目 录

关于本博客的主题

因为本人对一些智能手持设备感兴趣，像手机，GPS 等，所以在以往的博客里会涉及到很多这方面的内容。最近，看到谷歌发布了 Android 这个开发平台，也就是说可以自己为谷歌的手机，也就是国内所说的*爱迪机*写软件。所以，迫不及待的上网找一些资料，发现可以参考的资料寥寥无几。而且绝大部分是英文版的。这不能不说是对国内的手机编程爱好者是一个很大的障碍。

我的想法是自己的学习的同时，通过自己的英文能力，把一些好的英文教材翻译出来，然后在这个空间里放出来，使国内想学习 Android 编程的人能有一个快速学习的机会。为何选择在这里发布，主要的原因是，我原先的博客里有很多其它的内容，所以不容易管理。而放在这里会比较容易管理和更新，而不至于让博客的主题显得不集中。

目前，我做的第一个项目就是翻译一本叫做：Android A Programmer's Guide（Android 程序员向导）这样一本教材。全书一共 300 多页。计划在今年 6 月份之前结束这个项目。在翻译和发布的过程中，为了节约时间，会把其中的插图省略。如果大家感兴趣，可以下载英文版，对照插图使用。

博客地址：

http://www.soxitoday.com/blog/post/Android-programmer's-guide.html

第一章 什么是 Android

什么是 Android - 嵌入式设备编程的历史-第一章（1）

暂时可以这样说，传统的桌面应用程序开发者已经被惯坏了。这个不是说桌面应用程序开发比其他开发很简单。总之作为桌面应用程序开发者，我们已经有能力按照我们的想法创造出各种应用程序。包括我自己，因为我也是从做桌面程序开始的。一方面，我们已经使得桌面程序更容易的与桌面操作系统来进行交互，并且和任何底部的硬件很自由的交互。这种类型独立自主的程序编制其实对于很小的开发者团体来说是不敢贸然趟手机开发这趟浑水的。

注意：

在本部分讨论中，我提到两种不同的开发者：传统的桌面应用程序开发，他们能使用任何的编程语言，而且最终的产品和程序是用来运行“桌面”操作系统的；还有就是 Android 的程序开发者，为 Android 平台开发程序的 JAVA 程序员。我不是想说谁更好或者其它的意图。区别仅仅在于想说明并比较桌面操作系统环境的开发风格，工具。

有很长一段时间，手机的开发者由大的著名开发组中的少数人组成，作为嵌入式设备的开发者。相对于桌面开发或者后续的网络开发，被视作更少“魅力”，而且嵌入式设备的开发通常因为硬件和操作系统而处于劣势。因为嵌入式设备的制造商们太小气，他们要保护他们硬件方面的秘密，所以他们给开发者们非常有限的库来运行。

嵌入设备与桌面系统显著不同的一部分是嵌入设备是个“芯片上的电脑”。例如：说起你的标准电话遥控。这个并不是一个非常强大并且复杂性的技术。当任何的按钮被按下去，一个芯片解释一个信号以一种方式已经被编程进了设备。这个允许设备知道什么是从输入设备（键盘）来的需要。并且如何的响应这些命令（比如，打开电视机）。这个是一个简单的嵌入式设备的编程。总之，不管你相不相信，像这样的简单设备绝对的和早期的手机和开发有着紧密的联系。

大多数的嵌入式设备运行（有些还在运行）在私有的操作系统。原因是选择并创建一个私有的操作系统而不同定制的系统是产品必然选择。简单的设备不需要非常健全和优化的操作系统。

作为一个产品的演化，更多复杂的嵌入式设备，如早期的 PDA，家庭安全系统和 GPS 等。5 年前某种程度上都转移标准的操作系统平台上。小的操作系统如 Linux，或者一个微软的嵌入式平台，已经在嵌入设备上变得普遍了。设备演变的那些时间里，手机已自己的路径开始分支出去。这个分支是显而易见的。

差不多开始的时候，手机作为一个外围设备并且运行私有软件，而这些软件被制造商们所拥有和控制，而且几乎可以被认为是一个“关闭”的系统。习惯使用私有操作系统主要是制造商自己开发硬件，或者至少定义了开发的目的只是用来运行手机。最终的结果就是使开放成为不可能。现有的软件包或者解决方案会可靠的和他们的硬件交互。而且，制造商想要保护他们硬件的商业秘密。以防

允许进入而发现设备软件的水准。所以风尚就是，而且大多数仍然是使用完全私有并且关闭的软件来运行他们的设备。任何人想为手机开发程序必须需要详尽的私有环境来运行软件的知识。而解决方案就是直接从制造商那里购买昂贵的开发工具。这就孤立了很多的“自制软件”的开发者。

注意：

一个关于自制软件开发的文化包含了手机程序的开发。“自制软件”是指开发者通常不是工作在手机开发公司内，通常利用自己的时间在他们的设备上生产小的，一次性的产品。

另外，使手机开发无法出手的是硬件制造商对于“内存和需要”左右为难的解决方案。直到最近，手机才能执行比打出和接听电话，查找联系人，发送和接收短消息。不是今天“瑞士军刀”的技术。及时在 2002 年，在消费者的手上，带照相机的手机还是不多见。在 1997 年，小的应用程序如计算器和游戏爬进了手机内，但是强大的功能仍然是手机的拨号盘本身。手机还不想今天一样是一个多用途，多功能工具。没有人预见互联网浏览的需求，MP3 播放，或者更多的是我们今天定制的功能。在 1997 年，手机制造商们没有预见消费者需要的是一个一体化的设备。但是，即使这个需求展现出来，设备内存和存储容量还是一个需要克服的大的障碍。更多的人可能想要他们的设备是一个多功能一体化的工具，但是制造商们不许跨越他们的障碍。

让问题变得简单，就要在任何的设备让内存来存储并运行程序，包括手机。手机作为一个设备，直到最近还没有足够多内存来执行“额外”的程序。在最近的两年里，内存的价格已经达到了非常低的水平。设备制造商们有足够的能力压低价格来包含更多的内存。很多的现在的手机标准内存已经超过了 90 年代中期电脑内存。于是，现在我们有需求，而且有内存。我们可以直接跳到为手机开发酷的应用程序了，对吗？不完全是这样。设备的制造商们仍然紧密的保护他们的操作系统。有一些在手机上开放 JAVA 为基础的小运行环境。更多的是不允许。即使允许运行 JAVA 应用程序但还是不允许进入核心的系统。而这些是桌面开发者习惯于拥有的。

开放手机联盟和 Android-（2）

这个对于应用程序开发的障碍开始在 2007 年的 11 月份被打破，当 Google 在开放手机联盟下发布 Android。开放手机联盟是一个硬件和软件开发者的集合，包括谷歌， NTT DoCoMo,Sprint Nextel 和 HTC。他们的目标是创建一个更多的开放手机环境。在开放联盟第一个被发布的产品就是移动设备操作系统 Android。（更多关于开放手机联盟的信息，见：www.openhandsetalliance.com）。

对于这个 Android 的发布，谷歌使很多开发工具和向导成为可能来帮助在新系统上可能的开发者。帮助系统，平台软件开发包(SDK)，甚至一个开发者的论

坛，可以在谷歌的 Android 的网站上找到，http://code.google.com/android. 这个网站应该是你的起点，而且我极度推荐你去访问。

注意：

谷歌为了推动这个新的 Android 操作系统，甚至为寻找新的 Android 程序而设立了 1000 万美元的奖金。

运行 Linux, Windows 或者即使 PalmOS 的手机是很容易找到，如本文所述，没有硬件平台已经宣告可以来运行 Android.HTC，LG 电子，摩托罗拉和三星都是开发手机成员，在 Android 的发布下，我们希望在不久的将来有一些 Android 为基的设备。在 2007 年 11 月发布时，系统自身还仍旧是一个测试版的程序。这是个对开发者的好新闻因为它给了我们一个罕见的提前看到将来的设备和有机会来开始开发应用程序，而当硬件发布时就可以运行。

注意：

这个策略明确的给了开放手机联盟一个大的优势，超越其它手机操作系统开发者。因为当第一代设备发布时会有数不尽的可用开发程序可以运行。

介绍 Android 第一章(3)

Android，作为一个系统，是一个运行在 Linux2.6 核心上的 JAVA 基础的操作系统。系统是非常轻量型的而且全特性。

图 显示了一个未经修改的 Android 桌面屏幕。

Android 应用程序用 JAVA 开发而且很容易被放置到新的平台上。如果你没有下载 JAVA 或者不确定那一个版本你需要，我在第二章详细列出了开发环境的安装。其他 Android 的特点包括一个加速 3-D 图形引擎（基于硬件支持），被 SQLite 推动的数据库支持，和一个完整的网页浏览器。

如果你熟悉 JAVA 编程或者是任何种类的 OOP 开发者，你可能使用程序用户接口(UI)开发 - 那就是，UI 安置是直接在程序代码中有句柄的。Android，识别并许可 UI 开发，而且支持新生，XML 为基础的 UI 布局。XML UI 布局对普通桌面开发者是一个非常新的概念。我会在本书的相关章节里描述 XML UI 布局和程序化 UI 开发。

Android 另一个更令人激动和关注的特点是因为它的样式，第三方应用程序——包括“自制的”——会和系统捆绑的有着同样的优先权。这是和大多数系统不同之处，但是给了嵌入式系统程序一个比由第三方开发者创建的线性优先权大的优先执行权。而且，每一个应用程序在虚拟计算机上以一个非常轻量的方式按照自己的线路执行。

除了大量的 SDK 和成型的类库可以用之外，对激动人心的特性对于 Android 的开发者来说是我们现在可以进入到操作系统可以进入的地方。也就是说，如果你要创建一个应用程序打一个电话，你已经进入到电话的拨号盘。加入你要创建一个应用程序来使用电话内部的 GPS（如果安装了），你已经进入了。对于开发者创建动态和令人好奇的程序已经敞开大门。

和上面这些可用的特点相同，谷歌已经非常迫切的奉送一些特性。Android 的开发者可以将自己的应用程序和谷歌提供的如谷歌地图和无所不在的谷歌搜索绑在一起。假设你要写程序在谷歌地图上显示一个来电话者的的位置，或者你要储存一般的搜索结果到你的联系人中。在 Android 中，这个门已经完全打开。

第二章开始你 Android 的开发旅程。你会学到如何和为什么使用特定的开发环境或者综合的开发环境（IDE），而且你会下载并且安装 JAVA IDE Edipse.

问专家：

Q:谷歌和开放手机联盟的区别在哪里？

A:谷歌是开放手机联盟的一个成员。谷歌在收购了 Android 的原开发后，在开放手机联盟发布了操作系统。

Q:Android 有能力运行任何的 Linux 软件吗？

A:没必要。我坚信会有一种方式绕开大多数的开源系统和应用程序用 Android SDK 编译而用于 Android。主要原因是 Android 程序执行特定的文件格式，这会在后续的章节中讨论。

Android 示例 - 第四章（4）

Android 示例在 SDK/SAMPLES 内，包含了 6 个示例可以很好的描述 Android 的一些功能：

- API Demos
- Hello, Activity!
- Lunar Lander
- Note Pad
- Skeleton App
- Snake

这些示例由谷歌提供来给你一个快速的印象，那就是如何快速的开发 Android 的应用程序。每一个应用程序描述 Android 不同功能的一块。你可以用 Eclipse 打开并且运行这些应用程序。下面是对于每一个示例的简要描述。

API Demos
这个 API 示例应用程序说明在一个单独的 Activity 内如何展示多个 API 功能的示例。

提示：

一个 Activity 是一个 Android 的应用程序。Activities 会在后续的章节中深入展开 。

如下图（略）所示的，这个 API 示例应用程序包好了很多的，小的不同的 Android 功能的例子。这些例子包含 3-D 图形变换，列表，过程对话框和一个手指-画图示例。

运行 API 样本示例应用程序

使用 **Eclipse**，装载 API 示例应用现场作为一个 Android 项目。要做到这个，在 Eclipse 菜单选择文件|新建|项目，一个新的 Android 项目向导会启动。现在不用担心向导页面上的一些选项。只是选择从现有的项目中创建项目就好了，并且浏览到 API 示例所在的目录，点击这个示例。当项目装载好了，选择运行，在 Android 模拟器中来查看。用你自己的方式去查看超过 40 个示例吧，使用每一个示例去熟悉这些术语和功能。

Android 的几个示例 - 第四章（5）

Hello, Activity!

Hello, Actoviry 应用程序，是一个简单的 Hello World 风格的应用程序。虽然设计简单，但是它展示了平台的能力。在下一章，你会创建自己的 Hello World!风格的程序。

Lunar Lander 月球登陆
Lunar Lander 是一个在 Android 模拟器上玩的游戏。这个游戏是 2-D 的游戏它在 Android 上工作是多么的简单。控制非常的简单，而且游戏不是非常的复杂。总之，对游戏开发来说是一个良好的开始。

月球登陆执行一个简单控制方案（上，下，左，右）。游戏同时显示相关的非固定的图形并且对平台来说，令人印象深刻。复杂游戏的理论如冲突检出是以一个简单的方式使用的。虽然本书没有包含 Android 平台游戏编程的内容，加入你有兴趣来做这个，你或许可以从月球登陆中获得某些启发。

Note Pad 写字板
Note Pad，允许你打开，创建并且编辑小的笔记。写字板不是一个全功能的字符编辑器，所以不要期待是和 Windows Mobile 中 word 的竞争对手。但是，作为一个演示工具，使用非常少的代码就能实现这个效果已经非常的棒了。
Skeleton App 框架应用
Skeleton App 这是一个基本的程序，展示了几个不同的应用程序功能。如字体，按钮，图形和表格。如果你想自己运行 Skeleton App，真的不应当把它排除在外，参考 Skelete App，它会提供不少关于如何执行特定的条款。

Snake 蛇

最有一个在 Android SDK 的示例就是这个蛇了。这是一个小的 SNAFU 风格游戏，比月球登陆复杂。

注意：

如果你打开每一个示例应用程序的文件夹，你会看到一个文件夹命名为 src。这个是给出示例源代码的文件夹。你可以为其他任何的应用程序来查看，编辑并且重新编译这些代码。利用这些源代码来学一些 Android 平台技巧和提示。

第二章 下载和安装 Eclipse 总则

下载和安装 Eclipse 总则 - 第二章（1）

-关键技能&概念

-选择一个开发环境

-下载 Eclipse

-安装和配置 Eclipse

Android 应用程序是在 JAVA 下开发的。Android 自身不是一个语言，但是是一个运行应用程序的环境。这样，理论上你可以使用任何发布或者综合开发环境（IDE)来开始你的开发。事实上，你可以选择非 IDE 开发。

提示：

在本章稍后，我会介绍不使用 IDE 或者“命令行接口"（CLI)来开发 Android 应用程序。这期间，我不会在书中的每一个例子都使用这种技术，你将会学到如何在 CLI 里开发的基础知识。

假如你对使用 JAVA 的 IDE 比较舒服，如 Borland 的 JBuilder 或者开源 NetBeans,你可以尽管去使用。有了中等的水平的经验，你应当可以适应本书大部分的例子。但是，开放手机联盟和谷歌认同一个 JAVA 的 IDE,那就是：Eclipse.

注意：

如果你选择不用 Eclipse 来跟从本书的例子，你需要看看你的 IDE 文档关于编译和测试你的 Android 的程序。书中的例子只给了如何在 Eclipse 中编译和测试程序的说明，在 Eclipse 中使用 Android 的 plugin。

本章简明的描述了如何下载和安装 Eclipse 以及所要求的 JAVA Runtime Environment（JRE)。很多的时候，安装向导和教材趋向于跳过简单的步骤。我已经发现跳过简单的步骤经常忽略重要的条目。因为这个原因，我在本章内包含了从下载到安装的所有步骤。

为什么是 Eclipse?

为什么 Eclipse 是推荐的 Android 程序开发的 IDE 呢？对这个特定的认同有一些原因：

1、为了保持开发手机联盟真正开放移动开发市场的宗旨，Eclipse 是有着同样显著特点的，免费的 Java IDE 可以使用。Eclipse 同样容易使用，最少的学习时间。这些特性让 Eclipse 对于固定的,开放的 Java 开发成为吸引人的 IDE。

2、开发手机联盟已经为 Eclipse 发布了一个 Android 的 plugin，允许你来创建 Android-定义项目，编译它们，并且使用 Android 模拟器来运行和调试程序。当你开发你的第一个 Android 程序时，这些工具和能力将会是非常宝贵的。你还是可以用其它的 IDE 来创建 Android 程序,但是 Android 的 plugin 为 Elipse 创建某些元素——如，文件和编译设定。这些来自 Android-plugin 的帮助将缩

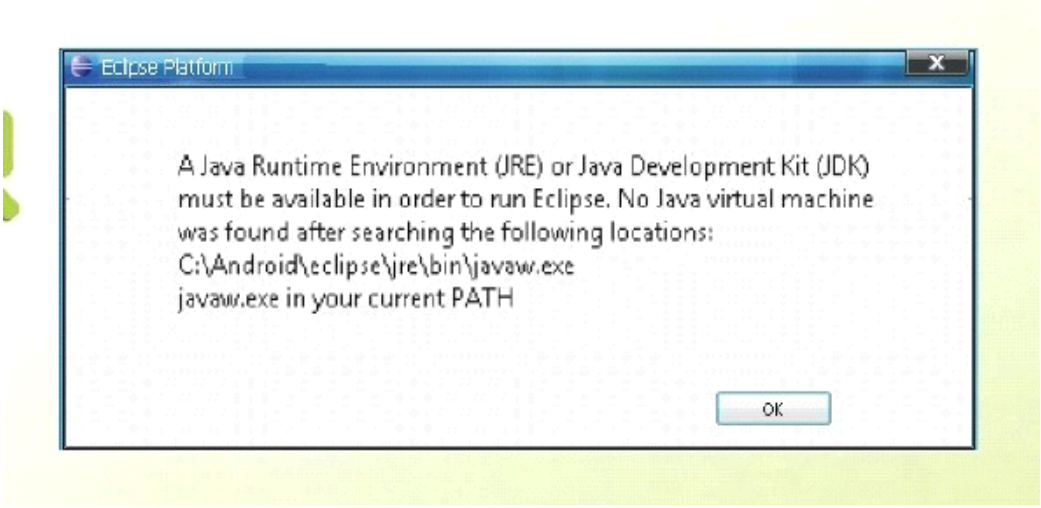
短你宝贵的开发时间并减少学习的弯路，那就意味着你可以花费更多的时间来创建惊人的应用程序了。

注意：

Elipse 同样也可用于苹果和 Linux 系统，有着强大的能力，在不同的操作系统，意味着几乎每个人可以在任何的电脑上开发 Android 的应用程序。不过，本书的例子和电脑截图觉来自与微软 Windows 版本的 Eclipse。记住这一点，如果你使用其他的电脑操作系统。你的界面可能看上去会有轻微的不同，但是总体的功能不会改变。如果在 Linux 的 Eclipse 有一些主要的操作不同点的话，我会举例说明。我会举出一些在 Linux 上的列子。而主要的例子会是 Linux/Android 的命令行环境（CLE)。

下载和安装 JRE - 第二章（2）

在你下载和安装 Eclipse 之前，你必须确保在电脑上**下载并安装了 Java Runtime Environment（JRE)**。因为 **Eclipse** 作为一个程序是由 Java 写成，它要求 JRE 来运行。如果 JRE 没有安装或被检测到，如果你试着打开 Eclipse，你会看见下面的错误：



如果你已经是一个 Java 的开发者并且已经在电脑上安装了 Java，你还是要按照提示安装，确保安装了正确版本的 JRE。

注意：

大多数使用过网络或者以网络为基础的应用程序的人，安装过 JRE。JRE 允许你运行 Java 基础的应用程序，但是它不允许你去创建它。要创建 Java 应用程序，你需要下载并安装 Java Development Kit（JDK),这个包含了创建 Java 应用程序所需的所有工具和库。如果你不熟悉 Java,记住这一点就行了。对于书中提到的例子，我会下载 JDK,因为它也包含了 JRE.虽然你不需要 JDK 来运行 Eclipse,但是你还是可以在本书后续章节的开发中使用。

导航到 Sun 公司的下载页面，http://developers.sun.com/downloads/,如下面的插图（略）所示。正常情况你只需要 JRE 来运行 Eclipse，但是对于本书

的目的，你应当下载包含了 JRE 的完整的 JDK,下载 JDK 的原因是在本书的后面，我会提到只使用 JDK 而非 Eclipse 来开发 Android 程序。如果你想跟从教材的话，你会需要完整的 JDK。

从 SUN 的下载页面，导航到适当 JDK 的下载部分。选择并下载，如下图（略）：

对于书中例子，我选择使用 Java 5 JDK Updata 14,因为在 Eclipse 文档中明确说明这是个支持的版本。要下载 Java 5 JDK，选择你要下载的平台来下载。你可能简单的跟着下载 Java 6 JDK。但是，如果你要下载旧的 JDK 5，你需要点击前一个发布的链接，如图（略）：

注意：

下载前，你必须同意并接受 Sun 公司的专利使用权转让协定。

在 Java Se 以前一个发布下载页面，点击 J2SE 5.0 下载链接，然后点击 JDK 5.0 Update x 下载按钮，x 是最后的升级号码（14 是本书写的时候的号码，你下载的时候可能会有所不同）。

如果你正在下载一个到微软 Windows 的环境，当你见到如下图（略）所示的通知时，点击 Run 来开始 JDK 的安装。

提醒：

如果你想要保存一份 JDK 包的备份，点击 Save 而并非 Run。总之，当你选择保存了 JDK，确保注意保存位置。在下载结束后，你需要导航到下载位置并且手动执行安装包。

在安装期间，你会被提醒阅读协议，如下图（略）。同意之后，点击 Next，然后就可以选择你的定制安装选项了。

这里只有一点你需要改变的，除非你是一个成熟使用 Java 的人并且需要选择特定的选项，在这种情况下，请自由的改变你需要的安装选项。下面是 Java JDK 安装的定制安装图（略）。

为了保持过程的简单性，并且完全地标准化，你应当接受软件自身的安装建议——选择缺省的设定——并且点击 Next 来继续安装。再次强调，如果你想要订制改变，请按照你自己的方式进行。总之，如果在后面的章节你遇到麻烦，你会需要修改你的安装选项。当安装完成的页面出现，如下图:（略），点击 Finish，然后你的安装就会完成。

一旦你完成 Java JDK 的安装——而且根据缺省，JRE 也会安装——你可以开始安装 Eclipse 了。

下载和安装 Eclipse - 第二章（3）

导航到 www.eclipse.org/downloads 的下载页面，如下图（略）。根据开放段落申明，**需要 JRE 运行环境**（推荐 Java 5 JRE）来开发 **Eclipse**，而这个我们已经在上节描述过了。在这个站点下载为 Java 开发者准备的 Eclipse 的 IDE。软件包比较小（79MB）并且应当下载很快。确保你不是下载了 Eclipse IDE for Java EE 的开发包，因为这个是有点不同的产品而且我不会介绍它的使用说明。

在你下载了 Elipse 以后，是时候来安装它了。导航到软件包下载的位置。写这本书的时候，最新的 Eclipse 软件包 Windows 版本的文件是 eclipse-java-curopa-fall2-win32.zip.解压缩软件包并且运行 Eclipse.exe。Eclipse 按照缺省方式安装到以用户目录（微软 Windows），但是你或许想安装到你的程序文件目录下。这样会保持你应用程序的有序而且允许你设定不同的目录作为工作空间。下图（略）显示了软件启动的欢迎画面。

注意：

如果你没有看见欢迎画面，试着重新启动电脑。如果重启后没有帮助的话，只下载并安装 Java 5 JRE。

一旦 Eclipse 安装开始，你会被提醒来创建一个缺省的工作空间，或者文件夹。和其他大多数开发环境一样，项目被创建，并且保存到这个工作空间内。缺省的工作空间路径是你的用户路径，选择不同路径，点击 Browse 来导航。如图（略）。

我建议你同样也选中选择框来定义你所有的项目到一个工作空间。选中这个框，当创建新项目时，你就会少一个需要担心的事情，而且你总是会知道在哪个路径里能找到你的源文件。在本书内，有时你需要导航到项目文件，并且在 Android 开发环境的外部工作，所以知道你文件的所在位置是非常有帮助的。

选择工作空间之后，点击 OK。 在这里，你的开发环境被下载好和安装。虽然 Eclipse 的安装似乎很快，你仍然需要在创建你的第一个 Android 项目前配置 Eclipse。很多的配置工作都是和 Android SDK 和 Android plugin 有关。

下一步你需要下载并安装 Android SDK，并且为 Eclipse 下载并安装 Android plugin。然后配置 Eclipse 设定。在第三章的结尾，你会有一个可以开发应用程序的完整的开发环境。然后你会浏览 Android SDK 并且在第五章创建你的第一个 Hello World! 应用程序。

问专家

Q：Eclipse 是用来开发 Java 的，但是 Android 能运行其他语言所写的程序吗？
A:写这本书时，没有 SDK 或者模拟器可以让 Android 来运行 Java 以外的程序。

Q：能使用 Eclipse（和 Android SDK)和 JRE 非 5 的版本 一起工作吗？

A：技术上说你可以使用 Eclipse 和版本 5 或者更新的版本一起工作，但是最新版本的 Eclipse 仅仅在 Java 5 JRE 上进行过测试。

第三章 下载和安装 Android SDK

下载和安装 Android SDK - 第三章（1）

关键技能和概念

-下载 Android SDK

-使用 Eclipse 的可升级特性

-为 Eclipse 下载，安装并配置 Android Plugin

-检查 PATH 声明

在前面的章节中，你下载并安装了主要的开发环境，Eclipse。现在，你的原始开发环境已经建立了，使用 Eclipse 作为你的 Java IDE，你可以用它来开发 Java 的应用程序。你必须以某种方式来配置它，以减轻 Android 的开发。

因为 Eclipse 是 Java 开发环境，你可以很简单的创建并编辑 Java 项目。但是，如果没有可以理解的库，规定 Android 应用程序应当如何工作，你就无法开发任何应用可以在 Android 为基础的设备上运行的程序。要开始创建 Android 项目，你需要下载并安装 Android SDK。然后你需要为 Eclipse 下载相关的 Android plugiin 来使用 SDK。有了这些部件的支撑，你就可以开始开发工作了。

如果你已经拥有任何的开发经验，很可能你已经熟悉使用 SDK 的过程。桌面程序的开发者，不管在哪一种的开发平台上开发，使用 SDK 来创建他们希望运行的系统上的应用程序。Android SDK 和其它的 SDK 相比没有任何的不同，它包含了所有的创建运行在特有的 Android 平台上应用程序所需的 Java 代码库。SDK 还包括帮助文件，文档和 Android 模拟器，大量的开发和调试工具。

注意：

第四章深入的阐述了 Android SDK 大多数的功能。

作为开始，你准备从谷歌 Android 开发网站上下载 Android SDK，网址：http://code.google.com/android 谷歌 Android 开发的主页上包含为 Android 平台开发的大量有价值的工具和文档，包括链接到 Android 开发者论坛。

提示：

如果你在开发的过程中遇到问题，你第一个找答案的地方应该就是 Android 开发者论坛。http://code.google.com/android/groups.html.这里有新手，开发者和黑客的讨论组。并且一个常规问题讨论组。考虑到 Android 是一个全新的平台，Android 开发者论坛是较少的能找到综合，可靠信息的地方。

下载 Android SDK - 第三章（2）

从谷歌的 http://code.google.com/android 网页可以很容易的找到 Android SDK 软件包。从开发的主页，点击下载 SDK 的链接开始。在你同意了 Android SDK 的软件许可协议后，你会看见 Android SDK 的下载页面。Android SDK 软件包对于 Windows 版本是 79MB 大小，你应当能够很快的下载。根据你的操作系统选择软件包开始下载。

注意：

软件包的大小根据不同的操作系统可能不一样。

说到 Android SDK，这里没有“setup”或者安装过程。这里，你必须跟着下面一些列的设置,在 Eclipse 开发环境里配置 Android SDK。第一步是获得 Android plugin，然后配置它。

为 Eclipse 下载和安装 Android Plugin ，设置 Android SDK 的第一步就是为 Eclipse 开发环境下载和安装 Android Plugin。Plugin 的下载和安装可以同时进行，而且非常的简单。

1.打开 Eclipse 应用程序,你将会下载为 Eclipse IDE 准备的 Android Plugin。
2.选择帮助|软件升级|寻找和安装。

3.在安装/升级的窗口,会允许你执行安装和下载在 Eclipse 任何可用的 plugin,点击搜索新特性选项，然后点击下一步。
4.Update sites to Visit 这个窗口会列出所有可获得 Eclipse plugin 的网站。但是，你所需要的 Android for Eclipse 没有列在这里，所以要下载这个 Android plugin 你必须要告诉 Eclipse 到哪里去找它。所以点击 New Remote Site 这个按钮。

5.在 New Remote Site 对话框内，你要提供两个信息：网站的名称和网址。名字只是便于显示并不影响下载。我们可以输入 Android Plugin。在 URL 字段。输入：https://dl-ssl.google.com/android/clipse.点击 OK。

注意:

这里填写的名字只是帮助你识别。你可以输入任何你想要的名字。

6.现在新的站点 Android Plugin 应当在可用的站点列表上了。这时，Eclipse 还没有开始寻找 plugin,这只是个路径你告诉 Eclipse。

7.选中 Android plugin 的选择框然后点击完成。Eclipse 开始任何可用的 plugin。

8.在搜索结果页面，选择 Android Plugin 然后点击完成。

9.在特性安装的许可页面，点击接受许可协议，然后点下一步。

注意：

记住所有的 plugin 都安装在/eclipse/plugins 的路径里。这个信息会帮助你假如你需要自己放置 Android plugin。

10.Eclipse 下载 Android plugin。本书写作时，plugin 的版本是 0.4.0.200802081635.在最终的 plugin 的安装页面，是特性核实，点击安装所有

来完成 Android plugin 的安装。

安装完成后就是必须去配置 plugin。

为 Eclipse 配置 Android Plugin - 第三章（3）

在完成了 **Android plugin** 的安装之后，**Eclipse** 应当提示你重新启动应用程序。如果它没有提示你，现在就重新启动 Eclipse。重启会确保安装的 plugin 有机会被初始化。安装下面的方式来配置是非常重要的。

配置 Android plugin 的方式是从 Eclipse 的 Preferences 窗口开始的，按照下面的步骤：

1.从 Eclipse 的程序主窗口中|Windows|preferences.

2.再出现的窗口中，在左边选择 Android 菜单。在窗体的右边点击 Browse，找到 **Android SDK** 的在硬盘的存放位置。输入到 SDK Location 的字段中。Eclipse 需要这个信息来进入到 Android 提供的工具，比如模拟器。选中 Automatically Syne Projects to Current SDK 选择框，然后点击应用。

注意：

Android plugin for windows 是以 zip 文件格式发布的。而且它包含了一个非常长的文件名称。android_m5-rc14-win32.重命名到一个比较容易管理的名字，这会在将来的章节中对你有帮助，特别是到命令行编程。你可能也会解压缩它到程序文件路径里。

4.Android SDK 的最后一个设置是把它放到 PATH 声明内。如果你用的是微软的 Windows，右击我的电脑，选择属性，然后选择高级。

5.点击环境变量。在这里可以编辑 PATH 声明。

6.在系统变量中，找到 PATH 然后双击它。

7. 在编辑系统变量的对话框中增加你的 Android SDK 路径，使用分号来分别现有的系统路径。点击 OK。在环境变量的窗口再次点击 OK。

现在，Android SDK, Eclipse 和 Android plugin 被完全的配置好了并且准备被用来开发了。在下一章，你会浏览 Android SDK,了解它的特性。Android SDK 包含很多工具来帮助你来开发全功能手机应用程序，并且下一章提供一个好的概述。

问专家：

Q:Android SDK 可以用在非 Java 的语言上吗？

A:不行。Android 应用程序只能在 Java 系统上被开发。

Q:会有更新的 Android SDK 吗？

A:是的!在写本书的时候，一个 SDK 的升级发布了，并且解决了平台上的很多问题。我建议经常检查开发页面的更新。

Q:如果升级了，我如何更新我的 SDK?

A:更新 SDK 是非常棘手的。当一个新的 SDK 发布，必须是 plugin 也发布。在写

本书时，新的 SDK 和新的 plugin 都发布了。我试图使用“Provided（提供的）”的升级工具来改变版本。最终无果并留给了我两个的版本，都工作不正常。我最终不得不卸载了它们并且重新安装最新的一个。然后那个最新的 SDK 工作正常了。我建议任何面对 SDK 或者 plugin 升级的人都采用相同的过程。简单的卸载老版本，然后安装新版本。不要升级。

第四章 浏览 Android SDK

浏览 Android SDK - 第四章（1）

关键技能和观念
—使用 Android SDK 文档
—使用 Android 工具
—使用 sample 应用程序
—学习 Android 程序的生命周期

现在，你已经建立了开发环境，准备去浏览 Android SDK 了，它包含了很多的文件和特别的工具，可以帮助你设计并开发运行在 Android 平台上的应用程序。这些工具设计的非常的好，而且可以帮助你制作一些难以置信的应用程序。在开始编程之前你真的需要熟悉 Android SDK 和它所带的工具。

Android SDK 还包含了一些可以让应用程序进入 Android 特性的库，比如和电话功能关联的（呼出和接电话），GPS 功能，和短消息。这些库组成了 SDK 的核心而且会是你经常会使用到的，所以，有一些时间来学习所有关于核心的库。

这一章包括了所有这些在 Android SDK 重要的条款，在本章的结尾，在你自己熟悉了 Android SDK 内容之后，你会足够舒适的开始写你的应用程序。总之，任何的事物都是这样，在你开始练习之前，你必须熟悉这些内容和指示。

注意：
我不会去介绍 Android SDK 的每一个细节，谷歌已经在 SDK 内做了非常好的文档。为了避开花费不必要的时间来讨论如何工作，我已经尽量少的做一些简要的说明。我只是会讨论一些重要的话题和条款，然后按照你自己的步伐去探索更深的层次。

Android SDK 是什么 - 第四章（2）

Android SDK 下载后会是一个简单的 ZIP 文件压缩包。Android SDK 的主体是一些文件，连续性的文档，可编程的 API，工具，例子和其它。本部分详细的说明这个 Android SDK 到底有些什么。

提示：

第三章建议你解压缩 Android SDK 到程序文件的文件夹，所以容易被找到。如果你找不到 SDK，因为你使用解压缩的缺省设定，应当在下面的文件夹/%downloadfolder%/android-sdk_m5-rc14_windows/android-sdk_m5-rc14_windows.（译者注：根据下载的文件名不同，这个文件夹也会不同哦）.

找到解压后的 Android SDK 的文件夹，然后可以在文件夹内浏览。在根目录会有几个文件，像 android.jar(一个编译过的，包含核心 SDK 库和 api 的 Java

应用程序）并且一些发布笔记，剩下的 Android SDK 被分成 3 个主要的文件夹：

- Docs 包括所有的 **Android 文档**

注意：

这些文档同样也可以在 Android 开发网站上找到 http://code.google.com/android .

- Samples 可以在 Eclipse 内编译和测试的 6 个应用程序例子
- Tools 包含所有在开发过程中需要的开发和调试工具

下面的部分会讨论更多关于在每一个文件夹内的内容。每一个 API 示例被编译过并且可插入至 Android。在后续学习如何在 windows 和 Linux 中使用命令行选项创建和编译应用程序的章节中会讨论更多的工具。

Android 文档 - 第四章（3）

Android 文档被放在 Android SDK 内的 Docs 的文件夹内。文档内提供了如何下载和安装 SDK 的每一个步骤，“Getting Started”开发应用程序的快速步骤和软件包定义。文档是 HTML 格式并且有一个 documentation.html 在 SDK 的根目录可以进入整个文档。下面的插图（略）就是 Android SDK 文档的主页。

你可以从 documentation.html 上提供的链接导航到 Android SDK 内包含的文档。

注意：

当你浏览 Android SDK 时，你可能想到一些页面是一些错误的链接或者丢失了。因为当你点击某些链接时，屏幕右边可能会显示空白，不过，如果你再往下滚动页面你将会明白页面只是没有被排列好。

在这个 Android SDK 内，我已经发现有一些部分比其他的部分更重要。对于我来说最重要的 Android SDK 文档如下（它们会出现在导航条上）：

- Reference Information
- Class Index
- List of Permissions
- List of Resource Types
- FAQs
- Troubleshooting

当你开始开发，Troubleshooting 文档的分类部分将会特别有作用。当你深入本书并且开始开发你自己的应用程序，你会发现文档的 Reference Information 部分会更有帮组。例如，List of Permissions 分类部分将会非常的有帮助，当你跟着本书创建更复杂的应用程序时。虽然这个现在对你用处不大。花些时间熟悉一下 Android 文档吧。

Android 示例 - 第四章（4）

Android 示例在 SDK/SAMPLES 内，包含了 6 个示例可以很好的描述 Android 的一些功能：

- API Demos
- Hello, Activity!
- Lunar Lander
- Note Pad
- Skeleton App
- Snake

这些示例由谷歌提供来给你一个快速的印象，那就是如何快速的开发 Android 的应用程序。每一个应用程序描述 Android 不同功能的一块。你可以用 Eclipse 打开并且运行这些应用程序。下面是对于每一个示例的简要描述。

API Demos
这个 API 示例应用程序说明在一个单独的 Activity 内如何展示多个 API 功能的示例。

提示：

一个 Activity 是一个 Android 的应用程序。Activities 会在后续的章节中深入展开。

如下图（略）所示的，这个 API 示例应用程序包括了很多的，小的不同的 Android 功能的例子。这些例子包含 3-D 图形变换，列表，过程对话框和一个手指-画图示例。

运行 API 样本示例应用程序

使用 **Eclipse**，装载 API 示例应用现场作为一个 Android 项目。要做到这个，在 Eclipse 菜单选择文件|新建|项目，一个新的 Android 项目向导会启动。现在不用担心向导页面上的一些选项。只是选择从现有的项目中创建项目就好了，并且浏览到 API 示例所在的目录，点击这个示例。当项目装载好了，选择运行，在 Android 模拟器中来查看。用你自己的方式去查看超过 40 个示例吧，使用每一个示例去熟悉这些术语和功能。

Android 的几个示例 - 第四章（5）

Hello, Activity 应用程序，是一个简单的 Hello World!风格的应用程序。虽然设计简单，但是它展示了平台的能力。在下一章，你会创建自己的 Hello World 风格的程序。

Lunar Lander 月球登陆
Lunar Lander, 是一个在 **Android 模拟器**上玩的游戏。这个游戏一个 2-D 的游戏在 Android 上工作是多么的简单。控制非常的简单，而且游戏不是非常的复杂。总之，对游戏开发来说是一个良好的开始。

月球登陆执行一个简单控制方案（上，下，左，右）。游戏同时显示相关的非固定的图形并且对平台来说，令人印象深刻。复杂游戏的理论如冲突检出是以一个

简单的方式使用的。虽然本书没有包含 Android 平台游戏编程的内容，加入你有兴趣来做这个，你或许可以从月球登陆中获得某些启发。

Note Pad 写字板

Note Pad, 允许你打开，创建并且编辑小的笔记。写字板不是一个全功能的字符编辑器，所以不要期待是和 Windows Mobile 中 word 的竞争对手。但是，作为一个演示工具，使用非常少的代码就能实现这个效果已经非常的棒了。

Skeleton App 框架应用

Skeleton App, 这是一个基本的程序展示了几个不同的应用程序的功能。如字体，按钮，图形和表格。如果你想自己运行 Skeleton App，真的不应当把它排除在外，参考 Skelete App，它会提供不少关于如何执行特定的条款。

Snake 蛇

最后一个在 **Android SDK** 的示例就是这个蛇了。这是一个小的 SNAFU 风格游戏，比月球登陆复杂。

注意：

如果你打开每一个示例应用程序的文件夹，你会看到一个文件夹命名为 src。这个是给出示例源代码的文件夹。你可以为其他任何的应用程序来查看，编辑并且重新编译这些代码。利用这些源代码来学一些 Android 平台技巧和提示。

Android 工具 - 第四章（6）

Android SDK 提供给开发者一系列功能强大并且有用的工具。在本书内，你会直接使用它们。本部分对其中的一些工具做一个快速的查看，而在后续的章节中会更加深入的进行，那就是在命令行开发中。

注意：

对于 Android SDK 中包含的更多的工具，请查看 Android 文档。

emulator.exe

Android SDk 中一个最重要的工具就是这个 emulator.exe。emulator.exe 启动 Android 模拟器。Android 模拟器被用来在一个假的 Android 环境中运行你的应用程序。在本书写作时，还没有发布 Android 平台可用的硬件，emulator.exe 将会是唯一的方法作为测试应用程序的平台。

你可以从 Eclipse 或者命令行中来运行 emulator.exe。在本书中，通常会使用 Eclipse 启动 Android 模拟器环境。总之，为了给你所有信息关于在 Eclipse 之外用 Android SDK 编程。在第六章里会介绍 emulator.exe 的命令行使用来创建 Hello World 应用程序。

当使用 Android 模拟器来测试你的应用程序，有两个选择可以导航到用户界面。第一，带按钮的模拟器。你可以使用这些导航按钮来导航 Android 和任何的你为这个平台开发的应用程序。

提示：

电源 On/Off,声音的大小按钮被隐藏在虚拟设备的旁边。当你用鼠标移过它们时，会被自动识别。

很多的高端手机现在都包含了触摸屏，第二个输入选项就是这个模拟的触摸屏。使用你的鼠标作为一个尖笔。模拟器屏幕上的对象可以相应鼠标的动作。

adb.exe

当你使用命令行编辑器时另外一个工具会变得非常的有用，它就是 Android 调试桥，或者 adb.exe。这个工具允许你发出命令到模拟器工具。当你在命令行环境下工作时，这个 adb 工具允许你做下列工作。

- 开始并且停止服务
- 安装和卸载应用程序
- 移动文件至模拟器或者从那里移动

MKSDCARD.exe

MKSDCARD.exe 是一个非常有用的工具，当你测试一个应用程序，而这个程序需要读取或者写入文件到一个插入到移动设备的 SD 储存卡中。MKSDCARD.exe 在你的驱动器中创建一个小的驱动并且会保留测试文件。然后模拟器会把这个小的部分当成一个 SD 储存卡。

DX.exe

DX.exe 是 Android SDK 的编译器。当运行你的 Java 文件，DX.exe 将创建一个带有.dex 后缀—Dalvik 可执行格式的文件。这些会被 Android 设备正确的理解和运行。

注意：

Android 可执行文件是叫做 Dalvik 可执行文件，Dalvik 虚拟机器以自己脉络来运行每一个应用程序，而且程序的优先权和 Android 核心程序一致。

activityCreator(.bat 或者 .pn)

activityCreator 是一个简单的命令行工具被用来设定基本的开发环境。当从命令行运行时，activityCreator 将设置一个需要的基本 Android 应用程序所需的壳文件。有了这些壳文件是非常有用的，特别是你不使用 Eclipse。当你创建一个新项目时，Android plugin for Eclipse 通过呼叫 activityCreator 来设置这些壳文件。依据你运行的是哪一种环境类型，你会看到不同的 activityCreator 的脚本文件。如果你使用 Windows 环境，这个就会是.bat 文件，否则就是 python(.pn)脚本。简单的执行这些脚本，就会依次的使用正确的参数来呼叫真正的 activityCreator 过程。

Android APIs - 第四章（7）

APIs 或者叫做应用程序编程接口，是 Android SDK 的核心。一个 API 是应用程序开发者在特定平台上创建程序的功能，方法，属性，类别和库的集合。Android API 包含所有你创建与 Android 为基础程序交互的特定信息。

Android SDk 同样包含 2 套 api，—谷歌的 API 和可选的 API.后续的章节中将重点放在这些 API 上，因为你将利用它们写程序。现在，让我们快速的说明一

下它们包含哪些你熟悉的使用。

谷歌 api

谷歌 API 含在 Android SDK 中并且包含编程参考允许你绑定你的程序到现有的谷歌服务中。假如你写一个应用程序允许你的用户通过你的程序进入到谷歌提供的服务中，你需要包含谷歌的 API.

找到 android.jar 文件，谷歌的 API 包含在 com.google.*包装中。只有很少的包含了谷歌的 API.一些包装随着 API 一起发布包含了图形，移动性，联系人和日历等工具。总之，我们会把本书中把重点放在谷歌地图上。

使用 com.google.android.maps 包装，这个包含了谷歌的地图，你可以创建一个应用程序无缝的和熟悉的谷歌地图界面对接。这个包装打开了一个等待着被开发的整个有用的应用程序世界。

谷歌 api 还包含了一套有用的包装，来允许你利用由 Jabber 开放源码社区开发的最新的 Extensible Messaging 和 Presence Protocl（XMPP)。使用 XMPP,应用程序可以快速知道户主在场或者是否可用（从信息和通信中）。如果你要利用电话的短信功能来创建一个聊天类的程序，这个处理 XMPP 的 API 是非常有用的。

可选的 api

Android SDK 包含了一些可选的 api，它包括了一些标准 Android api 未包含的内容。说它们是可选的 api 意味着这些功能在手持设备上可能出现也可能不出现。也就是说一些为 Android 平台创建的设备可能包含升级或者一些特性而其他的没有。当利用在你的应用程序中利用这些可选的 API 时，包含了你的编程选项。

其中的一个可选特性（本书的后面会使用）就叫做电话基础的 GPS.Android LBS(位置基础的服务）api 需要接受并利用设备上 GPS 单元的信息。如果结合 Android LBS api 和谷歌地图 api，你或许有一个非常有用的应用程序会实时的显示你的位置。

其它可选的 api 包含利用蓝牙，Wi-Fi，播放 MP3，进入并激活 3-D-OpenGL 硬件等。

应用程序生命周期 - 第四章（8)

如果你有相当好的编程经验的话，你对应用程序的生命周期这一概念应该熟悉。一个应用程序的生命周期，由一些应用程序由开始执行到终止的步骤组成。每一个应用程序，不管是哪一种语言所写，都有一定的生命周期。Android 应用程序也没有例外。本部分会仔细对比 ASP 应用程序和 Android 的应用程序的生命周期。

标准 ASP 程序应用程序生命周期

标准 ASP 应用程序的生命周期和一个 Android 的程序生命周期非常的类似。ASP 应用程序从开始到结束有 5 个步骤。这些步骤对所有的 ASP 程序是一致的。

并且界定了 ASP 程序是什么。这些步骤按照次序如下：
1. Application_Start（程序开始）
2. Event（事件）
3. HTTPApplication.Init
4. Disposal
5. Application_End
提示：

有些 ASP 的参考材料考虑 Disposal 和 Application_End 在生命周期中成为一个步骤。但是，Disposal 呼叫可以到达 Application_End 之前被打断。这个可以允许程序在真正结束之前执行特定的功能。

当应用程序被从服务器要求执行，开始呼叫 Application_Start。这个过程依次的通向过程处理。当所有相关的应用程序模块被装载，HTTPApplicaation.Init 被呼叫。程序执行事件，并且当用户试图去关闭它，Dispose 被呼叫。Dispose 然后转移过程到 Application_End 过程，来关闭程序。

这是一个相当标准的应用程序生命周期。大多数的程序是这个生命周期：一个应用程序被创建，装载，拥有事件，并且被关闭。下面说明和 Android 应用程序生命周期的对比。

Android 应用程序生命周期是唯一一个系统控制多的应用程序生命周期。所有的 Android 应用程序，或者 Actiities 都运行在自有的过程中。所用的运行过程都被 Android 观察，并且取决于活动是如何运行的（就是说，一个前台活动，一个后台活动）Android 可能选择去结束一个消耗系统资源的活动。

注意：

当决定是否关闭一个活动时，Android 会考虑一些因素，如用户输入，内存使用和过程时间。一个 Android 或者的生命周期以一些特定的方式被称呼：

- onCreate
- onStart
- Process-specific events (for example: launching activities or accessing a database)
- onStop
- onDestroy

与其它程序的逻辑一样，一个 Android 应用程序被创建，过程开始，事件被执行，过程停止，并且应用程序结束。虽然有一些不同，很多的程序开发者应该不会对这样的生命周期感到别扭。

问专家：
Q: 谷歌会升级 Android SDK 吗？

A:是的。从我开始写这本书的时候，谷歌已经升级了 Android SDK 很多次了.谷歌会在 Android 的网站上发布最新的版本。
Q: 会有任何 API 试用版出现在最终产品中吗？
A: 或许不会。API 试用版创建出来是为了炫耀产品能力的。虽然它们可能是核心解除的包含一些在 API 试用版里元素的应用程序，我们应该看不到月球登陆这

个游戏出现在最终产品中。

第五章 Android 程序：Hello World!

Android 程序：Hello World! -第五章（1）

关键技能和概念

- 创建新的 Android 项目
- 同 Views 一起工作
- 使用一个 TextView
- 修改 main.xml 文件
- 在 Android 模拟器上运行应用程序

为了让你能够对在 Android 上编程有一个良好的印象，在第六章，你会在 Windwos 平台和 Linux 平台上使用 Android SDK 创建命令行应用程序。或者说，本章包含了在 Eclipse 创建程序的过程，第六章包含了使用命令行工具的创建过程。因此，在继续之前，你应当检查你的 Eclipse 的开发环境是否被正确的配置。再次回顾一下第三章关于 Adnroid SDK 的 PATH 声明。同时要确保 JRE 也是在你的 PATH 声明中。

提示

如果当你运行命令行示例，有任何与配置有关的问题时，请参考第二章和第三章提到的步骤，并且查看 Android SDK 文档。

在 Eclipse 中创建你的第一个 Android 项目

要开始你的第一个 Android 项目，打开 Eclipse.当你第一次打开 Eclipse，它会打开一个空开发环境，这就是你要开始的地方。你的第一个任务是设置并且命名一个工作空间。选择 文件| 新建 |Android 项目，使你能够创建一个 Android 特有的应用程序向导。

注意：

不要从新建菜单上选择 Java 项目。你的 Android 应用程序是在 Java 中写的，并且你在 Java 项目中进行开发，这个选项会创建一个标准的 Java 应用程序。选择 Android 项目来创建一个 Android 特有的应用程序。

如果你没有看到啊 Android 项目这个选项，这就说明在 Eclipse 中，Android plugin 没有被完全或者正确的安装。重新检查第三章中关于 Android plugin 的安装程序来修正这个问题。

新的 Android 项目向导为你创建 2 个东西：

- 一个绑住 Android SDK 的壳程序。这个将允许你使用所有 Android 库和包来进行编码工作，并且允许你在合适的环境中调试程序。

● 新程序的第一个壳文件。这些壳文件包含一些必要的支撑你将要编写程序的文件。就如同一个在 Visual Studio 中，它会在你的文件中产生的一些代码。使用 Eclipse 中的 Android 项目向导产生一些初始的程序文件和一些 Android 创建的代码。

另外，新的 Android 项目向导包含一些你必须输入的选项。

在项目的名称那个字段，只是为了举例，使用 HelloWorldText 这个名字，这个名字非常的容易把这个 Hello World 项目从其它你将要在本章中创建的项目分别开。

在内容那个区域，保持缺省的选择：在工作区中创建一个新的项目这个选项必须被选中。并且使用缺省的位置这个选择框也应当被选中。这个将允许 Eclipse 在你缺省的工作区路径中创建你的项目。这样做的好处是很容易对你的项目进行排序，管理和查找。例如，如果你在工作在一个 Unix 基础的开发环境中，这个路径指向 Home 路径。如果你工作在一个 Windows 的环境中，工作路径将会是 C:/Users/<username>/workspace。总之，有一些原因，你可能需要不选中缺省位置的选择框并且选择一个其它的路径。如果是这样的话，不管那个位置的选项，自己选一个好了。

另外一方面，如果你在 Eclipse 设定（在第二章的最后一节中）中没有选中“使用这个作为缺省并且不要再询问"，你可能被要求定义一个项目的位置。在 Eclipse 的设置中选中“所有的新项目使用缺省工作空间路径设定”（并且提供在新 Android 项目向导位置字段）。如果你在 Eclipse 设定过程中不选中这个选择框，你需要通过点击浏览按钮并导航来选择一个路径。最后三个选项是在属性区域中。这些属性定义了你的项目是如何被统一到 Android 环境中。在包装名称字段，你为程序包装定义，例如：android.app.Activity 或者 com.google.android.map.MapActivity.

注意：

包装名称遵从了标准的 java 命名指导方针，这个方针的建立是为了减少同名程序发布的风险。最高层的包装名称是公司的域名（如 com,org 和 net）这个遵从了域名，如 google。最后，一个为包装内容的描述性标题被提供。在本章中，Hello Wrold 的包装名称将省略 com 来识别，因为这只是一个文本程序而且不会被发布。所有在本书中将来创建的包装将是可发布的并且是用 com 标识符

对于这个 HelloWorldText 应用程序，使用_programmers_guide.HelloWorldText 这个名字。这个名字识别了属于这个程序的编码而且区别开你将开发的其他应用程序。

注意：

如果你注意到你输入的这个屏幕，你会注意到当你输入程序名称，一个错误显示在本向导页面的顶端说你必须正确的填写所有的字段来继续。这个错误信息是提前并且有一些难以理解的因为你还没有填写完其他的字段。如果你看到这样的错误提示信息，忽略它并且继续完成下面两个字段的填写。

下一个属性字段，活动的名称，这个要求输入是因为它会在程序的主屏幕上被提到。想一下，活动是一个显示你应用程序的窗口。没有活动，你的应用程序

将无法做更多的工作。因为 Android 应用程序可以被一些活动组成，新 Android 项目向导需要知道哪一个活动回事缺省的。活动名称的字段是要求输入并且没有缺省值的，所以如果你必须提供一个来继续下去，本例使用 HelloWorldText。这个保证了程序的简单而且在这里是个描述。

最后的属性字段，应用程序名称，应用程序名称描述。这个就是安装在设备上用来管理的应用程序名称。再次说明，为了方便起见，使用 HelloWorldText 作为程序的名称。

提示：

程序名称和活动名称字段不一定要匹配。事实上，很多的编程者习惯于一个老的惯例，那就是程序的开始画面或者叫做主页。使用你感觉舒服的名字，为了说明的目的，本章假定你使用了建议的名字。

点击结束来结束创建过程。本向导运行一个后台程序自动产生支持一个 Android 应用程序所需要所要求文件和文件夹。当过程结束时，你会有你的第一个 Android 应用程序项目，

提示：

如果结束的按钮不可用，你可能字在属性页面的字段内犯了一个错误。确保属性页面填写正确，Eclipse 不会允许任何输入错误的可能引起问题发生的信息。返回确保所有的属性字段是正确的。下一节会仔细检查自动产生的 Android 文件和一些为你应用程序产生的壳条目的目的。

仔细查看 Android 创建的文件 - 第五章(2)

本部分讨论 Android 刚刚创建的新文件。一个非常全面的结构已经为你创建好了，而且，如果你不知道要看什么的话，你最终或许会在不应该放置代码的地方放上代码。有一些 Android 提供的文件你需要去修改，并且有一些你不能修改。知道这些信息会避免你不得不去重新创建项目。

在你打开的项目中，首先看一下 Package Explorer，一个或者二个在主要开发区域面板的左边上的制表符。

注意：

如果你的 Package Explorer 没有打开，你可以通过选择 Windows | Show View | Package Explorer. 激活它。

你应当看到一个根目录，本例中叫做 HelloWorldText。根目录是你所有项目文件的“家”或者“容器”，你自己和 Android 创建的文件都会放在这里。从 Package Explorer 很容易进入。现在会有比较少的一些项目在你的根目录里：一个 AndroidManifest.xml 文件，在一个参考库里的一个包装，和三个目录（res, assets 和 src）。我们轮流来讨论这些项目。

AndroidManifest.xml

AndroidManifest.xml 文件是一个指定全局设定的地方。如果你是一个 ASP.NET 的开发者，你可以认为 AndroidManifest.xml 是 Web.config 和 Global.asax 的

二合一。（如果你不是 APS.NET 的开发者，AndroidManifest.xml 就是意味着是个存放设定的地方）。AndoridManfiest.xml 将包括如程序许可，活动，和意向过滤器等的设定。标准的 AndroidManifest.xml 文件应当包含下面的信息：

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android=http://schemas.android.com/apk/res/android
package="testPackage.HelloWorldText">
<application android:icon="@drawable/icon">
<activity class=".HelloWorldText" android:label="@string/app_name">
<intent-filter>
<action android:value="android.intent.action.MAIN" />
<category android:value="android.intent.category.LAUNCHER"
/>
</intent-filter>
</activity>
</application>
</manifest>
```

如果你创建一个应用程序，你将要在这个文件里增加信息。注意，你提供的包装名称已经列在这里了，同样包括你的活动所需的动作。

引用库和目录 - 第五章(3)

一个引用库的列表也包含在根目录里了。通常，对于一个新手的项目，你应当只看一个库。扩展引用库的分支并且仔细查看当前你的应用程序项目所引用的库。由于它是一个新的 Android 项目，你会在项目引用里看到一个库,那就是 android.jar,Android SDK(如果你熟悉 Java SDK，android.java 是同 Java 的 rt.java 非常类似的文件。在 rt.java 里封装了很多 java 的 API）。Android Plugin 确保了这个是唯一被你应用程序引用的库。应用程序需要引用 SDK 来获得进入在 SDK 库内所有类别，比如你的 Views，Controls 或者甚至谷歌的 API。

注意：

Eclipse 可以允许你增加用户定义的外部库。但是除非你确信这些外部的引用将在 Android 应用程序上工作，所以增加它们之前请三思而后行。

目录（路径）

有三个目录在项目的根目录——res,assets 和 src。每一个都有着显著的目的。这些目录在你应用程序的运行中扮演着完整的角色。

res 目录

res 目录是你项目资源放置并且编译你的应用程序的地方。当你创建一个新的 Android 项目，res 目录包含 3 个子目录:drawable, layout, 和 values。你会在很多的项目中使用 drawable 和 layout 分别放置并显示图形和布局。而 values 目录放置遍及程序全局的字符串。

注意：

一个引用到 res 目录和它内容是被包含在 R.java 文件中，在 src 目录中。我们会在本章的后面详细的讲解。drawable 目录包含你程序可以使用和引用的真实图形。layout 目录放置 XML 文件，当构造它的界面时，main.xml 文件被应用程序引用。在本书的绝大多数应用程序中，你会去编辑在 layout 目录下的 main.xml 文件。这将允许你插入 Views 到程序的可视布局并显示它们。一个原始的 main.xml 文件包含下列代码：

```
<?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="fill_parent"
android:layout_height="wrap_content"
android:text="Hello World, HelloWorldText"
/>
</LinearLayout>
```

最后一个在 res 目录下的文件夹是 values，放置了 XML 文件命名字符串。strings.xml 文件是用来放置程序引用的全局字符串。

assets 目录

assets 目录用来放置“原料”文件的。在这个目录中可以包含为流媒体和动画准备的音频文件。因为 Android 模拟器的 beta 音频驱动没有优化，我不会在本书的应用程序中使用任何的音频文件。

src 目录

src 目录包含项目里所有的源文件。当项目一创立，就会包含两个文件 R.java 和<活动>.Java（本例中是 HelloWorldText.java）

注意：

<activity>.java 总是根据你的活动来命名。

Hello World！自动产生文件的详解 - 第五章(4)

R.java 是一个由 Android plugin 自动产生并添加到你的应用程序中的文件。这个文件包含到 drawable,layout 和 values 目录的指针（或者目录里其它的项目，本例中是字符串和图标）。你不应当必须直接修改这个文件。在你大多数的程序里会总是提到 R.java.为 HelloWorldText 自动产生的代码如下：

```
/* AUTO-GENERATED FILE. DO NOT MODIFY.
*
* This class was automatically generated by the
* aapt tool from the resource data it found. It
* should not be modified by hand.
*/
package testPackage.HelloWorldText;
public final class R {
public static final class attr {
}
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=0x7f040000;
}
}
```

注意：

R.java 文件的注释部分提供了关于这个文件起源的解释。它说明文件由 aapt 工具创建。在第六章，当你创建命令行版本的 Hello World 时，你将用命令行工具创建所有自动产生的文件。

<activity>.java 文件在 src 目录下，你会花费大多数时间在这个文件上。本例是 HelloWorldText.java.这个是你的创建新的 Android 程序向导时由 Android plugin 创建并与活动名称匹配来命名的。不像本部分大多数你已经看过的文件，这个文件完全可以编辑。事实上，如果你不修改代码，它会为了做一点点的事情。

```
Package android_programmers_guide.HelloWorldText;
import android.app.Activity;
import android.os.Bundle;
public class HelloWorldText extends Activity {
/** Called when the activity is first created. */
@Override
public void onCreate(Bundle icicle) {
super.onCreate(icicle);
setContentView(R.layout.main);
}
}
```

在文件上面的三行是标准预处理器指令——那就是，如大多数的编程语言，在程序处理前声明是指令到编译然后运行。在本例中，你在 package

android_programmers_guide.HelloWorldText.有了定义和包含。

下两行通过 android.java 从 Android SDK 中导入特别的包装。

```
import android.app.Activity;
```

和

```
import android.os.Bundle;
```

这些行告诉项目去包括所有你程序里面的代码之前包括所有来自导入包装的代码。这两行对于基本的 Android 程序非常的重要并且不应当被移除。

提示：

如果你在项目里没有看到.android.os.Bundle 的输入声明，在开发窗口展开树形。Eclipse 会给出在第一个下面所有输入的声明，所以你必须展开树形结构来看其余的声明。

现在让我们关注到你的类 HelloWorldText,你会看到它扩展了 Activity class.Activity 被从前一行导入。所有的程序源于 Activity class，并且在 Android 上运行一个程序会需要这个起源。运行并在屏幕上显示某些东西必须从 Activity 起源。

HelloWorldText 的类保持了需要创建，显示并且允许程序的代码。在 HelloWorldText 的类中，现在只有一个方式来定义代码 onCreate( ).

onCreate()方法把冰柱作为一个束。那就是所有点钱状态的信息被搜集作为一个冰柱对象并且被保存在内存了。在本程序中你不能直接处理冰柱，但是你需要知道它的存在和目的。

文件中的下一行是真正可感受到的动作：

```
setContentView(R.layout.main);
```

setContentView()方法把 Activity 的内容设置到指定的源。在本例中，我们通过 R.java 文件中的指针使用 layout 目录里的 main.xml 文件。现在的 main.xml 文件包含了 HelloWorldText 的屏幕和一个 TextView。TextView 起源于 View 并且被用来在 Android 环境中显示文本。回头再看 main.xml 的内容，你会看到它包含了下面的行：

```
android:text="Hello World, HelloWorldText"
```

setContentView()方法被告诉去设置 main.xml 作为当前的 View，并且 main.xml 包含了一个宣称“Hello World, HelloWorldText”的 TextView。现在可能比较安全的去编译并运行 HelloWorldText。要测试这个，运行你的 HellowWorldText 程序。选择 Run|Run to open the Run As dialog box, 选择一个 Android 应用程序，然后点击 OK。

你新建立的项目包含创建 Hello World 应用程序自身的代码。总之，这个并不是太吸引人，而且也没有教你太多参与 Android 应用程序编程。你需要仔细研究项目本身并且看看项目是如何显示“Hello World!”信息的。

当你创建一个新的，由 Android plugin 修改的 main.xml 程序的 Android 项目时究竟发生了什么。这是一个完美修改 Android 中 UI 的一个例子。当项目

被创建时，下面的代码行由 Android SDK 增加到 main.xml 中：

```
<TextView
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:text="Hello World, HelloWorldText"
/>
```

在我讨论 xml 文件中存在的 TextView 时，我没有说它为什么能没有任何相应的代码就可以工作。在本书的早些时候提到有两种方式来为 Android 设计 UI：通过代码，和通过 main.xml 文件。在先前的代码例子中，在 xml 文件中创建了一个 TextView 并且设定文本为“Hello World, HelloWorldText”。编辑 main.xml 中的这一行，按照下面的方式：

```
android:text="This is the text of an Android TextView!"
```

重新运行项目，并且你的运行结果应该如图所示（略）。利用一些时间并且用 xml 的 TextView 做实验。然后你可以转移到用另外一种方式来创建一个 Hello World!应用程序了。

Hellow World! 再来一次 - 第五章(5)

在本部分中，你将创建另外一个 Hello World!这次，你会使用编程代码而不是使用 xml 文件，并且你会自己来做大部分工作。第一步就是把 main.xml 里面已经有的 TextView 代码删除。下面就是 TextView 部分的代码。完全的删除它，使你的应用程序是一个空的壳。

```
<TextView
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:text="Hello World, HelloWorldText"
/>
```

在移除了 TextView 代码以后，你的 main.xml 文件应该像下面这样：

```
<?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"
>
</LinearLayout>
```

现在你有一个干净的 main.xml 文件了，并且一个干净的应用程序壳，你可以开始增加可以在屏幕上显示“Hello World!”的代码了。从打开 HelloWorldText.java 并移除下面的代码行开始：

```
setContentView(R.layout.main);
```

注意：

你仍旧需要为你新的应用程序来设置一个 ContentView；但是你需要执行和现在的这个有一点细微的不同，所以在这里最好把完整的声明移除。

这条使用 setContentView()来把 main.xml 显示在屏幕上。因为你不会去使用 main.xml 来定义你的 TextView,所以你不会去设置你的 view。取而代之，你会用代码来构建 TextView。

下一步是从 android.widget 中导入 TextView 包装。这样你可以进入到 TextView 并且允许你创建自己的实例。把这些代码放置到当前 HelloWorldText.java 文件靠近顶部，现有导入声明的 import android.widget.TextView 的地方;

现在，创建一个 TextView 的实例。通过创建这个 TextView 实例，你可以在屏幕上显示文本而不需要直接修改 main.xml 文件。在 onCreate()声明的后面放置下面的代码：

```
TextView HelloWorldTextView = new TextView(this);
```

注意

TextView 在当前上下文中取得一个句柄作为一个变量。传递这个到 TextView 并和当前的上下文相关联。如果你跟从 SDK 的等级，HelloWorldText 扩展至Activity,而Activity扩展至ApplicationContext,而再扩展至 Context。这就是你如果传递 TextView 的。

先前的代码行创建一个名叫 HelloWorldTextView 的 TextView 的实例,然后例示 HelloWorldTextView,通过设置它到一个新的 TextView。这个被上下文传递的新的 TextView 被完全的例示。

现在，这个 TextView 已经被定义好了，你可以在里面增加文本。下面的代码指定“Hello World!”文本到 TextView:

```
HelloWorldTextView.setText("Hello World!");
```

这一行允许你设定你的 TextView 文本。setText()允许你赋值一个字符串到 TextView。

你的 TextView 已经被创建而且现在包含了你想要显示的信息。但是，如果简单的指定“Hello World”到 TextView 中不会在屏幕上显示任何的东西。如前面所讨论的那样，你需要设置 ContentView 来在屏幕上显示东西。所以，你必须使用下面的代码来设置 TextView 到上下文并且在屏幕上显示：

```
setContentView(HelloWorldTextView);
```

仔细查看本行代码，你会发现你把 setContentView 到 TextView。前面的三行代码是制作你的 Hello World!应用程序。你创建一个 Textview，赋值你的文本，然后显示在屏幕上。所有的事情就是这样，根本不复杂。完整的 HelloWorldText.java 文件应当像下面这样：

```
package android_programmers_guide.HelloWorldText;
import android.app.Activity;
import android.os.Bundle;
import android.widget.TextView;
```

```
public class HelloWorldText extends Activity {
/** Called when the activity is first created. */
@Override
public void onCreate(Bundle icicle) {
super.onCreate(icicle);
/**Hello World JFD */
/**BEGIN */
/**Create TextView */
TextView HelloWorldTextView = new TextView(this);
/**Set text to Hello World */
HelloWorldTextView.setText("Hello World!");
/**Set ContentView to TextView */
setContentView(HelloWorldTextView);
/**END */
}
}
```

现在在 Android 模拟器中那个编译并且允许这个新的 Hello World!应用程序。选择 Run|Run 或者按下 CTRL-F11 在 Android 模拟器中启动这个应用程序。

你刚刚创建了一个完整的 Android 活动。这个小的项目展示了一个常规 Hello World!应用程序的运行。你在Android模拟器中通过设置TextView到Activity's ContentView 中并且在手机上显示“Hello World!”信息。下一节中会用一个细微不同的方式执行一个 Hello World!,使用一个图形。

Hello World! 使用一个图形 - 第五章(6)

在本章，你会使用一个在编程中大家熟知的活动来熟悉 Hello World!应用程序：显示图形。现在的电脑如果不显示图形就太过分了。这些图形显示的重点在于如何让它在屏幕上显示出来的能力。

大概 5 年以前，在手机上显示图形是非常困难的一件事。和图形一起工作是我们这些现代电脑用户认为理所应当的事情之一了。我们每天看着不同类型的窗口，甚至没有想到这个是影像被发送到屏幕的。这个版本的 Hello World! 程序将显示一个 Hello World!的图片。

对于这个应用程序，使用 New Android Project wizard（新 Android 项目向导）来创建一个新的项目并且命名为 HelloWorldImage。

程序创建好后，找到 main.xml 文件并把其中的 TextView 代码删除，这样你就有一个干净的项目文件了。如果你没有删除这个代码，最终将再次显示文本类型的 Hello World!程序。

在你开始写代码之前，你需要一个需要显示的图片。在你可选的图形程序内创建一个小的图片。为了方便起见，选择 Microsoft Paint，但是任何的程序都可以给你想要的图片。

为这个图片命名为 helloworld.png 并且把它保存到

%workspace%/HelloWorldImage/
res/drawable 目录下。

注意：

不要把图片的名称大小写搞混了。图片的名称只应当是小写字母。如果你插入了大写字母，当你试着在 Eclipse 中用这个文件时，会得到一个错误的提示。

在复制这个文件到正确的目录之后。helloworld.png 这个图片应当显示在项目窗口中，在 drawable 目录下。

打开 R.java 并且看一下它的代码。Eclipse 应当增加了一个指针到 helloworld.png.你的 R.java 文件应当同下面的类似：

```
/* AUTO-GENERATED FILE. DO NOT MODIFY.
*
* This class was automatically generated by the
* aapt tool from the resource data it found. It
* should not be modified by hand.
*/
package android_programmers_guide.HelloWorldImage;
public final class R {
public static final class attr {
}
public static final class drawable {
public static final int helloworld=0x7f020000;
public static final int icon=0x7f020001;
}
public static final class layout {
public static final int main=0x7f030000;
}
public static final class string {
public static final int app_name=0x7f040000;
}
}
```

有了一个干净的壳作为起点，并且一个可用的句柄到你要显示的图像，你可以开始增加你的代码了。可以以两种观点来看这个应用程序：XML 基础的 UI 和代码为基的 UI。

Hello World！代码为基的 UI-第五章（7）

假定你对 HelloWorldText 那个部分能够理解了，这个版本的 Hello World!会比较的熟悉了。要开始的话，你需要输入显示图片功能的包装。文本显示使用一个 TextView，图片显示就要用 ImageView 了。因此，你必须输入 ImageView 包装。和 TextView 一样，ImageView 包含在 android.widgets 里：

```
import android.widgets.ImageView;
```

注意

TextView 和 ImageView 都是从 View 派生的。这样就使得两者结构非常的类似并且容易执行。

包装导入（输入）后，你可以创建你的 ImageView 并且在屏幕上显示它了。示例 ImageView 和示例 ImageView 是一样的。创建一个 ImageVIew 示例并且把它传递给上下文使用下面的代码：

```
ImageView HelloWorldImageView = new ImageView(this);
```

下面是能够看到在 ImageView 和 TextView 之间的不同之处。这一步是关于设定你想要显示的东西。在 TextView 例子中，你使用 setText()来设定 TextView 的文本为“Hello World!”，虽然 TextView 和 ImageVIew 都是派生于 View，但是它们还是不同的并且因此要求不同的方式。显然，你也不会来为 ImageView 使用 setText()。你需要使用 setImageResource()来在 ImageView 中显示图片。并把句柄从 R.java（句柄的语法是 R.drawable.helloworld）传递到 helloworld.png:

```
HelloWorldImageView.setImageResource (R.drawable.helloworld);
```

最后，要把图片传输到屏幕，你必须设定 ContentView。正如你在 TextView 做的一样，把 ImageView 传递到 ContentView 中。ContentView 的工作就是把设定到对象的东西传递到屏幕上。

```
SetContentView (HelloWorldImageView);
Your final HelloWorldImage.java file should look like this:
package android_programmers_guide.HelloWorldImage;
import android.app.Activity;
import android.os.Bundle;
import android.widget.ImageView;
public class HelloWorldImage extends Activity {
/** Called when the activity is first created. */
Chapter 5: Application: Hello World! 77
@Override
public void onCreate (Bundle icicle) {
super.onCreate (icicle);
/**Hello World Image JFD*/
/**BEGIN */
/**Create the ImageView */
ImageView HelloWorldImageView = new ImageView(this);
/**Set the ImageView to helloworld.png */
HelloWorldImageView.setImageResource(R.drawable.helloworld);
/**Set the ContentView to the ImageView */
setContentView(HelloWorldImageView);
/**END */
}
}
```

编译 HelloWorldImage 并且在 Android 模拟器中运行。在下一节，你将再次

显示 helloworld.png 但是这次使用 XML 而不是代码。

Hello World! XML 为基的 UI - 第五章（8）

本章通过比较使用 XML 为基的 UI 和代码为基的 UI 来给你一个比较的例子。正如你将要看到的，使用 main.xml 要求和代码为基的方式差不多同样多的代码来把图片发送到屏幕上。但是两个过程的句法不同。

如果在上个例子中所作使用同样的项目，从 HelloWorldImage.java 文件中移除 TextView 代码。干净的文件应该看起来像这个一样：

```
package android_programmers_guide.HelloWorldImage;
import android.app.Activity;
import android.os.Bundle;
public class HelloWorldImage extends Activity {
/** Called when the activity is first created. */
@Override
public void onCreate(Bundle icicle) {
super.onCreate(icicle);
}
}
```

现在你有一个清白的历史可以开始，把上面的移到 main.xml 中，你需要为一个 ImageView 增加定义。开始增加一个空的 ImageView 标签到你的 main.xml 中：

```
<ImageView
/>
```

你需要编辑 ImageView 的 4 个属性：android:id, android:layout_width,android:layout_height, 和 android:src。你会把这些属性添加到标签中，这些控制标签如何在屏幕上显示。

android:id 属性被用来作为 ImageView 的识别符。android:id 属性可以在 ImageView 代码中被提交处理。可以等一会儿在 R.layout.imageview 中使用 @+id/<name>句法来给 ImageView 赋值一个识别符：

```
android:id="@+id/imageview"
```

本行插入一个以 imageview 命名,自动产生的 ID,@+id 到 R.java 内。

下两个你必须定义的属性是：android:layout_width 和 android:layout_height。这些属性控制图片如何填充屏幕。有两个可选择的选项。fill_parent 值定义全部显示图片，wrap_content 显示定义的图片尺寸，可能会丢失图像清晰度。本例中使用 wrap_content:

```
android:layout_width="wrap_content"
android:layout_height="wrap_content"
```

最后一个需要赋值的属性是最重要的变量型的属性：android:src.这个属性指向你要显示的图片。例如，指向属性到 drawable/helloworld 图片：

```
android:src="@drawable/helloworld"
```

Your full ImageView tag should look like this:

```
<ImageView android:id="@+id/imageview"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:src="@drawable/helloworld"
/>
```

最后，在图像显示前，你必须把 main.xml 通过 setContentView 传递到 HelloWorldImage.java 中：

```
setContentView(R.layout.main);
```

编译并运行 HelloWorldImage。

在本章结束前，再试一下另外一件事。回到 main.xml 中并且把 wrap_content 改成 fill_parent。完成后，你的 main.xml 文件应当如下：

```
<?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"
>
<ImageView android:id="@+id/imageview"
android:layout_width="fill_parent"
android:layout_height="fill_parent"
android:src="@drawable/helloworld"
/>
</LinearLayout>
```

再次运行程序来检查 wrap_content 和 fill_parent 之间的不同之处。

使用 TextView 和 ImageView

使用一些在本章学到的技巧和技术来创建一个新的 Hello World!应用程序。创建一个即用 TextView 也用 ImageView 的程序，图片放在屏幕上并且带有一个文本的标题。这个比在一个 Activity 中使用一个 View 多一点难度。多用 Views 看看你能创建什么。

下一章还会待在 Hello World!上，不过还讨论命令行编程。

问专家

Q:Android 和大多数的 APIs 一样有 Label 或者 LabelView 可以用吗？

A:没有。所有的文本显示通过 TextView 显示。你可以，和其他人一样，自定义一个和 Label 功能的 View，并把它命名为 LabelView,但是 Android 本身并没有 LabelView 这个包装。

Q:有任何的优点使用<应用程序>.java，而不是 main.xml 来创建 Views 吗?

A:没有文件说速度或者处理器方面的二者之间的差别，但是一个关键的优点是，使用 main.xml，需要为你的 Activity 预先确定一组 Views 。然后，在编码中，

你可以从一个 View 跳到另外一个 View，而不需要手动创建它们。

第六章 使用命令行工具和 Android 模拟器

使用命令行工具和 Android 模拟器 - 第六章(1)

关键技能和概念

- 使用 Android SDK 命令行工具
- 创建一个命令环境
- 用一个壳导航到 Android 服务
- 在 Linux 里使用 Android SDK

到目前为止，本书包含了一些非常宽的科目关于学习如何运行 Android 平台。就这一点来说，对于使用 Eclipse 来创建并运行一个小的 Adnroid 应用程序，你应该非常的舒服。你创建一个新的项目，编辑 main.xml 文件和<activity>.java 文件，然后编译 R.java 文件。这些是创建 Android 应用程序的一些基本技能。

在本章中，你会通过用命令行应用程序开发来扩展你的这些技能。Android 开发没必要必须限定在 Eclipse IDE 环境里进行。Android SDK 提供了许多命令行工具，可以在没有图形化的 IDE 的帮助下，开发完整的应用程序。你会使用这些命令行工具首先在 Windows，然后再 Linux 下来创建，编译和运行 Hello World! 应用程序。

利用 Windows CLI 创建一个壳活动 - 第六章(2)

Android SDK 有很多的工具来帮助你创建并编译 Android 应用程序。这些工具帮助那些不愿意使用，或者没有所支持 GUI IDE 系统的用户来工作的。总之，如果你一直在用 Eclipse 在编程，你仍然需要知道 Android SDK 命令行工具和它的功能。

当你运行 Android 相关的功能，如创建一个 Android 项目或者在 Android 模拟器内运行一个应用程序，你实际上是在呼叫到命令行工具的连接器。无论是从命令行接口或者 GUI IDE 运行，这些 Android 命令行工具是 Android SDK 的真正核心。

在下面的章节中，我演示 Android 工具的功能。ActivityCreator.bat 是一个强有力的工具，被用来为你的程序建立一个活动壳（Activty Shell）。

运行 ActivityCreator.bat - 第六章(3)

ActivityCreator.bat 文件应当在 Android SDK 的…/tools/文件目录下。大多数“前向”命令行工具都放置在工具目录的根目录下。“前向”工具是依次呼叫在工具根目录下更深目录的工具。ActivityCreator.bat 是工具根目录下一个示例的工具，它运行时会呼叫另外一个工具。使用 vi，Notepad 或者一个文本

编辑器，打开 ActivityCreator.bat。它应当包含下面的代码：

注意

ActivityCreator.bat 是定义为 Microsoft Windows 版本的 Android SDK。在本章的后面部分，你将会学习 ActivityVreator.py。这个是 Linux 版本的 ActivityCreator。

```
@echo off
rem Copyright (C) 2007 Google Inc.
rem
rem Licensed under the Apache License, Version 2.0 (the "License");
rem you may not use this file except in compliance with the License.
rem You may obtain a copy of the License at
rem
rem http://www.apache.org/licenses/LICENSE-2.0
rem
rem Unless required by applicable law or agreed to in writing,
software
rem distributed under the License is distributed on an "AS IS" BASIS,
rem WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or
implied.
rem See the License for the specific language governing permissions
and
rem limitations under the License.
rem don't modify the caller's environment
setlocal
"%~dp0\lib\activityCreator\activityCreator.exe" %*
```

浏览整个的 rem 声明（批处理文件注释声明），你会看到在文件的底部有一个实用的代码。ActivityCreator.bat 被用来呼叫…/tools/lib/activitycreator/ 目录里的 ActivityCreator.exe。这个 ActivityCreator.bat 是一个工具的示例，它只是放置在 SDK 其它工具的前端。

所以，ActivityCreator.bat(或者 ActivityCreator.exe)做了什么？ActivityCreator 被用来建立指向在哪里需要开始开发你的应用程序初始文件的开发环境。这个路径结构和在第五章第五章(1)程序：Hello World!讨论过的结构一致。ActivityCreator.bat 创建 R.java, AndroidManifest.xml, 和所有你应用程序需要的支持文件。

让我们现在转到命令行环境并且浏览 ActivityCreator。在开始菜单，点击运行，在运行的对话框内输入 CMD 或者 COMMAND，然后点击确定。

执行这个命令会启动命令窗口。这个窗口和老版本的 DOS 操作系统环境相同。命令窗口出现后，在光标>后输入 ActivityCreator

提示

Microsoft 命令提示符接口没有大小写限制。在缺省情况下，如果你使用了大小写限制的不同的开发环境，本章中显示的屏幕截图会不同。

运行命令 ActivityCreator，实际运行的是 ActivityCreator.bat，产生下面的输出：

```
Activity Creator Script
Usage:
activityCreator [--out outdir] [--ide intellij] yourpackagename.ActivityName
Creates the structure of a minimal Android application.
The following will be created:
- AndroidManifest.xml: The application manifest file.
- build.xml: An Ant script to build/package the application.
- Res: The resource directory.
- Src: The source directory.
- src/your/package/name/ActivityName.java the Activity java class.
packageName
is a fully qualified java Package in the format <package1>.<package2>... (With at
least two components).
- Bin: The output folder for the build script.
Options:
--out <folder>: specifies where to create the files/folders.
--ide intellij: creates project files for IntelliJ
```

这个输出简单的指示了你需要提供更多的信息来运行 ActivityCreator。更确切的是，你需要传递给命令一个你需要建造的壳应用程序的位置。

注意

从 ActivityCreator 输出的命令给了你很多不仅仅是你没有提供足够信息的信息。它给了你一个完整的使用工具创建的文件列表。这个文件列表和第五章第五章(1)程序：Hello World!看起来相似。虽然 build.xml 没有直接展示在 Eclipse 用户目前。

回到命令窗口并且使用下面的选项运行 ActivityCreator（如果你使用 Unix/Linux 环境编程，ActivityCreator 同样接受 unix-风格路径参数）：

```
--out
c:\AndroidHelloWorld\android_programmers_guide.HelloWorldCommandLine
```

--out 选项告诉 ActivityCreator 你想要它输出点东西。这个命令选项使用两个参数，<folder>和<package.activity>。第一行告诉 ActivityCreator 在一个不存在的文件夹里创建壳应用程序。c:\AndroidHelloWorld.

提示

如果你定义的文件夹或者路径不存在，AcitivityCreator 将会在过程中自动创建一个。

--out 的第二个参数是包装名称和活动名称。根据前面章节的习俗，这个实例在本项目中使用 android_programmers_guide 作为包装的名字并且 HelloWorldCommandLine 作为活动的名称。

注意

成功运行 ActivityCreator 并设置你的初始环境和运行新的 Android Project wizard 是一致的。

NOTE
the parameters needed to successfully run ActivityCreator and set up your initial
environment are the same as those required by the New Android Project wizard.

在新命令行选项和参数下运行 ActivityCreator。你应当从工具的输出看到下面的内容：（略）。下面的章节涵盖了由 ActivityCreator 创建的文件，因为和由 Eclipse 创建的文件有一些不同。

项目结构 - 第六章（4）

ActivityCreator 为你的开发创建了一组文件目录和文件。浏览 c:\AndroidHelloWorld\来看一下它的结构。ActivityCreator 创建的结构如下图所示（略）

因为在 Eclipse 环境之外工作，你有一个不同的环境。当你在例如 Eclipse 的 IDE 内工作时，一些特定的功能在场景之后为你工作。假如你工作时没有任何 IDE 的帮助，ActivityCreator 创建了一个文件来概述编译器如何的工作来创建你的项目。当手动运行 ActivityCreator,为你创建 build.xml 文件。当你使用 Eclipse 来开始一个 Android 项目时，这个文件并没有被创建。

它包含了一个指令组，解释了如果转变你的.java 文件到一个功能性的 Android 项目中。这个 build.xml 文件告诉编译器它需要怎么做来创建你的应用程序。在这个例子中的编译器是 Apache ANT，一个 Java 基础的工具，被用来使用构造脚本文件到编译的项目。从 http://ant.apache.org/bindownload.cgi. 下载 ANT。

一旦你下载并安装了 ANT，你必须把它增加的 PATH 声明中。在 Windows 环境中，右击“我的电脑"，并且选择“属性”来改变 PATH 声明。build.xml 文件被特地为 ANT 创建用于编译你的 Android 应用程序。它应该在你项目的根目录，如上图所示。用文本编辑器打开 build.xml 并看一下它里面有些什么。

第一个部分的 build.xml 包含了可以被用户编辑的代码。这部分是其它部分的开始，因为剩下的部分不应当被修改。

```
<?xml version="1.0" ?>
<project name="HelloWorldCommandLine" default="package">
<property name="sdk-folder" value="c:\Android\android-sdk_m5
rc14_windows\android-sdk_m5-rc14_windows" />
<property name="android-tools" value="c:\Android\android-sdk_m5
rc14_windows\android-sdk_m5-rc14_windows\tools" />
<property name="android-framework" value="${android-
tools}/lib/framework.aidl"
/>
```

```
<!-- The intermediates directory -->
<!-- Eclipse uses "bin" for its own output, so we do the same. -->
<property name="outdir" value="bin" />
```

第一部分的 build.xml 包含下面属性的值：

- Project name 项目名称
- Android SDK location Android SDK 位置
- Android tools location Android 工具位置
- Android framework location Android 框架位置
- Output location 输出位置

如果你需要为项目改变任何的这些参数，你可以在这个文件做。但是在 build.xml 接下来的参数，你会立即看到通知你的警告，告诉你不应当去编辑剩下部分的值：

```
<!-- No user servicable parts below. -->
```

Following this warning in build.xml is a list of parameters and values that are critical to the proper creation of your project. This list includes compiler options, input directories, and tool locations. Take a look at the following output of the core processing information of build.xml:

注意

当 Android 建议反对改变下面这些参数时，如果你非常熟悉 ANT 是如何工作的，你可以修改这些选项来符合你特定的需求。

```
<!-- Input directories -->
<property name="resource-dir" value="res" />
<property name="asset-dir" value="assets" />
<property name="srcdir" value="src" />
<!-- Output directories -->
<property name="outdir-classes" value="${outdir}/classes" />
<!-- Create R.java in the source directory -->
<property name="outdir-r" value="src" />
<!-- Intermediate files -->
<property name="dex-file" value="classes.dex" />
<property name="intermediate-dex" value="${outdir}/${dex-file}" />
<!-- The final package file to generate -->
<property                                          name="out-package"
value="${outdir}/${ant.project.name}.apk"/>
<!-- Tools -->
<property name="aapt" value="${android-tools}/aapt" />
<property name="aidl" value="${android-tools}/aidl" />
<property name="dx" value="${android-tools}/dx" />
<property name="adb" value="${android-tools}/adb" />
<property name="android-jar" value="${sdk-folder}/android.jar" />
```

```
<property name="zip" value="zip" />
<!-- Rules -->
<!-- Create the output directories if they don't exist yet. -->
<target name="dirs">
<mkdir dir="${outdir}" />
<mkdir dir="${outdir-classes}" />
</target>
<! -- Generate the R.java file for this project's resources. -->
<target name="resource-src" depends="dirs">
<echo>Generating R.java...</echo>
<exec executable="${aapt}" failonerror="true">
<arg value="compile" />
<arg value="-m" />
<arg value="-J" />
<arg value="${outdir-r}" />
<arg value="-M" />
<arg value="AndroidManifest.xml" />
<arg value="-S" />
<arg value="${resource-dir}" />
<arg value="-I" />
<arg value="${android-jar}" />
</exec>
</target>
<!-- Generate java classes from .aidl files. -->
<target name="aidl" depends="dirs">
<apply executable="${aidl}" failonerror="true">
<arg value="-p${android-framework}" />
<arg value="-I${srcdir}" />
<fileset dir="${srcdir}">
<include name="**/*.aidl"/>
</fileset>
</apply>
</target>
<!-- Compile this project's .java files into .class files. -->
<target name="compile" depends="dirs, resource-src, aidl">
<javac encoding="ascii" target="1.5" debug="true" extdirs=""
srcdir="."
destdir="${outdir-classes}"
bootclasspath="${android-jar}" />
</target>
<! -- Convert this project's .class files into .dex files. -->
```

```
<target name="dex" depends="compile">
<exec executable="${dx}" failonerror="true">
```

```
<arg value="-JXmx384M" />
<arg value="--dex" />
<arg value="--output=${basedir}/${intermediate-dex}" />
<arg value="--locals=full" />
<arg value="--positions=lines" />
<arg path="${basedir}/${outdir-classes}" />
</exec>
</target>
<!-- Put the project's resources into the output package file. -->
<target name="package-res-and-assets">
<echo>Packaging resources and assets...</echo>
<exec executable="${aapt}" failonerror="true">
<arg value="package" />
<arg value="-f" />
<arg value="-c" />
<arg value="-M" />
<arg value="AndroidManifest.xml" />
<arg value="-S" />
<arg value="${resource-dir}" />
<arg value="-A" />
<arg value="${asset-dir}" />
<arg value="-I" />
<arg value="${android-jar}" />
<arg value="${out-package}" />
</exec>
</target>
<!-- Same as package-res-and-assets, but without "-A ${asset-dir}" -
->
<target name="package-res-no-assets">
<echo>Packaging resources...</echo>
<exec executable="${aapt}" failonerror="true">
<arg value="package" />
<arg value="-f" />
<arg value="-c" />
<arg value="-M" />
<arg value="AndroidManifest.xml" />
<arg value="-S" />
<arg value="${resource-dir}" />
<!-- No assets directory -->
<arg value="-I" />
<arg value="${android-jar}" />
<arg value="${out-package}" />
</exec>
</target>
```

```
<!-- Invoke the proper target depending on whether or not
an assets directory is present. -->
<! -- TODO: find a nicer way to include the "-A ${asset-dir}"
argument
only when the assets dir exists. -->
<target name="package-res">
<available file="${asset-dir}" type="dir"
property="res-target" value="and-assets" />
<property name="res-target" value="no-assets" />
<antcall target="package-res-${res-target}" />
</target>
<!-- Put the project's .class files into the output package file. -->
<target name="package-java" depends="compile, package-res">
<echo>Packaging java...</echo>
<jar destfile="${out-package}"
basedir="${outdir-classes}"
update="true" />
</target>
<!-- Put the project's .dex files into the output package file.
Use the zip command, available on most unix/Linux/MacOS systems,
to create the new package (Ant 1.7 has an internal zip command,
however Ant 1.6.5 lacks it and is still widely installed.)
-->
<target name="package-dex" depends="dex, package-res">
<echo>Packaging dex...</echo>
<exec executable="${zip}" failonerror="true">
<arg value="-qj" />
<arg value="${out-package}" />
<arg value="${intermediate-dex}" />
</exec>
</target>
<!-- Create the package file for this project from the sources. -->
<target name="package" depends="package-dex" />
<!-- Create the package and install package on the default emulator -
->
<target name="install" depends="package">
<echo>Sending package to default emulator...</echo>
<exec executable="${adb}" failonerror="true">
<arg value="install" />
<arg value="${out-package}" />
</exec>
</target>
</project>
```

现在你对于 build.xml 在人工下，命令行创建的 Android 项目是如何使用应

该有了好的理解，你可以开始来编辑你的项目文件并且创建一个 Android 活动。第一个你需要看的文件是 main.xml。使用 Windows 资源管理器，在 AndroidHelloWorld\res\layout 目录下找到 main.xml。

在 Windows CLI 下创建 Hello World!活动 - 第六章（5）

在本部分中，你会使用 Windows 命令行接口来编辑项目文件。在上一章中，项目文件是由 ActivityCreator.bat 创建的。你将不使用 Eclipse 来编辑这些文件并增加一些代码。

编辑项目文件

在一个 XML 编译器或者（如果你没有一个 XML 编辑器）记事本打开 main.xml 文件。这样你就可以编辑文件并且删除里面的<TextView/>定义。保存过后的 main.xml 文件是一个空壳。这样你就得到一个编辑<activity>.java 文件的平台了。<activity>.java 文件在更深层次的文件夹里，AndroidHelloWorld\src\android\programmers\guide。来创建你的 Hello World!应用程序，增加下面的代码行来创建，设置并使用一个 TextView:

```
/**Hello World JFD */
/**BEGIN */
/**Create TextView */
TextView HelloWorldTextView = new TextView(this);
/**Set text to Hello World */
HelloWorldTextView.setText("Hello World!");
/**Set ContentView to TextView */
setContentView (HelloWorldTextView);
/**END */
```

别忘了增加 TextView 包装到文件的开始部分：

```
import android.widget.TextView;
```

完成后的 HelloWorldCommandLine.java 文件应当看上去和下面一样（略）。你的项目文件现在应当被设置了。你现在可以在 Android 模拟器内编译并运行你的应用程序了

增加 JAVA_HOME 第六章 （6）

在编译你的项目之前，必须增加另外一个环境变量到你的 PC-JAVA_HOME，可以指向你的 JDK。即使它只是个 PATH 声明，你也必须创建一个新的名为 JAVA_HOME 的变量。

注意
JAVA_HOME 变量是必须的仅仅是因为你使用命令行环境。如果你只使用 Eclipse，你不用增加它。

1、右击“我的电脑”，并选择“属性”。

2、在系统属性下，选择高级选项并点击环境变量按钮。然后会打开一个环境变量窗口。
3、点击新建按钮来增加一个变量名为_HOME，它的值应该是你 Java SDK 的完整路径。见下图（略）

编译并安装应用程序 第六章(7)

是时候来做一个真正的测试了。你现在可以编辑你的命令行项目了。要编译项目，使用 ANT。一旦项目编译完成，你需要在模拟器中安装它。

用 ANT 编译项目

如果运行 ANT 时出错该怎么办？ 第六章(8)

在你设置好 JAVA_HOME 环境变量并且 ANT 在你的 PATH 声明之后，你应当可以导航到含有 build.xml 文件的文件夹，并且只要简单的运行 ant 命令。在你的项目路径下运行 ant。如下：（略）。

运行 ant 的结果就是一个.apk 文件会直接安装到手机（模拟器）中，总之，Eclipse 在模拟器中直接为你安装。而这里你需要使用 Andorid Debug Bridge（adb)工具来安装应用程序，下一节叙述

如果运行 ANT 时出错该怎么办？

当你运行 ANT 时出错该怎么办呢？不用害怕。因为在写本书时，Android 还只是一个刚发布的阶段，有些项目可能需要被纠正，当你使用一项新技术时，总会有一些小的更改会发生。当我第一次试着运行 ant 并且编译我的项目时，我收到了一个错误，如下图（略）。

一些问题的研究在谷歌的 Android 开发者论坛上，一个重写的 build.xml 纠正了一些提供到 ANT 的命令。在修改过的主要部分已经被加粗。和原来的那个文件比较一下你会注意到明显的不同。

```
<? Xml version="1.0" ?>
<project name="HelloWorldCommandLine" default="package" basedir=".">
<property name="sdk-folder" value="c:\Android\android-sdk_m5
rc14_windows\android-sdk_m5-rc14_windows" />
<property name="android-tools" value="c:\Android\android-sdk_m5
rc14_windows\android-sdk_m5-rc14_windows\tools" />
<property          name="android-framework"          value="${android-
tools}/lib/framework.aidl"
/>
<! -- The intermediates directory -->
<! -- Eclipse uses "bin" for its own output, so we do the same. -->
<! -- Use full path for output dir - FIX - BLOCK START -->
<property name="outdir" value="${basedir}/bin" />
<! -- Use full path for output dir - FIX - BLOCK END -->
<! -- No user servicable parts below. -->
<!-- Input directories -->
```

```
<property name="resource-dir" value="res" />
<property name="asset-dir" value="assets" />
<property name="srcdir" value="src" />
<!-- Output directories -->
<property name="outdir-classes" value="${outdir}/classes" />
<!-- Create R.java in the source directory -->
<property name="outdir-r" value="src" />
<!-- Intermediate files -->
<property name="dex-file" value="classes.dex" />
<property name="intermediate-dex" value="${outdir}/${dex-file}" />
<!-- The final package file to generate -->
<property                                          name="out-package"
value="${outdir}/${ant.project.name}.apk"/>
<!-- Tools -->
<property name="aapt" value="${android-tools}/aapt" />
<property name="aidl" value="${android-tools}/aidl" />
<condition        property="dx"        value="${android-tools}/dx.bat"
else="${android
tools}/dx" >
<os family="windows"/>
</condition>
<property name="dx" value="${android-tools}/dx" />
<property name="zip" value="zip" />
<property name="android-jar" value="${sdk-folder}/android.jar" />
<!-- Rules -->
<!-- Create the output directories if they don't exist yet. -->
<target name="dirs">
<mkdir dir="${outdir}" />
<mkdir dir="${outdir-classes}" />
</target>
<!-- Generate the R.java file for this project's resources. -->
<target name="resource-src" depends="dirs">
<echo>Generating R.java...</echo>
<exec executable="${aapt}" failonerror="true">
<arg value="compile" />
<arg value="-m" />
<arg value="-J" />
<arg value="${outdir-r}" />
<arg value="-M" />
<arg value="AndroidManifest.xml" />
<arg value="-S" />
<arg value="${resource-dir}" />
<arg value="-I" />
<arg value="${android-jar}" />
```

```
</exec>
</target>
<!-- Generate java classes from .aidl files. -->
<target name="aidl" depends="dirs">
<apply executable="${aidl}" failonerror="true">
<fileset dir="${srcdir}">
<include name="**/*.aidl"/>
</fileset>
</apply>
</target>
<!-- Compile this project's .java files into .class files. -->
<target name="compile" depends="dirs, resource-src, aidl">
<javac encoding="ascii" target="1.5" debug="true" extdirs=""
srcdir="."
destdir="${outdir-classes}"
bootclasspath="${android-jar}" />
</target>
<!-- Convert this project's .class files into .dex files. -->
<target name="package-dex" depends="dex, package-res">
<echo>Packaging dex...</echo>
<exec executable="${zip}" failonerror="true">
<!--<arg value="-Xmx384M" />-->
<!-- Move Xmx parameter to dx.bat - FIX - BLOCK END -->
<arg value="--dex" />
<arg value="--output=${intermediate-dex}" />
<arg value="--locals=full" />
<arg value="--positions=lines" />
<arg path="${outdir-classes}" />
</exec>
</target>
<!-- Put the project's resources into the output package file. -->
<target name="package-res-and-assets">
<echo>Packaging resources and assets...</echo>
<exec executable="${aapt}" failonerror="true">
<arg value="package" />
<arg value="-f" />
<arg value="-c" />
<arg value="-M" />
<arg value="AndroidManifest.xml" />
<arg value="-S" />
<arg value="${resource-dir}" />
<arg value="-A" />
<arg value="${asset-dir}" />
<arg value="-I" />
```

```
<arg value="${android-jar}" />
<arg value="${out-package}" />
</exec>
</target>
<!-- Same as package-res-and-assets, but without "-A ${asset-dir}" -
->
<target name="package-res-no-assets">
<echo>Packaging resources...</echo>
<exec executable="${aapt}" failonerror="true">
<arg value="package" />
<arg value="-f" />
<arg value="-c" />
<arg value="-M" />
<arg value="AndroidManifest.xml" />
<arg value="-S" />
<arg value="${resource-dir}" />
<!-- No assets directory -->
<arg value="-I" />
<arg value="${android-jar}" />
<arg value="${out-package}" />
</exec>
</target>
<!-- Invoke the proper target depending on whether or not
an assets directory is present. -->
<!-- TODO: find a nicer way to include the "-A ${asset-dir}" argument
only when the assets dir exists. -->
<target name="package-res">
<available file="${asset-dir}" type="dir"
property="res-target" value="and-assets" />
<property name="res-target" value="no-assets" />
<antcall target="package-res-${res-target}" />
</target>
<!-- Put the project's .class files into the output package file. -->
<target name="package-java" depends="compile, package-res">
<echo>Packaging java...</echo>
<jar destfile="${out-package}"
basedir="${outdir-classes}"
update="true" />
</target>
<!-- Put the project's .dex files into the output package file. -->
<target name="package-dex" depends="dex, package-res">
<echo>Packaging dex...</echo>
<exec executable="${zip}" failonerror="true">
<arg value="-qj" />
```

```
<arg value="${out-package}" />
<arg value="${intermediate-dex}" />
</exec>
</target>
<!-- Create the package file for this project from the sources. -->
<target name="package" depends="package-dex" />
</project>
```

在修改过 build.xml 之后，你可以重新运行

用 adb 安装你的应用程序 第六章(9)

第一步是启动你的模拟器。在 Android/tools 文件夹找到 emulator.exe 文件并且执行它。这样就会启动 Android 服务器。那就是启动了模拟器同时在你的电脑上启动了一个虚拟的手机。如下图(略）。然后你就可以使用不同的工具来和服务器交互了，和安装应用程序和呼叫一个壳环境一样。要在 Android 服务器安装你的命令行应用程序，你需要使用 adb。adb 是你到服务器的连接，同模拟器一同开启。

adb 包含了很多有用的功能允许你和 Android 服务器交互；其中一个功能可以让你安装应用程序。

表格 6-1 列出了 adb 接受的命令描述。

要复制你的应用程序到服务器，打开一个 Windows 命令提示符窗口并且导航到 build.xml 文件所在的路径。对于 adb，syntax 命令如下：

```
adb install <apk path>
```

如果应用程序正确的安装到手机，你会得到一个命令行关于包装大小的反馈。如下。（略）。

命令	描述
install <path>	安装应用程序到服务器
pull <remote file> <local file>	将远程文件拉出服务器
push <local file> <remote file>	将本地文件推进服务器
shell	在服务器上打开一个壳环境
forward <local port> <remote port>	从一个端口转递流量到另外一个端口（到或者从服务器上）
start-server	启动服务器
kill-server	停止服务器
ppp <tty> <params>	通过 USB 使用一个 ppp 连接

devices	列出可用的模拟器
help	列出 adb 的命令
version	显示 adb 的版本

表 6-1 adb 命令

转到运行的模拟器，你将会看到应用程序安装到手机上。

运行应用程序产生了一个错误怎么办 - 第六章(10)

我第一次在使用新的 build.xml 文件后，运行这个应用程序时，我在 Android 模拟器上接受到了一个错误。如下图（略）。错误指出一个丢失的类。

注意

你可能会或者不会遇到同样的错误。关键看本书发行时，哪个版本的 Android SDK 是可用的，你应当跟从这里的问题解决步骤，因为在后续的项目中会对你有所帮助。

这个错误似乎指出了一个事实，那就是在 HelloWorldCommandLine.apk 文件中丢失了一个类。我可以简单的自己去纠正这个错误而不用任何的 Android SDK 命令行工具。

根据结果，.apk 文件就是一个.zip 文件。就是说你可以用.zip 解压缩文件打开它。下面的插图就是用 winrar 打开 HelloWorldCommandLine.apk 文件后的样子。（略）。

丢失的是 classes.dex。这个是我的类的编译过的 Dalvik 可执行性文件。导航到 Android 项目下 bin 文件夹，我可以看到 ANT 成功的编译并且创建了 classes.dex 文件。这个文件只是被留在了 HelloWorldCommandLine.apk 文件之外了。在 Winrar 打开的 HelloWorldCommandLine.apk 状态下，我可以把 classes.dex 文件拖进 HelloWorldCommandLine.apk。在 classes.dex 文件被加入 HelloWorldCommandLine.apk 后可以保存并关闭文件了。

卸载一个较早的活动 - 第六章(11)

在你增加文件到运行的服务器之前，你将要卸载前一个版本的 HelloWorldCommandLine。在安装另外一个程序之前，卸载前一个版本的程序不是必须要做的事。但是，为了更好的查看如何的与服务器交互，在开始前，还是卸载前一个版本的程序吧。

保持 Android 模拟器在开启状态，返回到命令行提示符环境并且允许 adb 壳命令，它会打开 Android 服务器的壳环境。如果你成功了，你的命令提示符会从>变成#。现在你在 Android 服务器中有一个打开的壳。有很多的功能现在可以用，但是现在只关注一个：移除旧的 HelloWorldCommandLine.apk 文件。

提示

记住，Android 是一个操作环境。你在壳中可以使用的是标准的 POSIX 命令。

在 Android 服务器中，用户安装的程序被保留在/data/app 路径下。使用 cd，导航到 app 路径，如下图（略）。运行 ls 命令来列出这个路径下所有的文件。你将看到一个 HelloWorldCommandLine.apk 文件。这个文件展示了你活动的安装。

现在你已经在服务器上定位了应用程序，你可以移除它了。使用命令语法 rm HelloWorldCommandLine.apk 来移除应用程序。下图就是 rm 命令（略）。如果成功，不会返回任何的信息。随后使用 ls 命令表明，文件已被移除。

警告

因为技术上你通过壳登入了一个 Linux 服务器，所有在壳内运行的命令是区分大小写的。

应用程序移除后，输入 exit 来退出壳并返回到你的命令提示符。

重新安装并启动应用程序 - 第六章（12）

你现在可以使用 adb 重新安装应用程序了：

```
Adb install HelloWorldCommandLine.apk
```

一旦应用程序被安装回服务器，转到模拟器。从模拟器中启动应用程序。它应当能正常工作，如下图（略）。

现在我们已经谈论过如何在 Windows 内创建和编辑文件的过程，让我们看看在 Linux 上会怎么样。即使你是一个顽固的 Windows 用户，你可能需要注意下面的章节。我发现了对编程绝对有利的开源工具。

Linux 上的 Hello World! 第六章（13）

很多的程序员，特别是对开放源代码软件有兴趣的程序员喜欢使用 Linux 作为平台。谷歌和开放手机联盟已经为这些程序员准备了 Android SDK。这个 SDK 实际上是同样的 SDK（因为 java 是移动性的），但是被创建的工具特定的运行在 Linux 上。当我开始写这本书的时候，我在使用一个老版本的红帽 Linux 作为我的 Linux 平台。我下载并安装了 Eclipse 和 Android SDK。然而，它很快成为可以安全运行 Android 的 Linux 的一些限制。因为最低要求，你必须有一个支持 libstdc++.so.6 的 Linux。Android 文档列出了 Ubuntu Dapper Drake 作为一个 Linux 的测试版本。

如果你还没有决定使用哪一个版本，你可以放心的使用。不幸的是，当我试图安装最新版本的 Ubuntu 的时候，我电脑的硬件有个问题。于是我决定移除推荐的并且试着用一些新的东西。

当我决定放弃红帽，我决定使用 Fedora 8。本书的下面部分所使用的 Linux 版本的例子都是从 Fedora 8 而来。不过，它们应当在你选择的软件版本上工作的没有问题。

注意

如果你选择 Fedora 8，会有一个叫做 Fedora Eclipse 的定制包装。如果你试图

为 Fedora Linux 安装 Android Plugin（使用本书早些时候的概述），它会提示一个错误要求 plugin org.eclipse.wst.sse.u。你可以要下面两种方式解决：

下载最新 Linux 版的 Eclipse，或者使用 Fedora 的自动升级程序，这个可以使 Linux 版的 Fedora Eclipse 成为最新。然后可以在这个 Eclipse 里使用 Android SDK 了。

配置 PATH 声明

第一步就是配置 PATH 声明。路径就是一个路径清单，当一个命令被执行时，操作系统会在这个路径下寻找该命令。要查看你当前配置的路径，从一个 terminal 里运行下面的内容：

```
echo $PATH
```

你会得到像下图(略)一样的一些路径声明。使用输出命令来增加 Android 到 PATH 声明中：

```
export PATH=$PATH:<android path>
```

在 Linux 中编辑 PATH 声明会只是在当前的 terminal 部分改变 PATH 声明。要永久的改变你的 PATH 声明。你必须编辑.bash_profile.使用 vi 来编辑.bash_profile，如下图所示（略）。

如你所见，PATH 声明清晰可见。使用命令：i 来使 vi 成为插入模式，然后增加 Android 到 PATH 中。然后按下 ESC 按钮，使用命令:w 来写文件，然后使用:q 来退出。

Linux 版本的 Android SDK 与一个 Python 脚本一起提供，activityCreator.py，这个被用来创建你的初始项目。不管怎么说，我喜欢手动创建路径来确保它在我需要它在的地方。使用 mkdir 来为你的项目创建一个目录。

创建好项目目录后，你可以运行 activityCreator.py Python 脚本。这个脚本的语法非常接近于 Windows.bat 文件：

```
activityCreator.py --out <output directory>
package.activityName
```

使用 activityCreator.py 脚本来设置你的项目。看看下图 activityCreator.py 输出的脚本（略）。

提示

activityCreator.py 命令是由 sudo 前缀。sudo 命令被用来模拟其他用户的许可（本例为根目录），如果你没有足够的许可来运行要求的命令。在我安装的 Fedora，我的用户帐号没有权利与根目录交互。

项目建好后，编辑 HelloWorldLinux.java 文件并增加 TextView。你可以用很多中方法在 Linux 中编辑.java 文件。可以再次使用 vi，或者你可以使用一个如下图的标准文本编辑器（略）。

最后，从 main.xml 中移除定义的 TextView。你现在编译你的 Linux 版本的

Hello World!应用程序有两个小的改变。要编译应用程序，使用 ANT（这个在 Windows 环境一样）。伺服 ANT 应当被预先安装在你的 Linux 下，特别是当你使用 Fedora 8.如果你没有使用 Fedora 8,你需要为伺服 ANT 下载，安装并设置路径。

当你运行 ant，你应当看到一个如下图的输出（略）。

最后，你需要启动 Android 模拟器来安装你的应用程序。保持模拟器开启的状态下，执行下面的命令：

```
adb install HelloWorldLinux.apk
```

这个将安装应用程序到 Linux Android 服务器。如果命令运行成功，你应当可以在模拟器运行活动了。下一章，研究如何使用 Android SDK 来对图片事件作出反应。

在 CLI 中创建一个图片基础的 Hello World! 第六章（14）

在本章中使用命令行工具来从第五章中重新创建图片为基础的 Hello World! 当你创建这个项目时，记住下面的事宜：

- 在 res 文件夹中放置图片。
- 检查创建 R.java 所需要的带有指向图片的任何工具。
- 使用 ANT 编译项目。
- 使用 adb 命令来安装并推动应用程序到你的模拟器中。

问专家：

Q：当为 Android 编程时，有一种操作系统比其他的更好吗？

A：在使用过一些操作系统之后，我还没有注意到哪一种操作系统带有明显的优势。真的只是个人喜好。但是，经常发生的情况是，你可能会看到更多非官方的工具为了 Linux 平台而发布。因为 Linux 和 Android 是开放源码，更多的开放源码开发者倾向于为另一个开放源码平台创建工具。这个最终会给 Android 带来比 Linux 更多的好处。

Q：还有其它的命令可以从 adb 壳环境运行吗？

A：是的。例如，一个有趣的命令就是服务命令，可以被用来检查一个过程的状态，如：service check phone

假定手机正在运行，你应当能得到反馈：

Service phone: found

另外使用服务命令的方法是打一个电话。模拟器开启的情况下，输入命令并检查模拟器界面：

service call phone 2 s16 "15555551212"

题外话：终于完成第六章的翻译工作了。

第七章 使用 Intents 和电话拨号盘

使用 Intents 和电话拨号盘 第七章(1)

关键技能 & 概念

- 使用 Intents
- 创建和电话硬件交互的代码
- 学习拨号和呼叫的差异

本书到目前为止已经介绍了 Android 编程的基础知识。你已经仔细检查了 Android 应用程序的概要并且安装了你的第一个应用程序到 Android 服务器中。你已经学习了如何使用 Views 和 SetContentView()，同时知道如何在一个 XML 中创建 UI。这些技能已经帮助你创建一个静态的应用程序。你还没有做的就是使用应用程序接口来和这个平台的硬件——手机来产生交互。

你不应该忘记一个事实，那就是 Android 创建的平台仍然是一个手机。这个 Android 会运行的设备潜在的硬件，是设计为个人与个人通信目的的。如果你揭开 Android SDK 外在的浮华之物，它最基本的能力必须要能接或者打电话。

基于这个原因，本章重点放在与手机硬件交互的代码上，你应当有一些与手机基本功能交互的技能。你将能使用拨号盘来接受和打电话。这些工具和技能将会是你在这个灵活平台创建应用程序的关键所在。

你在读者本书是因为你想要设计运行在一个手机上的应用程序 ，所以，显而易见你应当学习 Android 如何允许和手机硬件交互——特别是，打出电话和接收电话的过程。

当我们想到手机，一些基本的功能会出现在我们的脑海里。首先，绝大多数情况，是能打出并接收电话。这是一个不容争辩的手机核心功能。还有一些非核心的特点使得手机易于使用，比如有能力保留并管理联系人，有能力储存没有接到的电话。通过阅读本章的内容，你会进入并熟练操作这些功能的代码。

本章中，你看到的一个手机功能就是打出一个电话。你会创建一个应用程序，使用一个 Intent，它将控制电话拨号盘并促使它呼叫一个号码。作为文章的进展，你将扩展这个应用程序并增加一些装饰到程序中。

注意

在 Android 平台，拨号和呼叫是不一样的。当你拨一个号码，你输入数字到键盘

（或者通过程序）。但是没有呼叫实际发生。这就是，拨号没有包括呼叫按钮。但你呼叫一个号码，你从手机上发送一个信号。那就是在输入号码到拨号盘以后，你按下呼叫按钮——物理上或者程序上。你需要知道两个动作的不同来理解你会在本章中创建程序的应用范围。

Intents 是什么？

Intents 是什么？ 第七章(2)

在你开始与拨号盘交互之前，需要你理解你要使用的代码类型。Andriod 使用 Intent 在应用程序中定义工作。一旦你掌握了 Intents 的使用，一个全新的应用程序开发世界将会向你敞开。本节定义了 Intent 是什么和如何使用它。

一个 Intent 是 Android 从一个 Activity(活动)传递信息到另外一个活动的方法。你可以认为一个 Intent 是一个活动间交换的信息。例如，假定你有一个活动需要来打开一个网页浏览器并且在 Android 设备上显示一个页面。你的活动应当发送一个“在网页浏览器中打开某页的 Intent(意图)”，就像一个 WEB_SEARCH_ACTION 的 Intent，一个 Android Intent 解答器。Intent 解答器从语法上分析一个活动的列表并且选择最匹配你的 Intent 的一个。那就是，网页浏览器的活动。Intent 解答器然后传递你的网页到浏览器中并且启动网页浏览器活动。

Intents 被分成两个主要目录

● Activity Action Intents（活动动作意图）Intents 用来呼叫应用程序以外的活动。只有一个活动可以处理 Intent。例如，对于网页浏览器，你需要打开网页浏览器活动来显示一个页面。

● Broadcast Intents （广播意图）Intents 被送出到多个活动来处理。一个被 Android 发出的广播意图的例子就是，当前电池的电量。任何活动处理这个意图并适时的反应。——例如，如果电池电量低到一定程度，取消一个活动。

表格 7-1 列出并且描述了通用的，可以使用活动动作意图。正如你注意到的一样，大多数情况下，从 Intent 名字可以看出这个 Intent 是做什么的。

Activity Action Intent	Message
ADD_SHORTCUT_ACTION	增加一个功能快捷菜单到 Android 的主屏
ALL_APPS_ACTION	列出设备上可用的所有应用程序
ANSWER_ACTION	接电话
BUG_REPORT_ACTION	打开调试报告活动
CALL_ACTION	呼叫一个提供的位置
DELETE_ACTION	删除定义的数据

DIAL_ACTION	打开拨号活动并且拨打一个定义好的号码
EDIT_ACTION	对有权使用的数据提供编辑
EMERGENCY_DIAL_ACTION	拨打一个紧急号码
FACTORY_TEST_ACTION	回复工厂测试设定
GET_CONTENT_ACTION	选择并返回定义的数据
INSERT_ACTION	插入一个空的条目
MAIN_ACTION	建立一个活动开始点
PICK_ACTION	挑选一个条目并且返回一个选择
PICK_ACTIVITY_ACTION	挑选一个特定的活动（返回一个类）
RUN_ACTION	执行特定的数据
SEARCH_ACTION	在系统上启动搜索
SEND_ACTION	发送数据给没有定义的接收者
SENDTO_ACTION	发送数据到指定的接收者
SETTINGS_ACTION	启动系统设定
SYNC_ACTION	和外部的源同步手机
VIEW_ACTION (DEFAULT_ACTION)	打开一个视图
WALLPAPER_SETTINGS_ACTION	显示修改 Android 墙纸的设定
WEB_SEARCH_ACTION	打开谷歌搜索，或者其它定义过的网页

注意

本章中的应用程序会用到列在表 7-1 中的 Intents：
CALL_ACTION 和 DIAL_ACTION。这些 Intents 使你有进入手机拨号和呼叫的能力。

表格 7-2 列出并描述了通用的广播意图。当你需要为一个定义的 Intent 建立一个接受器时，请参考这个表。

Broadcast Intent	信息
CALL_FORWARDING_STATE_CHANGED_ACTION	电话呼叫转接状态已经改变

CAMERA_BUTTON_ACTION	照相机的按钮被按下
CONFIGURATION_CHANGED_ACTION	设备配置发生改变
DATA_ACTIVITY_STATE_CHANGED_ACTION	设备的数据活动状态改变
DATA_CONNECTION_STATE_CHANGED_ACTION	数据连接状态改变
DATE_CHANGED_ACTION	手机系统数据改变
FOTA_CANCEL_ACTION	取消未决的系统更新下载
FOTA_INSTALL_ACTION	升级已经下载必须立即安装(由系统发送)
FOTA_READY_ACTION	升级已经下载可以延迟安装(由系统发送)
FOTA_RESTART_ACTION	重启一个系统升级下载
FOTA_UPDATE_ACTION	开始系统升级下载
GTALK_SERVICES_CONNECTED_ACTION	发送当 GTALK 已经成功建立
GTALK_SERVICES_DISCONNECTED_ACTION	发送当 GTALK 已经断开
MEDIA_BAD_REMOVAL_ACTION	发送当一个 SD 储存卡移开但是从系统中未成功移除
MEDIA_BUTTON_ACTION	发送当媒体按钮按下
MEDIA_EJECT_ACTION	发送当弹出动作为一个 SD 储存卡被初始化
MEDIA_MOUNTED_ACTION	发送当一个 SD 储存卡在系统中成功安装
MEDIA_REMOVED_ACTION	发送当检测到储存卡移出
MEDIA_SCANNER_FINISHED_ACTION	发送当扫描器完成
MEDIA_SHARED_STARTED_ACTION	发送当扫描器开始

MEDIA_UNMOUNTED_ACTION	发送当 SD 卡被检测到但是没有被安装
MESSAGE_WAITING_STATE_CHANGED	手机“信息等待”状态发生变化
NETWORK_TICKLE_RECEIVED_ACTION	一个新网络设备通知被接受
PACKAGE_ADDED_ACTION	当一个新的包装被安装在设备上发送
PACKAGE_CHANGE_ACTION	发送当现存的包装发生改变
PACKAGE_INSTALL_ACTION	一个包装可以被下载和安装
PACKAGE_REMOVED_ACTION	一个包装已经被移除
PHONE_INTERFACE_ADDED_ACTION	设备的手机界面已经被建立
PHONE_STATE_CHANGED_ACTION	设备的手机状态已经改变
PROVIDER_CHANGED_ACTION	设备从一个接收者处接收到通知
PROVISIONING_CHECK_ACTION	从供给服务中检测最新的设定
SCREEN_OFF_ACTION	屏幕被关闭（设备发送）
SCREEN_ON_ACTION	屏幕被打开（设备发送）
SERVICE_STATE_CHANGED_ACTION	服务状态被改变
SIGNAL_STRENGTH_CHANGED_ACTION	信号强度改变

注意

一些广播意图经常被发送，如 TIME_TICK_ACTION
和 SIGNAL_STRENGTH_CHANGED_ACTION。使用时请谨慎处理。你不应当试着去同时接受这样的广播。Intent 只是大约三分之一。其实 Intent 只是做了某些事情，而且它不能自己来做任何事。你需要 Intent 过滤器和 Intent 接受器来听，翻译 Intents.一个 Intent 接收器就像一个 Activity 的邮箱。Intent 接收器被用来允许一个活动来接受定义的 Intent。使用前一个网页浏览器的例子，网页浏览

器活动被设定来接受网页浏览器 Intent。一个像这样的系统允许不相关的活动来忽略不能处理的 Intent。它同时允许需要其它活动辅助的活动利用这个活动，而不需要知道如何呼叫它。

有了 Intents 和 Intents 接收器，一个活动可以发送一个 Intent 并且另外一个可以接受。不过，需要一些东西来管理两个活动之间的信息类型。这就是为什么要用 Intent 过滤器了。

Intent 过滤器被活动用来描述要接受的 Intent 类型。更重要的是，它们在 Intent 的内部概括了传递的数据类型。因此，在我们例子的方案中，我们要网页浏览器来打开网页。Intent 过滤器将会陈述数据使用 WEB_SEARCH_ACTION Intent 应当是 URL 格式的。

在下一节中，你将开始使用 Intent 来打开和利用电话的拨号盘。

使用拨号盘 第七章(3)

现在你知道 Intent 是什么了，是时候来看它如何运转的了。本节向你展示如何使用 DIAL_ACTION 这个 Intent 来打开电话的拨号盘。你将用你的 Intent 来传递一个电话号码。如果应用程序工作正常，你将会看到由 Intent 传递，而显示在拨号盘内的号码。

第一步是为这个活动创建一个项目（具体操作见第五章：Android 程序员向导目录）。把项目命名为 AndroidPhoneDialer。下面的插图就是这个项目的新 Android 项目向导（略）。

在 Eclipse 内打开的新的应用程序，第一个要做的就是从 main.xml 中移除包含 Hello World 声明的 TextView。在删除了 TextView 后，main.xml 文件应当看起来如下：

```
<?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"
>
</LinearLayout>
```

你需要增加两个新的包装到你的项目中来使用 DIAL_ACTION Intent，如下，第一个包装允许你设置 Intents 并且第二个允许你来分析 URIs。

```
import android.content.Intent;
import android.net.Uri;
```

注意

对于 DIAL_ACTION 这个 Intent 有一些不同的 Intent 过滤器可以使用。你正在使用的是允许你把号码作为了一个 URI 来传递的过滤器。

下一步就是来创建你的 Intent。创建一个 Intent 的语法如下：

```
Intent <intent_name> = new Intent(<Android_Intent>,<data>)
```

对于你的应用程序，把第一个参数<intent_name>用 DialIntent 替换掉。要获得第二个参数的数值，请参考 Activity Action 中的列表。(列表在文章中：什么是 Intent)。要呼叫拨号盘，你需要使用 DIAL_ACTION Intent。要正确的呼叫 Intent，使用 Intent.DIAL_ACTION 这个格式。最后的参数<data>，就是电话号码。DIAL_ACTION intent 把号码作为一个 URI。因此，你需要使用 Uri.parse 来分析出电话号码。使用 Uri.parse 将确保 DIAL_ACTION intent 能够理解你试图拨打的号码。你传递了一个 Uri.parse 的字符串来展示你要拨打的号码，在本例中是 "tel:5551212" 。

为你项目创建的最后一个呼叫应该像这样：

```
Intent DialIntent = new
Intent(Intent.DIAL_ACTION,Uri.parse("tel:5551212"));
```

提示

你使用记号 tel:<phone_number>来呼叫一个指定的电话号码。你还可以使用 voicemail 来替代 tel:呼出一个电话 voicemail 的快捷方式。

Intent 创建后，你现在必须告诉 Android 你想要拨号盘在新的活动中被启动。要这样做，你使用 setLaunchFlags()的 Intent 方法。你必须为启动来传递 setLaunchFlags()合适的参数。下面是可以设置接受启动旗帜的一组列表：

注意

在其它情况下，可能会有超过一个的旗帜被设置来完成希望的结果。

- NO_HISTORY_LAUNCH 启动活动，不记录在系统启动历史中
- SINGLE_TOP_LAUNCH 告诉系统不要启动活动，如果该活动已经在运行
- NEW_TASK_LAUNCH 启动活动
- MULTIPLE_TASK_LAUNCH 启动活动，即使它已经在运行了
- FORWARD_RESULT_LAUNCH 允许新的活动来接受结果，这个结果通常被转递给现存的活动。本例中，你要使用 intent.NEW_TASK_LAUNCH，这样可以简单的让你打开一个新的拨号盘活动示例：

```
DialIntent.setLaunchFlags(Intent.NEW_TASK_LAUNCH );
```

创建拨号盘的最后一步是启动活动。（更精确的说，你告诉 Android 你有一个作为新任务来启动的拨号盘。最终由 Android 来启动拨号盘活动）。要告诉 Android 你要启动拨号盘，你需要使用 startActivity()：

```
startActivity(DialIntent);
```

请注意到你把 intent 传递到 startActivity()。这个 Intent 然后传递到 Andriod，然后活动被执行。完整的 AndroidPhoneDialer.java 文件代码应当如下：

```
package android_programmers_guide.AndroidPhoneDialer;
```

```
import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.net.Uri;
public class AndroidPhoneDialer extends Activity {
/** Called when the Activity is first created. */
@Override
public void onCreate(Bundle icicle) {
super.onCreate(icicle);
setContentView(R.layout.main);
/** Create our Intent to call the Dialer */
/** Pass the Dialer the number 5551212 */
Intent DialIntent = new
Intent(Intent.DIAL_ACTION,Uri.parse("tel:5551212"));
/** Use NEW_TASK_LAUNCH to launch the Dialer
Activity */
DialIntent.setLaunchFlags(Intent.NEW_TASK_LAUNCH );
/** Finally start the Activity */
startActivity(DialIntent);
}
}
```

你现在应当来编译 AndroidPhoneDialer 并且在模拟器中运行它。处理编译和运行应用程序的过程在前面的章节中描述过了。你应当已经熟悉这些过程了。一旦你运行应用程序，模拟器启动。在漫长的启动过程后，你的活动被启动。

提示

保持模拟器运行是一个好主意，即使你完成了你的活动并且以及返回到代码窗口。大多数人的本能习惯是在他们完成了测试活动后关闭模拟器。但是，我发现使模拟器一直开启会帮助两个主要的问题。第一个就是启动模拟器要花费大量的时间。保持模拟器开启会避开漫长的开启时间。第二，我已经注意到有好几次当我做一些小的修改到一个活动，而且它们没有被复制到模拟器。保持模拟器开启似乎可以缓解这个问题。如果你在模拟器中有问题，在你的电脑中移除 userdata-qemu.img 文件。这个会让模拟器从一个干净的镜像启动。

如果你正确的跟从本例中的代码，你应当能看到下面的结果（略）：

如你所见，你已经打开了电话的拨号盘。这个拨号盘显示了你传递的号码，5551212。使用模拟器，点击呼叫按钮。现在电话应当虚拟的呼叫 555-1212。显示拨号盘是有用的，加入你创建了一个应用程序运行用户来在呼叫前可以编辑号码，或者确认他们真的想要呼叫这个号码。那么你应当怎么做来让应用程序为你打电话呢？答案就在下一节中。

从你的活动中打出电话 第七章(4)

在本节中你将会学到呼叫拨号盘时增加什么样的 Intent。你还会学到在活

动代码中的哪一个地方增加选择的 Intent。另外，你将学习如何分析一个作为 URI 的电话号码。从拨号盘活动代码变成呼叫活动你需要更改一些代码。在本节中，你回去编辑 AndroidPhoneDialer 活动，在打开拨号盘后，来打一个电话。

在活动中增加一个 Intent，你还是需要 Intent 和 Uri 包装，所以，在 AndroidPhoneDialer.java 的文件头部保留这一部分。

```
import android.content.Intent;
import android.net.Uri;
```

这些包装将确保你不仅需要 intent 而且同样会传递需要的电话号码数据到 Intent 中（用 Uri 包装）。

提示

如果你不按照顺序匆匆看完这个章节，而且没有运作前一节实际的项目，那么就简单的创建一个新的项目，命名为 AndroidPhoneDialer，然后增加前面提到的两个包装进去。这样会赶上进度。

现在看看在本章早些时候表格 7-1 中可以使用的 Activity Action Intents。你真正需要的是 CALL_ACTION。很多的时候 DIAL_ACTION 打开 Andriod 拨号盘，CALL_ACTION 将会启动电话的呼叫过程并且开始呼叫提供的号码。

要创建 Intent，使用和创建拨号盘同样的程序，只是这次使用 CALL_ACTION:

```
Intent CallIntent = new
Intent(Intent.CALL_ACTION,Uri.parse("tel:5551212"));
```

请注意你使用 Uri.parse 来传递一个正确的电话号码到活动中。下一步是告诉 Android 你要把这个活动设为启动，并且启动它。使用下面的两行代码来实现：

```
CallIntent.setLaunchFlags(Intent.NEW_TASK_LAUNCH );
startActivity(CallIntent);
```

在第一行，你发送启动旗帜到 NEW_TASK_LAUNCH。这个会启动一个呼叫的新示例。最后，你告诉 Android 使用你的 Intent 启动活动。当结束时，你的 AndroidPhoneDialer.java 文件应当如下：

```
package android_programmers_guide.AndroidPhoneDialer;

import android.app.Activity;
Chapter 7: Using Intents and the Phone Dialer 129
import android.content.Intent;
import android.os.Bundle;
import android.net.Uri;
public class AndroidPhoneDialer extends Activity {
/** Called when the Activity is first created. */
@Override
public void onCreate(Bundle icicle) {
super.onCreate(icicle);
```

```
setContentView(R.layout.main);
/** Create our Intent to call the device's Call
Activity */
/** Pass the Call the number 5551212 */
Intent CallIntent = new
Intent(Intent.CALL_ACTION,Uri.parse("tel:5551212"));
/** Use NEW_TASK_LAUNCH to launch the Call Activity
*/
CallIntent.setLaunchFlags(Intent.NEW_TASK_LAUNCH );
/** Finally start the Activity */
startActivity(CallIntent);
}
}
```

编译这个应用程序并且观察结果，你应当看到如下类似的错误信息。我实际上有意的要你看看这个错误，因为它展示了我们还没有发现的 Android 的另一面，错误的文本如下：

```
Application_Error:
...
Java.lang.SecurityException:
Permission Denial: starting Intent
...
```

Android 通过要求许可被执行来准许恰当的行动，在下一节叙述。

编辑活动许可

编辑活动许可 第七章(5)

大多数的Activity Action Intents是在需要许可在Android允许它行动之前的目录内的。和大多数的系统一样，Android 只是需要确保有资格的活动来执行在它们之外的活动。这儿是许可可以使用的活动：

● ACCESS_ASSISTED_GPS	● INTERNAL_SYSTEM_WINDOW
● ACCESS_CELL_ID	● RAISED_THREAD_PRIORITY
● ACCESS_GPS	● READ_CONTACTS
● ACCESS_LOCATION	● READ_FRAME_BUFFER
● ACCESS_SURFACE_FLINGER	● RECEIVE_BOOT_COMPLETED

● ADD_SYSTEM_SERVICE	● RECEIVE_SMS
● BROADCAST_PACKAGE_REMOVED	● RECEIVE_WAP_PUSH
● BROADCAST_STICKY	● RUN_INSTRUMENTATION
● CALL_PHONE	● SET_ACTIVITY_WATCHER
● CHANGE_COMPONENT_ENABLED_STATE	● SET_PREFERRED_APPLICATIONS
● DELETE_PACKAGES	● SIGNAL_PERSISTENT_PROCESSES
● DUMP	● SYSTEM_ALERT_WINDOW
● FOTA_UPDATE	● WRITE_CONTACTS
● GET_TASKS	● WRITE_SETTINGS
● INSTALL_PACKAGES	

把这个许可列表和表格 7-1 做比较你应当发现大多数的 Intent 可以匹配。CALL_ACTION 也不例外。你需要赋值 CALL_PHONE 活动许可来执行 Intent。

要赋值相关的许可到活动，第一，你需要知道需要赋值哪一种许可。当前的例子是使用拨号盘活动。进入拨号盘活动是由 CALL_PHONE 许可管理的。通过赋值这个许可到你的活动，Android 将允许你的 Intent 启动拨号盘活动。

怎么增加许可到活动中呢？你需要编辑活动的 Manifest。如果你使用 Eclipse，双击 AndroidManifest.xml 文件，打开 Android Manifest 窗口，如下图（略）。

要编辑活动的许可，点击 Permission 链接。会把你带到 Manifest Permissions 窗口，如下图（略）。这个窗口列出了当前赋值到你活动的许可。假定你在一个新的项目中，还没有任何的赋值。因此，点击增加按钮来开始进程。在对话框中，选择使用许可并且点击 OK。

回到 Android Manifest Permission 窗口，在名称的下拉框中，选择 android.permission.CALL_PHONE，如下所示（略）。这样就会增加 CALL_PHONE 许可到你的活动中。现在，你已经增加了 CALL_PHONE 许可，看看 AndroidManifest.xml 文件。它应当和下面相类似：

```
<?xml version="1.0" encoding="utf-8"?>
<manifest
xmlns:android=http://schemas.android.com/apk/res/android
package="android_programmers_guide.AndroidPhoneDialer">
<application android:icon="@drawable/icon">
```

```
<activity android:name=".AndroidPhoneDialer"
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>
<uses-permission
android:name="android.permission.CALL_PHONE">
</uses-permission></manifest>
```

最有意思的一行实在文件的最后：

```
<uses-permission
android:name="android.permission.CALL_PHONE">
</uses-permission>
```

这行代码是由 Androd plugin for Eclipse 增加的。如果你需要，你可以直接编辑 AndroidManifefst.xml 文件来赋值。但是，如果有多次情况当你不确定需要增加哪一种许可，或者什么语法来增加，你可以使用 Manifest 的向导。

现在许可已经到位了，重新编译并且允许你的活动。你的模拟器应当可以呼叫电话号码了，如下图（略）。

你创建的活动已经使用了一个 Intent 来启动设备的呼叫活动并且呼叫号码 555-1212。这个演示了使用 Intent 的好处。总而言之，这个应用程序实际的为你做了一些事情。那就是说，启动一个带有电话号码代码的活动，只是打一个电话？在下一节中，你会通过增加一个按钮来启动 Call_Action 的 Intent，增加一个文本框来运行用户输入他们选择的电话号码来更多的制作应用程序。

修改 AndroidPhoneDialer

修改 AndroidPhoneDialer 第七章(6)

本节展示如何通过修改 AndroidPhoneDialer 来增加一些特性，使得它更加的具有实际价值。到本节结束时，你不仅仅对使用 intent 的得心应手，而且还会使用 EditTexts 和 Buttons。

警告

如果你没有跟从上一节的项目，回去并创建那个活动。本节的教程假定你已经完成了从上个项目的可以自行支配的编码工作。

增加一个按钮

本节向你展示如何修改你的项目来包含一个按钮。当活动被启动，替代启动 Intent 的将会是一个按钮。除了文本，按钮是应用程序里最常用的对象。按钮

组织用户和程序之间的交互作用。在 Android 里学会如何创建并利用按钮是必要的，如果要创建一个友好的活动。

你将要在 main.xml 里创建按钮了。回想一下第五章，你为 Hello World!活动创建了 TextView。TextView 有一个清晰的结构，就像这样：

注意

记住，当你在 main.xml 里创建一个 View 时，你只是告诉 Android，你想让这个 View 看上去是什么样的。你仍旧需要在 AndroidPhoneDialer.java 内把功能赋值给它。

```
<View android:id=<id>
android:layout_width=<width>
android:layout_height=<height>
>
```

这个格式对于所有的 views 都有用，并且 Button 也不例外。你需要为你的 Button 设定的 XML 属性是 android:id， android:layout_width，android:layout_height，和 android:text。这 4 个 XML 属性充分的描述了按钮，那样你就可以在活动中使用它了。

1、赋值你的按钮的 ID 到 callButton：

```
android:id="@+id/callButton"
```

2、独自设置 layout_width 和 layout_height 到 fill_parent 和 wrap_content。

```
android:layout_width="fill_parent"
android:layout_height="wrap_content"
```

3、设置按钮的文本为“Show Dialer”，这个是一个清晰的描述来告知按钮的作用：

```
android:text="Show Dialer"
The full XML for the Button, with attributes,
looks like this:
<Button android:id="@+id/callButton"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:text="Show Dialer" />
```

现在看看完成后的 main.xml 文件。按钮在上下文中显示并且等待着你的编码。

```
<?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"
>
<Button android:id="@+id/callButton"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:text="Show Dialer" />
</LinearLayout>
```

要开始为按钮增加功能，你需要增加其它的包装到 AndroidPhoneDialer.java 这个文件。包含按钮 View 的是 android.widget.Button;

现在你已经在项目里输入按钮窗口小部件了（Button widget）。这些给了你必要的信息来开始你项目的编码工作。你项目的最终编码结果应当是有一个按钮在活动中，当点击它，启动呼叫活动。呼叫活动应当和数据“tel:5551212”一起启动。屏幕显示结果应该和初始的 AndroidPhoneDialer 相匹配。描述的功能围绕一些不同的概念。首先，你必须编程一个按钮，按钮的属性在 main.xml 文件内建立。下一步，你必须创建一个功能来启动前一个编码项目的 CALL_ACTION 这个 Intent。最后,你的按钮应该能执行这个功能并且启动 Intent。

创建按钮的语法是

```
final Button <button_name> = <button>
```

等号左边的部分在代码中创建按钮。右边是从 main.xml 文件中调用按钮的属性。要调用它的属性，你使用 findViewByID()把结果投递为一个按钮。这个听起来比它实际情况复杂。

记住，当你增加按钮的属性到 main.xml 文件中，你给了这个按钮一个定义好的 android:id，callButton，而这个通过 Android plugin for Eclipse 在 id 文件中以 R.id.callbutton 注册。使用 findViewById（）来从 main.xml 文件中通过传递 id callButton 来找回：

```
findViewById(R.id.callButton)
```

别忘了投递为按钮

```
(Button) findViewById(R.id.callButton)
```

这个声明构成了等号右边的内容。创建完整的按钮应该是这样的：

```
final Button callButton = (Button)
findViewById(R.id.callButton);
```

现在你有一个可以工作的按钮了，但是你需要它来做点什么。按钮本身在没有代码的情况下无法完成太多的工作。根据本例的目的，你需要让它来执行 CALL_ACTION 这个 Intent。因此，你需要在你现有的 Intent 呼叫中创建一个小的功能。这样，当按钮被按下时，你可以呼叫了。

如果你熟悉 Java 编程，这里就没什么好奇怪的了。你将要设置 onClick() 方法来从前一个部分呼叫 Intent 代码。onClick（）方法拿去一个 View 作为一个变量；但是，本例中，并没有在 onClick()方法本身内呼叫任何的 View：

```
public void onClick(View v){
Intent callIntent = new
Intent(Intent.CALL_ACTION,Uri.parse("tel:5551212"));
callIntent.setLaunchFlags(Intent.NEW_TASK_LAUNCH );
startActivity(callIntent);
}
```

代码左边是程序中仅有的绑定按钮到 onClick 的接受器。接收器对于 Java 编程者是非常熟悉的。对于不熟悉 Java 或者接收器的人来说。接收器是 Java 对象可以从其它对象“接受”的方法。同样的概念也适用于 Android。你可以在 Android 中建立接收器，来允许 Android 的 Views 从其它的输入中处理呼叫。对于本项目，你需要为你的按钮创建一个接收器来接受活动中的按钮的 onclick 事件。当用户按下按钮，接收器，接收器会呼叫在 onClick()方式中的代码。要建立接收器，你需要使用按钮的 setOnClickListener()方法。

如果你熟悉 Java 开发，这个结构不应当看起来陌生。这个是 Java 中典型的 onClickListener 接口执行。你在这里将会看到的是使用一个 Java 匿名的类来为按钮执行 onClickListener。还有，作为一个匿名类，你可以用作本地变量来使用-本例中是按钮，如果变量被定义为最终的。

setOnClickListener( ) 方法抓取一对变量。第一个是 onClickListener() 的实例。第二个是早前建立的 onClick。你的 setOnClickListener()应当看上去想这样：

```
callButton.setOnClickListener(new
Button.OnClickListener() {
public void onClick(View v){
Intent callIntent = new
Intent(Intent.CALL_ACTION,Uri.parse("tel:5551212"));
callIntent.setLaunchFlags(Intent.NEW_TASK_LAUNCH );
startActivity(callIntent);
}
});
```

这部分代码陈述了当 callButton 被按下，onClickListerner 将执行 onClick 中的代码。onClick 中的代码将执行 CALL_ACTION Intent 并且呼叫电话号码 555-1212。

你完成的 AndroidPhoneDialer.java 看起来像这样：

```
package android_programmers_guide.AndroidPhoneDialer;

import android.app.Activity;
import android.os.Bundle;
```

```
import android.widget.Button;
import android.view.View;
import android.content.Intent;
import android.net.Uri;
public class AndroidPhoneDialer extends Activity {
/** Called when the activity is first created. */
@Override
public void onCreate(Bundle icicle) {
super.onCreate(icicle);
setContentView(R.layout.main );
/** Create the Button */
final Button callButton = (Button)
findViewById(R.id.callButton);
/** Set the onClickListener to call the onClick */
callButton.setOnClickListener(new
Button.OnClickListener() {
/** Use the onClick to call the existing Intent code
*/
public void onClick(View v){
Intent callIntent = new
Intent(Intent.CALL_ACTION,Uri.parse("tel:5551212"));
callIntent.setLaunchFlags(Intent.NEW_TASK_LAUNCH );
startActivity(callIntent);
}
});
}
}
```

在模拟器中编译并运行这个活动。主要的活动将显示标签为 Show Dialer 的按钮。点击这个按钮，它应当打开 Call 活动并拨号 555-1212。主要活动应该如下图（略）。

如你所见，Android 是一个非常健全并灵活的平台。有几行相关的代码，比一页少，你已经创建了一个利用设备电话硬件的活动并且由按钮启动。同样，就这一点来说，你应当对 Android 处理活动，Intent 和 Views 的方式刚到舒适。

你的 AndroidPhoneDialer 活动仍然不是很健全。你需要增加一个更多的条目。本章的最后一节展示如何使用 EditText View 来运行用户输入一个电话号码。这个号码然后会被传递到 CALL_ACTION Intent(而不是代码写好的数值：tel:5551212)。

执行一个 EditText View 第七章(7)

你需要增加一个 View 到活动中来使得用户输入一些文本。然后你会分析那个文本并把它发送到前一节的 Intent 呼叫中。因为所有的视图是从基本的视图中

派生出来的，它们在结构和使用方面非常的相似。你会发现执行一个 EditText 是一个非常简单的操作。

首先，在 main.xml 文件中放置 Views。实际上这里要放两个 View：一个 TextView 来实现作为一个标签并且给出一些指示给用户，另外一个就是 EditView 来接收用户的输入。这个 Views 一起将增加深度和实用性到你的活动中。

因为你组成活动的外观，记住.xml 是在视觉上构成的。这个就意味着加入你要 TextView 在最后的活动中显示在 EditText 的上面，你应当在 main.xml 文件中把它放在 EditText 之前。

因为你已经用过好几次 TextViews 了，所以这里不会讲的太多。简单的看一些你设置的 TextView 的属性：

```
<TextView android:id="@+id/textLabel"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:text="Enter Number to Dial:"
/>
```

没什么特别的地方。这只是个简单的 TextView 用文本输入号码来拨号：。这个 TextView 将会用做 EditView 的显示标签。这里是你为 EditView 设置的属性:

```
<EditText android:id="@+id/phoneNumber"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
/>
```

注意

你没有必要一定去设置 androd:text 属性，因为你不需要任何缺省的文本。

这个 id 被用来设为 phoneNumber，这是个名字，你将要用来在代码中参阅到 EditText。再说一次，当设置 main.xml 时，没什么特别之处。最后的文件应当看起来如下：

```
<?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"
142 Android: A Programmer’ s Guide
Chapter 7: Using Intents and the Phone Dialer 143
android:layout_height="fill_parent"
>
<TextView android:id="@+id/textLabel"
android:layout_width="fill_parent"
```

```
android:layout_height="wrap_content"
android:text="Enter Number to Dial:"
/>
<EditText android:id="@+id/phoneNumber"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
/>
<Button android:id="@+id/callButton"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:layout_alignParentRight="true"
android:text="Show Dialer" />
</LinearLayout>
```

main.xml 现在完成了。你可以转移到 AndroidPhoneDialer.java 来继续工作。假如你不在使用一个现存的 AndroidPhoneDialer.java 文件——本章中的前一个项目——你可能需要参阅前一部分去看看增加到.java 文件中是什么样的代码。这样会确保你从代码中的正确部分开始。

在.java 文件中你第一个增加的条目是包装定义。你不仅需要增加包装到 Uri，按钮和 Intent，同时还要到 EditText：

```
import android.widget.Button;
import android.content.Intent;
import android.net.Uri;
import android.widget.EditText;
```

设置 EditText View 的语法和设置按钮的语法一致：

```
final EditText <edittext_name> = <edittext>
```

再说一次，呼叫你的 EditText phoneNumber。创建 EditText 的代码如下：

```
final EditText phoneNumber = (EditText)
findViewById(R.id.phoneNumber);
```

一旦你的 phoneNumber 这个 EditTexT 创建好了，你可以使用它来参阅在设备上输入的文本。现在你要做的就是呼叫 phoneNumber.getText()来找回用户的输入。在下面的行里替换代码数值“tel:5551212”

```
Intent(Intent.CALL_ACTION,Uri.parse("tel:5551212"));
with the value of getText( ):
Intent(Intent.CALL_ACTION,Uri.parse("tel:"          +
phoneNumber.getText()));
```

这就是本项目所有你需要更新的新代码。有了这两个新的附加内容，你可以给用户一个可以输入电话号码的对象，并且把号码发送到电话的呼叫活动中。完

整的.java 文件中的代码如下：

```
package
android_programmers_guide.AndroidPhoneDialer;
import android.app.Activity;
import android.os.Bundle;
import android.widget.Button;
import android.view.View;
import android.content.Intent;
import android.net.Uri;
import android.widget.EditText;
public class AndroidPhoneDialer extends Activity {
/** Called when the activity is first created. */
@Override
public void onCreate(Bundle icicle) {
super.onCreate(icicle);
setContentView(R.layout.main );
final EditText phoneNumber = (EditText)
findViewById(R.id.phoneNumber
);
final Button callButton = (Button)
findViewById(R.id.callButton);
callButton.setOnClickListener(new
Button.OnClickListener() {
public void onClick(View v){
Intent CallIntent = new
Intent(Intent.CALL_ACTION,Uri.parse("tel:" +
phoneNumber.getText()));
CallIntent.setLaunchFlags(Intent.NEW_TASK_LAUNCH );
startActivity(CallIntent);
}
});
}
}
```

当你在模拟器中运行应用程序，你应当看到一个类似于下面插图的屏幕（略）。

试试这个：修改 AndoridPhoneDialer 项目

试试这个：修改 AndoridPhoneDialer 项目_第七章(8)

如果你一直在搞最后版本的 AndroidPhoneDialer，你或许发现有些东西错过了。不幸的是，项目当前的写作方式总是允许你输入任何类型的数值到

EditText View 中，并且发送到 Call 活动中。这个真不是最佳的应用程序开发。研究一下并且增加验证到 EditText 中。使用下面的参数来修改项目：

● 使用常规的表达式来验证一个号码被输入到 EditText 中(package java.regex)。
● 使用 showAlert( ) 语法来显示一条信息告诉用户他们输入的内容和你的常规表达式不匹配。当你觉得已经找到解决方案，和下面的代码做个比较：

main.xml

```
<?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:id="@+id/textLabel"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:text="Enter Number to Dial:"
/>
<EditText android:id="@+id/phoneNumber"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
/>
<Button android:id="@+id/callButton"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:layout_alignParentRight="true"
android:text="Show Dialer" />
</LinearLayout>
AndroidPhoneDialer.java
package android_programmers_guide.AndroidPhoneDialer;
import android.app.Activity;
import android.os.Bundle;
import android.widget.Button;
import android.view.View;
import android.content.Intent;
import android.net.Uri;
import android.widget.EditText;
import java.util.regex.*;
public class AndroidPhoneDialer extends Activity {
/** Called when the activity is first created. */
@Override
public void onCreate(Bundle icicle) {
```

```
super.onCreate(icicle);
setContentView(R.layout.main );
final EditText phoneNumber = (EditText)
findViewById(R.id.phoneNumber );
final Button callButton = (Button)
findViewById(R.id.callButton);
callButton.setOnClickListener(new
Button.OnClickListener() {
Chapter 7: Using Intents and the Phone Dialer 147
public void onClick(View v){
if
(validatePhoneNumber(phoneNumber.getText().toString())){
Intent CallIntent = new
Intent(Intent.CALL_ACTION,Uri.parse("tel:" +
phoneNumber.getText()));
CallIntent.setLaunchFlags(Intent.NEW_TASK_LAUNCH );
startActivity(CallIntent);
}
else{
showAlert("Please enter a phone number in the X-XXX-XXX-
XXXX
format.",0, "Format Error", "Re-enter Number",false);
}
}
});
}
public boolean validatePhoneNumber(String number){
Pattern phoneNumber =
Pattern.compile("(\\d-)?(\\d{3}-)?\\d{3}
\\d{4}");
Matcher matcher = phoneNumber.matcher(number);
return matcher.matches();
}
}
```

当你运行本项目，它应当产生一个信息和下图类似（略）。

在下一章中，你将学习更多的 Views。你将创建创建一个多活动应用程序来允许你浏览并创建本书中还没有谈论过的 Views。你也会创建并利用一个菜单系统来启动你的活动。

问专家

Q：有办法来建立一个呼叫或者从模拟器中来确保这些活动正在工作吗？

A：当本书写的时候，还没有办法。但是，谷歌内部的讨论是在将来发布的 SDK，开发者将有可能打开两个模拟器并且在两个模拟器之间呼叫。

Q：还有其它类型的按钮可用于活动吗？看上去或者感觉不一样的。

A：是的。你可以使用风格属性来创建小按钮，或者上下左右指示的小按钮。

题外话：到现在为止，第七章的翻译工作完成，下面是第八章。时间已过去一个半月。

第八章 列表，菜单和其它 Views

列表，菜单和其它 Views 第八章(1)

关键技能 & 概念

- 建造活动
- 使用 Android 菜单
- 使用 AutoComplete TextView

本章会讨论更深层次的 Views 和 Intents，正如被证明这些是作为一个 Android 新手最好要掌握的特性。这个两个实体将组成早期活动的主体。几乎每一个你创建的活动需要至少一个 View，并且其中的大多数将需要呼叫一个到两个 Intent。学习这些东西的最好办法就是动手实践。阅读这些话题并且回顾属性清单是另外一个需要做的事情，但是自己执行代码却是另外一件事。那就是如你在上一章所作，你将建造一个使用 Views 和 Intents 的活动。通过构造这个应用程序，你将获得对 Views 和 Intents 最好的体验。

前两章通过创建非常简单的，开发 Views 和 Intents 的少量基本功能的活动来大致介绍了 Views 和 Intents。在本章中，你会去建造一个稍微复杂一点的使用 Intent 来呼叫一个新活动的活动，而这个新活动你也要创建。这个新活动会展示在本版本 SDK 中可用的 Views。

本章解释这些 View 的功能性，如 AutoComplete 清单和图表种类，并介绍每个 View 属性的不同。作为开始，创建一个新的 Eclipse 项目并命名为 AdnrodiViews。创建有下列插图（略）参数的项目：包装的名称为 android_programmers_guide.AndroidViews，活动的名称为：AndroidViews，并且应用程序的名称为 AndroidViews。

项目创建好以后打开 main.xml 文件，从里面把 Hello World!代码移除。有了一个创建的项目和干净的 main.xml 文件，你可以开始来增加你的代码了。

建造活动

到目前为止，你仅仅创建过单活动应用程序。那就是说，你创建的非常简单的应用程序，只包含了一个屏幕的数据。稍等片刻，想一下你使用的最后几个应用程序。偶尔它们使用了超过一个“窗口”。大多数应用程序使用多重窗口来采集，显示并保存数据。你的 Android 应用程序应当是一样的。

虽然你还没有学习如何创建在 Android 上运行多重活动的应用程序，但是在最后几章中你得到了如何改变多重活动的提示。你使用了一个新的概念——Intents 来呼叫并且运行一个核心的 Android 活动。这个概念在本章中仍旧适用，但是，当你需要来呼叫创建的活动，与呼叫核心的 Android 活动相反，它

执行起来是不同的。

你一件你要做的事情是构造活动。然后你可以创建呼叫它们的 Intents。构造活动时，需要按照下列步骤进行。

- 为.xml 文件准备 Intent 代码
- 为.java 文件准备 Intent 代码
- 使用一个 Intent 呼叫活动

一旦你创建你的第一个附加活动，其它的应当非常的简单。

注意

这些步骤并不是必须的。你可以已任何的次序来执行它们。

NOTE

为.xml 文件准备 Intent 代码

记住 Android 活动包含三个主要部分：包含代码的.java 文件，控制输出的.xml 文件和包装的 Manifest。到本书的这里，你只用到 main.xml 来控制单活动的输出。但是，要更好的使用多活动，你必须有 multiple.xml 输出文件。

要创建一个新的.xml 文件，打开你的 Eclipse 项目并且找到包装资源管理器。打开 res 目录，右击 layout 文件夹，并且选择新建文件。在新建文件对话框，如下所示（略），命名为 test.xml。

警告

确保以小写字母输入 test.xml。新的.xml 文件名必须是小写字母。

输出文件创建后。要使活动正常的工作，增加下列代码到 test.xml 文件中。这个代码会为你的输出提供一个基础。如果你需要，你可以简单的复制在 main.xml 文件现有的代码。

```
<?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"
>
</LinearLayout>
```

为.java 文件准备 Intent 代码

再次使用包装资源浏览器，找到 src 目录，打开它，并在 android_programmers_guide.AndroidViews 这个包装上右击，如下所示（略）。

再一次，你将在这文件夹中增加新文件。在你右击 AndroidViews 包装后，从上下文菜单中选择新建文件。这个文件会在本项目中为第二个活动控制所有的代码。把文件命名为 test.java。你现在有了一个非常好的，新的（但是是空的）.java 文件，只需要在其中增加代码来使它工作了：

```
package testPackage.test;
```

```
import android.app.Activity;
import android.os.Bundle;
public class test extends Activity {
/** Called when the Activity is first created. */
@Override
public void onCreate(Bundle icicle) {
super.onCreate(icicle);
setContentView(R.layout.test);
/** This is our Test Activity
All code goes below */
}
}
```

注意，使用上下文 R.layout.test，你以 setContentView 方法呼叫 test.xml，。这行告诉新的活动为本“页”使用你创建的.xml 文件作为输出文件。

修改 AndroidManifest.xml 文件

修改 AndroidManifest.xml 文件 第八章(2)

在 Eclipse 内打开你的 Androidmanifest.xml 文件。在本书中还没有大量的讨论这个 Androidmanifest.xml 文件呢。Androidmanifest.xml 文件包含项目的全局设置。更重要的是，Androidmanifest.xml 还为项目包含了 Intent 过滤器。

第七章讨论了 Android 如何使用过滤器来排列哪种 Intent 可以被哪种活动所接受。使这个过程方便的信息就保留在 Androidmanifest.xml 中了。

注意

每个项目只能有一个 Androidmanifest.xml 文件。

如果你的 Androidmanifest.xml 文件是打开的，它应当如下显示：

```
<activity android:name=".AndroidViews"
android:label="@string/app_name">
<intent-filter>
<action android:name="android.intent.action.MAIN"
/>
<category
android:name="android.intent.category.LAUNCHER"
/>
</intent-filter>
</activity>
```

你在这里要看的是AndroidView活动——项目创建的主要活动的 Intent 过滤器。

对于这个文件，你可以增加任何其它的 Intent 过滤器来交给项目处理。本例中，你要增加处理你创建的 Test 活动的过滤器。下面是你需要为 Intent 过滤器增加的代码到 Androidmanifest.xml 文件中。

```
<activity                    android:name=".Test"
android:label="Test Activity">
<intent-filter>
<action  android:name="android.intent.action.MAIN"
/>
<category
android:name="android.intent.category.LAUNCHER"
/>
</intent-filter>
</activity>
```

增加代码到 AndroidManifest.xml 文件中确保 Android 为 Test 活动传递 Intent 到正确的地方。完整的 AndroidManifest.xml 文件应当如下：

```
<?xml version="1.0" encoding="utf-8"?>
<manifest
xmlns:android=http://schemas.android.com/apk/res/android

package="android_programmers_guide.AndroidViews">
<application android:icon="@drawable/icon">
<activity                    android:name=".AndroidViews"
android:label="@string/app_name">
<intent-filter>
<action android:name="android.intent.action.MAIN" />
<category
android:name="android.intent.category.LAUNCHER" />
</intent-filter>
</activity>
<activity                    android:name=".AutoComplete"
android:label="AutoComplete">
<intent-filter>
<action android:name="android.intent.action.MAIN" />
<category
android:name="android.intent.category.LAUNCHER"
/>
</intent-filter>
</activity>
</application>
</manifest>
```

现在你的活动可以为 Test 活动处理 Intent 呼叫了。要让你的 Intent 呼叫

Test 活动，你将要使用和在第七章呼叫电话拨号盘非常类似的结构。下面的代码会设置你的 Intent：

注意

当你启动应用程序，将要打开的活动是你创建项目的 AndroidViews 活动。因此，放置下面的代码在 AndroidViews.java 中来启动 Test 活动。

```
Intent testActivity = new Intent(this,
test.class);
```

这一行创建一个叫做 testActivity 的 Intent。参数 test.class 告诉呼叫，你要 testActivity 这个 Intent 来展示创建的和本活动相关联的 Test 活动。

警告

当你使用 Intents 时，不要忘记输入 android.content.intent 包装。

最后，使用 startActivity()方法来精确启动 Test 活动：

```
startActivity(autocomplete);
Your completed AndroidViews.java file should look
like this:
package android_programmers_guide.AndroidViews;
import android.app.Activity;
import android.os.Bundle;
import android.view.Menu;
import android.content.Intent;
public class AndroidViews extends Activity {
/** Called when the Activity is first created. /
@Override
public void onCreate(Bundle icicle) {
super.onCreate(icicle);
setContentView(R.layout.main);
/**Set up our Intent /
```

在模拟器中运行这个应用程序。Android 应当启动 AndroidViews 活动，紧跟着 Test 活动。

在下一节中，你将使用这些技巧来创建一个启动多重活动的应用程序。每个活动将在一个 View 里，这样你可以应用不同的选项。这个将会给你大量的练习显示并熟练掌握 Views 和使用活动。

注意

要使用本章剩下的例子，移除本节创建的 Test 活动。你要继续做没有 Test 活动的 AndroidViews 项目的作品。

使用菜单

使用菜单 第八章(3)

在本节中，你将建造一个应用程序来允许用户从一些不同的 Views 中进行选择。当用户选择一个 View，一个新的活动将被启动。你将要使用给用户选择的工具就是 Android 菜单。看一下这个插图（略）。当用户激活菜单按钮，菜单就会显示。

如你所见，从 Android 主屏幕选择菜单按钮产生一个墙纸的设定选项。你将为你的主活动创建一个类似的菜单，该菜单为 View 保留所有用户可以从中选择的选项。现在，AndroidViews.java 文件的编码应当如下：

```
package android_programmers_guide.AndroidViews;
import android.app.Activity;
import android.os.Bundle;
public class AndroidViews extends Activity {
/** Called when the Activity is first created. */
@Override
public void onCreate(Bundle icicle) {
super.onCreate(icicle);
setContentView(R.layout.main);
}
}
```

你增加任何东西到活动中，你需要输入包装一个包装来创建你的菜单。输入 android.view.Menu 到 AndroidViews 活动中：

```
Import android.view.Menu;
```

要创建菜单，你需要优先活动的 onCreateOptionMenu()方法。onCreateOptionMenu()方法是一个以布尔方式被呼叫的当用户第一次选择菜单按钮。你将使用这个方式来建造菜单并增加可选择的条目到其中。增加下面的代码到 AndroidViews.java：

```
@Override
public boolean onCreateOptionsMenu(Menu menu) {
super.onCreateOptionsMenu(menu);
}
```

你将增加代码在 onCreaterOptionMenu()内增加菜单。需要增加的条目是你将要在本项目中创建的 Views。下面是将要增加到菜单的清单：

- AutoComplete 自动完成
- Button 按钮
- CheckBox 选择框
- EditText
- RadioGroup 按钮组
- Spinner

在前面创建的代码优先 onCreateOptionsMenu()方法，你在菜单中传递一个菜单变量呼叫菜单。这个变量代表实际的在 Android 界面中创建的菜单条目。要

增加这些条目到菜单中，你要使用 menu.add（）方法。这个呼叫的语句是：

```
menu.add(<group>,<id>,<title>)
```

参数组被用来与菜单条目相关联。在本例中，你将不使用组。但是，这个值是非常重要的。参数 id 被用来检测哪一个菜单条目被选择。最后，参数标题是显示在菜单上的文本。

增加下列代码到 onCreateOptionsMenu( )方法中：

```
menu.add(0, 0, "AutoComplete");
menu.add(0, 1, "Button");
menu.add(0, 2, "CheckBox");
menu.add(0, 3, "EditText");
menu.add(0, 4, "RadioGroup");
menu.add(0, 5, "Spinner");
```

你完整的 AndroidViews.java 文件应当看上去像这样：

```
package android_programmers_guide.AndroidViews;
import android.app.Activity;
import android.os.Bundle;
import android.view.Menu;
public class AndroidViews extends Activity {
/** Called when the Activity is first created. */
@Override
public void onCreate(Bundle icicle) {
super.onCreate(icicle);
setContentView(R.layout.main);
}
@Override
public boolean onCreateOptionsMenu(Menu menu) {
super.onCreateOptionsMenu(menu);
menu.add(0, 0, "AutoComplete");
menu.add(0, 1, "Button");
menu.add(0, 2, "CheckBox");
menu.add(0, 3, "EditText");
menu.add(0, 4, "RadioGroup");
menu.add(0, 5, "Spinner");
return true;
}
}
```

如果你执行如上面代码所写，你应当看见如下图所示的菜单（略）。

这个就是你想要达成的。试着去点击菜单中的任意一个选项。当用户选择一个菜单项目时，你在处理世间的活动中还什么都没有呢。

你增加的方法优先处理到菜单的呼叫是 onOptionsItemSelected( )。再一

次，像 onCreateOptionsMenu()，当菜单项被选择，onOptionsItemSelected( ) 是你需要优先一个拥有代码，被执行的布尔方法。优先代码应当如下：

```
@Override
public boolean onOptionsItemSelected(Menu.Item
item){
}
```

这个代码有个问题：当任何菜单项被选择，onOptionsItemSelected( ) 是个常规被呼叫的方法。你需要给定 onOptionsItemSelected( )一条路来识别不同菜单项之间的差别并执行相应的代码。因此，使用一个 switch/case 声明来帮助方法从不同的项目中选择。当你创建了菜单项，你定义了一系列的从 0 到 5 的数字作为菜单项的值。你可以在 case 声明中使用一个呼叫来 getI （）来检测哪一个菜单项被选择：

```
switch (item.getId()) {
case 0:
return true;
case 1:
return true;
case 2:
return true;
case 3:
return true;
case 4:
return true;
case 5:
return true;
}
return true;
```

在这个 case 声明中，对于每一个 id 当前设定的动作是返回 true.这个不会做任何的事情但是会保留一个开放的可以增加代码的区域。你的 AndroidViews.java 文件现在可以被用来创建被新菜单系统启动的活动了。完整的 AndroidViews.java 文件代码应当看上去如下：

```
package android_programmers_guide.AndroidViews;
import android.app.Activity;
import android.os.Bundle;
import android.view.Menu;
public class AndroidViews extends Activity {
/** Called when the Activity is first created. /
@Override
public void onCreate(Bundle icicle) {
super.onCreate(icicle);
setContentView(R.layout.main);
```

```
}
@Override
public boolean onCreateOptionsMenu(Menu menu) {
super.onCreateOptionsMenu(menu);
/** Add one menu item for each View in our
project */
menu.add(0, 0, "AutoComplete");
menu.add(0, 1, "Button");
menu.add(0, 2, "CheckBox");
menu.add(0, 3, "EditText");
menu.add(0, 4, "RadioGroup");
menu.add(0, 5, "Spinner");
return true;
}
/** Override onOptionsItemSelected to execute
code for each
menu item */
@Override
public boolean onOptionsItemSelected(Menu.Item
item){
}
/** Select statement to handle calls
to specific menu items */
switch (item.getId()) {
case 0:
return true;
case 1:
return true;
case 2:
return true;
case 3:
return true;
case 4:
return true;
case 5:
return true;
}
return true;
}
}
```

完成了 AndroidViews.java，你可以重点去创建其它的活动了，在下一节中，你将在项目中为每一个 View 创建一个活动并增加代码来启动 case 声明中 view 的活动。

为 AutoComplete 创建一个活动

为 AutoComplete 创建一个活动 第八章(4)

在本节中，你将创建一个突出 AutoCompleteTextView 的活动。AutoCompleteTextView 对你的有应用程序来说可以成为一个非常有力的工具。特别是对于 Android 主屏幕有限的空间来说。

AutoCompleteTextView，正如这个名字所说，是修改后的 TextView，而它可以参考到可用的单词或者短语并自动完成输入。这样的 Views 是在移动应用程序里是非常有用的当你不想花费大量的空间到一个 ListView，或者你想要加速你输入文本的过程。

要开始为 AutoCompleteTextView 创建活动，你需要为布局增加一个新的.xml 文件，为代码增加一个.java 文件，并且一个 Intent 过滤器来处理呼叫。

提示

创建这些条目的过程出现在本章“构造活动”一节中。创建下面项目的部分时可以参考那个部分。

创建一个 autocomplete.xml 文件

在你的 AndroidViews 项目中创建一个新的.xml 文件，并命名为 autocomplete.xml。记住文件名必须用小写。这个文件应当出现在 layout 文件夹。双击这个文件来编辑它。这个文件会控制 AutoCompleteTextView 活动的布局，所以你需要在布局中有一个 AutoCompleteTextView。增加过 AutoCompleteTextView 的 XML 文件应当如下：

```
<AutoCompleteTextView
android:id="@+id/testAutoComplete"
android:layout_width="fill_parent"
android:layout_height="wrap_content"/>
```

你已经在.xml 文件中创建了几个 Views 了，所以你应当熟悉这个格式。对于 AutoCompleteTextView，没什么特别之处。你设定 id 到 testAutoComplete，还有相应的宽度和高度到 fill_parent 和 wrap_content，还应该为两个按钮增加布局。这些按钮将被用于你将改变的属性控制。命名按钮为 autoCompleteButton 和 textColorButton，如下：

```
<Button android:id="@+id/autoCompleteButton"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:text="Change Layout"/>
<Button android:id="@+id/textColorButton"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
```

```
android:text="Change Text Color"/>
```

有了新增的三个 View 布局，你完成的 autocomplete.xml 文件看上去应该像这样：

```
<?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"
>
<AutoCompleteTextView android:id="@+id/testAutoComplete"
android:layout_width="fill_parent"
android:layout_height="wrap_content"/>
<Button android:id="@+id/autoCompleteButton"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:text="Change Layout"/>
<Button android:id="@+id/textColorButton"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:text="Change Text Color"/>
</LinearLayout>
```

创建一个 autocomplete.java 文件

跟从本章“创建一个新的 java 文件”的介绍。第一件要做的事就是为你的 Views 输入包装。在这个活动中，使用了两个 Views，AutoCompleteTextView 和按钮。你还需要设置颜色和一个 ArrayAdapter。因此，输入下面包装到活动中：

```
package android_programmers_guide.AndroidViews;
import android.app.Activity;
import android.os.Bundle;
import android.view.View;
import android.widget.ArrayAdapter;
import android.widget.AutoCompleteTextView;
import android.widget.Button;
import android.graphics.Color;
```

注意

现在你可能不知道它们的用途，先加入它们吧，我会解释的。

为 AutoCOmplete 类增加初始的结构到 autocomplete.java 文件中：

```
public class AutoComplete extends Activity {
@Override
```

```
public void onCreate(Bundle icicle) {
super.onCreate(icicle);
}
}
```

这个类给了你建造活动其它部分的基础。这个活动的所有功能将会被围绕这个类建造。第一件要做的事就是从 autocomplete.xml 中装载布局：

```
setContentView(R.layout.autocomplete);
```

对于本例，将创建 AutoComplete TextView，所以它包含一年中的月份。当一个用户在框中输入，它会猜测那个月份用户试图去输入。假定 AutoComplete TextView 将包含月份的清单，你需要来创建一个可以被赋值到 AutoCompleteTextView 的清单。

创建字符串数组并赋值月份数值到其中：

```
static final String[] Months = new String[]{
"January","February","March","April","May","June","July","August",
"September","October","November","December"
};
```

下一个任务是复制这个字符串到 AutoCompleteTextView。到目前为止，你已经创建了一些 Views 了。所以，创建 AutoCompleteTextView 的代码看上去应该很熟悉。你之前没看到过的就是把字符串赋值给 View：

```
ArrayAdapter<String> monthArray = new
ArrayAdapter<String>(this,
android.R.layout.simple_list_item_1, Months);
final AutoCompleteTextView textView =
(AutoCompleteTextView)
findViewById(R.id.testAutoComplete);
textView.setAdapter(monthArray);
```

在第一行，拿去创建的字符串数组并且复制到一个名为 monthArray 的 ArrayAdapter。下一步，你通过在 .xml 文件中定位来例示 AutoCompleteTextView。最后，使用 setAdapter() 方法来赋值 monthArray ArrayAdapter 到 AutoCompleteTextView 中。

下一个零星的代码例示那两个按钮。与上一章的代码相同。唯一和你所写代码不同的是你正在呼叫两个函数，changeOption 和 changeOption2，而这些，你还没有创建呢。

注意

你在传递 AutoCompleteTextView 到函数呼叫。当你创建函数还需要创建参数。

```
final Button changeButton = (Button)
findViewById(R.id.autoCompleteButton);
changeButton.setOnClickListener(new Button.OnClic
```

```
kListener() {
public void onClick(View v){
changeOption(textView);
}
});
final Button changeButton2 = (Button)
findViewById(R.id.textColorButton);
changeButton2.setOnClickListener(new
Button.OnClickListener() {
public void onClick(View v){
changeOption2(textView);
}
});
```

被这些按钮呼叫的函数将被用于在 AutoComplete TextView 改变布局属性。这两个我选择来修改的属性（通过两个按钮）是布局的高度和文本的颜色。你将设置一个按钮来改变 AutoCompleteTextView 的布局高度，从 30 到 100 并且返回。另一个按钮将改变 View 内文本的为红色。

函数 changeOption()会改变 AutoCompleteTextView 的布局高度。代码非常的简单：

```
public void changeOption(AutoCompleteTextView
text){
if (text.getHeight()==100){
text.setHeight(30);
}
else{
text.setHeight(100);
}
}
```

在这个函数中你要做的就是检查当前 AutoCompleteTextView 的高度。如果高度是 100，把它设为 30，否则设为 100。

changeOption2()函数也简单：

```
public void changeOption2(AutoCompleteTextView
text){
text.setTextColor(Color.RED);
}
}
```

这个函数简单的把 AutoCompleteTextView 的文本颜色设为 Color.RED。

Color.RED 的数值从 android.graphics.Color 包装中导入。你可以浏览这个包装并且改变这个颜色到任何的数值。我选择红色是因为它比较突出。

完整的 autocomplete.java 文件应当看起来像这样：

```
package android_programmers_guide.AndroidViews;
import android.app.Activity;
import android.os.Bundle;
import android.view.View;
import android.widget.ArrayAdapter;
import android.widget.AutoCompleteTextView;
import android.widget.Button;
import android.graphics.Color;
public class AutoComplete extends Activity {
@Override
public void onCreate(Bundle icicle) {
super.onCreate(icicle);
setContentView(R.layout.autocomplete);
ArrayAdapter<String> monthArray = new ArrayAdapter<String>(this,
android.R.layout.simple_list_item_1, Months);
final AutoCompleteTextView textView = (AutoCompleteTextView)
findViewById(R.id.testAutoComplete);
textView.setAdapter(monthArray);
final Button changeButton = (Button)
findViewById(R.id.autoCompleteButton);
changeButton.setOnClickListener(new Button.OnClickListener() {
public void onClick(View v){
changeOption(textView);
}
});
final Button changeButton2 = (Button)
findViewById(R.id.textColorButton);
changeButton2.setOnClickListener(new Button.OnClickListener() {
public void onClick(View v){
changeOption2(textView);
}
});
}
static final String[] Months = new String[]{
"January","February","March","April","May","June","July","August",
"September","October","November","December"
};
public void changeOption(AutoCompleteTextView text){
if (text.getHeight()==100){
text.setHeight(30);
}
else{
text.setHeight(100);
}
```

```
}
public void changeOption2(AutoCompleteTextView text){
text.setTextColor(Color.RED);
}
}
```

增加一个 Intent 过滤器

在运行这个应用程序之前最后一件事就是在 AndroidManifest.xml 文件中设置 intent 过滤器。然后你能从菜单中呼叫 intent 了。Intent 过滤器的代码如下：

```
<activity android:name=".AutoComplete"
android:label="AutoComplete">
<intent-filter>
<action android:name="android.intent.action.MAIN"
/>
<category
android:name="android.intent.category.LAUNCHER"
/>
</intent-filter>
</activity>
```

这儿是本项目完成的 AndroidManifest.xml 文件：

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

```
</application>
</manifest>
Handling the Intent Call
With AndroidManifest.xml complete, add the following
function to AndroidViews.java:
public void showAutoComplete(){
Intent autocomplete = new Intent(this,
AutoComplete.class);
startActivity(autocomplete);
}
```

当从 select/case 声明中呼叫,这个函数将打开 autocomplete 活动,编辑 select 声明的 case 0 来让它呼叫新的函数:

```
case 0:
showAutoComplete();
return true;
```

在 Android 模拟器中运行这个应用程序。当主活动启动后，点击菜单按钮的 AutoComplete 菜单项。点击后应当把你带到 autocomplete 活动。

要测试 AutoCompleteTextView，开始输入单词 January。在你输入几个字母后，你将会看到 January 出现在文本框中。下一步，点击 Change Layout Button 按钮，结果会是一个扩展的文本输入框。现在点击 chang Text Color 按钮并且输入一些文本。

下一节会给你项目中剩下 5 个 Views 的代码支持。

按钮

按钮 第八章(5)

看看下面的代码。这段代码代表了四个文件，AndroidManifest.xml, Button.xml, testButton.java, 和 AndroidViews.java。增加代码到现存的 AndroidViews 活动中。

警告

如果你没有一开始就跟从本章，你执行代码时可能会遇到麻烦。要确保得到完整的项目，请从本章的开始开始阅读。

AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest
xmlns:android=http://schemas.android.com/apk/res/android
package="android_programmers_guide.AndroidViews"
<application android:icon="@drawable/icon">
```

```
<activity android:name=".AndroidViews"
android:label="@string/app_name">
<intent-filter>
<action android:name="android.intent.action.MAIN" />
<category
android:name="android.intent.category.LAUNCHER" />
</intent-filter>
</activity>
<activity                    android:name=".AutoComplete"
android:label="AutoComplete">
<intent-filter>
<action android:name="android.intent.action.MAIN" />
<category
android:name="android.intent.category.LAUNCHER"/>
</intent-filter>
</activity>
<activity                      android:name=".testButton"
android:label="TestButton">
<intent-filter>
<action android:name="android.intent.action.MAIN" />
<category
android:name="android.intent.category.LAUNCHER"/>
</intent-filter>
</activity>
</application>
</manifest>
```

Button.xml

```
<?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"
Chapter 8: Lists, Menus, and Other Views 173
android:layout_height="fill_parent">
<Button android:id="@+id/testButton"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:text="This is the test Button"/>
<Button android:id="@+id/layoutButton"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:text="Change Layout"/>
```

```
<Button android:id="@+id/textColorButton"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:text="Change Text Color"/>
</LinearLayout>
```

testButton.java

```
package android_programmers_guide.AndroidViews;
import android.app.Activity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.graphics.Color;
public class testButton extends Activity {
@Override
public void onCreate(Bundle icicle) {
super.onCreate(icicle);
setContentView(R.layout.Button);
final Button Button = (Button)
findViewById(R.id.testButton);
final Button changeButton =
(Button)findViewById(R.id.layoutButton);
changeButton.setOnClickListener(new
Button.OnClickListener() {
public void onClick(View v){
changeOption(Button); }
});
final Button changeButton2 = (Button)
findViewById(R.id.textColorButton);
changeButton2.setOnClickListener(new
Button.OnClickListener() {
public void onClick(View v){
changeOption2(Button);
}
});
}
public void changeOption(Button Button){
if (Button.getHeight()==100){
Button.setHeight(30);
}
174 Android: A Programmer's Guide
Chapter 8: Lists, Menus, and Other Views 175
else{
```

```
Button.setHeight(100);
}
}
public void changeOption2(Button Button){
Button.setTextColor(Color.RED);
}
}
```

AndroidViews.java

```
package android_programmers_guide.AndroidViews;
import android.app.Activity;
import android.os.Bundle;
import android.view.Menu;
import android.content.Intent;
public class AndroidViews extends Activity {
/** Called when the Activity is first created. */
@Override
public void onCreate(Bundle icicle) {
super.onCreate(icicle);
setContentView(R.layout.main);
}
@Override
public boolean onCreateOptionsMenu(Menu menu) {
super.onCreateOptionsMenu(menu);
menu.add(0, 0, "AutoComplete");
menu.add(0, 1, "Button");
menu.add(0, 2, "CheckBox");
menu.add(0, 3, "EditText");
menu.add(0, 4, "RadioGroup");
menu.add(0, 5, "Spinner");
return true;
}
@Override
public boolean onOptionsItemSelected(Menu.Item
item){
switch (item.getId()) {
case 0:
showAutoComplete();
return true;
case 1:
showButton();
return true;
case 2:
```

```
return true;
case 3:
return true;
case 4:
return true;
case 5:
return true;
}
return true;
}
public void showButton() {
Intent showButton = new Intent(this,
testButton.class);
startActivity(showButton);
}
public void showAutoComplete(){
Intent autocomplete = new Intent(this,
AutoComplete.class);
startActivity(autocomplete);
}
}
```

启动你的应用程序并且选择从菜单上选择按钮选项。试着点击 Change Layout 按钮。再一次，对文本来说，结果是一个较宽的显示区域，点击改变 Text Color 按钮并且文本变成红色。

CheckBox

CheckBox 第八章(6)

在本节中，将为 CheckBox View 创建一个活动。创建活动的步骤和前面的章节一致。因此，将会为你提供三个主要活动文件——AndroidManifest.xml, checkbox.xml, 和 testCheckBox.java。这些文件在下面的部分提供。

AndroidManifest.xml

这个部分包含当前 AndroidView 的完整代码。如果你使用 Eclipse，修改你活动的 AndroidManifest.xml 文件，来使得它和下面一样：

```
<?xml version="1.0" encoding="utf-8"?>
<manifest
xmlns:android=http://schemas.android.com/apk/res/android
package="android_programmers_guide.AndroidViews">
<application android:icon="@drawable/icon">
```

```
<activity android:name=".AndroidViews"
android:label="@string/app_name">
<intent-filter>
<action android:name="android.intent.action.MAIN" />
<category
android:name="android.intent.category.LAUNCHER" />
</intent-filter>
</activity>
<activity                    android:name=".AutoComplete"
android:label="AutoComplete">
<intent-filter>
<action android:name="android.intent.action.MAIN" />
<category
android:name="android.intent.category.LAUNCHER" />
</intent-filter>
</activity>
<activity                      android:name=".testButton"
android:label="TestButton">
<intent-filter>
<action android:name="android.intent.action.MAIN" />
<category
android:name="android.intent.category.LAUNCHER"/>
</intent-filter>
</activity>
<activity                    android:name=".testCheckBox"
android:label="TestCheckBox">
<intent-filter>
<action android:name="android.intent.action.MAIN" />
<category
android:name="android.intent.category.LAUNCHER"/>
</intent-filter>
</activity>
</application>
</manifest>
```

checkbox.xml

这个部分显示了完整的 checkbox.xml 代码。使用在本章早些时候提到的方法，在项目中创建一个新的 XML 文件，把它命名为 checkbox.xml。使用下面的代码修改你的文件。

```
<?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"
>
<CheckBox android:id="@+id/testCheckBox"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:text="This is the test CheckBox"/>
<Button android:id="@+id/layoutButton"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:text="Change Layout"/>
<Button android:id="@+id/textColorButton"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:text="Change Text Color"/>
</LinearLayout>
```

testCheckBox.java

本部分涵盖了执行 Checkbox 活动所需的最后新文件。在项目中创建一个新的.java 文件，并把它命名为 testCheckBox.java。这个是活动的主文件并且包含可执行代码。在 testCheckBox.java 文件中使用下列代码：

```
package android_programmers_guide.AndroidViews;
import android.app.Activity;
import android.os.Bundle;
import android.view.View;
import android.widget.CheckBox;
import android.widget.Button;
import android.graphics.Color;
public class testCheckBox extends Activity {
180 Android: A Programmer’ s Guide
@Override
public void onCreate(Bundle icicle) {
super.onCreate(icicle);
setContentView(R.layout.checkbox);
final CheckBox checkbox =
(CheckBox)findViewById(R.id.testCheckBox);
final Button changeButton =
(Button)findViewById(R.id.layoutButton);
changeButton.setOnClickListener(new
Button.OnClickListener() {
public void onClick(View v){
changeOption(checkbox); }
});
final Button changeButton2 = (Button)
```

```
findViewById(R.id.textColorButton);
changeButton2.setOnClickListener(new
Button.OnClickListener() {
public void onClick(View v){
changeOption2(checkbox);
}
});
}
public void changeOption(CheckBox checkbox){
if (checkbox.getHeight()==100){
checkbox.setHeight(30);
}
else{
checkbox.setHeight(100);
}
}
public void changeOption2(CheckBox checkbox){
checkbox.setTextColor(Color.RED);
}
}
```

AndroidViews.java

创建这个活动的最后一步就是编辑 AndroidViews.java 文件。如果你要从 AndroidViews 主活动中呼叫 testCheckBox 活动，你必须增加代码到 AndroidViews.java 中。用下面的代码和你现存在 AndroidViews.java 中的代码作个比较。

```
package android_programmers_guide.AndroidViews;
import android.app.Activity;
import android.os.Bundle;
import android.view.Menu;
import android.content.Intent;
Chapter 8: Lists, Menus, and Other Views 181
public class AndroidViews extends Activity {
/** Called when the Activity is first created. */
@Override
public void onCreate(Bundle icicle) {
super.onCreate(icicle);
setContentView(R.layout.main);
}
@Override
public boolean onCreateOptionsMenu(Menu menu) {
super.onCreateOptionsMenu(menu);
menu.add(0, 0, "AutoComplete");
menu.add(0, 1, "Button");
```

```
menu.add(0, 2, "CheckBox");
menu.add(0, 3, "EditText");
menu.add(0, 4, "RadioGroup");
menu.add(0, 5, "Spinner");
return true;
}
@Override
public boolean onOptionsItemSelected(Menu.Item
item){
switch (item.getId()) {
case 0:
showAutoComplete();
return true;
case 1:
showButton();
return true;
case 2:
showCheckBox()
return true;
case 3:
return true;
case 4:
return true;
case 5:
return true;
}
return true;
}
public void showButton() {
Intent showButton = new Intent(this,
testButton.class);
startActivity(showButton);
}
public void showAutoComplete(){
Intent autocomplete = new Intent(this,
AutoComplete.class);
startActivity(autocomplete);
}
public void showCheckBox() {
Intent checkbox = new Intent(this,
testCheckBox.class);
startActivity(checkbox);
}
}
```

启动应用程序并从菜单中选择 CheckBox 选项。点击 Change Layout 和 Change Text Color 按钮。

EditText

EditText 第八章(7)

在本节中，和上一节很类似，你为 EditText View 创建一个活动。创建活动的步骤和前几节是一样的。因此，你将被提供三个主要活动文件的代码。—AndroidManifest.xml, edittext.xml, 和 testEditText.java。这些在下面提供给你。

AndroidManifest.xml

本部分包含当前 AndroidView 的 AndroidManifest.xml 完整代码。如果你使用 Eclipse，修改你的活动的 AndroidManifest.xml 文件，使它和下面类似：

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android=http://schemas.android.com/apk/res/android
package="android_programmers_guide.AndroidViews">
<application android:icon="@drawable/icon">
<activity android:name=".AndroidViews"
android:label="@string/app_name">
<intent-filter>
<action android:name="android.intent.action.MAIN" />
<category android:name="android.intent.category.LAUNCHER" />
</intent-filter>
</activity>
<activity android:name=".AutoComplete" android:label="AutoComplete">
<intent-filter>
<action android:name="android.intent.action.MAIN" />
<category android:name="android.intent.category.LAUNCHER"/>
</intent-filter>
</activity>
<activity android:name=".testButton" android:label="TestButton">
<intent-filter>
<action android:name="android.intent.action.MAIN" />
<category android:name="android.intent.category.LAUNCHER"/>
</intent-filter>
</activity>
<activity android:name=".testCheckBox" android:label="TestCheckBox">
<intent-filter>
<action android:name="android.intent.action.MAIN" />
<category android:name="android.intent.category.LAUNCHER"/>
</intent-filter>
</activity>
```

```
<activity android:name=".testEditText" android:label="TestEditText">
<intent-filter>
<action android:name="android.intent.action.MAIN" />
<category android:name="android.intent.category.LAUNCHER"/>
</intent-filter>
</activity>
</application>
</manifest>
```

edittext.xml

这个部分展示了 edittext.xml 文件的完整代码。使用本章前面的指示，在项目中创建一个名为 edittext.xml 的文件。使用下面的代码来修改你的文件。

```
<?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"
>
<EditText android:id="@+id/testEditText"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
/>
<Button android:id="@+id/layoutButton"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:text="Change Layout"/>
<Button android:id="@+id/textColorButton"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:text="Change Text Color"/>
</LinearLayout>
```

testEditText.java

本节包含了执行 EditText 活动需要的最后一个文件。在项目中创建一个名为 testEditText.java 的文件。这是个活动的主要文件并且包含了可执行代码。在 testEditText.java 文件中使用下面的代码来完成这个活动。

```
package android_programmers_guide.AndroidViews;
import android.app.Activity;
import android.os.Bundle;
import android.view.View;
import android.widget.EditText;
import android.widget.Button;
```

```
import android.graphics.Color;
public class testEditText extends Activity {
@Override
public void onCreate(Bundle icicle) {
super.onCreate(icicle);
setContentView(R.layout.edittext);
final EditText edittext = (EditText)findViewById(R.id.testEditText);
final Button changeButton = (Button)findViewById(R.id.layoutButton);
changeButton.setOnClickListener(new Button.OnClickListener() {
public void onClick(View v){
changeOption(edittext); }
});
final Button changeButton2 = (Button)
findViewById(R.id.textColorButton);
changeButton2.setOnClickListener(new Button.OnClickListener() {
public void onClick(View v){
changeOption2(edittext);
}
});
}
public void changeOption(EditText edittext){
if (edittext.getHeight()==100){
edittext.setHeight(30);
}
else{
edittext.setHeight(100);
}
}
public void changeOption2(EditText edittext){
edittext.setTextColor(Color.RED);
}
}
```

AndroidViews.java

创建这个活动的最后一步是编辑 AndroidView.java。如果你要从主要 AndroidView 活动中呼叫 testEditText 活动，你必须增加代码到 AndroidView.java 中。与你当前 AndroidViews.java 文件中的代码与下面进行比较。增加需求的代码来完成文件。

```
package android_programmers_guide.AndroidViews;
import android.app.Activity;
import android.os.Bundle;
import android.view.Menu;
import android.content.Intent;
```

```
public class AndroidViews extends Activity {
/** Called when the Activity is first created. */
@Override
public void onCreate(Bundle icicle) {
super.onCreate(icicle);
setContentView(R.layout.main);
}
@Override
public boolean onCreateOptionsMenu(Menu menu) {
super.onCreateOptionsMenu(menu);
menu.add(0, 0, "AutoComplete");
menu.add(0, 1, "Button");
menu.add(0, 2, "CheckBox");
menu.add(0, 3, "EditText");
menu.add(0, 4, "RadioGroup");
menu.add(0, 5, "Spinner");
return true;
}
@Override
public boolean onOptionsItemSelected(Menu.Item item){
switch (item.getId()) {
case 0:
showAutoComplete();
return true;
case 1:
showButton();
return true;
case 2:
showCheckBox();
return true;
case 3:
showEditText();
return true;
case 4:
showRadioGroup();
return true;
case 5:
showSpinner();
return true;
}
return true;
}
public void showButton() {
Intent showButton = new Intent(this, testButton.class);
```

```
startActivity(showButton);
}
public void showAutoComplete(){
Intent autocomplete = new Intent(this, AutoComplete.class);
Chapter 8: Lists, Menus, and Other Views 187
startActivity(autocomplete);
}
public void showCheckBox(){
Intent checkbox = new Intent(this, testCheckBox.class);
startActivity(checkbox);
}
public void showEditText() {
Intent edittext = new Intent(this, testEditText.class);
startActivity(edittext);
}
}
```

启动应用程序并从菜单中选择 EditText 选项。点击 Changae Layout 和 Change Test Color 按钮。

RadioGroup

RadioGroup 第八章(8)

在本章中将为 RadioGroup View 创建一个活动。创建活动的步骤和前节一致。因此会为你提供三个主要文件—AndroidManifest.xml， radiogroup.xml， 和 testRadioGroup.java。这些文件将在下面提供给你。

AndroidManifest.xml

这个部分包含当前 AndroidView AndroidManifest.xml 的完整代码。如果你使用 Eclipse，修改活动的 AndnroidManifest.xml 文件如下：

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

```
<action android:name="android.intent.action.MAIN" />
<category android:name="android.intent.category.LAUNCHER"/>
</intent-filter>
</activity>
<activity android:name=".testButton" android:label="TestButton">
<intent-filter>
<action android:name="android.intent.action.MAIN" />
<category android:name="android.intent.category.LAUNCHER"/>
</intent-filter>
</activity>
<activity android:name=".testCheckBox" android:label="TestCheckBox">
<intent-filter>
<action android:name="android.intent.action.MAIN" />
<category android:name="android.intent.category.LAUNCHER"/>
</intent-filter>
</activity>
<activity android:name=".testEditText" android:label="TestEditText">
<intent-filter>
<action android:name="android.intent.action.MAIN" />
<category android:name="android.intent.category.LAUNCHER"/>
</intent-filter>
</activity>
<activity android:name=".testRadioGroup" android:label="Test
RadioGroup">
<intent-filter>
<action android:name="android.intent.action.MAIN" />
<category android:name="android.intent.category.LAUNCHER"/>
</intent-filter>
</activity>
</application>
</manifest>
```

radiogroup.xml

这个部分展示了完整的 radiogroup.xml 文件的完整代码。使用本章前节描述的方法，在项目中创建一个名为 radiogroup.xml 的文件。

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
xmlns:android=http://schemas.android.com/apk/res/android
Chapter 8: Lists, Menus, and Other Views 191
android:orientation="vertical"
android:layout_width="fill_parent"
android:layout_height="fill_parent"
>
<RadioGroup android:id="@+id/testRadioGroup"
```

```
android:layout_width="fill_parent"
android:layout_height="wrap_content" >
<RadioButton
android:text="Radio 1"
android:id="@+id/radio1"
/>
<RadioButton
android:text="Radio 2"
android:id="@+id/radio2" />
</RadioGroup>
<Button android:id="@+id/enableButton"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:text="Set isEnabled"/>
<Button android:id="@+id/backgroundColorButton"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:text="Change Background Color"/>
</LinearLayout>
```

testRadioGroup.java

本部分包含执行 RadioGroup 活动最后所需的文件。在项目中创建一个名为 testRadioGroup.java 的文件。这是个活动的主要文件并且包含可执行的代码。在 testRadioGroup.java 文件中使用下面的代码来完成文件。

```
package android_programmers_guide.AndroidViews;
import android.app.Activity;
import android.os.Bundle;
import android.view.View;
import android.widget.RadioGroup;
import android.widget.Button;
import android.graphics.Color;
public class testRadioGroup extends Activity {
@Override
public void onCreate(Bundle icicle) {
super.onCreate(icicle);
setContentView(R.layout.radiogroup);
final RadioGroup radiogroup = (RadioGroup)
findViewById(R.id.testRadioGroup);
final Button changeButton = (Button)findViewById(R.id.enableButton);
changeButton.setOnClickListener(new Button.OnClickListener() {
public void onClick(View v){
changeOption(radiogroup); }
});
final Button changeButton2 = (Button)
```

```
findViewById(R.id.backgroundColorButton);
changeButton2.setOnClickListener(new Button.OnClickListener() {
public void onClick(View v){
changeOption2(radiogroup);
}
});
}
public void changeOption(RadioGroup radiogroup){
if (radiogroup.isEnabled()){
radiogroup.setEnabled(false);
}
else{
radiogroup.setEnabled(true);
} }
public void changeOption2(RadioGroup radiogroup){
radiogroup.setBackgroundColor(Color.RED);
}
}
```

AndroidViews.java

最后创建活动的部分是编辑 AndroidViews.java。如果你要从主要活动中呼叫 testRadioGroup 活动，你必须在 AndroidViews.java 文件中增加代码。与你当前 AndroidViews.java 文件中的代码相比较，增加所需的代码来完成文件。

```
package android_programmers_guide.AndroidViews;
import android.app.Activity;
import android.os.Bundle;
import android.view.Menu;
import android.content.Intent;
public class AndroidViews extends Activity {
/** Called when the Activity is first created. */
@Override
public void onCreate(Bundle icicle) {
super.onCreate(icicle);
setContentView(R.layout.main);
}
@Override
public boolean onCreateOptionsMenu(Menu menu) {
super.onCreateOptionsMenu(menu);
menu.add(0, 0, "AutoComplete");
menu.add(0, 1, "Button");
menu.add(0, 2, "CheckBox");
menu.add(0, 3, "EditText");
menu.add(0, 4, "RadioGroup");
```

```
menu.add(0, 5, "Spinner");
return true;
}
@Override
public boolean onOptionsItemSelected(Menu.Item item){
switch (item.getId()) {
case 0:
showAutoComplete();
return true;
case 1:
showButton();
return true;
case 2:
showCheckBox();
return true;
case 3:
showEditText();
return true;
case 4:
showRadioGroup();
return true;
case 5:
showSpinner();
return true;
}
return true;
}
public void showButton() {
Intent showButton = new Intent(this, testButton.class);
startActivity(showButton);
Chapter 8: Lists, Menus, and Other Views 193
194 Android: A Programmer’s Guide
}
public void showAutoComplete(){
Intent autocomplete = new Intent(this, AutoComplete.class);
startActivity(autocomplete);
}
public void showCheckBox(){
Intent checkbox = new Intent(this, testCheckBox.class);
startActivity(checkbox);
}
public void showEditText() {
Intent edittext = new Intent(this, testEditText.class);
startActivity(edittext);
```

```
}
public void showRadioGroup(){
Intent radiogroup = new Intent(this, testRadioGroup.class);
startActivity(radiogroup);
}
public void showSpinner(){
}
}
```

启动应用程序并从菜单中选择 RadioGroup 选项。

试着点击 Set isEnabled 和 Change BackGroud Color 按钮。注意 Set isEnabled 按钮把 RadioGroup 设为不可用，而 Change Backgroud Color 按钮改变组的背景色。

Spinner

Spinner 第八章(9)

在本节中将为 Spinner View 创建一个活动。Spinner View 和其它编程语言里的 ComboBox 相类似。创建这个活动的步骤和前面部分的一样。因此，还是会提供给你三个主要活动的代码文件—AndroidManifest.xml, spinner.xml, 和 testSpinner.java。下面就是这些提供的文件。

AndroidManifest.xml

本节包含当前 AndroidViews 的 AndroidManifest.xml 文件的完整代码。如果你使用 Eclipse，修改活动的 AndroidManifest.xml 文件使它和下面一样：

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

```
</activity>
<activity android:name=".testButton" android:label="TestButton">
<intent-filter>
<action android:name="android.intent.action.MAIN" />
<category android:name="android.intent.category.LAUNCHER"/>
</intent-filter>
</activity>
<activity                                    android:name=".testCheckBox"
android:label="TestCheckBox">
<intent-filter>
<action android:name="android.intent.action.MAIN" />
<category android:name="android.intent.category.LAUNCHER"/>
</intent-filter>
</activity>
<activity                                    android:name=".testEditText"
android:label="TestEditText">
<intent-filter>
<action android:name="android.intent.action.MAIN" />
<category android:name="android.intent.category.LAUNCHER"/>
</intent-filter>
</activity>
<activity android:name=".testRadioGroup" android:label="Test
RadioGroup">
<intent-filter>
<action android:name="android.intent.action.MAIN" />
<category android:name="android.intent.category.LAUNCHER"/>
</intent-filter>
</activity>
<activity android:name=".testSpinner" android:label="Test Spinner">
<intent-filter>
<action android:name="android.intent.action.MAIN" />
<category android:name="android.intent.category.LAUNCHER" />
</intent-filter>
</activity>
</application>
</manifest>
```

spinner.xml

本节展示了 spinner.xml 文件的完整代码。在项目中创建一个名为 spinner.xml 的文件。使用下面的代码修改你的文件。

```
<?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"
>
<Spinner android:id="@+id/testSpinner"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
/>
<Button android:id="@+id/enableButton"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:text="Set isEnabled"/>
<Button android:id="@+id/backgroundColorButton"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:text="Change Background Color"/>
</LinearLayout>
```

testSpinner.java

本节包含了执行 Spinner 活动所需要的最后一个文件。在项目中创建一个名为 testSpinner.java 的新文件。这是个活动的主要文件并且包含可执行代码。在 testSpinner.java 文件中使用下面的代码来完成这个活动。

```
package android_programmers_guide.AndroidViews;
import android.app.Activity;
import android.os.Bundle;
import android.view.View;
import android.widget.ArrayAdapter;
import android.widget.Spinner;
import android.widget.Button;
import android.graphics.Color;
198 Android: A Programmer’s Guide
public class testSpinner extends Activity {
@Override
public void onCreate(Bundle icicle) {
super.onCreate(icicle);
setContentView(R.layout.spinner);
final Spinner spinner = (Spinner) findViewById(R.id.testSpinner);
ArrayAdapter<String> adapter = new ArrayAdapter<String>(this,
android.R.layout.simple_spinner_item, Months);
adapter.setDropDownViewResource(android.R.layout.simple_spinner_dropd
own_item);
spinner.setAdapter(adapter);
```

```
final Button changeButton = (Button)findViewById(R.id.enableButton);
changeButton.setOnClickListener(new Button.OnClickListener() {
public void onClick(View v){
changeOption(spinner); }
});
final Button changeButton2 = (Button)
findViewById(R.id.backgroundColorButton);
changeButton2.setOnClickListener(new Button.OnClickListener() {
public void onClick(View v){
changeOption2(spinner);
}
});
}
static final String[] Months = new String[]{
"January","February","March","April","May","June","July","August",
"September","October","November","December"
};
public void changeOption(Spinner spinner){
if (spinner.isEnabled()){
spinner.setEnabled(false);
}
else{
spinner.setEnabled(true);
}
}
public void changeOption2(Spinner spinner){
spinner.setBackgroundColor(Color.RED);
}
}
```

AndroidViews.java

创建活动的最后一个步骤就是编辑 AndroidViews.java。如果你要从主活动 AndroidViews 中呼叫 testSpinner 活动，你必须增加代码到 AndroidViews.java 中。用当前的 AndroidViews.java 和下面的代码作个比较。增加代码来完成文件。

```
package android_programmers_guide.AndroidViews;
import android.app.Activity;
import android.os.Bundle;
import android.view.Menu;
import android.content.Intent;
public class AndroidViews extends Activity {
/** Called when the Activity is first created. */
@Override
public void onCreate(Bundle icicle) {
```

```
super.onCreate(icicle);
setContentView(R.layout.main);
}
@Override
public boolean onCreateOptionsMenu(Menu menu) {
super.onCreateOptionsMenu(menu);
menu.add(0, 0, "AutoComplete");
menu.add(0, 1, "Button");
menu.add(0, 2, "CheckBox");
menu.add(0, 3, "EditText");
menu.add(0, 4, "RadioGroup");
menu.add(0, 5, "Spinner");
return true;
}
@Override
public boolean onOptionsItemSelected(Menu.Item item){
switch (item.getId()) {
case 0:
showAutoComplete();
return true;
case 1:
showButton();
return true;
case 2:
showCheckBox();
return true;
case 3:
showEditText();
return true;
case 4:
showRadioGroup();
return true;
case 5:
showSpinner();
return true;
}
return true;
}
public void showButton() {
Intent showButton = new Intent(this, testButton.class);
startActivity(showButton);
}
public void showAutoComplete(){
Intent autocomplete = new Intent(this, AutoComplete.class);
```

```
startActivity(autocomplete);
}
public void showCheckBox(){
Intent checkbox = new Intent(this, testCheckBox.class);
startActivity(checkbox);
}
public void showEditText() {
Intent edittext = new Intent(this, testEditText.class);
startActivity(edittext);
}
public void showRadioGroup(){
Intent radiogroup = new Intent(this, testRadioGroup.class);
startActivity(radiogroup);
}
public void showSpinner(){
Intent spinner = new Intent(this, testSpinner.class);
startActivity(spinner);
}
}
```

启动应用程序并从菜单中选择 Spinner 选项。试着点击 Set isEnabled 和 Change Backgroud Color 按钮。

试试这个：修改更多的 View 属性

试试这个：修改更多的 View 属性 第八章(10)

为活动修改按钮动作在每个 View 中来改变不同的属性：

- 使用 Eclipse 的特性列表来查看每个 View 有哪些属性。
- 在任一个给定的活动中编辑两个按钮的功能来更改按钮和 View 的交互活动。

在下一章中，将用到谷歌的 API（Google API)。将创建一个应用程序和 GTalk 交互。这将会给你更多的知识关于如何构造独特的应用程序。

问专家

Q：如果我在应用程序中使用多个 Views，我可以使用 android.widget.*输入整个 widget 包装吗？

A：是的。但是，我会保守的使用呼叫。当你输入整个包装，你增加了包装的所有代码到活动中。这个管理的不好会让活动速度变慢。我会只输入我需要的包装，试图减少活动中的代码。

第九章 使用手机的 GPS 功能

使用手机的 GPS 功能 第九章(1)

关键技能 & 概念

- 使用 Android 的定位服务 APIs
- 从 GPS 硬件获得坐标数据
- 改变活动的外观并且和 RelativeLayout 接触
- 使用一个 MapView 来绘制你的当前位置
- 使用谷歌地图来找到你的当前位置

在本章中，你将学习关于 Android 定位的 API。本章的作用是非常重要的，如果你想要让 Android 和 GPS 硬件一起工作的话。你将使用位置基础的 API 来收集你当前的位置并把它在屏幕上显示出来。到本章的结尾，你将在手机上使用谷歌地图来显示你的当前位置。

你还会学习到一些关于活动的更深，创新的技巧。资源，如 RelativeLayouts 和小按钮将允许你创建更友好，可视的活动。第一节，你将学习使用设备的 GPS 硬件来获得当前的位置。但是，在跳到那个部分之前，你需要先创建一个项目。在 Eclipse 中创建一个项目并命名为 AndroidLBS。

使用 Android 位置基础 API

Android SDK 包含了一个 API，它被定制为帮助接口活动与设备上任何的 GPS 硬件。这章假定你的设备包含 GPS 硬件。

警告

Android 平台的手机不要求包含一个照相机，也不要求包含 GPS 硬件，虽然很多的型号可能包含照相机和 GPS 硬件。Android 包含了 Android 位置基础 API 预期 GPS 硬件会被包含在很多手机上。

因为你工作在一个软件模拟器中，并且不是一个真的设备，GPS 硬件没有被模拟。在本例中，Android 在 adb 服务器中提供了一个文件模拟 GPS 硬件。这个文件放置在 data/misc/location/<provider>，<provider>代表位置信息提供者。Android 提供的<provider>是 data/misc/location/gps

提示

你可以有多重的提供者来模拟不同的方案。因此，你可以创建一个提供者为 test 或者 gps1;无论你愿意用哪个。在具体的 provider 的文件夹内可以用任何数量的文件保留你想要 Android 使用的例子。当你使用 Android 模拟器，你可以使用下面类型的文件来储存/找回 GPS 文体的坐标。每一个文件类型有个不同的格式来提供信息给 Android 位置基础 API

- kml
- nmea
- track

我们来看看每一个文件都做些什么并且互相之间有什么不同。

创建一个 kml 文件

一个.kml 文件 Keyhole Markup 语言文件。这些文件通常被开发用于并且可以被 Google Earth（一款 Google 软件）。Adnroid 位置基础 API 可以分析一个.kml 文件来模拟一个 GPS。

注意

假如你没有 Google Earth。可以从 Google 免费下载。如果你想要开发更多的 Android 位置基础 API 活动，安装这个软件是值得的。

要从 Google Earth 创建一个.kml 文件，打开 Google Earth 并且导航到一个位置。选择文件|另存为并选择 KML。在本例中，它为你所导航到的地点产生一个.kml 文件。下面的.kml 代码就是来自这个文件。仔细看看<coordinates>标签，那就是 Android 位置基础 API 会读取的。

```
<?xml version="1.0" encoding="UTF-8"?>
<kml xmlns="http://earth.google.com/kml/2.2">
<Document>
<name>Tampa, FL.kml</name>
<Styleid="default+icon=http://maps.google.com/mapfiles/kml/pal3/icon52.png">
<IconStyle>
<scale>1.1</scale>
<Icon>
<href>http://maps.google.com/mapfiles/kml/pal3/icon52.png</href>
</Icon>
</IconStyle>
<LabelStyle>
<scale>1.1</scale>
</LabelStyle>
</Style>
<Styleid="default+icon=http://maps.google.com/mapfiles/kml/pal3/icon60.png">
<IconStyle>
<Icon>
<href>http://maps.google.com/mapfiles/kml/pal3/icon60.png</href>
</Icon>
</IconStyle>
</Style>
<StyleMapid="default+nicon=http://maps.google.com/mapfiles/kml/pal3/
icon60.png+hicon=http://maps.google.com/mapfiles/kml/pal3/icon52.png">
<Pair>
<key>normal</key>
<styleUrl>#default+icon=http://maps.google.com/mapfiles/kml/pal3/
icon60.png</styleUrl>
</Pair>
<Pair>
<key>highlight</key>
```

```
<styleUrl>#default+icon=http://maps.google.com/mapfiles/kml/pal3/
icon52.png</styleUrl>
</Pair>
</StyleMap>
<Placemark>
<name>Tampa, FL</name>
<open>1</open>
<address>Tampa, FL</address>
<LookAt>
<longitude>-82.451142</longitude>
<latitude>27.98146</latitude>
<altitude>0</altitude>
<range>38427.828125</range>
<tilt>0</tilt>
<heading>0</heading>
</LookAt>
<styleUrl>#default+nicon=http://maps.google.com/mapfiles/kml/pal3/
icon60.png+hicon=http://maps.google.com/mapfiles/kml/pal3/icon52.png</styleUrl>
<Point>
<coordinates>-82.451142,27.98146,0</coordinates>
</Point>
</Placemark>
</Document>
</kml>
```

你可以用 Google Earth 来创建自己的.kml 文件来模拟不同的位置。当你想要制作一个相应用户不同位置的活动时，这个非常有用。创建.kml 文件如此简单使得模拟 GPS 硬件非常的灵活。

什么是轨迹文件

什么是轨迹文件 第九章(2)

Android 提供的在 gps 文件夹里的文件是一个.**nmea 文件**（国家海事电子协会文件）。一个.nmea 文件可以从任何通用的 **GPS** 产品中输出。这些文件是常用格式并且可以包含多重坐标和海拔，来表现行程和轨迹。下面的部分讨论并且在 Windows 和 Linux 下各自打开这个文件。

在 Windows 中得到 nmea 文件

Android 提供的 nmea 文件展示了一个贯穿旧金山的短的线路。让我们看看 nmea 文件的内部。使用 **adb 工具**把文件从服务器中 pull 到你的桌面：

adb pull<远程文件><本地文件>

下面的插图描述使用 adb 工具 **pull 命令**来检索文件（略）。如果命令执行成功，你应当看到一条消息指示文件下载的大小。导航到 C:\Android 文件夹，你可以看到 adb pull 工具放在这里。

现在 nmea 文件在桌面上，把它与 Notepad 关联。最后打开它来看看它的内容。你会看到很多的坐标数据。

在 Linux 中得到 nmea 文件

如果你在使用 Linux 开发 Android，启动一个终端部分来进入 adb 服务器。让我们来看看如何在 Linux 中检索并且编辑 nmea 文件。

注意（和插图有关，略）

第一步是打开一个新的终端部分（Applications | System Tools | Terminal）。

下一步，使用 adb pull 命令来 pull nmea 文件到 Android 文件夹：

adb pull data/misc/location/gps/nmea Android/

如果你读了关于 Windows 如何得到 nmea 文件的说明，你会发现语法上的不同。C:\ 是没有必要的因为路径结构的不同。

从终端中执行了命令后，结果应当如下所示：

使用 Is 命令来在 Android 文件夹中列出文件。如果命令执行正确，nmea 文件应当出现。我使用 **Fedora GUI** 来导航并且使用系统的 **Text Editor** 打开它。

提示

你也可以使用 **vi 编辑器**从命令行来打开，读取并且编辑 nmea 文件。

现在你已经查看了 nmea 文件并且知道模拟一个 GPS 设备的不同方式，你可以开始来使用 Android 位置基础 API 来创建一个完整特性的活动了。

使用 Android 位置基础 API 读取 GPS

使用 Android 位置基础 API 读取 GPS 第九章(3)

本章剩下的部分是致力于建造一个活动，AndroidLBS，它会从服务器中 nmea 文件中识别用户的位置。本活动的第一个过程非常的简单。

你会创建一个简单的过程，该过程会得到用户当前的 GPS 位置。然后你可以在屏幕上显示这个位置的坐标。在做这个的时候，你会了解到一个对 **Android 位置基础 API** 比较到位的介绍和它的功用。

创建 AndroidLBS 活动

下面是创建这个简单活动的步骤：

1. 调整许可的权限

2. 创建活动的布局

3. 书写代码来允许活动。

4. 运行活动。

调整许可的权限

使用 Android 位置基础 API 是调整认可的权限。使用 Android 位置基础本身不要求任何特别的许可。但是在 GPS 使用 Android 位置基础来存取位置信息需要。

从 Eclipse 中有两种方式可以设置许可。第一个是通过 Android Manifest 许可向导，这个你在第七章用过。在 Eclipse 中，双击 AndroidManifest.xml 来打开 Android Manifest 综览窗口。点击许可链接并使用第七章的方法增加 ACCESS_GPS 和 ACCESS_LOCATION 使用许可。

第二种方法是，你可以手动编辑 AndroidManifest.xml 文件增加许可值到活动中。你会需要下面的代码行到 AndroidManifest.xml 中：

```
<uses-permission android:name="android.permission.ACCESS_GPS">
</uses-permission>
<uses-permission android:name="android.permission.ACCESS_LOCATION">
</uses-permission>
```

这里的语句是用来在<uses-permission>标签中增加许可名称。

当你结束了增加许可，你的 AndroidManifest.xml 文件应当像下面的代码片段。这样的代码应当非常的熟悉了。你在 Intent 过滤器内使用了一个活动和一对许可。

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android=http://schemas.android.com/apk/res/android
package="android_programmers_guide.AndroidLBS">
<application android:icon="@drawable/icon">
<activity android:name=".AndroidLBS"
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>
<uses-permission android:name="android.permission.ACCESS_GPS">
</uses-permission><uses-permission
android:name="android.permission.ACCESS_LOCATION">
</uses-permission></manifest>
```

创建你的布局

要开始创建布局，**在 Eclipse 中打开 main.xml**，你会一共增加一个按钮和 4 个 TextViews 到布局中。按钮可以从 GPS 呼叫信息并显示到 TextViews 中。

按照下面设置按钮，也就是在屏幕的顶部并使用“Where am I”作为显示文本。

```
<Button
android:id="@+id/gpsButton"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:text="Where Am I"
/>
```

下一步，设置 4 个 Texviews，你应当在布局中安排它们，2 个 TextViews 会显示在另外 2 个 TextViews 之上。这样可以把其中的 2 个作为另外 2 个的标签来使用。要完成这个工作，你需要多两个 LinearLayouts.

请注意在 main.xml 文件中的所有元素是包含在一个 LinearLayout 标签中。这个标签用某些规则来绑定内含的元素。对于 LinearLayouts，元素被以依次水平或者垂直的方乡堆栈。

LinerLayout 的方向由 android:orientation 属性管理。假如属性没有被赋值，初始的设定是水平方式。

请注意，有一些槽（或者架子）是垂直放置的。你可以在屏幕上的这些槽上放置元素。总之，如果你要在立式 LinearLayout 布局的同一个架子上放置少量的条目，你需要先在架子上放置一个水平的 LinearLayout。

现在你可以上下左右的码放这些元素了。这是个在本活动中需要利用的概念。因此，在按钮的下面，增加一个水平的 LinearLayout 来放置 2 个 TextViews。

```
<LinearLayout xmlns:android=http://schemas.android.com/apk/res/android
android:layout_width="wrap_content"
android:layout_height="wrap_content"
>
<TextView
android:id="@+id/latLabel"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="Latitude: "
/>
<TextView
android:id="@+id/latText"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
Figure 9-2 Vertical LinearLayout with embedded horizontal LinearLayout
Horizontal LinearLayout
Android screen
/>
```

```
</LinearLayout>
```

这两个 TextViews 保留标签和你将要从 GPS 中采集的纬度数据。下一个，增加另一个水平 LinearLayout 来保留剩下的 TextViews:

```
<LinearLayout xmlns:android=http://schemas.android.com/apk/res/android
android:layout_width="wrap_content"
android:layout_height="wrap_content"
>
<TextView
android:id="@+id/lngLabel"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="Longitude: "
/>
<TextView
android:id="@+id/lngText"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
/>
</LinearLayout>
```

这个为特定的活动提供了比较好的布局。你完成的 main.xml 文件应当如下：

```
<?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"
>
<Button
android:id="@+id/gpsButton"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:text="Where Am I"
/>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
>
<TextView
android:id="@+id/latLabel"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="Latitude: "
```

```
/>
<TextView
android:id="@+id/latText"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
/>
</LinearLayout>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
>
<TextView
android:id="@+id/lngLabel"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="Longitude: "
/>
<TextView
android:id="@+id/lngText"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
/>
</LinearLayout>
</LinearLayout>
```

书写代码来允许活动

书写代码来允许活动 第九章(4)

现在已经创建了布局，你可以开始写代码来允许活动了。你的按钮需要**从 GPS 中来呼叫用户当前的位置**。一旦你有了这些信息，你可以发送纬度和经度坐标到 TextViews 中了。

首先，你需要增加输入声明。你需要输入 4 个包装：

```
import android.view.View;
import android.widget.TextView;
import android.content.Context;
import android.widget.Button;
```

和一个 **Android 位置基础 API**:

```
import android.location.LocationManager;
```

下一步，为按钮创建代码。目标是**从 GPS 中检索当前坐标信息**。你已经在本书中创建了一些按钮了，而且这个的格式没有不同。你需要设置按钮并且从 main.xml

中装载布局。然后你可以设置 onClick 事件来呼叫一个函数，LoadCoords()。

```
final Button gpsButton = (Button) findViewById(R.id.gpsButton);
gpsButton.setOnClickListener(new Button.OnClickListener() {
public void onClick(View v){
LoadCoords();
}});
```

创建活动的最后步骤是填充代码到 LoadCoords()函数中。创建 TextViews 需要接受坐标数据：

```
TextView latText = (TextView) findViewById(R.id.latText);
TextView lngText = (TextView) findViewById(R.id.lngText);
```

注意

你没必要必须创建两个 TextViews 来作为标签，因为你不会发送任何东西到它们。

现在，创建一个可以 pull 坐标数据的 LocationManager。这个示例的重要部分是你必须要传递给 LocationManager 一个上下文；使用 LOCATION_SERVICE:

```
LocationManager myManager =
(LocationManager)getSystemService(Context.LOCATION_SERVICE);
```

要从 myManager 中 pull 坐标，使用 getCurrentLocation()方法。这个方法需要一个参数，一个提供者，它们会展示 API 将要从中 pull 的坐标。在这种情况下，Android 提供了一个本章早些时候讨论过的包含 nmea 文件的模拟位置 GPS：

```
Double latPoint = myManager.getCurrentLocation("gps").getLatitude();
Double lngPoint = myManager.getCurrentLocation("gps").getLongitude();
```

最后，拿去双击数据并且把它们**传递到 TextViews**:

```
latText.setText(latPoint.toString());
lngText.setText(lngPoint.toString());
Your finished code should look like this:
package android_programmers_guide.AndroidLBS;
import android.app.Activity;
import android.os.Bundle;
import android.location.LocationManager;
import android.view.View;
import android.widget.TextView;
import android.content.Context;
import android.widget.Button;
public class AndroidLBS extends Activity {
/** Called when the activity is first created. */
@Override
public void onCreate(Bundle icicle) {
```

```
super.onCreate(icicle);
setContentView(R.layout.main);
final Button gpsButton = (Button) findViewById(R.id.gpsButton);
gpsButton.setOnClickListener(new Button.OnClickListener() {
public void onClick(View v){
LoadCoords();
}});
}
public void LoadCoords(){
TextView latText = (TextView) findViewById(R.id.latText);
TextView lngText = (TextView) findViewById(R.id.lngText);
LocationManager myManager = (LocationManager)
getSystemService(Context.LOCATION_SERVICE);
Double latPoint = myManager.getCurrentLocation("gps").getLatitude();
Double lngPoint = myManager.getCurrentLocation("gps").getLongitude();
latText.setText(latPoint.toString());
lngText.setText(lngPoint.toString());
}
}
```

运行活动

在 **Android 模拟器中运行你的活动**。该活动将会打开如下的屏幕（略）。点击“Where Am I”按钮。你将会看到图片中显示的坐标。

传递坐标到 Google 地图

传递坐标到 Google 地图 第九章(5)

在本节中，你将继续在前一节的基础上构造。对 AndroidLBS 活动的主要修改就是**传递坐标到 Google 地图中**。你将使用 Google 地图来显示用户的当前位置。在 main.xml 文件中的唯一修改指出就是为 **MpaView** 增加一个布局。在目前版本的 Android SDK 中，MapView 被建立为一个类 View。可能在将来的版本中 MapView 会相当于这个布局。

```
<view class="com.google.android.maps.MapView"
android:id="@+id/myMap"
android:layout_width="wrap_content"
android:layout_height="wrap_content"/>
```

完成后的 main.xml 文件应当像这样：

```
<?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"
>
<Button
android:id="@+id/gpsButton"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:text="Where Am I"
/>
<LinearLayout xmlns:android=http://schemas.android.com/apk/res/android
android:layout_width="wrap_content"
android:layout_height="wrap_content"
>
<TextView
android:id="@+id/latLabel"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="Latitude: "
/>
<TextView
android:id="@+id/latText"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
/>
</LinearLayout>
<LinearLayout xmlns:android=http://schemas.android.com/apk/res/android
android:layout_width="wrap_content"
android:layout_height="wrap_content"
>
<TextView
android:id="@+id/lngLabel"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="Longitude: "
/>
<TextView
android:id="@+id/lngText"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
/>
</LinearLayout>
<view class="com.google.android.maps.MapView"
android:id="@+id/myMap"
android:layout_width="wrap_content"
android:layout_height="wrap_content"/>
```

```
</LinearLayout>
```

因为在这个活动中嵌入 MapView，你需要改变类的定义。现在，主要类扩展了活动。但是要正确的使用 Google MapView，你必须扩展 MapActivity。因此，你需要输入 MapActivity 包装并且替换在头部的 Activity 包装。

输入下列包装：

```
import com.google.android.maps.MapActivity;
import com.google.android.maps.MapView;
import com.google.android.maps.Point;
import com.google.android.maps.MapController
```

Point 包装将被用于保留 point 的值，它就是展示地图坐标的，而 MapController 将你的 point 置于地图中央。这两个包装在使用 MapView 时非常的关键。

现在准备增加建立地图并传递坐标的代码。首先，设置一个一个 MapView，并且从 main.xml 文件中把它赋值到布局：

```
MapView myMap = (MapView) findViewById(R.id.myMap);
```

下一步，设置一个 Point 并且把从 GPS 检索的数值赋值给 latPoint 和 IngPoint：

```
Point myLocation = new Point(latPoint.intValue(),lngPoint.intValue());
```

现在，可以创建 MapController 了，它将被用于移动 Google 地图来定位你定义的 Point。从 MapView 使用 getController()方法在定制的地图中建立一个控制器：

```
MapController myMapController = myMap.getController();
```

唯一剩下的工作就是使用控制器来移动地图到你的位置（要让地图更容易辨认，把 zoom 设定为 9）：

```
myMapController.centerMapTo(myLocation, false);
myMapController.zoomTo(9);
```

你刚才所写的所有代码就是从活动中**利用 Google 地图**。完整的类应当像这样：

```
package android_programmers_guide.AndroidLBS;
import android.os.Bundle;
import android.location.LocationManager;
import android.view.View;
import android.widget.TextView;
import android.content.Context;
import android.widget.Button;
import com.google.android.maps.MapActivity;
import com.google.android.maps.MapView;
import com.google.android.maps.Point;
import com.google.android.maps.MapController;
```

```
public class AndroidLBS extends MapActivity {
/** Called when the activity is first created. */
@Override
public void onCreate(Bundle icicle) {
super.onCreate(icicle);
setContentView(R.layout.main);
final Button gpsButton = (Button) findViewById(R.id.gpsButton);
gpsButton.setOnClickListener(new Button.OnClickListener() {
public void onClick(View v){
LoadProviders();
}});
}
public void LoadProviders(){
TextView latText = (TextView) findViewById(R.id.latText);
TextView lngText = (TextView) findViewById(R.id.lngText);
LocationManager myManager = (LocationManager)
getSystemService(Context.LOCATION_SERVICE);
Double latPoint =
myManager.getCurrentLocation("gps").getLatitude()*1E6;
Double lngPoint =
myManager.getCurrentLocation("gps").getLongitude()*1E6;
latText.setText(latPoint.toString());
lngText.setText(lngPoint.toString());
MapView myMap = (MapView) findViewById(R.id.myMap);
Point myLocation = new Point(latPoint.intValue(),lngPoint.intValue());
MapController myMapController = myMap.getController();
myMapController.centerMapTo(myLocation, false);
myMapController.zoomTo(9);
}
}
```

在模拟器中运行活动。活动应当打开一个空白的地图。点击“Where Am I”按钮，应当会看到地图聚焦并且放大到旧金山。看看下图就会知道地图会如何出现（略）。

增加缩放控制

增加缩放控制 第九章(6)

本章的最后一个练习是再增加两个按钮到 AndroidLBS 活动中。这些按钮将**控制 Google MapView 放大和缩小的方法**。让这个修改有一点不同的是我将为 main.xml 文件介绍一个布局的新类型：RelativeLayout。LinearLayouts 允许你直接的一个接一个的放置 Views，RelativeLayouts 允许你在每一个 View 上放置。

对于这个活动，将会放置两个按钮到 Google Map 上，要实现这个效果，你可以

增加在地图上的按钮。

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
android:orientation="vertical"
android:layout_width="fill_parent"
android:layout_height="fill_parent"
>
<view class="com.google.android.maps.MapView"
android:id="@+id/myMap"
android:layout_width="wrap_content"
android:layout_height="wrap_content"/>
</RelativeLayout>
```

现在可以增加另外的两个按钮了。它们会出现在 MapView 的左上方和左下方。你需要对 Button 布局做一个修改。按照默认的方式，RelativeLayout 增加 Button 来和锚视图顶部的边缘排列，本例中，就是这个 MapView。因此，在这个布局，使用 android:layout_alignBottom 属性并赋值 MapView 的 id。这样就排列按钮到地图的底部了。

```
<Button android:id="@+id/buttonZoomIn"
style="?android:attr/buttonStyleSmall"
android:text="+"
android:layout_width="wrap_content"
android:layout_height="wrap_content" />
<Button android:id="@+id/buttonZoomOut"
style="?android:attr/buttonStyleSmall"
android:text="-"
android:layout_alignBottom="@+id/myMap"
android:layout_width="wrap_content"
android:layout_height="wrap_content" />
```

提示

仔细看一下按钮的布局属性。我使用一个新的属性，style，来把这个按钮改小。

完整的 main.xml 应当看上去像这样：

```
<?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"
>
<Button
android:id="@+id/gpsButton"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:text="Where Am I"
```

```
/>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
>
<TextView
android:id="@+id/latLabel"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="Latitude: "
/>
<TextView
android:id="@+id/latText"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
/>
</LinearLayout>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
>
<TextView
android:id="@+id/lngLabel"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="Longitude: "
/>
<TextView
android:id="@+id/lngText"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
/>
</LinearLayout>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
android:orientation="vertical"
android:layout_width="fill_parent"
android:layout_height="fill_parent"
>
<view class="com.google.android.maps.MapView"
android:id="@+id/myMap"
android:layout_width="wrap_content"
android:layout_height="wrap_content"/>
<Button android:id="@+id/buttonZoomIn"
style="?android:attr/buttonStyleSmall"
```

```
android:text="+"
android:layout_width="wrap_content"
android:layout_height="wrap_content" />
<Button android:id="@+id/buttonZoomOut"
style="?android:attr/buttonStyleSmall"
android:text="-"
android:layout_alignBottom="@+id/myMap"
android:layout_width="wrap_content"
android:layout_height="wrap_content" />
</RelativeLayout>
</LinearLayout>
```

你会去适当的修改这个代码。除了为新的 views 增加代码之外，你需要移除现存的一些代码。为了让活动更灵活，你需要移除 MapView 的示例和类主要部分的 MapController。这样将会允许你传递这些项目到其它所需的函数（比如将要创建的放大和缩小特性）。

```
final MapView myMap = (MapView) findViewById(R.id.myMap);
final MapController myMapController = myMap.getController();
```

现在可以创建两个新按钮的代码了。和你之前创建的按钮一样，增加呼叫到下一步将要构建的函数：

```
final Button zoomIn = (Button) findViewById(R.id.buttonZoomIn);
zoomIn.setOnClickListener(new Button.OnClickListener() {
public void onClick(View v){
ZoomIn(myMap,myMapController);
}});
final Button zoomOut = (Button) findViewById(R.id.buttonZoomOut);
zoomOut.setOnClickListener(new Button.OnClickListener() {
public void onClick(View v){
ZoomOut(myMap,myMapController);
}});
```

最后，创建控制放大缩小特性的函数。最大放大位置是 21 并且最小是 1.因此，在函数中，在调整前测试当前的位置。这样将确保不会出现任何的运行问题。

```
public void ZoomIn(MapView mv, MapController mc){
if(mv.getZoomLevel()!=21){
mc.zoomTo(mv.getZoomLevel()+ 1);
}
}
public void ZoomOut(MapView mv, MapController mc){
if(mv.getZoomLevel()!=1){
mc.zoomTo(mv.getZoomLevel()- 1);
}
```

```
}
```

注意你传递 MapView 和 MapController 到函数中，从那里，对缩放位置进行整数处理非常简单。唯一的关键是这个函数是 MapController 本身移动 MapView 到希望的缩放位置，而 MapView 本身保留缩放位置。

提示

想想这个关系和一个遥控器和电视机相类似。遥控器切换电视到第 5 频道，但是频道本身是储存在电视上的。

你完成的 AndroidLBS.java 文件应当看上去像这样：

```
package android_programmers_guide.AndroidLBS;
import android.os.Bundle;
import android.location.LocationManager;
import android.view.View;
import android.widget.TextView;
import android.content.Context;
import android.widget.Button;
import com.google.android.maps.MapActivity;
import com.google.android.maps.MapView;
import com.google.android.maps.Point;
import com.google.android.maps.MapController;
public class AndroidLBS extends MapActivity {
/** Called when the activity is first created. */
@Override
public void onCreate(Bundle icicle) {
super.onCreate(icicle);
setContentView(R.layout.main);
final MapView myMap = (MapView) findViewById(R.id.myMap);
final MapController myMapController = myMap.getController();
final Button zoomIn = (Button) findViewById(R.id.buttonZoomIn);
zoomIn.setOnClickListener(new Button.OnClickListener() {
public void onClick(View v){
ZoomIn(myMap,myMapController);
}});
final Button zoomOut = (Button) findViewById(R.id.buttonZoomOut);
zoomOut.setOnClickListener(new Button.OnClickListener() {
public void onClick(View v){
ZoomOut(myMap,myMapController);
}});
final Button gpsButton = (Button) findViewById(R.id.gpsButton);
gpsButton.setOnClickListener(new Button.OnClickListener() {
public void onClick(View v){
LoadProviders(myMap,myMapController);
```

```
}});
}
public void LoadProviders(MapView mv, MapController mc){
TextView latText = (TextView) findViewById(R.id.latText);
TextView lngText = (TextView) findViewById(R.id.lngText);
LocationManager myManager = (LocationManager)
getSystemService(Context.LOCATION_SERVICE);
Double latPoint = myManager.getCurrentLocation("gps").getLatitude()*1E6;
Double lngPoint =
myManager.getCurrentLocation("gps").getLongitude()*1E6;
latText.setText(latPoint.toString());
lngText.setText(lngPoint.toString());
Point myLocation = new Point(latPoint.intValue(),lngPoint.intValue());
mc.centerMapTo(myLocation, false);
mc.zoomTo(9);
}
public void ZoomIn(MapView mv, MapController mc){
if(mv.getZoomLevel()!=21){
mc.zoomTo(mv.getZoomLevel()+ 1);
}
}
public void ZoomOut(MapView mv, MapController mc){
if(mv.getZoomLevel()!=1){
mc.zoomTo(mv.getZoomLevel()- 1);
}
}
}
```

在 Android 模拟器中运行这个活动。活动会打开一个新的 MapView，如图所示放了按钮（略）。

试一下放大和缩小按钮。当你放大，你应当看到和下面类似的图形（略）。

试试这个：在 MapView 之间转换

试试这个：在 MapView 之间转换 第九章(7)

标准视图和卫星视图再编辑 AndroidLBS 活动一次。再增加两个按钮到 RelativeLayout。这些按钮可以转换**标准视图和卫星视图**。下面是需要考虑的地方：

- 使用排列布局属性使这转换按钮在 MapView 的另外一面。
- 研究 MapView 来找到转换的方式。
- 创建一个可以传递并转换 MapView 的函数。

完成的 main.xml 和 AndroidLBS.java 文件应当如下：

main.xml

```
<?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"
>
<Button
android:id="@+id/gpsButton"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:text="Where Am I"
/>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
>
<TextView
android:id="@+id/latLabel"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="Latitude: "
/>
<TextView
android:id="@+id/latText"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
/>
</LinearLayout>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
>
<TextView
android:id="@+id/lngLabel"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="Longitude: "
/>
<TextView
android:id="@+id/lngText"
android:layout_width="wrap_content"
```

```
android:layout_height="wrap_content"
/>
</LinearLayout>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
android:orientation="vertical"
android:layout_width="fill_parent"
android:layout_height="fill_parent"
>
<view class="com.google.android.maps.MapView"
android:id="@+id/myMap"
android:layout_width="wrap_content"
android:layout_height="wrap_content"/>
<Button android:id="@+id/buttonZoomIn"
style="?android:attr/buttonStyleSmall"
android:text="+"
android:layout_width="wrap_content"
android:layout_height="wrap_content" />
<Button android:id="@+id/buttonMapView"
style="?android:attr/buttonStyleSmall"
android:text="Map"
android:layout_alignRight="@+id/myMap"
android:layout_width="wrap_content"
android:layout_height="wrap_content" />
<Button android:id="@+id/buttonSatView"
style="?android:attr/buttonStyleSmall"
android:text="Sat"
android:layout_alignRight="@+id/myMap"
android:layout_alignBottom="@+id/myMap"
android:layout_width="wrap_content"
android:layout_height="wrap_content" />
<Button android:id="@+id/buttonZoomOut"
style="?android:attr/buttonStyleSmall"
android:text="-"
android:layout_alignBottom="@+id/myMap"
android:layout_width="wrap_content"
android:layout_height="wrap_content" />
</RelativeLayout>
</LinearLayout>
```

AndroidLBS.java

```
package android_programmers_guide.AndroidLBS;
import android.os.Bundle;
import android.location.LocationManager;
import android.view.View;
```

```
import android.widget.TextView;
import android.content.Context;
import android.widget.Button;
import com.google.android.maps.MapActivity;
import com.google.android.maps.MapView;
import com.google.android.maps.Point;
import com.google.android.maps.MapController;
public class AndroidLBS extends MapActivity {
/** Called when the activity is first created. */
@Override
public void onCreate(Bundle icicle) {
super.onCreate(icicle);
setContentView(R.layout.main);
final MapView myMap = (MapView) findViewById(R.id.myMap);
final MapController myMapController = myMap.getController();
final Button zoomIn = (Button) findViewById(R.id.buttonZoomIn);
zoomIn.setOnClickListener(new Button.OnClickListener() {
public void onClick(View v){
ZoomIn(myMap,myMapController);
}});
final Button zoomOut = (Button) findViewById(R.id.buttonZoomOut);
zoomOut.setOnClickListener(new Button.OnClickListener() {
public void onClick(View v){
ZoomOut(myMap,myMapController);
}});
final Button gpsButton = (Button) findViewById(R.id.gpsButton);
gpsButton.setOnClickListener(new Button.OnClickListener() {
public void onClick(View v){
LoadProviders(myMap,myMapController);
}});
final Button viewMap = (Button) findViewById(R.id.buttonMapView);
viewMap.setOnClickListener(new Button.OnClickListener() {
public void onClick(View v){
ShowMap(myMap);
}});
final Button viewSat = (Button) findViewById(R.id.buttonSatView);
viewSat.setOnClickListener(new Button.OnClickListener() {
public void onClick(View v){
ShowSat(myMap);
}});
}
public void LoadProviders(MapView mv, MapController mc){
TextView latText = (TextView) findViewById(R.id.latText);
TextView lngText = (TextView) findViewById(R.id.lngText);
```

```
LocationManager myManager = (LocationManager)
getSystemService(Context.LOCATION_SERVICE);
Double latPoint =
myManager.getCurrentLocation("gps").getLatitude()*1E6;
Double lngPoint =
myManager.getCurrentLocation("gps").getLongitude()*1E6;
latText.setText(latPoint.toString());
lngText.setText(lngPoint.toString());
Point myLocation = new Point(latPoint.intValue(),lngPoint.intValue());
mc.centerMapTo(myLocation, false);
mc.zoomTo(9);
}
public void ZoomIn(MapView mv, MapController mc){
if(mv.getZoomLevel()!=21){
mc.zoomTo(mv.getZoomLevel()+ 1);
}
}
public void ZoomOut(MapView mv, MapController mc){
if(mv.getZoomLevel()!=1){
mc.zoomTo(mv.getZoomLevel()- 1);
}
}
public void ShowMap(MapView mv){
if (mv.isSatellite()){
mv.toggleSatellite();
}
}
public void ShowSat(MapView mv){
if (!mv.isSatellite()){
mv.toggleSatellite();
}
}
}
```

当你运行活动时，应当可以**启动和关闭卫星视图**，如下图（略）。

在下一章，你将进入更深层次的 **Google API**。第十章将一步一步学习使用 Google API 从 Android 手机发送信息到 **GTalk**。

问专家

Q:最终版本的 Android 还会继续使用.kml 或者.nmea 文件吗？

A:本书写的时候，最终的 Android 还没有发布，可以假定的是，是的，最后版本的 **Android** 还会利用.kml 和/或者.nmea 文件。这将允许应用程序开发者在应用程序内使用包括静态坐标文件。

Q:有没有可能来创建有标记的 Google Map?

A:是的，在第十一章，你将学习如何熟练操作 Google 地图 Overlays。这些视图允许在 Google 地图的上面你绘制文本，标记和其它形状。

第十章 使用 Google API 的 Gtalk

使用 Google API 的 GTalk 第十章(1)

关键技能 & 概念

- 执行一个 Google API 包装
- 为 Google 存取配置 XMPP 开发环境设置
- 执行 View.OnClickListener( ) 方法

第九章为你介绍了 Google API。你创建了一个影响 Google API 和 Google 地图的活动。因为 API 的易用和灵活性，可以快速的在 Google 地图上显示用户的当前位置。同样，你还学习到了如何使用很少量的相关代码来熟练操控地图。

Google API 包括了不仅是进入 Google 地图的功能。在上一章中，你使用了很大的 API 中很小的一个部分。对于 Google API 的基本包装是 com.google。从这个基础中，Google API 包含了允许你创建操控 GTalk（Google 的聊天服务），Google 日历，Google 文档，Google 电子表格和 Google 服务等等的活动的权力。

当我看是写这本书时，这个版本的 Android SDK 是 m3-rc22。写完书时，Google 不提倡这些包装中的一些，但是还是留在 SDK 中。有显示 Google 日历，Google 电子表格和 Google 服务仍旧需要升级，遗憾的是，在 m5-rc15 版本的 SDK 中还处于未完成的状态。为了避免混乱，Google 还移除了任何与这些包装相关联的帮助文件。因此，本章的重点是在最新发布的 Android SDK 中工作很好的 GTalk。

在本章中，将会构造一个小的，利用 Android SDK 的 GTalk 包装的活动。当活动完成，你将可以利用手机发送并接受信息到/从另外一个 GTalk 用户。

注意

在第一个 Google API 的反复中，处理 GTalk 的包装是一个非常广泛的 XMPP 包装。（XMPP 是很多聊天平台的基础协议，包括 GTalk 和 Jabber）。使用最新版的 SDK，初始的 XMPP 包装为反映 GTalk 的特性而加强并重命名。要开始，用 Eclipse 创建一个新项目，并且命名 GoogleAPI。

为 GTalk 配置 Android 模拟器

在开始为本项目写代码之前，你需要在 Android 模拟器中调整开发环境设定，XMPP 设定。

在项目打开的状态下，需要离开常规的程序一会儿。如果你熟悉 GTalk，你知道只有登录 Google 帐户以后才可以使用这个产品。因此，你必须要采取特别的步骤来确保你的设备（本例是 Android 模拟器）可以登录你的 Google 帐户，这样，可以确保发送和接受信息。

导航到 AndroidSDK/tools 文件夹并且启动模拟器。你可以从 Eclipse 开发环境中启动它，但是那样会需要同时启动还没有写代码的活动。为了节约时间，手动启动模拟器。

模拟器启动后，点击所有的快捷方式（All shortcut）。找到 **Dev Tools** 条目并且启动它。你将会看到和下面类似的图（略）。

滚动 Dev Tools 菜单直到看见 XMPP 设置。选择 XMPP 设定。

注意

当你打开 XMPP 设定，活动的名称是 GTalk 设定。这个可能是个表象说明 Google 将在 Google API 包装中留下这个包装。命名显而易见的断开可能是从不同 SDK 版本之间改变而剩下的。

活动应当读取账户:<None>，如图所示（略）。这表明了设备中没有储存登录信息。你需要为 Google 帐户增加登录信息来允许活动来有权使用 Google 的服务器。

点击增加帐户来显示一个屏幕。输入用户名和密码之后，点击登录。Android 模拟器应当试着去验证你的信息。当模拟器尝试验证信息时，它显示“Authenticating”信息。

警告

取决于你的连接和你是否有一个调试器和模拟器相连接。你可能会看到“Authenticating”信息一会儿。如果你的帐户几分钟后无法被验证，重新启动模拟器再试试。

一旦你的信息被验证，你应当看到下图（略）。注意，这里没有返回按钮，点击模拟器中的 Home 键返回到主屏幕。

现在模拟器配置好了，并且项目也被设置好，可以开始为活动写代码了。

在 Android 中执行 GTalk

在 Android 中执行 GTalk 第十章(2)

在本节中将使用 **Google API** 来创建一个使用 **GTalk** 的活动。这个活动会从 GTalk 网络发送并接收信息，在屏幕上保存它们，并在通知条中显示。活动可以和其它 GTalk 用户通信，而不管他们是在使用 **Android 手机**或者电脑上的 GTalk。

下一部分将从如何创建应用程序的布局开始。第一步为活动的编码工作就是增加布局到 GoogleAPI.xml 中。

在 GoogleAPI.xml 中创建活动的布局

本活动由一些 Views 组成。需要一个 **ListView** 来显示需要发送的文本信息。同样需要两个 **EditText**，用于接收者的地址和信息，还有一个发送的 Button。

首先设置一个 id 是 messageList 的 ListView，如下所示。在本布局中，将使用一个新的属性，android:scrollbars。设置这个属性为立式会让你可以在信息列表中滚动。

```
<ListView
android:id="@+id/messageList"
android:layout_width="fill_parent"
android:layout_height="0dip"
android:scrollbars="vertical"
android:layout_weight="1"
android:drawSelectorOnTop="false" />
```

把 ListView 放到主要布局标签中。在 ListView 布局的下方，放置一个 EditText 的布局，如下。这个 EditText 保留你会发信息的收信人地址。

```
<EditText
android:id="@+id/messageTo"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:textSize="16sp"
android:minWidth="250dp"
android:scrollHorizontally="true" />
```

这个 EditText 视图应该没有什么特殊之处。

最后，再创建一个水平的布局来保留信息内容——EditText 和发送按钮：

```
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
android:orientation="horizontal"
android:layout_width="fill_parent"
android:layout_height="wrap_content">
<EditText
android:id="@+id/messageText"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:textSize="16sp"
android:minWidth="250dp"
android:scrollHorizontally="true" />
<Button
android:id="@+id/btnSend"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="Send Msg">
</Button>
</LinearLayout>
```

这个布局会以线性的方式放置视图，所以它们相互排成一排。把这个 **LinearLayout** 放置进主要的 LinearLayout。最后，GoogleAPI.xml 文件将会如下：

```
<?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">
<ListView
android:id="@+id/messageList"
android:layout_width="fill_parent"
android:layout_height="0dip"
android:scrollbars="vertical"
android:layout_weight="1"
android:drawSelectorOnTop="false" />
<EditText
android:id="@+id/messageTo"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:textSize="16sp"
android:minWidth="250dp"
android:scrollHorizontally="true" />
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
android:orientation="horizontal"
android:layout_width="fill_parent"
246 Android: A Programmer’s Guide
android:layout_height="wrap_content">
<EditText
android:id="@+id/messageText"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:textSize="16sp"
android:minWidth="250dp"
android:scrollHorizontally="true" />
<Button
android:id="@+id/btnSend"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="Send Msg">
</Button>
</LinearLayout>
</LinearLayout>
```

为 GoogleAPI.java 增加包装

布局文件完成后，有一些新的包装需要增加到 GoogleAPI.java 文件中。第一个

必须导入的包装应该和增加到布局中的 Views 相对应。因此，必须为 EditText，ListView，**ListAdapter**，和按钮导入包装：

```
import android.widget.EditText;
import android.widget.ListView;
import android.widget.ListAdapter;
import android.widget.Button;
```

同样需要导入的是 Google API 和 GTalk 打交道的包装：

```
import com.google.android.gtalkservice.IGTalkSession;
import com.google.android.gtalkservice.IGTalkService;
import com.google.android.gtalkservice.GTalkServiceConstants;
import com.google.android.gtalkservice.IChatSession;
```

其它需要的包装有 intent，ServiceConnection, Color, 和 Im. 完整的列表如下：

```
import android.content.ComponentName;
import android.content.Intent;
import android.content.ServiceConnection;
import android.database.Cursor;
Chapter 10: Using the Google API with GTalk 247
248 Android: A Programmer’s Guide
import android.os.Bundle;
import android.os.DeadObjectException;
import android.os.IBinder;
import android.provider.Im;
import android.graphics.Color;
import android.view.View;
import android.widget.SimpleCursorAdapter;
```

如你所见，本活动所需的包装不多。而且，你会发现，发送和接受信息所需要的代码非常的少。现在，需要执行一个 onClickListnerner 来允许代码。

执行 View.OnClickListener

需要修改 GoogleAPI 的类来执行 **View.OnClickListener**。当任何按钮被点击，这将允许你从活动的主类来呼叫 onClick()方法。通常，这种执行 onClick()方法是有效的：只有当你在一个活动中有一组按钮，并且以一种方式来出处理所有的 onClick 呼叫。但是，我感觉你仍旧需要看看这个方式是如何工作的，那样你就可以在将来用于自己的代码中。记住，展示这个方式是因为在很多情况下，它可以是非常有用的工具

```
public class GoogleAPI extends Activity implements View.OnClickListener {
}
```

在活动中执行常规变量是本书中另外一个未曾谈到的内容。你需要在活动中建立一些常规变量，这样就可以以很多方法来使用：

```
EditText messageText;
ListView messageList;
IGTalkSession myIGTalkSession;
EditText messageTo;
Button sendButton;
```

在 onCreate()方法中，你将执行正常的初始化。应当赋值布局到活动并且设置 IGTalkSession 为 null。同样，在活动中增加点小乐趣，改变 ListView 的背景色为灰色。

```
myIGTalkSession = null;
messageText = (EditText) findViewById(R.id.messageText);
messageList = (ListView) findViewById(R.id.messageList);
messageTo = (EditText) findViewById(R.id.messageTo);
sendButton = (Button) findViewById(R.id.btnSend);
sendButton.setOnClickListener(this);
messageList.setBackgroundColor(Color.GRAY );
```

提示

因为你是从类中执行 View.OnClickListener 的，可以设置 Send Button 的 OnClickListener()方式到其中。最后在 onCreate()方法内执行是绑定你的服务。这个过程创建创建需要使用的连接，由 Google 帐户使用，传递 GTalk 信息:

```
this.bindService(new
Intent().setComponent(GTalkServiceConstants.GTALK_SERVICE_COMPONENT),
connection, 0);
```

在上面的 bindService 声明中，传递到 setComponent()方法的一个参数就是连接。这个变量表示一个 ServiceConnection 执行 onServiceConnected（）和 onServiceDisconnected（）方法。下面的代码构造约束前面 bindService 声明的连接：

```
private ServiceConnection connection = new ServiceConnection() {
public void onServiceConnected(ComponentName name, IBinder service) {
try {
myIGTalkSession =
IGTalkService.Stub.asInterface(service).getDefaultSession();
} catch (DeadObjectException e) {
myIGTalkSession = null;
}
}
public void onServiceDisconnected(ComponentName name) {
myIGTalkSession = null;
}
};
```

在 onServiceConnected（）方法中，你建立了一个使用 IGTalkService.Stub 的

片段。如果这个过程失败，你需要再次把这个片段设为 null。现在，可以为类的 onClick 事件创建代码。在每一个 onClick 事件中你应当执行一些行动：

1. 为任何的信息检查数据库。
2. 从查询的结果中创建一个 ListAdapter 并显示到 ListView 中。
3. 创建一个 ChatSession 到 EditView 中的地址并发送你的信息文本。

注意

Android 服务器包括 SQLite 数据库，你可以使用来保留任何你认为需要放入的活动相关条目和任何的定制数据。这个数据库在第十一章做深入的介绍。

下面的代码为你和接受者之间发生的信息**查询数据库**：

```
Cursor cursor = managedQuery(Im.Messages.CONTENT_URI, null,
"contact=\'" + messageTo.getText().toString() + "\'", null, null);
```

使用下面的代码来从查询结果创建一个 ListAdapter 并且赋值接收器到 ListView。在前一个活动，你已经使用了一个类似的过程，所以，对你应该不陌生。

```
ListAdapter adapter = new SimpleCursorAdapter(this,
android.R.layout.simple_list_item_1, cursor,
new String[]{Im.MessagesColumns.BODY},
new int[]{android.R.id.text1});
this.messageList.setAdapter(adapter);
```

信息可以显示了，最后的步骤是发送你的信息。下面的代码用定义的 messageTO address 创建一个 IchatSession。这个信息文本然后被从这里传递到接受者。

```
try {
IChatSession chatSession;
chatSession =
myIGTalkSession.createChatSession(messageTo.getText().toString(););
chatSession.sendTextMessage(messageText.getText().toString());
} catch (DeadObjectException ex) {
myIGTalkSession = null;
}
```

所有的东西在一起，完成后的 GoogleAPI.java 文件如下：

```
package android_programmers_guide.GoogleAPI;
import android.app.Activity;
import android.content.ComponentName;
import android.content.Intent;
import android.content.ServiceConnection;
import android.database.Cursor;
import android.os.Bundle;
```

```
import android.os.DeadObjectException;
import android.os.IBinder;
import android.provider.Im;
import android.graphics.Color;
import android.view.View;
import android.widget.EditText;
import android.widget.ListView;
import android.widget.ListAdapter;
import android.widget.Button;
import android.widget.SimpleCursorAdapter;
import com.google.android.gtalkservice.IGTalkSession;
import com.google.android.gtalkservice.IGTalkService;
import com.google.android.gtalkservice.GTalkServiceConstants;
import com.google.android.gtalkservice.IChatSession;
public class GoogleAPI extends Activity implements View.OnClickListener {
EditText messageText;
ListView messageList;
IGTalkSession myIGTalkSession;
EditText messageTo;
Button mSend;
@Override
public void onCreate(Bundle icicle) {
super.onCreate(icicle);
setContentView(R.layout.main);
myIGTalkSession = null;
messageText = (EditText) findViewById(R.id.messageText);
messageList = (ListView) findViewById(R.id.messageList);
messageTo = (EditText) findViewById(R.id.messageTo);
mSend = (Button) findViewById(R.id.btnSend);
mSend.setOnClickListener(this);
messageList.setBackgroundColor(Color.GRAY );
Chapter 10: Using the Google API with GTalk 251
this.bindService(new
Intent().setComponent(GTalkServiceConstants.GTALK_SERVICE_COMPONENT),
connection, 0);
}
private ServiceConnection connection = new ServiceConnection() {
public void onServiceConnected(ComponentName name, IBinder service) {
try {
myIGTalkSession =
IGTalkService.Stub.asInterface(service).getDefaultSession();
} catch (DeadObjectException e) {
myIGTalkSession = null;
}
```

```
}
public void onServiceDisconnected(ComponentName name) {
myIGTalkSession = null;
}
};
public void onClick(View view) {
Cursor cursor = managedQuery(Im.Messages.CONTENT_URI, null,
"contact=\'" + messageTo.getText().toString() + "\'",
null, null);
ListAdapter adapter = new SimpleCursorAdapter(this,
android.R.layout.simple_list_item_1, cursor,
new String[]{Im.MessagesColumns.BODY},
new int[]{android.R.id.text1});
this.messageList.setAdapter(adapter);
try {
IChatSession chatSession;
chatSession =
myIGTalkSession.createChatSession(messageTo.getText().toString());
chatSession.sendTextMessage(messageText.getText().toString());
} catch (DeadObjectException ex) {
myIGTalkSession = null;
}
}
}
```

编译并运行 GoogleAPI

编译并运行 GoogleAPI 第十章(3)

现在，在模拟器中运行 GoogleAPI。如何你的连接成功，你应当能看见下面的屏幕。

要测试活动，我发送“hello”信息给 androidprogrammersguide@gmail.com，显示如下：

下一个插图，你将看到点击发送按钮后，移动我发送的信息到 ListView。

当我作为 androidprogrammersguide 登录，我发现信息确实到了预想的收件人，如下：

我回复“Greetings!”，要查看这个，看下面的两个图。注意活动屏幕顶部的信息条。紧接的图，你可以看到发送者的信息被显示。

下一章，将创建最后一个应用程序，你会在 Google 地图上使用 SQLite 数据库和 Google 地图 Overlays 来绘制数据记录。这些是非常有力的技术来提升 Android 超越其它的移动操作系统。

试试这个：为 GoogleAPI 活动增加设置特性

试试这个：为 GoogleAPI 活动增加设置特性 第十章(4)

编辑 GoogleAPI 活动来包括一个设置特性。使用第八章的 AndroidViews 活动作为向导，增加一个可以改变应用程序布局属性按钮到 GoogleAPI 活动。这里是一些提示关于如何设置按钮：

- 改变信息列表的字体
- 改变信息列表的字体颜色，使得发送和接受相反。
- 改变信息列表的背景色。

问专家

Q:GTalk API 可用于其它基于 XMPP 的聊天客户端吗？

A: 对于这个的回答还不清晰。m3-rc22 版本的 SDK 包括一个 XMPP API 而 M5-15 SDK 只含了 GTalk API。有可能这两者在将来发布的 Android SDK 绑在一起。那样，GTalk API 可以被用来与其它基于 XMPP 聊天客户端的程序通信。

第十一章 应用程序：找一个朋友

应用程序：找一个朋友 第十一章(1)

关键技能 & 概念

- 创建一个 SQLite 数据库
- 创建一个定制内容提供者
- 从数据库检索条目并且传递到一个 Google Maps Overlay

这是你将创建应用程序的最后一章，但是会是本书介绍的最大的一个应用程序。我将介绍一些到目前为止没有遇到过的话题，而且你会用到这些话题所谈到的技能创建一个非常健全的应用程序。

在本章中，会学习如何在 Android 模拟器中创建 SQLite 数据库。我会向你展示如何在定制的数据库中读取，写入并且输出数据。这个过程包括创建并使用你自己的 **Content Provider** 来和数据库一同工作。然后，你拿取存储在数据库中的数据并写入到 **Google Maps Overlay** 中。当你在前一章用 Google 地图时，还没有使用 Overlay。使用 Google Maps Overlays 在地图上绘制形状并且写文本，得到一个有信息量的地图。在这个项目中，将创建一个两部分应用程序。应用程序的第一部分将允许用户输入“friends”到移动数据库中。（一个 friend 由姓名和地理坐标位置组成）。用户将能增加，修改并且删除 friends.

第二个部分将包括一个菜单项目。当用户选择这个菜单项目，**应用程序将显示一个 Google 地图**。这个 Google 地图和第九章创建的 Google 地图不同之处就是，这个地图会包含一个 Google Maps Overlay，它会允许你在 Google 地图的标题处写入姓名，给定信息并且绘制项目。

要开始，在 Eclipse 内创建一个新的 Android 项目，命名为 FindFriend，使用下图的设定（略）。

现在，你应该对创建一个 Android 应用程序非常的熟悉了，创建本项目会需要一点小小的帮助。Google 在 **Android SDK** 里有一个应用程序叫 **NotePad**，非常简单但是允许你储存，修改并且删除数据库里的“notes”。你会去修改这个例子的一些代码来创建 Friends 数据库。

如果你想要知道 Google NotePad 如何工作，在继续之前，在 Eclipse 中装载项目并且在模拟器中运行它。不久将会开始修改这个代码，但是首先，在下一节里，将**创建你的第一个 SQLite 数据库**。

创建一个 SQLite 数据库

创建一个 SQLite 数据库 第十一章(2)

Android 设备将发布时会有一个内部的 **SQLite 数据库**。这个数据库的目的是允许用户和开发者一个可以在活动中储存信息的地方。

如果你用过 **Microsoft SQL 服务器**或者 SQLite，使用 Andorid 的 SQLite 数据库的结构和过程对你将不会陌生。不管你有多少的经验，这个部分将涵盖所有需要创建和使用全功能 SQlite 数据库的技能。你将要在 Android 模拟器上创建一个数据库。要实现这个，你需要**进入 Android SDK 命令行编辑器工具**并使用 shell 命令来进入 Android 服务器。

提示

参考第三章来重拾你的记忆**关于路径声明和使用命令行工具**。

一旦你进入服务器，你需要导航到数据库的位置。所有的 Android SQLite 数据库的位置是在 data/data/<package>/

databases 目录。使用 cd 命令来从当前的目录改变到 data 目录，并且再到<package>目录。如果你不确定<package>目录的名称，使用 ls 来列出文件和当前目录。改变目录至<package>android_programmers_
guide.FindAFriend，如下所示（略）

警告

如果你没有 android_programmers_guide.FindAFriend 目录，按照前一部分描述的方式创建你的应用程序并且运行“Hello World!”默认的由项目创建的应用程序，那样会确保你有个正确的目录。

找到 android_programmers_guide.FindAFriend 目录后，运行 Is 命令。这个命令列出特定文件夹内所有的文件和目录。改命令应当返回空的内容。因为，此时在该目录内没有文件和文件夹。

假定 SQLite 数据库必须在本目录下的一个数据库目录内，是时候来创建一个了。**mkdir 工具为你创建目录**。因此，运行 mkdir databases 命令。它将创建保留数据库的目录。

警告

现在，你几乎是在服务器的根目录上。因此你刚刚创建的目录将被作为根目录进入。当你运行活动时，可能会出问题，因为每一个活动有一个不同的用户。出于开发的目的，要解决这个问题，**运行 chmod 777 databases** 来准许每个人都能进入到数据库目录。将来，你必须对给予每个人的权力到一些敏感的 Android 条目非常谨慎才行。只给予特定的用户需要使用特定条目的权力。

已经创建了数据库目录了，可以创建数据库了。**使用 cd 命令导航到数据库目录**。在数据库目录后，**使用 sqlite3 工具来创建数据库**并命名它为 friends.db，如下：

```
# sqlite friends.db
```

如果执行命令成功，你应当能看到一个 SQLite3 版本信息，本例是 3.5.0，和一个 SQLite3 prompt—sqlite>。这说明数据库已被建立但是是空的。数据库没有包含表格和数据。记住，下一步是为活动数据创建一个表格。

你需要创建一个名为 friends 的表格。这个表将保留 id,name,location,created, 和 modified 字段。这些字段将为你的项目提供足够的信息。

提示

如果你对 SQLite 不熟悉，**一个 SQLite 命令必须以分号结束**。如果你想要跨越一个命令这个会有帮助。没有终止 SQLite 命令的情况下，按下 ENTER 键会继续给你一个提示符，…>。你不能在提示符继续输入命令，除非你使用分号。一旦分号被使用，SQLite 将把连续的命令作为一个完整的命令。

要在数据库内创建 friends 表格，在 sqlite>提示符输入下列命令：

```
CREATE TABLE friends (_id INTEGER PRIMARY KEY, name TEXT, location TEXT,
created INTEGER, modified INTEGER);
```

如果命令执行成功，将返回到 sqlite>提示符，如下图所示（略）。

数据库现在可以被使用了，你可以退出 SQLite 了。**使用.exit 来退出**。然后可以退出 shell 部分返回到 Eclipse.

创建数据库是创建应用程序的第一步。现在数据库和相应的表格已经被创建，你需要一个方法来存储数据。受雇 Android 数据的存储方式是一个 **Content Provider**。下面的部分带你走进如何**为新数据库创建一个定制 Content Provider** 并存储数据。

创建一个定制的 Content Provider

创建一个定制的 Content Provider 第十一章(3)

Android 使用 Content Provider 来存储数据。你在第九章使用过 Content Provider 存储并从一个 GPS 中读取坐标信息。同样的过程应用于数据库。有这样一些 Content Provider Contact Lists，IMs，和 Recent Calls。总之，没有为你数据库准备的 Content Provider。Android 是非常之灵活并且允许你为定制的数据创建定制的 Content Provider。在本节，将创建一个和 Friends 数据工作的 Content Provider。这是存取 friend 数据和最终在屏幕上显示它们的关键所在。

下一节，让我们来编辑 string.xml 文件。这个文件保留全局遍及活动的字符串内容。

编辑 string.xml 文件

首先，为项目编辑 string.xml 文件。string.xml 文件由每个项目创建但是你还没有使用过它。这个文件保留可以在活动中使用的静态字符串。

通常，你不大可能在写这些字符串之前仔细查看所有的部分。那就是，当你构造活动时，你通常增加到 string.xml 的入口。因为那样会打断本书的流程，所以我预先给你所有 string.xml 文件的所有内容。编辑 string.xml 文件使它看上去像这样：

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
<string name="app_name">FindAFriend</string>
<string name="menu_delete">Delete</string>
<string name="menu_insert">Add Friend</string>
<string name="find_friends">Find Friends</string>
<string name="menu_revert">Revert</string>
<string name="menu_discard">Discard</string>
<string name="resolve_edit">Edit location</string>
<string name="resolve_title">Edit name</string>
<string name="title_create">Create Friend</string>
<string name="title_edit">Edit Friend</string>
```

```
<string name="title_notes_list">Friends</string>
<string name="title_note">Location</string>
<string name="title_edit_title">Friend Name:</string>
<string name="button_ok">OK</string>
<string name="error_title">Error</string>
<string name="error_message">Error loading note</string>
</resources>
```

完成 string.xml 文件后，需要创建一个.java 文件来保留你的代码。应该命名这个文件为 FriendsProvider.java。还要创建另外一个.java 文件来保留数据定义。命名这个文件为 Friends.java，因为它会定义一个 Friends 数据结构像什么样子并且允许你的 Content Provider 正常进入。（因为 provider 将会是项目中的一个类，没有必要来构建一个相应的.xml 布局文件）。

提示

技术上，对于应用程序，你定制的 Content Provider 不需要定居在同一项目或包装内作为代码的剩余部分。为了简单明了，我在 FindAFriend 项目里做了一个类。假如你计划创建一个可能用户多重项目的 Content Provider，在单独的包装中创建它吧。这样，当你只想要使用 Content Provider 时，会允许你呼叫一个包装。

让我们开始 Friends.java 文件。你只要为相关的类需要输入两个包装：

```
import android.net.Uri;
import android.provider.BaseColumns;
```

BaseColumns 将被一个从主 Friends 类中的 subclass 执行。命名这个 subclass 为 Friend，因为它代表 Friends 数据集中的一个 friend。下面的代码显示如何设置类的概要：

```
public final class Friends {
public static final class Friend implements BaseColumns {
}
}
```

这个类将保留一下静态变量，它们定义 Friends 数据库中的每一列，Content URI，和记录的默认排序。

提示

Content URI 被用于识别将要处理的内容。这个数值必须唯一。

需要定义的字符串如下：

```
public static final Uri CONTENT_URI
=
Uri.parse("content://android_programmers_guide.FindAFriend.Friends/friend");
public static final String DEFAULT_SORT_ORDER = "modified DESC";
public static final String NAME = "name";
```

```
public static final String LOCATION = "location";
public static final String CREATED_DATE = "created";
public static final String MODIFIED_DATE = "modified";
```

有了变量的设定以后，Friends 类放在一起应该是这样的：

```
package android_programmers_guide.FindAFriend;
import android.net.Uri;
import android.provider.BaseColumns;
public final class Friends {
public static final class Friend implements BaseColumns {
public static final Uri CONTENT_URI
=
Uri.parse("content://android_programmers_guide.FindAFriend.Friends/friend");
public static final String DEFAULT_SORT_ORDER = "modified DESC";
public static final String NAME = "name";
public static final String LOCATION = "location";
public static final String CREATED_DATE = "created";
public static final String MODIFIED_DATE = "modified";
}
}
```

创建 Content Provider

创建 Content Provider 第十一章(4)

使用 Eclipse 打开将会成为项目 **Content Provider** 的 FriendsProvider.java 文件。你将要在活动中使用这个定制的 Content Provider 来从 Friends 数据库中检索数据。

和往常一样。让我们从导入文件开始。你需要输入 Friends 类和一些其它的类：

```
import android_programmers_guide.FindAFriend.Friends;
import android.content.*;
import android.database.Cursor;
import android.database.SQLException;
import android.database.sqlite.SQLiteOpenHelper;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteQueryBuilder;
import android.net.Uri;
import android.text.TextUtils;
import android.util.Log;
import java.util.HashMap;
```

如你所见，你输入类的大多数和 **SQL** 打交道。当你用这些包装的时候，我再向你解释。

第一个将要使用的包装是 android.content。要成为一个 Content Provider，需要利用并优先要求的方法，你的 FriendsProvider 类需要扩展 ContentProvider。看一下下面类的概要，它们包括一些将要使用在 Provider 的变量定义：

```
public class FriendsProvider extends ContentProvider {
private SQLiteDatabase mDB;
private static final String TAG = "FriendsProvider";
private static final String DATABASE_NAME = "friends";
private static final int DATABASE_VERSION = 2;
private static HashMap<String, String> FRIENDS_PROJECTION_MAP;
private static final int FRIENDS = 1;
private static final int FRIENDS_ID = 2;
private static final UriMatcher URL_MATCHER;}
```

Content Provider 包含一些需要优先的方法，包括 **onCreate( )**, **query( )**, **insert( )**, **delete( )**, 和 **update( )**。因为这些方法将被活动使用 Content Provider 呼叫，你必须优先使用它们来进入 Friends 数据库。

你将优先 onCreate( ) 方法呼叫一个 SQLiteOpenHelper。因此，在能优先 ContentProvider 的 onCreate() 之前，你不得不创建一个类扩展 SQLiteIpenHelper。

代码块跟从的是一个 Content Provider 的子类。它扩展 SQLiteOpenHelper：

```
private static class DatabaseHelper extends SQLiteOpenHelper {
@Override
public void onCreate(SQLiteDatabase db) {
db.execSQL("CREATE TABLE friends (_id INTEGER PRIMARY KEY,"
+ "name TEXT," + "location TEXT," + "created INTEGER,"
+ "modified INTEGER" + ");");
}
@Override
public void onUpgrade(SQLiteDatabase db, int oldVersion, int
newVersion) {
Log.w(TAG, "Upgrading database from version " + oldVersion + "to "
+ newVersion + ", which will destroy all old data");
db.execSQL("DROP TABLE IF EXISTS friends");
onCreate(db);
}
}
```

刚创建的 DatabaseHelper 类包含两个优先方法：onCreater()和 onUpgrade()。onCreate()方法被用于当从代码中创建数据库，或者表格定义不存在的示例中。

注意

假定你从 adb 壳中**创建数据库结构**，你不会依赖于 DatabaseHelper 的

onCreate()方法来建立你的数据库。

DatabaseHelper 类创建后，你现在可以为 Content Provider 优先 onCreate()方法了：

```
@Override
public boolean onCreate() {
DatabaseHelper dbHelper = new DatabaseHelper();
mDB = dbHelper.openDatabase(getContext(), DATABASE_NAME, null,
DATABASE_VERSION);
return (mDB == null) ? false : true;
}
```

这是个非常之简单的方法，结果就是返回一个布尔值代表你的数据库是否可以被打开。你使用在兄弟类中创建的 SQLiteOpenHelper 来打开 Friends 数据库。注意你传递数据库名称到 DatabaseHelper 类。如果数据库对象——mDB 返回的不是 null，然后数据库就成功的被打开并可以查询了。

下一步，优先 ContentProvider 类的 query()方法。这将是 Content Provider 的主要部分。query()方法是从活动通过 Content Provider 被呼叫来从数据库中获得记录。看下优先版本的 query()方法代码：

```
@Override
public Cursor query(Uri url, String[] projection, String selection,
String[] selectionArgs, String sort) {
SQLiteQueryBuilder qb = new SQLiteQueryBuilder();
switch (URL_MATCHER.match(url)) {
case FRIENDS:
qb.setTables("friends");
qb.setProjectionMap(FRIENDS_PROJECTION_MAP);
break;
case FRIENDS_ID:
qb.setTables("friends");
qb.appendWhere("_id=" + url.getPathSegments().get(1));
break;
default:
throw new IllegalArgumentException("Unknown URL " + url);
}
String orderBy;
if (TextUtils.isEmpty(sort)) {
orderBy = Friends.Friend.DEFAULT_SORT_ORDER;
} else {
orderBy = sort;
}
Cursor c = qb.query(mDB, projection, selection, selectionArgs, null,
```

```
null, orderBy);
c.setNotificationUri(getContext().getContentResolver(), url);
return c;
}
```

query()方法做了一点家务管理之类的事宜，通过检查传递到其中的数据库URL的有效性和定义一个查询分类序列达到的。**URL检查是为了确保你只是要进入Friends数据库**。如果你试图从其它活动进入数据库，或者从其它的Content Provider，query()方法投递一个例外。

到方法的结尾，你使用SQLiteQueryBuilder来执行一个查询。通过下面的代码，导致的数据集被赋值到一个光标：

```
Cursor c = qb.query(mDB, projection, selection, selectionArgs, null,
null, orderBy);
```

注意

光标是一个设备允许你移动记录并从数据列中返回信息。

update( )，delete( )， 和insert( ) 方法同样的简单。看一下这三个方法，应当优先它们：

```
@Override
public Uri insert(Uri url, ContentValues initialValues) {
long rowID;
ContentValues values;
if (initialValues != null) {
values = new ContentValues(initialValues);
} else {
values = new ContentValues();
}
if (URL_MATCHER.match(url) != FRIENDS) {
throw new IllegalArgumentException("Unknown URL " + url);
}
Long now = Long.valueOf(System.currentTimeMillis());
Resources r = Resources.getSystem();
if (values.containsKey(Friends.Friend.CREATED_DATE ) == false) {
values.put(Friends.Friend.CREATED_DATE, now);
}
if (values.containsKey(Friends.Friend.MODIFIED_DATE) == false) {
values.put(Friends.Friend.MODIFIED_DATE, now);
}
if (values.containsKey(Friends.Friend.NAME) == false) {
values.put(Friends.Friend.NAME,
r.getString(android.R.string.untitled));
```

```
}
if (values.containsKey(Friends.Friend.LOCATION) == false) {
values.put(Friends.Friend.LOCATION , "");
}
270 Android: A Programmer' s Guide
rowID = mDB.insert("friends", "friend", values);
if (rowID > 0) {
Uri uri = ContentUris.withAppendedId(Friends.Friend.CONTENT_URI
, rowID);
getContext().getContentResolver().notifyChange(uri, null);
return uri;
}
throw new SQLException("Failed to insert row into " + url);
}
@Override
public int delete(Uri url, String where, String[] whereArgs) {
int count;
long rowId = 0;
switch (URL_MATCHER.match(url)) {
case FRIENDS:
count = mDB.delete("friends", where, whereArgs);
break;
case FRIENDS_ID:
String segment = url.getPathSegments().get(1);
rowId = Long.parseLong(segment);
count = mDB
.delete("friends", "_id="
+ segment
+ (!TextUtils.isEmpty(where) ? " AND (" + where
+ ')' : ""), whereArgs);
break;
default:
throw new IllegalArgumentException("Unknown URL " + url);
}
getContext().getContentResolver().notifyChange(url, null);
return count;
}
@Override
public int update(Uri url, ContentValues values, String where, String[]
whereArgs) {
int count;
switch (URL_MATCHER.match(url)) {
case FRIENDS:
count = mDB.update("friends", values, where, whereArgs);
```

```
break;
case FRIENDS_ID:
String segment = url.getPathSegments().get(1);
count = mDB
.update("friends", values, "_id=" + segment
Chapter 11: Application: Find a Friend 271
+ (!TextUtils.isEmpty(where) ? " AND (" + where + ')' : ""), whereArgs);
break;
default:
throw new IllegalArgumentException("Unknown URL " + url);
}
getContext().getContentResolver().notifyChange(url, null);
return count;
}
```

这些方法的代码应当不需要再加以说明了。如果看过在每个方法的处理，代码的核心就是发出一个数据库声明来执行要求的动作——更新，删除，或者插入。

最后的 Content Provider 的部分就是一个 **getType()** 方法，它返回 Friends 数据类型。当创建自己的类型，应当跟从一下协议：

```
vnd.android.cursor.dir/vnd.<package>
Take a look at the getType( ) method:
@Override
public String getType(Uri url) {
switch (URL_MATCHER.match(url)) {
case FRIENDS:
return
"vnd.android.cursor.dir/vnd.android_programmers_guide.friend";
case FRIENDS_ID:
return
"vnd.android.cursor.item/vnd.android_programmers_guide.friend";
default:
throw new IllegalArgumentException("Unknown URL " + url);
}
}
```

这样就完成了新的定制的 Content Provider。看一下完成的 FriendsProvider 代码：

```
package android_programmers_guide.FindAFriend;
import android_programmers_guide.FindAFriend.Friends;
import android.content.*;
import android.database.Cursor;
import android.database.SQLException;
import android.database.sqlite.SQLiteOpenHelper;
```

```
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteQueryBuilder;
import android.net.Uri;
import android.text.TextUtils;
import android.util.Log;
import java.util.HashMap;
public class FriendsProvider extends ContentProvider {
private SQLiteDatabase mDB;
private static final String TAG = "FriendsProvider";
private static final String DATABASE_NAME = "friends";
private static final int DATABASE_VERSION = 2;
private static HashMap<String, String> FRIENDS_PROJECTION_MAP;
private static final int FRIENDS = 1;
private static final int FRIENDS_ID = 2;
private static final UriMatcher URL_MATCHER;
private static class DatabaseHelper extends SQLiteOpenHelper {
@Override
public void onCreate(SQLiteDatabase db) {
db.execSQL("CREATE TABLE friends (_id INTEGER PRIMARY KEY,"
+ "name TEXT," + "location TEXT," + "created INTEGER,"
+ "modified INTEGER" + ");");
}
@Override
public void onUpgrade(SQLiteDatabase db, int oldVersion, int
newVersion) {
Log.w(TAG, "Upgrading database from version " + oldVersion + "to "
+ newVersion + ", which will destroy all old data");
db.execSQL("DROP TABLE IF EXISTS friends");
onCreate(db);
}
}
@Override
public boolean onCreate() {
DatabaseHelper dbHelper = new DatabaseHelper();
mDB = dbHelper.openDatabase(getContext(), DATABASE_NAME, null,
DATABASE_VERSION);
return (mDB == null) ? false : true;
}
@Override
public Cursor query(Uri url, String[] projection, String selection,
String[] selectionArgs, String sort) {
SQLiteQueryBuilder qb = new SQLiteQueryBuilder();
switch (URL_MATCHER.match(url)) {
case FRIENDS:
```

```
qb.setTables("friends");
qb.setProjectionMap(FRIENDS_PROJECTION_MAP);
break;
case FRIENDS_ID:
qb.setTables("friends");
qb.appendWhere("_id=" + url.getPathSegments().get(1));
break;
default:
throw new IllegalArgumentException("Unknown URL " + url);
}
String orderBy;
if (TextUtils.isEmpty(sort)) {
orderBy = Friends.Friend.DEFAULT_SORT_ORDER;
} else {
orderBy = sort;
}
Cursor c = qb.query(mDB, projection, selection, selectionArgs, null,
null, orderBy);
c.setNotificationUri(getContext().getContentResolver(), url);
return c;
}
@Override
public String getType(Uri url) {
switch (URL_MATCHER.match(url)) {
case FRIENDS:
return
"vnd.android.cursor.dir/vnd.android_programmers_guide.friend";
case FRIENDS_ID:
return
"vnd.android.cursor.item/vnd.android_programmers_guide.friend";
default:
throw new IllegalArgumentException("Unknown URL " + url);
}
}
@Override
public Uri insert(Uri url, ContentValues initialValues) {
long rowID;
ContentValues values;
if (initialValues != null) {
values = new ContentValues(initialValues);
} else {
values = new ContentValues();
}
if (URL_MATCHER.match(url) != FRIENDS) {
```

```
throw new IllegalArgumentException("Unknown URL " + url);
}
Long now = Long.valueOf(System.currentTimeMillis());
Resources r = Resources.getSystem();
if (values.containsKey(Friends.Friend.CREATED_DATE ) == false) {
values.put(Friends.Friend.CREATED_DATE, now);
}
if (values.containsKey(Friends.Friend.MODIFIED_DATE) == false) {
values.put(Friends.Friend.MODIFIED_DATE, now);
}
if (values.containsKey(Friends.Friend.NAME) == false) {
values.put(Friends.Friend.NAME,
r.getString(android.R.string.untitled));
}
if (values.containsKey(Friends.Friend.LOCATION) == false) {
values.put(Friends.Friend.LOCATION , "");
}
rowID = mDB.insert("friends", "friend", values);
if (rowID > 0) {
Uri uri = ContentUris.withAppendedId(Friends.Friend.CONTENT_URI
, rowID);
getContext().getContentResolver().notifyChange(uri, null);
return uri;
}
throw new SQLException("Failed to insert row into " + url);
}
@Override
public int delete(Uri url, String where, String[] whereArgs) {
int count;
long rowId = 0;
switch (URL_MATCHER.match(url)) {
case FRIENDS:
Chapter 11: Application: Find a Friend 275
count = mDB.delete("friends", where, whereArgs);
break;
case FRIENDS_ID:
String segment = url.getPathSegments().get(1);
rowId = Long.parseLong(segment);
count = mDB
.delete("friends", "_id="
+ segment
+ (!TextUtils.isEmpty(where) ? " AND (" + where
+ ')' : ""), whereArgs);
break;
```

```
default:
throw new IllegalArgumentException("Unknown URL " + url);
}
getContext().getContentResolver().notifyChange(url, null);
return count;
}
@Override
public int update(Uri url, ContentValues values, String where, String[]
whereArgs) {
int count;
switch (URL_MATCHER.match(url)) {
case FRIENDS:
count = mDB.update("friends", values, where, whereArgs);
break;
case FRIENDS_ID:
String segment = url.getPathSegments().get(1);
count = mDB
.update("friends", values, "_id=" + segment
+ (!TextUtils.isEmpty(where) ? " AND (" + where + ')' : ""), whereArgs);
break;
default:
throw new IllegalArgumentException("Unknown URL " + url);
}
getContext().getContentResolver().notifyChange(url, null);
return count;
}
static {
URL_MATCHER = new UriMatcher(UriMatcher.NO_MATCH);
URL_MATCHER.addURI("android_programmers_guide.FindAFriend.Friends",
"friend", FRIENDS);
URL_MATCHER.addURI("android_programmers_guide.FindAFriend.Friends",
"friend/#", FRIENDS_ID);
FRIENDS_PROJECTION_MAP = new HashMap<String, String>();
FRIENDS_PROJECTION_MAP.put(Friends.Friend._ID, "_id");
FRIENDS_PROJECTION_MAP.put(Friends.Friend.NAME, "name");
FRIENDS_PROJECTION_MAP.put(Friends.Friend.LOCATION, "location");
FRIENDS_PROJECTION_MAP.put(Friends.Friend.CREATED_DATE, "created");
FRIENDS_PROJECTION_MAP.put(Friends.Friend.MODIFIED_DATE,
"modified");
}
}
```

有了最根本的数据素材（数据库，定义和 Content Provider），你可以开始来建

造周围的活动。记住，活动将使用数据库内的数据，显示到列表，并允许用户启动另一个活动来放置数据库条目到一个 Google Maps Overlay。在下一节，将构建活动并完成 FindFriend 应用程序。

创建 FindAFriend 活动

创建 FindAFriend 活动 第十一章(5)

如果你花了些时间**运行 Google 的 NotePad 示例**，那么你就对活动的布局非常熟悉了。你将修改 NotePad 的接口来使用 Friends 数据库和 Google 地图。FindAFriend 活动将和一些小的活动交互：**NameEditor**, **LocationEditor**, 和 FriendsMap。下面的章节中你将构建所有的这些活动。

注意

除了 NotePad，**Google 提供了一些写的非常好的示例活动**，展示了多编程状态的基本技术。

如你所做的过去的几个活动，从文件 AndroidManifest.xml 开始。要想熟悉复杂的应用程序，你需要多次更改 AndroidManifest.xml 文件。

编辑 AndroidManifest.xml

看看下面为 FindAFriend 项目准备的 AndroidManifest.xml 文件。需要为新活动增加一些 Intent 过滤器，包括一个编辑 friend 的姓名，并且启动你的 Google 地图。

同样，需要仔细注意每个 Intent 过滤器的动作。这些动作会被传递到每个活动并处理 Intent。最后，不要忘记增加 Access_Location and Access_GPS 许可，这样就可以增加你的当前闻之了。完整的 AndroidManifest.xml 文件应当像这样显示：

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
package="android_programmers_guide.FindAFriend">
<application android:icon="@drawable/icon">
<provider android:name="FriendsProvider"
android:authorities="android_programmers_guide.FindAFriend.Friends" />
<activity android:name=".FindAFriend"
android:label="@string/app_name">
<intent-filter>
<action android:name="android.intent.action.MAIN" />
<category android:name="android.intent.category.LAUNCHER" />
</intent-filter>
<intent-filter>
<action android:name="android.intent.action.VIEW" />
<action android:name="android.intent.action.EDIT" />
<action android:name="android.intent.action.PICK" />
```

```
<category android:name="android.intent.category.DEFAULT" />
<dataandroid:mimeType="vnd.android.cursor.dir/
vnd.android_programmers_guide.friend" />
</intent-filter>
<intent-filter>
<action android:name="android.intent.action.GET_CONTENT" />
<category android:name="android.intent.category.DEFAULT" />
<dataandroid:mimeType="vnd.android.cursor.item/
vnd.android_programmers_guide.friend" />
</intent-filter>
</activity>
<activity android:name=".FriendsMap" android:label="FriendsMap">
<intent-filter>
<action android:name="android.intent.action.MAIN" />
<category android:name="android.intent.category.LAUNCHER"
/>
</intent-filter>
</activity>
<activity android:name="LocationEditor"
android:label="@string/title_note">
<intent-filter android:label="@string/resolve_edit">
<action android:name="android.intent.action.VIEW" />
<action android:name="android.intent.action.EDIT" />
<action
android:name="com.google.android.notepad.action.EDIT_LOCATION" />
<category android:name="android.intent.category.DEFAULT" />
<dataandroid:mimeType="vnd.android.cursor.item/
vnd.android_programmers_guide.friend" />
</intent-filter>
<intent-filter>
<action android:name="android.intent.action.INSERT" />
<category android:name="android.intent.category.DEFAULT" />
<dataandroid:mimeType="vnd.android.cursor.dir/
vnd.android_programmers_guide.friend" />
</intent-filter>
</activity>
<activity android:name="NameEditor"
android:label="@string/title_edit_title"
android:theme="@android:style/Theme.Dialog">
<intent-filter android:label="@string/resolve_title">
<action android:name="com.google.android.notepad.action.EDIT_NAME"
/>
<category android:name="android.intent.category.DEFAULT" />
<category android:name="android.intent.category.ALTERNATIVE" />
```

```
<category android:name="android.intent.category.SELECTED_ALTERNATIVE" />
<dataandroid:mimeType="vnd.android.cursor.item/
vnd.android_programmers_guide.friend" />
</intent-filter>
</activity>
</application>
<uses-permission android:name="android.permission.ACCESS_GPS">
</uses-permission><uses-permission
android:name="android.permission.ACCESS_LOCATION">
</uses-permission></manifest>
```

在下一节中，将为 NameEditor 项目创建第一个活动。正如名称所示，当用户期望编辑一个 friend 的名称时，这个活动将被启动。

创建 NameEditor 活动

创建 NameEditor 活动 第十一章(6)

在本节中，你将为 FindAFriend 项目创建 NameEditor 活动。这个活动将被从主活动 FindAFriend 菜单项目中启动（还没有创建）。NameEditor 活动的目的是修改一个 Friend 记录的姓名字段。

增加一个 name_editor.xml 布局文件并且一个相应的 NameEditor.java 文件到应用程序中。你将编辑这些文件来创建你的活动。

首先，编辑 name_editor.xml 文件来为活动创建布局文件。活动将有一个 **EditText** 和一个 **Button**。EditText 将允许你修改名称字段，Button 将写入结果并退出。如果你一开始就跟从本书的话，你已经增加了不少的 View 布局到 XML 文件中。因此，我可以免去细节，玩战的 name_editor.xml 文件应当如下所示：

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:orientation="vertical"
android:paddingLeft="6dip"
android:paddingRight="6dip"
android:paddingBottom="3dip">
<EditText android:id="@+id/name"
android:maxLines="1"
android:layout_marginTop="2dip"
android:layout_width="wrap_content"
android:ems="25"
android:layout_height="wrap_content"
android:autoText="true"
android:capitalize="sentences"
```

```
android:scrollHorizontally="true" />
<Button android:id="@+id/ok"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_gravity="right"
android:text="@string/button_ok" />
</LinearLayout>
```

现在，编辑 NameEditor.java 并开始写代码。需要导入前一节的 Friends 类并且导入 Cursor 包装来帮助你使用数据库记录：

```
import android.app.Activity;
import android.database.Cursor;
import android.net.Uri;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
```

应当建立活动来执行 View.OnClickListener()。这将允许你在活动中优先 OnClickListener()方法。这部分代码显示 NameEditor 类的概要和一些你需要的变量定义：

```
public class NameEditor extends Activity implements View.OnClickListener {
public static final String EDIT_NAME_ACTION =
"android_programmers_guide.FindAFriend.action.EDIT_NAME";
private static final int NAME_INDEX = 1;
private static final String[] PROJECTION = new String[] {
Friends.Friend._ID,
Friends.Friend.NAME,
};
Cursor mCursor;
EditText mText;
}
```

下一步，需要优先一些方法，从 onCreate()开始。已经在其它章节里看过方法被优先。通常，当活动被创建，它保留所有被执行的代码：

```
public void onCreate(Bundle icicle) {
super.onCreate(icicle);
setContentView(R.layout.name_editor);
Uri uri = getIntent().getData();
mCursor = managedQuery(uri, PROJECTION, null, null);
mText = (EditText) this.findViewById(R.id.name);
mText.setOnClickListener(this);
Button b = (Button) findViewById(R.id.ok);
```

```
b.setOnClickListener(this);
}
```

注意，在前面的代码示例中，你赋值布局到各自的 Views 并且初始化你的变量。可是，你可能想知道为姓名字段准备的数据在哪里。那就是，你已经创建了一个光标，但是从中还没有检索任何东西。为此，你将使用 onResume()方法。

下面两个优先的方法是，onResume()和 onPause()，作用是独自的读取和写入数据库。在 Android 生命周期中，当活动被打开并且在焦点之上，onResume()被呼叫。onPause()被呼叫的情况是当活动被打开，但是在焦点被传递到另一个活动之前。

优先 onResume()方法来读取数据库并检索姓名字段：

```
protected void onResume() {
super.onResume();
if (mCursor != null) {
mCursor.first();
String title = mCursor.getString(NAME_INDEX);
mText.setText(title);
}
}
```

在这种方式下，你移动光标到第一个记录，使用前面赋值的索引从中读取姓名字段，然后设置 EditText 到姓名字段的内容。下一步，修改 onPause()方法来写回 EditText 内容到数据库：

```
protected void onPause() {
super.onPause();
if (mCursor != null) {
String title = mText.getText().toString();
mCursor.updateString(NAME_INDEX, title);
mCursor.commitUpdates();
}
}
```

最后，从 onClick 句柄呼叫活动方法 finish()。它将清除并关闭活动。完成后的 NameEditor.java 文件应当如下：

```
package android_programmers_guide.FindAFriend;
import android_programmers_guide.FindAFriend.Friends;
import android.app.Activity;
import android.database.Cursor;
import android.net.Uri;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
```

```
public class NameEditor extends Activity implements View.OnClickListener {
public static final String EDIT_NAME_ACTION =
"android_programmers_guide.FindAFriend.action.EDIT_NAME";
private static final int NAME_INDEX = 1;
private static final String[] PROJECTION = new String[] {
Friends.Friend._ID,
Chapter 11: Application: Find a Friend 281
Friends.Friend.NAME,
};
Cursor mCursor;
EditText mText;
@Override
public void onCreate(Bundle icicle) {
super.onCreate(icicle);
setContentView(R.layout.name_editor);
Uri uri = getIntent().getData();
mCursor = managedQuery(uri, PROJECTION, null, null);
mText = (EditText) this.findViewById(R.id.name);
mText.setOnClickListener(this);
Button b = (Button) findViewById(R.id.ok);
b.setOnClickListener(this);
}
@Override
protected void onResume() {
super.onResume();
if (mCursor != null) {
mCursor.first();
String title = mCursor.getString(NAME_INDEX);
mText.setText(title);
}
}
@Override
protected void onPause() {
super.onPause();
if (mCursor != null) {
String title = mText.getText().toString();
mCursor.updateString(NAME_INDEX, title);
mCursor.commitUpdates();
}
}
public void onClick(View v) {
finish();
}
}
```

这样，你可以在 Friends 数据库编辑姓名的值。总之，数据库中的两个字段重要，姓名和位置。下一节，将为位置字段创建编辑器。

创建 LocationEditor 活动

创建 LocationEditor 活动 第十一章(7)

在本节中，你将为 Friends 数据库的“位置”字段创建一个编辑器。做这个活动，你将会从 NameEditor 活动做一点小的修改。因此，代码和过程不同。

如果你浏览了 Google NotePad 演示版，你应当注意到“notes”编辑器是一个白色屏幕，带有动态自身重复划线。这是个使用定制 View 执行的结果。你会用这个相同的 View 来制作 LocationEditor。

location_editor.xml

第一步是独自创建 location_editor.xml 和 LocationEditor.java 文件的布局和代码。布局文件应当包含一个到定制 View 布局的呼叫。完整的布局文件如下：

```
<?xml version="1.0" encoding="utf-8"?>
<view xmlns:android="http://schemas.android.com/apk/res/android"
class="android_programmers_guide.FindAFriend.LocationEditor$MyEditText"
android:id="@+id/location"
android:layout_width="fill_parent"
android:layout_height="fill_parent"
android:background="#ffffff"
android:padding="10dip"
android:scrollbars="vertical"
android:fadingEdge="vertical" />
```

LocationEditor 还会包含一个菜单系统，允许用户废弃，删除或者恢复他们做的任何改变。这会是一个非常复杂的活动。因此，最好从头开始，那就是 LocationEditor.java 文件的输入部分。

LocationEditor.java

看看这个活动的输入，很多是处理屏幕上 View 的图样：

```
import android.app.Activity;
import android.content.ComponentName;
import android.content.Context;
import android.content.Intent;
import android.database.Cursor;
import android.graphics.Canvas;
import android.graphics.Paint;
import android.graphics.Rect;
import android.net.Uri;
import android.os.Bundle;
```

```
import android.util.AttributeSet;
import android.view.Menu;
import android.widget.EditText;
import java.util.Map;
```

下一步，设置活动的主类概要。使用 LocationEditor 期间，这里有不少的变量需要定义：

```
public class LocationEditor extends Activity {
private static final String TAG = "Friends";
private static final int FRIEND_INDEX = 1;
private static final int NAME_INDEX = 2;
private static final int MODIFIED_INDEX = 3;
private static final String[] PROJECTION = new String[] {
Friends.Friend._ID, // 0
Friends.Friend.LOCATION, // 1
Friends.Friend.NAME, // 2
Friends.Friend.MODIFIED_DATE // 3
};
private static final String ORIGINAL_CONTENT = "origContent";
private static final int REVERT_ID = Menu.FIRST;
private static final int DISCARD_ID = Menu.FIRST + 1;
private static final int DELETE_ID = Menu.FIRST + 2;
private static final int STATE_EDIT = 0;
private static final int STATE_INSERT = 1;
private int mState;
private boolean mNoteOnly = false;
private Uri mURI;
private Cursor mCursor;
Chapter 11: Application: Find a Friend 285
private EditText mText;
private String mOriginalContent;
}
```

所有本章上一节执行的任务，变量的定义是无需加以说明的。

下一小块的代码展示需要创建的子类。这个子类将把 EditText 画在屏幕上。你把它分开，这样可以从活动中按照需要呼叫它。记住，你会在屏幕上根据用户的需要动态的绘制一个新的 EditText。特别注意需要优先的 onDraw 类：

```
public static class MyEditText extends EditText {
private Rect mRect;
private Paint mPaint;
public MyEditText(Context context, AttributeSet attrs, Map params) {
super(context, attrs, params);
mRect = new Rect();
```

```
mPaint = new Paint();
mPaint.setStyle(Paint.Style.STROKE);
mPaint.setColor(0xFF0000FF);
}
@Override
protected void onDraw(Canvas canvas) {
int count = getLineCount();
Rect r = mRect;
Paint paint = mPaint;
for (int i = 0; i < count; i++) {
int baseline = getLineBounds(i, r);
canvas.drawLine(r.left, baseline + 1, r.right, baseline + 1,
paint);
}
super.onDraw(canvas);
}
}
```

还有，看上去好像很多的代码，但是都不陌生。这个子类只是绘制一个所需的新 EditText 到屏幕上。

和 NameEditor 一样，你会使用 onResume()和 onPause()方法来使用数据库。看看下面每个的代码：

```
protected void onResume() {
super.onResume();
if (mCursor != null) {
mCursor.first();
if (mState == STATE_EDIT) {
setTitle(getText(R.string.title_edit));
} else if (mState == STATE_INSERT) {
setTitle(getText(R.string.title_create));
}
String note = mCursor.getString(FRIEND_INDEX);
mText.setTextKeepState(note);
if (mOriginalContent == null) {
mOriginalContent = note;
}
} else {
setTitle(getText(R.string.error_title));
mText.setText(getText(R.string.error_message));
}
}
protected void onPause() {
super.onPause();
```

```
if (mCursor != null) {
String text = mText.getText().toString();
int length = text.length();
if (isFinishing() && (length == 0) && !mNoteOnly) {
setResult(RESULT_CANCELED);
deleteFriend();
} else {
if (!mNoteOnly) {
mCursor.updateLong(MODIFIED_INDEX,
System.currentTimeMillis());
if (mState == STATE_INSERT) {
String title = text.substring(0, Math.min(30,
length));
Chapter 11: Application: Find a Friend 287
if (length > 30) {
int lastSpace = title.lastIndexOf(' ');
if (lastSpace > 0) {
title = title.substring(0, lastSpace);
}
}
mCursor.updateString(NAME_INDEX, title);
}
}
mCursor.updateString(FRIEND_INDEX, text);
managedCommitUpdates(mCursor);
}
}
}
```

和 NameEditor 很像，在 onResume()期间从数据库读取，在 onPause()期间写回到数据库。在 LocationEditor 中增加的一个和 NameEditor 相反的特性是当你修改时，还写完修改的数据。

最后，需要两个方法来取消和删除 friends。这些方法将被从菜单系统中呼叫：

```
private final void cancelFriend() {
if (mCursor != null) {
if (mState == STATE_EDIT) {
mCursor.updateString(FRIEND_INDEX, mOriginalContent);
mCursor.commitUpdates();
mCursor.deactivate();
mCursor = null;
} else if (mState == STATE_INSERT) {
deleteFriend();
}
```

```
}
setResult(RESULT_CANCELED);
finish();
}
private final void deleteFriend() {
if (mCursor != null) {
mText.setText("");
mCursor.deleteRow();
mCursor.deactivate();
mCursor = null;
}
}
```

假定你已经学会了第八章中的创建菜单系统，简单的检查一下完整的LocationEditor.java文件来所有的这些方法和子类如何一起工作：

```
package android_programmers_guide.FindAFriend;
import android.app.Activity;
import android.content.ComponentName;
import android.content.Context;
import android.content.Intent;
import android.database.Cursor;
import android.graphics.Canvas;
import android.graphics.Paint;
import android.graphics.Rect;
import android.net.Uri;
import android.os.Bundle;
import android.util.AttributeSet;
import android.view.Menu;
import android.widget.EditText;
import java.util.Map;
public class LocationEditor extends Activity {
private static final String TAG = "Friends";
private static final int FRIEND_INDEX = 1;
private static final int NAME_INDEX = 2;
private static final int MODIFIED_INDEX = 3;
private static final String[] PROJECTION = new String[] {
Friends.Friend._ID, // 0
Friends.Friend.LOCATION, // 1
Friends.Friend.NAME, // 2
Friends.Friend.MODIFIED_DATE // 3
};
private static final String ORIGINAL_CONTENT = "origContent";
private static final int REVERT_ID = Menu.FIRST;
private static final int DISCARD_ID = Menu.FIRST + 1;
```

```
private static final int DELETE_ID = Menu.FIRST + 2;
private static final int STATE_EDIT = 0;
private static final int STATE_INSERT = 1;
private int mState;
288 Android: A Programmer’ s Guide
Chapter 11: Application: Find a Friend 289
private boolean mNoteOnly = false;
private Uri mURI;
private Cursor mCursor;
private EditText mText;
private String mOriginalContent;
public static class MyEditText extends EditText {
private Rect mRect;
private Paint mPaint;
public MyEditText(Context context, AttributeSet attrs, Map params) {
super(context, attrs, params);
mRect = new Rect();
mPaint = new Paint();
mPaint.setStyle(Paint.Style.STROKE);
mPaint.setColor(0xFF0000FF);
}
@Override
protected void onDraw(Canvas canvas) {
int count = getLineCount();
Rect r = mRect;
Paint paint = mPaint;
for (int i = 0; i < count; i++) {
int baseline = getLineBounds(i, r);
canvas.drawLine(r.left, baseline + 1, r.right, baseline + 1,
paint);
}
super.onDraw(canvas);
}
}
@Override
protected void onCreate(Bundle icicle) {
super.onCreate(icicle);
final Intent intent = getIntent();
final String type = intent.resolveType(this);
final String action = intent.getAction();
if (action.equals(Intent.EDIT_ACTION)) {
mState = STATE_EDIT;
mURI = intent.getData();
} else if (action.equals(Intent.INSERT_ACTION)) {
```

```
mState = STATE_INSERT;
290 Android: A Programmer’s Guide
mURI = getContentResolver().insert(intent.getData(), null);
if (mURI == null) {
finish();
return;
}
setResult(RESULT_OK, mURI.toString());
} else {
finish();
return;
}
setContentView(R.layout.location_editor);
mText = (EditText) findViewById(R.id.location);
mCursor = managedQuery(mURI, PROJECTION, null, null);
if (icicle != null) {
mOriginalContent = icicle.getString(ORIGINAL_CONTENT);
}
}
@Override
protected void onResume() {
super.onResume();
if (mCursor != null) {
mCursor.first();
if (mState == STATE_EDIT) {
setTitle(getText(R.string.title_edit));
} else if (mState == STATE_INSERT) {
setTitle(getText(R.string.title_create));
}
String note = mCursor.getString(FRIEND_INDEX);
mText.setTextKeepState(note);
if (mOriginalContent == null) {
mOriginalContent = note;
}
} else {
setTitle(getText(R.string.error_title));
mText.setText(getText(R.string.error_message));
}
}
Chapter 11: Application: Find a Friend 291
@Override
protected void onFreeze(Bundle outState) {
outState.putString(ORIGINAL_CONTENT, mOriginalContent);
}
```

```
@Override
protected void onPause() {
super.onPause();
if (mCursor != null) {
String text = mText.getText().toString();
int length = text.length();
if (isFinishing() && (length == 0) && !mNoteOnly) {
setResult(RESULT_CANCELED);
deleteFriend();
} else {
if (!mNoteOnly) {
mCursor.updateLong(MODIFIED_INDEX,
System.currentTimeMillis());
if (mState == STATE_INSERT) {
String title = text.substring(0, Math.min(30,
length));
if (length > 30) {
int lastSpace = title.lastIndexOf(' ');
if (lastSpace > 0) {
title = title.substring(0, lastSpace);
}
}
mCursor.updateString(NAME_INDEX, title);
}
}
mCursor.updateString(FRIEND_INDEX, text);
managedCommitUpdates(mCursor);
}
}
}
@Override
public boolean onCreateOptionsMenu(Menu menu) {
super.onCreateOptionsMenu(menu);
if (mState == STATE_EDIT) {
menu.add(0, REVERT_ID, R.string.menu_revert).setShortcut('0',
'r');
if (!mNoteOnly) {
menu.add(0, DELETE_ID,
R.string.menu_delete).setShortcut('1', 'd');
}
} else {
menu.add(0, DISCARD_ID, R.string.menu_discard).setShortcut('0',
'd');
}
```

```
if (!mNoteOnly) {
Intent intent = new Intent(null, getIntent().getData());
intent.addCategory(Intent.ALTERNATIVE_CATEGORY);
menu.addIntentOptions(
Menu.ALTERNATIVE, 0,
new ComponentName(this, LocationEditor.class), null,
intent, 0, null);
}
return true;
}
@Override
public boolean onOptionsItemSelected(Menu.Item item) {
switch (item.getId()) {
case DELETE_ID:
deleteFriend();
finish();
break;
case DISCARD_ID:
cancelFriend();
break;
case REVERT_ID:
cancelFriend();
break;
}
return super.onOptionsItemSelected(item);
}
private final void cancelFriend() {
if (mCursor != null) {
if (mState == STATE_EDIT) {
mCursor.updateString(FRIEND_INDEX, mOriginalContent);
mCursor.commitUpdates();
mCursor.deactivate();
mCursor = null;
} else if (mState == STATE_INSERT) {
deleteFriend();
}
}
setResult(RESULT_CANCELED);
finish();
}
private final void deleteFriend() {
if (mCursor != null) {
mText.setText("");
292 Android: A Programmer’s Guide
```

```
mCursor.deleteRow();
mCursor.deactivate();
mCursor = null;
}
}
}
```

在下一节，会创建绘制 Google Maps Overlay 的活动，FriendsMap 活动将从 Friends 数据库中读取完整的 friends 记录集并把每一个写入 Overlay。

创建 FriendsMap 活动 第十一章(8)

FriendsMap 是最后一个可以从主程序中呼叫的活动。本活动将从 Friends 数据库中呼叫数据集并且在 Google Map 上为每一个 friend 画一个圆圈。该活动还将为你的当前位置画一个圆圈。

从增加两个新文件来开始本项目，friendsmap.xml 和 FriendsMap.java 文件。因为你已经在第九章内看过 friendsmap.xml 文件了，所以没有必要再解释一遍。使用的是一个 RelativeLayout 来把 4 个按钮放在一个 Google Map 上的。完整的 friendsmap.xml 文件应当看上去如下：

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
android:orientation="vertical"
android:layout_width="fill_parent"
android:layout_height="fill_parent"
>
<view class="com.google.android.maps.MapView"
android:id="@+id/myMap"
android:layout_width="wrap_content"
android:layout_height="wrap_content"/>
<Button android:id="@+id/buttonZoomIn"
style="?android:attr/buttonStyleSmall"
android:text="+"
android:layout_width="wrap_content"
android:layout_height="wrap_content" />
<Button android:id="@+id/buttonMapView"
style="?android:attr/buttonStyleSmall"
android:text="Map"
android:layout_alignRight="@+id/myMap"
android:layout_width="wrap_content"
android:layout_height="wrap_content" />
<Button android:id="@+id/buttonSatView"
style="?android:attr/buttonStyleSmall"
```

```
android:text="Sat"
android:layout_alignRight="@+id/myMap"
android:layout_alignBottom="@+id/myMap"
android:layout_width="wrap_content"
android:layout_height="wrap_content" />
<Button android:id="@+id/buttonZoomOut"
style="?android:attr/buttonStyleSmall"
android:text="-"
android:layout_alignBottom="@+id/myMap"
android:layout_width="wrap_content"
android:layout_height="wrap_content" />
</RelativeLayout>
```

因为在第九章已经看过绝大多数的 FriendsMap.java 文件了，我不会再阐述每一个细节。但是，有一个方法需要解释。

你会创建一个叫做 LoadFriends()的方法来存取数据库，读取记录，并且绘制 Overlay。看一下 LoadFriends()的代码。注意，你打开数据库，匹配并分析位置字段，从位置字段的纬度和经度创建点，并且绘制点到 Overlay 中。这个方法最后所做的事情就是从 GPS 上抓取坐标并且在 Overlay 上绘制出来，用标签“ME”表示。

```
public void LoadFriends(MapView mv, MapController mc, Cursor c){
Point myLocation = null;
Double latPoint = null;
Double lngPoint = null;
c.first();
do{
if (c.getString(c.getColumnIndex("location")) != null) {
final String geoPattern = "(geo:[\\-]?[0-9]{1,3}\\.[0
9]{1,6}\\,[\\-]?[0-9]{1,3}\\.[0-9]{1,6}\\#)";
Pattern pattern = Pattern.compile(geoPattern);
CharSequence inputStr =
c.getString(c.getColumnIndex("location"));
Matcher matcher = Pattern.matcher(inputStr);
boolean matchFound = matcher.find();
if (matchFound) {
String groupStr = matcher.group(0);
latPoint =
Double.valueOf(groupStr.substring(groupStr.indexOf(":") + 1,
groupStr.indexOf(","))) ;
lngPoint =
Double.valueOf(groupStr.substring(groupStr.indexOf(",") + 1,
groupStr.indexOf("#"))) ;
Point friendLocation = new
Point(latPoint.intValue(),lngPoint.intValue());
```

```
294 Android: A Programmer's Guide
drawFriendsOverlay.addNewFriend(c.getString(c.getColumnIndex("name")),
friendLocation);
}
}
}while(c.next());
LocationManager myManager = (LocationManager)
getSystemService(Context.LOCATION_SERVICE);
Double myLatPoint =
myManager.getCurrentLocation("gps").getLatitude()*1E6;
Double myLngPoint =
myManager.getCurrentLocation("gps").getLongitude()*1E6;
myLocation = new Point(myLatPoint.intValue(),myLngPoint.intValue());
drawFriendsOverlay.addNewFriend("Me", myLocation);
mc.centerMapTo(myLocation, false);
mc.zoomTo(9);
mv = null;
}
```

剩余的 FriendsMap.java 文件操控第十章介绍的缩放和棒性按钮：

```
package android_programmers_guide.FindAFriend;
import android.os.Bundle;
import android.location.LocationManager;
import android.view.View;
import android.content.Context;
import android.content.Intent;
import android.database.Cursor;
import android.widget.Button;
import java.util.regex.Pattern;
import java.util.regex.Matcher;
import android.graphics.Canvas;
import android.graphics.RectF;
import android.graphics.Paint;
import com.google.android.maps.MapActivity;
import com.google.android.maps.MapView;
import com.google.android.maps.Point;
import com.google.android.maps.MapController;
import com.google.android.maps.Overlay;
import com.google.android.maps.OverlayController;
public class FriendsMap extends MapActivity {
private static final String[] PROJECTION = new String[] {
Friends.Friend.NAME, Friends.Friend.LOCATION};
public Cursor mCursor;
DrawFriendsOverlay drawFriendsOverlay = new DrawFriendsOverlay();
@Override
```

```
public void onCreate(Bundle icicle) {
super.onCreate(icicle);
setContentView(R.layout.friendsmap);
Intent intent = getIntent();
if (intent.getData() == null) {
intent.setData(Friends.Friend.CONTENT_URI);
}
mCursor = managedQuery(getIntent().getData(), PROJECTION, null,null);
final MapView myMap = (MapView) findViewById(R.id.myMap);
final MapController myMapController = myMap.getController();
LoadFriends(myMap, myMapController, mCursor);
OverlayController myOverlayController =
myMap.createOverlayController();
myOverlayController.add(drawFriendsOverlay, true);
final Button zoomIn = (Button) findViewById(R.id.buttonZoomIn);
zoomIn.setOnClickListener(new Button.OnClickListener() {
public void onClick(View v){
ZoomIn(myMap,myMapController);
}});
final Button zoomOut = (Button) findViewById(R.id.buttonZoomOut);
zoomOut.setOnClickListener(new Button.OnClickListener() {
public void onClick(View v){
ZoomOut(myMap,myMapController);
}});
final Button viewMap = (Button) findViewById(R.id.buttonMapView);
viewMap.setOnClickListener(new Button.OnClickListener() {
public void onClick(View v){
ShowMap(myMap,myMapController);
}});
final Button viewSat = (Button) findViewById(R.id.buttonSatView);
viewSat.setOnClickListener(new Button.OnClickListener() {
public void onClick(View v){
ShowSat(myMap,myMapController);
}});
}
public void LoadFriends(MapView mv, MapController mc, Cursor c){
Point myLocation = null;
Double latPoint = null;
Double lngPoint = null;
c.first();
do{
if (c.getString(c.getColumnIndex("location")) != null) {
final String geoPattern = "(geo:[\\-]?[0-9]{1,3}\\.[0
9]{1,6}\\,[\\-]?[0-9]{1,3}\\.[0-9]{1,6}\\#)";
```

```
Pattern pattern = Pattern.compile(geoPattern);
CharSequence inputStr =
c.getString(c.getColumnIndex("location"));
Matcher matcher = pattern.matcher(inputStr);
boolean matchFound = matcher.find();
if (matchFound) {
String groupStr = matcher.group(0);
latPoint =
Double.valueOf(groupStr.substring(groupStr.indexOf(":") + 1,
groupStr.indexOf(","))) ;
lngPoint =
Double.valueOf(groupStr.substring(groupStr.indexOf(",") + 1,
groupStr.indexOf("#"))) ;
Point friendLocation = new
Point(latPoint.intValue(),lngPoint.intValue());
drawFriendsOverlay.addNewFriend(c.getString(c.getColumnIndex("name")),
friendLocation);
}
}
}while(c.next());
LocationManager myManager = (LocationManager)
getSystemService(Context.LOCATION_SERVICE);
Double myLatPoint =
myManager.getCurrentLocation("gps").getLatitude()*1E6;
Double myLngPoint =
myManager.getCurrentLocation("gps").getLongitude()*1E6;
myLocation = new Point(myLatPoint.intValue(),myLngPoint.intValue());
drawFriendsOverlay.addNewFriend("Me", myLocation);
mc.centerMapTo(myLocation, false);
mc.zoomTo(9);
mv = null;
}
public void ZoomIn(MapView mv, MapController mc){
if(mv.getZoomLevel()!=21){
mc.zoomTo(mv.getZoomLevel()+ 1);
}
}
public void ZoomOut(MapView mv, MapController mc){
if(mv.getZoomLevel()!=1){
mc.zoomTo(mv.getZoomLevel()- 1);
}
}
public void ShowMap(MapView mv, MapController mc){
if (mv.isSatellite()){
```

```
mv.toggleSatellite();
}
}
public void ShowSat(MapView mv, MapController mc){
if (!mv.isSatellite()){
mv.toggleSatellite();
}
}
protected class DrawFriendsOverlay extends Overlay{
public String[] friendName = new String[0];
public Point[] friendPoint = new Point[0];
final Paint paint = new Paint();
@Override
public void draw(Canvas canvas, PixelCalculator calculator, Boolean
shadow){
for(int x=0;x<friendPoint.length; x++){
int[] coords = new int[2];
calculator.getPointXY(friendPoint[x], coords);
RectF oval = new RectF(coords[0] - 7, coords[1] + 7,
coords[0] + 7, coords[1] - 7);
paint.setTextSize(14);
canvas.drawText(friendName[x],
coords[0] +9, coords[1], paint);
canvas.drawOval(oval, paint);
}
}
public void addNewFriend(String name,Point point ){
int x = friendPoint.length;
String[] friendNameB = new String[x + 1];
Point[] friendPointB = new Point[x + 1];
System.arraycopy(friendName, 0, friendNameB, 0, x );
System.arraycopy(friendPoint, 0, friendPointB, 0, x);
friendNameB[x] = name;
friendPointB[x]= point;
friendName = new String[x + 1];
friendPoint = new Point[x + 1];
System.arraycopy(friendNameB, 0, friendName, 0, x + 1 );
System.arraycopy(friendPointB, 0, friendPoint, 0, x + 1 );
}
}
}
```

完成本项目的最后一个任务是创建主活动，FindAFriend。该活动被放置在一个壳内来呼叫本章创建的活动。

创建 FindAFriend 活动 第十一章(9)

要开始本节，创建两个文件，findafriend.xml 和 FindAFriend.java。再说一次，这些文件将为当前部分独自保留你的布局和代码。布局文件非常的基本并且只有一个 TextView。这个TextView 将被用来写入到 friends 的列表中。完整的 findafriend.xml 文件应当显示如下：

```
<?xml version="1.0" encoding="utf-8"?>
<TextView xmlns:android="http://schemas.android.com/apk/res/android"
android:id="@android:id/text1"
android:layout_width="fill_parent"
android:layout_height="?android:attr/listPreferredItemHeight"
android:textAppearance="?android:attr/textAppearanceLargeInverse"
android:gravity="center_vertical"
android:paddingLeft="27dip"
/>
```

完整的 FindAFriend.java 文件如下。文件中的所有代码在本章中已经讨论过。首先，读取数据库并且写入结果到一个 ListView。给予用户一个菜单项目来编辑或者删除条目，或者启动 FriendsMap 活动。很简单，对不对？

```
package android_programmers_guide.FindAFriend;
import android_programmers_guide.FindAFriend.Friends;
import android.app.ListActivity;
import android.content.ComponentName;
import android.content.Intent;
import android.content.ContentUris;
import android.database.Cursor;
import android.graphics.Color;
import android.net.Uri;
import android.os.Bundle;
import android.view.Menu;
import android.view.View;
import android.view.View.MeasureSpec;
import android.widget.ListAdapter;
import android.widget.ListView;
import android.widget.SimpleCursorAdapter;
import android.widget.TextView;
public class FindAFriend extends ListActivity {
300 Android: A Programmer’s Guide
public static final int DELETE_ID = Menu.FIRST;
public static final int INSERT_ID = Menu.FIRST + 1;
public static final int FIND_FRIENDS = Menu.FIRST + 2;
private static final String[] PROJECTION = new String[] {
Friends.Friend._ID, Friends.Friend.NAME};
```

```
private Cursor mCursor;
@Override
protected void onCreate(Bundle icicle) {
super.onCreate(icicle);
setDefaultKeyMode(SHORTCUT_DEFAULT_KEYS);
Intent intent = getIntent();
if (intent.getData() == null) {
intent.setData(Friends.Friend.CONTENT_URI);
}
setupList();
mCursor = managedQuery(getIntent().getData(), PROJECTION, null,
null);
ListAdapter adapter = new SimpleCursorAdapter(this,
R.layout.findafriend_item, mCursor,
new String[] {Friends.Friend.NAME}, new int[]
{android.R.id.text1});
setListAdapter(adapter);
}
private void setupList() {
View view = getViewInflate().inflate(
android.R.layout.simple_list_item_1, null, null);
TextView v = (TextView) view.findViewById(android.R.id.text1);
v.setText("X");
getListView().setBackgroundColor(Color.GRAY);
v.measure(MeasureSpec.makeMeasureSpec(View.MeasureSpec.EXACTLY,
100),
MeasureSpec.makeMeasureSpec(View.MeasureSpec.UNSPECIFIED,
0));
}
@Override
public boolean onCreateOptionsMenu(Menu menu) {
super.onCreateOptionsMenu(menu);
menu.add(0, INSERT_ID, R.string.menu_insert).setShortcut('3', 'a');
Intent intent = new Intent(null, getIntent().getData());
intent.addCategory(Intent.ALTERNATIVE_CATEGORY);
menu.addIntentOptions(
Menu.ALTERNATIVE, 0, new ComponentName(this, FindAFriend.class),
null, intent, 0, null);
return true;
}
@Override
public boolean onPrepareOptionsMenu(Menu menu) {
super.onPrepareOptionsMenu(menu);
final boolean haveItems = mCursor.count() > 0;
```

```
if (haveItems) {
Uri uri = ContentUris.withAppendedId(getIntent().getData(),
getSelectedItemId());
Intent[] specifics = new Intent[1];
specifics[0] = new Intent(Intent.EDIT_ACTION, uri);
Menu.Item[] items = new Menu.Item[1];
Intent intent = new Intent(null, uri);
intent.addCategory(Intent.SELECTED_ALTERNATIVE_CATEGORY);
menu.addIntentOptions(Menu.SELECTED_ALTERNATIVE, 0, null,
specifics, intent, 0, items);
menu.add(Menu.SELECTED_ALTERNATIVE, DELETE_ID,
R.string.menu_delete)
.setShortcut('2', 'd');
menu.add(Menu.SELECTED_ALTERNATIVE, FIND_FRIENDS,
R.string.find_friends).setShortcut('4', 'f');
if (items[0] != null) {
items[0].setShortcut('1', 'e');
}
} else {
menu.removeGroup(Menu.SELECTED_ALTERNATIVE);
}
menu.setItemShown(DELETE_ID, haveItems);
return true;
}
@Override
public boolean onOptionsItemSelected(Menu.Item item) {
switch (item.getId()) {
case DELETE_ID:
deleteItem();
return true;
case INSERT_ID:
insertItem();
return true;
case FIND_FRIENDS:
Intent findfriends = new Intent(this, FriendsMap.class);
startActivity(findfriends);
return true;
}
return super.onOptionsItemSelected(item);
}
@Override
protected void onListItemClick(ListView l, View v, int position, long
id) {
Uri url = ContentUris.withAppendedId(getIntent().getData(), id);
```

```
String action = getIntent().getAction();
if (Intent.PICK_ACTION.equals(action)
|| Intent.GET_CONTENT_ACTION.equals(action)) {
setResult(RESULT_OK, url.toString());
} else {
startActivity(new Intent(Intent.EDIT_ACTION, url));
}
}
private final void deleteItem() {
mCursor.moveTo(getSelectedItemPosition());
mCursor.deleteRow();
}
private final void insertItem() {
startActivity(new Intent(Intent.INSERT_ACTION,
getIntent().getData()));
}
}
```

这个是本书中最长的一个活动，需要注意的是你所做的相关所需的编程工作还是非常的少。下一步，运行这个活动并且查看所有工作的成果。

运行 FindAFriend 活动 第十一章(10)

在 Android 模拟器中运行 FindAFriend 活动。应当从一个空的列表开始，如下图（略）。要增加第一个朋友，点击菜单按钮并选择 Add Friend 选项。

本选项启动你创建的定制 View。在提供的行处输入一个朋友的名称，点击返回按钮返回到主活动。

现在在 ListView 中应当有一个朋友的名称。再次点击菜单按钮，显然你拥有更多的选项了。如下图（略）。选择 Edit Location 选项。会再次来到定制控制。输入坐标信息。

最后，返回到主活动并选择 Find Friend 选项。这样分别会清除在旧金山的位置和你朋友在非洲海岸的位置。

试试这个：实时位置更新

试着修改 FindAFriend 应用程序，当你移动时，来更新“ME”标记。这个应该很容易通过使用 update()方法实现。

问专家

Q:SQLite 数据库可以通过代码的方式创建吗？

A:是的。但是，为了更好的了解 Android 的目的，我选择了手工创建数据库例子。可以在 FriendsProvider Content Provider 通过代码创建数据库的 creation 方法自行修改本项目。

Q:需要一个分开的类来执行 BaseColumns?

A:不。可以直接从 Friends 类中定义条目。如果你创建的一个 Content Provider 被其它开发者使用，但是他们不知道数据库的底层结构，那么你就需要提供一个定义的类。

Android SDK 工具参考 第十二章 （完）

Android SDK 工具参考 第十二章 （完）

本章提供了一些有价值的 Android SDK 工具参考项目，这些工具你已经在本书的课程中使用过了。它们给了你一些命令行选项，可以在 Android 模拟器和 Android 调试桥中使用。

Android 模拟器命令

下表包含了大多数常规的 Android 模拟器命令。这些命令在 2008 年 3 月份发布的 SDK 版本中可用。每个命令提供了简短的描述。

Emulator Command 模拟器命令	功能
emulator -console	Enables the console shell on the current terminal 在当前终端上激活控制台外壳
emulator -data <filename>	Uses a different file as the working user-data disk image 使用一个不同的文件作为工作用户数据磁盘镜像
emulator -debug-kernel	Sends kernel output to the console 发送核心输出到控制台
emulator -flash-keys	Flashes keypresses on the device skin 在设备皮肤上闪烁 keypress
emulator -help	Prints a list of all Emulator commands 列出模拟器列表
emulator -http-proxy <proxy>	Makes all TCP connections through a specified HTTP/HTTPS proxy 通过一个定义的 HTTP/HTTPS 代理制作所有的 TCP 连接。
emulator -image <file> Uses <file>	as the system image 作为系统镜像
emulator -kernel <file> Uses <file>	as the emulated kernel 作为模拟的核心

emulator -logcat <logtags>	Enables logcat output with given tags 用指定的标签激活 logcat 输出
emulator -mic <device or file>	Uses device or WAV file for audio input 使用设备或者 WAV 文件作为音频输入
emulator -netdelay <delay>	设置网络反应时间模拟到<delay>Sets network latency emulation to <delay>. (The <delay> parameter simulates the delay experienced on specific types of networks.<delay>参数模拟在定义类型的网络) The <delay>s you can use are as follows 可以使用的<delay>如下: ● Gprs ● Edge ● Umts ● None ● <num> ● <min>:<max>
emulator -netfast Shortcut for -netspeed full -netdelay	none
emulator -netspeed <speed>	设置网络速度模拟到<speed>。Sets network speed emulation to <speed>. (The <speed> parameter simulates the data speed experienced on specific types of networks.) The <speed>s you can use are as follows: ● Gsm ● Hscsd ● Gprs ● Edge ● Umts ● Hsdpa ● Full ● <num>

	● <up>:<down>
emulator -noaudio	Disables Android audio support 废除 Android 音频支持
emulator -nojni	Disables JNI checks in the Dalvik virtual machine 在 Dalvik virtual machine 中废除 JNI 检查
emulator -noskin	Specifies not to use any Emulator skin 定义为不使用任何模拟器皮肤
emulator -onion <image>	Uses overlay image over screen 在屏幕上使用覆盖图像
emulator -onion-alpha <percent>	Specifies onion skin translucency value (as percent) 定义半透明皮肤值（百分比）
emulator -qemu	Passes arguments to QEMU 传递参数到 QEMU
emulator -qemu -h	Displays QEMU help 显示 QEMU 帮助
emulator -radio <device>	Redirects the radio modem interface to a host character device 重定向无线调制解调器接口到一个主字符设备
emulator -ramdisk <file> Uses <file>	as the ramdisk image 作为 ramdisk 镜像
emulator -raw-keys	Disables Unicode keyboard reverse mapping 禁用 Unicode 键盘转换映射
emulator -sdcard <file> Uses <file>	as the SD Memory Card image 作为 SD 内存卡镜像
emulator -skin <skinID>	使用定义的皮肤启动模拟器 Starts the Emulator with the specified skin: ● HVGA-L 480x320, landscape ● HVGA-P 320x480, portrait (default) ● QVGA-L 320x240, landscape ● QVGA-P 240x320, portrait
emulator -skindir <dir>	Searches for Emulator skins in <dir> 在<dir>内查找模拟器皮肤

emulator -system <dir>	Searches system, ramdisk, and user-data disk images in <dir> 在<dir>内查找系统，ramdisk 和用户数据磁盘镜像
emulator -trace <name>	Enables code profiling (press F9 to start), written to a specified file 启动代码分析（按 F9 启动），写入一个定义的文件
emulator -useaudio	Enables Android audio support 启动 Android 音频支持
emulator -verbose	Enables verbose output 激活冗长的输出
emulator -verbose-keys	Enables verbose keypress messages 激活冗长的的按键信息
emulator -verbose-proxy	Enables verbose proxy debug messages 激活冗长的的代理调试信息
emulator -wipe-data	Deletes all data on the user-data disk image 在用户数据磁盘镜像中删除所有数据（开始前参见 emulator -data<filename>) (see emulator -data <filename>) before starting

Android 调试桥命令

下面的是 gsm 命令。通过连接到 Android 模拟器的终端控制台使用。如果你不了解端口终端控制台，它是一个比调试端口少的端口。执行 adb 来获得活动的设备列表和相关的端口号。

adb Bugreport	Prints dumpsys, dumpstate, and logcat data to the screen, for the purposes of bug reporting 为故障报告在屏幕上打印 dumpsys,dumpstate 和 logcat 数据
adb call <phonenumber>	Simulates an inbound phone call from <phonenumber>模拟一个呼入的电话
adb cancel <phonenumber>	Cancels an inbound phone call from <phonenumber>取消一个呼入的电话
adb -d {<ID>|<serialNumber>}	Lets you direct an adb command to a specific Emulator/device instance, re

	ferred to by its adb-assigned ID or serial number 允许你指引一个 adb 命令到一个定义的模拟器/设备示例中，通过它的赋值 adb ID 或者序列号来引用
adb data <state>	Changes the state of the GPRS data connection to <state>改变 GPRS 数据连接状态到<state>
adb Devices	Prints a list of all attached mulator/device instances 打印所附模拟器列表/设备示例
adb forward <local> <remote>	Forwards socket connections from a specified local port to a specified remote port on the Emulator/device instance 在模拟器/设备示例中从一个定义的本来端口转递 socket 连接到一个定义的远程端口
adb get-serialno	Prints the adb instance identifier string 打印 adb 示例标识符字符串
adb get-state	Prints the adb state of an emulator/device instance 打印一个模拟器/设备示例的 adb 状态
adb help	Prints a list of supported adb commands 打印支持的 adb 命令列表
adb install <path-to-apk>	Pushes an Android application (specified as a full path to an .apk file) to the data file of an Emulator/device 推入 Android 应用程序(作为一个完整路径一个.apk 文件）到模拟器/设备示例
adb jdwp	Prints a list of available JDWP processes on a given device 在指定的设备上列出可用的 JDWP 进程
adb kill-server	Terminates the adb server process 终止 adb 服务器进程

adb logcat [<option>] [<filter-specs>]	Prints log data to the screen在屏幕上打印 log 数据
adb ppp <tty> [parm]...	Runs PPP over USB 通过 USB 运行 PPP: ● <tty> The tty for PPP stream; for example, dev:/dev/omap_csmi_ttyl <tty>ppp 流；例如，dev:/dev/omap_csmi_ttyl ● [parm]... Zero or more PPP/PPPD options, such as defaultroute, local, notty, etc. 0 或者更多的 PPP/PPPD 选项，如 defaultroute, local,notty 等等 Note that you should not automatically start a PDP connection. 注意，你不应当自动启动一个 PDP 连接
adb pull <remote> <local>	Copies a specified file from an Emulator/device instance to your development computer 从模拟器/设备复制定义的文件到你的电脑
adb push <local> <remote>	Copies a specified file from your development computer to an Emulator/device instance 从电脑中复制文件到模拟器/设备
adb Shell	Starts a remote shell in the target Emulator/device instance 在目标模拟器/设备上启动远程外壳
adb start-server	Checks whether the adb server process is running and, if not, starts it 检测 adb 服务器进程是否启动，如果否，启动它
adb Status	Reports the current GSM voice/data state 报告当前 GSM 声音/数据状态
adb unregistered	Indicates no network is available 指示无网络可用

adb Version	Prints the adb version number 打印 adb 版本号
adb voice <state>	Changes the state of the GPRS voice connection to <state>改变 GPRS 声音状态连接到<state>
adb wait-for-bootloader	Blocks execution until the bootloader is online—that is, until the instance state is bootloader 阻止执行直到引导装入完成。也就是除非设备完成引导
adb wait-for-device	Blocks execution until the device is online—that is, until the instance state is device 阻止执行直到设备在线。也就是示例状态是设备

注意：为不产生歧义，本表格的解释部分仍保留英文。

至此，历经两个多月的时间，本书的翻译工作全部完成，在英文版的书中还有索引部分，这里就没有必要翻译了。

Android SDK 1.5 - 包装索引

这些就是 Android APIs。

android
包含由标准 Android 应用程序使用的资源类。

android.app

包含了所有 Android 应用程序模块的高级类。

android.appwidget

Android 允许应用程序推动 views 内嵌于其它应用程序。这些 views 被称作为窗口小部件（widgets），并且由“AppWidget Providers”发布。可以包含 widgets 的组件被称作为“AppWidget hose”.

AppWidget Providers

Declaring a widget in the AndroidManifest
Adding the AppWidgetProviderInfo meta-data
Using the AppWidgetProvider class
AppWidget Configuration UI
AppWidget Broadcast Intents
AppWidget Hosts
android.content

包含了为在设备上存取或者发布数据的类。

android.content.pm
包含了关于应用程序包装的存取信息，包括活动，许可，服务，签名和提供者的信息。

android.content.res

包含了存取应用程序资源的类，如原始资源文件，颜色，可绘制的，媒体或者其它在包装中的文件，还有影响应用程序行为的重要设备配置细节（目标，输入类型等等）。

android.database
包含通过一个内容提供者反馈的数据浏览类。

android.database.sqlite
包含 SQlite 数据库管理类，就是应用程序管理的自身数据库。

android.graphics
提供一般的如油画布，颜色过滤器，和矩形等可以用来直接在屏幕上绘制图形的工具。

android.graphics.drawable
提供类来管理为显示而准备的多重元素，如位图和变化。

android.graphics.drawable.shapes
包含绘制几何图形的类。

android.hardware
为可能不会出现在每一个 Android 设备上的设备提供硬件支持。

android.inputmethodservice
写入输入方法的基础类。

android.location

定义 Android 位置基础服务和相关服务的类。

android.media
提供管理不同音频和视频媒体接口的类。

android.net
帮助网络存取，除了常规 java.net.*APIs 的类。

android.net.http

android.net.wifi
提供管理设备 Wi-Fi 功能的类。

android.opengl
提供 OpenGL 功用。

android.os
提供基本操作系统服务，信息传递，和设备内进程通信。

android.preference
提供管理应用程序参数选择和执行参数选择 UI 的类。

android.provider
提供适宜的类，由 Android 存取内容提供者。

android.sax
一个可以简单使用 SAX 处理器的框架。

android.speech

android.telephony

提供 APIs 来监控基本电话功能，如网络类型和连接状态，另外加上操作电话号码字符串。

android.telephony.gsm
提供使用 GSM 电话特性的 APIs，如文本/数据/PDU 短信息。

android.test
写 Android 测试事件的框架。

android.test.mock
共用类，提供不同 Android 框架积木的存根或者防治品。

android.test.suitebuilder
工具类，支持测试试运行类。

android.text

用于追踪屏幕上的文本或者文本 spans。

android.text.format

android.text.method

检测或者修改键区输入。

android.text.style

在一个 View 对象中查看或者改变一段文本的风格。

android.text.util
转换文本串到可点击的链接并创建 RFC822-类型信息（SMTP)象征。

android.util
常规方法，如时期/时间操作，基本 64 位编码和解码，字符串和数据转换方法，XML 功用。

android.view
显示用于处理屏幕输出和交互的基本用户接口类

android.view.animation

处理中间动画。

android.view.inputmethod
views 和输入方法之间的框架（如一个软键盘）

android.webkit
浏览网络的工具

android.widget
widget 包装包含 UI 元素来用于应用程序的屏幕

com.android.internal.os

dalvik.bytecode

Dalvik 字节码的类

dalvik.system

定义给 Dalvik VM 的工具和系统信息类

java.awt.font

java.beans

java.io

依靠流，文件系统存取和串行化的输入输出工具

java.lang

Android 环境核心类

java.lang.annotation

为注释支持定义接口和例外必备

java.lang.ref

java.lang.reflect

java.math
提供独断精度整数和小数

java.net

提供网络相关功能，如流和自带寻址信息界面程序，处理网络地址，和处理 HTTP 请求

java.nio

提供缓存来帮助处理数据

java.nio.channels

通道提供了一个连接到数据源的方式，如文件，界面程序或者其它允许输入和/或者输出数据的结构。

java.nio.channels.spi

为 nio 通道的服务提供者类。

java.nio.charset

处理字节和不同字符集的包装

java.nio.charset.spi

为 nio 字符集的服务提供者类

java.security

提供所有组成 Java 安全框架的类和接口

java.security.acl

本包装提供构建 Access Control Lists 所需的类和接口

java.security.cert

提供产生，管理并验证 X.509 证明所需的所有类和接口。

java.security.interfaces

提供需要产生下列键的接口（1）为不对称编码运输法则使用 PKCS#1 标准的键；（2）由 FIPS-186 定义的数字签名法则（DSA)的键；（3）一般椭圆形不对称编码法则的键

java.security.spec

为编码和签名法则所需定义的键和参数提供类和接口

java.sql

为存取 SQL 数据库提供标准的接口

java.text

java.text 包准允许在应用程序中从自然语言分开文本

java.util

提供大量的工具类

java.util.concurrent

在并发程序设计内有用的工具类

java.util.concurrent.atomic
一个小的类工具包，支持单变量 lock-free thread-safe 编程

java.util.concurrent.locks

接口和类提供一个框架来为状态的锁定和等待。该状态从内建同步和监视中分开

java.util.jar

java.jar 包装可以存取来读取和写入一个 java 存档文件或者 JAR 文件。

java.util.logging

运行增加 logging 到任何应用程序

java.util.prefs

提供参数选择途径。写入配置数据给一个不断的数据存储并从中检索。

java.util.regex

常规表达式执行，用于对指定样式匹配，查找和替换字符串。

java.util.zip

压缩或者解压缩 ZIP 和 GZIP 文件

javax.crypto

应用程序加密或者解密执行法则的接口

javax.crypto.interfaces

需要执行 PKCS#3 定义，Diffie-Hellman(DH)键协议法则的接口

javax.crypto.spec

为加密定义键和参数的类和接口

javax.microedition.khronos.egl

javax.microedition.khronos.opengles

javax.net

提供工厂类来创建界面程序和服务器程序

javax.net.ssl

所有执行基于 SSL 协议 SSSLc3.0 或者 TLSv1.2 的安全界面程序所需的类和接口

javax.security.auth

执行和编制不同用户的认可和角色基础用户许可所需的类和接口

javax.security.auth.callback

交互应用程序来执行许可和许可进程所需的类和接口

javax.security.auth.login

基于从 Unix-PAM 模块的概念，提供一个可插入和可堆栈许可系统

javax.security.auth.x500

提供需要存储 X.500 原则和它们的资格证书

javax.security.cert

只为兼容原因提供

javax.sql

扩展进入 SQL 数据库标准接口

javax.xml

有 XML 常数的工具类

javax.xml.parsers

提供分解 XML 文档的能力，从中构建文档对象模块树（DOM）

junit.framework
junit 测试框架

junit.runner
支持 junit 测试框架的工具类

org.apache.http
HTTP 组件的核心接口和类

org.apache.http.auth
相对于服务器的客户端 HTTP 许可 API。通常作为 HttpAuth.

org.apache.http.auth.params
配置 HttpAuth 的参数

org.apache.http.client
客户端 HTTP 通信 API 和 HttpClient 模块的输入点

org.apache.http.client.entity

org.apache.http.client.methods
要求执行不同的 HTTP 方法，如 GET 和 POST

org.apache.http.client.params
配置 HttpClient 的参数

org.apache.http.client.protocol
额外请求和相应拦截器

org.apache.http.client.utils
HttpClient. 的帮助和工具类

org.apache.http.conn
在 HttpConn 核心的客户端连接管理和处理 API

org.apache.http.conn.params
配置 HttpConn 的参数

org.apache.http.conn.routing
客户端路由代理和追踪 API，HttpConn 的部分

org.apache.http.conn.scheme

org.apache.http.conn.ssl

TLS/SSL 定义的 HttpConn API 部分

org.apache.http.conn.util

org.apache.http.cookie
通过 cookies 管理的客户端陈述管理 API，通常作 HttpCookie.

org.apache.http.cookie.params
配置 HttpCookie 的参数

org.apache.http.entity
HTTP 信息实体代理

org.apache.http.impl
为 org.apache.http 内接口默认执行

org.apache.http.impl.auth

org.apache.http.impl.client

org.apache.http.impl.conn

org.apache.http.impl.conn.tsccm
thread-safe 客户端连接管理器的执行

org.apache.http.impl.cookie

org.apache.http.impl.entity
在 org.apache.http.entity 内接口的默认执行

org.apache.http.impl.io

在 org.apache.http.io 内接口的默认执行

org.apache.http.io
HTTP 组件传输层提取

org.apache.http.message
HTTP message 选集执行

org.apache.http.params
HTTP 组件的确定参数框架

org.apache.http.protocol
HTTP 协议执行框架

org.apache.http.util
为多重目的准备的有静态帮助方法的多数工具类。

org.json

org.w3c.dom

官方 W3C java 约束文档对象模块，2 级核心

org.xml.sax

提供核心 SAX APIs

org.xml.sax.ext

包含到 SAX2 的能力，没必要支持符合 SAX 驱动

org.xml.sax.helpers

包含“helper”类，包括支持 bootstrapping SAX 基础的应用程序

org.xmlpull.v1

org.xmlpull.v1.sax2