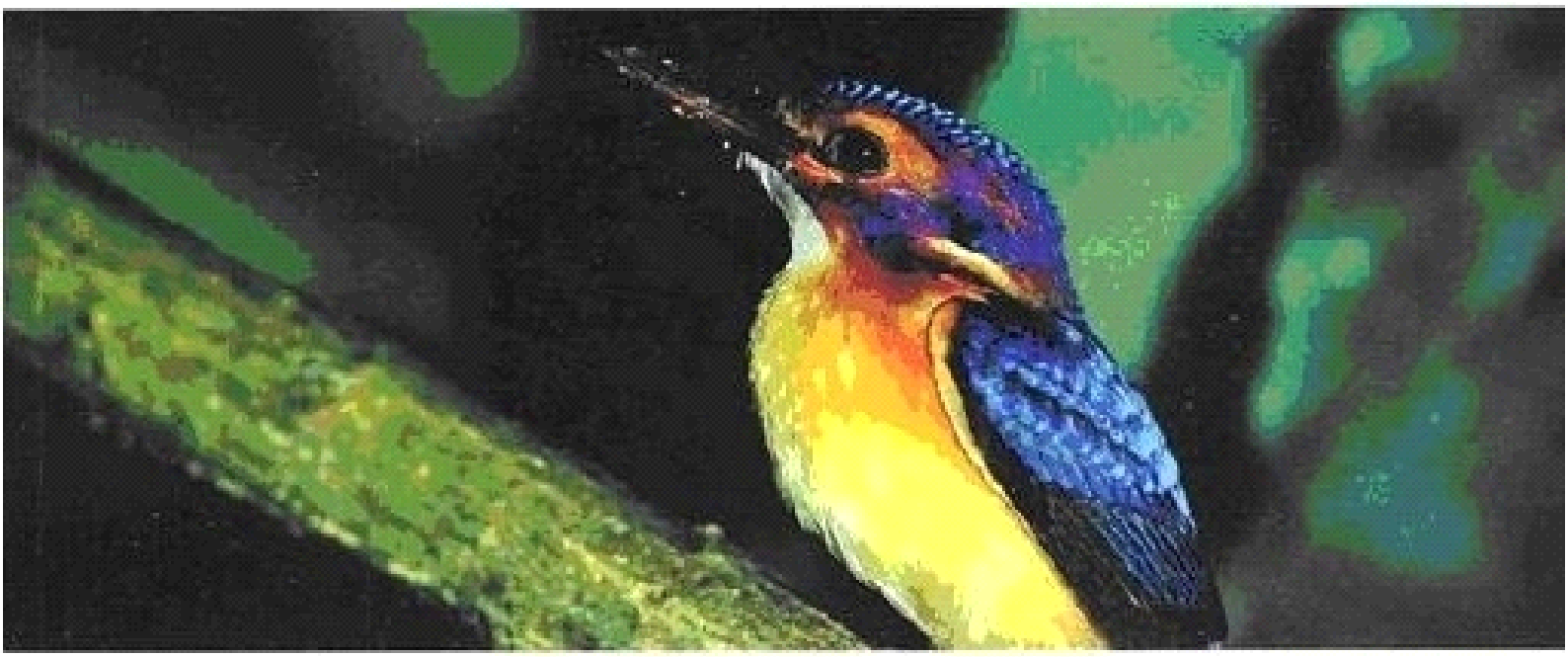
国内首本基于Android 2.0的经典著作，5大专业社区一致鼎力推荐！

揭秘系列丛书 UNLEASH

Android Unleashed

Android

应用开发揭秘



杨丰盛◎著

机械工业出版社
China Machine Press

作者简介

杨丰盛，Android 应用开发先驱，对 Android 有深入研究，实战经验极其丰富。精通 Java、C、C++等语言，专注于移动通信软件开发，在机顶盒软件开发和 MTK 平台软件开发方面有非常深厚的积累。

2007 年获得中国软件行业协会游戏软件分会(CGIA)认证及国际游戏开发教育联合会国际认证。曾经领导和参与《三国群英传说》、《大航海传奇》、《美少女养成计划》等经典游戏的开发。

编辑推荐

本书内容全面，不仅详细讲解了 Android 框架、Android 组件等基础知识，而且还深入阐述了传感器、语音识别、桌面组件开发等高级知识，最重要的是还全面介绍了如何利用原生的 C/C++（NDK）和 Python、Lua 等脚本语言来开发 Android 应用。

本书实战性强，书中的每个知识点都有配精心设计的示例。

本书简介

国内第一本基于 Android 2.0 的经典著作，5 大专业社区联袂推荐，权威性毋庸置疑！

本书内容全面，不仅详细讲解了 Android 框架、Android 组件、用户界面开发、游戏开发、数据存储、多媒体开发和网络开发等基础知识，而且还深入阐述了传感器、语音识别、桌面组件开发、Android 游戏引擎设计、Android 应用优化、OpenGL 等高级知识，最重要的是还全面介绍了如何利用原生的 C/C++（NDK）和 Python、Lua 等脚本语言（Android Scripting Environment）来开发 Android 应用；本书实战性强，书中的每个知识点都有配精心设计的示例，尤为值得一提的是，它还以迭代的方式重现了各种常用的 Android 应用和经典 Android 游戏的开发全过程，既可以以它们为范例进行实战演练，又可以将它们直接应用到实际开发中去。

本 书 赞 誉

20 世纪 90 年代初，裘伯君、鲍岳桥等 IT 行业的前辈“单打独斗”就能开发出脍炙人口的应用，如今做一个项目动辄就需要数百人的大规模团队和千万级的巨额开发费用，程序员真的没有了展露个人才华和创意的机会吗？我们正站在移动技术改变人类生活方式的十字路口，而移动互联网正是这一切的关键。Android 以全新的开放平台和全球化的市场，为小团队提供了一个充分展现自己的舞台。本书为所有 Android 开发者提供了绝佳的参考，不可不读！

—Android 中文站（http://www.androidin.com/）

与已经出版的所有同类书相比，本书内容更全面，几乎涵盖了 Android 开发的所有方面；实战性更强，不仅各个知识点都有翔实的范例，而且还包含多个实用的完整案例；主题更新颖，Android 2.0 中的各种最新特性一览无余……本书值得各种水平层次的 Android 应用开发者阅读，强烈推荐！

—Android 中文用户组

随着 3G 技术的成熟和智能手机的不断普及，移动应用的需求与日俱增，移动应用开发成为当下最热门的技术之一。在 Google 和 Android 手机联盟的共同推动下，Android 在众多移动应用开发平台中脱颖而出。本书的出版对于广大 Android 应用开发者来说不啻是一种福音，它将为 Android 开发者社区注入强大的活力！

—毕惠子 Android 实验室（http://www.androidlab.cn/）

随着移动智能设备的普及，我国移动应用的需求即将迎来“井喷”，本书能让你轻松转型为 Android 开发者，助你笑傲移动应用开发之巅。极力推荐！

—谷奥（http://www.google.org.cn/）

这是一本参考手册，内容的完整性和系统性几乎无可挑剔，可作为广大 Android 开发者的案头必备书；这是一部权威指南，基础知识部分翔实而丰富，高级知识部分深入且饱含最佳实践，能从本质上提升开发者对 Android 的理解和开发水平。尤为值得一提的是，Android 2.0 中新增了大量激动人心的新特性，不仅支持多点触摸设备、软键盘，而且还支持多账户在线管理、蓝牙……作为国内第一本基于 Android 2.0 的著作，本书可谓极具前瞻性，第一时间将这些新特性完美地呈现给了广大读者。

—安卓网（http://www.hiapk.com/）

前　言

3G 牌照在国内发放后，3G、Andriod、iPhone、Google、苹果、手机软件、移动开发等词越来越充斥于耳。随着 3G 网络的大规模建设和智能手机的迅速普及，移动互联网时代已经微笑着迎面而来。

以创新的搜索引擎技术而一跃成为互联网巨头的 Google，无线搜索成为 Google 进军移动互联网的一块基石。早在 2007 年，Google 中国就把无线搜索当作战略重心，不断推出新产品，尝试通过户外媒体推广移动搜索产品，并积极与运营商、终端厂商、浏览器厂商等达成战略合作。

Android 操作系统是 Google 最具杀伤力的武器之一。苹果以其天才的创新，使得 iPhone 在全球迅速拥有了数百万忠实“粉丝”，而 Android 作为第一个完整、开放、免费的手机平台，使开发者在为其开发程序时拥有更大的自由。与 Windows Mobile、Symbian 等厂商不同的是，Android 操作系统免费向开发人员提供，这样可节省近三成成本，得到了众多厂商与开发者的拥护。最早进入

Andriod 市场的宏达电已经陆续在一年内推出了 G1、Magic、Hero、Tattoo 等 4 款手机，三星也在近期推出了 Galaxyi7500，连摩托罗拉也推出了新款 Andorid 手机 Cliq，中国移动也以 Android 为基础开发了 Ophone 平台。这些发展证明 Android 已经成为智能手机市场的重要发展趋势。

从技术角度而言，Android 与 iPhone 相似，采用 WebKit 浏览器引擎，具备触摸屏、高级图形显示和上网功能，用户能够在手机上查收电子邮件、搜索网址和观看视频节目等。Android 手机比 iPhone 等其他手机更强调搜索功能，界面更强大，可以说是一种融入了全部 Web 应用的平台。Android 的版本包括 Android 1.1、Android 1.5、Android 1.6，Android 2.0 刚发布不久。随着版本的更新，从最初的触屏到现在的多点触摸，从普通的联系人到现在的数据同步，从简单的 Google Map 到现在的导航系统，从基本的网页浏览到现在的 HTML 5，这都说明 Android 已经逐渐稳定，而且功能越来越强大。此外，Android 平台不仅支持 Java、C、C++等主流的编程语言，还支持 Ruby、Python 等脚本语言，甚至 Google 专为 Android 的应用开发推出了 Simple 语言，这使得 Android 有着非常广泛的开发群体。

我们都知道，无论是产品还是技术，商业应用是它最大的发展动力。Android 如此受厂商与开发者的青睐，它的前景一片光明。伴随着装有 Android 操作系统的移动设备的增加，基于 Android 的应用需求势必也会增加。Android 作为新的平台、新的技术，国内目前介绍其技术的书籍甚少，不能满足各个层次的开发者，为了帮助众多开发人员和爱好者进入移动互联网领域，并提高程序开发水平，笔者写作了本书。

本书面向的读者

阅读本书的唯一条件是具有一定的 Java 基础，当然扩展篇可能会涉及 C、C++和脚本语言的知识。

本书面向的读者群包括毫无 Android 开发经验的初学者，以及有一定的 Android 开发经验但缺乏系统学习的开发人员。

如何阅读本书

本书从基础入手，循序渐进地讲述了 Android 的主要功能和用法，使读者对其有完整的认识，掌握其结构框架。同时，从实战的角度出发，通过大量的示例程序，让读者边学习边实践，更深刻地理解 Android 系统的优点所在。

另外，本书为每个功能和知识点都提供了一个示例程序，可操作性极强，建议在阅读书本书的同时，一定要结合本书所附带的示例程序（完整的示例程序源代码可登录华章网站 www.hzbook.com 下载）。本书所附的示例程序都是基于最新的 Android 2.0 的 SDK，源代码目录结构如图 1 所示，章节中每一个示例，都

可以根据所在的章节及所指定的项目名称在所附源代码中找到对应的项目文件夹。每个项目文件夹都按如图 2 所示的目录结构来存放项目所需的所有源文件。

在安装了 Android 开发环境之后，可以直接将 Android 项目导入到 Eclipse 中，步骤如下：

首先，启动 Eclipse，选择“File”→“Import...”菜单，展开“General”项，选择“Existing Projects into Workspace”导入项目到工作区，如图 3 所示。

然后，点击“Next”按钮，进入选择项目文件目录，如图 4 所示，选择好项目目录后，点击“Finish”按钮，等待导入完成即可。如果需要将项目文件一起拷贝到工作区，就需要在图 4 的界面上选择“Copy projects into workspace”复选框。

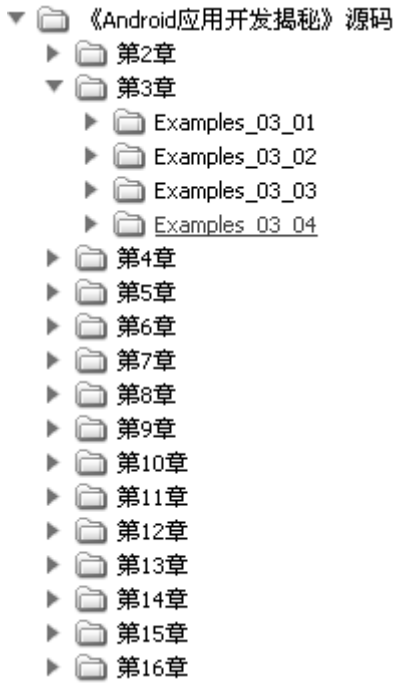


图 1 源码结构图

图 2 项目结构图

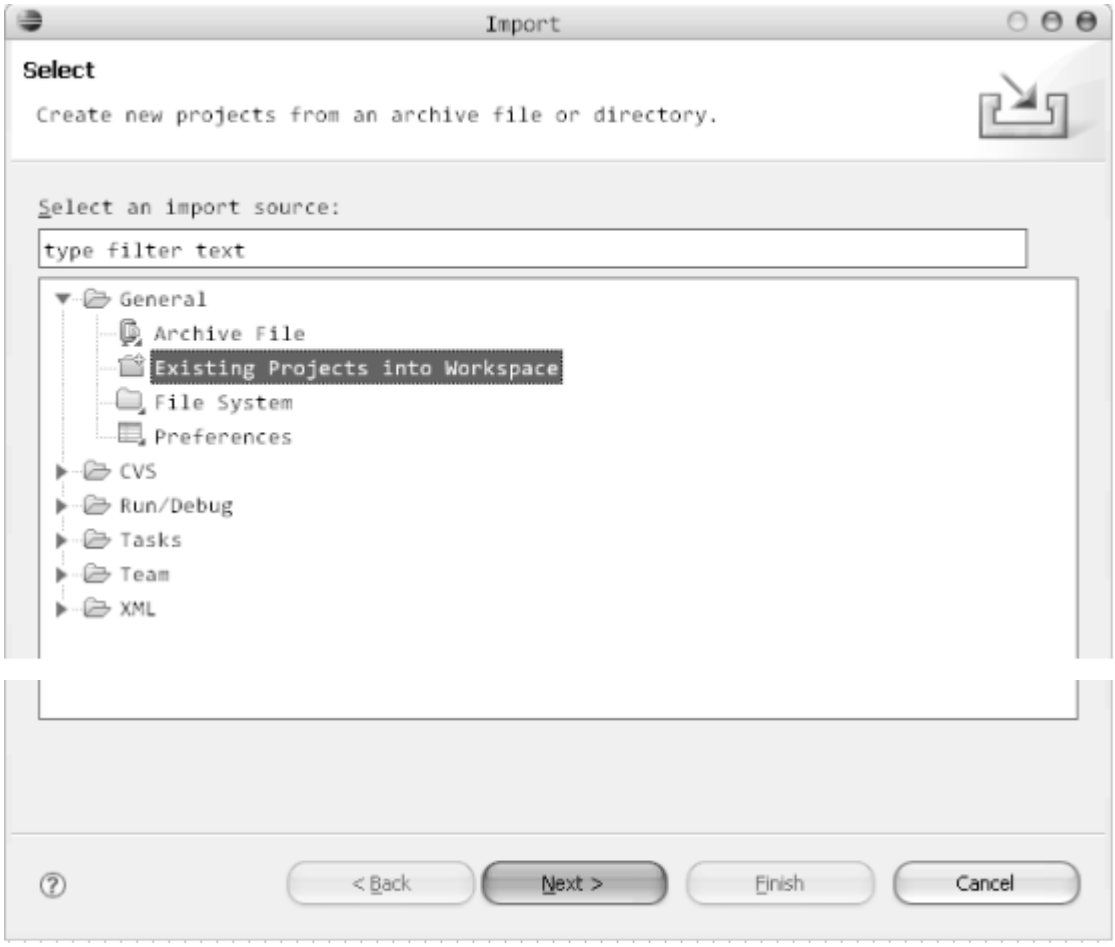

Import
Select
Create new projects from an archive file or directory.
Select an import source:
type filter text
General
Archive File
Existing Projects into Workspace
File System
Preferences
CVS
Run/Debug
Tasks
Team
XML
< Back
Next >
Finish
Cancel

图 3 导入项目到工作区

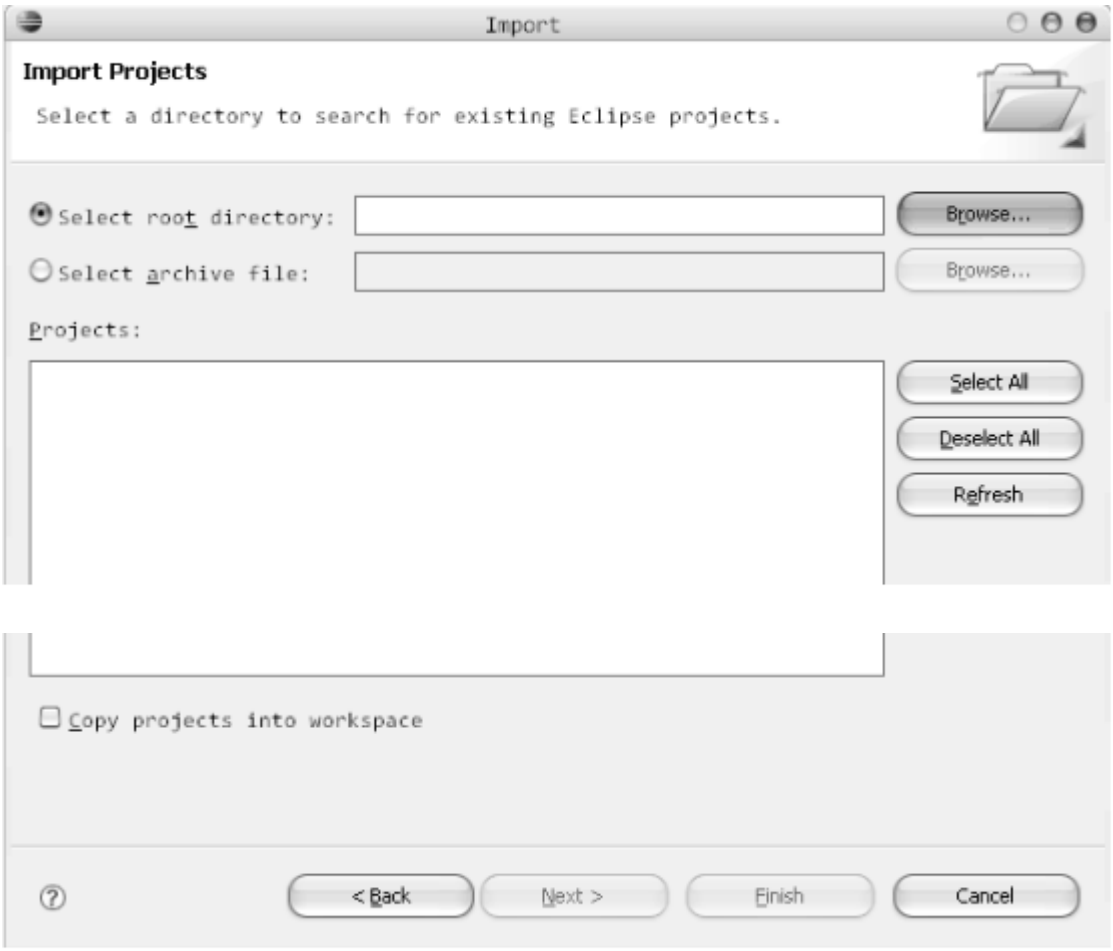


图 4 选择项目文件

致 谢

感谢所有在本书写作过程中给予我指导、帮助和鼓励的朋友，尤其是本书的策划编辑杨福川，他不仅对本书提出了宝贵的写作建议，而且还和他的同事曾珊对书稿进行了仔细的审阅。

感谢一直以来信任、鼓励、支持我的父母和亲人。

最后还要感谢我的女友，正是她的爱与支持，才使我有了今天的收获。

虽然我们热切地希望与广大读者朋友分享使用 Android 平台的应用开发经验，但由于时间有限，书中难免存在疏漏与错误，诚恳地希望各位读者批评、指正。如果你发现书中有任何问题，抑或是想和本书的作者和读者交流关于 Android 开发中的相关话题，你可以申请加入华章俱乐部，这里不仅有技术专家，还有很多志同道合的朋友，大家共同进步吧！

目　　录

第一部分　准备篇

第 1 章　Android 开发简介

在 Google 及其开放手机联盟推出基于 Linux 平台的开源手机操作系统 Android 之后，Google 又不惜重金举办了 Android 开发者大赛，吸引了众多开发者的目光。Android 不仅功能强大，而且具有开放和免费等先天优势，全球范围内的电信行业、手机制造商因此毫不犹豫地加入到 Android 开放手机联盟中来。2008 年 9 月 22 日，美国运营商 T-Mobile USA 在纽约正式发布了第一款基于 Android 的手机--T-Mobile G1。这让更多的移动设备厂商看到了 Android 的光明前景，并纷纷加入其中，Android 甚至已经涉足上网本市场。中国移动也在 Android 的基础之上推出了自己的操作系统 OMS，而基于 OMS 操作系统的联想 O1 手机也即将上市，2009 年年底将会有更多的 Android 手机出现。

随着 Android 手机的普及，Android 应用的需求势必会越来越大，这将是一个潜力巨大的市场，会吸引无数软件开发厂商和开发者投身其中。作为程序员的我们，当然也不应该落后于人，赶快加入到 Android 应用的开发阵营中来吧！

1.1　Android 基本概念

Android 一词本意是指“机器人”，当然现在大家都知道它是 Google 推出的开源手机操作系统。Android 基于 Linux 平台，由操作系统、中间件、用户界面和应用软件组成，号称是首个为移动终端打造的真正开放和完整的移动软件。它是由一个由 30 多家科技公司和手机公司组成的“开放手机联盟”共同研发的，这将大大降低新型手机设备的研发成本。完全整合的全移动功能性产品成为“开放手机联盟”的最终目标。

1.1.1　Android 简介

Android 作为 Google 移动互联网战略的重要组成部分，将进一步推进“随时随地为每个人提供信息”这一企业目标的实现。Google 的目标是让移动通信不依赖于设备，甚至是平台。出于这个目的，Android 将完善而不是替代 Google 长期以来推行的移动发展战略：通过与全球各地的手机制造商和移动运营商成为合作伙伴，开发既实用又有吸引力的移动服务，并推广这些产品。

Android 平台的研发队伍阵容强大，包括 Google、HTC（宏达电）、T-Mobile、高通、摩托罗拉、三星、LG 以及中国移动在内的 30 多家企业都将基于该平台开发手机的新型业务，应用之间的通用性和互联性将在最大程度上得到保持。“开放手机联盟”表示，Android 平台可以促使移动设备的创新，让用户体验到最优质的移动服务。同时，开发商也将得到一个新的开放级别，更方便地进行协同合作，从而保障新型移动设备的研发速度。因此 Android 是第一个完整、开放、免费的手机平台。下面我们来欣赏一下第一款基于 Android 操作系统的手机 G1，外观相当漂亮，如图 1-1 所示。

Android 系统具有如下 5 个特点：

* 开放性。Google 与开放手机联盟合作开发了 Android，Google 通过与运营商、设备制造商、开发商和其他有关各方结成深层次的合作伙伴关系，希望通过建立标准化、开放式的移动电话软件平台，在移动产业内形成一个开放式的生态系统。
* 应用程序无界限。Android 上的应用程序可以通过标准 API 访问核心移动设备功能。通过互联网，应用程序可以声明它们的功能可供其他应用程序使用。
* 应用程序是在平等的条件下创建的。移动设备上的应用程序可以被替换或扩展，即使是拨号程序或主屏幕这样的核心组件。
* 应用程序可以轻松地嵌入网络。应用程序可以轻松地嵌入 HTML、JavaScript 和样式表，还可以通过 WebView 显示网络内容。
* 应用程序可以并行运行。Android 是一种完整的多任务环境，应用程序可以在其中并行运行。在后台运行时，应用程序可以生成通知以引起注意。

为什么 Android 手机如此受用户青睐，下面我们来看看 Android 究竟有些什么功能在吸引着我们。

（1）智能虚拟键盘。虚拟键盘的出现意味着基于 Android 1.5 或以上版本（Android 2.0）的移动设备可以同时支持物理键盘和虚拟键盘。不同的输入方式可满足用户在特定场景的需求。Android 虚拟键盘可以在任何应用中提供，包括 Gmail、浏览器、SMS，当然也包括大量的第三方应用，如自动校正、推荐、用户词典等。不同于其他手机平台，Android 1.5 及其以上的版本还支持第三方虚拟键盘应用的安装，如图 1-2 所示。

（2）使用 Widget 实现桌面个性化。可以用 Widget“武装”自己的桌面。大多数小的 Web 应用都是从网络上获得实时数据并展示给用户的。Android 预装了 5 个桌面 Widget，包括数字时钟、日历、音乐播放器、相框和搜索。不同于 iPhone，Android 通过内置的应用程序库安装第三方 Widget，如图 1-3 所示。

图 1-1 Android G1

图 1-2　虚拟键盘

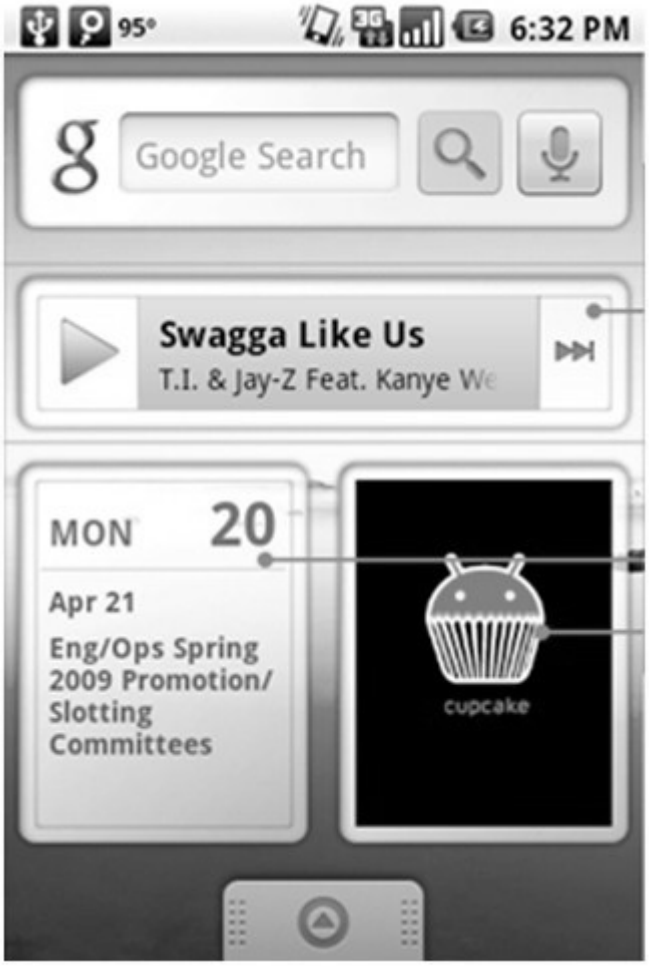


图 1-3　用 Widget 实现个性化桌面

（3）用在线文件夹快速浏览在线数据。类似于 OS X Leopard 的 QuickLook 特征，Android 的在线文件夹可显示常见的数据条目，比如联系人、喜欢的应用、E-mail 信息、播放列表、书签、RSS 源等，并不需要运行系统程序处理特定的数据条目。在线文件夹数据实时更新，就像通过云或是本地创建新的数据。什么是最好的，开发者可以拓展通用数据条目和注册新数据类型的内置支持。例如，Twitter 客户端程序可以注册 tweet 作为新数据类型，因此可以让你从你的朋友那里创建 tweet 的在线文件。Android 可以为我们的个人桌面提供一组在线文件夹，从而帮助我们快速、方便地浏览联系人、股市、书签等信息。

（4）视频录制和分享。Android 还有录制和分享视频的功能，对回放和 MPEG-4、3GP 等视频格式也有了更好的支持。可以通过 E-mail、MMS 或直接上传到 YouTube 等方式来分享视频，使用隐私控制来决定是分享给朋友还是每个人。上传视频的同时，可以继续使用手机，甚至可以继续录制和上传新的视频。如图 1-4 所示，通过 YouTube 分享录制的视频。

（5）图片上传。在线分享图片需要的点击更少。完成照相后，当浏览图片或选择 Google 在线图片服务 Picasa 时，只需轻点“分享”就会拥有 1GB 的免费图片存储空间。

（6）更快、更兼容的浏览器。Android 的基于 Webkit 内核的浏览器带来了重要的调速装置（SpeedPumb），这得益于新的 Webkit 渲染引擎和优化的 Java 脚本编译器（SquireIFish）。当使用包含大量 Java 脚本的复杂 Web 应用时，可以体验到更佳的性能。除提高速度外，Android 的浏览器还支持 Web 页面内的复制和粘贴操作，用户可以选中文本并复制，然后粘贴到搜索框中进行搜索。

（7）Voice Search 语音搜索。带有语音识别技术的 Google 手机已于 2008 年 11 月面世，它支持语音搜索功能。该功能增强了默认的搜索能力，已超过纯文本搜索。当你大声说出要搜索的内容后，Android 将上传数字信号并记录到 Google 服务器中。在服务器中，语音识别技术能将语音转化为特定的文本搜索，使之通过 Google 搜索引擎，通过地理位置的筛选，将结果反馈到手机设备。图 1-5 显示了 Google 文本和语音搜索桌面。

（8）立体声蓝牙和免提电话。除了增强的免提电话体验，Android 还支持立体声蓝牙（A2DP 和 AVCRP），并有自动配对功能。

（9）强大的 GPS 技术。Android 内部提供了大量 GPS 组件，我们可以很轻松地获得设备当前的位置等信息，让导航等功能更加完美。

（10）Android 系统硬件检测。Android 可自动检测和修复 SD 卡的文件系统，允许第三方应用显示 Android 系统的硬件特征。为了让用户下载到与自己的设备更匹配的应用，我们可以检测用户设备的硬件信息，让满足应用要求的设备安装该程序，当更多的 Android 设备建立在不同的硬件上时，这个功能会显得很实用。

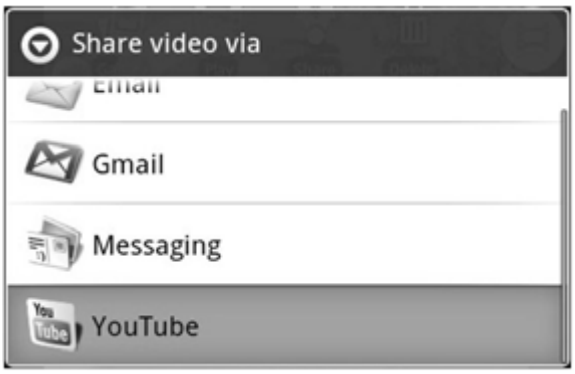


图 1-4 通过 YouTube 分享录制的视频

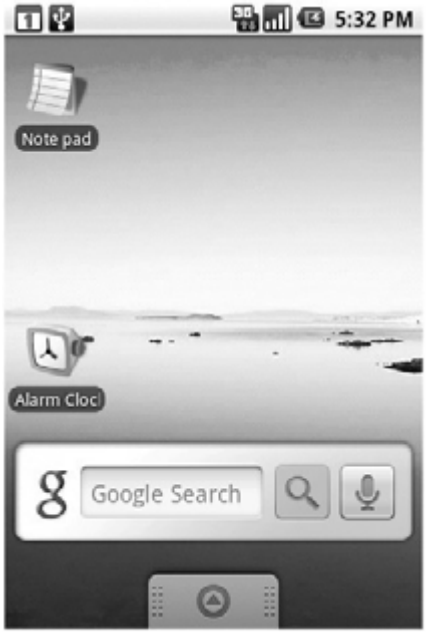


图 1-5 Google 文本和语音搜索桌面

1.1.2 Android 的系统构架

通过上一节的介绍，我们对 Android 的特点以及它为什么会如此受欢迎有了初步的了解。下面将讨论 Android 的系统架构，我们先来看看 Android 的体系结构，如图 1-6 所示。

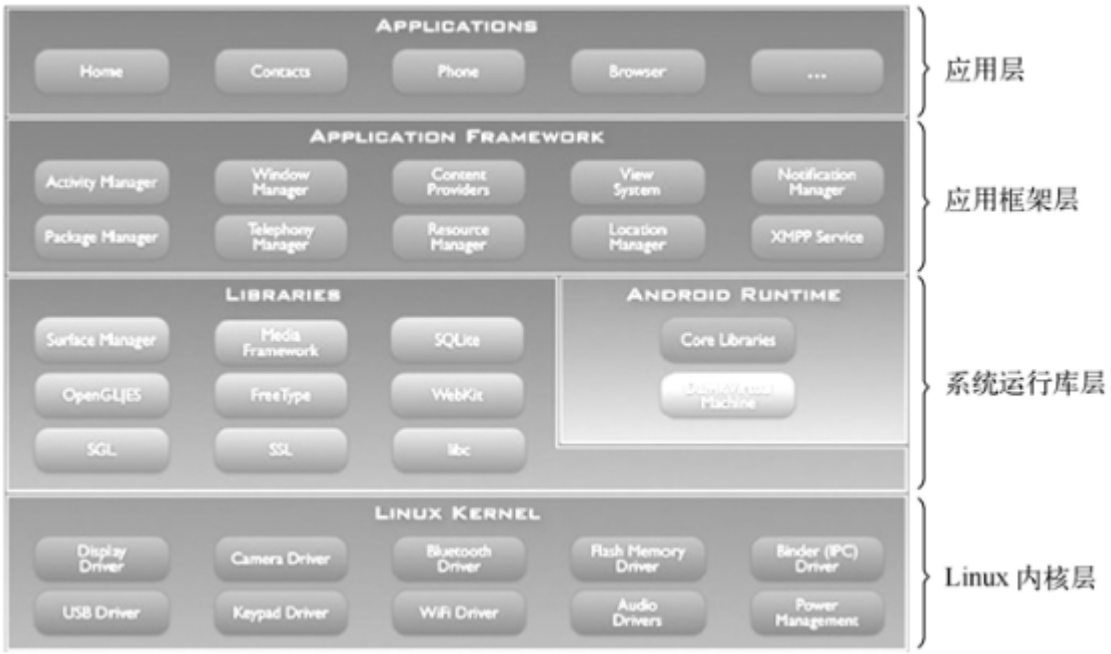


图 1-6　Android 系统结构图

从图 1-6 可以看出 Android 分为 4 层，从高到底分别是应用层、应用框架层、系统运行库层和 Linux 内核层。下面将对这 4 层进行简要的分析和介绍。

1．应用层

应用是用 Java 语言编写的运行在虚拟机上的程序，如图 1-6 中最上层部分所示。其实，Google 最开始时就在 Android 系统中捆绑了一些核心应用，比如 E-mail 客户端、SMS 短消息程序、日历、地图、浏览器、联系人管理程序，等等。

2．应用框架层

这一层是编写 Google 发布的核心应用时所使用的 API 框架，开发人员同样可以使用这些框架来开发自己的应用，这样便简化了程序开发的架构设计，但是必须遵守其框架的开发原则。

从图 1-6 中可以看出，Android 提供了如下一些组件。

* 丰富而又可扩展的视图(View)：可以用来构建应用程序，它包括列表(List)、网格（Grid）、文本框（Text Box）、按钮（Button），以及可嵌入的 Web 浏览器。
* 内容提供器（Content Providers）：它可以让一个应用访问另一个应用的数据（如联系人数据库）， 或共享它们自己的数据。
* 资源管理器（Resource Manager）：提供非代码资源的访问，如本地字符串、图形和布局文件（Layout file）。
* 通知管理器 （Notification Manager）：应用可以在状态栏中显示自定义的提示信息。
* 活动管理器（Activity Manager）：用来管理应用程序生命周期并提供常用的导航退回功能。
* 窗口管理器（Window Manager）：管理所有的窗口程序。
* 包管理器（Package Manager）：Android 系统内的程序管理。

后面的章节将进一步介绍这些组件的使用。

3．系统运行库（C/C++库以及 Android 运行库）层

当使用 Android 应用框架时，Android 系统会通过一些 C/C++库来支持我们使用的各个组件，使其能更好地为我们服务。

* Bionic 系统 C 库：C 语言标准库，系统最底层的库，C 库通过 Linux 系统来调用。
* 多媒体库（MediaFramework）：Android 系统多媒体库，基于 PacketVideo OpenCORE，该库支持多种常见格式的音频、视频的回放和录制，以及图片，比如 MPEG4、MP3、AAC、AMR、JPG、PNG 等。
* SGL：2D 图形引擎库。
* SSL：位于 TCP/IP 协议与各种应用层协议之间，为数据通信提供支持。
* OpenGL ES 1.0：3D 效果的支持。
* SQLite：关系数据库。
* Webkit：Web 浏览器引擎。
* FreeType：位图（bitmap）及矢量（vector）。

每个 Java 程序都运行在 Dalvik 虚拟机之上。与 PC 一样，每个 Android 应用程序都有自己的进程，Dalvik 虚拟机只执行.dex 的可执行文件。当 Java 程序通过编译，最后还需要通过 SDK 中的 dx 工具转化成.dex 格式才能正常在虚拟机上执行。

Google 于 2007 年底正式发布了 Android SDK, 作为 Android 系统的重要特性，Dalvik 虚拟机也第一次进入了人们的视野。它对内存的高效使用，以及在低速 CPU 上表现出的高性能，确实令人刮目相看。Android 系统可以简单地完成进程隔离和线程管理。每一个 Android 应用在底层都会对应一个独立的 Dalvik 虚拟机实例，其代码在虚拟机的解释下得以执行。

很多人认为 Dalvik 虚拟机是一个 Java 虚拟机，因为 Android 的编程语言恰恰就是 Java 语言。但是这种说法并不准确，因为 Dalvik 虚拟机并不是按照 Java 虚拟机的规范来实现的，两者并不兼容。它们有两个明显的不同：Java 虚拟机运行的是 Java 字节码，而 Dalvik 虚拟机运行的则是其专有的文件格式为 dex（Dalvik Executable）的文件。在 Java SE 程序中的 Java 类会被编译成一个或者多个字节码文件（.class）然后打包到 jar 文件，而后 Java 虚拟机会从相应的 class 文件和 jar 文件中获取相应的字节码;Android 应用虽然也是使用 Java 语言进行编程，但是在编译成 class 文件后，还会通过一个工具（dx）将应用所有的 class 文件转换成一个 dex 文件，而后 Dalvik 虚拟机会从其中读取指令和数据。

Dalvik 虚拟机非常适合在移动终端上使用，相对于在桌面系统和服务器系统运行的虚拟机而言，它不需要很快的 CPU 计算速度和大量的内存空间。根据 Google 的测算，64MB 的内存已经能够让系统正常运转了。其中 24MB 被用于底层系统的初始化和启动，另外 20MB 被用于启动高层服务。当然，随着系统服务的增多和应用功能的扩展，其所消耗的内存也势必越来越大。归纳起来，Dalvik 虚拟机有如下几个主要特征：

（1）专有的 dex 文件格式。dex 是 Dalvik 虚拟机专用的文件格式，而为什么弃用已有的字节码文件（.class 文件）而采用新的格式呢？原因如下：

* 每个应用中会定义很多类，编译完成后即会有很多相应的 class 文件，class 文件中会有大量冗余信息，而 dex 文件格式会把所有的 class 文件内容整合到一个文件中。这样，除了减少整体的文件尺寸和 I/O 操作外，也提高了类的查找速度。
* 增加了对新的操作码的支持。
* 文件结构尽量简洁，使用等长的指令，借以提高解析速度。
* 尽量扩大只读结构的大小，借以提高跨进程的数据共享。

（2）dex 的优化。dex 文件的结构是紧凑的，但是如果还想运行时的性能有进一步提高，就需要对 dex 文件进一步优化。优化主要针对以下几个方面：

* 调整所有字段的字节序（LITTLE_ENDIAN）和对齐结构中的每一个域。
* 验证 DEX 文件中的所有类。
* 对一些特定的类和方法里的操作码进行优化。

（3）基于寄存器。相对于基于堆栈实现的虚拟机，基于寄存器实现的虚拟机虽然在硬件、通用性上要差一些，但是它在代码的执行效率上却更胜一筹。

（4）一个应用，一个虚拟机实例，一个进程。每一个 Android 应用都运行在一个 Dalvik 虚拟机实例中，而每一个虚拟机实例都是一个独立的进程空间。虚拟机的线程机制、内存分配和管理、Mutex 等的实现都依赖底层操作系统。所有 Android 应用的线程都对应一个 Linux 线程，虚拟机因而可以更多地依赖操作系统的线程调度和管理机制。不同的应用在不同的进程空间里运行，对不同来源的应用都使用不同的 Linux 用户来运行，可以最大程度地保护应用的安全和独立运行。

4. Linux 内核层

Android 的核心系统服务基于 Linux 2.6 内核，如安全性、内存管理、进程管理、网络协议栈和驱动模型等都依赖于该内核。Linux 内核同时也作为硬件和软件栈之间的抽象层。

Android 更多的是需要一些与移动设备相关的驱动程序，主要的驱动如下所示。

* 显示驱动（Display Driver）：基于 Linux 的帧缓冲（Frame Buffer）驱动。
* 键盘驱动（KeyBoard Driver）：作为输入设备的键盘驱动。
* Flash 内存驱动（Flash Memory Driver）：基于 MTD 的 Flash 驱动程序。
* 照相机驱动（Camera Driver）：常用的基于 Linux 的 v4l2（Video for Linux）驱动。
* 音频驱动（Audio Driver）：常用的基于 ALSA（Advanced Linux Sound Architecture）的高级 Linux 声音体系驱动。
* 蓝牙驱动（Bluetooth Driver）：基于 IEEE 802.15.1 标准的无线传输技术。
* WiFi 驱动：基于 IEEE 802.11 标准的驱动程序。
* Binder IPC 驱动：Android 的一个特殊的驱动程序，具有单独的设备节点，提供进程间通信的功能。
* Power Management（电源管理）：比如电池电量等。

1.1.3 Android 应用程序框架

上一节我们对 Android 的系统构架进行了详细剖析，Android 分为应用层、应用框架层、系统运行库层和 Linux 内核层。我们在开发应用时都是通过框架来与 Android 底层进行交互，接触最多的就是应用框架层了。

什么是应用程序框架呢？框架可以说是一个应用程序的核心，是所有参与开发的程序员共同使用和遵守的约定，大家在其约定上进行必要的扩展，但程序始终保持主体结构的一致性。其作用是让程序保持清晰和一目了然，在满足不同需求的同时又不互相影响。

Android 系统提供给应用开发者的本身就是一个框架，所有的应用开发都必须遵守这个框架的原则。我们在开发应用时就是在这个框架上进行扩展，下面来看看 Android 这个框架都有些什么功能可供我们使用。

* android.app：提供高层的程序模型和基本的运行环境。
* android.content：包含对各种设备上的数据进行访问和发布。
* android.database：通过内容提供者浏览和操作数据库。
* android.graphics：底层的图形库，包含画布、颜色过滤、点、矩形，可以将它们直接绘制到屏幕上。
* android.location ：定位和相关服务的类。
* android.media：提供一些类管理多种音频、视频的媒体接口。
* android.net ：提供帮助网络访问的类，超过通常的 java.net.* 接口。
* android.os ：提供了系统服务、消息传输和 IPC 机制。
* android.opengl：提供 OpenGL 的工具。
* android.provider：提供访问 Android 内容提供者的类。
* android.telephony：提供与拨打电话相关的 API 交互。
* android.view：提供基础的用户界面接口框架。
* android.util ：涉及工具性的方法，例如时间日期的操作。
* android.webkit ：默认浏览器操作接口。
* android.widget：包含各种 UI 元素（大部分是可见的）在应用程序的布局中使用。

1.2 OMS 介绍

OMS 是 Open Mobile System 的简称，即面向移动互联网的开放型移动智能终端软件平台，它包括基于 Linux 2.6 内核的移动终端下层操作系统、上层应用软件、中间件、Java 虚拟机、硬件参考设计以及基于 WebKit 的各类应用。它具有强大的兼容性、扩展性和安全性，以及简单易用、友好的人机界面等，而且具有完全自主的知识产权。在此之上，OMS 拥有开放统一的 API 开发接口、完备的集成开发环境和活跃的在线生态环境，极大地方便了移动应用的开发。

OMS 的可移植性将使该软件平台在其他领域具有广泛的应用，如航空航天、军事、制造业等。

1.2.1 OPhone 介绍

OPhone 是基于 Linux 的面向移动互联网的终端基础软件及系统解决方案。由于 OPhone 与 Android 兼容，都是基于 Java 开发的，因此可以同时用 OMS API

和 Android API 来开发 OMS 应用。任何用 Android API 开发的应用都可以在 OMS 终端上正确地运行。然而，不能在 Android 终端上运行由扩展的 OMS API 开发的程序，因为这些 OMS API 是 OMS 平台独有的，而且在运行时是必需的。

OPhone 是指采用了 OMS 智能操作系统的手机。为了突破 TD 终端瓶颈，以及促进手机终端与中国移动的网络和应用服务进行无缝对接，中国移动在 Android 操作系统基础上自主开发了 OMS 系统，该系统直接内置了中国移动的服务菜单、音乐随身听、手机导航、号簿管家、139 邮箱、飞信、快讯和移动梦网等特色业务，如图 1-7 所示。

1.2.2 Widget 介绍

OMS 除了支持基于 Java 的应用，还支持 Widget 应用开发。Widget 应用是 OMS 的精华，而 Android 从 1.5 版本开始同样支持 Widget 应用开发，但是所采用的标准则和 OMS 不同，我们会在后面的章节详细讲解。

Widget 应用采用了 JIL （Joint Innovation Lab）Widget 标准。JIL Widget 是一个采用 HTML、JavaScript 和 CSS 等网络技术的应用程序。Widget 应用是在 Widget 引擎上运行的独立的应用程序。Widget 已经成为手机上非常流行的技术，可以为用户带来良好的移动互联网体验，随时随地获取有用的资讯，如天气预报、股票信息、头条新闻等。从用户的角度来看，Widget 应用和 OPhone 应用没有什么区别。实际上，Widget 应用不同于 OPhone 应用。OPhone 应用是采用 Java 技术的应用程序，而 Widget 应用则是采用 HTML、JavaScript 和 CSS 等网络技术的应用程序。相比较而言，Widget 应用的开发更加方便快捷。此外，JIL Widget 还提供了许多 JavaScript API 来扩展 Widget 应用的能力，如访问手机电话本、手机文件系统等。Widget 应用运行效果如图 1-8 所示。

图 1-7 OPhone 系统界面

图 1-8 HelloWidget 效果预览

1.3 小结

本章主要介绍了与 Android 相关的一些基本概念，同时分析了 Android 系统的特点及其功能，简单介绍了目前主流的 7 个 Linux 平台手机以及中国移动的 OMS 操作系统。在介绍 Android 基本概念时重点介绍了 Android 系统架构和应用框架，其中应用层即是我们使用 Java 语言编写的一些运行在虚拟机上的程序，应用框架层是我们开发应用时接触最为紧密的一层，在开发应用程序时必须遵守其规则，才能保证所开发的应用程序能在 Android 上安全地运行。大家应着重理解这两层，这样才能开发出效率更高的应用程序。

据称今年底将会有 18 款之多的以 Android 平台为系统的 Linux 手机上市，Android 将在移动开发中具有更广阔的前景。要想成为一个优秀的 Android 手机开发者，还需要从基础做起，希望大家好好掌握本章的内容。

第 2 章　Android 开发环境搭建

本章讲解如何配置Android开发环境首先介绍Android开发所需要的开发包和工具，以及获得它们的方式；其次介绍如何正确安装和配置这些开发包；最后，为了测试安装的开发环境，创建了第一个 Android 项目——HelloAndroid，然后在模拟器上运行和调试该程序，并将该应用程序安装到 Android 手机上。

2.1 Android 开发准备工作

配置 Android 开发环境之前，首先需要了解 Android 对操作系统的要求。它可以使用 Windows XP 及其以上版本、Mac OS、Linux 等操作系统，本书以 Windows XP 为例进行讲解。Android 开发所需软件的版本及其下载地址如表 2-1 所示。

表 2-1　Android 开发所需软件的版本及其下载地址

软件名称	所用版本	下载地址
JDK	1.6	http://java.sun.com
Eclipse	Ganymede(3.4)	http://www.eclipse.org
Android SDK	Android SDK 2.0	http://developer.android.com/sdk/index.html
ADT	0.9.4	https://dl-ssl.google.com/android/eclipse/

2.2 开发包及其工具的安装和配置

Android 以 Java 作为开发语言，JDK 是进行 Java 开发时必需的开发包。Eclipse 是一款非常优秀的开源 IDE，在大量插件的“配合”下，完全可以满足从企业级 Java 应用到手机终端 Java 游戏的开发。Google 官方也提供了基于 Eclipse 的 Android 开发插件 ADT，所以本书选择 Eclipse 作为开发 IDE。

2.2.1 安装 JDK 和配置 Java 开发环境

很多人不能够很好地进行 Java 开发，原因就在于对 Java 运行环境不了解或是了解得不够透彻。如果连一个普通的 Java 程序运行环境都搭建不好，就更不

要说理解 J2EE、J2ME 以及本书所讲的 Android 等的运行环境了。因此，这里我们先讲如何安装 JDK 以及 Java 环境的配置，教大家搭建一个学习 Java 的基础平台，让大家少走一些弯路，多学到一些小窃门。

（1）登录 http://java.sun.com，下载最新版 JDK。

（2）安装 JDK，安装包中包含了 JDK 和 JRE 两部分，笔者建议将它们安装在同一个盘符下。双击安装程序，选择安装的目录，点击“下一步”，等待安装程序自动完成安装即可。

（3）右键单击“我的电脑”，选择“属性”菜单项，选择“高级”选项卡，选择“环境变量”，找到“Path”变量名（如果没有就新建一个名为“Path”的变量），点击“编辑”按钮，添加 JDK 安装目录中“bin”文件夹路径，如图 2-1 所示。然后点击“确定”按钮完成。再找到“ClASSPATH”变量（如果没有，同样可以新建），输入 JDK 安装目录中“lib”以及“demo”的路径，如图 2-2 所示，单击“确定”按钮完成。

图 2-1 “Path”变量配置

图 2-2 “ClASSPATH”变量配置

（4）安装配置完成之后，要测试是否安装成功。点击开始→运行，输入“CMD”，打开命令行模式。键入命令“java -version”，检测 JDK 是否安装成功，如果运行结果如图 2-3 所示，即表示安装成功。

图 2-3 “java -version”测试命令

2.2.2　Eclipse 的安装与汉化

Eclipse 的安装非常简单，直接将下载的压缩包解压即可。老版本的 Eclipse 的多国语言项目只更新到 3.2.1 版本，以后就再也没有更新了。Eclipse 最近发布了一个名为 Babel project 的项目，这个项目就是用来解决国际化的问题，旨在为每一个插件提供独立的语言包。这样，当做 RCP 项目的时候，根据需要对语言进行打包即可！

Babel 的安装方法和步骤如下所示：

（1）启动 Eclipse 开发工具，依次点击“Help ”　→选择“Software Update ...”菜单命令，打开“Software Updates and Add-ons”对话框，选择“Avaliable Software”项。接着点击“Add Site...”按钮，在“Location”文本框中输入 Babel 更新地址：http://download.eclipse.org/technology/babel/update-site/ganymede，然后点击 OK 按钮，如图 2-4 所示。

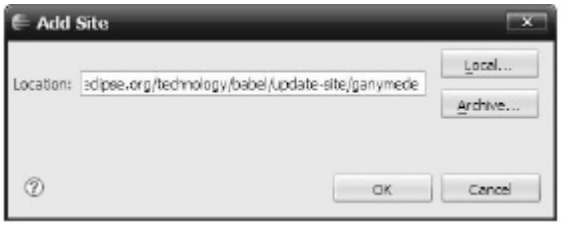


图 2-4　添加语言包更新地址

（2）“Avaliable Software”表中会多出一项 http://download.eclipse.org/technology/babel/update-site/ganymede/，点击该项左边的箭头，就会出现网络更新软件列表，如图 2-5 所示。

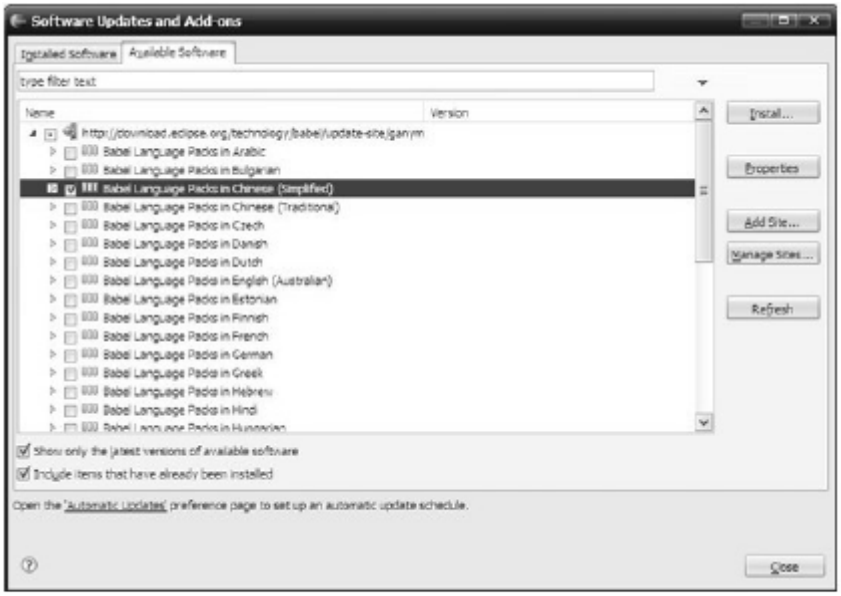


图 2-5　Avaliable Software 选择框

（3）选择“Simplified Chinese”语言包后，点击 “Install... ”按钮，等待 Eclipse 处理。

处理完成后会出现“Install”对话框，这时会提示你选择要安装的语言包。根据提示，很容易完成后面的操作，这里就不再赘述了。

安装完毕后，重新启动 Eclipse 即可完成全部汉化过程。

如果重启 Eclipse 后不显示中文，请用命令行 “eclipse.exe -nl zh_CN”重新启动 Eclipse。

2.2.3 SDK 和 ADT 的安装和配置

安装了 JDK 和 Eclipse 后，现在就要安装 Android SDK 和 ADT 插件了。

1. Android SDK 安装

（1）解压缩下载好的 SDK 安装包到要安装 SDK 的路径，然后运行“SDK Setup.exe”。

（2）如果遇到了消息为“Failed to fetch URL…”的错误提示，那么需要将 HTTPS 方式改为 HTTP 方式，在“Android SDK and AVD Manager”窗口的左侧选择 “Settings”，选中 “Force https://…”选项（如图 2-6 所示），点击“Save & Apply”并重新运行 SDK Setup.exe。

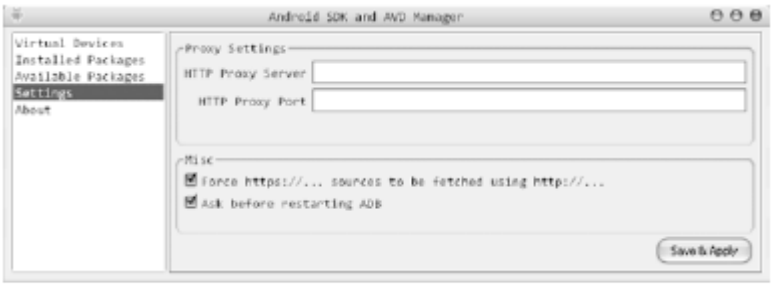


图 2-6 更改 HTTP 方式

（3）点击“Available Packages”，选择要安装的 API 版本及 USB 驱动和 SDK 文档，如图 2-7 所示。这里为了测试方便，所以全部选择了。

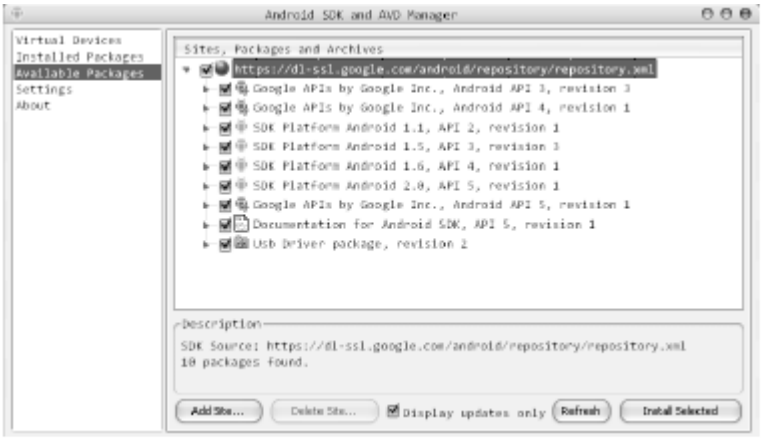


图 2-7 选择 API 版本

（4）选择好之后点击“Install Selected”按钮，安装选中的软件包，在接下来出现的界面中依次点击“Accept All”单选按钮和“Install Accepted”按钮，开始下载所选择的安装包。

下载完成之后，根据提示即可完成后续的安装操作。

到这里，我们就完成了 Android SDK 的安装，下面来配置 Android SDK。

2. Android SDK 配置

需要将 Android SDK 安装目录中的 tools 文件夹路径添加到环境变量中以便使用，操作步骤如下：

（1）右键点击“我的电脑”，依次选择“属性”→“高级”→“环境变量”选项，如图 2-8 所示。

（2）选择“系统变量”中变量名为“Path”的项，点击“编辑”按钮，将 Android SDK 安装文件夹下的 tools 文件夹的路径加入到“Path”变量中，注意用“、”隔开，如图 2-9 所示。

图 2-8　环境变量

图 2-9　编辑系统环境变量

（3）依次点击“确定”，完成环境变量配置。

3. 安装和配置 ADT

下面我们来安装和配置 ADT 插件，步骤如下：

（1）启动 Eclipse，点击“Help”菜单，依次选择“Software Update...”项和“Avaiable Software”选项卡，点击“Add Site...”按钮，输入地址 https://dl-ssl.google.com/android/eclipse/，结果如图 2-10 所示。

（2）点击“OK”，这时可能会出现如图 2-11 所示的错误。

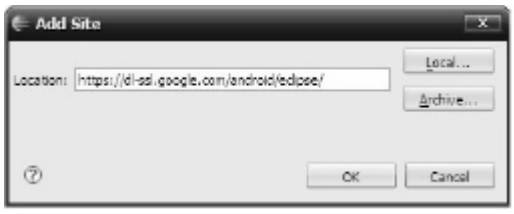


图 2-10　添加 ADT 的更新地址

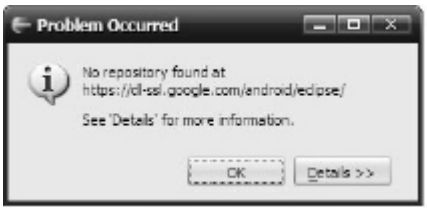


图 2-11　更新地址错误

解决这个问题的方法是：将“https://dl-ssl.google.com/android/eclipse/”中的“https”更改为“http”，在接下来的对话框中选中“Name”下的所有选项，根据提示即可完成后续的安装过程。

（3）打开菜单“Windows”，依次选择“Preferences”→“Android”，点击“Browse...”按钮，选择 Android SDK 的安装路径，如图 2-12 所示。

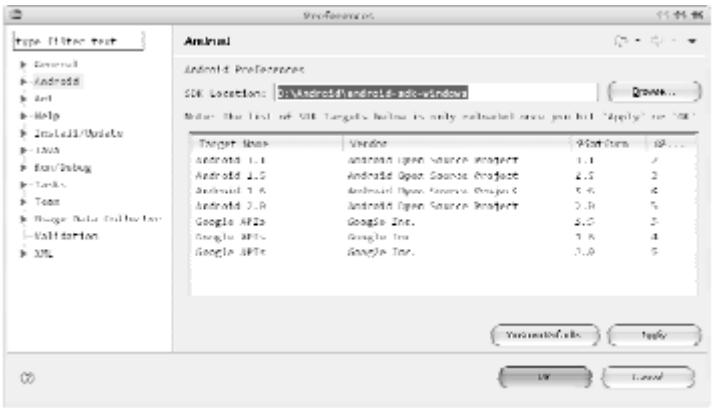


图 2-12　Eclipse 首选项

（4）点击“OK”按钮，开打菜单“File”，依次选择“NEW”→“Project...”菜单命令，出现如图 2-13 所示的“Android Projest”选项，则表示安装配置成功。

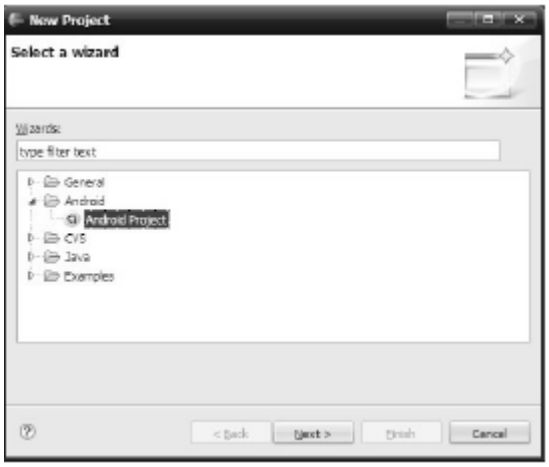


图 2-13　新建工程界面

到这里，我们的准备工作已经就绪，可以在 Android 平台上开发我们的应用了，很心动吧！神奇的 Android 之旅即将开始。

2.3 创建第一个 Android 项目——HelloAndroid

为了便于第一次开发 Android 应用的朋友能对整个开发过程有系统性的了解，并能亲自动手创建自己的应用，我们特在本书的开篇准备了一个简单的实例项目——HelloAndroid。

2.3.1 创建 HelloAndroid 项目

ADT 提供了简单的生成 Andriod 应用框架的功能，我们现在使用 ADT 通过 Eclipse 创建一个 Android 工程，其步骤如下。

（1）打开 Eclipse 开发工具，新建一个项目，在弹出的“New Project”对话框的列表中展开“Android”项，然后选择“Android Project”子项，如图 2-14 所示。

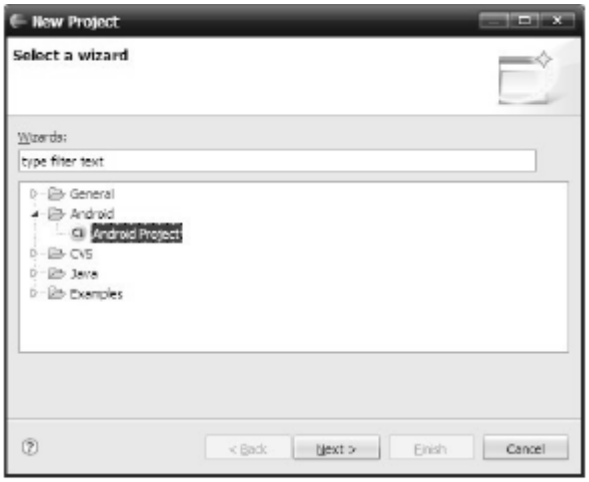


图 2-14 新建一个 Android 工程

（2）点击“Next”按钮，在“Project name”文本框中输入“HelloAndroid”，然后在“Build Target”选项框中选择“Android SDK 1.5”，在 Application name 文本框中输入这个应用程序的名字 HelloAndroid)，在 Package name 文本框中输入应用程序包的名字（com.yarin.Android.HelloAndroid），在 Create Activity 文本框中输入 Activity 的名字（HelloAndroid），如图 2-15 所示。

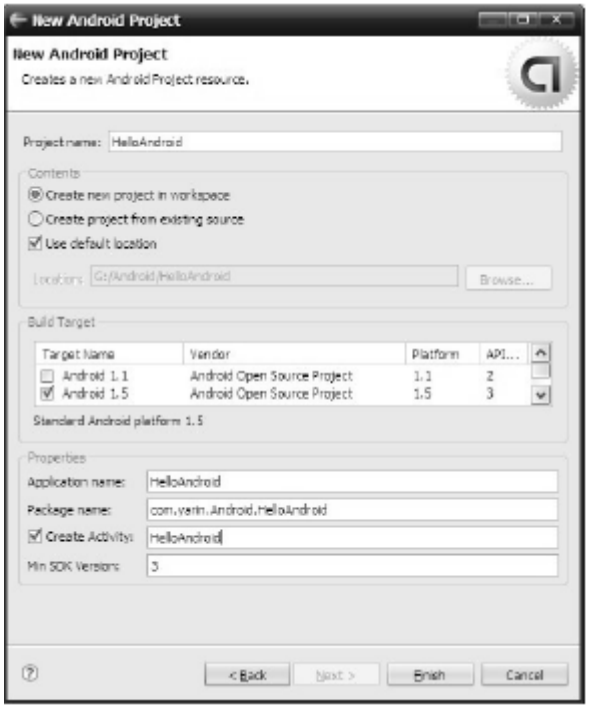


图 2-15　新建 HelloAndroid 工程

（3）单击“Finish”按钮，此时 Eclipse 会自动完成 Android 项目的创建，这时 Eclipse 开发平台左边的导航器中显示了刚才创建的项目“HelloAndroid”。如果没有出现导航器，则可以通过单击“Window”→“Show View”→“Package Explorer”菜单命令来显示导航器，如图 2-16 所示。

到这里，HelloAndroid 项目已经创建好，而且这个项目是由我们前面安装的 ADT 插件自动生成，所以不用编写代码即可运行。下面我们将讲述如何在模拟器中运行刚刚创建的 HelloAndroid 项目。

2.3.2　运行 HelloAndroid 及模拟器的使用

上面我们已经利用 ADT 插件通过 Eclipse 创建好了第一个 Android 项目，而且没有编写任何代码，我们很想看看运行之后的结果！不要着急，在模拟器中运行该应用之前，有必要了解一下模拟器的使用和配置。

从 Android 1.5 开始引入了 AVD（Android Virtual Device）这个概念。AVD 是一个经过配置的模拟器。在创建 AVD 时可以配置的选项有：模拟器影像大小、触摸屏、轨迹球、摄像头、屏幕分辨率、键盘、GSM 、GPS、Audio 录放、SD 卡支持、缓存区大小等。配置 Android 模拟器的具体步骤如下所示。

（1）首先打开“Android SDK and AVD Manager”，如图 2-17 所示。

图 2-16　显示项目管理器

图 2-17　Android SDK and AVD Manager 菜单

（2）点击左边的“Virtual Devices”选项，再点击右边的“New...”按钮，新建一个 AVD。

（3）在“Name”标签处填写 AVD 的名字，在“Target”标签处选择 API 等级，在“Size”标签处填写要创建的 SD 卡的大小，在“Skin”标签中设置模拟器的风格，如图 2-18 所示。

（4）到这里，我们便可以运行第一个 Android 项目了吗？还是不行，还需要配置模拟器运行的 AVD。操作步骤为：点击“Run”，选择“Run Configurations”菜单命令，打开“Run Configurations”对话框，如图 2-19 所示。

（5）双击“Run Configurations”对话框左边的导航器中的“Android Application”菜单命令，创建一个 Android 项目运行配置。在右边的“Name”文本框中输入 Android 项目运行配置的名字（HelloAndroid），在“Android”选项卡中的“Project”文本框中输入要运行的 Android 项目，同样可以点击右边的“Browse...”按钮来选择 Android 项目，如图 2-20 所示。

图 2-18　创建 AVD

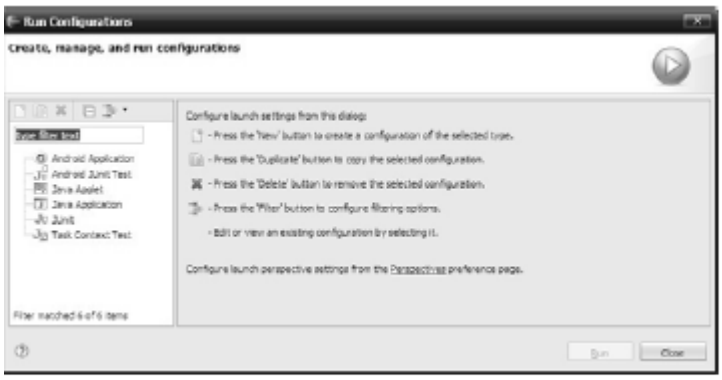


图 2-19　运行配置界面

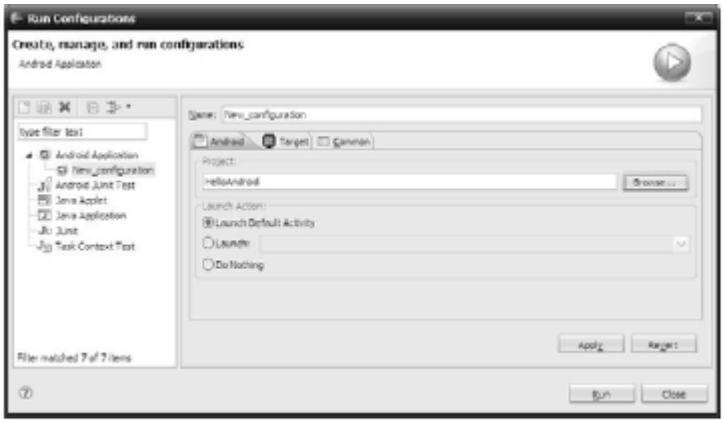


图 2-20　配置要运行的 HelloAndroid 项目

（6）点击“Target”选项卡，选择“Automatic”单选框，然后在 AVD 列表框中选择我们刚才创建的 AVD，如图 2-21 所示。

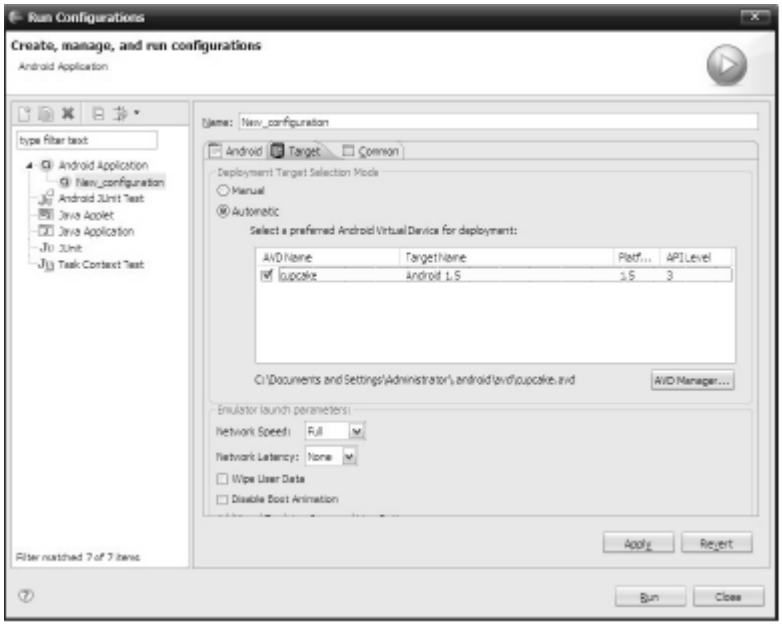


图 2-21　制定运行 HelloAndroid 项目的 AVD

（7)点击“Run”按钮,这样便可以运行 HelloAndroid 项目了,不过 Android 模拟器启动非常慢，慢慢等吧。但是 Android 的模拟器做得非常漂亮，终于可以看到第一个 Android 项目的运行效果了，如图 2-22 所示。

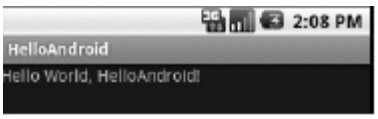


图 2-22HelloAndroid 项目在模拟器中的运行效果　　图 2-23 Android 模拟器显示中文界面

从 Android SDK 1.5 版本开始，Android 模拟器开始支持中文了，也内置了中文输入法（谷歌拼音输入法），下面我们就将模拟器改为中文环境。操作步骤为：启动 Android 模拟器，进入 Android 模拟器菜单，选择“Settings”菜单项,开打“Settings”菜单，选择“Locale&text”菜单项，打开“Locale&text”菜单，依次选择“Select locale”项和“Chinese(China)”项，这样就设置为中文了，然后返回桌面，如图 2-23 所示。

上文我们使用 ADT 插件在 Eclipse 开发工具中创建了 AVD 及设置模拟器等操作，同样可以在命令行模式下完成上面的操作。

扩展学习

大家已经看到了 Android 的模拟界面了,这款模拟器功能非常齐全,电话本、通话等功能都可正常使用（当然不是真的从模拟器中打电话）。甚至其内置的浏览器和 Google Maps 都可以联网。用户可以使用键盘输入，鼠标点击模拟器按键输入，甚至还可以使用鼠标点击、拖动屏幕进行操纵。我们在开发项目时，这个模拟器完全可以满足我们测试的需求。下面我们列举一些常用的模拟器操作。

* 列出模拟器类型：android list targets。
* 创建模拟器：android create avd --target 2 --name cupcake，cupcake 为新建模拟器的名字。
* 列出自己创建的模拟器：android list avd。
* 切换模拟器样式：在创建命令后面加上 “--skin QVGA”即可。切换样式: Windows 操作系统按 F7 键即可。
* 删除模拟器：android delete avd --name cupcake，cupcake 为删除的模拟器的名字。

* 指定用什么模拟器启动：emulator -debug avd_config -avd cupcake，cupcake 为模拟器的名字。
* 将 apk 文件安装到 Android 模拟器。操作步骤为：首先启动 Android 模拟器，然后打开命令行对话框，进入命令行模式。在命令行模式下进入 Android SDK 安装目录下面的 tools 文件夹，输入“adb install c:\ poker80.apk”（c:\ poker80.apk 是要安装的文件的路径），这样便可以将 apk 文件安装到模拟器上，如图 2-24 所示。

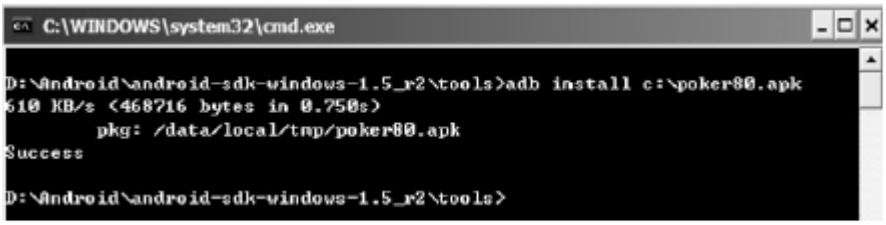


图 2-24　安装 apk 文件到模拟器

* 卸载模拟器中的 apk 文件。操作步骤为：首先启动 Android 模拟器，进入命令行模式。在命令行模式下进入 Android SDK 安装目录下面的 tools 文件夹，然后在命令行处依次输入“adb shell”、“cd data”、“cd app”、“ls”（主要是针对不知道包下面的文件的情况，可以用 ls 命令列表显示出来）、“rm com.fungsing.poker80.apk”命令（“com.fungsing.poker80.apk.apk”是你要卸载的 apk 包），如图 2-25 所示。

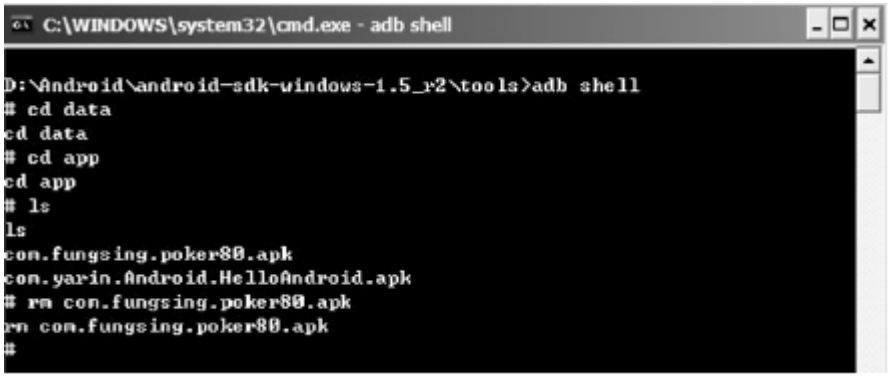


图 2-25　从 Android 模拟器卸载 apk 文件

2.3.3　调试 HelloAndroid

在 Eclipse 开发工具中调试程序的方法很多，使用 Eclipse 调试 Android 程序时需要注意一些细节上的问题。许多刚接触 Android 的开发者，在调试 Android 程序时总是不能迅速地找到程序的错误所在，Eclipse+ADT 的开发环境中没有直接跟踪对象内容的方法，但是我们可以使用 Google 提供的 ADT 插件 DDMS（Dalvik Debug Monitor Service）在 Eclipse 上轻松地调试 Android 程序。DDMS 为我们提供了很多功能，例如：测试设备截屏，针对特定的进程查看正在运行的线程以及堆信息，Logcat，广播状态信息，模拟电话呼叫，接收 SMS，虚拟地理坐标等等，下面我们通过 DDMS 来调试我们的 HelloAndroid 项目。

（1）将 Eclipse 开发工具的工作界面切换到 DDMS 标签。首先确定 Eclipse 开发工具右上角是否有“DDMS”标签，如果有，则直接点击该标签即可切换到

DDMS 工作界面，如图 2-26 所示。如果没有，则点击“Open Perspective”按钮，选择“Other...”命令按钮，打开“Open Perspective”对话框，如图 2-27 所示。在“Open Perspective”对话框中选择“DDMS”选项，然后点击“OK”按钮，如图 2-28 所示。

图 2-26 DDMS 工作界面切换

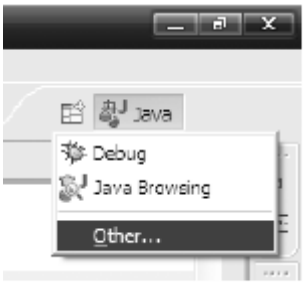


图 2-27　打开视图布局显示操作

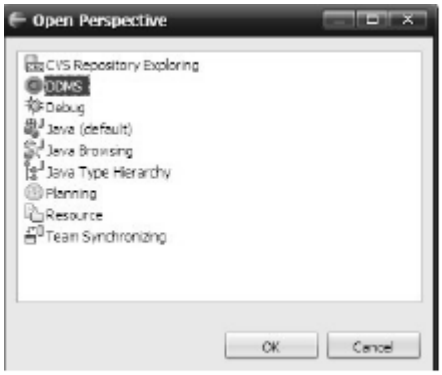


图 2-28 视图布局选择框

（2）在“DDMS”界面中选择“Devices”标签，查看其菜单的功能，可以看到 Debug Process（调试进程）、Update Threads（更新线程）、Update Heap（更新堆）、Cause GC（引起垃圾回收）、Stop Process（停止进程）、Screen Capture（屏幕截图）、Reset adb（重启 Android Debug Bridge）菜单选项，如图 2-29 所示。

从图 2-29 中可以观察到 Android 程序运行时的各种状态，比如进程信息、线程分析、堆内存的占用，结束一个进程等。当然，这些操作都是在 DDMS 框架下进行的，日常开发的程序是无法执行调用的。如果 adb 调试桥运行不稳定，可以选择“Reset adb”来重新启动“adb.exe”进程。下面我们介绍如何使用 DDMS 的“Logcat”来调试 Android 程序，步骤如下：

（1）“Logcat”通过“android.util.Log”类的静态方法来查找错误和打印系统日志消息。它是一个进行日志输出的 API，我们在 Android 程序中可以随时为某一个对象插入一个 Log，然后在 DDMS 中观察 Logcat 的输出是否正常。android.util.Log 常用的方法有以下 5 个：

```
*  Log.v(String tag, String msg);
*  Log.d(String tag, String msg);
*  Log.i(String tag, String msg);
```

* Log.w(String tag, String msg);
* Log.e(String tag, String msg)。

图 2-29 DDMS 操作菜单

这 5 种方法的首字母分别对应 VERBOSE、DEBUG、INFO、WARN、ERROR。当利用 DDMS 进行调试时，它们的区别并不大，只是显示的颜色不同，可以控制要显示的某一类错误，一般如果使用“断点”方式来调试程序，则使用 Log.e 比较合适。但是根据规范建议 Log.v、Log.d 信息应当只存在于开发过程中，最终版本只可以包含 Log.i、Log.w、Log.e 这三种日志信息。下面我们对“HelloAndroid”程序进行调试，首先修改“HelloAndroid.java”如代码清代 2-1 所示。我们在代码中加入了需要输出的日志信息。

代码清单 2-1　第 2 章
\HelloAndroid\src\com\yarin\Android\HelloAndroid\HelloAndroid.java
Java 代码

```
1. /* 定义 TAG 标签，这样可以很好地区分打印出来的 Log */
2. private static final String TAG = "HelloAndroid";
3. public void onCreate(Bundle savedInstanceState)
4. {
5.  super.onCreate(savedInstanceState);
6.  /* 打印出不同的 Log 信息 */
7.  Log.v(TAG,"VERBOSE");
8.  Log.d(TAG,"DEBUG");
9.  Log.i(TAG,"INFO");
10. Log.w(TAG,"WARN");
11. Log.e(TAG,"ERROR");
12. setContentView(R.layout.main);
13.}
```

（2）点击“Run”→“Debug”菜单命令，进入调试模式，如图 2-30 所示。

（3）切换到“DDMS”界面，点击“Logcat”标签，即可查看我们刚刚在程序中打印的 Log 信息。用不同颜色表示了不同等级的信息，这样就可方便地对程序进行跟踪，使得调试 Android 程序更加方便。

图 2-30 调试菜单命令

在调试 Android 程序时，同样可以通过设置断点的方式来调试程序。在启动应用程序进行调试时，Eclipse 会自动切换到 Debug 透视图。毫无疑问，最常见的调试步骤是设置断点，这样可以检查条件语句或循环内的变量和值。要在 Java 透视图的 Package Explorer 视图中设置断点，双击选择的源代码文件，在一个编辑器中打开它。遍历代码，将鼠标放在可疑代码一行的标记栏（在编辑器区域的左侧）上，双击即可设置断点，如图 2-31 所示。

注意 最好不要将多条语句放在一行上，因为会无法单步执行，也不能为同一行上的多条语句设置行断点。

一旦找到错误发生的位置，你可能想知道在程序崩溃之前它在做什么。一种方法是单步执行程序的每行语句，直到运行到可疑的那一行。有时候最好只运行一段代码，在可疑处停止运行，检查数据。另一种方法是声明条件断点，断点在表达式值发生变化时触发。如图 2-32 所示，我们设置条件“savedInstanceState == null”，当满足这个条件时，程序就会挂起。除此之外，在输入条件表达式时，也可以使用代码帮助。为了在 Debug 透视图的编辑器中计算表达式的值，选择设置了断点的那行代码，在上下文菜单中，通过 Ctrl+Shift+I 或右键单击你感兴趣的变量并选择 Inspect 选项。在当前堆栈框架的上下文中会计算表达式的值，在 Display 窗口的 Expressions 视图中会显示结果。

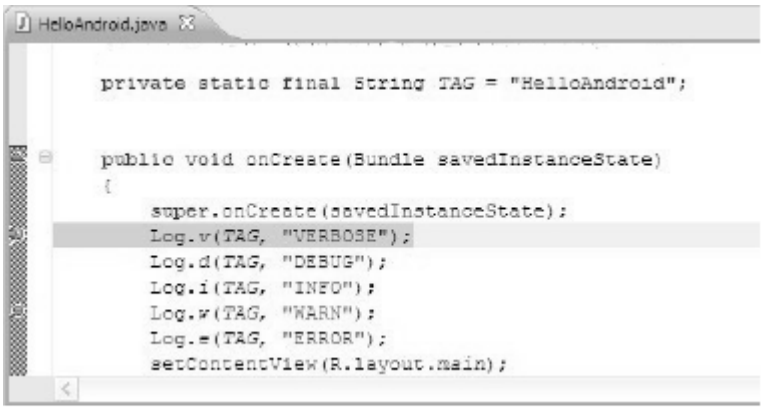


图 2-31 设置“断点”

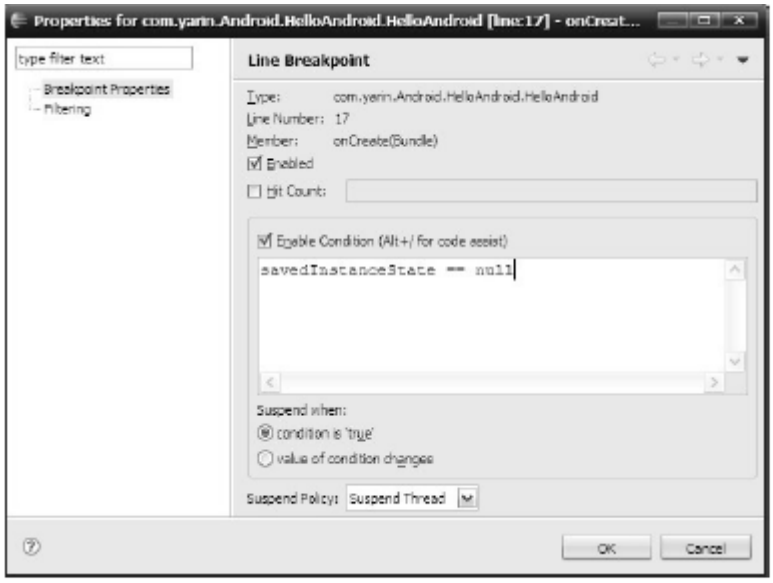


图 2-32　设置条件断点

要在 Debug 视图中挂起执行线程，选择一个运行线程，单击 Debug 视图工具栏中的 Suspend。该线程的当前调用堆栈就会显示出来，当前执行的代码行就会在 Debug 透视图中的编辑器中高亮显示。挂起一个线程时，将鼠标放在 Java 编辑器中的变量上，该变量的值就会在一个小的悬停窗口中显示出来。此时，该线程的顶部堆栈框架也会自动选中，其中的可视变量也会在 Variables 视图中显示出来，可以通过单击 Variables 视图中合适的变量名来检查变量。

以上列举了一些在 Eclipse 编辑器中常用的调试方式，当然调试的方式很多，读者同样可以根据自己的需要选择不同的方式进行调试。希望读者能够根据不同的错误采取不同的方式进行调试，使错误能快速地出现在眼前。

2.4　小结

本章主要对 Android 应用开发的前期工作进行了整理，即 Android 开发工具的准备、环境的搭建及配置，最后为了测试我们的环境安装是否正确，写出了一个最经典的 HelloAndroid 程序。同时，了解了 Android 平台如何调试程序，以辅助我们后期能够快速开发出 Android 应用。本章是 Android 应用开发的基础，大家好好把握，下面我们将正式对 Android 进行系统学习。

第二部分 基础篇

第 3 章　Android 程序设计基础

通过上一章的学习，我们对 Eclipse+ADT 开发流程有了初步的认识和了解，对初学者来说，这一章的内容比较繁琐，但是又必须掌握，这也是进行 Android 开发必须经过的第一步，有了这个基础，我们下面将进行正式开始 Android 应用程序设计。

3.1　Android 程序框架

上一章我们建立了 HelloAndroid 项目，代码是由 ADT 插件自动生成的，我们没有对其进行编码，所以没有对其框架进行分析。其实每一个平台都有自己的结构框架，比如我们在最初学习 Java 或者 C\C++时，第一个程序总是 main 函数，以及文件类型和存储方式等。这一节将对 Android 平台的目录结构、文件类型及其负责的功能和 Android 平台的 main 函数进行剖析。

3.1.1 Android 项目目录结构

有了前面两章的基础，现在我们再来打开上一章建立的 HelloAndroid 项目，分析其项目目录结构，对 Android 项目进一步地深入了解。首先启动 Eclipse，展开“Package Explorer”导航器中的“HelloAndroid”项目，如图 3-1 所示。

图 3-1 HelloAndroid 项目

与一般的 Java 项目一样，src 文件夹是项目的所有包及源文件（.java），res 文件夹中则包含了项目中的所有资源，比如：程序图标（drawable）、布局文件（layout）、常量（values）等。下面来介绍其他 Java 项目中没有的的 gen 文件夹中的 R.java 文件和每个 Android 项目都必须有的 AndroidManfest.xml 文件。

* R.java 是在建立项目时自动生成的，这个文件是只读模式，不能更改，R.java 文件是定义该项目所有资源的索引文件。先来看看 HelloAndroid 项目的 R.java 文件，如代码清单 3-1 所示。

代码清单 3-1 第 2 章
\HelloAndroid\gen\com\yarin\Android\HelloAndroid\R.java
Java 代码

```
1. package com.yarin.Android.HelloAndroid;
2. public final class R {
3.     public static final class attr {
4.     }
```

```
    public static final class drawable {
        public static final int icon=0x7f020000;
    }
    public static final class layout {
        public static final int main=0x7f030000;
    }
    public static final class string {
        public static final int app_name=0x7f040001;
        public static final int hello=0x7f040000;
    }
}
```

可以看到这里定义了很多常量，这些常量的名字都与 res 文件夹中的文件名相同，这再次证明了 R.java 文件中所存储的是该项目所有资源的索引。有了这个文件，可以很快地找到要使用的资源，由于这个文件不能手动编辑，所以当在项目中加入了新的资源时，只需要刷新一下该项目，R.java 文件便自动生成了所有资源的索引。

* AndroidManfest.xml 文件则包含了该项目中所使用的 Activity、Service、Receiver，我们先来打开 HelloAndroid 项目中的 AndroidManfest.xml 文件，如代码清单 3-2 所示。

代码清单 3-2　第 2 章\HelloAndroid\AndroidManfest.xml
Java 代码

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
      package="com.yarin.Android.HelloAndroid"
      android:versionCode="1"
      android:versionName="1.0">
    <application android:icon="@drawable/icon" android:label="@string/app_name">
        <activity android:name=".HelloAndroid"
                  android:label="@string/app_name">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
```

```
15.     <uses-sdk android:minSdkVersion="5" />
16.</manifest>
```

代码清单 3-2 中，itent-filters 描述了 Activity 启动的位置和时间。每当一个 Activity（或者操作系统）要执行一个操作时，它将创建出一个 Intent 的对象，这个 Intent 对象能承载的信息可描述你想做什么，你想处理什么数据，数据的类型，以及一些其他信息。而 Android 则会和每个 Application 所暴露的 intent-filter 的数据进行比较，找到最合适 Activity 来处理调用者所指定的数据和操作。下面我们来仔细分析 AndroidManfest.xml 文件，如表 3-1 所示。

表 3-1　AndroidManfest.xml 分析

项	说　明
manifest	根节点，描述了 package 中所有的内容
xmlns:android	包含命名空间的声明。xmlns:android=http://schemas.android.com/apk/res/android，使得 Android 中各种标准属性能在文件中使用，提供了大部分元素中的数据
Package	声明应用程序包
application	包含 package 中 application 级别组件声明的根节点。此元素也可包含 application 的一些全局和默认的属性，如标签、icon、主题、必要的权限，等等。一个 manifest 能包含零个或一个此元素（不能大余一个）
android:icon	应用程序图标
android:label	应用程序名字
Activity	用来与用户交互的主要工具。Activity 是用户打开一个应用程序的初始页面，大部分被使用到的其他页面也由不同的 Activity 所实现，并声明在另外的 Activity 标记中。注意，每一个 Activity 必须有一个 <activity>标记对应，无论它给外部使用或是只用于自己的 package 中。如果一个 Activity 没有对应的标记，你将不能运行它。另外，为了支持运行时查找 Activity，可包含一个或多个<intent-filter>元素来描述 Activity 所支持的操作
android:name	应用程序默认启动的 Activity
intent-filter	声明了指定的一组组件支持的 Intent 值，从而形成了 IntentFilter。除了能在此元素下指定不同类型的值，属性也能放在这里来描述一个操作所需的唯一的标签、icon 和其他信息
action	组件支持的 Intent action
category	组件支持的 Intent Category。这里指定了应用程序默认启动的 Activity
uses-sdk	该应用程序所使用的 sdk 版本

下面我们看看资源文件中一些常量的定义，如 String.xml，如代码清单 3-3 所示。

代码清单 3-3　第 2 章\HelloAndroid\String.xml
Java 代码

```
1. <?xml version="1.0" encoding="utf-8"?>
2. <resources>
3.     <string name="hello">Hello World, HelloAndroid!</string>
4.     <string name="app_name">HelloAndroid</string>
5. </resources>
```

这个文件很简单，就定义了两个字符串资源，因此，我们可以在代码清单3-1中看到如下内容，即定义了“app_name”和“hello”两个常量，分别指向代码清单3-3中的两个字符串资源。

Java 代码

```
1. public static final class string {
2.     public static final int app_name=0x7f040001;
3.     public static final int hello=0x7f040000;
4. }
```

那么如何在程序中使用我们所定义的这些资源呢？首先，通过Context的getResources实例化一个Resources对象，然后通过Resources的getString方法取得指定索引的字符串，代码如下：

Java 代码

```
1. Resources r = this.getContext().getResources();
2. String appname= ((String) r.getString(R.string.app_name));
3. String hello= ((String) r.getString(R.string.hello));
```

项目中所有使用的常量都可以通过这种XML文件的方式定义，比如，下面是我们通过XML文件定义的一些有关颜色的资源。

Java 代码

```
1. <?xml version="1.0" encoding="utf-8"?>
2. <resources>
3.  <color name="status_idle">#cccccc</color>
4.  <color name="status_done">#637a47</color>
5.  <color name="status_sync">#cc9900</color>
6.  <color name="status_error">#ac4444</color>
7. </resources>
```

现在来分析HelloAndroid项目的布局文件（layout），首先打开res->layout->main.xml文件，如代码清单3-4所示。

代码清单3-4　第2章\HelloAndroid\res\layout\main.xml

Java 代码

```
1. <?xml version="1.0" encoding="utf-8"?>
2. <LinearLayout xmlns:android="http://schemas.android.com/apk/res
   /android"
3.     android:orientation="vertical"
4.     android:layout_width="fill_parent"
```

```
5.      android:layout_height="fill_parent"
6.      >
7.  <TextView
8.      android:layout_width="fill_parent"
9.      android:layout_height="wrap_content"
10.     android:text="@string/hello"
11.     />
12.</LinearLayout>
```

代码清单 3-4 中，有以下几个布局和参数。

* <LinearLayout>：线性版面配置，在这个标签中，所有元件都是按由上到下的排列排成的。
* android:orientation：表示这个介质的版面配置方式，其中“vertical”代表从上到下垂直布局，而“horizontal”代表从左到右水平布局。
* android:layout_width：定义当前视图在屏幕上所占的宽度，fill_parent 即填充整个屏幕。
* android:layout_height：定义当前视图在屏幕上所占的高度，fill_parent 即填充整个屏幕。
* wrap_content：随着文字栏位的不同而改变这个视图的宽度或高度。

layout_weight 用于给一个线性布局中的多个视图的重要度赋值。所有视图都有 layout_weight 值，默认为零，即需要显示多大的视图就占据多大的屏幕空间。如果值大于零，则将父视图中的可用空间分割，分割大小具体取决于每一个视图的 layout_weight 值和该值在当前屏幕布局的整体 layout_weight 值，以及在其他视图屏幕布局的 layout_weight 值中所占的比例。

在这里，布局中设置了一个 TextView，用来配置文本标签 Widget，其中设置的属性 android:layout_width 为整个屏幕的宽度，android:layout_height 可以根据文字来改变高度，而 android:text 则设置了这个 TextView 要显示的文字内容，这里引用了@string 中的 hello 字符串，即 String.xml 文件中的 hello 所代表的字符串资源。hello 字符串的内容“Hello World, HelloAndroid!”就是我们在 HelloAndroid 项目运行时看到的字符串。

最后，我们来分析 HelloAndroid 项目的主程序文件 HelloAndroid.java，如代码清单 3-5 所示。

代码清单 3-5　第 2 章
\HelloAndroid\src\com\yarin\Android\HelloAndroid\HelloAndroid.java
Java 代码

```
1. ...
2.  public void onCreate(Bundle savedInstanceState)
```

```
3.  {
4.   super.onCreate(savedInstanceState);
5.   /* 设置 Activity 要显示的布局为(R.layout.main) */
6.   setContentView(R.layout.main);
7.  }
8. ...
```

主程序 HelloAndroid 类继承自 Activity 类，重写了 void onCreate(Bundle savedInstanceState)方法。在 onCreate 方法中通过 setContentView(R.layout.main)设置 Activity 要显示的布局文件（\layout\main.xml）。

到这里，是不是明白了为什么我们在创建项目时没有进行编码就可以直接运行程序呢？当然，这也是 Android 开发的特点，这样可以很轻松地将代码和 UI 分开，在国际化和程序维护方面有着巨大的作用。如果你的 Android 程序需要适应国际化，比如说多国语言等问题，那么就可以定义不同语言的 UI 布局，在程序装载时调用不同的布局。而且，如果我们需要修改 UI 的一些问题，就不必查看代码了，直接更改这些布局文件即可，是不是很方便？当然，这需要开发者在开发时使用这种 MVC 框架，尽量减少使用“硬编码”。笔者个人建议使用这种框架。

3.1.2 Android 应用解析

上面我们了解了 Android 应用程序的目录结构和其中每个文件的功能，要进行应用开发，还需要对 Android 应用构造进行深入分析。Android 应用程序由 4 个模块构造而成：Activity，Intent，Content Provider，Service。

当然，也不是每个 Android 应用程序都必须由这 4 部分组成，可以根据开发者需求进行组合，比如上面建立的 HelloAndroid 项目就只使用了 Activity 这一个模块。但是，任何一个应用程序都必须在 AndroidManfest.xml 文件中声明使用到的这些模块。

1. Activity

Activity 是最基本的模块，我们在 HelloAndroid 项目中已经使用过。我们称之为“活动”，在应用程序中，一个 Activity 通常就是一个单独的屏幕。每一个活动都被实现为一个独立的类，并且从活动基类中继承而来，活动类将会显示由视图控件组成的用户接口，并对事件作出响应。例如 HelloAndroid 项目中的 HelloAndroid.java 即继承了 Activity 类。大多数的应用都是由多个 Activity 显示组成，例如，对一个文本信息应用而言，第一个屏幕用来显示发送消息的联系人列表，第二个屏幕用来写文本消息和选择收件人，第三个屏幕查看消息历史或者消息设置操作等。

这里的每一个屏幕就是一个活动，很容易实现从一个屏幕到一个新的屏幕，并且完成新的活动。当一个新的屏幕打开后，前一个屏幕将会暂停，并保存在历

史栈中。用户可以返回到历史栈中的前一个屏幕，当屏幕不再使用时，还可以从历史栈中删除。

简单理解，Activity 代表一个用户所能看到的屏幕，主要用于处理应用程序的整体性工作，例如，监听系统事件（按键事件、触摸屏事件等），为用户显示指定的 View，启动其他 Activity 等。所有应用的 Activity 都继承于 android.app.Activity 类，该类是 Android 提供的基层类，其他的 Activity 继承该父类后，通过父类的方法来实现各种功能，这种设计在其他领域也较为常见。

2. Intent

Android 用 Intent 这个特殊类实现在 Activity 与 Activity 之间的切换。Intent 类用于描述应用的功能。在 Intent 的描述结构中，有两个最重要的部分：动作和动作对应的数据。典型的动作类型有 MAIN、VIEW、PICK、EDIT 等，而动作对应的数据则以 URI 的形式表示。例如，要查看一个人的联系方式，需要创建一个动作类型为 VIEW 的 Intent，以及一个表示这个人的 URI。

通过解析各种 Intent，从一个屏幕导航到另一个屏幕是很简单的。当向前导航时，Activity 将会调用 startActivity(IntentmyIntent)方法。然后，系统会在所有已安装的应用程序中定义的 IntentFilter 中查找，找到最匹配 myIntent 的 Intent 对应的 Activity。新的 Activity 接收到 myIntent 的通知后，开始运行。当 startActivity 方法被调用时，将触发解析 myIntent 的动作，该机制提供了两个关键好处：

* Activity 能够重复利用从其他组件中以 Intent 形式产生的请求。
* Activity 可以在任何时候被具有相同 IntentFilter 的新的 Activity 取代。

下面我们举例说明两个 Activity 之间的切换。运行效果：当应用程序启动时显示布局 main.xml，如图 3-2 所示，当点击“切换”按钮时，屏幕显示布局 main2.xml，如图 3-3 所示，再点击“切换”按钮，又回到如图 3-2 所示界面。就这样通过 Intent 完成了两个 Activity 之间的切换。

图 3-2　Activity01

图 3-3　Activity02

下面我们来分析一下代码的具体实现，我们知道该项目是由两个 Activity 构成，在这两个 Activity 中分别显示了一个文本标签和一个按钮，关于界面的布局会在本书第 4 章进行详细讲解，要实现两个 Activity 的跳转，我们可以将要跳转的 Activity 类名绑定到 Intent 对象中，然后通过 startActivity 方法激活 Intent 对象中所指定的 Activity。关键代码如代码清单 3-6 所示。

代码清单 3-6　第 3 章
\Examples_03_01\src\com\yarin\android\Examples_03_01\Activity01.java
Java 代码

```
1. /* 监听 button 的事件信息 */
2. button.setOnClickListener(new Button.OnClickListener() {
3.  public void onClick(View v)
4.  {
5.   /* 新建一个 Intent 对象 */
6.   Intent intent = new Intent();
7.   /* 指定 intent 要启动的类 */
8.   intent.setClass(Activity01.this, Activity02.class);
9.   /* 启动一个新的 Activity */
10.  startActivity(intent);
11.  /* 关闭当前的 Activity */
12.  Activity01.this.finish();
13. }
14.});
```

然后，我们要从 Activity02 跳转到 Activity01 时，就只是需要在 Activity02.java 中使用同样的方法返回 Activity01 中。大家可以参考本书所附代码：第 3 章\Examples_03_01\src\com\yarin\android\ Examples_03_01\ Activity02.java。值得注意的是，该项目中我们使用了两个 Activity，每一个 Activity 都需要在 AndroidManifest.xml 文件中进行声明，声明方法如代码清单 3-7 所示。

代码清单 3-7　第 3 章\Examples_03_01\AndroidManifest.xml
Java 代码

```
1. <activity android:name=".Activity01"
2.           android:label="@string/app_name">
3.     <intent-filter>
4.         <action android:name="android.intent.action.MAIN" />
5.         <category android:name="android.intent.category.LAUNCHE
   R" />
6.     </intent-filter>
7. </activity>
8. <activity android:name="Activity02"></activity>
9.
```

如果希望 Android 应用能够对外部事件（如当电话呼入时，或者数据网络可用时，或者到了晚上时）做出响应，可以使用 IntentReceiver。虽然 IntentReceiver 在感兴趣的事件发生时会使用 NotificationManager 通知用户，

但它并不能生成UI。IntentReceiver在AndroidManifest.xml中注册，但也可以在代码中使用Context.registerReceiver()进行注册。当IntentReceiver被触发时，应用不必对请求调用IntentReceiver，系统会在需要时启动应用。各种应用还可以通过使用Context.broadcastIntent()将它们自己的IntentReceiver广播给其他应用。

3. Content Provider

Android应用能够将它们的数据保存到文件和SQLite数据库中，甚至是任何有效的设备中。当想将应用数据与其他的应用共享时，Content Provider就可以发挥作用了。因为Content Provider类实现了一组标准的方法，能够让其他的应用保存或读取此内容提供器处理的各种数据类型。

数据是应用的核心。在Android中，默认使用鼎鼎大名的SQLite作为系统数据库。但是在Android中，使用方法有点不一样。在Android中，每一个应用都运行在各自的进程中，当一个应用需要访问其他应用的数据时，也就是数据需要在不同的虚拟机之间传递，这样的情况操作起来可能有些困难（正常情况下，不能读取其他应用的db文件），Content Provider正是用来解决在不同的应用包之间共享数据的工具。

在Android中，Content Provider是一个特殊的存储数据的类型，它提供了一套标准的接口用来获取和操作数据。并且，Android自身也提供了现成的Content Provider：Contacts、Browser、CallLog、Settings、MediaStore。应用可以通过唯一的ContentResolver界面来使用具体的某个Content Provider，然后就可以用ContentResolver提供的方法来使用你需要的Content Provider了。其中，ContentResolver提供的方法包括query()、insert()、update()等。要使用这些方法，还会涉及URI。你可以将它理解成string形式的Content Provider的完全路径。

下面我们通过一个例子来学习Content Provider的使用，该例子主要实现通过Content Provider获得电话本中的数据，然后显示到一个TextView中，在运行程序之前我们先看看电话本中存储的电话号码，如图3-4所示，然后再运行程序看看我们获得的数据，如图3-5所示，并看看我们通过Content Provider获得的数据是否正确。

图3-4　电话本数据

图3-5　通过ContentProvider获得电话本数据

下面我们来分析一下如何实现通过 ContentProvider 取得电话本的数据，首先通过 getContentResolver 方法来取得一个 ContentResolver 对象，然后通过其 query 方法查询出符合标准的电话本记录，最后将这些数据都显示在一个 TextView 中即可，如代码清单 3-8 所示。

代码清单 3-8　第 3 章
\Examples_03_02\src\com\yarin\android\Examples_03_02\Activity01.java
Java 代码

```
1. public class Activity01 extends Activity
2. {
3.  public void onCreate(Bundle savedInstanceState)
4.  {
5.   TextView tv = new TextView(this);
6.   String string = "";
7.   super.onCreate(savedInstanceState);
8.   //得到 ContentResolver 对象
9.  ContentResolver cr = getContentResolver();
10. //取得电话本中开始一项的光标
11. Cursor cursor = cr.query(ContactsContract.Contacts.CONTENT_URI,
    null, null,
12. null, null);
13. //向下移动光标
14. while(cursor.moveToNext())
15. {
16.  //取得联系人名字
17.  int nameFieldColumnIndex = cursor.getColumnIndex(PhoneLookup.

18.  DISPLAY_NAME);
19.  String contact = cursor.getString(nameFieldColumnIndex);
20.  //取得电话号码
21.  int numberFieldColumnIndex = cursor.getColumnIndex(PhoneLooku
   p.
22.  NUMBER);
23.  String number = cursor.getString(numberFieldColumnIndex);
24.
25.  string += (contact+":"+number+"\n");
26. }
27. cursor.close();
28.  //设置 TextView 显示的内容
29.  tv.setText(string);
30.  //显示到屏幕
31.  setContentView(tv);
32. }
```

```
33.}
```

前面强调过，要使用这些模块，需要在 AndroidManifest.xml 声明，本例中我们使用了读取联系人的 API，因此，声明方式如下所示：

Java 代码

```
1. <uses-permission
2.  android:name="android.permission.READ_CONTACTS">
3.  </uses-permission>
```

4. Service

Service 即“服务”的意思，既然是服务，那么 Service 将是一个生命周期长且没有用户界面的程序。比如一个正在从播放列表中播放歌曲的媒体播放器，在这个媒体播放器应用中，应该会有多个 Activity，让使用者可以选择歌曲并播放歌曲。然而，音乐重放这个功能并没有对应的 Activity，因为使用者会认为在导航到其他屏幕时音乐应该还在播放。在这个例子中，媒体播放器这个 Activity 会使用 Context.startService()来启动一个 Service，从而可以在后台保持音乐的播放。同时，系统也将保持这个 Service 一直执行，直到这个 Service 运行结束。另外，还可以通过使用 Context.bindService()方法连接到一个 Service 上（如果这个 Service 当前还没有处于启动状态，则将启动它）。当连接到一个 Service 之后，还可用 Service 提供的接口与它进行通信。以媒体播放器为例，我们还可以执行暂停、重播等操作。

下面通过一个例子来学习 Service 的使用，该例子通过 Service 来播放一首 MP3，如图 3-6 所示。当用户点击“开始”按钮，音乐开始播放；点击“停止”按钮，停止音乐播放。当然，这里需要在资源文件中添加一首 MP3 歌曲，如图 3-7 所示。

要实现音乐的播放，需要在界面中放置两个按钮，用来控制音乐的播放和停止。而我们的音乐播放是通过一个服务来实现的，所以我们可以通过 startService 和 stopService 方法来开启和停止这个播放音乐的服务，如代码清单 3-9 所示。

图 3-6　使用 Service 播放音乐

图 3-7　test.mp3

代码清单 3-9　第 3 章
\Examples_03_03\src\com\yarin\android\Examples_03_03\ Activity01.java
Java 代码

```
1. //开始按钮
2. private OnClickListener start = new OnClickListener()
3. {
4.     public void onClick(View v)
5.     {
6.         //开启 Service
7.         startService(new Intent("com.yarin.Android.MUSIC"));
8.     }
9. };
10.//停止按钮
11.private OnClickListener stop = new OnClickListener()
12.{
13.    public void onClick(View v)
14.    {
15.            //停止 Service
16.        stopService(new Intent("com.yarin.Android.MUSIC"));

17.    }
18.};
```

下面是该例子的核心内容。如何通过 Service 来播放音乐，其实也很简单，首先创建一个 MusicService 继承自 Service，然后通过 start 和 stop 方法来控制音乐的播放，如代码清单 3-10 所示。具体实现请参见本书所附代码：第 3 章\Examples_03_03。

代码清单 3-10　第 3 章
\Examples_03_03\src\com\yarin\android\Examples_03_03
\MusicService.java
Java 代码

```
1. public class MusicService extends Service
2. {
3.  //MediaPlayer 对象
4.  private MediaPlayer player;
5.  public IBinder onBind(Intent arg0)
6.  {
7.   return null;
8.  }
9.  public void onStart(Intent intent, int startId)
10. {
```

```
11.   super.onStart(intent, startId);
12.   //这里可以理解为装载音乐文件
13.   player = MediaPlayer.create(this, R.raw.test);
14.   //开始播放
15.   player.start();
16.  }
17.  public void onDestroy()
18.  {
19.   super.onDestroy();
20.   //停止音乐—停止 Service
21.   player.stop();
22.  }
23.}
```

我们使用 Service 时同样需要在 AndroidManifest.xml 中声明，声明方式如代码清单 3-11 所示。

代码清单 3-11　第 3 章\Examples_03_03 \AndroidManifest.xml
Java 代码

```
1. <service android:name=".MusicService">
2.     <intent-filter>
3.         <action android:name="com.yarin.Android.MUSIC" />
4.         <category android:name="android.intent.category.default
   " />
5.     </intent-filter>
6. </service>
```

3.2　Android 的生命周期

在前面的几个例子中，我们发现所有继承自 Activity 的类都重写了 onCreate 方法，程序运行就会自动进入这个方法。其实 Activity 类中还有很多类似于 onCreate 的方法，比如 onStart、onResume、onPause、onDestroy 等，而这些方法都是系统自动调用，从名字上大概就可以看出这是一些关于生命周期的方法，那么这些方法被调用的先后顺序是怎样的呢？Android 应用的生命周期又是如何呢？下面通过一个例子来进一步分析。

当应用程序启动时，进入如图 3-8 所示的 Activity01 界面，此时，点击“Activity02”按钮，进入 Activity02 界面，如图 3-9 所示。再点击“Activity01”按钮，返回 Activity01 界面，最后点击“Exit”按钮退出整个应用程序。

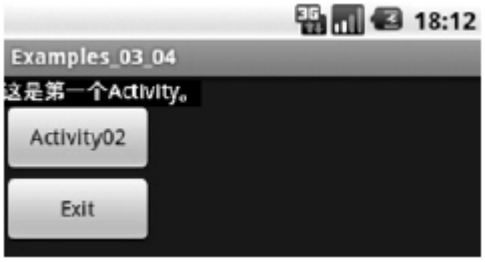


图 3-8　Activity01 界面

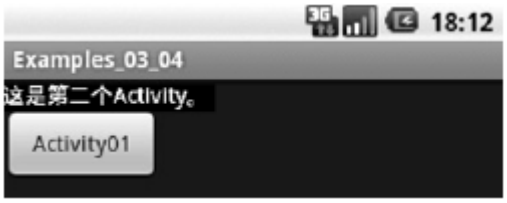


图 3-9　Activity02 界面

我们在这些类似于 onCreate 的方法中都加入了 log 函数，输出不同的信息，以便我们能更好地跟踪程序运行的过程，具体实现参见本书所附代码：第 3 章\Examples_03_04。

首先，我们需要在程序启动所默认的第一个界面中，加入一些 Log 函数，用于显示和输出 Log 信息，以帮助我们分析程序的执行流程，如代码清单 3-12 所示。

代码清单 3-12　第 3 章\Examples_03_04\src\com\yarin\android\Examples_03_04 \Activity01.java
Java 代码

```
1. public class Activity01 extends Activity
2. {
3.  private static final String TAG = "Activity01";
4.  public void onCreate(Bundle savedInstanceState)
5.  {
6.   super.onCreate(savedInstanceState);
7.   setContentView(R.layout.main);
8.   Log.v(TAG, "onCreate");
9.
10.  Button button1 = (Button) findViewById(R.id.button1);
11.  /* 监听 button 的事件信息 */
12.  button1.setOnClickListener(new Button.OnClickListener() {
13.   public void onClick(View v)
14.   {
15.    /* 新建一个 Intent 对象 */
16.    Intent intent = new Intent();
17.    /* 指定 intent 要启动的类 */
18.    intent.setClass(Activity01.this, Activity02.class);
19.    /* 启动一个新的 Activity */
20.    startActivity(intent);
21.    /* 关闭当前的 Activity */
22.    Activity01.this.finish();
23.   }
```

```
24.   });
25.   /*****************************/
26.   Button button3 = (Button) findViewById(R.id.button3);
27.   /* 监听 button 的事件信息 */
28.   button3.setOnClickListener(new Button.OnClickListener() {
29.    public void onClick(View v)
30.    {
31.     /* 关闭当前的 Activity */
32.     Activity01.this.finish();
33.    }
34.   });
35. }
36. public void onStart()
37. {
38.  super.onStart();
39.  Log.v(TAG, "onStart");
40. }
41.
42. public void onResume()
43. {
44.  super.onResume();
45.  Log.v(TAG, "onResume");
46. }
47.
48. public void onPause()
49. {
50.  super.onPause();
51.  Log.v(TAG, "onPause");
52. }
53.
54. public void onStop()
55. {
56.  super.onStop();
57.  Log.v(TAG, "onStop");
58. }
59. public void onDestroy()
60. {
61.  super.onDestroy();
62.  Log.v(TAG, "onDestroy");
63. }
64. public void onRestart()
65. {
66.  super.onRestart();
67.  Log.v(TAG, "onReStart");
```

```
68. }
69.
70.}
```

在第二个界面中，同第一个界面一样，加入一些可以区分的不同的 Log 信息。

同样需要在 AndroidManifest.xml 文件中声明所使用的两个 Activity 模块，如代码清单 3-13 所示。具体实现请参见本书所附代码：第 3 章\Examples_03_04。

代码清单 3-13　第 3 章\Examples_03_04\AndroidManifest.xml
Java 代码

```
1. <activity android:name=".Activity01"
2.           android:label="@string/app_name">
3.     <intent-filter>
4.         <action android:name="android.intent.action.MAIN" />
5.         <category android:name="android.intent.category.LAUNCHE
   R" />
6.     </intent-filter>
7. </activity>
8. <activity android:name="Activity02"></activity>
```

当在 Debug 该项目时，切换到 DDMS 标签即可以看到所打印出来的 Log 信息，这样就可以很清楚地分析程序的运行过程。

当程序第一次启动时，打印的 Log 信息如图 3-10 所示。我们看到程序的运行顺序为：Activity01 onCreate→Activity01 onStart →Activity01 onResume。这里我们可以看到，当一个 Activity 启动时，不是“创建”之后“开始”就完了，而是要经过“创建”，然后“开始”，最后“重绘”。

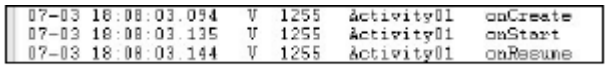


图 3-10　第一次启动进入 Activity01 界面

当我们进入 Activity02 界面时，打印出的 Log 信息如图 3-11 所示。我们看到程序的运行顺序为：Activity01 onPause→Activity02 onCreate→Activity02 onStart→Activity02 onResume→Activity01 onStop→Activity01 onDestroy。这里我们看到，当程序从 Activity01 界面进入 Activity02 界面时，并不是马上将 Activity01 销毁，而是待 Activity02 启动之后将 Activity01 停止并销毁。

当我们返回 Activity01 界面时，打印出的 Log 信息如图 3-12 所示。我们看到程序的运行顺序为：Activity02 onPause→Activity01 onCreate→Activity01 onStart→Activity01 onResume→Activity02 onStop→Activity02 onDestroy。这里我们看到，当程序从 Activity02 界面返回 Activity01 界面时，并不是马上将 Activity02 销毁，而是待 Activity01 启动之后将 Activity02 停止并销毁。

Time		pid	tag	Message
07-03 18:09:54.045	I	584	Activity	Starting .
07-03 18:09:54.114	V	1255	Activity01	onPause
07-03 18:09:54.434	V	1255	Activity02	onCreate
07-03 18:09:54.444	V	1255	Activity02	onStart
07-03 18:09:54.463	V	1255	Activity02	onResume
07-03 18:09:54.844	I	584	Activity...	Displayed
07-03 18:09:54.895	V	1255	Activity01	onStop
07-03 18:09:54.914	V	1255	Activity01	onDestroy

图 3-11　进入 Activity02 界面

Time		pid	tag	Message
07-03 18:10:51.725	I	584	Activity	Starting .
07-03 18:10:51.834	V	1255	Activity02	onPause
07-03 18:10:52.244	V	1255	Activity01	onCreate
07-03 18:10:52.253	V	1255	Activity01	onStart
07-03 18:10:52.274	V	1255	Activity01	onResume
07-03 18:10:52.694	I	584	Activity...	Displayed
07-03 18:10:52.714	V	1255	Activity02	onStop
07-03 18:10:52.724	V	1255	Activity02	onDestroy

图 3-12　返回 Activity01 界面

最后，当我们点击“Exit”按钮退出应用程序时，打印出的 Log 信息如图 3-13 所示。这里我们看到程序的运行顺序为：Activity01 onPause→Activity01 onStop→Activity01 onDestroy。这里我们看到当一个应用程序在退出时，并不是马上“停止”且“销毁”，而是经过“暂停”，到“停止”，然后再“销毁”。

Time		pid	tag	Message
07-03 18:11:24.835	V	1255	&ctivity01	onPause
07-03 18:11:25.273	W	627	IInputCo...	showStatus
07-03 18:11:25.315	V	1255	&ctivity01	onStop
07-03 18:11:25.324	V	1255	&ctivity01	onDestroy

图 3-13　退出应用程序

通过上面的例子，我们得出 Android 应用程序的生命周期如图 3-14 所示。

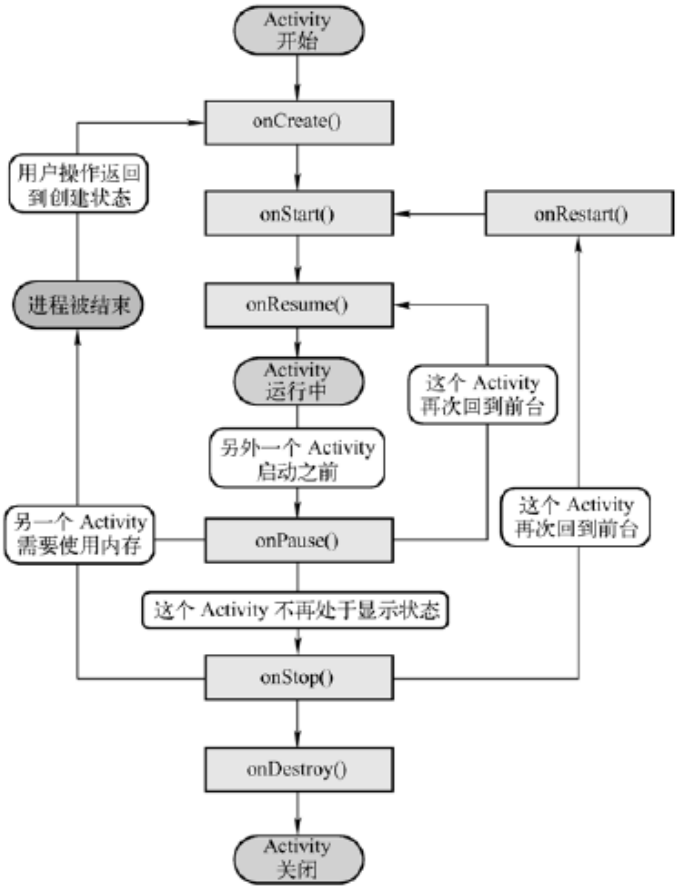


图 3-14　Android 应用的生命周期

3.3　Android 程序 UI 设计

在前面章节的例子中，我们已经接触了一些 UI 控件，比如 TextView、Button 等，其实这里所说的 UI 就是在我们所说的布局文件，UI 是一个应用程序的脸面，一个应用程序要想受用户喜爱， UI 可不能差。自从 Android SDK 1.0_r2 版本开始，ADT 提供了 UI 预览的功能。现在我们只需要打开一个 Android 项目的“/res/ layout/main.xml”并右键单击，依次选择“Open With”→“Android layout Editor”菜单命令，或者直接双击 main.xml 文件，即可以切换到 UI 设计界面，如图 3-15 所示。

图 3-15　Android Layout Editor 命令

左边的 Layouts 标签的内容则是一些线性布局，可以使用它轻松地完成对布局的排版，比如横向或者纵向布局。Views 标签则是一些 UI 控件，可以将这些控件直接拖动到右边的窗口进行编辑，如图 3-16 所示。

图 3-16　Android Layout Editor

当然，还可以点击右下角的 main.xml 标签来切换到 XML 编辑器，对代码进行编排，如图 3-17 所示。将这些功能配合起来使用，基本可以满足开发者需求。除了这个还不算太成熟的 UI 编辑器之外，笔者曾经使用过一个第三方的工具 DroidDraw，DroidDraw 是一个公开了源代码的 UI 设计器，可以根据自己的开发需要进行修改。www. DroidDraw.org 提供了在线使用 DroidDraw 的功能，当然也可以下载到本地来进行编辑，下载地址：
http://code.google.com/p/droiddraw/。

图 3-17　XML 编辑器

DroidDraw 的功能比较强大，可以直接拖动控件到窗口，然后设置其属性、参数等，这样便可以随心所欲地设计自己需要的 UI，然后点击“Generate”按钮即可生成出对应的布局代码，同时也可以点击“Load”按钮来载入已经编辑好的布局文件，如图 3-18 所示。

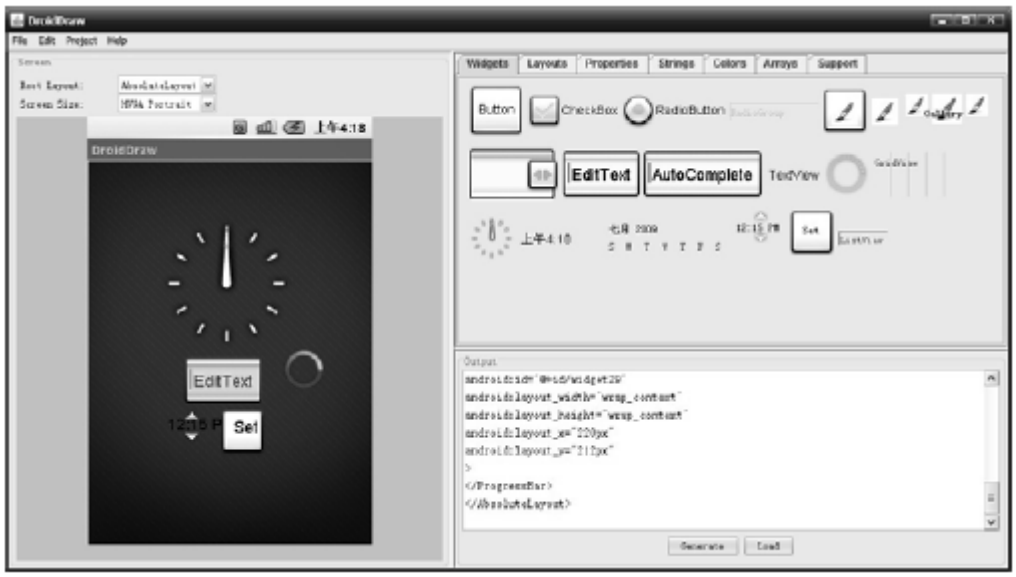


图 3-18　DroidDraw 操作界面

3.4　小结

本章主要介绍了 Android 应用程序框架及其生命周期，以及 UI 设计。首先彻底分析了上一章的 HelloAndroid 项目，从其项目目录结构、文件功能等方面分析了 Android 应用程序的基本框架，其次逐一分析了构成 Android 应用程序的 4 个模块（Activity、Intent、Content Provider、Service），分别通过示例程序演示了其功能的运用。接着通过一个示例程序验证了 Android 应用程序的运行流程，从而得出 Android 应用程序的生命周期流程图。最后介绍了两个有关 UI 设计的工具，使得程序界面更加漂亮。

相信通过本章的学习，你已经开始“喜欢”上 Android 了，有你的这份热情和执着，加上每一章的示例，让你边学边做，理论加实践，轻轻松松学会 Android 应用开发。加油吧！后面的内容更精彩。