 11:21:41 - Emulator]
```



```
# 在模拟器启动时, Eclipse 会不停扫描系统端口, 一旦模拟器启动完毕, Eclipse 就会连接上它
[2012-11-09 11:21:42 - cn.hzbook.android.test.chapter1.MainActivity] New emulator
found: emulator-5554
[2012-11-09 11:21:42 - cn.hzbook.android.test.chapter1.MainActivity] Waiting
for HOME ('android.process.acore') to be launched...
[2012-11-09 11:22:13 - cn.hzbook.android.test.chapter1.MainActivity] HOME is up
on device 'emulator-5554'
#Eclipse 将编译好的应用上传到模拟器
[2012-11-09 11:22:13 - cn.hzbook.android.test.chapter1.MainActivity] Uploading
cn.hzbook.android.test.chapter1.MainActivity.apk onto device 'emulator-5554'
# 接着安装应用
[2012-11-09 11:22:13 - cn.hzbook.android.test.chapter1.MainActivity] Installing
cn.hzbook.android.test.chapter1.MainActivity.apk...
[2012-11-09 11:22:35 - cn.hzbook.android.test.chapter1.MainActivity] Success!
# 成功安装后, 就直接启动应用
[2012-11-09 11:22:35 - cn.hzbook.android.test.chapter1.MainActivity] Starting
activity cn.hzbook.android.test.chapter1.MainActivity on device emulator-5554
[2012-11-09 11:22:37 - cn.hzbook.android.test.chapter1.MainActivity] ActivityManager:
Starting: Intent { act=android.intent.action.MAIN cat=[android.intent.category.
LAUNCHER] cmp=cn.hzbook.android.test.chapter1/.MainActivity }
```

注意

模拟器启动后, 默认处于锁屏状态, 需要手动解锁, 解锁后就会看到应用已经启动。自动解锁的方式在 10.3 节的代码清单 10-10 中说明。

1.3 第一个 Android 应用测试工程

本章先建立一个简单的 Android 自动化单元测试工程来演示 Android 自动化测试的流程。在 Android 系统中, Android 自动化单元测试也是一个 Android 应用工程, 它跟普通 Android 应用工程不同的地方是启动的方式不一样, 这在本书第 3 章会讲到。

Android 的 Eclipse ADT 插件提供了 Android 自动化单元测试的模板, 方便我们创建自动化测试项目, 这里新建一个测试工程:

- 1) 启动 Eclipse, 这次可以看到之前在工作空间已被导入的 Android 工程。
- 2) 依次单击 Eclipse 菜单栏里的“File”、“New”、“Project...”菜单项, 如图 1-8 所示。

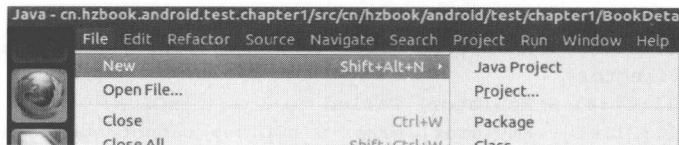


图 1-8 在 Eclipse 中新建工程 (1)

3) 在弹出的“New Project”对话框中, 展开“Android”列表项, 并选择“Android Test Project”项来指明要创建一个 Android 自动化测试工程, 然后单击“Next”按钮, 如