Google 手机操作系统平台

Android 开发指南

Android 开发入门资料之一

目　录

Android 核心模块及相关技术

Android 作为一个移动设备的平台，其软件层次结构包括了一个操作系统（OS），中间件（MiddleWare）和应用程序（Application）。根据 Android 的软件框图，其软件层次结构自下而上分为以下几个层次：

第一、操作系统层（OS）

第二、各种库（Libraries）和 Android 运行环境（RunTime）

第三、应用程序框架（Application Framework）

第四、应用程序（Application）

以下分别介绍 Andoid 各个层次的软件的重点及其相关技术：

操作系统层（OS）

Android 使用 Linux2.6 作为操作系统，Linux2.6 是一种标准的技术，Linux 也是一个开放的操作系统。Android 对操作系统的使用包括核心和驱动程序两部分，Android 的 Linux 核心为标准的 Linux2.6 内核，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）：常用基于 Linux 的帧缓冲（Frame Buffer）驱动。

Flash 内存驱动（Flash Memory Driver）

照相机驱动（Camera Driver）：常用基于 Linux 的 v4l（Video for ）驱动。

音频驱动（Audio Driver）：常用基于 ALSA（Advanced Linux Sound Architecture，高级 Linux 声音体系）驱动。

WiFi 驱动（Camera Driver）：基于 IEEE 802.11 标准的驱动程序

键盘驱动（KeyBoard Driver）

蓝牙驱动（Bluetooth Driver）

Binder IPC 驱动：Andoid 一个特殊的驱动程序，具有单独的设备节点，提供进程间通讯的功能。

Power Management（能源管理）

各种库（Libraries）和 Android 运行环境（RunTime）

本层次对应一般嵌入式系统，相当于中间件层次。Android 的本层次分成两个部分一个是各种库，另一个是 Android 运行环境。本层的内容大多是使用 C++实现的。

在其中，各种库包括：

- C 库 ：C 语言的标准库，这也是系统中一个最为底层的库，C 库是通过 Linux 的系统调用来实现。
- 多媒体框架（MediaFrameword）：这部分内容是 Android 多媒体的核心部分，基于 PacketVideo（即 PV）的 OpenCORE，从功能上本库一共分为两大部分，一个部分是音频、视频的回放（PlayBack），另一部分是则是音视频的纪录（Recorder）。
- SGL：2D 图像引擎。
- SSL：即 Secure Socket Layer 位于 TCP/IP 协议与各种应用层协议之间，为数据通讯提供安全支持。
- OpenGL ES 1.0 ：本部分提供了对 3D 的支持。
- 界面管理工具（Surface Management）：本部分提供了对管理显示子系统等功能。
- SQLite：一个通用的嵌入式数据库
- WebKit：网络浏览器的核心
- FreeType：位图和矢量字体的功能。

Android 的各种库一般是以系统中间件的形式提供的，它们均有的一个显着特点就是与移动设备的平台的应用密切相关。

Android 运行环境主要指的虚拟机技术——Dalvik。Dalvik 虚拟机和一般 JAVA 虚拟机（Java VM）不同，它执行的不是 JAVA 标准的字节码（bytecode）而是 Dalvik 可执行格式（.dex）中执行文件。在执行的过程中，每一个应用程序即一个进程（Linux 的一个 Process）。二者最大的区别在于 Java VM 是以基于栈的虚拟机（Stack-based），而 Dalvik 是基于寄存器的虚拟机（Register-based）。显然，后者最大的好处在于可以根据硬件实现更大的优化，这更适合移动设备的特点。

应用程序框架（Application Framework）

Android 的应用程序框架为应用程序层的开发者提供 APIs，它实际上是一个应用程序的框架。由于上层的应用程序是以 JAVA 构建的，因此本层次提供的首先包含了 UI 程序中所需要的各种控件。

例如：Views （视图组件）包括 lists（列表）， grids（栅格）， text boxes（文本框），buttons（按钮）等。甚至一个嵌入式的 Web 浏览器。

一个 Andoid 的应用程序可以利用应用程序框架中的以下几个部分：

Activity （活动）

Broadcast Intent Receiver （广播意图接收者）

Service （服务）

Content Provider （内容提供者）

应用程序（Application）

Android 的应用程序主要是用户界面（User Interface）方面的，通常以 JAVA 程序编写，其中还可以包含各种资源文件（放置在 res 目录中）JAVA 程序及相关资源经过编译后，将生成一个 APK 包。Android 本身提供了主屏幕（Home），联系人（Contact），电话（Phone），浏览器（Browser）等众多的核心应用。同时应用程序的开发者还可以使用应用程序框架层的 API 实现自己的程序。这也是 Android 开源的巨大潜力的体现。

第一章

第一节 Android 介绍

2007 年 11 月 5 日，Google 与其他 33 家手机制造商（包含摩托罗拉、宏达电子、三星、LG）、手机芯片供应商、软硬件供应商、电信业者所联合组成的开放手持装置联盟（Open Handset Alliance），发布了名为「Android」 的开放手机软硬件平台。参与开放手持装置联盟的这些厂商，都会基于 Android 平台，来开发新的手机业务。

紧接着，在 Android 平台公布的一周之后（11 月 12 日），Google 随即发布了可以免费自由下载，能在 Windows、Mac OS X、Linux 多平台上使用的 Android 软件开发工具（Software Development Kit， SDK）与相关文档。间隔数天，又再次发布操作系统核心（kernel），与部分驱动程序的原代码。一步步展示 Google 欲将手机这个现代人的随身工具推往开放平台，让人们可以自由修改创作出符合自己需求的手机应用的决心。

身为 Google 公司创始人之一的 Sergey Brin，也在 Android 软件开发工具（SDK）发布的同时，现身于视讯广告影片中，为大众介绍 Android 平台。Sergey Brin 也同时宣布举办总奖金高达 1000 万美元的开发者大奖赛，鼓励程序开发者去深入探究 Android 平台的能力。写出具创意、实用性十足、能提供使用者更好的手机使用经验的应用程序。

2008 年 9 月 24 日，T-Mobile 首度公布第一台 Android 手机（G1）的细节，同日 Google 也释出了 Andorid SDK 1.0 rc1。对应用程序开发者而言，1.0 代表了开发者可以放心地使用 API，而不必再担心 API 有太大的变动。G1 在同年 10 月 20 正式发售。在发售前针对原 T-Mobile 用户的预购活动中，已经被预购了 150 万台。 在 10/21 日，Open Handset Alliance 公开了全部 Android 的原代码。从此，开发者拥有了一个完全开放的手机平台。

1.1 认识 Android

Android 是什么？机器人？

android 一词的本义确实是指“机器人”，同时 Android 也是 Google 于 07 年 11 月 5 日宣布的基于 Linux 平台开源手机操作系统名称，该平台由操作系统、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。Android SDK，基于这些，我们就可以使用 Java 编程语言在 Android 平台上开发应用程序了。

2008 年 9 月 22 日，美国运营商 T-Mobile USA 在纽约正式发布第一款 Google 手机——T-Mobile G1。该款手机为宏达电子制造，是世界上第一部使用 Android 操作系统的手机，支持 WCDMA/HSPA 网络，理论下载速率 7.2Mbps，并支持 Wi-Fi。

HTC G1 操作界面

从不同角度来说，Android 代表着

一个崭新的开放原代码操作系统平台----设备制造商

一个友善的免费应用程序开发环境 ----程序开发者

一个与世界各地的程序开发者，站在相同起跑点上的公平竞争机会----程序开发者。

1.1.1 Android 介绍

Android 是 Google 开发的基于 Linux 平台的开源手机操作系统。它包括操作系统、用户界面和应用程序——移动电话工作所需的全部软件，而且不存在任何以往阻碍移动产业创新的专有权障碍。谷歌与开放手机联盟合作开发了 Android，这个联盟由包括中国移动、摩托罗拉、高通、宏达和 T-Mobile 在内的 30 多家技术和无线应用的领军企业组成。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，我们希望借助建立标准化、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。我们认为此举必将推进更好、更快的创新，为移动用户提供不可预知的应用和服务。

Android SDK 已经可以下载了。其中包括了附含源代码的样例工程、开发工具、仿真器，当然了，还有你构建工程所必需的类库。

Android 作为谷歌企业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。我们发现，全球为数众多的移动电话用户从未使用过任何基于 Android 的电话。谷歌的目标是让（移动通讯）不依赖于设备甚至平台。出于这个目的，Android 将补充，而不会替代谷歌长期以来奉行的移动发展战略：通过与全球各地的手机制造商和移动运营商结成合作伙伴，开发既有用又有吸引力的移动服务，并推广这些产品。

开放手机联盟的成立和 Android 的推出是对现状的重大改变，在带来初步效益之前，还需要不小的耐心和高昂的投入。但是，我们认为全球移动用户从中能获得的潜在利益是值得付出这些努力的。如果你也是一个开发者，并对我们的想法感兴趣，就请再给我们一星期的时间，届时谷歌便能提供 SDK 了。如果你是一名移动用户，只需再等一段时间，我们的一些合作伙伴计划在 2008 年下半年推出基于 Android 平台的电话产品。如果你已经拥有一部你了解并喜爱的电话，请登录 mobile.google.com ，确保你已经安装谷歌手机地图、Gmail 以及其他一些专为你的手机开发的精彩应用。谷歌将继续努力，让这些服务变得更好，同时

也将添加更有吸引力的特性、应用和服务。

1.1.2 Android 团队

白色版 HTC G1

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 34 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，让用户体验到最优越的移动服务，同时，开发商也将得到一个新的开放级别，更方便的进行协同合作，从而保障新型移动设备的研发速度。

34 家企业的加盟，也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。

这 34 家企业中并不包含全球手机第一巨头诺基亚，把持 UIQ 平台的索尼爱立信，以及凭借着 iPhone 风光正在的苹果公司，美国运营商 AT&T 和 Verizon，当然微软没有加入，独树一帜的加拿大 RIM 和他们的 Blackberry 也被挡在门外。

手机开放联盟大家庭成员名单：

一、手机制造商：

台湾宏达国际电子（HTC）（Palm 等多款智能手机的代工厂）

摩托罗拉（美国最大的手机制造商）

韩国三星电子（仅次于诺基亚的全球第二大手机制造商）

韩国 LG 电子

中国移动（全球最大的移动运营商，有 3.5 亿用户）

日本 KDDI（2900 万用户）

日本 NTT DoCoMo（5200 万用户）

美国 Sprint Nextel（美国第三大移动运营商，5400 万用户）

意大利电信（Telecom Italia）（意大利主要的移动运营商，3400 万用户）

西班牙 Telefónica （在欧洲和拉美有 1.5 亿用户）

T-Mobile（德意志电信旗下公司，在美国和欧洲有 1.1 亿用户）

二、半导体公司：

Audience Corp（声音处理器公司）

Broadcom Corp（无线半导体主要提供商）

英特尔（Intel）

Marvell Technology Group

Nvidia （图形处理器公司）

SiRF（GPS 技术提供商）

Synaptics（手机用户界面技术）

德州仪器（Texas Instruments）

高通（Qualcomm ）

三、软件公司：

Aplix

Ascender

eBay 的 Skype

Esmertec

Living Image

NMS Communications

Noser Engineering AG

Nuance Communications

PacketVideo

SkyPop

Sonix Network

TAT-The Astonishing Tribe

Wind River Systems

1.1.3 Android 的未来发展

虽然没有看到 Gphone 的真正模样，但据了解，Google 的 Android 平台手机将在 2008 年下半年正式揭开面纱。对于消费者来说，Google 手机将是一款通用的、功能强大的、设备完整的手机产品。

美国咨询研究集团 Strategy Analytics 的最新报告指出，Google 最近公布的 android 手机软件平台，很可能在 2008 年时获得全球智能手机软件平台 2%的份额。

而老牌智能手机软件平台制造商 Symbian 发言人则表示：Google 的 android 只不过是另一个 linux，symbian 对其它软件与其形成的竞争并不感到担心。除了北美之外，Symbian 在其它地区智能手机市场都占有大部分市场份额。

与 iPhone 相似，Android 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看电子邮件、搜索网址和观看视频节目等，比 iPhone 等其他手机…应用的单一平台。

但其最震撼人心之处在于 Android 手机系统的开放性和服务免费。Android 是一个对第三方软件完全开放的平台，开发者在为其开发程序时拥有更大的自由度，突破了 iPhone 等只能添加为数不多的固定软件的枷锁；同时与 Windows Mobile、Symbian 等厂商不同，Android 操作系统免费向开发人员提供，这样可节省近三成成本。

Android 项目目前正在从手机运营商、手机厂商、开发者和消费者那里获得大力支持。谷歌移动平台主管安迪 鲁宾（Andy Rubin）表示，与软件开发合作伙伴的密切接触正在进行中。从去年 11 月开始，谷歌开始向服务提供商、芯片厂商和手机销售商提供 Android 平台，并组建“开放手机联盟”，其成员超过 30 家。

1.1.4 Android 的市场前景

Google 手机将于 08 年 10 月 22 日正式上市，与运营商捆绑的合约价为 179 美元（约合人民币 1200 元），现有的 T-Mobile 用户可以通过网络订购。在 10 月 22 日发布当天，将有 22 个国家和地区可以买到谷歌手机。在 11 月，将增至 27 个国家和地区。谷歌手机将只能在 T-Mobile 网络内使用，将会有 SIM 卡锁定限制。T-Mobile USA 提供了两种流量和短信包月计划。第一档为 25 美元，包括不限制的网络流量，以及一定数量的短信；第二档为 35 美元，包括不限制的网络流量和不限数量的短信。G1 支持各种谷歌的服务，包括 Gmail、Google Maps、YouTube、Google 日历和 Google Talk，内置 Chrome Lite 浏览器。使用这款手机也需要 Gmail 账号。

“我爱死我的 G1 了！”谷歌联合创始人拉里 佩奇（Larry Page）憧憬：“对于我来说，谷歌手机最激动人心的一点就是它的未来。它的功能很强，正如几年前的电脑一样。你可以上网，安装软件。”

T-Mobile USA 为德国电信旗下移动子公司，运营 GSM/HSPA 网络，拥有用户近 3000 万。此次发布的谷歌手机采用高通 MSM7201A 处理器，支持 7.2Mbps 下载速率，美国运营商 3G 网络设定的套餐速率一般可达 1Mbps 以上。MSM7201A 为单芯片、双核，整合硬件加速多媒体功能、支持 3D 图形，300 万像素的摄像功能可以扫描条形码，并且有 GPS 功能。

在美国四大移动运营商中，T-Mobile USA 处于相对弱势的地位，不过 T-Mobile 在欧洲有更为强势的是 CDMA EV-DO 网络，而 T-Mobile USA 和苹果 iPhone 的独家运营商 AT&T 为 GSM/HSPA 网络，后者在 2005 年 12 月就推出了全球第一个 HSDPA 商用服务。

T-Mobile USA 今年 5 月方才在纽约推出了 3G 服务，目前已经在奥斯汀、巴尔的摩、波士顿、达拉斯、休斯敦、拉斯维加斯、迈阿密、明尼阿波利斯、凤凰城、波特兰、圣 安东尼奥和圣地亚哥等 12 个城市增加了 3G 覆盖，公司计划年底前将 3G 覆盖的城市从目前的 13 个增加到 27 个，能够为其超过 2/3 的用户提供 3G 服务。

1.1.5 Android 在中国的前景

发布会并未提及 G1 何时进入中国市场。T-Mobile USA 和谷歌高管在回答手机破解问题时表示，179 美元的定价非常有吸引力，希望顾客在专为谷歌手机优化过的网络里使用。中国移动和中国联通的网络都可以运行 G1，不过在联通明年开通 WCDMA/HSPA 网络之前，用户只能像破解 iPhone 那样降级使用 GSM 网络，无法体验高速下载的移动互联网。

Android 作为一个开放的平台，三星、摩托罗拉、索尼爱立信、LG 等厂商都有意生产 Android 系统的手机。此次谷歌手机发布，尽管宏达电拔得头筹，其他一些终端厂商如华为也已证实正在研发基于 Android 平台的手机，目前已经制成样机。

来自 Google 官方的报道，Google Android G1 手机预售量已达 150 万部。

1.1.6 Android 资源

Android 官方网站 http://www.android.com/

开放手持设备联盟（Open Handset Alliance） http://www.openhandsetalliance.com/

Google Android 开发者部落格 http://android-developers.blogspot.com/

Android 开发者大赛网站 http://code.google.com/android/adc.html

Android 文档 http://code.google.com/android/documentation.html

http://www.onlamp.com/pub/a/onlamp/2007/11/12/google-calling-inside-the-gphone-sdk.html

CNet 专访：Google 手机平台背后的原创者 http://www.zdnet.com.tw/news/comm/0，2000085675，20125898，00.htm

Android 原代码网站 http://source.android.com

第二节 Android 平台

1.2 Android 平台介绍

Android 平台是一组面向移动设备的软件包，它包含了一个操作系统、中间件和关键应用程序。开发人员可以使用 Android SDK（软件开发工具包）为这个平台创造应用程序。应用程序使用 Java 语言编写并在 Dalvik 之内运行。Dalvik 是一款量身定制的虚拟机，它专为嵌入式应用设计，运行在 Linux 内核上层。

1.2.1 Android 平台特性

应用程序框架 支持组件的复用和更换

Dalvik 虚拟机 专门为移动设备进行过优化

集成的浏览器 基于开源的 Webkit 引擎

优化的图形机制自定义的 2D 图形库，基于 OpenGL ES 1.0 规范的 3D 图形实现（本项为硬件加速器可选）

SQLite 轻量级的数据库，支持结构化数据的存储

媒体支持 面向常见的音频、视频以及静态图形档案格式

（MPEG4， H.264， MP3， AAC， AMR， JPG， PNG， GIF）

GSM 技术 GSM：global system for mobile communications （依赖硬件支持）

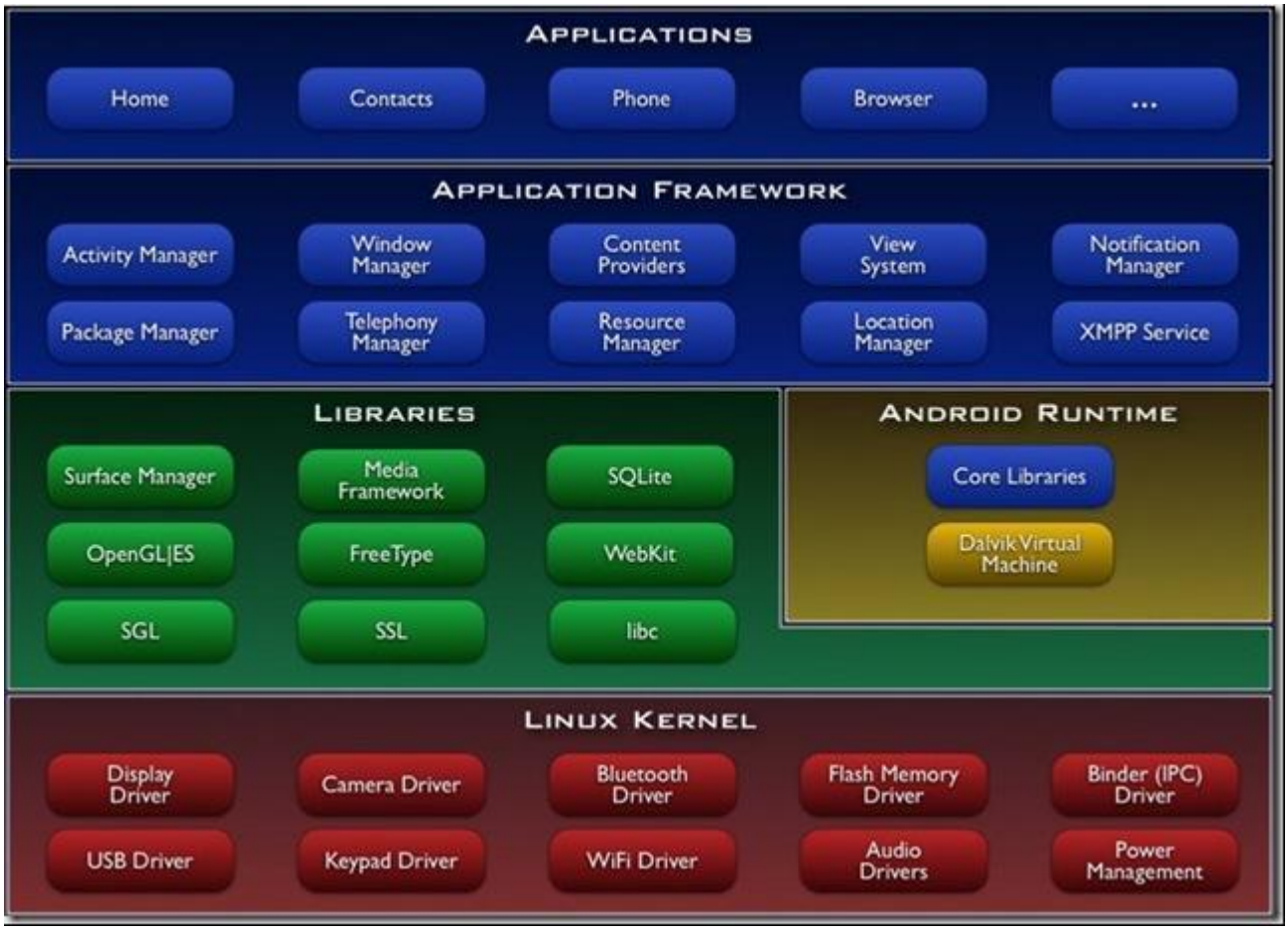
Bluetooth， EDGE， 3G， and WiFi （依赖硬件支持）

Camera， GPS， compass， and accelerometer （依赖硬件支持）

Rich development environment 丰富的开发环境，包含一套硬件仿真器，一些用于程序调试、内存和性能剖析的工具，以及支持 Eclipse 集成开发环境的插件（ADT ）。

1.2.2 Android 平台架构

下方图表显示了 Android 操作系统的主要组件。要知道关于各个部分更多的细节，下文还有描述，请继续看。



Android 平台架构

1.2.3 Applications（应用）

Android 将预装一组核心应用程序，包括 email 客户端、短信服务、日历日程、地图服务、浏览器、联系人和其他应用程序。所有应用程序都是 Java 编程语言编写的。

1.2.4 Application Frameworks（应用框架）

上文所提的核心应用程序就是依赖框架层次 API 开发的，程序员们当然也可以充分使用这些 API。应用架构设计的初衷是：简化组件复用机制；任何应用都能发布自己的功能，这些功能又可以被任何其他应用使用（当然要受来自框架的强制安全规范的约束）。和复用机制相同，框架允许组件的更换。

所有应用都是一组服务和系统，一般包含：

一套丰富且可扩展的视图 组件，含有 lists， grids， text boxes， buttons， 甚至嵌入的网络浏览器

Content Providers（内容提供器）使一个应用可以访问另外一个应用的数据（如联系人），或者使一个应用内部可以共享自身数据

Resource Manager （资源管理器），提供对非编码资源——例如本地化字符串、图形和布局文件——的访问通道

Notification Manager （通告管理器），使应用在状态栏显示自定义的警报通知。

Activity Manager （行动管理器）负责管理应用的生命周期，提供通用导航回退支持

要知道更多细节，了解应用内幕，请参考教程 《Android 应用编写》

1.2.5 Libraries（库）

Android 包含一套 C/C++库，Android 系统的各式组件都在使用。这些功能通过 Android 应用框架暴露给开发人员。下面列举一些核心库：

System C library - 衍生于 BSD 的标准 C 系统库（libc）实现（注：BSD：Berkeley Software Distribution，伯克利软件套件，是 Unix 的 衍生系统，1970 年代由伯克利加州大学开创），尤其支持嵌入式的基于 Linux 的设备。

Media Libraries - 媒体库基于 PacketVideo's OpenCORE；媒体库支持很多流行音频和视频格式、静态图形文件（包括 MPEG4， H.264， MP3， AAC， AMR， JPG， and PNG ）的播放和录制

Surface Manager - 管理范围：对子系统显示功能的访问，跨应用的无缝组合 2D 和 2D 图形层

LibWebCore - 是流行的浏览器引擎，可以支持 Android 浏览器和嵌入应用的 WEB 视图组件

SGL - 底层的 2D 图形引擎

3D libraries - 基于 OpenGL ES 1.0 API 的实现； 该类库使用硬件 3D 加速器（如果硬件支持的话）或者内置的、高度优化的 3D 软件加速机制。

FreeType - 支持位图和矢量字体

SQLite - 能干、轻量级的关系型数据库引擎，面向所有应用

1.2.6 Android Runtime（Android 运行时）

Android 的核心类库提供 Java 类库所提供的绝大部分功能。

每个 Android 应用都运行在自己的进程上，享有 Dalvik 虚拟机为它分配的专有实例。为了支持多个虚拟机在同一个设备上高效运行，Dalvik 被改写过。

Dalvik 虚拟机执行的是 Dalvik 格式的可执行文件（.dex）——该格式经过优化，以降低内存耗用到最低。Java 编译器将 Java 源文件转为 class 文件，class 文件又被内置的 dx 工具转化为 dex 格式文件，这种文件在 Dalvik 虚拟机上注册并运行。

在一些底层功能——比如线程和低内存管理方面，Dalvik 虚拟机是依赖 Linux 内核的。

1.2.7 Linux Kernel（Linux 内核）

Android 依赖 Linux 2.6 版，提供核心系统服务：安全、内存管理、进程管理、网络组、驱动模型。内核部分还相当于一个介于硬件层和系统中其他软件组之间的一个抽象层次。

第二章

第一节 安装 SDK

Android 提供免费而且跨平台的整合开发环境，只要电脑能连接上网络，我们随时都能下载相关工具下来，并开始开发 Android 应用程序。有了轻松易用的开发工具，我们可以把心力专注于如何将想法实现到应用程序上。

2.1 安装 SDK

2.1.1 系统和软件配置要求

要通过 Android SDK 中提供的代码和工具进行 Android 应用程序的开发，需要一个合适的用于开发的电脑和合适的开发环境，具体要求如下：

（1）支持的操作系统：

Windows XP 或者 Vista

Mac OS X 10.4.8 或更新的版本（只支持 x86 架构）

Linux（在 Ubuntu Dapper Drake 上测试过）

（2）支持的开发环境：

Eclipse

Eclipse 3.2，3.3（Europa）

Android 开发工具插件（可选）

其他的开发环境或者 IDE

JDK5.0 或者 JDK6.0（仅有 JRE 是不够的）

与 GNU 的 Java 编译器不兼容

Apache Ant 对 Linux 和 Mac 版本需要 1.6.5 或更新，对 Windows 版本需要 1.7 或更新

2.1.2 安装 SDK

通过链接 Download the SDK 下载好 SDK 包后，将 zip 文件解压缩至合适的地方。在下文中，我们默认你的 SDK 安装目录为$SDK_ROOT 你可以选择将$SDK_ROOT/tools 加入到你的路径中。

Linux 下，打开文件~/.bash_profile 或者~/.bashrc，找到设定 PATH 环境变量的一行，将

$SDK_ROOT/tools 的完整路径加入其中。如果没有找到设定 PATH 变量的行，你可以自己添加一行：

export PATH=${PATH}:<你的$SDK_ROOT/tools 的完全路径>

Mac 下，在你的 home 目录中找到文件.bash_profile，和 Linux 的一样处理。如果还没有在机器上设定这个文件，你可以创建一个.bash_profile 文件。

Windows 下，右键点击【我的电脑】，选择【属性】，在【高级】页中，点击【环境变量】按键，在弹出的对话框中双击“系统变量”中的变量“Path”，将$SDK/tools 的完全路径加入其中。

通过将$SDK/tools 加入系统路径，在运行 adb 和其它一些命令行工具时就不需要键入完全路径名了。需要注意到是，当你升级了 SDK 后，如果安装路径有变动的话，不要忘记了更新你的 PATH 变量的设置，将其指向变动后的路径。

2.1.3 安装 Eclipse 插件（ADT）

如果你选择 Eclipse 作为 Android 的开发环境，可以安装一个专门为 Android 定制的插件：Android Development Tools（ADT），ADT 插件集成了对 Android 工程和工具的支持，它包含了大量功能强大的扩展， 使得创建、运行、调试 Android 程序更简单更快捷。

如果你不打算使用 Eclipse，那么就不需要下载或者安装 ADT 插件。

为了下载和安装 ADT 插件，请按照下面的步骤设置 Eclipse 的远程更新站点：

1. 启动 Eclipse，依次选择 Help > Software Updates > Find and Install....

2. 在弹出的窗口中，选择 Search for new features to install，然后点击 Next

3. 点击 New Remote Site

4. 在弹出的对话框中，为远程站点输入一个自定义的名字（例如：Android Plugin），然后输入下面的内容作为其默认的 URL：然后点击 OK

https://dl-ssl.google.com/android/eclipse/

1. 现在你应该可以在搜索列表中看见刚才新加入的站点了（默认已经选中了），点击 Finish

2. 在随后的 Search Results 对话框中，选中

Android Plugin > Eclipse Integration > Android Development Tools，然后点击 Next

3. 阅读许可协议，然后选择 Accept terms of the license agreement，点击 Next

4. 点击 Finish

5. ADT 插件没有 signed，在安装过程中会弹出确认窗口，你只需要点击 Install All 就可

以了

6. 重启 Eclipse

7. 重启后，在 Eclipse 的 Preferences 中指定 SDK 的路径

a. 选择 Window > Preferences... 打开 Preference 的面板。（Mac OS X:Eclipse > Preferences）

b.在左侧的面板中选择 Android

c.在主面板中，点击 Browse...定位到 SDK 的目录

d.点击 Apply，然后点击 OK

2.1.4 更新 ADT 插件

按照下面的步骤将你的 ADT 插件升级到最新版本：

1. 选择 Help > Software Updates > Find and Install...

2. 选择 Search for updates of the currently installed features，然后点击 Finish

3. 如果 ADT 插件有任何更新，选择并且安装

另一种更新的方法：

1. 选择 Help > Software Updates > Manage Configuration

2. 在其中找到 Android Development Tools <版本号>并选中

3. 在 Available Tasks 下选择 Scan for Updates

第二节 Eclipse 上开发 Android 应用程序

2.2 在 Eclipse 上开发 Android 应用程序

要开始在 Eclipse 上开发 Android 应用程序，首先需要创建一个 Android 工程并且设置好启动配置项。然后你就可以开始编写、运行和调试你的应用程序了。

2.2.1 创建 Android 工程

ADT 插件提供了一个创建新工程的向导以使你能够快速的创建一个新的 Eclipse 工程或者为现有代码创建一个新工程。按照以下步骤来创建一个工程：

1. 选择 File > New > Project
2. 选择 Android > Android Project，然后点击 Next
3. 在 Content 中选择 Project

选择 Create new project in workspace 以创建一个新的工程。 键入工程的名字，你程序的名称，base package 的名字，Activity class 的名字来创建一个 stub .java 文件

选择 Create project from existing source 从现有代码创建一个工程。通过这种方式，你可以编译运行 SDK 里面 sample 中的应用程序。sample 应用程序在 SDK 安装目录下的 samples/中，找到包含有现有代码的路径并点击 OK。如果路径中含有一个有效的 Android 的 manifest 文件，ADT 插件就会加载包名，程序名，activity 名

4. 点击 Finish

ADT 插件会根据你的工程的类型创建以下合适的文件夹和文件：

src/文件夹包含 stub .java activity file

res/文件夹包含你的资源文件

AndroidManifest.xml 是你工程的 manifest

2.2.2 导入工程

如果你的程序已位于工作环境（WorkSpace）资料夹下，想使用上述方法打开工程时，会得到欲打开的工程已在工作目录下的警告。因此我们得用另一个方法：导入工程。

在屏幕上方的菜单列上，选择「File->Import」选项，会跳出「Import」视窗，选择「General->Existing Projects into WorkSpace」项目，然后按下「Next」按钮带到新一个画面。在「Select Root Directory」栏旁，按下右方的「Browse...」按钮，选择对应的工程。选择好后，按下「Finish」按钮完成从现存在工作环境（WorkSpace）资料夹下的工程导入到 Eclipse 环境的动作。

2.2.3 修复工程

完成新增程序工程到 Eclipse 后，我们可以在左侧的「Package Explorer」中找到我们新增的项目。

如果发现开启后的文件夹图示上有个小小的黄色惊叹号，表示这个项目汇入后还有些问题，我们可以使用 ADT 内建的功能来试着修复项目属性。 在「Package Explorer」的「NotesList」 项目文件夹图示上点选右键，从「Android Tools」选单中选择「修复项目属性」（Fix Project Properties）。 （Android Tools->Fix Project Properties）

2.2.4 创建启动配置

要想在 Eclipse 下运行并调试你的程序，你需要先创建启动配置。启动配置中指定了要启动的工程，要用的模拟器选项等等。

要给程序创建启动配置，按照如下步骤：

1. 选择 Run > Open Run Dialog... 或者 Run > Open Debug Dialog...
2. 在左侧工程类型的列表中，右键点击 Android Application 并且选择 New
3. 给你的配置键入一个名字
4. 在 Android 的 Tab 页上，找到用于开始的工程和 Activity
5. 在模拟器 Tab 页上，设置需要的屏幕和网络属性，和其他一些模拟器启动选项
6. 在 Common 的 Tab 页上，你可以对一些附加的选项进行设置
7. 点击 Apply 保存启动配置，或者，点击 Run 或者 Debug

2.2.5 运行和调试程序

当你建立工程，并为你的程序设置好启动配置后，按照下面的步骤你就可以运行或者调试你的程序了。

在 Eclipse 的主菜单上选择 Run > Run 或者 Run > Debug，来运行或者调试可用的启动配置。需要注意的是可用的启动配置默认的是最近一次在启动配置中选择的那一个。并不一定是对应着你在 Eclipse 打开的那个程序。

要设置或者更改启动配置，你可以通过运行配置管理器即点击 Run > Open Run Dialog... 或者 Run > Open Debug Dialog...

运行或者调试应用程序会引发以下的动作 ：

启动模拟器

编译工程，并将程序安装在模拟器上

Run 启动程序

Debug 启动程序进入“等待调试器”模式，然后打开调试视图并连接至 Eclipse 的 Java 调试器

第三节 利用其他的开发环境和工具开发 Android 应用程序

2.3 利用其他的开发环境和工具开发 Android 应用程序

推荐使用带有 Android 插件的 Eclipse 来开发 Android 应用程序，ADT 插件提供了编辑、编译、调试功能并集成进了 IDE 中。然而，SDK 中包含了可以让你用其他 IDE 开发的工具，包括 intelliJ

2.3.1 创建 Android 工程

Android 的 SDK 中包含了一个程序 activityCreator，它可以为你的工程生成 stub 文件和 build 文件。你可以利用这个程序来创建新的工程或者从现有代码创建工程（比如 SDK 中的示例程序）。对于 Linux 和 Mac，SDK 提供了一个 python 的脚本 activityCreator.py；对于 Windows，相应的提供了 activityCreator.bat 这个批处理的脚本。不管是哪种平台，你都可以以同样的方式操作 activityCreator。按照下面的步骤来运行 activityCreator 并创建一个 Android 工程：

在命令行，进入 SDK 安装目录下面的 tools/目录，为你的工程文件创建一个新的目录。如果你是从现有代码创建工程，那么转到你的程序所在的根目录

运行 activityCreator.在命令行，你需要指明一个 full-qualified 类的名称作为参数。如果你是创建一个全新的项目，脚本创建的 stub 类的名称即使所要求的 full-qualified 的类的名称。如果你是从现有代码创建工程，需要指定包中的一个 Activity class 的名称。脚本的可选命令参数包含：

--out <folder>，设定输出目录。默认的输入目录就是当前目录。如果要为你的工程文件创建一个新的目录，使用这个来指定

--ide intellij，对新建的工程生成 IntelliJ IDEA 的工程文件

下面是一个例子：

```
~/android_linux_sdk/tools$ ./activityCreator.py --out myproject your.package.name.ActivityName
package: your.package.name
out_dir: myproject
activity_name: ActivityName
~/android_linux_sdk/tools$
```

activityCreator 脚本生成如下的文件和目录（不会覆盖已有的目录和文件）

AndroidManifest.xml:应用程序的 manifest 文件，与指定的 Activity class 是同步的

build.xml:用来编译和打包应用程序的 Ant 文件

*src/your/package/name/ActivityName.java:在你输入时指定的的 Activity class

your_activity.iml，your_activity.ipr，your_activity.iws:intelliJ 工程文件

res/:资源文件的目录

src/:源代码目录

bin/:编译脚本的输出目录

现在可以将你的目录移动到任何你想开发的地方，但是需要注意的是你只能通过 tools 目录下的 adb 程序来将文件发送到模拟器，因此，你需要保证你的解决方案和 tools/目录之间是可以互相读取的。

同时，尽量不要移动 SDK 开发包的位置，因为这将对 build 脚本产生影响（要使之能够重新运行你需要手动更新这些脚本来 reflect the new SDK location）

2.3.2 编译 Android 程序

利用 activityCreator 生成的 Ant build.xml 文件来编译你的程序。

如果你尚未安装 Ant，可以去 Apache Ant 的主页下载。安装 Ant，并确认它在你的可执行路径上，在运行 Ant 前，你需要声明 JAVA_HOME 环境变量来指定 JDK 的安装路径

如果你还没有做这些，请按照上述的说明来创建一个新的工程。

现在你可以在 build.xml 文件所在的目录下启动 ant 来运行 ant build file。每次你对源程序或者资源文件做出了改动的话，你需要重新运行 ant，它会将程序的最新版本打包使之得以部署

提示：在 Windows 上安装 JDK 的时候，默认安装在 Program Files 文件夹下。由于名字中有空格会导致 ant 无法启动。要解决这个问题，你可以这样来指定 JAVA_HOME 变量：JAVA_HOME=c:\Prora~1\Java\。

还有一种最简单的解决方法就是将 JDK 安装在一个目录中不含有空格的路径下，例如：c:\java\jdk1.6.0_02

2.3.3 运行 Android 程序

要运行已经编译好的程序，请按照下面的步骤，通过 adb 工具将.apk 文件上载到模拟器下面的/data/app/路径中：

启动模拟器（从命令行运行： <你的 SDK 安装目录>/tools/emulator）

在模拟器中，切换到主屏幕（在你将程序重新安装到模拟器的过程中，最好先让程序停止运行；点击 Home 键从那个程序中切换出来）

在命令行输入： adb install myproject/bin/<appname>.apk 将其上载至模拟器。例如，要安装示例程序中的 Lunar Lander，则在命令行下输入： /adb install ../sample/LunarLander/bin/LunarLander.apk

在模拟器中，打开可用的程序列表，选择你的程序并启动

提示：当你第一次安装一个 Activity 时，你可能需要重新启动模拟器。因为包管理器通常只是在模拟器启动的时候才完全检测 manifests。

2.3.4 将调试器关联到应用程序

这一节介绍如何在屏幕上显示调试信息（如 CPU 利用率）以及如何在 IDE 中调试在模拟器上运行的程序 Eclipse 的插件已经自动关联到了调试器，但是对于其他的 IDE，你可以配置其监听调试端口来接收调试信息。

1. 启动 Dalvik Debug Monitor Server（DDMS） tool，这个工具在你的 IDE 和模拟器之间起到一个转发服务的作用

2. 在模拟器上设置可选的调试配置项，例如在关联到调试器之前阻止应用程序启动为一个运行实例。要指出的是许多此类的调试选项都可以在没有 DDMS 的情况下使用，例如在模拟器上显示 CPU 使用率或者屏幕刷新率等

3. 配置你的 IDE 关联到 8700 端口来调试。这部分内容详见如何设置 Eclipse 来调试 Android 程序

2.3.5 配置 IDE 关联到调试端口

DDMS 将为它在模拟器上搜索到的每一个虚拟机分配一个指定的调试端口。你需要将 IDE 关联到这个指定的端口（在该虚拟机的 Info Tab 页列出），或者你可以使用默认的 8700 端口来连接到任何一个在已找到的虚拟机列表选中的程序。

你的 IDE 关联到正在模拟器上运行的程序，可以显示程序的线程并且允许你中止，检测其运行状态和设置断点。如果你在开发设置面板里选择了“等待调试器”，应用程序只有在 Eclipse 连接上的时候才会运行，因此你需要在连接上之前设定好断点。

对正在进行调试的应用程序进行了改动或者改动了“等待调试器”选项将使系统杀死选择的程序（如果该程序正在运行），你可以用这种方法来杀死一个状态异常的程序。

第四节 调试

2.4 调试

Android 有数量众多的工具用于支持调试程序：

DDMS－DDMS 是一个图形化的程序，支持端口转发（因此你可以在程序中设置断点），支持模拟器上的截屏，支持线程和堆栈信息和其他的一些特性。

logcat－Dump 一份系统消息的日志。这些消息包括模拟器抛出错误时的堆栈跟踪。要

运行 logcat，点击链接获取更详细的信息。

...

I/MemoryDealer（ 763）: MemoryDealer （this=0x54bda0）: Creating 2621440 bytes heap at 0x438db000

I/Logger（ 1858）: getView（） requesting item number 0

I/Logger（ 1858）: getView（） requesting item number 1

I/Logger（ 1858）: getView（） requesting item number 2

D/ActivityManager （ 763 ） : Stopping: HistoryRecord{409dbb20 com.google.android.home.AllApps}

...

Android Log－一个记录日志的类，用来将消息写入模拟器上的日志文件中。如果你在 DDMS 上运行 logcat 的话你可以就实时查看消息。在你的代码中加入几个写日志方法的调用。

为了使用 Log 类，你只需要调用

Log.v（）（verbose），Log.d（）（debug），Log.i（）（information），Log.w（）（warning）或者

Log.e（）（error），根据你想获得的日志信息来选择相应的方法

Log.i（"MyActivity"， "MyClass.getView（） — Requesting item number " + position）

你可以用 logcat 来读取这些信息

Traceview－Android 可以保存一个日志用来记录被调用的方法以及该方法被调用的次数，通过 Traceview 你可以在一个图形化的界面中查看这个日志文件。点击链接获取更详细的信息。

Eclipse 插件－Eclipse 的 Android 插件合并了一系列的调试工具（ADB，DDMS，logcat output 和其他一些函数）。点击链接查看更多详情。

Debug and Test Device Settings－Android 提供了一些设置用来获取诸如 CPU 使用率和帧速率等有用的信息。具体请看下面的设备上调试和测试的设置

如果想知道为什么你程序无法在模拟器上显示或者为什么程序没有启动，请查看常见问题这一节

第五节 设备上调试和测试的设置

2.5 设备上调试和测试的设置

Android 提供了众多的设置使你可以更容易的调试和测试程序。要进入开发设置页面，在模拟器中转到 Dev Tools > Development Settings。在该设置页面有以下选项：

Debug app：选择要调试的程序。你不需要设定其关联至调试器，但是设定这个值有两个效果：

在调试的时候，如果你在一个断点处暂停了过长的时间，这个设定会防止 Android 抛出一个错误

这个设定使你可以选择“等待调试器”选项，使程序只有在调试器关联上之后才启动

Wait for Debugger：阻塞所选的程序的加载直到有调试器关联上，这样你就可以在 onCreate（）中设置断点，这对于调试一个 Activity 的启动进程是非常重要的。当你对该选项进行了更改，任何正在运行的程序的实例都会被终止。你只有在上面的选项中选择了一个调试程序才能够选中该选项。你也可以在代码中添加 waitForDebugger（）来实现同样的功能。

Immediately destroy activities：告诉系统一旦一个 activity 停止了就销毁该 activity（例如当 Android 释放内存的时候）。这对于测试代码 onFreeze（Bundle）/onCreate（android.os.Bundle）是非常有用的，否则会比较困难。如果你的程序没有保存状态，那么选择这个选项很可能会引发很多问题。

Show screen updates：对于任何正在被重绘的 screen sections 都会在其上闪现一个粉红色的矩形。这对于发现不必要的 screen 绘制是很有必要的。

Show CPU usage：在屏幕上方显示 CPU 信息，显示有多少 CPU 资源正在被使用。上方红色条显示总的 CPU 使用率，它下方绿色的条显示 CPU 用在 compositing the screen 上的时间。注意：在没有重启模拟器之前，一旦你开启了该功能就不能关闭。

Show screen FPS：显示当前的帧率。这对于查看游戏达到的总的帧率是非常有用的。注意：在没有重启模拟器之前，一旦你开启了该功能就不能关闭。

Show background：当没有 activity screens 可见时，显示一个背景模式。一般是不会出现的，仅仅在 Debug 的时候会出现。

设定的选项在模拟器重启之后仍然有效，如果要取消设定的选项，在取消设定以后还要重启模拟器，才能生效。

第六节 重要的调试小提示

2.6.1 快速的堆栈 dump

在拟器上获得一个堆栈 dump，你可以通过 adb shell 登入，用“ps”找到你想要的进程，然后“kill -3”，堆栈跟踪信息就会记录到日志文件中了。

2.6.2 在模拟器屏幕上显示有用信息

设备上可以显示诸如 CPU 利用率或者对重绘区域的边缘高亮显示等有用信息，在开发设置窗口可以打开或者关闭这些功能，详细信息参见[安装 SDK#设备上调试和测试的设置|在模拟器上设置调试和测试配置]

2.6.3 从模拟器上获取系统状态信息（dumpstate）

你可以通过 Dalvik Debug Monitor Service 工具来获得 dumpstate 信息。参见 adb 章节的 dumpsys and dumpstate

2.6.4 从模拟器上获取程序状态信息（dumpsys）

你可以通过 Dalvik Debug Monitor Service 工具来获得 dumpsys 信息。参见 adb 章节的 dumpsys and dumpstate

2.6.5 获取无线连接信息

你可以通过 Dalvik Debug Monitor Service 工具来获得无线连接信息。在 Device 菜单，选择“Dump radio state”

2.6.6 日志记录跟踪数据

你可以在一个 activity 中通过调用 android.os.Debug.startMethodTracing（）来用日志来记录方法调用和其他跟踪数据。详细内容请参见 Running the Traceview Debugging Program

2.6.7 日志记录 Radio Data

默认情况下，radio 信息是不会记录在系统中的（因为数据量巨大）。然而，你可以通过下面的命令来开启 radio 记录

adb shell

logcat -b radio

2.6.8 运行 adb

Andoid 中自带了一个叫 adb 的工具，该工具功能强大，可以移动并同步文件到模拟器，

转发端口。在模拟器上运行一个 UNIX shell。详细内容参见 Using adb

2.6.9 从模拟器上获取屏幕截图

Dalvik Debug Monitor Server （DDMS）可以从模拟器上获取屏幕截图

2.6.10 利用调试帮助类

Android 为了开发者的方便提供了诸如 util.Log 和 Debug 等帮助类

2.6.11 切换模拟器布景

在菜单列上，选择「Run-> Debug Configurations...」，进入运行环境参数设定界面。在参数设定界面中，选择 Target 标签，其中「Screen Size:」栏中会列出所有支持的模拟器布景。预设有「HVGA」与「QVGA」可以选择，「HVGA」与「QVGA」又可以再各自分为「-L」（landscape， 横式） 与「-P」（portrait 竖式）。

2.6.12 切换屏幕

在 Windows 操作系统，NumLock 后，按下数字键 7，或是在 Mac OS X 操作系统上同时按下「fn」 和「7」键，屏幕就会从预设的直式显示改成横式显示，再按一次则切换回原来的直式显示。

2.6.13 新增模拟器布景

Android 的模拟器也允许使用者自行制作模拟器布景，以世界第一台发售的 Android 手机「T-Mobile G1」为例，到 http://www.jsharkey.org/downloads/G1.zip 下载 G1.zip，要新增模拟器布景时，只需把下载后的模拟器布景解开成一个文件夹，再将文件夹放到「android_sdk/tools/lib/images/skins」目录下。做完后打开 Eclipse 开发环境的菜单列，选择「Run-> Debug Configurations...」，进入运行环境参数设置界面。在参数设置界面中，选择 Target 标签，其中「Screen Size:」栏中所列出的模拟器布景即新增加了「G1」一项（可自行命名）。

2.6.14 移除模拟器布景

要移除一个模拟器布景，直接删除在「android_sdk/tools/lib/images/skins」中的对应目录即可。

2.6.15 移除程序

Android SDK 中提供一个 adb（Android Debugger）命令行工具（在 android-sdk/tools 中），我们可以用里面的 shell 工具连上模拟器来移除应用程序。在某些平台上，这些动作可能需要拥有 root 权限才能执行。

首先启动 adb shell

$ adb shell

接着切换到 data/app 目录中

$ cd data/app/

使用 ls 命令（等同 windows 上命令行的 dir 命令）来查询文件列表

ls

-rw-r--r-- system system 1325833 2007-11-11 20:59 ApiDemos.apk

接着使用 rm 命令来删除 ApiDemos 应用程序

rm ApiDemos.apk

ls

2.6.16 移除映像文件

模拟器的映像文件会保存住你之前安装的程序与操作。所以当你想要回复一个干净的模拟器环境时，你得先移除模拟器的映像文件。

在 Windows 上，模拟器的映像文件会保存在 C:\Documents and Settings\使用者账户目录\Local Settings\Application Data\Android\userdata.img 。

在 Mac 或 Linux 上则会保存在 ~/.android/userdata.img 。

我们可以直接删除，或在模拟器关闭的状况下，以指令

$ rm /.android/userdata.img

来删除模拟器的映像文件。

第七节 编译安装 Android 应用程序

2.7 编译安装 Android 应用程序

Android 需要专门的编译工具来正确的编译资源文件和 Android 程序的其他部分。基于此，你需要为你的程序准备一个专门的编译环境。

Andorid 的编译过程通常包括编译 XML 和其他资源文件、创建正确的输入格式。经过编译的 Android 程序是一个.apk 文件，.apk 文件是一个压缩文件，它其中包含了.dex 文件、资源文件、raw data 文件和其他文件。

Andoriod 暂时还不支持用本地代码（C/C++）开发第三方程序。

强烈推荐[安装 SDK#在 Eclipse 上开发 Android 应用程序 在 Eclipse 上开发 Andoid 应用程序]，通过 Eclipse＋ADT 插件可以支持编译、运行、调试 Andorid 程序。

如果你用其他 IDE，参见[安装 SDK#利用其他的开发环境和工具开发 Android 应用程序 利用其他的开发环境和工具开发 Android 应用程序]来编译和调试 Android 程序，但是他们不是集成在一起的。

第八节 Eclipse 小提示

2.8 Eclipse 小提示

2.8.1 在 Eclipse 上执行 arbitrary Java 表达式

在 Eclipse 中，当程序运行到断点处暂停时，你可以执行 arbitrary code。例如：在一个有一个 String 参数的函数“zip”，你可以获 取包的信息和调用类的方法。你同样可以调用静态方法：例如输入 android.os.Debug.startMethodTracing（）可以开始 dmTrace。

打开一个代码执行窗口，在主菜单中选择 Window > Show View > Display 打开显示窗口，一个简单的文本编辑器。输入你的表达式，高亮显示文本，点击图标“J”（或者 Ctrl+Shift+D）来运行你的代码。运行在所选择现成的上下文中的代码，肯定会停留在断点出或者单步执行点。（如果你手动挂起线程，那么单步执行一次，如果线程处于 Object.wait（）状态，这将不会起作用）

如果你此时在一个断点处暂停，你可以简单通过 Ctrl+Shift+D 来高亮并执行一段代码。 你也可以通过按下 ALT +SHIFT + UP ARROW 在同一个范围内高亮显示一段文本块，或者 DOWN ARROW 选择小一些的文本块。

下面是一些在 Eclipse 中显示窗口中的输入和响应的例子：

Input	Response
zip	（java.lang.String） /work/device/out/linux-x86-debug/android/app/android_sdk.zip
zip.endsWith（".zip"）	（boolean） true
zip.endsWit	（boolean） false

h（".jar"）	

通过 scrapbook，即使不处于 debug 状态，你也可以执行 Arbitrary code。在 Eclipse 文档中搜索 "scrapbook"，可以了解更多细节。

2.8.2 手动运行 DDMS

尽管调试推荐的方式是利用 ADT 插件，你仍然可以通过手动运行 DDMS 并配置 Eclipse 在 8700 端口调试。（注意：确定你已经启动了 DDMS）

第三章

第一节 创建 Android 应用程序

本章将专门介绍创建一般性的 Android 应用程序。

3.1 剖析 Android 应用

Android 应用有四个组件构成：

- Activity
- Intent Receiver
- Service
- Content Provider

并不是每一个应用都需要以上全部四个功能块，但是你的应用将使用上述四种功能块的某些组合。

一旦你决定在你应用程序中使用上述组件，你应在将他们列在 AndroidManifest.xml 文件中。AndroidManifest 文件是一个 XML 文件，其中定义了你的应用包含的组件以及他们的能力和要求。关于 AndroidManifest.xml 的职能将在后面讨论。

3.1.1 Activity

Activity 是 Android 构造块中最基本的一种，在应用中，一个 activity 通常就是一个单独的屏幕。每一个 activity 都被实现为一个独立的类，并且继承于 Activity 这个基类。这个 activity 类将会显示由几个 Views 控件组成的用户接口，并对事件做出响应。大部分的应用都会包含多个的屏幕。例如，一个短消息应用程序将会有一个屏幕用于显示联系人列表，第二个屏幕用于写短消息，同时还会有用于浏览旧短消息及进行系统设置的屏幕。每一个这样的屏幕，就是一个 activity。从一个屏幕导航到另一个屏幕是很简单的。在一些应用中，一个屏幕甚至会返回值给前一个屏幕。例如，一个允许用户选中的图片将把选中的图片返回给调用者。当一个新的屏幕打开后，前一个屏幕将会暂停，并保存在历史堆栈中。用户可以返回到历史堆栈中的前一个屏幕。当屏幕不再使用时，还可以从历史堆栈中删除。默认情况下，Android 将会保留从主屏幕到每一个应用的运行屏幕。

3.1.2 Intent Receiver

当你希望你的应用能够对一个外部的事件（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，你可以使用一个 Intent Receiver。 虽然 Intent Receiver 在感兴趣的事件发生时，会使用 NotificationManager 通知用户，但它并不能生成一个 UI。

Intent Receiver 在 AndroidManifest.xml 中注册，但也可以在代码中使用 Context.registerReceiver（）进行注册。当一个 intent receiver 被触发时，你的应用不必对请求调用 intent receiver，系统会在需要的时候启动你的应用。各种应用还可以通过使用 Context.broadcastIntent（）将它们自己的 intent receiver 广播给其它应用程序。

3.1.3 Service

一个 Service 是一段长生命周期的，没有用户界面的程序。比较好的一个例子就是一个正在从播放列表中播放歌曲的媒体播放器。在一个媒体播放器的应用中，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，因为使用者当然会认为在导航到其它屏幕时音乐应该还在播放的。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，我们还可以通过使用 Context.bindService（）方法，连接到一个 service 上（如果这个 service 还没有运行将启动它）。当连接到一个 service 之后，我们还可以 service 提供的接口与它进行通讯。拿媒体播放器这个例子来说，我们还可以进行暂停、回放等操作。

3.1.4 Content Provider

应用程序能够将它们的数据保存到文件中、SQL 数据库中，甚至是任何有效的设备中。当你想将你的应用数据与其它的应用共享时，Content Provider 将会很有用。一个 Content Provider 类实现了一组标准的方法，从而能够让其它的应用保存或读取此 Content Provider 处理的各种数据类型。

更详细的 Content Provider 资料，可以参考附带文档中的 Accessing Content Providers。

第二节 Android 用户界面

3.2 Android 用户界面

这个章节描述怎么实现一个基本的 Android 界面。它涉及构建屏幕基本元素，怎么在 xml（定义文件）内定义屏幕、用你的代码生成、在不同任务你需要操作你的用户接口。Android

生成屏幕有三种方式：xml 配置生成；通过你自己用户界面接口生成；直接用代码生成。根据 MVC 原则，UI 应该与程序逻辑相分离，因此，在 XML 中定义 UI 结构是高度推荐的。此外，一个程序从一个屏幕方案调整到另一个也容易得多。在 XML 中定义 UI 跟创建一个普通的 HTML 文档非常相似，例如，你有如下的一个文件：

```
<html>
<head>
<title>Page Title</title>
</head>
<body>
The content of the body element.
</body>
</html>
```

就如 Android 的 XML 布局一样，所有的元素都是结构化的，能够通过树形结构来表示：

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout   xmlns:android=http://schemas.android.com/apk/res/android
        android:orientation="vertical"
       android:layout_width="fill_parent"
       android:layout_height="fill_parent">
       <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Hello World"/>
</LinearLayout>
```

3.2.1 屏幕元素的层次

Android 应用程序的基础功能单元就是 Activity——android.app.Activity 类中的一个对象。一个 Activity 可以做很多事，但是他自己并不会显示到屏幕上。想要让你的 Activity 显示在屏幕上并且设计它的 UI，你需要使用 view 和 viewgroup——Android 平台基础的用户界面表达单元。

Views

一个 view 是一个 android.view.View 基础类的对象。它是一个存储有屏幕上特定的一个

矩形内布局和内容属性的数据结构。一个 View 对象处理测距和布局，绘图，焦点变换，滚动条，还有屏幕区域自己表现的按键和手势。

View 类作为一个基类，为 widget（窗体部件）服务，widget——是一组用于绘制交互屏幕元素的完全实现子类。Widget 处理它们自己的测距和绘图，所以你可以更快速地用它们去构建你的 UI。可用到的 widget 包括 Text，EditText，InputMethod，Button，RadioButton，Checkbox 和 ScrollView……。

Viewgroups

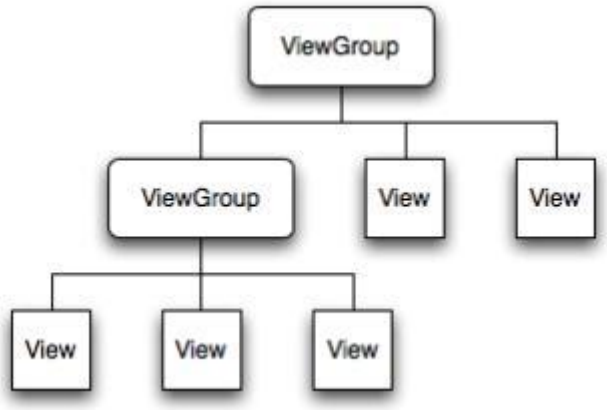
一个 ViewGroup 是一个 android.view.Viewgroup 类的对象。正如同它的名字表明的一样，一个 viewgroup 是一个特殊的 view 对象，它的功能是去装载和管理一组下层的 view 和其他 viewgroup，Viewgroup 让你可以为你的 UI 增加结构并且将复杂的屏幕元素构建成一个独立的实体。

Viewgroup 类作为一个基类为 layout（布局）服务，layout——是一组提供屏幕界面通用类型的完全实现子类。layout 让你可以为一组 view 构建一个结构。

一个树形结构的界面

在 Android 平台上，你用 view 树和 viewgroup 节点来定义一个 Activity 的 UI，就如同下面图表一样。这个树可以如你需要那样简单或者复杂，并且你可以使用 Android 的预定义 widget 和 layout 或者你自定义的 view 类型来构建它。

一个 view 和 viewgroup 树的样例:



Picture 4 Android UI - Tree structure

要将屏幕绑定一个树以便于渲染，你的 Activity 调用它的 setContentView（）方法并且传递一个参数给根节点对象。一旦 Android 系统获得了根节点的参数，它就可以直接通过节点来无效化、测距和绘制树。当你的 Activity 被激活并且获得焦点时，系统会通知你的 activity

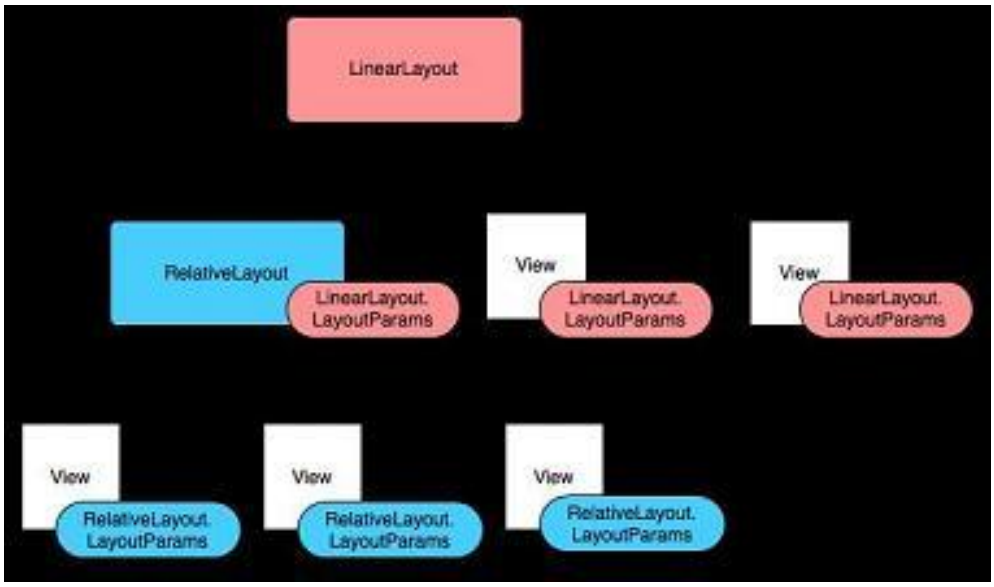
并且请求根节点去测距并绘制树，根节点就会请求它的子节点去绘制它们自己，同时，每个树上的 viewgroup 节点负责绘制它的直接子节点。

正如之前提到的，每个 viewgroup 都有测量它的有效空间，布局它的子对象，并且调用每个子对象的 Draw（）方法去绘制它们自己。子对象可能会请求获得一个它们在父对象中的大小和位置，但是父对象对于每个子对象的大小和位置有最终的决定权。

LayoutParams：一个子对象如何指定它的位置和大小

每个 viewgroup 类都会使用一个继承于 Viewgroup.LayoutParams 的嵌套类。这个子类包含了一系列的属性类型，这些属性类型定义一个子对象位置和大小，与 viewgroup 类相适应。

layoutparams 的一个样例：



要注意的是，每个 LayoutParams 子类都有它自己赋值的语法。每个子元素必须定义适用于它们父对象的 LayoutParams，尽管父对象可能会为子元素定义不同的 LayoutParams。

所有的 viewgroup 都包括宽和高。很多还包括边界的定义（margin 和 border）。你可以非常精确地描述宽和高，尽管你并不想经常这么做。更多时候你希望你的 view 自行调整到适应内容大小，或者适应容器大小。

Android 界面元素与 Swing 界面元素的比较

Android 界面元素	Swing 界面元素
Activities	Frame
Views	Components
TextViews	Labels
EditTexts	TextFields

Buttons	Buttons

Android 和 Swing 的监听者设置也几乎一样：

3.2.2 通用布局对象

下面为在你的应用中为最普遍的 viewgroups。这里介绍每种类型的一些基本信息；更深入的细节，请看每章前面的链接参考页。

FrameLayout

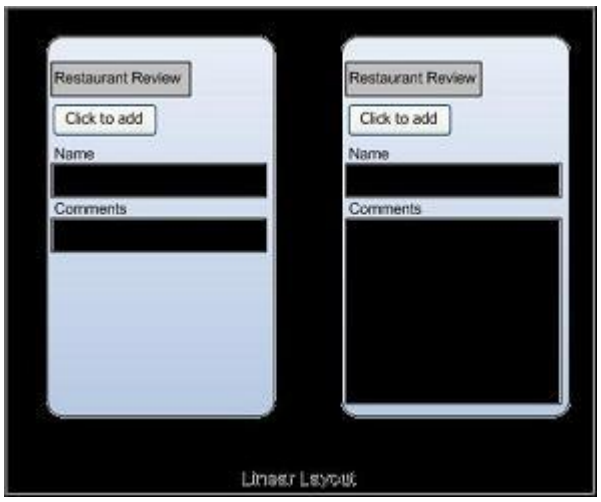
FrameLayout 是最简单的一个布局对象。它被定制为你屏幕上的一个空白备用区域，之后你可以在其中填充一个单一对象——比如，一张你要发布的图片。所有的子元素将会固定在屏幕的左上角；你不能为 FrameLayout 中的一个子元素指定一个位置。后一个子元素将会直接在前一个子元素之上进行覆盖填充，把它们部分或全部挡住（除非后一个子元素是透明的）。

LinearLayout

LinearLayout 以你为它设置的垂直或水平的属性值，来排列所有的子元素。所有的子元素都被堆放在其它元素之后，因此一个垂直列表的每一行只会有一个元素，而不管他们有多宽，而一个水平列表将会只有一个行高（高度为最高子元素的高度加上边框高度）。LinearLayout 保持子元素之间的间隔以及互相对齐（相对一个元素的右对齐、中间对齐或者左对齐）。

LinearLayout 还支持为单独的子元素指定 weight。好处就是允许子元素可以填充屏幕上的剩余空间。这也避免了在一个大屏幕中，一串小对象挤成一堆的情况，而是允许他们放大填充空白。子元素指定一个 weight 值，剩余的空间就会按这些子元素指定的 weight 比例分配给这些子元素。默认的 weight 值为 0。例如，如果有三个文本框，其中两个指定了 weight 值为 1，那么，这两个文本框将等比例地放大，并填满剩余的空间，而第三个文本框不会放大。

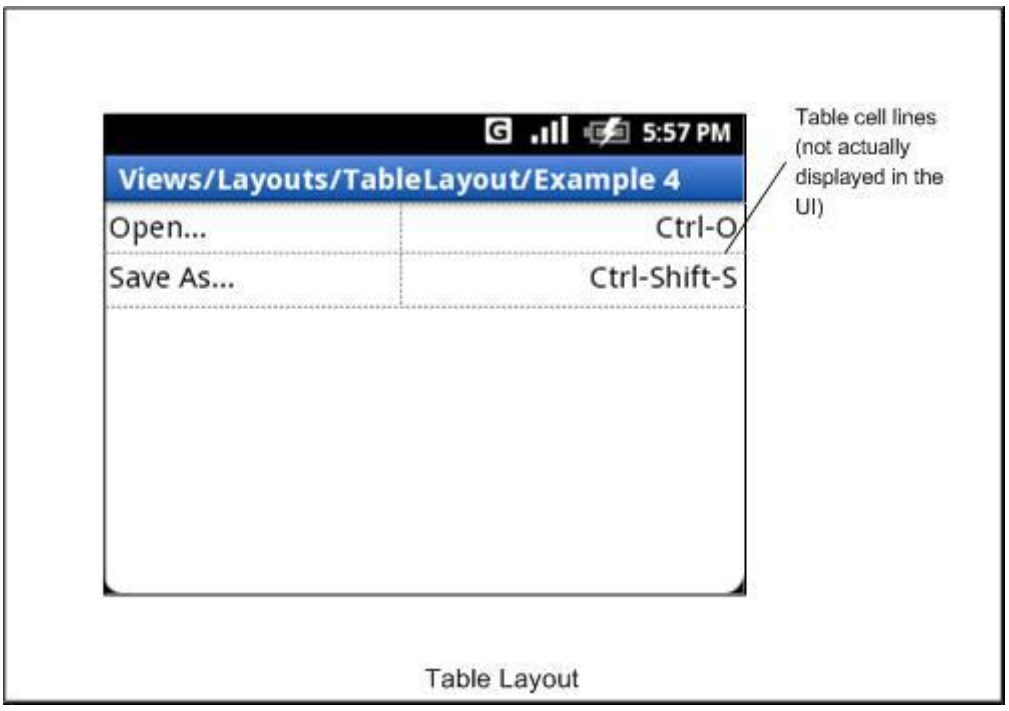
下面的两个窗体采用 LinearLayout，包含一组的元素：一个按钮，几个标签，几个文本框。两个窗体都为布局做了一番修饰。文本框的 width 被设置为 FILL_PARENT；其它元素的 width 被设置为 WRAP_CONTENT。默认的对齐方式为左对齐。左边的窗体没有设置 weight（默认为 0）；右边的窗体的 comments 文本框 weight 被设置为 1。如果 Name 文本框也被设置为 1，那么 Name 和 Comments 这两个文本框将会有 同样的高度。



在一个水平排列的 LinearLayout 中，各项按他们的文本基线进行排列（第一列第一行的元素，即最上或最左，被设定为参考基线）。因此，人们在一个窗体中检索元素时，就不需要七上八下地读元素的文本了。我们可以在 layout 的 XML 中设置 android:baselineAligned="false"，来关闭这个设置。

TableLayout

TableLayout 将子元素的位置分配到行或列中。android 的一个 TableLayout 由许多的 TableRow 组成，每个 TableRow 都会定义一个 row（事实上，你可以定义其它的子对象，这在下面会解释到）。TableLayout 容器不会显示 row、cloumns 或 cell 的边框线。每个 row 拥有 0 个或多个的 cell；每个 cell 拥有一个 View 对象。表格由列和行组成许多的单元格。表格允许单元格为空。单元格不能跨列，这与 HTML 中的不一样。下图显示了一个 TableLayout，图中的虚线代表不可视的单元格边框。



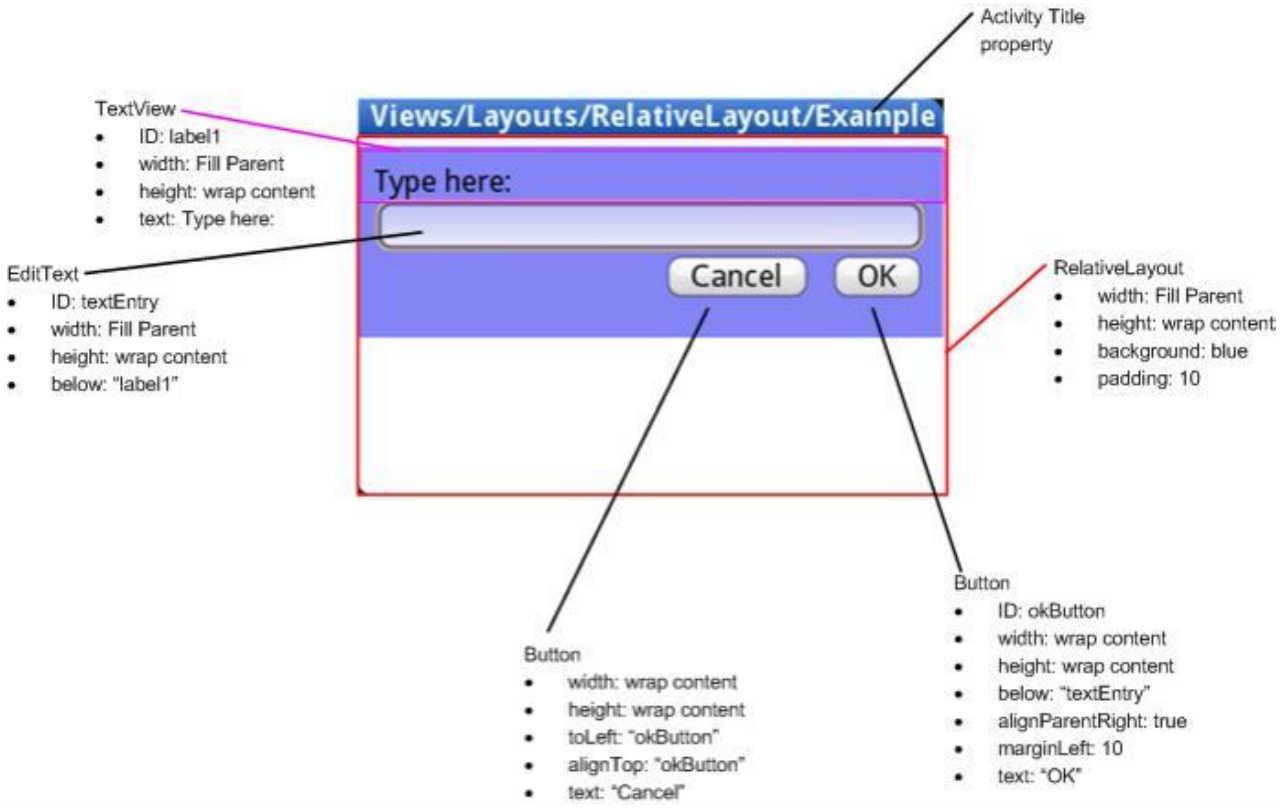
列可以被隐藏，也可以被设置为伸展的从而填充可利用的屏幕空间，也可以被设置为强制列收缩直到表格匹配屏幕大小。对于更详细信息，可以查看这个类的参考文档。

AbsoluteLayout

AbsoluteLayout 可以让子元素指定准确的 x/y 坐标值，并显示在屏幕上。（0， 0）为左上角，当向下或向右移动时，坐标值将变大。AbsoluteLayout 没有页边框，允许元素之间互相重叠（尽管不推荐）。我们通常不推荐使用 AbsoluteLayout，除非你有正当理由要使用它，因为它使界面代码太过刚性，以至于在不同的设备上可能不能很好地工作。

RelativeLayout

RelativeLayout 允许子元素指定他们相对于其它元素或父元素的位置（通过 ID 指定）。因此，你可以以右对齐，或上下，或置于屏幕中央的形式来排列两个元素。元素按顺序排列，因此如果第一个元素在屏幕的中央，那么相对于这个元素的其它元素将以屏幕中央的相对位置来排列。如果使用 XML 来指定这个 layout，在你定义它之前，被关联的元素必须定义。这是一个 RelativeLayout 例子，其中有可视的和不可视的元素。基础的屏幕 layout 对象是一个 RelativeLayout 对象。



这个视图显示了屏幕元素的类名称，下面是每个元素的属性列表。这些属性一部分是由元素直接提供，另一部分是由容器的 LayoutParams 成员 （RelativeLayout 的子类）提供。RelativeLayout 参数有 width，height，below，alignTop，toLeft，padding 和 marginLeft。注意，这些参数中的一部分，其值是相对于其它子元素而言的，所以才 RelativeLayout。这些参数包括 toLeft，alignTop 和 below，用来指定相对于其它元素的左，上和 下的位置。

Summary of Important View Groups

重要 View Group 摘要

These objects all hold child UI elements. Some provide visible UI， and others only handle c hild layout.

这些对象拥有 UI 子元素。一些提供可视的 UI，另一些只处理子元素的布局。

Class	Description
AbsoluteLayout	可以通过精确的坐标（如屏幕像素）指定子对象相对父容器的位置
FrameLayout	负责显示单一对象的 Layout
Gallery	一个以水平滚动方式显示有序图片列表的显示器
GridView	显示一个可滚动的有 m 列 n 行的表格
LinearLayout	以水平或垂直方式显示子元素的 Layout。如果窗体的长度超过了屏幕的长

	度，将会出现滚动条
ListView	显示一个可滚动的单列列表
PopupList	一个独立的带边框的元素弹出列表
RelativeLayout	能够指定子对象相对于其它对象（如 A 在 B 的左边）或父对象（如在父容器的顶部）的位置
ScrollView	一个垂直的元素滚动列
Spinner	在一个单行文本框中，同时只显示一个有序列表中的一个项。类似于一个可以水平或垂直滚动的单行 listbox
SurfaceView	提供直接访问一个可画图的界面。可以控制在界面顶部的子视图层。SurfaceView 是提供给需要直接画像素而不是使用窗体部件的应用使用的。
TabHost	提供一个页签选择列表，监视点击并在一个页签被点击时保证应用切换屏幕。
TableLayout	一个拥有任意行和列的表格 layout，每一个单元格拥有窗体部分。行会根据最大的列而自动调整大小。单元格边框不可见。
ViewFlipper	一个在单行文本框中同一时刻只显示一项的列表组件。它可以根据时间周期切换显示项，类似一个幻灯机。
ViewSwitcher	类似 ViewFlipper

3.2.3 数据绑定

有些 View groups 会有 UI。这些对象通常是 AdapterView 类的子类。例如包括图库和列表视图，它们具有两个共同的职责：

- 填充布局数据
- 处理用户操作

填充布局数据

填充布局数据通常通过把这个类绑定到一个 Adapter 来完成，Adapter 从某个地方获取它的数据，或者是代码提供的一个列表，或者是来自设备数据库的查询结果。

// Get a Spinner and bind it to an ArrayAdapter that references a String array.

Spinner s1 = （Spinner） findViewById（R.id.spinner1）;

ArrayAdapter adapter = ArrayAdapter.createFromResource（

```
   this， R.array.colors， android.R.layout.simple_spinner_item）；
adapter.setDropDownViewResource（android.R.layout.simple_spinner_dropdown_item）；
s1.setAdapter（adapter）；
// Load a Spinner and bind it to a data query.
private static String[] PROJECTION = new String[] {
      People._ID， People.NAME
   };
Spinner s2 = （Spinner） findViewById（R.id.spinner2）；
Cursor cur = managedQuery（People.CONTENT_URI， PROJECTION， null， null）；

SimpleCursorAdapter adapter2 = new SimpleCursorAdapter（this，
   android.R.layout.simple_spinner_item， // Use a template
                                        // that displays a
                                        // text view
   cur， // Give the cursor to the list adatper
   new String[] {People.NAME}， // Map the NAME column in the
                                      // people database to...
   new int[] {android.R.id.text1}）; // The "text1" view defined in
                                    // the XML template

adapter2.setDropDownViewResource（android.R.layout.simple_spinner_dropdown_item）；
s2.setAdapter（adapter2）；
```

注意：使用 CursorAdapter 时，必须有 People._ID， 否则将会发生异常。

处理用户操作

Android 通过设置类的 AdapterView.OnItemClickListener 成员到一个监听者并捕捉用户的操作事件，来处理用户的操作。

```
// Create a message handling object as an anonymous class.
private OnItemClickListener mMessageClickedHandler = new OnItemClickListener（） {
    public void onItemClick（AdapterView parent， View v， int position， long id）
    {
```

```
        // Display a messagebox.
        showAlert（"You've got an event"， "Clicked me!"， "ok"， false）;
    }
};

// Now hook into our object and set its onItemClickListener member
// to our class handler object.
mHistoryView = （ListView）findViewById（R.id.accept_button）;
mHistoryView.setOnItemClickListener（mMessageClickedHandler）;
```

第三节 数据存储与获取

3.3 数据存储与获取

典型的桌面操作系统提供一种公共文件系统——任何应用软件可以使用它来存储和读取文件，该文件也可以被其它的应用软件所读取（也许会有一些权限控制设定）。Andorid 采用了一种不同的系统：在 Android 上，所有的应用程序数据（包括文件）为该应用程序所私有。然而，Android 同样也提供了一种标准方式供应用程序将其私有数据开放给其它应用程序。这一章节描述一个应用程序存储和获取数据、开放数据给其它应用程序，以及从其他开放他们数据的应用程序请求数据的多种方式。

Android 提供如下机制以存储和获取数据：

3.3.1 参数

存储和检索原始的数据类型的键/值对的轻量级的机制。通常被用来保存程序的参数选择。

使用程序参数

你可以存储应用程序的参数，当程序启动时载入，比如默认问候语或文本字体。调用 Context.getSharedPreferences（）读取和写入参数值，如果你想将参数共享给包内的其它组件，请为参数集指定一个名字，或者使用 Activity.getPreferences（），不给名字以对调用的 activity 保持私有。你不能跨越包共享参数。

这里是一个为计算器静音按键模式设置用户参数的例子。

```
public class Calc extends Activity {
public static final String PREFS_NAME = "MyPrefsFile";
    ...
    @Override
    protected void onCreate（Bundle state）{
      super.onCreate（state）;

    ...

      // Restore preferences
      SharedPreferences settings = getSharedPreferences（PREFS_NAME， 0）;
      boolean silent = settings.getBoolean（"silentMode"， false）;
      setSilent（silent）;
    }

    @Override
    protected void onStop（）{
      super.onStop（）;

      // Save user preferences. We need an Editor object to
      // make changes. All objects are from android.context.Context
      SharedPreferences settings = getSharedPreferences（PREFS_NAME， 0）;
      SharedPreferences.Editor editor = settings.edit（）;
      editor.putBoolean（"silentMode"， mSilentMode）;
      // Don't forget to commit your edits!!!
      editor.commit（）;
    }
}
```

取自"http://www.androidcn.net/wiki/index.php/Devel/building-blocks"

3.3.2 文件

你可以将文件存储在手机或可移动存储媒介中。缺省情况下，其他程序不能访问这些文件

使用文件

Android 提供访问一个程序的本地文件的读取和写入流。调用带本地文件名和路径的 Context.openFileOutput（） 和 Context.openFileInput（）函数去读写文件。其他程序使用同样的文件名和路径字符串去调用这些方法将是无效的，你只能访问本地文件。

如果你有静态文件需要在编译时同程序一起打包，你可以把文件保存在工程的 res/raw/<mydatafile>下，这样就可以通过 Resources.openRawResource（R.raw.mydatafile）来获取它。

R.raw 目录下面所放的文件只能小写字母，如 mydatafile，不可以 MyDataFile。

3.3.3 数据库

Android API 包含有对 SQLite 的支持。你的应用程序可以创建并使用一个私有 SQLite 数据库。每个数据库为创建它的包所私有。

使用 SQLite 数据库

Android 支持 SQLite 数据库系统并开放数据库管理函数，这使你可以将复杂的数据集合储存到有用的对象中。例如，Android 定义了一种由字符串型姓名、字符串型地址、数字型电话号码、一个位图图像和多种其它个人信息描述字段组成的通讯录数据类型。使用 SQLiteOpenHelper 创建一个数据库并适当的读写数据（注意：位图这样的文件数据通常以文件路径字符串值形式存放在数据库中，这个字符串标识了本地文件的位置）

Android 与 sqlite3 数据库工具一起发布，这就使你可以在 SQLite 数据库中浏览表内容、运行 SQL 命令并执行其它有用的函数。参考 Examine databases （sqlite3）学习如何运行这个程序。

SQLite 及其它的所有的数据库，被储存于设备的/data/data/<package_name>/databases 路径下。

创建多少表、包含多少字段、如何连接，已经超越了这篇文档的讨论范围，但是 Android 没有施加任何越过 SQLite 观念的限制。我们极力推荐包含一个作为唯一 ID 的自动增量以便快速查找记录。对于私有数据，这并不需要，但是如果你实现一个 content provider，你必须包含这样的一个唯一 ID 字段。请参考 NotePad 示例程序中的示例类 NotePadProvider.java，那是一个创建和组装新数据库的例子。你创建的任何数据库可以供程序内部的任何一个类凭

借数据库名访问，但不能超出程序范围。

3.3.4 内容提供器

内容提供器是一个开放私有数据读写权限的可选组件，受限于该私有数据想施加的任何约束。内容提供器执行标准的数据请求语法、通过标准的数据访问机制返回数据。Android 提供了大量的针对标准数据类型（如个人通讯录）的内容提供器。

访问 Content Providers

如果希望公开你的数据，你可以创建（或调用）一个 content provider，它是一个所有程序存储和检索可访问数据的一个对象。这也是跨包共享数据的唯一方式——没有供所有包共享数据的公共存储区域。Android 发布了为常见的数据类型（音频、视频、图像、个人通讯录信息等等）发布了大量的 content providers。你可以在 provider 包中看到许多 Android 自带的 content providers。

实际数据储存的方式取决于 content provider 的具体实现，但是所有 content provider 都必须实现一个查询数据和一个返回结果的共同的协定。然而，content provider 可以实现自定义的帮助器函数，使被开放的指定数据更容易被存储/获取。

本文档包括 2 个 content providers 的两个主题：

- 使用一个 content provider（内容提供器）
- 创建一个 content provider（内容提供器）

使用内容提供器存储和获取数据

这一节阐述你或其他任何人如何使用一个内容提供器存储和获取数据。Adnroid 为广泛的数据类型开放了大量内容提供器，从音乐文件、图像文件到电话号码。你可以从 android.provider 包中的类里看到一系列开放的内容提供器列表。

Android 的内容提供器与他们的客户端松散地连接。每一个内容提供器开放一个唯一的字符串（URI）来识别将要操作的数据类型，客户端必须使用该字符串来存储和获取相应类型的数据。在《数据查询》章节中我们将对此做更多解释。

这一节描述下列活动：

- 查询数据
  - o 作出查询
  - o 查询返回什么
  - o 检索文件
  - o 读取获得的数据

- 修改数据
- 添加记录
- 删除记录

查询数据

每一个内容提供器都开放一个唯一公共 URI（由 URI 封装），它将被客户端用于从内容提供器查询/添加/更新/删除数据。URI 有 2 种形式：一是指出该类型数据的所有值（例如所有个人通讯录），二是指出该类型数据的特定记录（例如乔•史密斯的联络信息）

□ □ content://contacts/people/ 是一个可以从设备返回所有通讯录姓名列表的 URI

□ □ content://contacts/people/23是一个可以返回通讯录中 ID=23 的单行记录的 URI 字符串

当程序发送一个查询请求到设备，要求获取一般类型的项（所有电话号码）或特定项（鲍勃的电话号码）。Android 将返回一个包含结果记录集的游标。让我们来看一个假定的查询字符串和结果集（结果已被调整的更清晰一些）。

查询字符串 = content://contacts/people/

结果:

_ID	_COUNT	NUMBER	NUMBER_KEY	LABEL	NAME	TYPE
13	4	（425） 555 6677	425 555 6677	California office	Bully Pulpit	Work
44	4	（212） 555-1234	212 555 1234	NY apartment	Alan Vain	Home
45	4	（212） 555-6657	212 555 6657	Downtown office	Alan Vain	Work
53	4	201.555.4433	201 555 4433	Love Nest	Rex Cars	Home

注意：查询字符串不是一个标准的 SQL 查询字符串，而是一个描述了返回数据类型的 URI 字符串。这个 URI 由 3 部分组成：字符串“content://”；一个描述要检索的数据的类型的段；和一个可选的在特定内容类型的某特定记录项的 ID。下面是一些查询字符串的例子：

content://media/internal/images 返回设备上所有内部图像的 URI 字符串

content://media/external/images 返回设备上的主外存（如 SD 卡）上的图像的 URI 字符串

content://contacts/people/ 返回设备通讯录中所有姓名列表的 URI

content://contacts/people/23 返回通讯录中 ID=23 的单行记录的 URI

虽然有一个一般格式，但是这些查询 URIs 仍然有些专断和费解。因此，Android 在 android.provider 包中提供了一系列的辅助类，这些类定义了这些查询字符串，因此你不需要知道不同的数据类型的实际的 URI。这些辅助类为一个特定的数据类型定义了一个字符串（实际上，是一个 Uri 对象）CONTENT_URI。

通常你将使用已定义的 CONTENT_URI 对象查询，而不是完整的 URI 本身。因此，上面的每一个查询字符串（除了最后一个，指定了记录 ID）能够通过下面的 Uri 获取：

MediaStore.Images.Media.INTERNAL_CONTENT_URI

MediaStore.Images.Media.EXTERNAL_CONTENT_URI

Contacts.People.CONTENT_URI

要查询一个特定的记录 ID（如 content://contacts/people/23），你可以使用同一个 CONTENT_URI， 但是必须追加你需要的特定的 ID 值。这是一个你需要检查或修改 URI 字符串的情况。因此，如果你正在查找通讯录中的记录 23，你可能会如下执行一个查询：

```
// Get the base URI for contact with _ID=23.
// This is same as Uri.parse（"content://contacts/people/23"）;
Uri myPerson = ContentUris.withAppendedId（People.CONTENT_URI， 23）;
// Query for this record.
Cursor cur = managedQuery（myPerson， null， null， null）;
```

Tip: 你也可以使用 withAppendedPath（Uri， String）追加一个字符串到 URI.

这个查询返回一个数据库上的查询结果集的游标。你可以使用 Activity.managedQuery（） 方法来检索一个组织过的游标，一个组织过的游标处理一些细节，如当程序暂停时卸载它自己，当程序重启时，重新检索它自己。通过调用 Activity.startManagingCursor（），你可以要求 Android 组织一个没有组织的游标。

下面来看一个查询的例子，检索一系列联系人的姓名和主要电话号码。

```
// An array specifying which columns to return.
string[] projection = new string[] {
   People._ID，
```

```
    People.NAME，
    People.NUMBER，
};
// Get the base URI for People table in Contacts content provider.
// ie. content://contacts/people/
Uri mContacts = People.CONTENT_URI;

// Best way to retrieve a query; returns a managed query.
Cursor managedCursor = managedQuery（ mContacts，
                projection， // Which columns to return
                null，       // WHERE clause; which rows to return （all rows）
                null，       // WHERE clause selection arguments （none）
                People.NAME + " ASC"）; // Order-by clause （ascending by name）
```

这个查询将从 Contacts content provider 的联系人表中查询数据，它查询名字，主要电话号码和每一个联系人的唯一记录 ID。

查询返回什么

一个查询返回零到多个数据库记录集。对于 content provider，列名，顺序和类型是特定的，但是每一个查询都包含一个称为_id 的列，它是那个项在行中的 ID。如果一个查询返回二进制数据，如位图或音频文件，它将有一列已命名的，保存一个 content:// URI，你可以使用它去获取具体的数据。下面是上一个查询的结果：

_id	name	number
44	Alan Vain	212 555 1234
13	Bully Pulpit	425 555 6677
53	Rex Cars	201 555 4433

结果集展示了我们指定的需要返回的列的子集。可选择的结果集列表在查询的 projection 参数里被传递。一个内容管理器应该列出它所支持的哪些列，通过实现一批描述每一列的接口（参考 Contacts.People.Phones， 实现

了 BaseColumns ， PhonesColumns ， 和 PeopleColumns），或者以常量方式列出这些列名。注意，你需要知道 content provider 公开的列的数据类型以便读取它，域的读取方法与数据类型相关，且列的数据类型不在程序中公开。

检索到的数据通过一个游标访问，游标可以在结果集中前后移动。你可以使用这个游标，读取，修改或者删除行。添加新行需要另一个对象，后面将会讨论。

根据惯例，每一个记录集包含一个_id 域，特定记录的 ID，和一个_count 域，当前结果集的记录数。这些域名定义在 BaseColumns 中。

检索文件

之前的查询结果已经表明了如果返回一个结果集中的文件，这个文件域通常（但不是必须的）是这个文件的路径字符串。然而，调用者不能尝试直接读取和打开一个文件（权限问题）。因此，你应该调用 ContentResolver.openInputStream （） / ContentResolver.openOutputStream（）或者 content provider 的辅助函数。

读取检索到的数据

查询获取的游标对象提供了对结果集的访问，如果你通过 ID 查询一条特定的记录，这个数据集将只包含一条记录，反之，它可能包含多条记录。你可以读取记录中特定域的数据，但是你必须知道这些域的数据类型，因为读取数据时，不同数据类型使用不同的函数（如果你在多列上调用字符串读取方法，Android 将给你字符串表示的数据）。游标让你通过索引获取列名，或者通过列名获取索引。

如果你读取二进制数据，如一个图片文件，你应该在存储在列名上的字符串 content:// URI 上调用 ContentResolver.openOutputStream（） 。

下面的代码段展示了从电话号码查询中读入姓名和电话号码：

```
private void getColumnData（Cursor cur）{
  if （cur.moveToFirst（）） {
    String name;
    String phoneNumber;
    int nameColumn = cur.getColumnIndex（People.NAME）;
    int phoneColumn = cur.getColumnIndex（People.NUMBER）;
    String imagePath;

    do {
```

```
        // Get the field values
        name = cur.getString（nameColumn）;
        phoneNumber = cur.getString（phoneColumn）;

  // Do something with the values.
        ...
    } while （cur.moveToNext（））;
  }
}
```

修改数据

调用带有要改变的列和值的 ContentResolver.update（）方法，可以批更新一组记录（如，改变所有联系人域的"NY" 到 "New York"）。

增加一条新记录

调用 ContentResolver.insert（） 方法，可以添加一条新记录，这个函数可以带有要添加的项类型的 URI 和你想要在这条新记录上设置的任何值。这将可以返回新纪录的完整的 URI，包括记录号，记录号可以被用来查询和获取新记录的游标。

```
ContentValues values = new ContentValues（）;
Uri phoneUri = null;
Uri emailUri = null;
values.put（Contacts.People.NAME， "New Contact"）;
//1 = the new contact is added to favorites
//0 = the new contact is not added to favorites
values.put（Contacts.People.STARRED，1）;
//Add Phone Numbers
Uri uri = getContentResolver（）.insert（Contacts.People.CONTENT_URI， values）;
//The best way to add Contacts data like Phone， email， IM is to
//get the CONTENT_URI of the contact just inserted from People's table，
//and use withAppendedPath to construct the new Uri to insert into.
phoneUri = Uri.withAppendedPath（uri， Contacts.People.Phones.CONTENT_DIRECTORY）;
values.clear（）;
```

values.put（Contacts.Phones.TYPE， Phones.TYPE_MOBILE）；

values.put（Contacts.Phones.NUMBER， "1233214567"）；

getContentResolver（）.insert（phoneUri， values）；

//Add Email

emailUri = Uri.withAppendedPath（uri， ContactMethods.CONTENT_DIRECTORY）；

values.clear（）；

//ContactMethods.KIND is used to distinguish different kinds of

//contact data like email， im， etc.

values.put（ContactMethods.KIND， Contacts.KIND_EMAIL）；

values.put（ContactMethods.DATA， "test@example.com"）；

values.put（ContactMethods.TYPE， ContactMethods.TYPE_HOME）；

getContentResolver（）.insert（emailUri， values）；

你可以调用带有 URI 的 ContentResolver（）.openOutputStream（）方法保存一个文件，如下：

// Save the name and description in a map. Key is the content provider's

// column name， value is the value to save in that record field.

ContentValues values = new ContentValues（3）；

values.put（MediaStore.Images.Media.DISPLAY_NAME， "road_trip_1"）；

values.put（MediaStore.Images.Media.DESCRIPTION， "Day 1， trip to Los Angeles"）；

values.put（MediaStore.Images.Media.MIME_TYPE， "image/jpeg"）；

// Add a new record without the bitmap， but with the values.

// It returns the URI of the new record.

Uri uri = getContentResolver （ ） .insert （MediaStore.Images.Media.EXTERNAL_CONTENT_URI， values）；

try {

// Now get a handle to the file for that record， and save the data into it.

// sourceBitmap is a Bitmap object representing the file to save to the database.

OutputStream outStream = getContentResolver（）.openOutputStream（uri）；

sourceBitmap.compress（Bitmap.CompressFormat.JPEG， 50， outStream）；

```
   outStream.close（）；
} catch （Exception e） {
   Log.e（TAG， "exception while writing image"， e）；
}
```

删除一条记录

删除单条记录，调用 ContentResolver.delete（） 方法，指定特定行的 URI。

删除多行，调用 ContentResolver.delete（）方法，指定要删除的记录类型的 URI（如 android.provider.Contacts.People.CONTENT_URI） 和一个指定要删除行的 SQL WHERE 子句 （警告: 如果使用 ContentResolver.delete（）方法删除一个一般类型，要保证包含了一个有效的 WHERE 子句，否则你可能会删除一些不希望删除的记录）。

创建一个 Content Provider

下面是如何创建一个自己的 content provider，以便作为读写一个新的数据类型的公开的源泉：

1. 拓展 ContentProvider.

2. 定义一个命名为 CONTENT_URI 的 public static final Uri。这是表示你得 content provider 要处理得完整地"content://" URI 的字符串。你必须使用一个唯一的字符串，最好的办法就是使用类名的全称（小写），如，

```
 public static final Uri CONTENT_URI = Uri.parse
（ "content://com.google.codelab.rssprovider"）；
```

1. 为你的系统存储数据。大部分的 content providers 使用 Android 的文件存储方法或者 SQLite 数据库保存他们的数据，但是你也可以依你想要的任何方式保存数据，只要你遵循调用和返回值约定。

2. 定义列名。如果你将使用一个潜在的数据库，这些列名对于表示他们的 SQL 数据库的列名通常唯一的。在任何情况下，你应该包含一个整型的列名_id 来定义一个特定的记录号。如果使用 SQLite 数据库，它的类型应为 INTEGER PRIMARY KEY AUTOINCREMENT. AUTOINCREMENT ，描述符是可选的，但是缺省情况下，SQLite 自动增加一个 ID 统计域，使它大于当前存在表中最大的号。如果你删除了最后一行，添加的下一行将会使用这个被删除的行的 ID。为了避免这种情况，自动增加 ID 值到下一个最大的值，不管是否删除，将你得 ID 列赋予这种类型，

INTEGER PRIMARY KEY AUTOINCREMENT。（注意： 你应该有一个唯一的_id 域，不管你是否另有一个在记录中唯一的域，如 URL） 。Android 提供 SQLiteOpenHelper 类创建和管理你的数据库的版本。

3. 如果你需要公开二进制数据，如位图文件，保存这些数据的域应该是一个字符串域，保存特定文件的 content:// URI。这是客户端程序检索这个数据是将要是用的域。对于那种 content 类型，content provider 应该为那些记录定义一个_data 域，它也可以是同一个 content provider 或者另一个 content provider-如，保存图片时使用 media content provider。_data 域列出了那个文件在设备上的全路径。这个域只能通过 ContentResolver 读取，而不是客户端程序。客户端程序将在保存 URI 的可视的域上为这个项调用 ContentResolver.openOutputStream（） （如，图片列可能会有一个值 content://media/images/4453）。ContentResolver 将为那条记录请求_data 域，且因为 ContentResolver 有比客户端程序更高的权限，所以它可以直接访问那个文件，并将返回那个文件的 read wrapper 返回给客户端程序。

4. 申明 public static Strings ，客户端程序可以使用它指定要返回的列，或者从游标指定域的值。小心的存档每一个域的数据类型，记住那些文件域，如声频或者位图域，通常返回一个字符串的路径值。

5. 响应查询请求，返回一个包含记录集的游标对象。这就意味要实现 query（）， update（）， insert（）， 和 delete（） 方法，你可能需要调用 ContentResolver.notifyChange（） 通知有关更新信息给 listeners 。

6. 在 AndroidManifest.xml 中添加一个 <provider> 标签， 使用它的 authorities 属性定义它需要处理得 content type 的授权部分。如果你的 content 类型是 content://com.example.autos/auto， 请求所有 autos 的列表，这时，authorities 将会是 com.example.autos. 设置 multiprocess 属性为真，除非数据不必在多个 content provider 的运行版本之间同步。

7. 如果你正在处理一个新的数据类型，你必须定义一个新的 MIME 类型，作为 android.ContentProvider.getType（url） 的返回值。这个类型对应于 content:// URI 的 getType（），将成为这个 provider 处理的一种 content types.每一个 content type 的 MIME type 有两种形式：一个用于特定的纪录，另一个用于多个纪录。使用 Uri 方法，协助决定正在请求的事件。

下面是他们的一般格式：

o vnd.android.cursor.item/vnd.yourcompanyname.contenttype 为单一行，如，请求 122 条火车纪录使用 content://com.example.transportationprovider/trains/122

返回 MIME type vnd.android.cursor.item/vnd.example.rail

o vnd.android.cursor.dir/vnd.yourcompanyname.contenttype 为多行。如，请求所有火车记录，使用 content://com.example.transportationprovider/trains

返回 MIME type vnd.android.cursor.dir/vnd.example.rail

对于一个私有的 content provider 的视线，参考随 SDK 发布的记事本样例程序的 NodePadProvider 类。

下面是 content URI 重要部分的描述：

A. 标准的前缀，从不改变

B. 授权部分. 对于第三方程序，这需要一个全称的类名来确保为一行。这个对应于 <provider> 元素的 authorities 属性: <provider class="TransportationProvider" authorities="com.example.transportationprovider" />

C. Content provider 用来决定请求什么类型的数据的路径，这可以是零个或多个片断：如果 content provider 公开一种类型的数据（如，仅 train）， 这个段可以为空，如果他提供了多种类型，包括子类型，它可以由多个元素长度，如 "land/bus， land/train， sea/ship， and sea/submarine" ，提供四种可能性。

D. 假如仅仅要请求一个特定的纪录，_id 值所指定的特定的记录。如果要请求特定类型的所有记录，可以忽略它，并增加一个斜杠：content://com.example.transportationprovider/trains

3.3.5 网络

不要忘记你同样可以使用网络来存储和获取数据

Network Accesses with Android

In addition to all the on-device storage options， you can also store and retrieve data from the network （when available）. To do network operations， you'll want to use the following packages:

- java.net.*
- android.net.*

第四节 Security Model

3.4 Security Model

安全模式使你可以访问安全的系统资源和性能，并定义针对你自己的安全 Features 的访问控制。权限控制了一个给定的程序是否可以访问另一个程序的某种功能，如，那些程序可以拨打电话。本节将介绍权限是怎样工作的以及如果取得或者定义自己的权限。

3.4.1 Android 中的安全与许可

Android 是一个多进程系统，每个应用（以及系统的部分）运行在自己的进程中。很多应用和系统间的安全通过标准 Linux 工具在进程级执行，比如为应用分配的用户和组标识。额外的细粒度安全特性通过“许可”机制来提供，该机制能够对一个指定进程可实现的特定操作进行约束，以及授予对特定数据块的 ad-hoc 访问的 URI 许可进行约束。

3.4.2 安全架构

Android 安全架构的主要设计点是：缺省时，没有一个程序具有执行对其他程序，操作系统或者用户产生副作用的权限。这些包括读取或写入用户的私有数据（如联系人或电子邮件），读取或写入另一个程序文件，网络访问，保持设备处于工作状态等等。一个程序的进程是一个安全的盒子，它不能打断其他程序，除非显式的声明了它需要额外的（不在盒子的基本能力范围内）能力的权限。操作系统将以不同方式处理它所请求的权限，通常是根据证书或者用户提示，自动的允许或者不允许。程序静态的声明它所需要的权限，因此他们在安装的时候就已经被上层所知，之后也将不会改变。

3.4.3 注册程序（Application Signing）

所有的 Android 程序（.apk 文件）必须使用开发者所有的私钥注册。证书标识了这个程序的作者。这个证书不需要通过一个认证授权，它是允许的，Android 程序通常使用用户自注册的证书。证书仅被用来在程序之间建立信任关系，而不是批发一个程序可以安全那些控制。注册影响安全的一个最重要的方式是通过定义谁可以访问已经注册的许可，以及谁可以共享用户的 IDs。

3.4.4 用户标识和文件访问

每个安装在设备上的 Android 包（.apk）文件，Linux 都赋予它一个唯一的用户标识，创建一个沙盒（sandbox）并防止它触及其它应用（同样，也避免其它应用触及它）。这个用户标识是在应用安装在设备上时赋予的，并且通常该应用在设备上的整个生命周期内该标识保持不变。

因为安全 enforcement 发生在进程级，而正常情况下，每个进程都需要作为不同的 Linux 用户来运行，所以任意两个包的代码都不会运行在同一进程中。你可以使用每个包内 AndroidManifest.xml 中的 Manifest 标签下的 sharedUserId 属性来给它们赋予相同的用户标识。通过这样做，为了安全起见，这两个包就会被当作同一应用，具有相同的用户标识和文件许可。注意：为了保持安全，只有两个具有同一注册（并且请求相同的 sharedUserId）的应用才会赋予相同的用户标识。

应用创建的任何文件都会被赋予应用的用户标识，并且，正常情况下不能被其它包访问。当使用 getSharedPreferences（String， int），openFileOutput（String， int）或者 createDatabase（String， int， int， SQLiteDatabase.CursorFactory）来创建应用时，你可以使用 MODE_WORLD_READABLE 和/或 MODE_WORLD_WRITABLE 标志位来允许其它包读/写该文件。当设置了这些标志位，文件虽然始终为你的应用所有，但是它的全局可读/写许可已经被适当设置，所有的应用都可以看到它。

3.4.5 使用许可

一个基本 Android 应用没有与之关联的许可，这意味着它不能做任何会破坏该设备上用户体验或数据的事情。为了利用设备的保护特性，你必须在你的 AndroidManifest.xml 中包括一个或多个<uses-permission>标签来声明你的应用所需要的许可。

例如，需要监控流入 SMS 消息的应用会这样指定：

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.google.android.app.myapp" >
    <uses-permission id="android.permission.RECEIVE_SMS" />
</manifest>
```

在应用安装时，应用请求的许可由包安装器基于可信权威检查和与用户的交互情况来授予。在应用运行期间对用户不做检查：它要么在安装时被授予特定的许可，并且使用想用的特性；要么不被授予许可，并且使得一切使用特性的尝试失败而不提示用户。

通常，一个许可错误会导致产生 SecurityException 异常抛回给程序。然而，这并不保证在任何地方都会发生。例如，sendBroadcast（Intent）方法就是当数据被发送给到每个接收器时检查许可的，在方法调用返回之后，因此当许可失败时你不会收到一个异常。然而，几乎在所有情况中，许可失败都会被打印到系统日志中。

Android 系统提供的许可可以在 Manifest.permission 中找到。每个程序也可以定义和执行它自己的许可，因此这不是全面的所有可能的权限列表。

一个特定许可可以在你操作程序过程中的很多地方被执行：

- 在做系统调用时，防止应用执行某些功能。
- 在启动 Activity 时，防止一个应用启动其它应用的 activities。
- 发送和接收 Intent 广播时，控制谁能接收你的广播或者谁能发送广播给你。
- 当在 content provider 上进行访问或操作时。
- 绑定或启动一个服务。

3.4.6 声明和执行许可

为了执行你自己的许可，你必须首先在你的 AndroidManifest.xml 中使用一个或多个<permission>标签声明它们。

例如，一个应用程序想用控制谁能启动一个 activities，它可以为声明一个做这个操作的许可，如下:

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
  package="com.me.app.myapp" >
  <permission android:name="com.me.app.myapp.permission.DEADLY_ACTIVITY"
    android:label="@string/permlab_deadlyActivity"
    android:description="@string/permdesc_deadlyActivity"
    android:permissionGroup="android.permission-group.COST_MONEY"
    android:protectionLevel="dangerous" />
</manifest>
```

使用 <protectionLevel>属性告诉系统请求权限时如何通知程序，或者谁将拥有那个权限。 <permissionGroup> 属性是可选的，仅被用来帮助系统显示权限。你将经常需要为一个标准的系统组（在 android.Manifest.permission_group 中列出来的） 或者更难出现的情况—自己定义来设置它。推荐使用存在的组，这将使显示给用户的权限简易。

注意：应该为每个许可提供标签（label）和描述（description）。当用户浏览许可列表（android:label）或者一个许可的详细信息（android:description）时，它们是可以展示给用户的字符资源。标签（label）应该比较短，用几个字来描述该许可保护的关键功能点。描述（description）应该是一组句子，用于描述获得许可的用户可以做什么。我们写描述的习惯是两句话，第一句声明许可，第二句警告用户如果获取该许可时，会发生什么不好的事情。

下面是一个 CALL_PHONE 许可的例子：

```
 <string name="permlab_callPhone">directly call phone numbers</string>
```

<string name="permdesc_callPhone">Allows the application to call

phone numbers without your intervention. Malicious applications may

cause unexpected calls on your phone bill. Note that this does not

allow the application to call emergency numbers.</string>

通过 shell 命令 adb shell pm list permissions， 你可以查看当前定义系统中的权限。特定的， '-s' 选项用户查看这些权限的显示风格：

$ adb shell pm list permissions -s

All Permissions:

Network communication: view Wi-Fi state， create Bluetooth connections， full Internet access， view network state

Your location: access extra location provider commands， fine （GPS） location，mock location sources for testing， coarse （network-based） location

Services that cost you money: send SMS messages， directly call phone numbers

...

3.4.7 在 AndroidManifest.xml 中加上许可

用来约束到整个系统/应用组件的访问的高级许可，可以通过你的 AndroidManifest.xml 来设置。实现这种设置，需要在预期组件上有一个 android:permission 属性，并给该访问控制许可命名。

Activity 许可（应用于<activity>标签）约束谁能够启动相关活动。该许可在 Context.startActivity（）和 Activity.startSubActivity（）期间被检查；如果呼叫者没有必须的许可，那么该呼叫将会抛出 SecurityException.

Service 许可（应用于<service>标签）约束谁能够启动或绑定到相关的服务。该许可在 Context.startService（），Context.stopService（）以及 Context.bindService 期间被检查；如果呼叫者没有必须的许可，那么该呼叫将会抛出 SecurityException。

IntentReceiver 许可（应用于[[<receiver>]]标签）约束谁能向相关接收者广播 Intent 对象。该许可在 Context.broadcastIntent（）返回之后，在系统把提交的 Intent 发送给指定的接受者时被检查。所以，这种许可失败不会向呼叫者抛出异常；而只是不会发送 intent。同样，在 Context.registerReceiver()中也可以提供许可来控制谁能广播到一个已经程序注册的接收者。另一方面，当调用 Context.broadcastIntent（）时，许可可以用来约束哪些 IntentReceiver 对象被允许接受广播（见下方）。

ContentProvider 许可（应用于<provider>标签）约束谁能访问 ContentProvider 中的数据。于其他组件不同，你可以设置两个许可属性：android:readPermission 用来约束谁能从提供者那里读，android:writePermission 用来约束谁能向提供者写。注意：如果提供者被读写许可所保护，拥有写权限并不意味着你可以从提供者那里读到数据。该许可在你首次访问提供者（如果你不具有任一许可，将会抛出 SecurityException），并且在提供者上做操作时被检查。使用 ContentResolver.query（）需要有读许可；使用 ContentResolver.insert（），ContentResolver.update（），ContentResolver.delete（）或 Cursor.commitUpdates（）需要写许可。在所有这些情况下，没有必须的许可会导致呼叫抛出 SecurityException。

3.4.8 在广播 Intents 时 enforce 许可

许可除了可以 enforcing 谁能向注册的 IntentReceiver 发送 Intents（如上说述）之外，你也可以在广播 Intent 时指定一个许可。要想带一个许可字符串参数调用 Context.broadcastIntent（），要求接收者的应用必须具有该许可才能接受你的 Intent。

需要注意是接收者和广播者都能要求许可。当这种情况发生时，Intent 发送到相关目标必须要通过两个许可检查。

3.4.9 其它许可执行

在调用服务时，可以任意细粒度的许可检查。这是用 Context.checkCallingPermission（）方法来实现的。用一个理想的许可字符串调用，它将返回一个整型数来指示许可是否被目前调用过程认可。需要注意的是，这种方法只能在你执行其他进程的调用时使用，通常通过一个服务发布的 IDL 接口或一些进程指定的方法。

还有一些其他的好方法来检查许可。如果你有进程的 pid，你可以使用 Context 方法 Context.checkPermission（String， int， int）来向那个 pid 做许可检查。如果你有另一个应用的包名称，你可以使用直接的 PackageManager 方法 Context.checkPermission（String，String）来检查指定包是否认可一个特定许可。

第五节 资源和国际化

3.5 资源和国际化

资源是被代码使用的外部文件（非代码文件），在应用编译时被编译。Androi 支持大量不同类型的资源文件，包括 XML，PNG 和 JPEG 文件。XML 根据他们描述的不同又有很多

格式。

资源从源代码中被抽取出来，基于效率考虑，XML 文件被编译成二进制、可以快速加载的形式。字符串，同样被压缩为一种更富效率的存储形式。由于这些原因，在 Android 平台中我们就有了这些不同的资源类型。

该文档主要包含以下几个部分：

这是一个技术意味浓厚的文档，它和资源参考一起发布，它们包含了有关资源的很多信息。在使用 Android 的过程中并不需要把这个文档烂熟于胸，但是当你需要它时你应该知道来这里寻找信息。

3.5.1 资源

这个话题包含了与资源相关的术语列表，和一系列在代码中使用资源的实例。关于 Android 支持的所有资源类型的完整指南，参见 Resources.

Android 资源系统了解应用程序中所有非代码的资源。你可以使用 Resource 类访问你的程序中的资源；与你的应用程序关联的 Resource 实例一般可以通过 Context.getResource（）获得。

一个应用程序的资源在编译时被编译器编译到应用程序的二进制文件中。要使用一个资源，你必须把它们正确的安装源文件树中，并且编译到你的应用程序中。作为编译过程的一部分，每个资源的标记都会被生成，在你的源代码中可以使用这些标记——允许编译器验证你的应用程序代码是否和你定义的资源相匹配。

本部分的其余内容将作为一个在应用程序中如何使用资源的指南。

创建资源

Android 支持字符串、位图以及其他很多种类型的资源。每一种资源的语法、格式以及存放的位置，都会根据其类型的不同而不同。通常，你创建的资源一般来自于三种文件：XML 文件（除位图和 raw 之外的任何文件）、位图文件（图像）以及 Raw 文件（除前面以外的其他东西，如声音文件等等）。事实上，XML 文件也有两种不同的类型：被原封不动地编译进包内的文件和被 aapt 用来产生资源的文件。这里有一个每种资源类型的列表，包括文件格式、文件描述以及 XML 文件类型的细节。

你可以在你的项目中的 res/目录的适当的子目录中创建和保存资源文件。Android 有一个资源编译器（aapt），它依照资源所在的子目录及其格式对其进行编译。这里有一个每种资源的文件类型的列表，关于每种类型的描述、语法、格式以及其包含文件的格式或语法见 resource reference。

Directory	Resource Types
res/anim/	XML 文件，它们被编译进 frame by frame animation 或者 tweened animationa 对象
res/drawable/	.png、.9.png、.jpg 文件，它们被编译进以下的 Drawable 资源子类型中： 要获得这种类型的一个资源，可以使用 Resource.getDrawable（id） □ □9-patches（可变尺寸的位图） □ □位图文件
res/layout/	被编译为屏幕布局（或屏幕的一部分）的 XML 文件。参见布局（layout）
res/values/	可以被编译成很多种类型的资源的 XML 文件。 注意：不像其他的 res/文件夹，它可以保存任意数量的文件，这些文件保存了要创建资源的描述，而不是资源本身。XML 元素类型控制这些资源应该放在 R 类的什么地方。 尽管这个文件夹里的文件可以任意命名，不过下面是一些比较典型的文件（文件命名的惯例是将元素类型包含在该名称之中）： □ array.xml定义数据 □ colors.xml定义 color drawable 和颜色的字符串值

	（color string values）。使用 Resource.getDrawable（）和 Resources.getColor（）分别获得这些资源。 □ dimens.xml定义尺寸值（dimension value）。使用 Resources.getDimension（）获得这些资源。 □ strings.xml定义字符串（string）值（使用 Resources.getString（）或者 Resources.getText（）获取这些资源。getText（）会保留在 UI 字符串上应用的丰富的文本样式）。 □ □ styles.xml定义样式（style）对象。
res/xml/	任意的 XML 文件，在运行时可以通过调用 Resources.getXML（）读取。
res/raw/	直接复制到设备中的任意文件。它们无需编译，添加到你的应用程序编译产生的压缩文件中。要使用这些资源，可以调用 Resources.openRawResource（），参数是资源的 ID，即 R.raw.somefilename。

资源被编进最终的 APK 文件中。Android 创建了一个封装类，叫做 R，在代码中你可以使用它来引用这些资源。R 包含了根据资源文件的路径和名称命名的子类。

全局资源说明（Global Resource Notes）

一些资源允许你定义颜色值。Android 接受的颜色值可以使用多种 web 样式的形式——以下几种包含十六进制常数的形式：#RGB、#ARGB、#RRGGBB、#AARRGGBB。

所有颜色值支持设置透明度（alpha channel value），前两位的十六进制数指定了透明度。0 透明度值是全透明。默认值是不透明。

使用资源

这一部分描述如何使用你创建的资源。它包含以下主题：

- 代码中使用资源 ——如何在你的代码中调用资源进行实例化。
- 从其他资源中引用资源 ——你可以从其他资源中引用资源。这就使得你可以重用资源中公共资源值。
- 支持针对交替配置的交替资源 ——你可以根据主机硬件的语言或显示配置指定加载不同的资源。

在编译时，Android 产生一个名为 R 的类，它包含了你的程序中所有资源的资源标识符。这个类包含了一些子类，每一个子类针对一种 Android 支持的资源类型，或者你提供的一个资源文件。每一个类都包含了已编译资源的一个或多个资源标识符，你可以在代码中使用它们来加载资源。下面是一个小的资源文件，包含了字符串、布局（屏幕或屏幕的一部分）和图像资源。

> 注意：R 类是一个自动产生的文件，并没有设计为可以手动编辑。当资源更新时，它会根据需要重新产生。

```
package com.android.samples;
public final class R {
   public static final class string {
         public static final int greeting=0x0204000e;
         public static final int start_button_text=0x02040001;
         public static final int submit_button_text=0x02040008;
         public static final int main_screen_title=0x0204000a;
      };
      public static final class layout {
         public static final int start_screen=0x02070000;
         public static final int new_user_pane=0x02070001;
         public static final int select_user_list=0x02070002;

      };
      public static final class drawable {
         public static final int company_logo=0x02020005;
         public static final int smiling_cat=0x02020006;
         public static final int yellow_fade_background=0x02020007;
         public static final int stretch_button_1=0x02020008;

      };
   };
```

在代码中使用资源

在代码中使用资源，只是要知道所有资源 ID 和你的被编译的资源是什么类型。下面是一个引用资源的语法：

R.resource_type.resource_name

或者

android.R.resource_type.resource_name

其中 resource_type 是 R 的子类，保存资源的一个特定类型。resource_name 是在 XML 文件定义的资源的 name 属性，或者有其他文件类型为资源定义的文件名（不包含扩展名）。每一种资源类型都会根据其类型加为一个特定的 R 子类；要了解 R 的哪一个子类是关于你的资源类型的，请参考资源参考（resource reference）文档。被你的应用程序编译的资源可以不加包名引用（就像 R.resource_type.resource_name 这样简单）。Android 包含了很多标准资源，如屏幕样式和按钮背景。要在代码中引用这些资源，你必须使用 android 进行限定，如 android.R.drawable.button_background。

这里有一些还算不错的在代码中使用已编译资源的例子。

```
// Load a background for the current screen from a drawable resource.
this.getWindow（）.setBackgroundDrawableResource（R.drawable.my_background_image）;

// WRONG Sending a string resource reference into a
// method that expects a string.
this.getWindow（）.setTitle（R.string.main_title）;

// RIGHT Need to get the title from the Resources wrapper.
this.getWindow（）.setTitle（Resources.getText（R.string.main_title））;

// Load a custom layout for the current screen.
setContentView（R.layout.main_screen）;

// Set a slide in animation for a ViewFlipper object.
mFlipper.setInAnimation（AnimationUtils.loadAnimation（this，
    R.anim.hyperspace_in））;
```

```
// Set the text on a TextView object.
TextView msgTextView = （TextView）findViewByID（R.id.msg）;
msgTextView.setText（R.string.hello_message）;
```

引用资源

在属性（或资源）中提供的值也可以作为资源的引用。这种情况经常使用在布局文件中，以提供字符串（因此它们可以被本地化<将 UI 上的字符串放在一个单独的文件中，在做国际化时只需要将它们翻译成相应的语言版本，然后应用程序根据 locale 信息加载相应的字符串文件——译者注>）和图像（它们存在于另外的文件中），虽然引用可以是任何资源类型，包括颜色和整数。

例如，如果我们有颜色资源（color resources），我们可以编写一个布局文件，将文本的颜色设为那些资源中包含的值：

```
<?xml version="1.0" encoding="utf-8"?>
<EditText id="text"
   xmlns:android=http://schemas.android.com/apk/res/android
      android:layout_width="fill_parent"
      android:layout_height="fill_parent"
      android:textColor="@color/opaque_red"
      android:text="Hello， World!" />
```

注意，这里使用“@”前缀引入对一个资源的引用——在@[package:]type/name 形式中后面的文本是资源的名称。在这种情况下，我们不需要指定包名，因为我们引用的是我们自己包中的资源。要引用系统资源，你应该这样写：

```
<?xml version="1.0" encoding="utf-8"?>
<EditText id="text"
   xmlns:android=http://schemas.android.com/apk/res/android
      android:layout_width="fill_parent"
      android:layout_height="fill_parent"
      android:textColor="@android:color/opaque_red"
      android:text="Hello， World!" />
```

另外一个例子，你应该一直使用资源引用。当在布局文件中提供字符串时，你就可以将

它们本地化：

```
<?xml version="1.0" encoding="utf-8"?>
<EditText id="text"
   xmlns:android=http://schemas.android.com/apk/res/android
      android:layout_width="fill_parent"
      android:layout_height="fill_parent"
      android:textColor="@android:color/opaque_red"
      android:text="@string/hello_world" />
```

这种技巧还可以用来创建资源之间的引用。例如，我们可以创建新的 drawable 资源作为已存在资源的别名。

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
   <drawable id="my_background">@android:drawable/theme2_background</drawable>
</resources>
```

使用主题属性

另外一种资源值允许你引用当前主题中的属性的值。这个属性值只能在样式资源和 XML 属性中使用；它允许你通过将它们改变为当前主题提供的标准变化来改变 UI 元素的外观，而不是提供具体的值。

如例中所示，我们在布局资源中使用这个特性将文本颜色设定为标准颜色的一种，这些标准的颜色都是定义在基本系统主题中：

```
<?xml version="1.0" encoding="utf-8"?>
<EditText id="text"
   xmlns:android=http://schemas.android.com/apk/res/android
      android:layout_width="fill_parent" android:layout_height="fill_parent"
      android:textColor="?android:textDisabledColor"
      android:text="@string/hello_world" />
```

注意，这和资源引用非常类似，除了我们使用一个“?”前缀代替了“@”。当你使用这个标记时，你就提供了属性资源的名称，它将会在主题中被查找——因为资源工具知道需要的属性资源，所以你不需要显式声明这个类型（如果声明，其形式就是?android:attr/android:textDisabledColor）。

除了使用这个资源的标识符来查询主题中的值代替原始的资源，在这里关于该类型的 name 语法：?[namespace:]typpe/name 和“@”形式一样，也是可选。

使用系统资源

在系统中的包含了很多应用程序可以使用的资源。所有的这些资源都在“android.R”类下定义。例如，使用下面的代码你可以在屏幕上显示标准应用程序的图标：

```
public class MyActivity extends Activity
{
  public void onStart（）
    {
      requestScreenFeatures（FEATURE_BADGE_IMAGE）;
      super.onStart（）;
      setBadgeResource（android.R.drawable.sym_def_app_icon）;
    }
  }
```

以相似的方式，下面的代码将对你的屏幕应用系统定义的标准“绿色背景”视觉处理:

```
public class MyActivity extends Activity
{
  public void onStart（）
    {
      super.onStart（）;
      setTheme（android.R.style.Theme_Black）;
    }
  }
```

Supporting Alternate Resources for Alternate Languages and Configurations

你可以根据 UI 语言或者设备上的硬件配置，为你的产品提供不同的资源。注意，尽管你可以包含不同的字符串、布局和其他资源，然而 SDK 没有方法供你指定加载哪一个替换资源。Android 检测关于硬件和未知的适当配置，然后适当加载。用户可以使用设备上的设置面板选择替换语言设置。

为了包含替换资源，需要创建平行的资源文件夹，而文件夹的名字后面要使用限定符表示它要应用的配置（语言、屏幕方向等等）。例如，下面的工程包含了字符串资源，一个用

于英语，而另外一个用于法语：

```
MyApp/
   res/
        values-en/
           strings.xml
        values-fr/
           strings.xml
```

Android 支持几种类型的限定符，每一个都有不同的值。把它们连接在资源文件夹名称的后面，使用短横线隔开。你可以为每一个文件夹名称添加多个限定符，但是它们必须按照这里列出的顺序排列。例如，一个包含 drawable 资源的文件夹，对于一个完整详细的配置，可能看起来像：

```
MyApp/
   res/
        drawable-en-rUS-port-160dpi-finger-keysexposed-qwerty-dpad-480x320/
```

更典型的是，你只需指定一些特定的要定义资源的配置选项。你可以放弃完整列表中的任何值，但同时要保证剩下的值仍然保持列表中的顺序。

MyApp/ res/ drawable-en-rUS-finger/ drawable-port/ drawable-port-160dpi/ drawable-qwerty/

Qualifier Values

Language ISO 639-1 中定义的两个小写字母语言码，如: en， fr， es

Region 跟在小写字母“r”后面 ISO 3166-1-alpha-2 大写字母语言码 l，如: rUS， rFR， rES

Screen orientation port， land， square

Screen pixel density 92dpi， 108dpi， 等

Touchscreen type notouch， stylus， finger

Whether the keyboard is available to the user keysexposed， keyshidden

Primary text input method nokeys， qwerty， 12key

Primary non-touchscreennavigation method nonav， dpad， trackball， wheel

Screen dimensions 320x240， 640x480， etc. The larger dimension must be specified first.

这个表没有包含特定设备参数，如 carrier，branding， device/hardware，或 manufacturer. 任何应用想知道的有关设备的信息可以通过上表的资源限定符获得。

下面是一些有关资源目录名的一般方针：

- 破折号分割
- 区分大小写，如，

A portrait-specific drawable 目录必须命名为 drawable-port， 而不是 drawable-PORT.

你不可能有两个目录命名为 drawable-port 和 drawable-PORT， 即使你想让"port" 和 "PORT" 代表不同的参数值。

- 对任一种限定类型只能有一个值（也就是， 你不能指定 drawable-rEN-rFR/）
- 你可以为一个特定配置指定多个参数，但他们必须按照上面列的顺序。例如，drawable-en-rUS-land 将应用于 landscape view， US-English devices.
- Android 将试图为当前配置找到最匹配的目录

表中所列参数顺序被用在多限定附目录的情况

所有的目录，限定和非限定的，都在 res/ 文件夹下，限定的目录不能被嵌套.（不能有 res/drawable/drawable-en）

所 有的资源通过他们的简单的未加修饰的名字在代码中被引用，

如: MyApp/res/drawable-port-92dp /myimage.png 这样引用: R.drawable.myimage （code） @drawable/myimage （XML）

Android 如何找到最佳匹配目录

Android 将在运行时根据当前的配置选择资源文件，选择过程如下：

- 排除那些配置与当前设备配置不一致的资源。如，屏幕像素密度是 108dpi， 这将仅排除 MyApp/res/drawable-port-92dpi/.

MyApp/res/drawable/myimage.pngMyApp/res/drawable-en/myimage.pngMyApp/res/drawable-port/myimage.pngMyApp/res/drawable-port-92dpi/myimage.png

- 选择配置匹配度最高的资源。如， 你的地点是 en-GB，方向是 portrait，两个候选资源分别匹配一个： MyApp/res/drawable-en/ 和 MyApp/res/drawable-port/. 目录 MyApp/res/drawable/ 被排除，0 匹配，其他的留下。

MyApp/res/drawable/myimage.pngMyApp/res/drawable-en/myimage.pngMyApp/res/drawable-port/myimage.png

- 基于配置优先级确定最终资源文件，依赖于前表所列顺序。也就是说，语言比方向的优先级更高，因此选择特定语言文件 MyApp/res/drawable-en/.

MyApp/res/drawable-en/myimage.pngMyApp/res/drawable-port/myimage.png

术语

资源系统将许多不同片断合在一起形成最终的资源功能。为了帮助理解整个系统，下面是一些核心概念和组件的定义：

Asset: 应用程序相关的数据。包括编译过的 Java 源码，图形（如 PNG 图片），XML 文件等。这些文件在打包应用时组织在目录层次中，形成单一的 ZIP 文件。

aapt: Android Asset Packaging Tool. 这个工具生成了应用数据的最终 ZIP 文件。除了收集源数据之外，它也解析资源定义到二进制数据。

Resource Table: aapt 为你生成的特定资产，描述包含在应用或包中的所有资源。这个文件通过资源类访问，应用不能直接访问它。

Resource: 资源表的入口，描述单一资源值。有两种类型的资源：primitives and bags.

Resource Identifier: 在资源表中，所有的资源通过唯一整型数标识。在源代码中（资源描述，XML 文件，Java 源码），你可以使用符号名来代表实际的资源识别整数常量。

Primitive Resource: 所有的原始资源可以以字符串的形式被写，使用格式化来描述包含在资源系统中的不同原始类型：integers， colors， strings，references to other resources 等. 复杂的资源，如位图和 XML 描述，以原始字符串的形式存贮，原始字符串是实际数据的路径。

Bag Resource: 特殊的资源条目，保存一系列的 name/value 对。每一个名字是一个资源识别，每一个值保存同类型的字符串化的数据，如同普通资源。Bags 也支持继承： a bag 可以从其它包继续值，选择性的替换或扩展它们形成自己的内容。

Kind: 资源种类是为不同目的组织资源识别的途径。如，drawable 资源被用来实例化 Drawable 对象，因此他们的数据是一个原始资源包含一个眼色常量或者位图/XML 的字符串路径。其他常见资源种类是字符串（本地化原始字符串）， 颜色（颜色原始指）， 布局（描述 view layout 的 XML 文件的路径）， 和风格（描述用户界面属性的 bag 资源）. 有一个标准的 "attr" 资源种类，为命名 bag 项和 XML 属性而定义的资源识别符。

Style: 包含 bags 的资源种类，提供一系列的 UI 属性。如 style 资源提供 TextView 类，定义了它的文本大小,颜色和对齐方式。在一个 layout XML 文件中，你关联一个 bag 的 style，使用"style" 属性， style 资源的名字。

Style Class: 指定一个相关属性资源集。不在资源表中，但是被用来在代码中生成常量方便检索 style 资源和 XML 标签属性的值。如，Android 平台定义了 "View" style class，包含了标准的视图属性：padding， visibility， background， 等.; 当视图展开的时候，它使用这个 style class 从 XML 文件中检索那些值，并在实例中获取他们。

Configuration: 对于任何特定的资源识别，有多种不同的依赖于当前配置的值。配置包含地点，屏幕方向和屏幕密度等。当加载资源表时，当前配置被用来选择哪一个有效的资源值。

Theme: 为一个特定的上下文提供全局属性值的标准 style resource 。如，实现一个 Activity，应用开发者可以选择标准的主体，如 Theme.White 或者 Theme.Black; 这种风格提供了如屏幕背景图片/颜色，缺省文本颜色，按纽风格，文本编辑风格，文本大小等。当展开一个布局资源时，没有显式设置的大多数 widgets 的值来自于当前的主题，Layout 提供的风格和属性也可以使用在主题属性中显式指定的值赋予他们值。

Overlay:一个没有定义新资源集的资源表，但是替换了在另一个资源表中的值。像一个配置，在加载时被应用于资源数据；它可以添加新的配置数据（如新地点字符串），取代存在的值，和修改资源 bags。这为允许用户在不同的全局外表间选择提供了便利，或者下载新的外表的文件。

资源参考

资源参考 文档提供了不同类型的资源的详细列表和如何在 Java 代码或其他资源中使用它们。

3.5.2 国际化和本地化

国际化和本地化很关键，当前 SDK 还没有完全支持。随着 SDK 成熟，这部分将包含 Android 平台国际化和本地华的信息。http://www.androidcn.net/wiki/index.php/Devel/ui/layout

Android Intent 机制实例详解（1）

Android 中提供了 Intent 机制来协助应用间的交互与通讯，或者采用更准确的说法是，Intent 不仅可用于应用程序之间，也可用于应用程序内部的 Activity/Service 之间的交互。Intent 这个英语单词的本意是“目的、意向”等，对于较少从事于大型平台开发工作的程序员来说，这可能是一个不太容易理解的抽象概念，因为它与我们平常使用的简单函数/方法调用，或者上节中提到的通过库调用接口的方式不太一样。在 Intent 的使用中你看不到直接的函数调用，相对函数调用来说，Intent 是更为抽象的概念，利用 Intent 所实现的软件复用的粒度是 Activity/Service，比函数复用更高一些，另外耦合也更为松散。

Android 中与 Intent 相关的还有 Action/Category 及 Intent Filter 等，另外还有用于广播的 Intent，这些元素掺杂在一起，导致初学者不太容易迅速掌握 Intent 的用法。在讲解这些名词之前，我们先来从下面的例子中感受一下 Intent 的一些基本用法，看看它能做些什么，之后再来思考这种机制背后的意义。

理解 Intent 的关键之一是理解清楚 Intent 的两种基本用法：一种是显式的 Intent，即在构造 Intent 对象时就指定接收者，这种方式与普通的函数调用类似，只是复用的粒度有所差别；另一种是隐式的 Intent，即 Intent 的发送者在构造 Intent 对象时，并不知道也不关心接收者是谁，这种方式与函数调用差别比较大，有利于降低发送者和接收者之间的耦合。另外 Intent 除了发送外，还可用于广播，这些都将在后文进行详细讲述。

下面的一小节我们来看看显式 Intent 的用法。

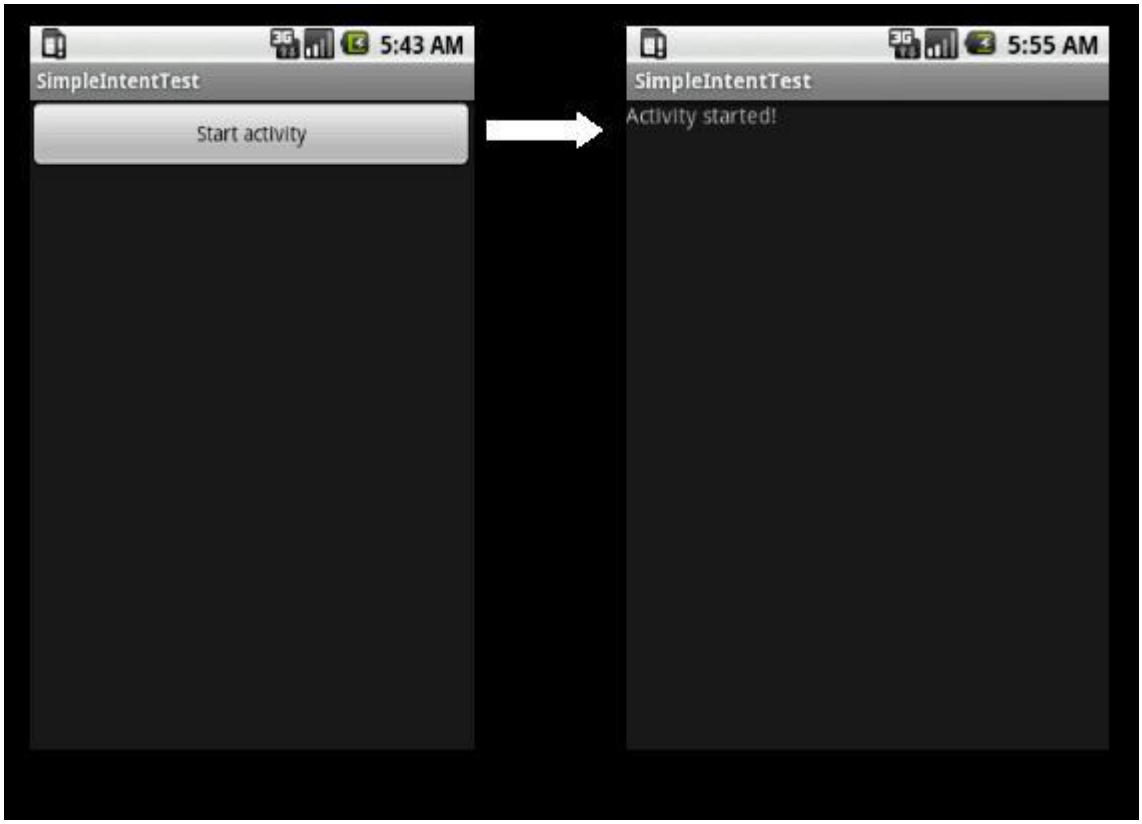
显式的 Intent（Explicit Intent）

□ □同一个应用程序中的 Activity 切换

我们在前面的章节已经讨论过 Activity 的概念，通常一个应用程序中需要多个 UI 屏幕，也就需要多个 Activity 类，并且在这些 Activity 之间进行切换，这种切换就是通过 Intent 机制来实现的。

在同一个应用程序中切换 Activity 时，我们通常都知道要启动的 Activity 具体是哪一个，因此常用显式的 Intent 来实现。下面的例子用来实现一个非常简单的应用程序 SimpleIntentTest，它包括两个 UI 屏幕也就是两个 Activity——SimpleIntentTest 类和

TestActivity 类，SimpleIntentTest 类有一个按钮用来启动 TestActivity，程序运行的截图如图 5.3 所示：



高手 Android 与 iPhone 平台开发经历对比 谁更好用？

作者：IT168 开心果 译 2009-07-15

几个月前我冒险进入移动开发世界，并创建了一个可以同时运行在 Android 和 iPhone 上的应用程序（Hudson Helper），这篇文章着重写的是关于我在开发这个程序时的一些经历，主要集中在 Android 和 iPhone 开发工具、平台和开发体验的对比上。

在你进一步阅读之前，我要指出的是我的对比中存在较大的偏见，因为我有过 12 年多的 Java 开发经验，可能我会更偏爱 Android。

1.语言、编程模型和平台

1.1 语言

iPhone 开发语言选择的是 Objective-C，Objective-C 是基于 C 语言的，并对其进行了一些面向对象的扩展，如类、继承、接口、消息、动态类型等，而开发 Android 应用程序时使用的是 Java 语言（但它实际上不能真正编译字节码）。

Java 对我而言就简单了，我必须要说的是我很高兴进入移动开发领域并没有因此而多学习一门语言，现有的编程技巧是来之不易的，如此重复使用专业知识是非常有价值的。

我还是花了一些时间来总结 Objective-C 的语言特性，我很快发现我喜欢上了某些语言特性，如消息传递、分类和命名参数，但是我发现 Objective-C 的语法有些笨重。通常我会感觉到为一个简单的概念不得不使用更多的表达式，而 IDE 提供不了多少帮助。

Objective-C 实际上是一种 80 年代的语言，某些问题如拆分头信息和执行文件，以及 DRY 破坏确实是在浪费时间，这些问题都不小，我发现我自己不停地在文件之间来回切换，这不仅在导航上会有成本（哪个文件处于打开状态?），而且每个打开的文件都需要重新创建（加字符号在哪里，选中了什么内容，我在文件中什么位置，文件是如何组织的这些都要消耗系统资源）。

至于 DRY，我声明一个属性时必须做 5 件事情??（在类定义中声明，声明 getter/settter，在 init 方法中初始化，在实施时@合成，在 dealloc 中发布），如：

Server.h

```
@interface Server : Updatable {
NSString *name; <-- declare the property
}
@property （nonatomic，retain） NSString *name; <--- declare the property again
Server.m
@synthesize name; <-- implement getter/setter
-（void） dealloc {
[name release]; <-- release memory
}
```

Java 也有类似的问题，但情况没有这么糟糕，IDE 可以帮助你编写 getter/setter。

Objective-C 中的指针虽然很强大，但也很浪费时间，在这里 Java 和它的垃圾回收机制就显得更迷人了。Objective-C 的另一个麻烦是要记住许多约定和规则。

虽然我理解了为什么这里会有 init 和 alloc，但指定[[aloc Foo] initWithArg: arg]还是过于啰嗦，为什么不是[new Foo arg]?或者是 Foo（arg），越说越象 Java 了。

Objective-C 的 import 和 forward 声明（@class）也很痛苦，虽然这些问题在 Java 中同样存在，但 Eclipse 的 JDT 非常好用，我都快忘了如何编写一个 import 了，只需要按 Ctrl+空格自动完成一个类名或按 Ctrl+Shift+O 来组织 import。

当然 Java 也不是完美的，但因为我已经与 Java 打了十多年交道，感觉又不一样了，有时我希望 Java 更类似于 Groovy，但 Java 的工具确实有很多选择，并且都比较好用。

1.2 平台

在 Android 上我发现可以立即使用 Java 运行类，但不是所有标准的 Java RT 类都是可用的，但我并没有遇到这个问题，因为大部分标准的 Java IO、网络和正则表达式库都是可以使用的。Android RT 类看起来是基于 Harmony 的，它是很稳定的，经住了时间的考验。

而在 iPhone 上，我发现我需要的功能是很困难的，类和方法组织得不好，当我在寻找一个静态方法时不是很方便，也依赖于使用的框架，命名规范和代码组织也可能不同，我认为这是旧平台遗留下来的产物。我发现规则表达式、字符串处理和 XML 解析也让我头痛，最终我不得不选择使用 Regex Kit Lite，对于 XML 解析，我是在 libxml 基础上提取出来的解析器。

在 iPhone 上如果遇到什么问题了，我只能求助于 Google，希望已经有人遇到过相同问

题并已经有解决办法了，这都是因为苹果公司实施的保密策略的后果，在某些时候我只能靠猜测或通过试验找到解决办法。

Android 得益于开源的好处，可以获得 Android 平台的全部源代码，甚至可以重新构建 SDK，确保与模拟器中运行类完全匹配。因此在 Android 平台上不仅可以看到事情是怎么实施的，还可以通过例子进行学习，我可以在模拟器中一步一步学习平台代码，发现我的代码为什么没有产出预期的结果。

一般说来，我发现 Android 平台类的布局、组织和命名约定是一致的，是可预知的，这就使得它的学习难度大大降低了。

1.3 编程模型

iPhone 平台最伟大的工作是促进了一个 MVC 设计模式，使用这一设计模式建立的平台，构建 UI 都非常简单，但我还没有找到如何组织我自己设计的 UI 组件，这就意味着当看到示例代码时，所有的组织方式都是一样的。

Android 催生的设计模式一样精彩，但它和 iPhone 的设计模式概念完全不同，Android 支持多进程和组件重用，设计结果具有更好的用户体验，但它为开发人员引入了一些复杂性。

Android 和 iPhone 都提供了用户首选项设置功能，都提供了 UI 来编辑这些首选项，都是保存在 XML 文件中，Android 的 XML 是可以扩展的，允许自定义 UI 组件，但 iPhone 开发人员如果希望自定义首选项，就不得不从零开始实现这个 UI，工作量就大多了。

1.4 测试和持续集成

我认为任何开发都应该包括单元测试，只要团队不止一人，还应该包括持续集成。Android 开发人员如果知道他们可以编写 JUnit 测试的话，他们一定会很高兴。我也曾看到一些 iPhone 单元测试的文档，但由于时间关系我没有仔细地研究，因此这里就不做过多的评论了。

2.资源

苹果公司为开发人员提供了大量的资源，重要的概念都配有视频教程，通过观看视频掌握概念就更容易了，虽然视频发布很慢，但苹果公司也提供了大量的示例程序和代码解释 API 的用法。

Android 开发人员也有很多资源可用，新手指南和 API 参考在安装 SDK 时就一起安装到计算机上了，因此即便是在离线状态也有很多资源可用。我发现 Android 的开发资源组织得非常好，只需要很少的时间进行查找，更多的时间留给发现，特别是 ApiDemos 示例应用程序提供了一个很好的开头。为了研究架构和 API 用法，我还下载了许多开源的 Android

项目，在这方面 Android 确实占了上风，因为苹果公司一直采取的是保密策略。

3.工具

我将会涉及到的工具分类包括：IDE，UI builder，调试器，分析器。

3.1 IDE

Android 开发主要使用的是优秀的 JDT 工具，基本上只要安装了 Eclipse 的计算机上都会安装 JDT，我使用这些工具已经有些年头了，至今仍然在用。JDT 最有特性的可能要算它的增量编译了，当你输入时它可以立即提供错误和警告反馈，它消除了 80 和 90 年代普遍存在的“代码编译--等待反馈”的循环，当我在编辑器中输入代码时，警告和错误实时更新，为我提供了即时反馈，直到我在 XCode 中编写 Objective-C 时我才意识到这项功能是多么伟大，这时我才意识到在等待编译的过程中会打断编程的思路。

其它 Eclipse 的关键特性包括：

1、内容助手

2、快速修复

3、组织导入

4、开放式（CTRL+Shift+T）

5、重构

集成 javadoc 和内容助手是学习一个不熟悉的 API 的最佳方法，在 Eclipse 中编写代码时不仅可以使用所有的类和方法，还可以使用它们的文档。

集成 javadoc 的内容助手

XCode 就太差劲了，我甚至不知道该如何下手，下面是我想到的 XCode 如果想继续活下去应该改进的列表：

1、内容助手要真正能够工作，XCode 提供的内容助手经常都是错的;

2、一个像样的窗口/编辑器管理系统，XCode 和它的辅助工具（调试器）喜欢打开很多窗口;

3、一个项目树视图，按字母顺序对文件进行排序;

4、集成 API 文档，我发现我经常在 IDE 和搜索 API 文档之间切换，这样会打断连续的思考。

3.2 UI Builder

iPhone 开发人员拥有一个界面美观的 UI builder，它很灵活并可以展示很多精密的 UI，给我的印象很深，但用起来并不顺手，可能要反复看几遍文档才能搞定。

Android UI builder 外观上就差多了，而且要对 UI 编写很多 XML 代码，不过也不用担心，有内容助手和校验的帮助，可以很快构建好 UI。

3.3 调试器

使用了 Eclipse 的 Java 调试器再去使用 Xcode 中的调试器简直就会崩溃，在 Eclipse 中可以看到并修改变量的值，但在 Xcode 中却不行，但这在调试代码时几乎是一个最常用的功能了，Xcode 经常混淆对象的类型，除了提供指针值外就没有其它内容了，这与 Eclipse 形成了鲜明的对比。

我还发现 Xcode 的调试器 UI 很难使用，在堆栈上点击时会打开一个新窗口显示代码编辑器，最终导致打开了无数个窗口。

3.4 分析器和堆栈分析

iPhone 开发工具擅长的是概要分析和堆栈分析，这些工具非常成熟且易于使用，在没有预先学习的情况下，我很快就掌握了它们的用法，并快速发现和修复程序中的内存泄漏问题。

Android 开发人员必须使用 Android 的单播跟踪程序，虽然工作得很好，但需要很努力地配置和操作才行，当我发现必须修改其源代码才能获得用于分析的跟踪文件时我非常惊讶。

我不确定 Android 是否可以提供 hprof 格式的堆栈转储文件，如果可以的话，就可以使用强大的 MAT 工具分析堆栈的使用了。

4. 应用程序商店

iPhone 的应用程序商店毋庸置疑是非常优秀的，但让人不愉快的是所有应用程序想要进入这个商店进行销售，都必须得经过苹果公司的审核，多数时候提交的申请都会被打回来，建议想在苹果应用程序商店销售软件的读者去看看这篇文章“避免 iPhone 应用程序遭到苹果公司的拒绝”（http://www.mobileorchard.com/avoiding-iphone-app-rejection-from-apple/）。还有一个问题就是想从苹果公司获得应用程序销售回扣也是比较困难的，至少销售额要超出 250 美元才有机会，但 Google 市场就不一样，只要有 1 美元就可以了，从苹果和 Google 公司获得的收益大概在应用程序销售价格的 30%左右。

苹果应用程序商店可以将应用程序销售到世界各地，但在 Google 市场中只能销售到少数几个国家，但在 Google 市场中你可以无限制地上传你开发的程序，无需经过审核。

5. 总结

Android 平台和开发工具非常优秀，使用 Java 和 Eclipse IDE 成为其胜出的主要因素，相比之下，iPhone 的开发工具就差得太远了，Objective-C 和平台 API 过于笨重，组织得也

不恰当，总的说来，我在开发 iPhone 应用程序时就象回到了 1993 年，开发 iPhone 应用程序时成本大约要上升三倍，iPhone 开发工具唯一的优势是它的分析器和堆栈分析能力。

从用户角度及目前在世界上的知名度而言，苹果应用程序商店是非常优秀的，在这方面 Google 市场就明显要弱一些。

可能过不了多久移动开发市场就会发生变化，至少今年会发布 18 款 Android 手机，对我而言，我更喜欢 Android，当然 iPhone 也很优秀。