Android 开发从入门到精通

developerWorks.

Android 是一种基于 Linux® V2.6 内核的综合操作环境。最初，Android 的部署目标是移动电话领域，包括智能电话和更廉价的翻盖手机。但是， Android 全面的计算服务和丰富的功能支持完全有能力扩展到移动电话市场以外。Android 也可以用于其他的平台和应用程序。本专题提供的教程、技术文章首先带大家了解 Android 开发，进而深入到 Android 开发的各个方面。

Android 开发入门

- Android 开发简介
  Android 是 Google 提供的移动、无线、计算机和通信平台。通过使用 Android Eclipse 插件，可以在强大的 Eclipse 环境中构建 Android 应用程序。本教程介绍如何用 Eclipse 插件 Android Development Tools 进行 Android 应用程序开发，包括对 Android 平台和 Android Development Tools 的介绍，并开发两个示例应用程序。
- 用 Eclipse 开发 Android 应用程序
  Android 是一种基于 Linux® V2.6 内核的综合操作环境。最初，Android 的部署目标是移动电话领域，包括智能电话和更廉价的翻盖手机。但是， Android 全面的计算服务和丰富的功能支持完全有能力扩展到移动电话市场以外。Android 也可以用于其他的平台和应用程序。在本文中，阅读对 Android 平台的简介，并学习如何编写基本的 Android 应用程序。

深入 Android 开发

- 手机上的 Scala
  Android 操作系统为移动开发提供强大、开放的平台。它利用了 Java 编程语言和 Eclipse 工具平台的威力。现在，还可以将 Scala 编程语言加入到其中。在本文中，您将看到如何使用 Scala 作为 Android 上的主要开发语言，从而可以使用一种更具表达力、更加类型安全的编程语言编写移动应用程序。
- 构建 Android 手机 RSS 阅读器
  本教程将展示如何使用 Android Developer Tools 读取、解析和显示 XML 数据。构建一个运行在 Android 平台上的 RSS 阅读器，并集成不同的 RSS 或其他 XML 数据源以构建自己的 mash-up 应用。
- Android 助力云计算
  开源的 Android 操作系统已经席卷全球，它允许您在任何位置运行复杂的云计算应用程序。它设计用于在电池供电设备（如 T-Mobile G1 智能电话）上高效工作，Android 本质上就是 Linux，Android 编程模型有多个层，允许创建为云计算量身定做的安全应用程序。使用 Android 达到新的高度并体验前所未有的移动计算吧。
- 使用 Android 实现联网
  Android 是面向应用程序开发的丰富平台，它提供一套出色的用户界面元素和数据管理功能。它还为开发连接到真实世界的应用程序提供了出色的网络选项。您希望连接到哪里？也许您希望告诉全世界您正在使用 Twitter。或者希望收到有关在本地洗衣店或熟食店的排号。如果希望将 Android 连接到您所生活的世界，那么请阅读本文。
- 深入探讨 Android 传感器
  Android 是一个面向应用程序开发的富平台，它拥有许多具有吸引力的用户界面元素和数据管理功能。Android 还提供了一组丰富的接口选项。在本文中，学习如何配合使用 Android 的各种传感器选项监控您的环境。样例代码展示了如何在 Android 电话中录制音频。想构建自己的婴儿监视器吗？想用声音来接听电话或者打开房门吗？请学习如何利用配备有 Android 的设备的硬件功能。

Android 开发简介

developerWorks.

开源的设备平台

Android 是一种基于 Linux® V2.6 内核的综合操作环境。最初，Android 的部署目标是移动电话领域，包括智能电话和更廉价的翻盖手机。但是， Android 全面的计算服务和丰富的功能支持完全有能力扩展到移动电话市场以外。Android 也可以用于其他的平台和应用程序。在本文中，阅读对 Android 平台的简介，并学习如何编写基本的 Android 应用程序。

简介

黑莓和 iPhone 都提供了受欢迎的、高容量的移动平台，但是却分别针对两个不同的消费群体。黑莓是企业业务用户的不二选择。但是，作为一种消费设备，它在易用性和“新奇特性”方面难以和 iPhone 抗衡。Android 则是一个年轻的、有待开发的平台，它有潜力同时涵盖移动电话的两个不同消费群体，甚至可能缩小工作和娱乐之间的差别

如今，很多基于网络或有网络支持的设备都运行某种 Linux 内核。这是一种可靠的平台：可经济有效地进行部署和提供支持，并且可直接作为面向部署的良好的设计方法。这些设备的 UI 通常是基于 HTML 的，可通过 PC 或 Mac 浏览器查看。但并不是每个设备都需要通过一个常规的计算设备来控制。想象一下传统的家用电器，例如电炉、微波炉或面包机。如果您的家用电器由 Android 控制，并且有一个彩色触摸屏，会怎么样？如果电炉上有一个 Android UI，那么操控者甚至可以烹饪点什么东西。

在本文中，了解 Android 平台，以及如何将它用于移动和非移动应用程序。安装 Android SDK，并构建一个简单的应用程序。下载 本文中的示例应用程序的源代码。

Android 简史

Android 平台是 Open Handset Alliance 的成果，Open Handset Alliance 组织由一群共同致力于构建更好的移动电话的公司组成。这个组织由 Google 领导，包括移动运营商、手持设备制造商、零部件制造商、软件解决方案和平台提供商以及市场营销公司。从软件开发的观点看，Android 正处在开源领域的中心位置。

市场上第一款支持 Android 的手机是由 HTC 制造并由 T-Mobile 供应的 G1。这款设备从设想到推出花了大约一年的时间，惟一可用的软件开发工具是一些实行增量改进的 SDK 发行版。随着 G1 发行日的临近，Android 团队发布了 SDK V1.0，用于这个新平台的应用程序也浮出水面。

为了鼓励创新，Google 举办了两届 “Android Developer Challenges”，为优胜的参赛作品提供数百万美金的奖励。G1 问世几个月之后，随后就发布了 Android Market，它使用户可以浏览应用程序，并且可以将应用程序直接下载到他们的手机上。经过大约 18 个月，一个新的移动平台进入公众领域。

Android 平台

Android 有丰富的功能，因此很容易与桌面操作系统混淆。Android 是一个分层的环境，构建在 Linux 内核的基础上，它包括丰富的功能。UI 子系统包括：

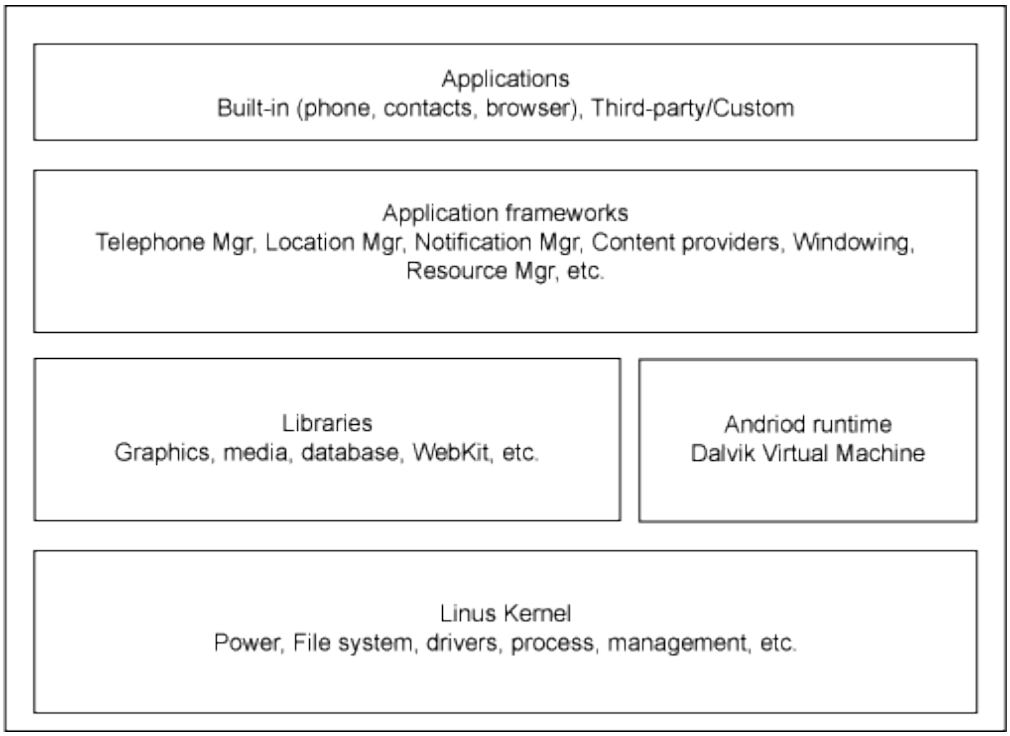
- 窗口
- 视图
- 用于显示一些常见组件（例如编辑框、列表和下拉列表）的小部件

Android 包括一个构建在 WebKit 基础上的可嵌入浏览器，iPhone 的 Mobile Safari 浏览器同样也是以 WebKit 为基础。

Android 提供多种连接选项，包括 WiFi、蓝牙和通过蜂窝（cellular）连接的无线数据传输（例如 GPRS、EDGE 和 3G）。Android 应用程序中一项流行的技术是链接到 Google 地图，以便在应用程序中显示地址。Android 软件栈还提供对基于位置的服务（例如 GPS）和加速计的支持，不过并不是所有的 Android 设备都配备了必需的硬件。另外还有摄像支持。

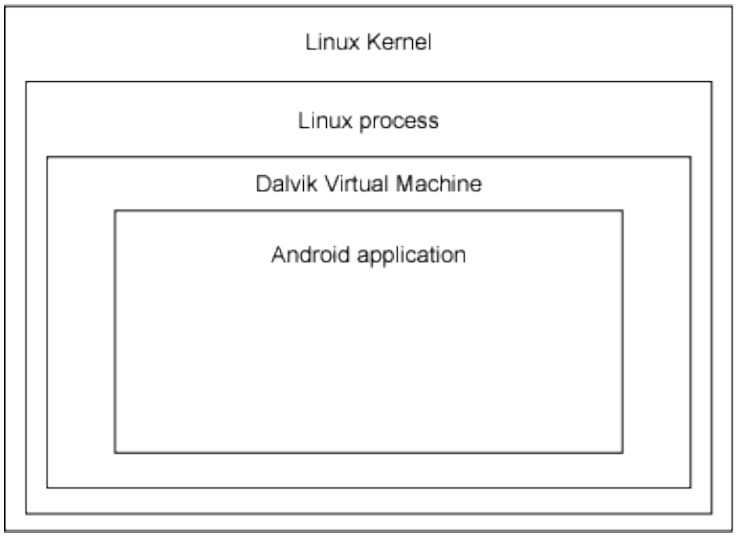
过去，移动应用程序努力向桌面应用程序看齐的两个领域分别是图形/媒体和数据存储方法。Android 通过提供对 2D 和 3D 图形的内置支持，包括 OpenGL 库，解决了图形方面的挑战。由于 Android 平台包括流行的开源 SQLite 数据库，因此缓解了数据存储的负担。图 1 显示一个简化的 Android 软件层次结构。

图 1. Android 软件层次结构



应用程序架构

如前所述，Android 运行在 Linux 内核上。Android 应用程序是用 Java 编程语言编写的，它们在一个虚拟机（VM）中运行。需要注意的是，这个 VM 并非您想象中的 JVM，而是 Dalvik Virtual Machine，这是一种开源技术。每个 Android 应用程序都在 Dalvik VM 的一个实例中运行，这个实例驻留在一个由 Linux 内核管理的进程中，如下图所示。



Android 应用程序由一个或多个组件组成：

活动

具有可视 UI 的应用程序是用活动实现的。当用户从主屏幕或应用程序启动器选择一个应用程序时，就会开始一个动作。

服务

服务应该用于任何需要持续较长时间的应用程序，例如网络监视器或更新检查应用程序。

内容提供程序

可以将内容提供程序看作数据库服务器。内容提供程序的任务是管理对持久数据的访问，例如 SQLite 数据库。如果应用程序非常简单，那么可能不需要创建内容提供程序。如果要构建一个较大的应用程序，或者构建需要为多个活动或应用程序提供数据的应用程序，那么可以使用内容提供程序实现数据访问。

广播接收器

Android 应用程序可用于处理一个数据元素，或者对一个事件（例如接收文本消息）做出响应。

Android 应用程序是连同一个 AndroidManifest.xml 文件一起部署到设备的。AndroidManifest.xml 包含必要的配置信息，以便将它适当地安装到设备。它包括必需的类名和应用程序能够处理的事件类型，以及运行应用程序所需的许可。例如，如果应用程序需要访问网络 — 例如为了下载一个文件 — 那么 manifest 文件中必须显式地列出该许可。很多应用程序可能启用了这个特定的许可。这种声明式安全性有助于减少恶意应用程序损害设备的可能性。

下一节讨论构建 Android 应用程序所需的开发环境。

所需的工具

开始开发 Android 应用程序的最简捷的方式是下载 Android SDK 和 Eclipse IDE（参见 参考资料）。Android 开发可以在 Microsoft® Windows®、Mac OS X 或 Linux 上进行。

本文假设您使用的是 Eclipse IDE 和用于 Eclipse 的 Android Developer Tools 插件。Android 应用程序是用 Java 语言编写的，但是是在 Dalvik VM（非 Java 虚拟机）中编译和执行的。在 Eclipse 中用 Java 语言编程非常简单；Eclipse 提供一个丰富的 Java 环境，包括上下文敏感帮助和代码提示。Java 代码通过编译后，Android Developer Tools 可确保适当地将它打包，包括 AndroidManifest.xml 文件。

虽然没有 Eclipse 和 Android Developer Tools 插件也可以开发 Android 应用程序，但是那样就需要熟悉 Android SDK。Android SDK 是作为一个 ZIP 文件发布的，可以将该文件解压到硬盘上的一个目录中。由于有多个 SDK 更新，建议有意识地组织开发环境，以便在不同的 SDK 安装之间轻松地切换。SDK 包括：

android.jar

Java 归档文件，其中包含构建应用程序所需的所有的 Android SDK 类。

documention.html 和 docs 目录

本地和网上提供的 SDK 文档。这些文档的主要形式为 JavaDocs，以便于在 SDK 中导航大量的包。文档还包括一个高级开发指南和 Android 社区的链接。

Samples 目录

samples 子目录包含各种应用程序的源代码，包括 ApiDemo，该应用程序演示了很多 API。这个示例应用程序可以作为 Android 应用程序开发的良好起点。

Tools 目录

包含所有用于构建 Android 应用程序的命令行工具。最常用、最有用的工具是 `adb` 实用程序（Android Debug Bridge）。

usb_driver

该目录包含将开发环境连接到支持 Android 的设备（例如 G1 或 Android Dev 1 解锁开发手机）所需的驱动程序。只有 Windows 平台的开发人员才需要这些文件。

Android 应用程序可以在实际的设备上运行，也可以在 Android SDK 附带的 Android Emulator 上运行。图 3 显示 Android Emulator 的主屏幕。

图 3. Android Emulator

Android Debug Bridge

adb 实用程序支持一些可选命令行参数，以提供强大的特性，例如复制文件到设备或从设备复制文件。可以使用 shell 命令行参数连接到手机本身，并发送基本的 shell 命令。图 4 显示在通过 USB 线连接到 Windows 笔记本电脑的一个实际设备上运行的 adb shell 命令。

图 4. 使用 adb shell 命令

```
C:\WINDOWS\system32\cmd.exe

M:\tools>adb -d shell
$

$ netcfg
netcfg
lo       UP    127.0.0.1       255.0.0.0       0x00000049
dummy0   DOWN  0.0.0.0         0.0.0.0         0x00000082
rmnet0   DOWN  25.1.184.133    255.255.255.252 0x00001002
rmnet1   DOWN  0.0.0.0         0.0.0.0         0x00001002
rmnet2   DOWN  0.0.0.0         0.0.0.0         0x00001002
tiwlan0  UP    192.168.2.105   255.255.255.0   0x00001043
$

$ echo $PATH
echo $PATH
/sbin:/system/sbin:/system/bin:/system/xbin
$

$ su
su
#

# cd /data/app
cd /data/app
#

# ls -l
ls -l
-rw-r--r-- system   system       8615 2009-03-22 18:38 com.msi.flashlight.apk
#

# ping google.com
ping google.com
PING google.com (74.125.45.100) 56(84) bytes of data.
64 bytes from yx-in-f100.google.com (74.125.45.100): icmp_seq=1 ttl=241 time=99.3 ms
64 bytes from yx-in-f100.google.com (74.125.45.100): icmp_seq=2 ttl=241 time=110 ms
64 bytes from yx-in-f100.google.com (74.125.45.100): icmp_seq=3 ttl=241 time=126 ms
^C
M:\tools>_
```

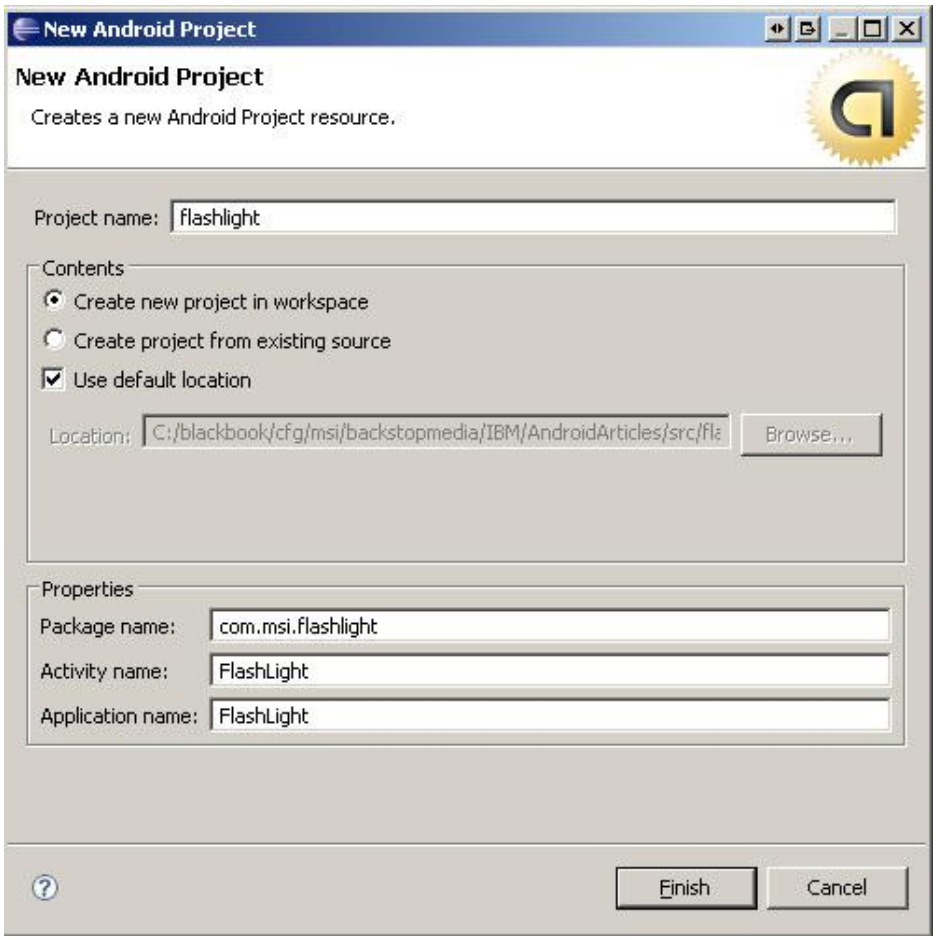
在这个 shell 环境中，可以：

- 显示网络配置，网络配置可显示多个网络连接。注意这多个网络连接：
  - lo 是本地或 loopback 连接。
  - tiwlan0 是 WiFi 连接，该连接由本地 DHCP 服务器提供一个地址。
- 显示 PATH 环境变量的内容。
- 执行 su 命令，以成为超级用户。
- 将目录改为 /data/app，其中存放用户应用程序。
- 列出包含某个应用程序的目录。Android 应用程序文件实际上是归档文件，可通过 WinZip 之类的软件查看。扩展名为 apk。
- 发出 ping 命令，查看 Google.com 是否可用。

从相同的命令提示符环境中，还可以与 SQLite 数据库交互，启动程序以及执行许多其他系统级任务。想像一下您正在连接到电话，因此这是非常了不起的功能。
在下一节，您将创建一个简单的 Android 应用程序。

编写一个基本的应用程序

本节展示如何构建一个 Android 应用程序。示例应用程序非常简单：一个修改后的“Hello Android”应用程序。您将进行一个微小的修改，使屏幕背景全部变为白色，以便把手机用作手电筒。这个例子不是很有创意，但是可以作为一个有用的例子。请 下载 完整的源代码。
为了在 Eclipse 中创建应用程序，选择 **File > New > Android project**，这将启动 New Android Project 向导。



接下来，创建一个简单的应用程序，该应用程序有一个活动，并且在 main.xml 中有一个 UI 布局。布局包含一个文本元素，您将修改这个文本元素，以显示 Android FlashLight。下面的清单显示了这个简单的布局。

清单 1. Flashlight 布局

```
<?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"
    android:background="@color/all_white">
<TextView
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="@string/hello" android:textColor="@color/all_black"
   android:gravity="center_horizontal"/>
</LinearLayout>
```

在 strings.xml 中创建两个颜色资源。

清单 2. strings.xml 中的颜色

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="hello">Android FlashLight</string>
    <string name="app_name">FlashLight</string>
    <color name="all_white">#FFFFFF</color>
    <color name="all_black">#000000</color>
</resources>
```

主屏幕布局有一个定义为 `all_white` 的背景色。在 strings.xml 文件中，可以看到 `all_white` 被定义为一个值为 #FFFFFF 的 RGB 三元组，即纯白。
布局包含一个 `TextView`，这实际上是一块静态文本。它是不可编辑的。文本被设为黑色，并通过 `gravity` 属性设为水平居中。
该应用程序有一个名为 FlashLight.java 的 Java 源文件，如以下清单所示。

清单 3. Flashlight.java

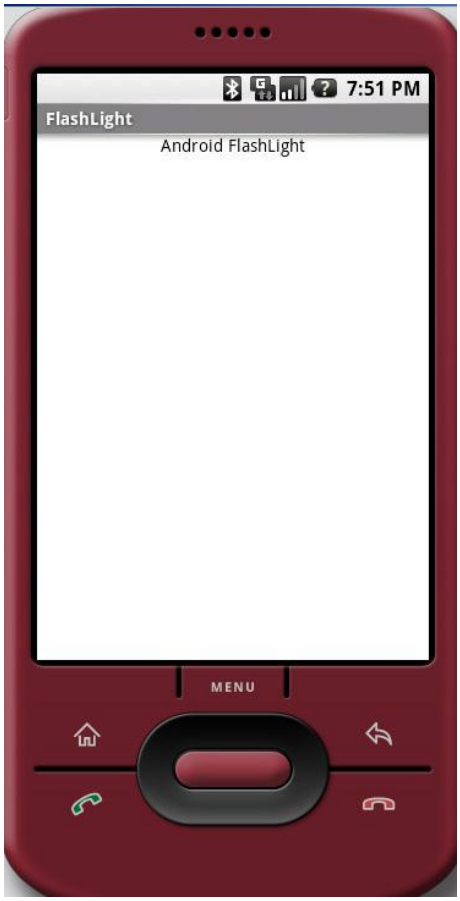
```
package com.msi.flashlight;
import android.app.Activity;
import android.os.Bundle;
public class FlashLight extends Activity {
    /** Called when the activity is first created. */
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
    }
```

```
}
```

该代码是直接来自 New Project 向导的模板代码：

- 它是 Java 包 com.msi.flashlight 的一部分。
- 它有两个 import：
  - 一个用于 activity 类
  - 一个用于 bundle 类
- 当该活动发起后，`onCreate` 方法被调用，传入一个 `savedInstanceState`。对于我们来说，不必关心这个 bundle。只有在暂停然后恢复活动时才会用到。
- `onCreate` 方法覆盖了同名的 activity 类方法。它调用超类的 `onCreate` 方法。
- 对 `setContentView()` 的调用将关联 main.xml 文件中定义的 UI 布局。main.xml 和 strings.xml 中的任何内容都自动映射到 R.java 源文件中定义的常量。任何时候都不要直接编辑这个文件，因为它随着每次构建而改变。

运行该应用程序可以看到一个白色屏幕，其中有黑色文本。



下面显示用于 FlashLight 应用程序的 AndroidManifest.xml 文件。

清单 4. 用于 FlashLight 的 AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
      package="com.msi.flashlight"
      android:versionCode="1"
      android:versionName="1.0.0">
    <application android:icon="@drawable/icon" android:label="@string/app_name">
```

```
      <activity android:name=".FlashLight"
              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>
</manifest>
```

该文件是由用于 Eclipse 的 Android Developer Tools 插件自动创建的。您不需要做任何事情。
当然，这个应用程序并不是很强大。但是如果希望读点书，又不想打扰正在睡觉的爱人，或者如果需要在断电时去地下室查看保险丝盒，那么这个应用程序还是很方便的。

结束语

在本文中，您阅读了对 Android 的简介，并构建了一个小型的应用程序。希望本文中的例子能激起您进一步探索 Android 平台的兴趣。Android 有望成为对市场产生深远影响的开源平台，它的用处将远远超越移动电话。

手机上的 Scala

developerWorks.

使用 Android、Scala 和 Eclipse 创建移动应用程序

Android 操作系统为移动开发提供强大、开放的平台。它利用了 Java™ 编程语言和 Eclipse 工具平台的威力。现在，还可以将 Scala 编程语言加入到其中。在本文中，您将看到如何使用 Scala 作为 Android 上的主要开发语言，从而可以使用一种更具表达力、更加类型安全的编程语言编写移动应用程序。

先决条件

在本文中，我们将创建一个在 Android 设备上运行的移动应用程序。您将需要安装 Android SDK；本文使用 V1.5 SDK。应用程序代码将用 Scala 编程语言编写。如果您从来没用过 Scala，那么没有关系，因为本文将解释 Scala 代码。但是，即使您不熟悉 Scala，建议您至少熟悉 Java 语言。本文使用 Scala V2.7.5 进行开发。对于 Android 和 Scala 都提供了很好的 Eclipse 插件。本文使用 Eclipse V3.4.2 和 Android Development Tools（ADT） V0.9.1 以及 Scala IDE 插件 V2.7.5。请参阅 参考资料，获得所有这些工具。

设置

编写 Android 应用程序听起来像是一个复杂的命题。Android 应用程序在它们自己的虚拟机中运行：Dalvik 虚拟机。但是，Android 应用程序的构建路径是开放的。下面表明了我们将使用的基本策略。

图 1. Android 上 Scala 的构建路径

其思想是，我们首先将所有 Scala 代码编译成 Java 类文件。这是 Scala 编译器的工作，所以这方面没什么太复杂的事情。接下来，获取 Java 类文件，使用 Android dex 编译器将类文件编译成 Android 设备上的 Dalvik VM 使用的格式。这就是所谓的 *dexing*，也是 Android 应用程序的常规编译路径。通常，要经历从 .java 文件到 .class 文件再到 .dex 文件的过程。在本文，惟一不同的是我们从 .scala 文件开始。最后，.dex 文件和其他应用程序资源被压缩成一个 APK 文件，该文件可安装到 Android 设备上。

那么，如何让这一切发生？我们将使用 Eclipse 做大部分工作。但是，此外还有一个较复杂的步骤：要让代码运行，还需要来自标准 Scala 库中的代码。在典型的 Scala 安装中，这是 /lib/scala-library.jar 中一个单独的 JAR。但是，这个 JAR 包括一些不受 Android 支持的代码。有些代码需要稍作调整，有些代码则必须移除。scala-library.jar 的定制构建是运行得最好的，至少目前是这样。请参阅 参考资料，了解这里使用的定制构建。我们将把这个 JAR 称作 Android 库 JAR。

有了这个 JAR，剩下的事情就很容易了。只需使用 Eclipse 的 ADT 插件创建一个 Android 项目。然后将一个 Scala 特性（nature）添加到项目中。用前面谈到的 Android 库替代标准的 Scala 库。最后，将输出目录添加到类路径中。现在，可以开始了。主 Scala 站点对此有更详细的描述（请参阅 参考资料）。现在，我们有了基本的设置，接下来看看我们将使用 Scala 创建的 Android 应用程序。

UnitsConverter

现在，我们知道如何利用 Scala 代码，将它转换成将在 Android 设备上运行的二进制格式，接下来可以使用 Scala 创建一个移

动应用程序。我们将创建的应用程序是一个简单的单位转换应用程序。通过这个应用程序可以方便地在英制单位与公制单位之间来回转换。这是一个非常简单的应用程序，但是我们将看到，即使是最简单的应用程序也可以从使用 Scala 中获益。我们首先看看 UnitsConverter 的布局元素。

创建布局

您也许对编写手机上运行的 Scala 感到兴奋，但是并非所有的移动开发编程都应该用 Scala 或 Java 语言完成。Android SDK 提供了一种很好的方式，使用基于 XML 的布局系统将用户界面代码与应用程序逻辑分离。我们来看看本文中的应用程序的主要布局文件，如清单 1 所示。

清单 1. Converter 应用程序的主要布局

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent" android:layout_height="fill_parent"
    android:gravity="center_horizontal" android:padding="10px"
    >
    <TextView android:id="@+id/prompt_label" android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/prompt_metric"/>
    <EditText android:id="@+id/amount" android:layout_below="@id/prompt_label"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"/>
    <TextView android:id="@+id/uom_label"
        android:layout_below="@id/amount"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/uom"/>
    <Spinner android:id="@+id/uom_value"
        android:layout_below="@id/uom_label"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"/>
    <Button android:id="@+id/convert_button"
        android:layout_below="@id/uom_value"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/convert_button_label"/>
    <TextView android:id="@+id/result_value"
        android:layout_below="@id/convert_button"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"/>
</RelativeLayout>
```

以上代码非常简洁地创建了该应用程序的主 UI。它的根节点是一个 RelativeLayout 容器元素。Android SDK 中有很多布局选项。RelativeLayout 指示运行时使用相对定位对不同的 UI 小部件进行布局。要使用相对定位，可添加可见元素 — 在这里是一个 TextView 元素。这是用于显示文本的一个简单的元素。它被赋予一个 ID prompt_label。接下来的元素，即一个 EditText 元素（一个文本输入框）将用到它。这个元素有一个 layout_below 属性，它的值等于 prompt_label ID。换句话说，EditText 应该放在名为 prompt_label 的元素的下方。

布局代码剩下的部分非常简单。有一个带标签的文本输入框、一个带标签的微调器（一个组合框或下拉框）、一个按钮和一个用于输

出的文本框。图 2 显示正在运行的应用程序的一个截图，其中标出了不同的元素。

图 2. Android lLayout — 分解图

那么，以上视图中看到的不同文本值来自哪里呢？注意，清单 1 中的一些元素有一个 text 属性。例如，`prompt_label` 元素有一个等于 `@string/prompt_metric` 的 text 属性。这表明它将使用 Android 应用程序中一个标准的资源文件：strings.xml 文件，如清单 2 所示。

清单 2. strings.xml 资源

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="prompt_metric">Enter amount (KM, g, L, C)</string>
    <string name="prompt_english">Enter amount (miles, lbs, gallons,
F)</string>
    <string name="uom">Units of Measure</string>
    <string name="convert_button_label">Convert</string>
    <string name="app_name">Converter</string>
    <string name="english_units">English</string>
    <string name="metric_units">Metric</string>
</resources>
```

现在可以看到，图 2 中所有的文本来自何处。微调器有一个下拉框，其中包含可用于度量的单位，那些单位在清单 2 中没有列出。相反，它们来自另一个文件 arrays.xml，如清单 3 所示。

清单 3. arrays.xml 资源

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <array name="english_units">
        <item>Fahrenheit</item>
        <item>Pounds</item>
        <item>Ounces</item>
        <item>Fluid Ounces</item>
        <item>Gallons</item>
        <item>Miles</item>
        <item>Inches</item>
    </array>
    <array name="metric_units">
        <item>Celsius</item>
        <item>Kilograms</item>
        <item>Grams</item>
        <item>Millileters</item>
        <item>Liters</item>
        <item>Kilometers</item>
        <item>Centimeters</item>
    </array>
</resources>
```

现在，我们可以看到将用于微调器的那些值。那么，这些值如何出现在微调器中，应用程序如何在英制单位与公制单位之间切换？要回答这些问题，我们需要看看应用程序代码本身。

Scala 应用程序代码

Converter 应用程序的代码非常简单 — 不管用什么语言编写。当然，用 Java 编写起来非常容易，但是用 Scala 编写也同样不复杂。首先我们看看前面见过的 UI 背后的代码。

视图背后的代码

解释创建 UI 的 Scala 代码的最简单方式是先看看代码，然后走查一遍。对于任何应用程序，都是在应用程序的 AndroidManifest.xml 文件中定义应用程序的默认活动。任何 UI 背后都有一个 `Activity` 类，默认的 `Activity` 定义当应用程序初次装载时执行的 `Activity` 类。对于像本文这样简单的应用程序，有一个 `Converter` 类，清单 4 中显示了它的源代码。

清单 4. `Converter` 活动类

```
class Converter extends Activity{
    import ConverterHelper._
    private[this] var amountValue:EditText = null
    private[this] var uom:Spinner= null
    private[this] var convertButton:Button = null
    private[this] var resultValue:TextView = null

    override def onCreate(savedInstanceState:Bundle){
      super.onCreate(savedInstanceState)
      setContentView(R.layout.main)
      uom = findViewById(R.id.uom_value).asInstanceOf[Spinner]
      this.setUomChoice(ENGLISH)
      amountValue = findViewById(R.id.amount).asInstanceOf[EditText]
      convertButton = findViewById(R.id.convert_button).asInstanceOf[Button]
      resultValue = findViewById(R.id.result_value).asInstanceOf[TextView]
      convertButton.setOnClickListener( () => {
          val unit = uom.getSelectedItem.asInstanceOf[String]
          val amount = parseDouble(amountValue.getText.toString)
          val result = UnitsConverter.convert(Measurement(unit,amount))
          resultValue.setText(result)
      })
    }
    override def onCreateOptionsMenu(menu:Menu) = {
      super.onCreateOptionsMenu(menu)
      menu.add(NONE, 0, 0, R.string.english_units)
      menu.add(NONE, 1, 1, R.string.metric_units)
      true
    }
    override def onMenuItemSelected(featureId:Int, item:MenuItem) = {
      super.onMenuItemSelected(featureId, item)
      setUomChoice(if (item.getItemId == 1) METRIC else ENGLISH)
      true
    }
    private
```

```
    def setUomChoice(unitOfMeasure:UnitsSystem){
      if (uom == null){
        uom = findViewById(R.id.uom_value).asInstanceOf[Spinner]
      }
      val arrayId = unitOfMeasure match {
        case METRIC => R.array.metric_units
        case _ => R.array.english_units
      }
      val units = new ArrayAdapter[String](this, R.layout.spinner_view,
        getResources.getStringArray(arrayId))
      uom.setAdapter(units)
    }
}
```

我们从这个类的顶部开始。它扩展 `android.app.Activity`。这是一个 Java 类，但是从 Scala 中可以对 Java 类轻松地进行细分。接下来，它有一些实例变量。每个实例变量对应前面定义的一个 UI 元素。注意，每个实例变量还被限定为 `private[this]`。这演示了 Scala 中特有的一种访问控制级别，而 Java 语言中不存在这种访问控制。这些变量不仅是私有的，而且只属于 `Converter` 类的特定实例。这种级别的访问控制对于移动应用程序来说有些大材小用，但是如果您是一名 Scala 开发人员，可以放心地在 Android 应用程序上使用您熟悉的语法。

回到清单 4 中的代码，注意，我们覆盖了 `onCreate` 方法。这是 `Activity` 类中定义的方法，通常被定制的 `Activity` 覆盖。如果用 Java 语言编写该代码，那么应该添加一个 `@Override` 标注。在 Scala 中，override 是一个关键词，用于确保正确性。这样可以防止误拼方法名之类的常见错误。如果误拼了方法名，Scala 编译器将捕捉到方法名并返回一个错误。注意，在这个方法上，以及任何其他方法上，不需要声明返回类型。Scala 编译器可以轻松推断出该信息，所以不需要多此一举。

`onCreate` 中的大部分代码类似于 Java 语言编写的代码。但是有几点比较有趣。注意，我们使用 `findViewById` 方法（在 `Activity` 子类中定义）获得不同 UI 元素的句柄。这个方法不是类型安全的，需要进行类型转换（cast）。在 Scala 中，要进行类型转换，可使用参数化方法 `asInstanceOf[T]`，其中 `T` 是要转换的类型。这种转换在功能上与 Java 语言中的转换一样。不过 Scala 有更好的语法。接下来，注意对 `setUomChoice` 的调用（稍后我们将详细谈到这个方法）。最后，注意上述代码获得一个在布局 XML 中创建的按钮的句柄，并添加一个单击事件处理程序。

如果用 Java 语言编写，那么必须传入 Android 接口 `OnClickListener` 的一个实现。这个接口只定义一个方法：`onClick`。实际上，您关心的只是那个方法，但是在 Java 语言中无法直接传入方法。而在 Scala 中则不同，在 Scala 中可以传入方法字面量（literal）或闭包。在这里，我们用语法 `() => { ... }` 表示闭包，其中方法的主体就是花括号中的内容。开始/结束括号表示一个不带参数的函数。但是，我将这个闭包传递到 `Button` 的一个实例上的 `setOnClickListener` 方法，`Button` 是 Android SDK 中定义的一个 Java 类。如何将 Scala 闭包传递到 Java API？我们来看看。

Android 上的函数式编程

为了理解如何让 Android API 使用函数字面量，看看 `Converter` 类定义的第一行。这是一条重要的语句。这是 Scala 的另一个很好的特性。您可以在代码的任何地方导入包、类等，它们的作用域限于导入它们的文件。在这里，我们导入 `ConverterHelper` 中的所有东西。清单 5 显示 `ConverterHelper` 代码。

清单 5. `ConverterHelper`

```
object ConverterHelper{
  import android.view.View.OnClickListener
```

```
  implicit def funcToClicker(f:View => Unit):OnClickListener =
    new OnClickListener(){ def onClick(v:View)=f.apply(v)}
  implicit def funcToClicker0(f:() => Unit):OnClickListener =
    new OnClickListener() { def onClick(v:View)=f.apply}
}
```

这是一个 Scala 单例（singleton），因为它使用对象声明，而不是类声明。单例模式被直接内置在 Scala 中，可以替代 Java 语言中的静态方法或变量。在这里，这个单例存放一对函数：funcToClicker 和 funcToClicker0。这两个函数以一个函数作为输入参数，并返回 OnClickListener 的一个实例，OnClickListener 是 Android SDK 中定义的一个接口。例如，funcToClicker 被定义为以一个函数 f 为参数。这个函数 f 的类型为带一个 View 类型（Android 中的另一个类）的输入参数的函数，并返回 Unit，它是 void 在 Scala 中的对等物。然后，它返回 OnClickListener 的一个实现，在这个实现中，该接口的 onClick 方法被实现为将输入函数 f 应用到 View 参数。另一个函数 funcToClick0 也做同样的事情，只是以一个不带输入参数的函数为参数。

这两个函数（funcToClicker 和 funcToClicker0）都被定义为隐式函数（implicit）。这是 Scala 的一个方便的特性。它可以让编译器隐式地将一种类型转换成另一种类型。在这里，当编译器解析 Converter 类的 onCreate 方法时，它遇到一个 setOnClickListener 调用。这个方法需要一个 OnClickListener 实例。但是，编译器却发现一个函数。在报错并出现编译失败之前，编译器将检查是否存在隐式函数，允许将函数转换为 OnClickListener。由于确实还有这样的函数，所以它执行转换，编译成功。现在，我们理解了如何使用 Android 中的闭包，接下来更仔细地看看应用程序逻辑 — 特别是，如何执行单位转换计算。

单位转换和计算

我们回到清单 4。传入 onClickListener 的函数收到用户输入的度量单位和值。然后，它创建一个 Measurement 实例，并将该实例传递到一个 UnitsConverter 对象。清单 6 显示相应的代码。

清单 6. Measurement 和 UnitsConverter

```
case class Measurement(uom:String, amount:Double)

object UnitsConverter{
     // constants
   val lbToKg = 0.45359237D
     val ozToG = 28.3495231
     val fOzToMl = 29.5735296
     val galToL = 3.78541178
     val milesToKm = 1.609344
     val inchToCm = 2.54

     def convert (measure:Measurement)= measure.uom match {
         case "Fahrenheit" => (5.0/9.0)*(measure.amount - 32.0) + " C"
          case "Pounds" => lbToKg*measure.amount + " kg"
          case "Ounces" => ozToG*measure.amount + " g"
          case "Fluid Ounces" => fOzToMl*measure.amount + " mL"
          case "Gallons" => galToL*measure.amount + " L"
          case "Miles" => milesToKm*measure.amount + " km"
          case "Inches" => inchToCm*measure.amount + " cm"
          case "Celsius" => (9.0/5.0*measure.amount + 32.0) + " F"
          case "Kilograms" => measure.amount/lbToKg + " lbs"
          case "Grams" => measure.amount/ozToG + " oz"
          case "Millileters" => measure.amount/fOzToMl + " fl. oz."
          case "Liters" => measure.amount/galToL + " gallons"
```

```
            case "Kilometers" => measure.amount/milesToKm + " miles"
            case "Centimeters" => measure.amount/inchToCm + " inches"
            case _ => ""
        }
}
```

Measurement 是一个 case 类。这是 Scala 中的一个方便的特性。用“case”修饰一个类会导致这个类生成这样一个构造函数：这个构造函数需要类的属性，以及 equals、 hashCode 和 toString 的实现。它对于像 Measurement 这样的数据结构类非常适合。它还为定义的属性（在这里就是 uom 和 amount）生成 getter 方法。也可以将那些属性定义为 vars（可变变量），然后也会生成 setter 方法。仅仅一行 Scala 代码可以做这么多事情！

接下来，UnitsConverter 也是一个单例模式，因为它是使用 object 关键词定义的。它只有一个 convert 方法。注意，convert 被定义为相当于一条单一语句 — 一条 match 语句。它是一个单一表达式，所以不需要额外的花括号。它使用 Scala 的模式匹配。这是函数式编程语言中常见的一个强大特性。它类似于 Java 语言和很多其他语言中的 switch 语句。但是，我们可以匹配字符串（实际上，还可以有比这高级得多的匹配）。如果字符串匹配，则执行适当的计算，并返回格式化的字符串，以供显示。最后，注意与 _ 匹配的最后一个 case。Scala 中的很多地方使用下划线作为通配符。在这里，它表示匹配任何东西，这类似于 Java 语言中的 default 语句。

现在，我们理解了应用程序中的计算，最后来看看剩下的 UI 设置和菜单。

UI 初始化和菜单

回到清单 4。我们说过要看看 setUomChoice。这个方法被定义为带有一个 UnitsSystem 类型的参数。我们来看看如何定义这个类型。

清单 7. UnitsSystem

```
sealed case class UnitsSystem()
case object ENGLISH extends UnitsSystem
case object METRIC extends UnitsSystem
```

我们看到，UnitsSystem 是一个密封的 case 类，没有属性。看上去它不是很有用。接下来，我们看看两个 case 对象。还记得吗，object 表示 Scala 中的一个单例。在这里，有两个 case 对象，每个 case 对象都扩展 UnitsSystem。这是 Scala 中的一个常见的特色，它可以提供更简单、更类型安全的枚举方式。

现在 setUomChoice 的实现更加合理。在获得微调器的一个句柄后，我们匹配传入的 UnitsSystem 的类型。这标识了我们在前面见到的 arrays.xml 中的一个数组。这是使用 Android SDK 生成的 R 类表示资源，例如 arrays.xml 文件。一旦知道使用哪个数组，我们就通过创建一个传入微调器的适配器（在这里是一个 ArrayAdapter），使用那个数组作为微调器的数据源。

最后，看看清单 4 中的 onCreateOptionsMenu 和 onMenuItemSelected 方法。这些方法是在 Activity 中定义的，我们将在 Converter 活动中覆盖这些方法。第一个方法创建一个菜单。第二个方法处理用户从菜单中选择 English 或 metric 的事件。它再次调用 setUomChoice。这使用户可以在从英制单位转换为公制单位与从公制单位转换为英制单位之间进行切换。

结束语

Android 平台的架构使它可以用于在 Java 虚拟机上运行的任何编程语言。我们看到了如何设置 Android 项目，使它使用 Scala 代码。这个过程也可以延伸到其他 JVM 编程语言，例如 Groovy、JRuby 或 Fan。当可以任意使用 Scala 编程语言时，编写 Android 应用程序将变得更轻松。您仍可以使用 Eclipse 进行开发。仍然可以在 Eclipse 中用模拟器和设备进行调试。您可以继续使用所有的工具，同时又得到一种生产率更高的编程语言。

使用 Android 实现联网

developerWorks.

连接到您的世界

Android 是面向应用程序开发的丰富平台，它提供一套出色的用户界面元素和数据管理功能。它还为开发连接到真实世界的应用程序提供了出色的网络选项。您希望连接到哪里？也许您希望告诉全世界您正在使用 Twitter。或者希望收到有关在本地洗衣店或熟食店的排号。如果希望将 Android 连接到您所生活的世界，那么请阅读本文。

简介

本文建立在“使用 Eclipse 开发 Android 应用程序”一文的基础之上，探究了 Android 的网络功能。了解如何利用 Android 的网络选项来实现有趣、有用的东西。Android 平台非常适合 Java™ 开发人员：他们可以使用已有的技能将网络连接带到一个移动或“嵌入式”平台中。

在本文中，了解用于 Android 应用程序的网络选项以及基本的 Android 联网技巧。本文研究一个真实的应用程序，它在结合使用环境监视系统时需要具备联网功能。这类系统为什么如此重要？原因之一是：如果您的朋友需要外出几个星期，在他离开后，他打电话给我，让我从他家里找到某样东西并邮寄给他。我来到他的家里，发现供暖设备已经被切断并且水管已经冻裂 — 场面非常混乱。如果备有一个温度监控系统，那么就可以避免出现这类事故。本文将探查 Android 在这类监控系统中扮演的角色。

Android 联网功能

Android 基于 Linux® 内核，包含一组优秀的联网功能。如果尚未安装 Android SDK，那么需要 下载 它才能实践本文的示例。表 1 展示了 Android SDK 中一些与网络有关的包。

表 1. Android SDK 网络包

包	描述
java.net	提供与联网有关的类，包括流和数据包（datagram）sockets、Internet 协议和常见 HTTP 处理。该包是一个多功能网络资源。有经验的 Java 开发人员可以立即使用这个熟悉的包创建应用程序。
java.io	虽然没有提供显式的联网功能，但是仍然非常重要。该包中的类由其他 Java 包中提供的 socket 和连接使用。它们还用于与本地文件（在与网络进行交互时会经常出现）的交互。
java.nio	包含表示特定数据类型的缓冲区的类。适合用于两个基于 Java 语言的端点之间的通信。
org.apache.*	表示许多为 HTTP 通信提供精确控制和功能的包。可以将 Apache 视为流行的开源 Web 服务器。
android.net	除核心 java.net.* 类以外，包含额外的网络访问 socket。该包包括 URI 类，后者频繁用于 Android 应用程序开发，而不仅仅是传统的联网方面。
android.net.http	包含处理 SSL 证书的类。
android.net.wifi	包含在 Android 平台上管理有关 WiFi（802.11 无线 Ethernet）所有方面的类。并不是所有设备都配备了 WiFi 功能，特别是 Android 在 Motorola 和 LG 等手机制造商的“翻盖手机”领域获得了成功。
android.telephony.gsm	包含用于管理和发送 SMS（文本）消息的类。一段时间后，可能会引入额外的包来来为非 GSM 网络提供类似的功能，比如 CDMA 或 android.telephony.cdma 等网络。

上表并没有列出所有包，但是可以让您清楚地意识到该平台的强大功能。下一小节将介绍一些简单的网络示例。

简单的网络示例

为了演示将 Android 连接到一个网络有多么简单，这个示例将展示如何从 Web 页面发送文本。可以 下载 本例的源代码。图 1 展示了应用程序的实际使用。

图 1. 从 Web 页面获取文本

本节提供了构建示例应用程序所需的代码。我们将首先查看 UI 部分，然后介绍与网络有关的代码。
共有三个 UI 元素：

- EditText 让用户能够进入一个 Web 页面（图 1 和 清单 2 所示的 http://developer.android.com）。
- 使用一个按钮告诉程序取回 Web 页面文本。
- 检索回数据后，它将显示在 TextView 中。

清单 1 展示了 main.xml 文件，这是该应用程序的完整 UI 布局。

清单 1. 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"
    >
<EditText
android:layout_height="wrap_content"
android:id="@+id/address"
android:layout_width="fill_parent"
android:text="http://google.com"
>
</EditText>
<Button
 android:id="@+id/ButtonGo"
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:text="go!"
 >
</Button>
<TextView
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:background="#ffffff"
    android:textColor="#000000"
          android:id="@+id/pagetext"
    />
</LinearLayout>
```

清单 2 展示了本示例使用的 Java 代码。

清单 2. GetWebPage.java

```
package com.msi.getwebpage;

import android.app.Activity;
import android.os.Bundle;
// used for interacting with user interface
import android.widget.Button;
import android.widget.TextView;
import android.widget.EditText;
import android.view.View;
// used for passing data
import android.os.Handler;
```

```
import android.os.Message;
// used for connectivity
import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.net.URL;
import java.net.URLConnection;


public class GetWebPage extends Activity {
    /** Called when the activity is first created. */

    Handler h;

          @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        final EditText eText = (EditText) findViewById(R.id.address);
        final TextView tView = (TextView) findViewById(R.id.pagetext);

        this.h = new Handler() {

            @Override
            public void handleMessage(Message msg) {
                // process incoming messages here
                switch (msg.what) {
                    case 0:
                     tView.append((String) msg.obj);
                     break;
                }
                super.handleMessage(msg);
            }
        };
        final Button button = (Button) findViewById(R.id.ButtonGo);
        button.setOnClickListener(new Button.OnClickListener() {
            public void onClick(View v) {
                    try      {
                             tView.setText("");
                // Perform action on click
                     URL url = new URL(eText.getText().toString());
                    URLConnection conn = url.openConnection();
                    // Get the response
                    BufferedReader rd = new BufferedReader(new
InputStreamReader(conn.getInputStream()));
                    String line = "";
                    while ((line = rd.readLine()) != null) {
```

```
                        Message lmsg;
                    lmsg = new Message();
                    lmsg.obj = line;
                    lmsg.what = 0;
                    GetWebPage.this.h.sendMessage(lmsg);
                }
                 }
                 catch (Exception e)        {
                 }
          }
       });
   }
}
```

这些代码可以分解成一些常见的部分。这里使用一些重要（必需）的导入语句来恰当地引用 UI、数据传递以及应用程序中使用的与网络有关的类。所有与网络相关的代码出现在 `OnClickListener` 的 `OnClick` 方法中。在选择 图 1 所示的标签为 `go!` 的按钮之后调用这些代码。

`URL` 和 `URLConnection` 类共同提供与用户所选的 Web 站点的连接。`BufferedReader` 的一个实例负责从 Web 站点连接中读取传入的数据。每读取一行代码，文本就被附加到一个 TextView。数据并没有直接指定给 TextView（但是在本例中可以）。我们引入了一种设计模式，即创建一个消息对象并将该对象发送到一个处理程序的实例。这是更新 UI 的一种比较可取的方法，对可能需要同时运行多个线程的应用程序而言尤其如此。

在示例中，Android 应用程序与 HTTP Web 服务器进行通信，比如 Apache 或 Internet Information Server（IIS 位于 Microsoft® 服务器上）。如果应用程序直接与 TCP socket 对话，那么您将以不同的方式实现应用程序。清单 3 所示的代码片段展示了另一种与远程服务器交互的方式。这个清单被实现为一个单独的线程。

清单 3. Daytime 客户机

```
   public class Requester extends Thread {
      Socket requestSocket;
      String message;
      StringBuilder returnStringBuffer = new StringBuilder();
      Message lmsg;
      int ch;
      @Override
      public void run() {
         try {
            this.requestSocket = new Socket("remote.servername.com", 13);
            InputStreamReader isr = new InputStreamReader(this.requestSocket.
getInputStream(), "ISO-8859-1");
            while ((this.ch = isr.read()) != -1) {
               this.returnStringBuffer.append((char) this.ch);
            }
            this.message = this.returnStringBuffer.toString();
            this.lmsg = new Message();
            this.lmsg.obj = this.message;
            this.lmsg.what = 0;
            h.sendMessage(this.lmsg);
```

```
            this.requestSocket.close();
        } catch (Exception ee) {
            Log.d("sample application", "failed to read data" + ee.getMessage());
        }
    }
}
```

与前面的示例类似，上面的代码使用消息和处理程序方法来将数据发送给调用者，调用者将更新 UI 并执行后续处理。与 清单 1 不同，这个例子并没有与 HTTP 服务器通信，因此没有使用 `URLConnection` 类。相反，使用了较低级的 `Socket` 类在端口 13 打开与远程服务器的基于流的 socket 连接。端口 13 是典型的 “Daytime Server” 应用程序。
Daytime Server 接受传入的 socket 连接并以文本的形式将日期和时间发送给调用 socket。一旦发送完数据，服务器将关闭 socket。示例也展示了 `InputStreamReader` 的使用和一个特定字符编码。
发送文本消息是您需要使用 Android 完成的另一项任务。清单 4 展示了一个示例。

清单 4. 发送一条文本消息

```
void sendMessage(String recipient,String myMessage) {
 SmsManager sm = SmsManager.getDefault();
 sm.sendTextMessage("destination number",null,"hello there",null,null);
}
```

发送文本消息非常简单。首先，使用静态方法 `getDefault()` 获取对 SmsManager 的引用。然后调用 `sendTextMessage` 方法。参数为：

接收者的手机号

包括区号。

服务中心电话号码

使用 null 值表示您同意使用默认服务中心来处理消息。除了非常特殊的应用程序外，几乎所有应用程序都对这个参数使用 null 值。

消息的实际内容

将消息长度保持在 160 字节以内，除非您可以接受将数据分为多个消息发送。

未收到消息 intent

如果消息被发送或出现了错误，那么将开始一个可选的 intent。如果不需要这类通知，那么可以为此参数传递一个 null 值。（参见 参考资料 了解有关 intent 和 Android 基本原理的更多信息）。

收到消息 intent

当收到发送确认后，将开始一个可选的 Intent。如果发送通知不重要的话，那么可以为这个参数传递一个 null 值。

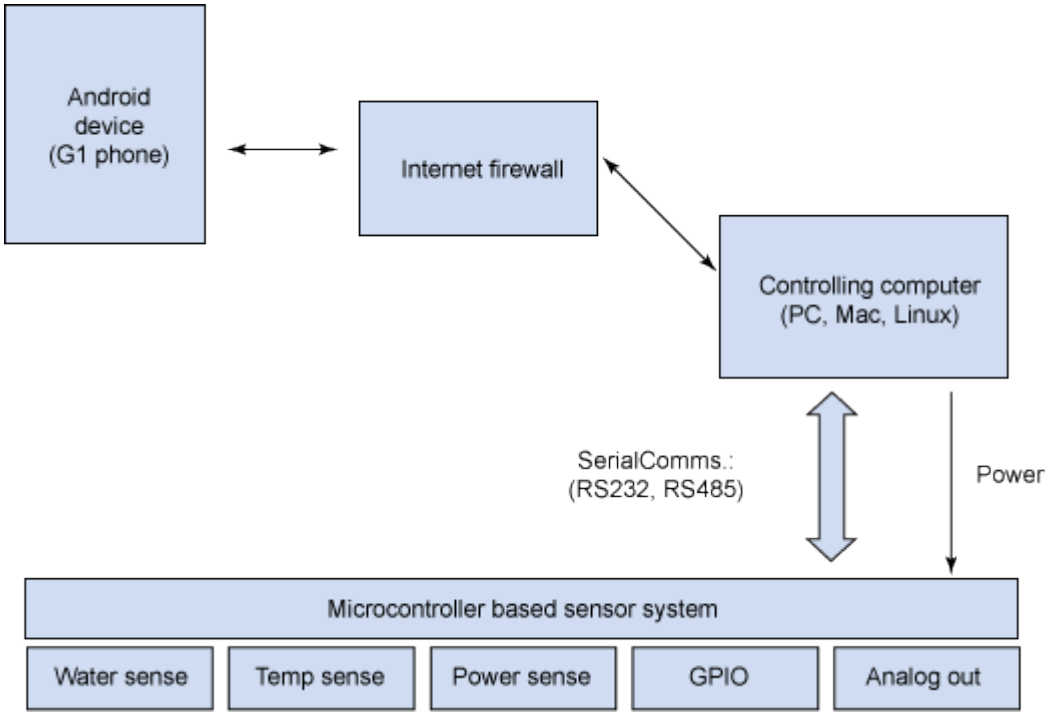
不管是连接到 Web 页面还是连接到定制 TCP 应用程序，Android 平台都可以立即反应并且能够提供帮助。如 清单 4 所示，发送文本消息非常简单。通过使用可选的 intent 参数，甚至可以在消息被发送并交付后采取操作。这是其他移动平台所不具备的强大特性。
下一节将快速浏览一个真实的应用程序设计。

环境监控系统

在这个场景中，我们假设您是企业所在的若干办公场所的资产管理员。管理资产与管理数据中心没有太大的差别 — 一般情况下都很枯燥，只有出现紧急的情况下工作才会比较有意思。几天前，一台使用了 10 年的热水器突然漏水，渗到一个装满老式 PC 和培

训手册的存储柜，您必须检查一下清理情况。幸运的是，您当时没有外出。如果您在旅途中的话，那么情形将非常糟糕。此类灾难性事故促使我们考虑使用 Android 来帮助监视资产的维护情况。图 2 展示了此类系统的一个高级方框图。

图 2. 监控系统的高级方框图



此架构是一种比较传统的方法，使用一个微控制器与一些简单场景进行交互以收集数据。数据随后通过一个串行通信协议（比如 RS232 或 RS485）发送到控制器。控制器可以是一个 PC 或类似的机器。随后可以穿过防火墙通过 Internet 访问数据。Android 电话（比如 TMobile G1）之间使用的协议可以是 HTTP 或私有协定。
在控制器和配备 Android 的设备之间发送的数据将是表示以下内容的基本数据：

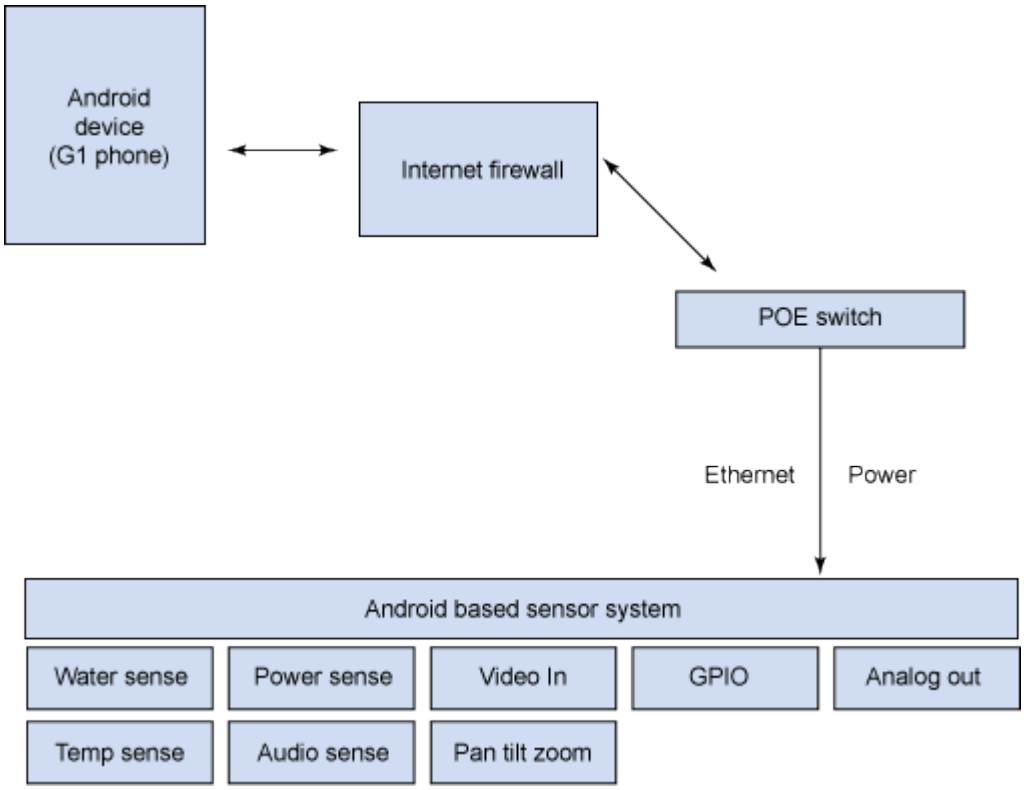
- 出现漏水
- 当前温度
- 消耗的功率
- 可能包含一些通用的类似数据和数字值

为什么需要关注消耗的功率？一个可能的原因就是有些人忘记关闭机器，因此电费单上的数字会一直增长。第二个理由有些复杂：假设您有一台非常大的冰箱，并且电源可能已被关闭。那么情况就复杂了，而且处理起来也需要很高的代价。或者，空调设备的断路器出现故障，因此机房无法保持恒定的温度。
基本的设计看上去是可行的。如果使用的是 Android，那么可以使用任何移动平台来替换 图 2 中的 Android。但如果使用配备了 Android 的设备替换微控制器，那应该怎么做呢？下一节将讨论对这个应用程序的扩展以及通过使用 Android 而启用的特性。

扩展应用程序

本文的第一个架构以一个微控制器为中心。微控制器可分为不同的外形和大小，从 Microchip 的 6 pin“10F” 到添加了外围设备、pin 和代码空间的 32 位大型微控制器。如果使用 Android 取代传统的微控制器放到设备中，会怎么样？对于某些应用程序而言，在成本方面是不可取的，但是根据图 3 的判断，这种方法也是可行的。

图 3. 在设备中使用 Android 的可能架构



使用嵌入式的方式部署 Android 为您提供了更加丰富的编程环境。您可以和以前一样继续监视湿度、温度和功率消耗特征，同时还可以观察到记录音频、视频和振动。您将拥有一个微报警、访问控制系统，以及一个环节监控工具。由于 Android 已经可以实现联网，您不需要使用控制器 PC 就可以实现监控并与网络直接对话。

这种方法还为现场更新软件提供了额外的好处。假设您希望为监控软件添加新的特性（或修复 bug）。如果使用传统的微控制器方法，那么任务执行起来将十分繁琐并且代价昂贵，甚至根本不可能实现。而对于 Android 而言，您可以获得更整洁的部署模型并拥有更好的灵活性。

Android 如今主要运行在移动手机中，但是它已经被移植到 NetBooks 和其他平台上。希望本文为您提供了一些好的思考内容。我现在该去运行我的系统了。您永远也不会知道下一次热水器漏水会在什么时候发生。

结束语

在本文中，我们大体介绍了 Android 的联网功能。您了解了一些自己可以创建的样例应用程序，包括与 Web 服务器交互和发送文本消息。您看到了如何将 Android 连接到一个真实的环境监控系统。通过代码示例，您了解到应该在什么时候将 Android 扩展到一些特殊应用程序中，比如嵌入式控制器。

请继续关注我的下一篇文章，它将介绍如何使用基于 Android 的电话构建一个婴儿监控系统。

深入探讨 Android 传感器

developerWorks.

随处监控您的环境

Android 是一个面向应用程序开发的富平台，它拥有许多具有吸引力的用户界面元素和数据管理功能。Android 还提供了一组丰富的接口选项。在本文中，学习如何配合使用 Android 的各种传感器选项监控您的环境。样例代码展示了如何在 Android 电话中录制音频。想构建自己的婴儿监视器吗？想用声音来接听电话或者打开房门吗？请学习如何利用配备有 Android 的设备的硬件功能。

简介

对于 Java™ 开发人员来说，Android 平台是通过使用硬件传感器创建创新应用程序的理想平台。我们将学习一些可用于 Android 应用程序的接口连接选项，包括使用传感器子系统和录制音频片段。

利用配备 Android 的设备的硬件功能可以构建哪些应用程序呢？任何需要电子监视和监听的应用程序都可以构建。婴儿监视器、安全系统，甚至地震仪都可以。理论上讲，您不能同时出现在两个地方，但 Android 可以利用一些可行的方法实现这一点。纵观本文始末，您必须记住，使用的 Android 设备不仅仅局限于“手机”，还可以是部署在固定位置、具有无线网络连接的设备，比如 EDGE 或 WiFi。下载 本文示例的源文件。

Android 传感器功能

使用 Android 平台有一个很新颖的地方，那就是您可以在设备内部访问一些“好工具”。过去，访问设备底层硬件的能力一度让移动开发人员感到非常棘手。尽管 Android Java 环境的角色仍然是您和设备的桥梁，但 Android 开发团队让许多硬件功能浮出了水面。该平台是一个开源平台，因此您可以自由地编写代码实现您的任务。

如果尚未安装 Android，您可以 下载 Android SDK。您还可以 浏览 android.hardware 包的内容并参考本文的示例。android.media 包 包含了一些提供有用和新颖功能的类。

Android SDK 中包含的一些面向硬件的功能描述如下。

表 1. Android SDK 中提供的面向硬件的特性

特性	描述
`android.hardware.Camera`	允许应用程序与相机交互的类，可以截取照片、获取预览屏幕的图像，修改用来治理相机操作的参数。
`android.hardware.SensorManager`	允许访问 Android 平台传感器的类。并非所有配备 Android 的设备都支持 `SensorManager` 中的所有传感器，虽然这种可能性让人非常兴奋。（可用传感器的简介见下文）
`android.hardware.SensorListener`	在传感器值实时更改时，希望接收更新的类要实现的接口。应用程序实现该接口来监视硬件中一个或多个可用传感器。例如，本文中的 代码 包含实现该接口的类，实现后可以监视设备的方向和内置的加速表。
`android.media.MediaRecorder`	用于录制媒体样例的类，对于录制特定位置（比如婴儿保育）的音频活动非常有用。还可以分析音频片段以便在访问控件或安全应用程序时进行身份鉴定。例如，它可以帮助您通过声音打开门，以节省时间，不需要从房产经纪人处获取钥匙。
`android.FaceDetector`	允许对人脸（以位图形式包含）进行基本识别的类。不可能有两张完全一样的脸。可以使用该类作为设备锁定方法，无需记密码 — 这是手机的生物特征识别功能。
android.os.*	包含几个有用类的包，可以与操作环境交互，包括电源管理、文件查看器、处理器和消息

	类。和许多可移动设备一样，支持 Android 的电话可能会消耗大量电能。让设备在正确的时间“醒来”以监视感兴趣的事件是在设计时需要首先关注的方面。
java.util.Date java.util.Timer java.util.TimerTask	当测量实际的事件时，数据和时间往往很重要。例如，java.util.Date 类允许您在遇到特定的事件或状况时获取时间戳。您可以使用 java.util.Timer 和 java.util.TimerTask 分别执行周期性任务或时间点任务。

android.hardware.SensorManager 包含几个常量，这表示 Android 传感器系统的不同方面，包括：

传感器类型

方向、加速表、光线、磁场、临近性、温度等。

采样率

最快、游戏、普通、用户界面。当应用程序请求特定的采样率时，其实只是对传感器子系统的一个提示，或者一个建议。不保证特定的采样率可用。

准确性

高、低、中、不可靠。

SensorListener 接口是传感器应用程序的中心。它包括两个必需方法：

- onSensorChanged(int sensor,float values[]) 方法在传感器值更改时调用。该方法只对受此应用程序监视的传感器调用（更多内容见下文）。该方法的参数包括：一个整数，指示更改的传感器；一个浮点值数组，表示传感器数据本身。有些传感器只提供一个数据值，另一些则提供三个浮点值。方向和加速表传感器都提供三个数据值。
- 当传感器的准确性更改时，将调用 onAccuracyChanged(int sensor,int accuracy) 方法。参数包括两个整数：一个表示传感器，另一个表示该传感器新的准确值。

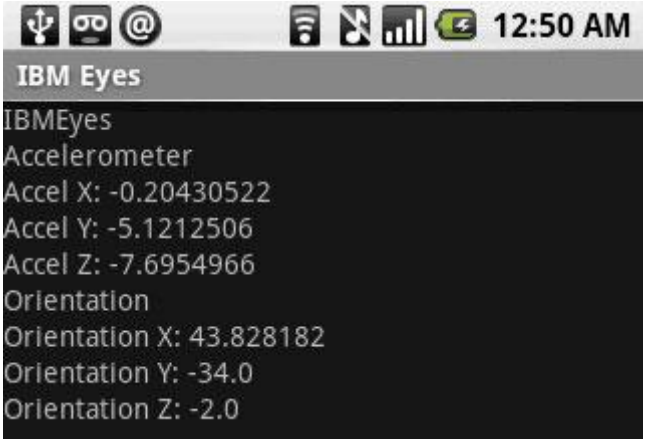
要与传感器交互，应用程序必须注册以侦听与一个或多个传感器相关的活动。注册使用 SensorManager 类的 registerListener 方法完成。本文中的 代码示例 演示了如何注册和注销 SensorListener。

记住，并非所有支持 Android 的设备都支持 SDK 中定义的所有传感器。如果某个传感器无法在特定的设备上使用，您的应用程序就会适当地降级。

传感器示例

样例应用程序仅监控对方向和加速表传感器的更改（源代码见 下载）。当收到更改时，传感器值在 TextView 小部件的屏幕上显示。图 1 展示了该应用程序的运行情况。

图 1. 监视加速和方向



使用 Eclipse 环境和 Android Developer Tools 插件创建的应用程序。（关于使用 Eclipse 开发 Android 应用程序的信息，请参见 参考资料。）清单 1 展示了该应用程序的代码。

清单 1. IBMEyes.java

```
package com.msi.ibm.eyes;
import android.app.Activity;
import android.os.Bundle;
import android.util.Log;
import android.widget.TextView;
import android.hardware.SensorManager;
import android.hardware.SensorListener;
public class IBMEyes extends Activity implements SensorListener {
    final String tag = "IBMEyes";
    SensorManager sm = null;
    TextView xViewA = null;
    TextView yViewA = null;
    TextView zViewA = null;
    TextView xViewO = null;
    TextView yViewO = null;
    TextView zViewO = null;

    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
       // get reference to SensorManager
        sm = (SensorManager) getSystemService(SENSOR_SERVICE);
        setContentView(R.layout.main);
        xViewA = (TextView) findViewById(R.id.xbox);
        yViewA = (TextView) findViewById(R.id.ybox);
        zViewA = (TextView) findViewById(R.id.zbox);
        xViewO = (TextView) findViewById(R.id.xboxo);
        yViewO = (TextView) findViewById(R.id.yboxo);
        zViewO = (TextView) findViewById(R.id.zboxo);
    }
    public void onSensorChanged(int sensor, float[] values) {
        synchronized (this) {
            Log.d(tag, "onSensorChanged: " + sensor + ", x: " +
values[0] + ", y: " + values[1] + ", z: " + values[2]);
            if (sensor == SensorManager.SENSOR_ORIENTATION) {
                xViewO.setText("Orientation X: " + values[0]);
                yViewO.setText("Orientation Y: " + values[1]);
                zViewO.setText("Orientation Z: " + values[2]);
            }
            if (sensor == SensorManager.SENSOR_ACCELEROMETER) {
                xViewA.setText("Accel X: " + values[0]);
```

```
            yViewA.setText("Accel Y: " + values[1]);
            zViewA.setText("Accel Z: " + values[2]);
         }
      }
   }

   public void onAccuracyChanged(int sensor, int accuracy) {
        Log.d(tag,"onAccuracyChanged: " + sensor + ", accuracy: " + accuracy);
   }
   @Override
   protected void onResume() {
      super.onResume();
     // register this class as a listener for the orientation and accelerometer sensors
      sm.registerListener(this,
            SensorManager.SENSOR_ORIENTATION |SensorManager.SENSOR_ACCELEROMETER,
            SensorManager.SENSOR_DELAY_NORMAL);
   }

   @Override
   protected void onStop() {
      // unregister listener
      sm.unregisterListener(this);
      super.onStop();
   }
}
```

编写应用程序必须基于常见的活动，因为它只是利用从传感器获取的数据更新屏幕。在设备可能在前台执行其他活动的应用程序中，将应用程序构建为服务可能更加合适。

该活动的 onCreate 方法可以引用 SensorManager，其中包含所有与传感器有关的函数。onCreate 方法还建立了对 6 个 TextView 小部件的引用，您需要使用传感器数据值更新这些小部件。

onResume() 方法使用对 SensorManager 的引用通过 registerListener 方法注册传感器更新：

- 第一个参数是实现 SensorListener 接口的类的实例。
- 第二个参数是所需传感器的位掩码。在本例中，应用程序从 SENSOR_ORIENTATION 和 SENSOR_ACCELEROMETER 请求数据。
- 第三个参数是一个系统提示，指出应用程序更新传感器值所需的速度。

应用程序（活动）暂停后，需要注销侦听器，这样以后就不会再收到传感器更新。这通过 SensorManager 的 unregisterListener 方法实现。惟一的参数是 SensorListener 的实例。

在 registerListener 和 unregisterListener 方法调用中，应用程序使用关键字 this。注意类定义中的 implements 关键字，其中声明了该类实现 SensorListener 接口。这就是要将它传递到 registerListener 和 unregisterListener 的原因。

SensorListener 必须实现两个方法 onSensorChange 和 onAccuracyChanged。示例应用程序不关心传感器的准确度，但关注传感器当前的 X、Y 和 Z 值。onAccuracyChanged 方法实质上不执行任何操作；它只在每次调用时添加一个日志项。

似乎经常需要调用 onSensorChanged 方法，因为加速表和方向传感器正在快速发送数据。查看第一个参数确定哪个传感器在发送数据。确认了发送数据的传感器之后，将使用方法第二个参数传递的浮点值数组中所包含的数据更新相应的 UI 元素。该示例只是显示这些值，但在更加高级的应用程序中，还可以分析这些值，比较原来的值，或者设置某种模式识别算法来确定用户（或外部环境）的行为。

现在您已经了解了传感器子系统，接下来的部分将回顾一个在 Android 手机上录制音频的代码样例。该样例运行在 DEV1 开发设

备上。

使用 MediaRecorder

android.media 包包含与媒体子系统交互的类。使用 android.media.MediaRecorder 类进行媒体采样，包括音频和视频。MediaRecorder 作为状态机运行。您需要设置不同的参数，比如源设备和格式。设置后，可执行任何时间长度的录制，直到用户停止。

清单 2 包含的代码在 Android 设备上录制音频。显示的代码不包括应用程序的 UI 元素（完整源代码见 下载）。

清单 2. 录制音频片段

```
MediaRecorder mrec ;
File audiofile = null;
private static final String TAG="SoundRecordingDemo";
protected void startRecording() throws IOException
{
   mrec.setAudioSource(MediaRecorder.AudioSource.MIC);
   mrec.setOutputFormat(MediaRecorder.OutputFormat.THREE_GPP);
   mrec.setAudioEncoder(MediaRecorder.AudioEncoder.AMR_NB);
   if (mSampleFile == null)
   {
       File sampleDir = Environment.getExternalStorageDirectory();
       try
       {
          audiofile = File.createTempFile("ibm", ".3gp", sampleDir);
       }
       catch (IOException e)
       {
           Log.e(TAG,"sdcard access error");
           return;
       }
   }
   mrec.setOutputFile(audiofile.getAbsolutePath());
   mrec.prepare();
   mrec.start();
}
protected void stopRecording()
{
   mrec.stop();
   mrec.release();
   processaudiofile(audiofile.getAbsolutePath());
}
protected void processaudiofile()
{
```

```
    ContentValues values = new ContentValues(3);
    long current = System.currentTimeMillis();
    values.put(MediaStore.Audio.Media.TITLE, "audio" + audiofile.getName());
    values.put(MediaStore.Audio.Media.DATE_ADDED, (int) (current / 1000));
    values.put(MediaStore.Audio.Media.MIME_TYPE, "audio/3gpp");
    values.put(MediaStore.Audio.Media.DATA, audiofile.getAbsolutePath());
    ContentResolver contentResolver = getContentResolver();

    Uri base = MediaStore.Audio.Media.EXTERNAL_CONTENT_URI;
    Uri newUri = contentResolver.insert(base, values);

    sendBroadcast(new Intent(Intent.ACTION_MEDIA_SCANNER_SCAN_FILE, newUri));
}
```

在 startRecording 方法中，实例化并初始化 MediaRecorder 的实例：

- 输入源被设置为麦克风（MIC）。
- 输出格式被设置为 3GPP（*.3gp 文件），这是移动设备专用的媒体格式。
- 编码器被设置为 AMR_NB，这是音频格式，采样率为 8 KHz。NB 表示窄频。SDK 文档 解释了不同的数据格式和可用的编码器。

音频文件存储在存储卡而不是内存中。External.getExternalStorageDirectory() 返回存储卡位置的名称，在该目录中将创建一个临时文件名。然后，通过调用 setOutputFile 方法将文件关联到 MediaRecorder 实例。音频数据将存储到该文件中。
调用 prepare 方法完成 MediaRecorder 的初始化。准备开始录制流程时，将调用 start 方法。在调用 stop 方法之前，将对存储卡上的文件进行录制。release 方法将释放分配给 MediaRecorder 实例的资源。
音频采样完成之后，需要采取以下步骤：

- 向设备的媒体库添加该音频。
- 执行一些模式识别步骤确定声音：
    - 这是婴儿的啼哭声吗？
    - 这是所有人的声音吗？是否要解锁手机？
    - 这是“芝麻开门”吗？是否要打开通往“秘密通道”的大门？
- 自动将音频文件上传到网络位置以便处理。

在该代码样例中，processaudiofile 方法将音频添加到媒体库。使用 Intent 通知设备上的媒体应用程序有新内容可用。
关于该代码片段最后要注意的是：如果您试用，它一开始不会录制音频。您将看到创建的文件，但是没有任何音频。您需要向 AndroidManifest.xml 文件添加权限：

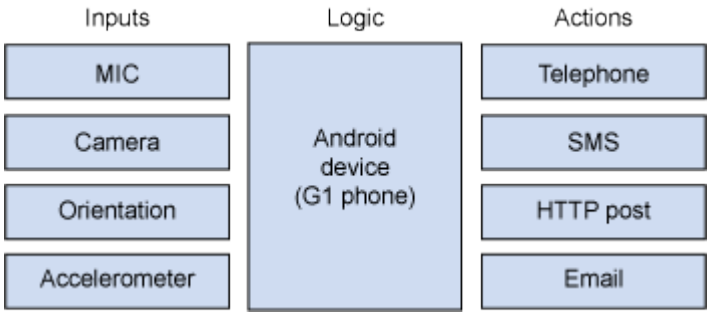
```
<uses-permission android:name="android.permission.RECORD_AUDIO"></uses-permission>
```

现在，您已经学了一点关于与 Android 传感器和录制音频相关的内容。下一节将更全面的介绍与数据采集和报告系统有关的应用程序架构。

Android 作为传感器平台

Android 平台包含各种用于监视环境的传感器选项。有了输入或模拟选项数组，以及高级计算和互联功能，Android 成为构建实际系统的最佳平台。图 2 显示了输入、应用程序逻辑、通知方法或输出之间的简单视图。

图 2. 以 Android 为中心的传感器系统的方块图



该架构很灵活；应用程序逻辑可以划分为本地 Android 设备和服务器端资源（可以实现更大的数据库和计算功能）。例如，本地 Android 设备上录制的音轨可以 `POST` 到 Web 服务器，其中将根据音频模式数据库比较数据。很明显，这仅仅是冰山一角。希望您能更深入地研究，让 Android 平台超越移动电话的范畴。

结束语

在本文中，我们介绍了 Android 传感器。样例应用程序度量了方向和加速，以及使用 `MediaRecorder` 类与录制功能进行交互。对于构建实际系统，Android 是一个灵活、有吸引力的平台。Android 领域发展迅速，并且不断壮大。请务必关注该平台。