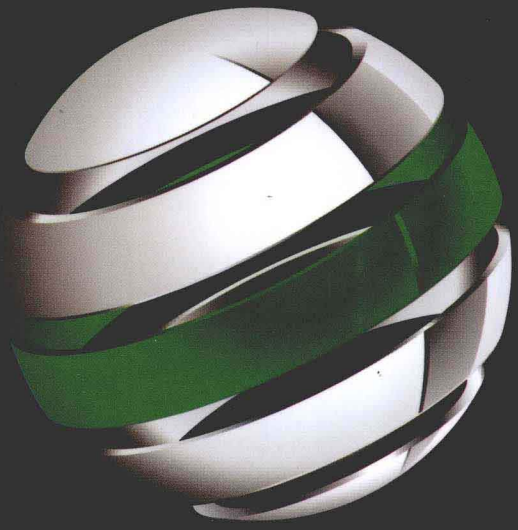
TURING 图灵程序设计丛书 移动开发系列

Apress®

- Android实用技巧大全
- 理论与实践的完美结合
- 亚马逊书店好评如潮



Android Recipes A Problem-Solution Approach

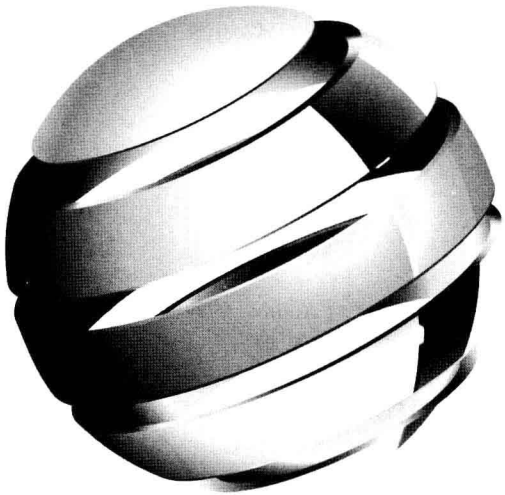
Android攻略

[美] Dave Smith
[加] Jeff Friesen 著

陈钢 译

人民邮电出版社
POSTS & TELECOM PRESS

TURING 图灵程序设计丛书 移动开发系列



Android Recipes A Problem-Solution Approach

Android攻略

[美] Dave Smith
[加] Jeff Friesen 著

陈钢 译

人民邮电出版社
北京

图书在版编目（CIP）数据

Android攻略 /（美）史密斯（Smith，D.），（加）弗里森（Friesen，J.）著；陈钢译. －北京：人民邮电出版社，2012.7
（图灵程序设计丛书）
书名原文：Android Recipes：A Problem-Solution Approach
ISBN 978-7-115-28451-8

Ⅰ.①A… Ⅱ.①史… ②弗… ③陈… Ⅲ.①移动终端－应用程序－程序设计 Ⅳ.①TN929.53

中国版本图书馆CIP数据核字（2012）第117670号

内 容 提 要

本书全方位讲解Google开放移动应用平台Android的各种开发技术。通过完整而真实的示例代码，介绍应用架构和多种特定于Android的API，讲述如何使用NetBeans、Eclipse、App Inventor等工具开发应用，如何通过Android NDK提高应用性能等。

编写Android应用程序的开发人员将是本书的最大获益者。

图灵程序设计丛书
Android攻略

◆ 著　［美］Dave Smith　［加］Jeff Friesen
译　陈　钢
责任编辑　毛倩倩
执行编辑　丁晓昀
◆ 人民邮电出版社出版发行　北京市崇文区夕照寺街14号
邮编　100061　电子邮件　315@ptpress.com.cn
网址　http://www.ptpress.com.cn
北京艺辉印刷有限公司印刷
◆ 开本：800×1000　1/16
印张：23.25
字数：550千字　2012年7月第1版
印数：1-4 000册　2012年7月北京第1次印刷
著作权合同登记号　图字：01-2011-3259号
ISBN 978-7-115-28451-8

定价：69.00元

读者服务热线：(010)51095186转604　印装质量热线：(010)67129223
反盗版热线：(010)67171154

版 权 声 明

序

Dave Smith 和 Jeff Friesen 为这本书的诞生付出了大量的时间和精力。我很早就在移动开发社区里认识了 Dave。我知道这本书中的每一章都凝结着他的努力，每一条攻略都经过了反复讨论。为什么我知道这些？因为我有幸每天跟 Dave 在一起工作，在我们推出 Android 软件的过程中，他为我们遇到的各种问题提供了系统化、标准化、严谨的解决方案。

Android 设备在短时间内爆炸式发展，这是一个改变移动计算未来发展的难得机遇。Android 支持电话、平板电脑、工业设备，将来甚至还会支持我们未曾见过的设备。各种各样的设备运行着同样的系统，这将开发人员“一次编写，随处部署”的梦想变为现实。本书中，Dave 和 Jeff 展示了他们在开发 Android 应用时遇到的实例，由此带你开始 Android 开发的学习旅程。把握这个机会，给用户创造出有着完美使用体验的移动应用。当你的应用上线时，这些 Android 设备就成了你的舞台。跟层出不穷的移动设备一起汹涌而来的还有各种各样的应用，不过其中大部分都是垃圾。潜心琢磨用户需求，解决他们所遇到的问题，创造出引以为傲的佳作。关注细节才能获得用户的青睐——记住，“Real Artists Ship”[①]。

Ben Reubenstein（@benr75）
benr@xcellentcreations.com
Xcellent Creations 公司

① 这是苹果公司创始人乔布斯的名言，“真正的艺术家才能让产品上市”，意即要追求完美的细节。——译者注

前　言

欢迎阅读《Android 攻略》!

如果你正在阅读本书，那移动设备给软件开发人员和用户带来的无限机遇就不用我再赘述了。近年来，Android 已经成为了最主要的移动平台之一。对开发人员而言，必须要了解 Android 才能确保自己能跟得上市场的变化，从而把握住各种潜在的机会。但是任何新平台在常见需求的开发和常见问题的解决方案上都会有不确定性。

我们撰写本书的目的是帮助开发人员解决实际开发中的问题，通过直观的例子告诉读者如何编写 Android 平台上的应用。本书不会很深入地介绍 Android SDK、NDK 或是其他工具。我们不会让隐藏其中的各种琐碎细节和高深理论打击读者的积极性。这并不是说这些细节没意思或是不重要。读者应该研究这些细节，以避免在开发中犯下错误。但在解决迫在眉睫的问题时，这些东西通常只会让人分心。

本书不会讲解 Java 编程，也不会介绍如何构建 Android 应用。本书略去了很多基础知识（例如，如何用 TextView 控件显示文本），因为我们觉得这些知识在学过之后就不会遗忘。相反，本书会帮助开发人员解决很多实际开发中经常要完成的任务，而这些复杂的任务不是寥寥几行代码就能完成的，自然也很难记住。

读者可以把本书当作一本可供随时查询的参考书、一本资源丰富的实例手册，随时都可以从中找到有助于高效完成工作的实用建议。

本书主要内容

尽管本书并不是针对新手的 Android 开发教程，但我们还是在第 1 章中概述了理解全书所需的 Android 基础知识。第 1 章还介绍了如何构建开发 Android 应用所需的开发环境。具体来说，就是如何安装 Android SDK、Eclipse 及其 ADT 插件。

随着 Android 开发经验的增长，为了节约时间，肯定要尽力避免重新发明轮子。开发人员应该创建和使用自己的可重用代码库，或者使用其他人开发的库。第 7 章会说明如何创建和使用自定义的 JAR 形式的代码库和 Android 库项目。此外，还会介绍一些不包括在 Android SDK 中的 Java 库，你可以适时使用。

在其他几章中，我们会深入讲解如何用 Android SDK 解决各种实际问题。你将会学到如何高效地创建能运行在各种设备上的用户界面。你将会成为整合各种硬件（收音机、传感器和摄像头）的专家，正是这些硬件让移动设备成为一个独具特色的平台。我们甚至还会讨论如何自

行定制这个系统，集成 Google 提供的各种服务和应用，并兼容各个设备制造商的产品。以此为目标，我们还会推荐一些由社区开发的工具，用于简化应用的开发和测试。

你对脚本语言（例如 Python 和 Ruby）感兴趣吗？如果感兴趣的话，你应该读一读附录 A，其中涵盖了 Scripting Layer for Android。这个特别的应用可以支持在 Android 上安装脚本语言解释器，在设备上编写脚本并运行，以提高开发速度。

如果想要开发成功的应用，性能问题是不可忽视的。大部分时候，这都不是问题，因为 Android（从 2.2 版开始）的 Dalvik 虚拟机有一个 Just-In-Time 的编译器，能将 Dalvik 字节码编译成设备的本地代码。如果这还不够，还可以利用 Android 的 NDK 进一步提升性能。附录 B 详述了 NDK，并用一个 OpenGL 的示例演示了它的用途。

在创建应用时，需要确保应用的性能好、响应速度快，且能与系统无缝衔接。低能耗、响应快、不会弹出 Application Not Responding（应用程序没有响应）窗口，且跟整个系统无缝衔接的应用才能让用户满意。此外，在将应用发布到 Google 的 Android Market 时，不能让不兼容的设备看到应用。应该要求 Android Market 过滤掉那些设备不兼容的用户，使之无法下载（甚至无法看到）你的应用。本书的附录 C 会指导你创建高性能、响应快而且与系统无缝衔接的应用，以及利用过滤功能只允许设备兼容的用户从 Android Market 下载该应用。

注意 API 级别

在整本书中，读者会看到绝大部分的解决方案都有一个相应的最低的 API 级别要求。本书中大部分解决方案都只需要 API Level 1，换言之就是这些代码能在 Android 1.0 以上的任何版本中运行。但是，有些地方也用到了较新版本中引入的 API。注意各个解决方案的 API 级别，确保代码与应用要支持的 Android 版本相匹配。

致　　谢

首先，我要感谢我的妻子 Lorie，感谢她在我撰写和构建本书所涉及的各种素材时给予我的支持和耐心。其次，感谢本书的另一位作者 Jeff Friensen，他对 Android 开发的新想法和新思路给本书带来了很多新意，让本书精彩绝伦。还要感谢我的好友和同事 Ben Reubenstein 在百忙中抽出时间为本书撰写序言，并将我引荐给 Apress 的编辑团队。最后，我要诚挚地感谢 Apress 给我和 Jeff 安排的支持团队：Steve Anglin、Corbin Collins、Tom Welsh、Paul Connolly 和其他所有的人。没有他们所投入的时间和精力，本书就不可能诞生。

——Dave Smith

我要感谢 Steve Anglin 邀请我撰写此书，感谢 Corbin Collins 在本书的各个方面给予我的指导，感谢 Tom Welsh 在我负责撰写的几章中给予的帮助，还要感谢 Paul Connolly 细心地找出了我书稿中的各种不足。最后还要感谢本书的另一位作者 Dave Simith 对本书的卓越贡献。

——Jeff Friesen

目　录

第1章 Android入门

Android 炙手可热，无数人在开发 Android 应用。你可能也想开发应用，但却苦于无从下手。Google 在网上发布的 *Android Developer's Guide*（http://developer.android.com/guide/index.html）中有开发应用所需的知识，但是其中浩如烟海的信息却容易让人晕头转向。与此不同的是，本章只会讲解一些理解 Android 所必需的理论知识。这些理论知识后面会有一些具体的解决方案介绍，告诉你如何开发应用，以及在将其发布到 Google 的 Android Market 前都要做哪些准备。

1.1 Android 简介

Android Developer's Guide 中将 Android 定义为针对移动设备的软件栈——用于实现完整的功能解决方案的软件子系统集合。这个栈中包括操作系统（由 Linux 内核修改而来）、部分基于 Java 的中间件（用于连接底层操作系统和高层应用的软件），以及诸如浏览器（即 Browser）和联系人管理器（即 Contacts）一类的关键应用（都是用 Java 编写的）。

Android 具有以下特点：

- 允许重用和替换应用组件的应用程序框架（本章稍后会展开讨论）；
- 支持蓝牙、EDGE、3G 和 Wi-Fi（需硬件支持）；
- 支持摄像头、GPS、罗盘和加速度计[①]（需硬件支持）；
- 针对移动设备优化的 Dalvik 虚拟机（Dalvik Virtual Machine，DVM）；
- GSM 电话支持（需硬件支持）；
- 集成了基于开源 WebKit 引擎的浏览器；
- 支持常见的音频、视频和静态图片格式（MPEG-4、H.264、MP3、AAC、AMR、JPG、PNG、GIF）；
- 自定义的 2D 图形库，提供了经优化的图形显示；基于 OpenGL ES 1.0 规范的 3D 图形显示（硬件加速可选）；
- 用于结构化数据存储的 SQLite。

Android 强大的开发环境（包括设备仿真器和 Eclipse IDE 的插件）并不是 Android 设备软

① 又称为重力感应器。——译者注

件栈的一部分，但也应该被视为Android的一大特点。

1.2 Android演化史

跟很多人想象的不一样，Android并不是源自Google。Android最早是由美国加州帕洛阿尔托市的一家名叫Android的创业公司开发的。2005年7月，Google收购了这家公司，并于2007年11月发布了Android SDK的预览版。

2008年8月中旬，Google发布了Android 0.9 SDK的beta版，一个月后发布了Android 1.0 SDK。表1-1罗列了SDK随后的更新。（从1.5版开始，每个主要版本都会有一个相应的代号，每个代号都是某种甜点的名称。[①]）

表1-1 Android的主要版本

SDK版本	发布日期和更新内容
1.1	Google在2009年2月9日发布了SDK 1.1。更新内容包括付费应用（通过Android Market）和支持语音搜索
1.5（Cupcake，杯式蛋糕）基于Linux内核2.6.27	Google在2009年4月30日发布了SDK 1.5。更新内容包括通过摄像模式拍摄和观看视频、上传视频到YouTube、上传照片到Picasa、在主屏幕上添加小部件及屏幕切换动画
1.6（Donut，甜甜圈）基于Linux内核2.6.29	Google在2009年9月15日发布了SDK 1.6。更新内容包括改进Android Market使用体验，集成了拍照、摄像和相册界面，提升了语音搜索速度，完善了其他一些功能，还升级了搜索体验
2.0/2.1（Eclair，泡芙）基于Linux内核2.6.29	Google在2009年10月26日发布了SDK 2.0。更新内容包括改良用户界面，新的联系人列表，支持微软Exchange，支持数码变焦，改进的Google地图（3.1.2版），浏览器支持HTML5，动态壁纸，支持蓝牙2.1。 Google随后在2009年12月3日发布了SDK 2.0.1，在2010年1月12日发布了SDK 2.1
2.2（Froyo，冻酸奶）基于Linux内核2.6.32	Google在2010年5月20日发布了SDK 2.2。更新内容包括在浏览器中集成Chrome的V8 JavaScript引擎，通过蓝牙实现语音拨号和共享联系人，支持Adobe Flash 10.1，利用JIT进一步提升应用速度，USB Tethering和Wi-Fi热点功能[②]
2.3（Gingerbread，姜饼）基于Linux内核2.6.35.7	Google在2010年12月6日发布了SDK 2.3。更新内容包括能提升应用响应速度的新的并发垃圾收集器，支持陀螺仪传感器，支持WebM视频格式，改进了其他视频的播放效果，支持近场通信，同时改善了社交网络功能。本书重点讨论基于Android 2.3的应用开发。 Google随后发布了SDK 2.3.1和SDK 2.3.3，前者修复了一些bug，后者对Android 2.3平台做了一些改进，添加了几个API

① 细心的读者会发现，随着版本号的增大，甜点的尺寸也越来越大。——编者注

② USB Tethering令电脑可以通过USB连接Android手机网络上网，Wi-Fi热点功能可以把手机变成一个Wi-Fi接入点，供多个设备上网。——编者注

（续）

SDK 版本	发布日期和更新内容
3.0（Honeycomb，蜂巢）基于 Linux 内核 2.6.36	Google 在 2011 年 2 月 22 日发布了 SDK 3.0。跟之前发布的版本不同，3.0 版针对的是平板电脑，例如（2011 年 2 月 24 日）发布的第一个 Android 平板电脑摩托罗拉 Zoom[①]。除了用户界面的改进，3.0 还改进了多任务，支持多核处理器，支持硬件加速，提供了 3D 桌面并带有全新设计的小部件

1.3 Android 系统架构

Android 软件栈的顶层是应用，中间是中间件（由应用框架、库和 Android 运行时组成），底层则是带有各种驱动的 Linux 内核。图 1-1 展示了这种分层架构。

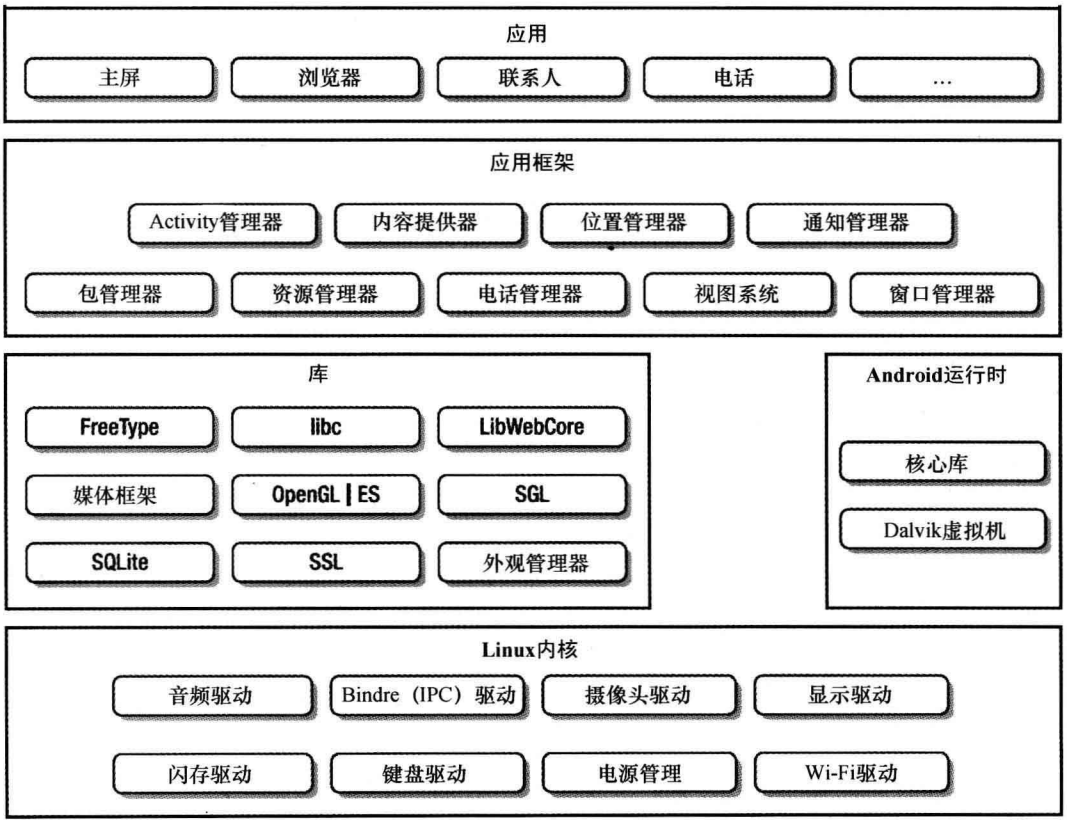


图 1-1 Android 的分层架构由若干主要部分构成

① 后改名为 Xoom。——译者注

用户关心的是应用，Android发行时就附带了很多有用的核心应用，包括浏览器（Browser）、联系人（Contacts）和电话（Phone）。所有的应用都是用Java语言编写的。应用构成了Android架构的最顶层。

直接支撑应用层的是*应用框架*（application framework），这是一组用于构建应用的高层构件。应用框架是预装在Android设备中的，其中包含以下组件。

- Activity管理器（Activity Manager）：该组件用于管理所有Activity的生命周期，并维护一个用于实现应用内部和应用之间切换Activity的共享活动栈。本章稍后会详细讨论。
- 内容提供器（Content Providers）：这些组件将数据（例如浏览器应用中的书签）封装成能在应用间共享的形式。
- 位置管理器（Location Manager）：该组件使Android能获取设备的物理位置。
- 通知管理器（Notification Manager）：该组件使应用可以将重要的事件（例如有新短消息）在状态栏中显示出来且不会打断用户当前的活动。
- 包管理器（Package Manager）：该组件使应用可以知道设备上安装的其他包的情况。（本章稍后会讨论应用包。）
- 资源管理器（Resource Manager）：该组件使应用可以访问各种资源，稍后会在攻略1-5中讨论该主题。
- 电话管理器（Telephony Manager）：该组件使应用可以知道设备的电话服务状态。它还负责电话的拨打和接听。
- 视图系统（View System）：该组件负责管理用户界面元素和生成面向用户界面的事件。（在攻略1-5中会简要探讨这些内容。）
- 窗口管理器（Window Manager）：该组件将屏幕上的元素组织到窗口中，分配绘图界面，同时执行其他窗口相关任务。

应用框架中的组件在完成各自任务时都要依赖一些C/C++库。开发人员通过应用框架中的API与下面这些库交互。

- FreeType：这个库用于支持点阵字和向量字的渲染。
- libc：这是一个源自BSD系统的标准C系统库的实现，该C库针对嵌入式Linux设备进行了优化。
- LibWebCore：这个库为Android浏览器和嵌入式Web视图提供了现代化的高速Web浏览器引擎。它是基于WebKit的(http://en.wikipedia.org/wiki/WebKit)，Google的Chrome和苹果的Safari浏览器使用的也是WebKit。
- 媒体框架（Media Framework）：这些基于PacketVideo的OpenCORE的库支持多种流行的影音格式的播放和录制，还能处理静态图片文件。支持的格式包括MPEG-4、H.264、MP3、AAC、AMR、JPEG和PNG。
- OpenGL | ES：这是Android基于OpenGL | ES 1.0 API的OpenGL实现的3D绘图库。它可以使用硬件3D加速（如果能用的话）或是内建的（经过高度优化的）3D软件光栅器。

- SGL：这个库提供了底层的 2D 绘图引擎。
- SQLite：这个库提供了一个功能强大的轻量级关系数据库引擎，可供所有应用使用。Mozilla 的 Firefox 和苹果的 iPhone 也用这个引擎实现持久化存储。
- SSL：这个库提供了基于 SSL（安全套接层）的网络通信安全机制。
- 外观管理器（Surface Manager）：这个库负责管理显示子系统的访问方式并将各种应用的 2D 和 3D 图形层无缝地组合在一起。

Android 运行时环境由核心库（Apache Harmony Java 第 5 版实现的一个子集）和 Dalvik 虚拟机组成。Dalvik 不是基于栈的，而是一个基于寄存器的非 Java 虚拟机。

注意 Google 的 Dan Bornstein 创造了 Dalvik，他用祖辈曾居住的一个冰岛渔村的名字命名了这个虚拟机。

每个 Android 应用都默认在它自己的 Linux 进程中运行，每个进程都包含一个 Dalvik 的实例。虚拟机使得设备能高效地运行多个虚拟机。这种高效很大程度上是因为 Dalvik 执行的是 DEX（Dalvik Executable）文件——一种针对小内存设备优化过的格式。

注意 应用中的任何部分要运行时，Android 都会启动一个进程。当应用不再需要该进程且有其他应用需要系统资源时，Android 就会将其关闭。

读者可能会觉得奇怪，非 Java 的虚拟机怎么能运行 Java 代码？答案是，Dalvik 并不运行 Java 代码。实际上，Android 将编译好的 Java 类文件转换成 DEX 格式，Dalvik 执行的是由此产生的代码。

库和 Android 运行时都依赖 Linux 内核（2.6 版）以实现底层的核心服务，例如线程、底层内存管理、网络栈、进程管理和驱动模型。此外，内核在硬件层和软件栈之间起到了抽象层的作用。

Android 安全模型

Android 的架构中还包含了一个安全模型，它能阻止应用执行对其他应用、Linux 或用户有害的操作。该安全模型将进程放入安全沙盒，基本上是利用标准的 Linux 功能（例如用户 ID 和组 ID）实现了进程级的安全机制。

默认情况下，应用在沙盒中不能读写用户的私人数据（例如联系人和电子邮件）和其他应用的文件，不能访问网络，不能保持设备唤醒，也无法使用摄像头等。应用必须要先获得相应的权限才能访问网络或是执行其他的敏感操作。

Android 有多种处理权限请求的方法，通常是根据证书自动允许或拒绝请求，或是提示用户授予或撤销权限。应用所需的权限在应用的 Manifest 文件（本章稍后会讨论）中声明，这样在安装应用时 Android 就会获悉应用所需的权限。在这之后，应用的权限不能再改变。

1.4 应用架构

Android应用的架构与桌面应用程序的架构不同。Android应用架构是基于组件的，这些组件存放在应用包中，组件之间的通信是通过在Manifest文件中描述的Intent实现的。

1.4.1 组件

应用就是运行在一个Linux进程中、由Android负责管理的若干组件（Activity、Service、Content Provider和Broudcast Receiver）的集合。这些组件共享一些资源，包括数据库、偏好设置、文件系统和Linux进程。

注意 并不是每个应用都包含所有组件。例如，某个应用可以只有Activity，而另一个应用则可能由几个Activity和一个Service组成。

面向组件的架构使得应用可以在其他应用允许的情况下重用其他应用的组件。组件重用降低了整体的内存需求，这对于内存受限的设备是非常重要的。

具体来说说“重用”这个概念。假设你要构建一个用户能从调色板中选取颜色的绘画应用，而另一个应用中已经开发了一个适用的颜色选取器，而且是允许重用的。在这种情况下，这个绘画应用就可以调用另一个应用中的颜色选取器，而不必自己开发。该绘画应用无需包含另一个应用的颜色选取器，甚至不必链接这个应用，仅在需要时启动另一个应用的颜色选取器即可。

当绘图应用需要另一个应用的某个部分（例如前面提到的颜色选取器）时，Android就会启动一个进程并且实例化这部分的Java对象。这就是Android应用的入口点（如没有C语言的`main()`函数）不唯一的原因。相反，在应用需要的时候，其所需的组件会被实例化并运行。

1. Activity

Activity组件负责呈现用户界面，实现用户和应用间的交互。例如，Android的联系人应用就有一个用于添加新联系人的Activity，电话应用含有一个用户拨号的Activity，计算器应用则包含了一个能执行各种基本运算的Activity（参见图1-2）。

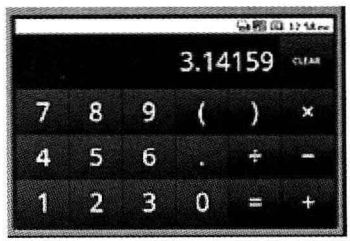


图1-2 Android的计算器应用的主Activity能让用户执行各种基本运算

尽管一个应用可以只含有一个Activity，但通常每个应用都会有多个Activity。例如，计算器还有一个让用户计算平方根、处理三角学问题和执行其他高级数学运算的“高级面板”Activity。

2. Service

Service组件长时间在后台运行，没有用户界面。跟Activity相比，Service运行在进程的主

线程上，它必须产生新的线程来执行耗时的操作。Service可以分为本地服务和远程服务。

- 本地服务（local service）跟应用的其他部分运行在同一个进程中。这类Service可以让应用方便地实现后台任务。
- 远程服务（remote service）运行在单独的进程中。这类service可以实现进程间通信。

注意 尽管可以指定Service运行在单独的进程中，但Service本身并不是一个单独的进程，也不是一个线程。相反，Service只是让应用告诉Android它想要在后台做一些事情（甚至在用户没有直接跟这个应用交互时也要执行），同时也可以让应用将部分功能提供给其他应用。

想象这样一个Service，当用户在Activity窗口中选择音乐后，就会播放该音乐。用户通过Activity选择歌曲，而Service则对这个选择作出响应。Service在另一个线程中播放音乐，从而避免了出现“Application Not Responding”对话框（在附录C中详述）。

注意 用Service播放音乐的原因是用户希望音乐一直播放，甚至在退出启动音乐播放的Activity后也不停止。

3. Broadcast Recevier

Broadcast Recevier组件负责接收和响应广播。很多广播都源自系统代码，例如告知用户时区改变或是电池电量低。

应用也可以发出广播。例如，应用在完成一些数据的网络下载后可以通知其他应用：现在轮到它们使用网络了。

4. Content Provider

Content Provider组件将应用中特定的数据提供给其他应用。数据可以存放在Android文件系统、SQLite数据库或其他任何地方。

Content Provider更适合直接访问原始数据，因为它分离了组件代码和原始数据格式。这种分离可以防止在数据格式变化时破坏代码。

1.4.2 Intent

Intent是用于描述要执行的操作（例如，发送电子邮件和选择图片）或是广播已发生的（例如设备的摄像头被激活了）和要声明的外部事件的消息。

Android中几乎所有事情都涉及Intent，所以有很多机会可以将已有的组件替换成你自己的组件。例如，Android提供了用于发送电子邮件的Intent。应用可以发送这个Intent激活标准的电子邮件应用，也可以另行注册一个Activity来响应“发送电子邮件”的Intent，用定制应用替换标准的电子邮件应用。

这些消息都是以`android.content.Intent`类的实例形式实现的。`Intent`对象描述的消息包括以下内容。

- Action（动作）：用于描述Intent执行动作名的字符串，在Broadcast Intent中则是所播报