Join the discussion @ p2p.wrox.com

Wrox Programmer to Programmer™

移动开发经典丛书

Professional Android Sensor Programming

Android 传感器高级编程

[美] Greg Milette 著
Adam Stroud
裴佳迪 译

清华大学出版社

移动开发经典丛书

Android 传感器高级编程

[美] Greg Milette
Adam Stroud 著

裴佳迪 译

清华大学出版社

北 京

Greg Milette, Adam Stroud
Professional Android Sensor Programming
EISBN：978-1-118-18348-9

北京市版权局著作权合同登记号 图字：01-2012-8288

图书在版编目(CIP)数据

Android 传感器高级编程/(美) 米内特(Milette,G.)，(美) 斯川德(Stroud,A.) 著；裴佳迪 译.
—北京：清华大学出版社，2013
(移动开发经典丛书)
书名原文：Professional Android Sensor Programming
ISBN 978-7-302-34077-5

Ⅰ. ①A…　Ⅱ. ①米…　②斯…　③裴…　Ⅲ. ①移动终端—应用程序—程序设计　Ⅳ. ①TN929.53

中国版本图书馆 CIP 数据核字(2013)第 240338 号

责任编辑：王　军　于　平
装帧设计：牛静敏
责任校对：邱晓玉
责任印制：

出版发行：清华大学出版社
网　　址：http://www.tup.com.cn，http://www.wqbook.com
地　　址：北京清华大学学研大厦 A 座　　邮　　编：100084
社 总 机：010-62770175　　邮　　购：010-62786544
投稿与读者服务：010-62776969，c-service@tup.tsinghua.edu.cn
质 量 反 馈：010-62772015，zhiliang@tup.tsinghua.edu.cn
印 刷 者：
装 订 者：
经　　销：全国新华书店
开　　本：185mm×260mm　　印　　张：31.25　　字　　数：761 千字
版　　次：2013 年 11 月第 1 版　　印　　次：2013 年 11 月第 1 次印刷
印　　数：1~3500
定　　价：79.80 元

产品编号：

译 者 序

翻看手机中的应用，就能发现大多数应用都已经使用了传感器。让微信彻底火起来的“附近的人”和“摇一摇”、碰一碰交换信息的 Bump、各种运动记录 app、神奇的“磁力探测仪”、火爆的游戏 Temple Run……手机设备中的传感器让应用的可玩性大大增加，提供了各种创新的交互方式，以及在 PC 上很难实现的功能。在我看来，移动设备相对 PC 来说的主要优势在于随时随地的便携性，还有就是丰富的传感器。用好传感器，已经是手机应用开发的必修课。

可是开始听到本书书名时，对翻译本书并没有什么兴趣。我自己也写过一本 Android 开发方面的书，觉得介绍传感器编程的话，一个章节足以。可是在看了英文书后，发现本书并不是简单地介绍 Android 传感器 API 的调用，而是非常专业细致地介绍了 Android 传感器。这本书将教会你真正用好 Android 传感器。

首先，本书对传感器的介绍都非常深入。我一直认为，要真正用好一个东西，必须了解其原理。本书对传感器的原理都作了必要的解读，就拿第 1 章来说，相信大家都知道 GPS 的原理，但是你知道 A-GPS 吗？S-GPS 呢？是否知道 Wi-Fi 定位和移动网络定位的原理呢？想要用好 Android 的定位服务，这些知识是必需的。我的一位朋友前几天还问我，他玩微信中“附近的人”，明明在杭州却老是定位在南京。其实原因就是他在使用 Wi-Fi 定位，而这个路由器在南京呆了 6 年，在 Google 的数据库中早就被定位在了南京的某地，Google 并没有即时更新该数据库。如果微信考虑这种情况，相信可以对这种状况做一些特别的处理。

其次，本书的范围很广。与其说是介绍传感器，倒不如说是介绍如何充分利用 Android 设备的硬件能力。本书分 4 个部分，涵盖了定位的相关内容、各种物理传感器的介绍(包括重力传感器、加速计、磁强计等，还介绍了 AOA 和 ADK)，同时还包括了 NFC、摄像头、图像处理、麦克风等，以及语音识别和文本转语音的内容。可以说，本书完全涵盖了 Android 的硬件设备能力，能够用来打造一个非常有趣的应用。运用好这些硬件，可以充分发挥 Android 设备智能和便携的特性。同时这也是相对 Web 应用来说非常大的优势。书中甚至包括了最近火热的 Arduino 的使用介绍，读者可以尽情地创新创意。

再次，本书的示例非常丰富，并且很多都可以简单复用。书中有着大量的示例，大多都是作者对传感器使用的最佳实践，并且结合了实际的使用场景，而不是简单的 demo。同时，作者提供了大量框架性的代码，读者都可以直接复用。通过这些示例，读者可以快速掌握这些传感器的用法。

在翻译本书的过程中，感谢我的家人，你们给了我最大的支持。感谢我的好朋友徐婷翻译了第 6 章，且对本书进行了一丝不苟的校对，让行文更加通顺优美，在很多专业知识上也提供了宝贵的建议，在整个翻译过程中给了我莫大的支持和鼓励。感谢我所有的朋友和同事，本书的完成少不了你们的鼓励和支持。感谢清华大学出版社的编辑给了足够的耐心，并给了我很多的建议和帮助，让我获益匪浅。

本书的翻译过程非常艰辛，可谓一波三折。在本书付梓之际，虽然高兴，却也多了几分惶恐。这也是我的第一本译作，自觉水平有限，而本书其实又由多个作者完成，语言风格迥异，虽然我已经尽了很大的努力，但是肯定难免会有一些错漏和偏颇之处。对于本书的任何想法和意见都欢迎发送邮件至 notice520@gmail.com。

最后，希望本书能帮助各位开发人员打造更美、更有创意、更有趣的移动应用。

裴佳迪

作 者 简 介

Greg Milette 是一位职业 Android 开发人员，同时也是 Gradison Technologies 公司(一家应用开发公司)的创始人。他喜欢创建实用的应用，例如 Digital Recipe Sidekick，也乐于在 StackOverflow 做出贡献。

Adam Stroud 为 RunKeeper Android 版本的领导开发人员。他自称为 phandroid，是 StackOverflow Android 虚拟社区以及 Android Google group 上的积极参与者。

贡献者简介

David N. Hutchison(http://davidnhutch.com)出生并成长于新西兰，目前为康奈尔大学的物理学博士研究生，他正在学校开发下一代惯性传感器。他喜欢在机加工车间设计带微控制器的小工具，他也骑摩托车、开公司。David 撰写了本书的第 5 章和第 6 章，并对第 7 章和第 10 章做出了一些贡献。

Jon Webb 是 Jon’s Java Imaging Library 的开发人员，拥有超过 30 年的职业软件开发经验。他喜欢在 Android 上的图像处理编程，因为这把他带回到了早年的美好回忆。Jon 撰写了本书的第 12 章和第 13 章。

Pearl Chen 从事跨学科的工作，从 HTML 到 LED，Android 到 Arduino。作为教育者和开发人员，Pearl 教编程和电子课程，同时还担任 http://thehungryveg.com 的 CTO。访问 http://klab.ca/arduino，可以获取更多关于 Pearl 即将举行的研讨会以及 Arduino 套件的信息。Pearl 撰写了本书的第 11 章，并对第 10 章有所贡献。

致　　谢

我们想感谢 David Hutchinson、Pearl Chen 和 Jon Webb 在本书的写作过程中提供的内容和指导。他们在物理、AOA、NFC 以及图像处理方面的专业知识让我们能够以本不可能达到的详细程度介绍了一些令人兴奋的 Android 用法。

我们想感谢我们的编辑，激励我们去撰写这本书，感谢他们为本书润色所作出的辛勤工作，使之更人性化，而不是像绿色机器人写的那样死板。

最后，如果没有 Android 开发人员社区里众人的帮助，我们不可能完成本书。他们分享知识，并帮助我们朝着一个共同的目标奋进。我们希望本书以及书中的代码有助于报答他们的青睐。

前　言

Android 是活的，它们可以定位自己，看、听以及理解对话。它们可以感知无线电信号并探测方向、移动以及环境属性。你的计算机能实现这一切吗？

传感器是让 Android 设备区别于其他计算机的其中一个功能。如果没有传感器，Android 设备只是一个动力不足、屏幕太小的 Web 浏览器，同时其输入机制也很笨拙。

传感器还能让应用做一些很棒的事情。例如，传感器可以帮助用户摆脱缓慢的手动输入和操作，传感器还可以帮助用户做一些之前不可能做到的事。正因为如此，应用想要成功，结合传感器可能是必需的。

传感器将继续成为 Android 平台的重要组成部分。随着 Android 设备硬件规格的提高，传感器的数量和质量也提高了。与此同时，用户会继续期待应用使用一些令人兴奋的新的传感器。因此，使用 Android 传感器是 Android 程序员必须掌握的一个至关重要的技能。本书将会为你提升该技能以及使用传感器开发出色的应用提供所需的知识和代码。

Android 传感器编程

编写使用 Android 传感器的应用涉及多方面的内容，包括理解 Android 设备的感知能力、选择合适的传感器，以及实现一个可以获取并解读传感器数据的应用。

Android 的感知能力

一台 Android 设备可以有多种传感器。本书使用的传感器定义结合了很多 Android 设备能力。在本书中，传感器指的是：

一种可以捕获对设备和其外部环境的测量的能力。

感知能力来源于 Android 设备的硬件能力以及对其有创造性的运用。这些能力可以直接使用来自可测量物理量的硬件的值，例如磁场传感器。也可以使用用户通常与之交互的硬件，例如摄像头和麦克风。某些能力甚至能使用一组硬件组合以及基于服务器的处理，例如语音识别。无论来源是什么，结果数据都可以通知应用设备状态和其目前的环境。

本书介绍了在以下类型的传感器中如何编写应用来处理数据。

- 位置传感器：使用包括 GPS 在内的多种传感器确定设备的位置。
- 物理传感器：检测设备特定的属性，例如方向、加速度、旋转，以及光、磁场、气压等环境属性。

- NFC 扫描器：检测近场通信(NFC)标签，并和其他带 NFC 功能的 Android 设备共享数据。
- 摄像头；收集可视图像。
- 麦克风：录制音频。
- 语音识别：组合使用从麦克风录制的音频以及识别算法将音频转换为文本。
- 外部传感器：使用 Android 开放外设(Android Open Accessory, AOA)机制链接的传感器。

选择感知任务

理解传感器的工作原理有助于你了解应用的哪个任务可以受益于传感器相关的输入。它还有助于你解释在不同情况下传感器的表现，并了解传感器的局限性。例如

- 位置：了解多种位置传感器的工作原理，如第 1 章所述，你就会预期设备在室内的精度会比较低。
- 物理传感器：了解物理传感器的测量信息，如第 5 章所述，有助于你理解应用对传感器输出会做出的合理推测。

使用 API 样板

在任何应用中，请求传感器数据都需要类似的代码。每种数据都需要不同的样板。在很多种情况下，初始化 API 以及获取数据都不是很方便。本书提供的代码示例和库有助于让其更容易实现。API 使用所涉及的一些难点示例包括如下。

- 摄像头：在应用可以分析图像之前，它需要从摄像头获取图像。然而，使用设备摄像头需要处理设备旋转、硬件限制，并且正确地使用 Camera 和 View 对象。第 12 章和第 13 章介绍的抽象类用于处理这些细节。
- NFC：使用 NFC 涉及了解各个所需的步骤来读取和写入 NFC 标签，以及存入的数据内容。第 11 章阐释了完整的代码示例，该代码非常易于适配。

收集传感器数据

一旦应用可以初始化并获取传感器数据，它就需要利用 API 在应用运行过程中收集数据。数据的收集方式可以有很多种，取决于应用的使用方式。本书介绍了在不同任务下不同的数据收集方法。其中一些示例包括如下。

- 位置：位置跟踪是位置传感器的常见用途。一些应用需要在应用执行其他任务时也能持久地跟踪位置。第 3 章介绍了几种方法来可靠地实现位置跟踪。
- 语音识别：为了获取语音识别结果，应用除了运行语音识别器之外，还需要其他的组件。应用还需要允许用户激活语音，并协调用户说话的时机以及应用侦听的时机。第 IV 部分介绍了实现完整的语音命令所需的所有软件组件。

解读传感器数据

在应用收集了一些传感器数据之后，它需要分析数据来获取想要的效果。每个传感器都需要不同的数据分析算法。其中一些示例包括如下。

- 物理传感器：解读物理传感器的数据涉及从原始数据到可用数据的转换计算，以及帮助检测变化和忽略噪声的算法。第Ⅱ部分介绍了原理。
- 摄像头：处理摄像头的图像涉及建立一个图像处理管道。应用必须减小摄像头采集的大图像的尺寸到一个可管理的大小，否则会导致无法存入内存，或者处理速度过慢。而后应用需要以多种方式变换收集到的图像，从而检测图像中的某些东西。
- 麦克风：分析音频数据涉及信号处理算法。第 14 章介绍了音量检测和频率估算算法。
- 语音识别：执行语音命令涉及使用文本搜索方法来匹配命令单词和用户所说的话。第 17 章介绍了提高匹配成功率的一些方法。

本书中的应用

本书中呈现的应用根据特定的目的来使用传感器。这些应用提供了常见问题的实用代码组件。

- 第 3 章：使用一个数据库和 BroadcastReceiver 来实现持久、可靠的位置跟踪。
- 第 4 章：使用服务来实现高效的接近警报，节省电池寿命。
- 第 7 章：使用多种物理传感器来确定设备是面朝下还是面朝上。
- 第 7 章：使用旋转矢量传感器来实现增强现实应用所需的功能。
- 第 8 章：使用加速度传感器来检测移动。
- 第 9 章：使用气压计来检测高度。
- 第 10 章：使用 AOA 从外部温度传感器收集数据。
- 第 11 章：使用含有自定义数据的 NFC 标签来跟踪库存。
- 第 13 章：使用摄像头来检测 Android logo。
- 第 14 章：使用麦克风通过检测大的响声和乐音来实现声控开关。
- 第 17 章和第 18 章：使用语音识别和文本转语音来实现查询和操作食物数据库中数据的语音命令。

Android 高级编程

本书适用于熟悉 Android 编程的开发人员。本书假设你已经理解了基础的 Android 概念，例如 Activity 和 Intent，但是还没有使用过传感器相关的 API。本书还假设你了解一些数学概念并充分阐释了所需了解的物理概念。

此外，本书专注于传感器编程。这种专注使得有足够的空间来充分介绍如何处理每种数据，而不只是简单地解释 API 的使用。

在传感器编程之外，本书介绍的技术适用于任何应用。例如，书中的章节展示了如何在各种任务中使用 BroadcastReceiver、Service、AsyncTask 以及数据库。

开始感知

应用可以利用传感器实现了不起的功能，这些功能都很独特并能节约用户的时间。Android 的感知能力将会提高并且继续成为很多应用中重要的一部分。本书用你所需的知识和代码来武装你，帮助你开发出更强大的应用。

Android Sensing Playground 应用

本书附带的应用称为 Android Sensing Playground。该应用让你可以执行书中的大多数应用和示例代码，并利用 playground 来观察相关的 API 在不同参数设置下的工作状况。

可以从 Google Play 下载该应用：https://play.google.com/store/apps/details?id=root.gast.playground。

Great Android Sensing Toolkit(GAST)

本书中的代码为开源项目 Great Android Sensing Toolkit(GAST)的一部分。最新的更新和代码都可以从 Github 获取，链接为：https://github.com/gast-lib。

如何下载本书的示例代码

在读者学习本书中的示例时，可以手工输入所有的代码，也可以使用本书附带的源代码文件。本书使用的所有源代码都可以从本书合作站点 http://www.wrox.com/和 http://www.tupwk.com.cn/downpage 上下载。登录到站点 http://www.wrox.com/上，使用 Search 工具或书名列表就可以找到本书。接着单击本书细目页面上的 Download Code 链接，就可以获得所有的源代码。

注释：

许多图书的书名都很相似，所以通过 ISBN 查找本书是最简单的，本书的 ISBN 是 978-1-118-18348-9。

在下载了代码后，只需用自己喜欢的解压缩软件对它进行解压缩即可。另外，也可以进入 http://www.wrox.com/dynamic/books/download.aspx 上的 Wrox 代码下载主页，查看本书和其他 Wrox 图书的所有代码。

勘误表

尽管我们已经尽了各种努力来保证文章或代码中不出现错误，但是错误总是难免的，如果你在本书中找到了错误，例如拼写错误或代码错误，请告诉我们，我们将非常感激。通过勘误表，可以让其他读者避免受挫，当然，这还有助于提供更高质量的信息。

要在网站上找到本书的勘误表，可以登录 http://www.wrox.com，通过 Search 工具或书名列表查找本书，然后在本书的细目页面上，单击 Book Errata 链接。在这个页面上可以查看 Wrox 编辑已提交和粘贴的所有勘误项。完整的图书列表还包括每本书的勘误表，网址是 www.wrox.com/misc-pages/ booklist.shtml。

如果在 Book Errata 页面上没有看到你找出的错误，请进入 www.worx.com/contact/techsupport.shtml，填写表单，发电子邮件，我们就会检查你的信息，如果是正确的，就在本书的勘误表中粘贴一个消息，我们将在本书的后续版本中采用。

p2p.wrox.com

P2P 邮件列表是为作者和读者之间的讨论而建立的。读者可以在 p2p.wrox.com 上加入 P2P 论坛。该论坛是一个基于 Web 的系统，用于传送与 Wrox 图书相关的信息和相关技术，与其他读者和技术用户交流。该论坛提供了订阅功能，当论坛上有新帖子时，会给你发送你选择的主题。Wrox 作者、编辑和其他业界专家和读者都会在这个论坛上进行讨论。

在 http://p2p.wrox.com 上有许多不同的论坛，帮助读者阅读本书，在读者开发自己的应用程序时，也可以从这个论坛中获益。要加入这个论坛，必须执行下面的步骤：

(1) 进入 p2p.wrox.com，单击 Register 链接。

(2) 阅读其内容，单击 Agree 按钮。

(3) 提供加入论坛所需的信息及愿意提供的可选信息，单击 Submit 按钮。

(4) 然后就可以收到一封电子邮件，其中的信息描述了如何验证账户，完成加入过程。

提示：

不加入 P2P 也可以阅读论坛上的信息，但只有加入论坛后，才能发送自己的信息。

加入论坛后，就可以发送新信息，回应其他用户的帖子。可以随时在 Web 上阅读信息。如果希望某个论坛给自己发送新信息，可以在论坛列表中单击该论坛对应的 Subscribe to this Forum 图标。

对于如何使用 Wrox P2P 的更多信息，可阅读 P2P FAQ，了解论坛软件的工作原理，以及许多针对 P2P 和 Wrox 图书的常见问题解答。要阅读 FAQ，可以单击任意 P2P 页面上的 FAQ 链接。

目　录

第 I 部分

位 置 服 务

第 1 章

Android 位置服务简介

本章内容:

- 概述在 Android 中如何提供位置信息
- 简单介绍 GPS
- 讨论在 Android 中使用 A-GPS 的原因
- 概述网络位置提供者

在移动开发领域，位置信息正变得越来越重要。之前和定位无关的应用利用位置信息提供了更丰富的用户体验。将最新的位置信息和简单的 Web 搜索引擎结合，就能让 Android 设备提供以前不可能实现的一系列功能。能够轻松地为应用检索和提供位置数据已经成为如今移动平台的一大主要特色。Android 通过它的位置服务提供了这个功能。

Android 的位置服务提供访问设备定位设施的接口。位置信息可以广泛用于多种功能，并且可以使设备和运行其上的软件对周边有更好的了解。

1.1 用于确定位置的方法

Android 使用不同的方法为应用提供位置信息。在 Android 中，这些设施称作位置提供者(location provider)，每一个设施都有自己特有的优点和缺点。另外，因为每个位置提供者的独特性，要在不同的情况下以不同的方式使用它们。

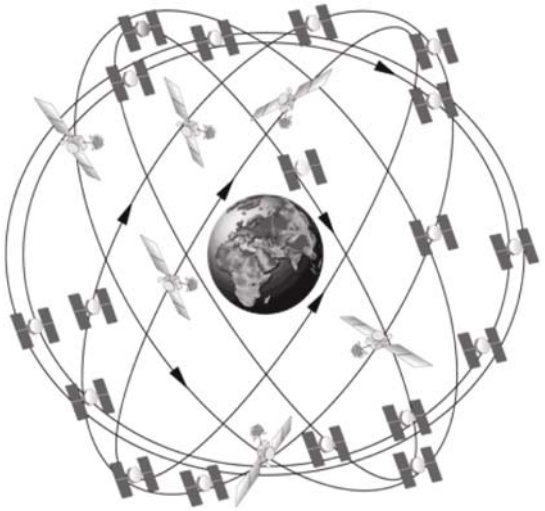
下面的小节会对获取位置的方法的工作原理给出更高层次的解释。虽然应用几乎无法控制提供者的工作过程，但是它可以决定使用哪个位置提供者。了解每个提供者如何工作，对于理解它的局限性和特点大有帮助。

1.1.1　GPS 提供者

Global Positioning System(全球定位系统，GPS)使用一套环绕地球的卫星系统来帮助接收器(在这里就是一部 Android 手机)确定它的位置。术语 GPS 指的是整个 GPS 系统，包括了卫星、接收器以及用来监控和调整它的控制站。脱离这套系统，手机中的接收器就是毫无用处的。

1. GPS 的工作原理

一般情况下，GPS 接收器使用环绕地球的 GPS 卫星提供的信息来计算当前所处的位置。GPS 系统包括 27 个围绕地球运行的卫星，它们会将信息传输到潜在的接收器。每一个卫星都遵循指定的轨道运动，保证至少在地球的任何位置、任何时间都有 4 个卫星是“可见”的。使用 GPS 来定位，必须确保“视线范围”内至少有 4 个卫星。图 1-1 展示了 GPS 卫星星座。



来源: http://gps.gov/multimedia/images

图 1-1　GPS 卫星星座

星座中的每个 GPS 卫星都会持续地发送自己当前的位置信息(ephemeris data，星历数据)和历书数据(almanac data)。历书数据包含了星座中每个卫星的数据，其中包括轨道数据以及系统整体状态信息。换一种方式说，星历数据是单一卫星的数据，而历书数据是所有卫星的数据。每个卫星会同时发送这两种信息。虽然星历数据和历书数据都会为给定的卫星提供位置数据，但是星历数据保证了位置计算的准确度。

为了计算位置，GPS 接收器必须能确定它和多个卫星之间的距离。通过星历数据可以做到这一点。从卫星传回的数据里包括位置数据以及传输开始的时间。每个 GPS 卫星都有一个非常精确的计时机制，使之和其他的卫星在时间上保持同步。为了精确地计算位置，GPS 卫星和 GPS 接收器的时钟必须高度同步。即使是最微小的时间差，也可能会导致位置

计算产生较大的误差。

通过传输开始时间，GPS 接收器可以计算传输过程所花费的时间(接收器能够获取传输结束时间)。这个计算是建立在无线电波以真空中的光速来传输数据这一假设之上的(并不总是如此)。通过开始时间、结束时间以及恒定的光速，GPS 接收器可以计算出自身和卫星之间的距离。

利用多个卫星至 GPS 接收器的距离，就可以使用三角定位确定当前的位置。从本质上讲，所有球体相交的点就是接收器的位置。最少需要三个卫星才可以确定一个二维的位置(纬度和经度)。如果和更多的卫星通信，GPS 接收器可以确定诸如海拔信息等更多的位置信息。GPS 接收器不会局限仅使用四个卫星。一般随着与之通信以传送数据的卫星越多，得到的位置精确度也越高(但是有一个上限)。

GPS 对于确定当前位置是很有用的，但是它也确实有一些缺点(特别是在移动平台上)，其中一点是它计算当前位置所耗费的时间。在计算位置之前，需要搜索到多个卫星。有很多卫星在环绕地球飞行，但是只有极少数是在任何时候都“可见”的，因为大多数卫星会在地平线以下，被地球遮挡(记住，卫星必须可见)。历书可以用来协助确定在给定时间和给定位置哪些卫星是可用的。但是，如果 GPS 并没有相关的当前历书，它就需要 GPS 卫星传输历书数据给它。这可能会是一个缓慢的过程。

2. GPS 的改进

虽然标准的 GPS 可以提供精确的位置数据，但是它的局限性使得移动设备很难使用它。因此，现代移动设备通过使用辅助 GPS(A-GPS)，也可能是同步 GPS(S-GPS)，以规避标准 GPS 的局限性。

A-GPS

A-GPS 通过移动网络将 GPS 历书和其他信息传输到移动设备。移动网络的使用使得历书数据的传输速度更快，这可能会加快定位速度。另外，因为历书包含了所有 GPS 的信息，设备可以得到可视范围内 GPS 卫星的大概位置。这也同样会加快定位速度。

可以通过查看 GPS 配置文件来深入了解 A-GPS 的数据来源。程序清单 1-1 是一个位于北美的 Android 设备里使用的 GPS 配置文件。

程序清单 1-1：位于/system/etc/gps.conf 中的一个 GPS 示例配置文件

```
NTP_SERVER=north-america.pool.ntp.org
XTRA_SERVER_1=http://xtra1.gpsonextra.net/xtra.bin
XTRA_SERVER_2=http://xtra2.gpsonextra.net/xtra.bin
XTRA_SERVER_3=http://xtra3.gpsonextra.net/xtra.bin
```

程序清单 1-1 中的 GPS 配置文件指定了 A-GPS 数据的下载地址(XTRA_SERVER-1、XTRA_SERVER_2 和 XTR_SERVER_3)以及用于协调时间的 Network Time Protocol(网络时间协议，NTP)服务器(NTP_SERVER)。NTP 可以用来强制协调时间。这是非常重要的，因

为 GPS 非常依赖 GPS 接收器与 GPS 卫星的时钟同步。尽管 NTP 的使用并不能保证同步到毫秒级别，但是确实有助于防止大的时间差。由于在计算时间时使用很多诸如光速这样的数值，即使一个小的时间差都会很大程度地影响位置计算的精确度。

大多数用户可以读取/system/etc/gps.conf，但是写入需要增加权限。一般情况下，用户不需要编辑这个文件。

S-GPS

使用标准 GPS 的设备可能用同样的硬件进行 GPS 卫星通信以及拨打电话。这意味着同一时间只能执行这类操作之一。S-GPS 解决了这个问题，通过增加额外的硬件，可以让 GPS 无线电和蜂窝网络无线电同时投入使用。同时有两个活动的无线电可以加速 GPS 数据传输，因为它可以在蜂窝网络无线电可用时获取数据。

3. 限制

虽然大多数时候 GPS 可以提供精确的位置数据，但是它也有一些难以解决的局限性。首先，GPS 接收器与 GPS 卫星通信需要一个通畅的路线。这意味着 GPS 接收器在室内无法使用，甚至在户外无法看到天空的地方(如茂密的树林)也是如此。此外，因为需要多个卫星来生成位置数据，很可能需要相当长的时间来定位。这就更加表明 GPS 需要低功耗的 GPS 无线电。鉴于这些原因，在有些时候需要其他的位置信息来源。

阻碍 GPS 信号的物体可能会导致信号在到达 GPS 接收器之前被反射。如前所述，信号到达 GPS 接收器所花费的时间用于计算 GPS 卫星和 GPS 接收器之间的距离。被物体反射的 GPS 信号的路径不同于正常 GPS 卫星到 GPS 接收器的路径，这就会导致距离计算错误。这类错误称作多路径(multipath)误差，可能会导致显示的位置从本来的地方跳到另一个地方。这样的情况通常出现在城区，GPS 信号常常因为高楼而反弹。

4. 控制 GPS

在大多数情况下，应用开发人员只要 GPS“可以工作”就可以了。通常情况下，开发人员没有理由会干预 A-GPS 的数据源，或者清除并重新初始化 A-GPS 的数据。

然而 Android 提供了一个 API 来控制某些 GPS 数据。LocationManager 类(第 2 章会详细介绍)包含一个 sendExtraCommand()方法，这个方法可以用来控制设备的 GPS 状态。LocationManager.sendExtraCommand()方法有三个参数：一个字符串用于指定位置提供者、一个附加的命令以及一个提供命令执行信息的 Bundle 对象。

在编写本书时，GPS 位置提供者只支持以下三个附加命令：

- delete_aiding_data
- force_time_injection
- force_extra_injection

delete_aiding_data 命令用于删除先前已下载的 A-GPS 数据。这是唯一使用 Bundle 参数的附加命令，Bundle 用于控制要删除的 A-GPS 数据。Bundle 可以包含布尔型的键值对

来指明要移除的数据。可用的键字符串如下所示：

- ephemeris
- almanac
- position
- time
- iono
- utc
- health
- svdir
- scsteer
- sadata
- rti
- celldb-info
- all

传递一个空的 Bundle 会删除所有的 A-GPS 数据。

force_time_injection 命令从配置的 NTP 服务器检索当前时间并进行更新，用来进行 GPS 计算。

force_extra_injection 命令从一个配置服务器中下载 A-GPS 数据，这些数据将被 GPS 位置提供者使用。

1.1.2　网络提供者(Network Provider)

在 Android 中，基于网络的位置可以使用不同的方法来确定设备的位置。在编写本书时，网络位置提供者可以使用手机信号发射塔或者基于无线网络信息来提供位置信息。

1. 使用无线网络接入点

基于无线网络接入点提供位置信息是 Android 通过网络提供者获取位置的方法之一。虽然它确实需要 Wi-Fi 无线电支持，但是 Wi-Fi 无线电往往比 GPS 硬件更省电。

工作原理

基于 Wi-Fi 的位置检测的工作原理，是通过能够检测到的 Wi-Fi 接入点以及这些接入点当前的信号强度对设备进行跟踪。然后设备查询 Google 位置服务(与 Android 的位置服务不同)，后者基于 Wi-Fi 信息提供位置数据。设备收集的 Wi-Fi 信息包括 Wi-Fi 接入点的 MAC(mandatory access control)地址以及这些接入点的信号强度。

为了基于可见的 Wi-Fi 接入点提供位置信息，Google 位置服务必须获得这些 Wi-Fi 接入点的信息以及位置。用户从 Location Settings(位置设置)界面开启 Google 位置服务后，这些信息将会被 Android 设备收集。图 1-2 是在开启 Google 位置服务作为位置数据来源时展示给用户的确认界面。

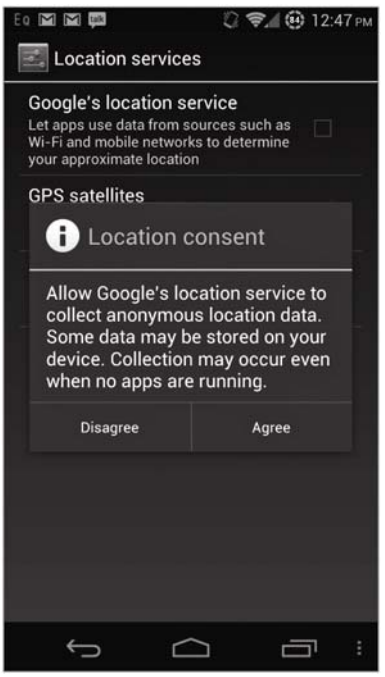


图 1-2　启用 Google 位置服务的确认界面

在此界面上按下 Agree 按钮则会允许设备记录 Wi-Fi 信息以及当前的位置信息(可能来自 GPS)，并上传到 Google。这实质上是让 Google 可以使用每一部 Android 设备来更新 Wi-Fi 位置信息，并使数据始终保持是最新的。

用 Wi-Fi 作为位置源的主要好处之一是，在 GPS 不能提供位置数据的环境下，设备依然能够获取位置信息。正如上一节所述，GPS 在室内甚至是在会导致信号问题的城市高楼环境下使用时都存在问题。与此相反，城市环境可能会增加 Wi-Fi 定位的准确性，因为有大量的可用 Wi-Fi 网络来确定设备的当前位置。

局限性

和 GPS 一样，使用 Wi-Fi 作为位置源也有局限性。首先，为了确定位置，必须有 Wi-Fi 网络在设备扫描范围内。此外，网络必须有一个未配置为对 Android 设备不可见的公共广播服务集标识(service set identifier，SSID)。如果接入点的 SSID 以_nomap 结尾，则它们的信息不会发送到 Google 位置服务。

此外，Wi-Fi 接入点位置的变化可能会导致生成的位置数据不准确。例如，现在很多人在家里有无线网络，作为日常使用。假设一个 Android 设备已经被配置为使用 Google 位置服务，Android 会将接入点的 MAC 地址和位置发送到 Google 位置服务。如果用户改变接入点的位置(如拿到度假屋)，那么在使用 Wi-Fi 作为位置源时，定位服务可能会定位到错误的位置。

虽然位置服务允许 Android 设备更新接入点的位置，但是 Google 不允许用户显式地设置一个接入点的位置。Android 设备会把信息上传到位置服务，但是可能要等到其他设备

能够确认这个位置变化，位置服务才会更新。

2. 使用基站 ID

除了使用 Wi-Fi 来确定设备位置，Android 也可以使用蜂窝网络。蜂窝网络使用和 Wi-Fi 接入点类似的方式来确定设备位置。

工作原理

为了正常工作，移动设备必须和一个基站通信。随着设备的移动，导致接近的基站信号增强，设备会连接到不同的基站。假设基站的位置是已知的，又知道了设备连接的基站 ID 以及设备先前连接过的基站，就可以了解设备的位置。

Android 和 Google 位置服务使用与处理 Wi-Fi 数据类似的方式来匹配基站 ID 和位置数据。一旦设备被配置为使用网络提供者，除了可见的无线网络之外，当前的基站数据也会被收集。对于基站来说，此数据包括设备当前连接的基站以及设备当前的 GPS 位置。通过这些信息，Google 位置服务就可以开发出一张基站“地图”。

同样，通过允许 Android 设备更新基站 ID 信息，Google 位置服务可以保持信息库不断更新，并且随着条目的增加，精确度也会随之提高。

在设备需要确定当前位置时，它会将当前连接的基站 ID 以及之前连接过的历史基站信息发送到 Google 位置服务。有了这些信息，Google 位置服务就可以基于它已有的基站网络提供设备当前的位置信息。如果多个基站 ID 被发送到 Google 位置服务，Google 就可以通过三角定位来提高位置准确度。如果设备只上传单一的基站 ID，Google 位置服务是无法做到这一点的。

局限性

使用基站 ID 定位的局限性和使用 Wi-Fi 定位所存在的局限性类似。然而，因为相比无线接入点来说，基站位置不太可能改变，所以由此带来的在 Wi-Fi 接入点中的一系列问题就不存在了。

然而，和 Wi-Fi 接入点数据一样，Google 位置服务必须有设备发送过来的基站 ID 数据，这样才能提供位置数据。

1.2　小结

本章概述了 Android 中位置提供者的工作原理以及它们的局限性。决定在不同情况下使用何种类型的位置提供者可以是一个复杂的主题，后面的章节将会详细讨论这个主题。

第 2 章

确定设备当前位置

本章内容：

- 介绍 Android Location API 组件
- 介绍 Android 中不同的位置信息源
- 使用 Location API 确定设备当前位置的范例

移动应用开发人员经常需要确定设备的当前位置。通过了解设备的位置，应用开发人员可以为各种应用增加更多的功能。位置数据对于 Google Maps 和 Google Navigator 这样的应用是一个关键的组成部分，与此同时，Google 搜索、Twitter 和 Facebook 也会使用位置数据为它们已经收集的数据增加另一个维度的内容。

对于已经决定使用位置数据的开发人员来说，Android 提供了相当强大的 API 来使用它的位置服务。虽然表面上看此 API 的使用很简单，但是大量的细节——如电池的寿命和位置数据的精确度，都是需要开发人员考虑的。

本章介绍 Android 中的位置服务，提供位置服务部分 API 的导览。另外，本章还给出一个范例应用，可以回答大多数基础的 Android 问题(“我在哪”)，并且把位置数据呈现在界面上。图 2-1 展示了应用的界面。

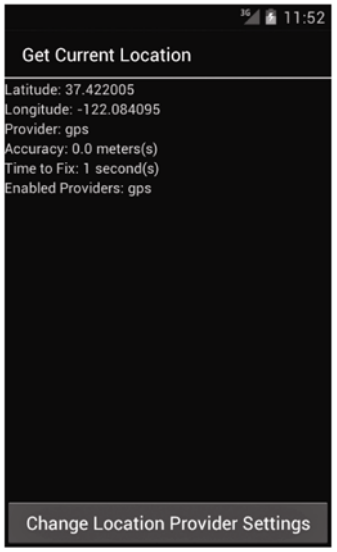


图 2-1　当前位置应用

2.1　了解你的工具

本章通过对 Android 提供的一些工具进行概览来开始对于位置服务的讨论。在解决问题之前，最先要做的是看看需要使用哪些工具。在 Android 中，获得位置数据最主要的几个类位于 android.location 包中。在范例应用中，需要使用 location 包中的 5 个成员。这 5 个成员正是在 Android 中处理位置数据时最常用到的。

类：

- LocationManager
- LocationProvider
- Location
- Criteria

接口：

- LocationListener

图 2-2 高度概括了位置组件之间的协作。因为这些成员非常重要且会频繁使用，后面的小节会对每个成员进行更详细的介绍。

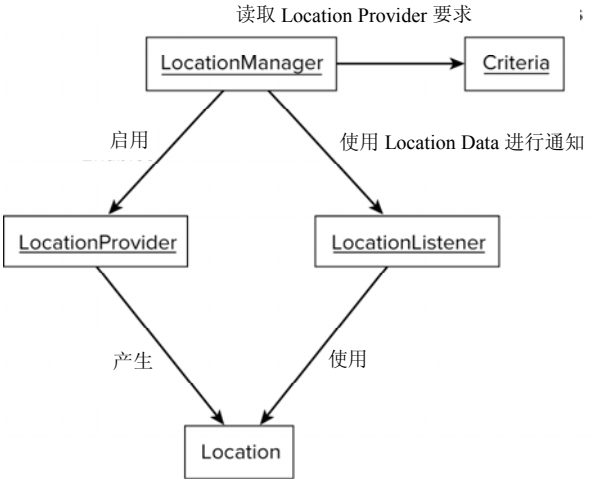


图 2-2　Android 位置组件

2.1.1　LocationManager

使用 Android 位置服务的主入口是 LocationManager。LoctionManager 使得应用可以告诉 Android 何时它想要获取位置信息的更新，何时它不再需要获取位置更新。另外，LocationManager 提供当前位置系统信息，如可用的位置提供者、已经启用的位置提供者、GPS 状态信息等。LocationManager 也可以提供最近已知的(缓存的)位置。

2.1.2　Location Provider

Location Provider 是 Android 中不同位置信息来源的抽象。Android 提供不同来源的数据，这些数据都有着显著不同的特征。虽然每个提供者产生不同的位置数据，但是它们都

以相同的方式与应用通信，并且以相同的方式为应用提供类似的数据。

2.1.3　Location

Location 类封装了从位置提供者提供给应用的位置数据。它包含了类似纬度、经度以及海拔这样的量化数据。当应用收到一个 Location 对象时，就可以对其进行应用特定的处理。

虽然 Location 类有着各种各样的位置数据属性，但不是所有的位置提供者都会填充所有的属性，这对于 Location 类是很重要的一点。举个例子，如果一个应用使用的位置提供者没有提供海拔数据，Location 的实例就不会包含海拔信息。Location 类还提供了一些方法，供应用检查实例中是否包含相应的信息(在这里就是 hasAltitude()方法)。

2.1.4　Criteria

应用可以使用 Criteria 类来查询 LocationManager，获取包含特定特征的位置提供者。在应用并不关心实际使用了哪个提供者，而是更关心位置提供者是否包含一些特定的特征时，这会很有用。Criteria 类使得应用不必担心直接使用单独的位置提供者的实施细节。一旦被实例化，应用可以设置/取消设置 Criteria 类的属性，以反映应用所关心的位置提供者的特征。表 2-1 提供了 Criteria 类的属性列表，使用这些属性可以选择位置提供者。

表 2-1　位置标准属性

属　　性	解　　释	可 能 的 值
accuracy	指明位置提供者的整体精确度级别	Criteria.ACCURACY_FINE 或者 Criteria.ACCURACY_COURSE
altitudeRequired	指明位置提供者是否需要提供海拔信息	true 或者 false
bearingRequired	指明位置提供者是否需要提供方位(移动方向)信息	true 或者 false
bearingAccuracy	方位信息所需的精确度	Criteria.ACCURACY_HIGH 或者 Criteria.ACCURACY_LOW
costAllowed	指明位置提供者是否允许收取用户费用	true 或者 false
horizontalAccuracy	经纬度值所需的精确度	Criteria.ACCURACY_LOW、Criteria.ACCURACY_MEDIUM 或者 Criteria.ACCURACY_LOW
powerRequirement	位置提供者所需要的电量	Criteria.POWER_LOW、Criteria.POWER_MEDIUM 或者 Criteria.POWER_LOW
speedRequired	指明位置提供者是否需要提供速度信息	true 或者 false
speedAccuracy	速度信息所需的精确度	Criteria.ACCURACY_HIGH 或者 Criteria.ACCURACY_LOW
verticalAccuracy	海拔信息所需的精确度	Criteria.ACCURACY_HIGH 或者 Criteria.ACCURACY_LOW

2.1.5　LocationListener

LocationLsitener 接口包含了一组回调方法，这些方法在设备当前位置或者位置服务状态有所改变时被调用。LocationManager 使得应用可以注册/注销一个用于处理状态变化的位置监听器的实现。

有两种方法可以从位置服务获取位置更新：使用 LocationListener 或者 PendingIntent。本章重点介绍 LocationListener 的使用，而 PendingIntent 的使用推迟到第 3 章介绍。

至此，用来实现应用的工具都已经介绍过了，下面的小节将会挖掘请求和处理位置信息的机制。

2.2　设置 Android 清单

和 Android 提供的很多服务一样，位置服务需要应用在 Android 清单中声明使用它的意图。Android 清单声明必须定义请求的位置数据精度。和其他 Android 权限一样，用户最终会在安装时看到这些需要的权限清单，用户也可以在看到这些清单以后拒绝安装。一些用户在不确定应用需要位置信息的目的时，对于同意应用确定他们的位置有些过于拘谨。添加一些多余的权限很容易就会把用户吓走。

处理实时位置数据的两个权限是 android.permission.ACCESS_FINE_LOCATION 和 android.permission.ACCESS_COARSE_LOCATION。顾名思义，这些权限用来定义从位置服务提供给应用的精确度级别。最终，这些权限决定应用中可以使用哪些位置提供者。因为 android.permission.ACCESS_FINE_LOCATION 提供更精确的位置数据，所以可以在未明确指定 android.permission.ACCESS_COARSE_LOCATION 权限时同时使用细粒度和粗粒度的位置数据。然而 android.permission.ACCESS_COARSE_LOCATION 仅允许提供给应用粗粒度的位置数据。

对于本章中的示例应用，需要高精确度的位置数据。所以，需要在 AndroidManifest.xml 中添加以下代码片段：

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

如果请求需要的权限失败，在请求位置更新时会抛出 java.lang.SecurityException 运行时异常。

2.3　确定合适的位置提供者

Android 中不同的位置数据源提供不同级别的精确度和电量消耗。确定何时使用何种

提供者对应用的整体用户体验有着很大的影响。在 Android 中可用的位置提供者有：

- GPS 位置提供者(GPS location provider)
- 网络位置提供者(Network location provider)
- 被动位置提供者(Passive location provider)

应用可以使用以下两种方式之一来声明使用何种位置提供者：一种是通过 LocationManager 明确地定义所需的位置提供者，另一种是指定一个 Criteria 对象的属性，并将这个对象传给 LocationManager。使用 Criteria 对象，对于用户在运行时自定义位置数据来源是非常有用的。这可能对用户来说是很重要的，因为使用有些位置提供者并不是免费的。

2.3.1　GPS 位置提供者

GPS 位置提供者使用轨道卫星和时间来确定设备的当前位置，而且往往会产生最准确的位置数据。然而，因为它依赖于单独的无线电模块，GPS 提供者也会比其他的位置提供者消耗更多的电量。这是否会是一个重大问题取决于应用需要活跃地获取和处理位置数据的时间长度。

除了消耗更多的电量外，GPS 位置提供者也会消耗较长的时间以获取修正(位置数据)。Time to first fix(首次修正时间，TTFF)值超过一分钟是常见的，并且不同的设备或者不同的 Android 版本之间会有很大的差别。另外，在室内获得 GPS 修正是不可能的，因为通常都需要能直接看到天空。留意 TTFF 非常重要，由于应用等待位置数据而阻塞用户执行任务，这通常是一个糟糕的主意。

2.3.2　网络位置提供者

网络位置提供者使用两种数据源来提供位置修正：Wi-Fi 网络定位以及基站定位。网络提供者的 TTFF 基本上小于 GPS 提供者。然而，网络提供者产生的位置数据精确度远低于 GPS 提供者。根据应用的不同需求，有时也需要用 GPS 提供者的定位精确度来换取网络提供者较低的 TTFF 值。网络提供者也可能比 GPS 提供者消耗更少的电量，因为它允许用户关掉 GPS 无线电和 Wi-Fi 无线电(有可能)。

2.3.3　被动位置提供者

被动位置提供者允许应用不必显式地从 LocationManager 请求位置更新信息，就能获取位置数据。被动提供者在另一个应用使用 GPS 或者网络提供者显式地请求位置更新时，提供相应的位置更新。这使得应用可以利用其他应用请求的位置信息，并且防止 Android 发出特殊的位置数据请求。

初看之下，并不能清楚如何使用这个提供者。从本质上讲，被动位置提供者需要应用在后台不消耗额外电量就获取位置更新，因为它只在其他应用获取更新时才进行更新。

这意味着应用无法控制用来获取位置更新的其他提供者，或者更新抵达的频率(其他应用会在创建 LocationManager 时定义这一点)。正因为如此，使用被动提供者必须授予使用 android.permission.ACCESS_FINE_LOCATION 权限，这样就可以获取 GPS 提供者和网络

提供者的数据。收到的 Location 对象会包含位置数据源的信息。

被动提供者不能确保会获取到位置更新。如果没有其他应用获取位置更新，被动提供者同样不会获取。正因为如此，被动提供者通常不适合当应用在前台且和用户交互时使用。在后台运行时保持最新的数据，并且不用明确地请求位置数据时，可以使用被动提供者。

在 Android 设备中做一个“好公民”，并且在应用退出时取消位置更新请求，这对于应用来说是很好的形式。如果一个应用直接关闭而不是使应用在后台运行(用户点击“back”按钮而不是按下“home”按钮)，那么应用即使是在使用被动提供者，也需要注销更新。

2.3.4　精确度与电池寿命

选择位置提供者时的常见主题就是在更高的精确度和增长的电量消耗之间做决定。虽然很多应用通过更精确的数据能够有更好的表现，但事实上，很多时候并不是真的需要精确度，尤其在高精确度会带来额外电量消耗的情况下。

表 2-2 提供了 Android 中可用的位置提供者汇总。

表 2-2　位置提供者

位置提供者	需要的权限	电量消耗	精确度
GPS 提供者	android.permission. ACCESS_FINE_LOCATION 或者 android.permission. ACCESS_COARSE_LOCATION	比其他提供者消耗更多的电量	提供最精确的数据
网络提供者	android.permission. ACCESS_COARSE_LOCATION	比 GPS 提供者消耗的电量要少	提供的精确度要比 GPS 提供者低
被动提供者	android.permission. ACCESS_FINE_LOCATION	N/A	N/A

在使用位置服务时，使用何种位置提供者作为合适的位置数据源是很重要的决定。和其他许多开发决策一样，需要考虑取舍。

2.4　获取位置更新

在深入查看 Java 代码之前，还有一个主题值得讨论：应用是如何获取位置更新的通知的。回顾一下之前的讨论，位置数据通过两种方式传递给应用：一种是直接调用 LocationListener，另一种是使用广播 Intent。LocationListener 方式是相对简单的方式(也是这章中的示例应用所使用过的)，但是通过广播 Intent 的方式可以提供更大的灵活性，特别是在需要提供位置更新信息到多个应用组件的时候。

无论采用哪一种方式，应用都需要告诉 LocationManager，何时准备开始获取更新以及何时停止获取位置更新。位置更新发送到应用的方式取决于应用使用 LocationManager 注

册位置更新的方式。

2.4.1　使用 LocationListener 获取位置更新

实现了 LocationListener 的对象会通过对其 onLocationChanged()方法的调用而收到位置更新通知。这个将会收到位置更新的 LcoationListener 实例是通过 LocationManager 注册的。当 LocationManager 有一个新的位置要提供时，它调用每个监听器的 onLocationChanged()方法。关于 LocationListener 的进一步讨论延后到 2.5.1 节，在这一节中会讲解示例应用的 Java 代码。

2.4.2　使用广播 Intent 来获取位置更新

在应用需要多个应用组件都能获取更新的情况下，有一个含有位置更新的 Intent 广播可以提供更高的灵活性。为了使用广播 Intent，应用需要实现 BroadcastReceiver，并且注册它以获取位置更新的 Intent。这可以在 Android 清单中完成，也可以在运行时完成。第 4 章创建的示例应用中将包含广播 Intent 的用法。

2.5　实现示例应用

本节提供把所有的定位 API 集中起来并开始获取位置数据的详细实现。

示例应用有一个活动，叫做 CurrentLocationActivity，用来展示当前的位置，而且包含了一个按钮，允许用户可以启用/停用位置提供者。一旦应用得到一个位置，就会在屏幕上显示位置的详细信息。这可以让用户在实际设备上看到位置服务的工作，并且让用户开始感受不同位置提供者之间精确度以及 TTFF 值的差别。这对于理解精确度和 TTFF 的细节，以及在开发应用的过程中对于如何关联到不同提供者的决策，是非常重要的。

2.5.1　实现 LocationListener

为了实现 LocationListener，一个类必须包含以下方法的具体实现：

- abstract void onLocationChanged(Location location)
- abstract void onProviderDisabled(String provider)
- abstract void onProviderEnabled(String provider)
- abstract void onStatusChanged(String provider, int status, Bundle extras)

接下来的小节将会讨论这些方法。

1. onLocationChanged()

应用最有可能与之交互的方法就是 onLocationChanged()方法。该方法会在一个新的位置已经准备好给应用使用时被调用。该方法的唯一参数是包含了详细位置(纬度、经度、海拔高度等)的 Location 对象。有时这是应用唯一需要从 LocationListener 实现的方法。然而，应用需要实现其他方法来避免出现编译错误，实现可以是空(我喜欢添加一个评论，指明是

为了以后的开发故意留下空白)。在这个应用中，onLocationChanged()方法简单地使用传过来的 Location 对象，并把它的数据呈现在 UI 控件上(见程序清单 2-1)。

程序清单 2-1：接收位置更新

```
@Override
public void onLocationChanged(Location location) {

    latitudeValue.setText(String.valueOf(location.getLatitude()));
    longitudeValue.setText(String.valueOf(location.getLongitude()));
    providerValue.setText(String.valueOf(location.getProvider()));
    accuracyValue.setText(String.valueOf(location.getAccuracy()));

    long timeToFix = SystemClock.uptimeMillis() - uptimeAtResume;

    timeToFixValue.setText(String.valueOf(timeToFix / 1000));

    findViewById(R.id.timeToFixUnits).setVisibility(View.VISIBLE);
    findViewById(R.id.accuracyUnits).setVisibility(View.VISIBLE);
}
```

2. onProviderDisabled()和 onProviderEnabled()

onProviderDisabled()和 onProviderEnabled()方法为应用提供了一个途径，使得用户从设置菜单中启用或者停用位置提供者时，应用将会得到通知。举例来说，假设一个用户现在正在运行一个应用，而且决定通过按下 home 按钮把应用放到后台并返回到桌面。从那里用户可以进入设备设置，并且启用或者禁用不同的位置提供者。应用可能对这些操作感兴趣。如果用户要启用 GPS 提供者，应用将会得到设备更精确的位置信息。

onProviderDisabled()和 onProviderEnabled()方法是 Android 中用于让应用知道提供者状态变化的方式。两个方法都会使用一个字符串作为参数，用于指定启用或者禁用的位置提供者名称。这个 String 类型的提供者名称会匹配 LocationManager 中的静态常量，从而确定哪个提供者的状态发生了改变。这两个方法和 LocationManager.getProviders()能很好地合作，LocationManager.getProviders()方法可以用来初始化注册当前可用的提供者，并且在 onProviderDisabled()或者 onProviderEnabled()方法被调用时动态地增加或删除被启用或者禁用的提供者。

3. onStatusChanged()

onStatusChanged()方法在提供者离线或者恢复在线时被调用。这和之前的小节中用户启用和禁用提供者是不同的情景。在这个情景中，用户并没有改变位置设置，而是实际的提供者状态发生了变化。

在该方法的参数中，一个字符串代表提供者，一个 int 值代表当前状态，一个 Bundle 含有可选的数据。提供者的名称与传给 onProviderEnabled()和 onProviderDisabled()方法的是相同的字符串。状态是在表 2-3 中列出的其中一种状态。

表 2-3　onStatusChanged()状态值

值	状　　态
LocationProvider.OUT_OF_SERVICE	LocationProvider 当前离线，并且可能不会很快恢复在线
LocationProvider.TEMPORARILY_UNAVAILABLE	LocationProvider 当前离线，并且将会很快恢复在线
LocationProvider.AVAILABLE	LocationProvider 当前在线

Bundle 参数包含了可选的提供者特定信息。例如，一个 GPS 提供者的 Bundle 会包含用于给出位置更新的卫星的数量。

2.5.2　获取 LocationManager 的句柄

因为 LocationManager 是进入位置服务的前门，应用需要得到它的引用。这是通过调用 Activity.getSystemService(LOCATION_SERVICE)来完成的。这通常在 Activity 的 onCreate()方法中完成，因为在 Activity 的生命周期中 LocationManager 被多次调用是很常见的。因为 onCreate()方法是活动的生命周期中最早被调用的方法，所以在这里获取位置管理器的引用是合适的。对于本章的示例应用，onCreate()方法剩下的工作就是获取用于保持和呈现位置数据的 UI 视图的引用。程序清单 2-2 给出了 onCreate()方法的实现。

程序清单 2-2：获取 LocationManager 的引用

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.current_location);

    locationManager = (LocationManager) getSystemService(LOCATION_SERVICE);

    latitudeValue = (TextView) findViewById(R.id.latitudeValue);
    longitudeValue = (TextView) findViewById(R.id.longitudeValue);
    providerValue = (TextView) findViewById(R.id.providerValue);
    accuracyValue = (TextView) findViewById(R.id.accuracyValue);
    timeToFixValue = (TextView) findViewById(R.id.timeToFixValue);
    enabledProvidersValue = (TextView) findViewById(R.id.enabledProvidersValue);
}
```

程序清单 2-3 给出了显示当前位置数据的活动的布局，位置数据有：latitudeValue、longitudeValue、providerValue、accuracyValue、timeToFixValue 和 enabledProviderValue。

程序清单 2-3：CurrentLocationActivity 的布局

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

```
        android:orientation="vertical"
        android:layout_width="match_parent"
        android:layout_height="match_parent">

        <TextView android:id="@+id/latitudeLabel"
            android:layout_height="wrap_content"
            android:layout_width="wrap_content"
            android:text="@string/latitudeLabel"
            android:layout_alignParentTop="true"
            android:layout_marginRight="4dip" />

        <TextView android:id="@+id/latitudeValue"
            android:layout_height="wrap_content"
            android:layout_width="wrap_content"
            android:layout_alignTop="@id/latitudeLabel"
            android:layout_toRightOf="@id/latitudeLabel" />

        <TextView android:id="@+id/longitudeLabel"
            android:layout_height="wrap_content"
            android:layout_width="wrap_content"
            android:text="@string/longitudeLabel"
            android:layout_below="@id/latitudeLabel"
            android:layout_marginRight="4dip" />

        <TextView android:id="@+id/longitudeValue"
            android:layout_height="wrap_content"
            android:layout_width="wrap_content"
            android:layout_alignTop="@id/longitudeLabel"
            android:layout_toRightOf="@id/longitudeLabel" />

        <TextView android:id="@+id/providerLabel"
            android:layout_height="wrap_content"
            android:layout_width="wrap_content"
            android:text="@string/providerLabel"
            android:layout_below="@id/longitudeLabel"
            android:layout_marginRight="4dip" />

        <TextView android:id="@+id/providerValue"
            android:layout_height="wrap_content"
            android:layout_width="wrap_content"
            android:layout_alignTop="@id/providerLabel"
            android:layout_toRightOf="@id/providerLabel" />

        <TextView android:id="@+id/accuracyLabel"
            android:layout_height="wrap_content"
            android:layout_width="wrap_content"
            android:text="@string/accuracyLabel"
            android:layout_below="@id/providerLabel"
            android:layout_marginRight="4dip" />
```

```
<TextView android:id="@+id/accuracyValue"
    android:layout_height="wrap_content"
    android:layout_width="wrap_content"
    android:layout_alignTop="@id/accuracyLabel"
    android:layout_toRightOf="@id/accuracyLabel" />

<TextView android:id="@+id/accuracyUnits"
    android:layout_height="wrap_content"
    android:layout_width="wrap_content"
    android:text="@string/metersUnit"
    android:layout_alignTop="@id/accuracyLabel"
    android:layout_toRightOf="@id/accuracyValue"
    android:layout_marginLeft="4dip" />

<TextView android:id="@+id/timeToFixLabel"
    android:layout_height="wrap_content"
    android:layout_width="wrap_content"
    android:text="@string/timeToFixLabel"
    android:layout_below="@id/accuracyLabel"
    android:layout_marginRight="4dip" />

<TextView android:id="@+id/timeToFixValue"
    android:layout_height="wrap_content"
    android:layout_width="wrap_content"
    android:layout_alignTop="@id/timeToFixLabel"
    android:layout_toRightOf="@id/timeToFixLabel" />

<TextView android:id="@+id/timeToFixUnits"
    android:layout_height="wrap_content"
    android:layout_width="wrap_content"
    android:text="@string/secondsUnit"
    android:layout_alignTop="@id/timeToFixLabel"
    android:layout_toRightOf="@id/timeToFixValue"
    android:layout_marginLeft="4dip" />

<TextView android:id="@+id/enabledProvidersLabel"
    android:layout_height="wrap_content"
    android:layout_width="wrap_content"
    android:text="@string/enabledProvidersLabel"
    android:layout_below="@id/timeToFixLabel"
    android:layout_marginRight="4dip" />

<TextView android:id="@+id/enabledProvidersValue"
    android:layout_height="wrap_content"
    android:layout_width="wrap_content"
    android:layout_alignTop="@id/enabledProvidersLabel"
    android:layout_toRightOf="@id/enabledProvidersLabel" />

<Button android:id="@+id/changeLocationProviderSettings"
    android:layout_height="wrap_content"
```

```
        android:layout_width="match_parent"
        android:text="@string/changeLocationProviderSettingsText"
        android:onClick="onChangeLocationProvidersSettingsClick"
        android:layout_alignParentBottom="true" />

</RelativeLayout>
```

代码片段 current_location.xml

现在应用有了 LocationManager 的引用，已经准备好请求位置服务以获取位置信息。

2.5.3　请求位置更新

应用现在已经准备好在它可用时让 Android 提供位置信息。重要的是要理解应用并不能按需要求 Android 提供位置信息。应用只能请求在更新的位置信息可用时收到通知。

示例应用只需要单个位置更新，所以调用 LocationManager 中的一个方法。在此需要使用 requestSingleLocation()方法。查看 Android 参考文档，可知道有两种类型的 LocationManager.requestSingleLocation()方法。一种类型传递一个 PendingIntent，以此广播一个带有位置数据的 Intent；另一种传递一个 LocationListener，以此直接获取回调。同样，这个实例应用还使用了 LocationListener 方式。CurrentLocationActivity 通过实现 LocationListener 接口创建了一个 LocationListener。这使得应用把所有的定位代码放在一个类里。

在通过 LocationManager 注册一个 LocationListener 之前，应用开发人员必须决定应用使用的位置提供者类型。对于这个应用，决定权留给最终用户，因为应用只会使用用户启用的位置提供者。为了获取启用的位置提供者清单，需要创建 Criteria 对象，并将其属性设置为同时包括网络位置提供者和 GPS 位置提供者。网络提供者和 GPS 提供者至少提供一个粗略的位置修正，所以将 Criteria.ACCURACY_COARSE 传给 Criteria.setAccuracy()方法时将会同时考虑两个提供者。然后，将已经初始化的 Criteria 实例以及一个指明仅返回可用位置提供者的布尔值(硬编码为 true)传给 getProviders()方法。返回列表中的每一个提供者就用来获取位置数据。

每次 CurrentLocationActivity 呈现给用户时都需要一个新的位置(因为用户允许在应用中启用或者禁用位置提供者)，因此应用在位置更新可用时，会在 onResume()方法中正式向 LocationManager 请求提供一个位置更新，如程序清单 2-4 所示。

程序清单 2-4：使用 LocationManager 注册

```
protected void onResume() {
    super.onResume();

    StringBuffer stringBuffer = new StringBuffer();

    Criteria criteria = new Criteria();
    criteria.setAccuracy(Criteria.ACCURACY_COARSE);
```

```
        enabledProviders = locationManager.getProviders(criteria, true);

        if (enabledProviders.isEmpty())
        {
            enabledProvidersValue.setText("");
        }
        else
        {
            for (String enabledProvider : enabledProviders)
            {
                stringBuffer.append(enabledProvider).append(" ");

                locationManager.requestSingleUpdate(enabledProvider,
                        this,
                        null);
            }
            enabledProvidersValue.setText(stringBuffer);
        }

        uptimeAtResume = SystemClock.uptimeMillis();

        latitudeValue.setText("");
        longitudeValue.setText("");
        providerValue.setText("");
        accuracyValue.setText("");
        timeToFixValue.setText("");

        findViewById(R.id.timeToFixUnits).setVisibility(View.GONE);
        findViewById(R.id.accuracyUnits).setVisibility(View.GONE);
    }
```

代码片段 CurrentLocation.java

到目前为止，本章介绍了如何注册一个 LocationListener 来获取位置更新。最后一步是在应用不需要那些位置更新时，将它们全部注销。

2.5.4　自行清理

此时，应用已经准备好在 CurrentLocationActivity 中获取和处理位置数据。最后一步是让应用在不需要位置更新时，通过注销位置监听器自行清理数据。忘记注销位置监听器可能导致提供者以及底层硬件始终保持活动状态，从而浪费了电池寿命。没有注销 GPS 提供者的位置监听器会导致(可用的)GPS 提供者仍然活跃地搜索和计算位置数据。这对用户是可见的，因为 GPS 提供者有它自己的图标来提醒用户这个问题。在不需要 GPS 时仍让它运行是很糟糕的做法，也会在 Android 市场引来负面的反馈。

在 CurrentLocationActivity 不再与用户交互时，应用不再需要任何位置更新。如程序清单 2-5 所示，它会在 onPause()方法中注销 LocationListener。

程序清单 2-5：移除 LocationListener

```
@Override
protected void onPause() {
    super.onPause();
    locationManager.removeUpdates(this);
}
```

如果使用了 BroadcastReceiver 方式，应用需要再次调用 locationManager.removeUpdates()方法，而且需要将之前传给 registerSingleUpdate()方法的 PendingIntent 对象传给它。

现在已经完成了初始化、处理位置更新以及自行清理这几部分的编码，示例应用实现的下一步就是在应用运行时，响应用户启用/禁用位置提供者的操作。

2.5.5　启动位置设置活动

应用最后值得讨论的细节是屏幕上可以让用户改变位置提供者设置的按钮。为了从特定的位置提供者获取位置数据，应用需要保证位置提供者已经被用户启用。如果用户当前并没有启用提供者，位置设置活动可以让用户能够在不离开应用的情况下启用提供者。在示例应用中的 Provider Settings 按钮被按下时，就会发生这种情况。正如程序清单 2-6 所证实的那样，这一点的实现其实很简单，这个操作在点击按钮的处理程序中完成。

程序清单 2-6：启动位置设置活动

```
public void onChangeLocationProvidersSettingsClick(View view)
{
    startActivity(new Intent(Settings.ACTION_LOCATION_SOURCE_SETTINGS));
}
```

图 2-3 即为位置设置活动界面。

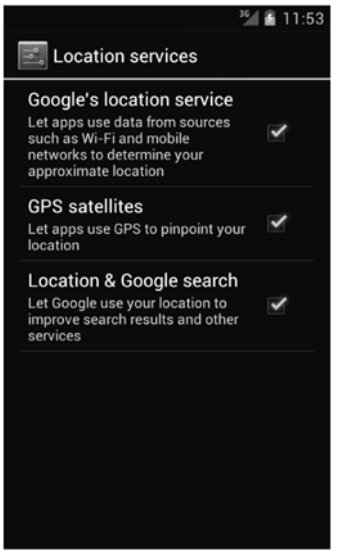


图 2-3　位置设置界面

现在你有了一个应用能够确定当前的位置，试着在实际设备上加载该应用，并且花一点时间尝试不同的位置提供者，特别注意它们在 TTFF 和精确度方面的差别。试着在不同的环境运行该应用(室内和室外、晴天和多云、城市和乡村)，观察它们是如何影响不同的位置提供者的。花一些时间了解提供者的工作原理，可以让你了解它们的局限性。

2.6　小结

本章概览了使用 Android 位置服务需要的基础 SDK 元素。其中讨论了一些基础类，并且研究了一个确定设备当前位置的简单应用的实现。该应用只是建立在理想状况下，并没有处理应用在使用位置服务时会遇到的现实世界中的问题。本章呈现的应用足以启动和运行，但下面的几章会介绍如何充分利用 Android 的位置服务。