HZ BOOKS
华章科技

资深Android工程师开发经验悉心总结，资深Android工程师翻译，译著双馨
汇集Android应用开发中50个常见的实用技巧和疑难问题解决方法，为打造高质量Android应用提供有价值的实践指导，迅速提升应用开发能力

移动开发

MANNING

50 Android Hacks

打造高质量Android应用

Android开发必知的50个诀窍

（美） Carlos Sessa 著
杨云君◎译

机械工业出版社
China Machine Press

移动开发

打造高质量Android应用：

Android开发必知的50个诀窍

50 Android Hacks

（美） Carlos Sessa 著

杨云君 译

机械工业出版社
China Machine Press

图书在版编目（CIP）数据

打造高质量 Android 应用：Android 开发必知的 50 个诀窍 /（美）塞萨（Sessa, C.）著；杨云君译 . —北京：机械工业出版社，2014.4
（移动开发）
书名原文：50 Android Hacks

ISBN 978-7-111-46136-4

I. 打… II. ① 塞… ② 杨… III. 移动终端 – 应用程序 – 程序设计 IV. TN929.53

中国版本图书馆 CIP 数据核字（2014）第 045393 号

本书版权登记号：图字：01-2014-0806

打造高质量 Android 应用：Android 开发必知的 50 个诀窍

（美） Carlos Sessa 著

出版发行：机械工业出版社（北京市西城区百万庄大街 22 号 邮政编码：100037）
责任编辑：谢晓芳
印 刷：
版 次：2014 年 4 月第 1 版第 1 次印刷
开 本：147mm × 210mm 1/32
印 张：7.625
书 号：ISBN 978-7-111-46136-4
定 价：49.00 元

凡购本书，如有缺页、倒页、脱页，由本社发行部调换
客服热线：（010）88378991 88361066 投稿热线：（010）88379604
购书热线：（010）68326294 88379649 68995259 读者信箱：hzjsj@hzbook.com

推 荐 序

Android 是一个迅速向各个领域扩张的生态系统。每天都会有厂商发布新的设备和外观设计，每天都会有客户购买和激活上百万台设备，每天都会有用户下载和试用新的应用程序。开发美观、有吸引力并且令用户满意的应用程序来丰富和完善这个生态系统是每一位开发者（希望也包括读者本人）应尽的责任，只有这样才能为用户提供更好的交互体验。

Android 是一个软件开发平台，它诞生于 2003 年年底，由 Danger 公司（开发流行的 Sidekick 手机的公司）的前雇员开发。2005 年，运营 Android 的 Danger 公司被 Google 公司收购。三年后，HTC Dream（G1）作为第一款运行 Android 操作系统的手机正式发布。此后三年，虽然硬件与平台发生了很大的更新和迭代，但是 Android 依然保持为一个单纯的手机操作系统。

2011 年，Google 公司为 Android 添加了新的特性，增加了对两种设备的支持：平板电脑和电视。这不仅标志着官方第一次扩充 Android 所支持设备的种类，还激发了厂商对其他潜在支持设备的兴趣。现在，Android 已经可以运行在笔记本电脑、手表、视频游戏机、车载音响等多种设备上。我相信在不久的将来 Android 会支持更多的设备。

作为应用开发者，理解平台的多样化和发展方向是非常重要的。在 Android 上做开发已经不像为竖屏手机设计软件那么简单了。尽管这意味

着开发者开发应用程序的工作量增加了，但是，最终结果却是无论应用程序运行在哪种设备上，都会为使用者提供良好的用户体验。

在开发应用程序的过程中，除了个人创造力和开发意愿以外，开发者还需要具备三样东西：平台开发文档、开源社区以及整合资源并融会贯通的能力。平台开发文档比较容易获取，最新版本托管在 http://developer.android.com 网站上。开源社区有 GitHub、Google Code、Stack Overflow 以及其他类似网站，这些网站提供了开源库、代码片段以及能够简化程序开发的设计模式。此外，开发者还需要具备把上述零散的知识整合到应用中的能力。这个整合的过程可不像搭积木一样简单，如果那样，任何人都可以开发应用了。本书便是一本分析如何整合资源的指南。

本书以示例程序的形式分析如何解决 Android 开发过程中出现的常见问题。书中有些示例程序相对简单，有些示例程序相当复杂。这些示例程序分享了一些只有零散或者零星文档可查但是却经常困扰开发者的问题。本书不仅仅是一本单纯学习和掌握 Android 开发技巧的书，更是一本填补空白的书。

精心设计一个能够动态支持所有 Android 设备的应用是一项艰巨的任务。通过学习本书以及类似出版物和在线资源提供的知识，我希望能提升读者开发和发布应用的能力。除此之外，我跟读者一样，也是一名开发者和热心用户，我也在耐心等待下一个精彩应用的出现，或许读者就是那个开发它的人。

Jake Wharton

Android 工程师

序

早在2009年我便开始研究Android。当时，Android 1.5刚刚发布并且显示出巨大的发展潜力。

在2009年7月，得益于澳大利亚的一位朋友，我拿到了第一台运行Android操作系统的设备，这台设备就是运行Android 1.5的HTC Magic手机。说实话，这台设备的运行速度比我想象的要慢，但是我依然通过它开始研究Android的API，并且根据自己的需求开发应用程序，然后在这台设备上运行这些应用。当时，我预感到Android会获得更多人的关注，我相信如果我能为Android开发一款应用程序，这款应用程序一定能被许多人使用。

事实证明，我的预感是正确的。不久后，Android开发的大幕便拉开了，而且发展得越来越快。一时间，许多支持Android平台的工具和第三方开发库便涌现出来，从cocos2d-x这样的游戏框架到Apache Maven这样的编译系统，几乎无所不包。

2010年11月，我受邀审阅Manning出版社出版的《Android in Practice》（www.manning.com/collins/）一书。当深度参与到这项工作之后，我突然想到，我可以用另一种方法写一本Android开发的书。我打算模仿Joshua Bloch所著的《Effective Java》（www.amazon.com/Effective-Java-2nd-Joshua-Bloch/dp/0321356683）一书的风格，向读者展示这些年来我在

Android 开发过程中总结的小窍门和开发模式。

从根本上讲，我想在一本书里涵盖开发过程中总结的我知道的所有小窍门，并为这些小窍门提供一定的解释性文档。本书汇聚了开发与众不同的 Android 应用程序所需要的技巧和窍门。

我喜欢《 Effective Java》一书的原因是，这本书没有特定的章节顺序，因此可以自由学习不同章节的内容。每隔一段时间，当回顾此书时，我总能为当前项目找到一些不同的应用程序。在写这本书的时候，我一直牢记一个信条：我想象读者在上班路上或者睡觉前会有兴趣学习本书里的某个 Hack，并且从这个 Hack 中获得对当前项目有益的启发。

我已经在新项目中应用了这本书的内容，我会在特定的工作任务中复用书中的示例代码，使用示例代码向同事解释特定的开发模式。实践证明，本书对我是十分有用的，当然，我也希望这本书对你同样有用。

撰写本书以及书中的示例代码时，我把最小 SDK 版本设置为 1.6。如非特别提及，本书中多数技巧都适用于 Android 1.6 及其以上版本。你会注意到，有一些技巧只适用于最新版本的 Android，但是多数建议和技巧都适用于所有版本。每个 Hack 都提供一个图标，用来表示这个 Hack 适用的最低 SDK 版本。

那么接下来，你就从本书的目录中挑选自己感兴趣的技巧开始学习吧。希望你能从我写的内容中学到尽可能多的知识。

致　谢

每当读到其他书的致谢时，我总会惊讶作者感谢的人竟然如此之多。现在终于明白为什么需要感谢这么多人，当写到这里的时候，我很紧张，生怕遗漏了某个人。

首先，我要感谢的是编辑 Cynthia Kane，她帮助我加工整理整本书的内容。她不仅指出本书中需要修改的每个地方，还处理我英语语言上需要润色的地方，并且帮助我理解图书出版过程中的每个关键环节。修订每一行文字，修改她发现的每一个不足之处，通过这个反复迭代的修正过程，终于完成了这本值得我骄傲的书。

其次应该感谢 Nicholas Chase，Nick 负责支持 Manning 出版社的 XML 文档结构和创作工具。幸运的是，每当我有问题需要请教他时，他的 Skype 总是在线。

Manning 出版团队的其他成员也参与了大量工作。参与这项工作的有 Ozren Harlovic、Kevin Sullivan、Tara McGoldrick Walsh、Benjamin Berg、Katie Tennant、Candace Gillhoolley、Martin Murtonen、Michael Stephens 以及 Maureen Spencer。

感谢我的合著者：William Sanville（Hack 40 和 Hack 41）、Chris King（Hack 26）以及 Christopher Orr（Hack 50）。他们分享了在这些领域的专业知识。

感谢 Cyril Mottier，他深入阅读了本书，并且毫不保留地指出书中他不喜欢或者认为需要改进的地方。他始终对本书保持高要求，我很喜欢跟他合作。非常感谢！

感谢我在 NASA Trained Monkeys 公司的合作伙伴们。他们帮助我审阅了书中大部分内容，并提出很多建设性意见。大部分很酷的 Hack 标题都来源于他们丰富的想象力。

感谢 Android 社区，特别感谢那些对开源软件库有贡献的人们。（这里只提及几个人的名字，他们是：Michael Burton、Manfred Moser、Matthias Käppler、Jake Wharton、Jeremy Feinstein、cocos2d-x 团队、Jan Berkel、Jeff Gilgelt、Xavi Rigau、Chris Banes、James Brechtel 和 Dmitry Skiba）。

感谢审阅本书的每一个人。你们的审阅意见帮助我及时发现疏漏的地方以及需要强化的主题。从我敬佩的人那里获得正面的评价是很有意义的事情。感谢以下审阅者，你们在百忙之中审阅本书，我也希望这本书对你们有一些启发，这些人是：Adam Koch、Alberto Pose、Bill Cruise、Christian Badenas、Frank Ableson、Ignacio Luciani、Jeff Goldschrafe、Joshua Skinner、Matthias Käppler、Maximiliano Gomez Vidal、“ Ming ”、Octavian Damiean、Paul Butcher、Robi Sen、Roger Binns、Shan Coster、Suzanne Alexandra 和 Will Turnage。

感谢我的家人和朋友——你们给予我巨大的支持。

最后要感谢 Mili，你的工作同样重要，每当我需要帮助的时候，你总是在我身边。我爱你。

关 于 本 书

Android 是一个发展势头很好的项目。Android 的第一个正式版本（Android 1.0）发布于 2008 年 9 月 23 日，截至 2010 年年底，Android 已经发展成为首要的智能手机平台。

每当有新版本发布，Android 都会引入一组新的 API 和新特性。尽管在 Android 1.5 ⊖的时期，市场上只有 HTC Dream 手机运行 Android 系统，但是发展至今，Android 系统不仅可以运行在从手机到电视等多种设备上，还可以运行在不同屏幕大小的平板电脑和笔记本电脑上。

上述情况给 Android 开发者带来了两个不小的难题。第一个难题是开发者必须面对和适配 Android 支持的不同类型的设备。虽然有很多方法处理不同的屏幕尺寸和像素密度，但是开发者必须开发出能够运行在各种设备上并且显示正常的应用。另外，需要对各种 Android 设备可能导致的用户体验不一致的情况做出处理。用户对手机和电视的使用习惯是不同的。

第二个难题是 Android 的版本更新问题。这个难题是周而复始的：使用新版本的 Android 系统，意味着开发者可以使用新的 API，新的 API 可以为应用程序增加优秀的功能；但是开发者必须同时支持旧的 Android 版本，因为并不是每个用户都会升级系统，而且目标用户获取和认可应用程序也需要一定时间。

⊖ 代号 Cupcake，纸杯蛋糕。——译者注

开发者需要在两者之间做出选择：要么使用新的 API 功能并发布一个特定版本的应用，满足那些使用新版本 Android 系统的用户；要么采取折中的方法，保证一些新的功能只适用于新版本的 Android 系统。

上述选择最终都由 Android 开发者决定，因此我写这本书的目的是帮助开发者解决这个难题。本书以“问题 / 解决方案”的形式提出开发过程中遇到的问题并给出其解决方法，并对一些已有问题提供了进一步的处理方案。

什么是 Android

Android 是一个基于 Linux 内核的开源操作系统。起初，Android 只支持手机设备，但是发展到现在，Android 可以运行于平板电脑、电视、电脑甚至汽车音响等多种设备。Android 在移动领域赢得了巨大的发展空间，到目前为止，50% 以上的移动设备运行了 Android 操作系统。

运行在 Android 操作系统上的应用通常使用 Java 语言开发，Android 提供了一个强大的 SDK（软件开发工具包）供开发者开发不同类型的应用程序。Android 允许开发者定制几乎所有模块，例如，开发者可以开发定制的墙纸、键盘、桌面以及在其他平台上想都不敢想的功能。

本书读者对象

本书适用于已经学习过 Android 开发的程序员，并且假定读者已经熟悉 Java 编程语言，并理解 Android 平台的基本概念。

本书不仅提供适用于 Android 初学者的技巧，还提供适用于高级开发者的技巧。如果读者正在开发 Android 应用程序，我相信通过本书，你可以学到很多有帮助的知识。

通过以下几个问题，读者可以知道本书是否适合自己：

- ❑ 你是一个 Android 应用程序开发者吗？
- ❑ 你正在绞尽脑汁思考更好的解决方案吗？

- ❑ 你正在寻找新的方法解决编程中出现的问题吗？
- ❑ 你想知道其他人是如何解决类似问题的吗？

如何使用本书

我的建议是：在读者学习每一条 Hack 前，先编译并运行示例代码，这样有助于读者更好地理解每个案例。此外，读者不需要按照特定顺序学习本书，读者可以随时跳转到自己感兴趣的章节开始学习。

本书结构

虽然读者可以灵活选择自己感兴趣的部分学习，不会因为前后章节顺序的原因出现阅读困难，但是读者仍然可以按顺序阅读本书。各章节的概要内容如下：

- ❑ 第 1 章包含 4 个 Hack，讲解布局相关的小窍门。
- ❑ 第 2 章包含 4 个 Hack，介绍动画处理相关的小窍门。
- ❑ 第 3 章包含 9 个 Hack，涵盖与 View 相关的小窍门。
- ❑ 第 4 章包含两个 Hack，概括除 IDE 以外的可用工具。
- ❑ 第 5 章包含 4 个 Hack，提供适用于 Android 开发的模式示例。
- ❑ 第 6 章包含 7 个 Hack，提供一组适用于 ListView 和 Adapter 类的小窍门。
- ❑ 第 7 章包含两个 Hack，解释如何在应用中使用第三方库。
- ❑ 第 8 章包含两个 Hack，通过一些例子，解释如何用 Java 以外的编程语言为 Android 编写程序。其中一个 Hack 分析如何与 Objective-C 语言交互，另一个 Hack 分析如何与 Scala 语言交互。
- ❑ 第 9 章包含 6 个 Hack，提供一些可以复用的代码片段。
- ❑ 第 10 章包含 3 个 Hack，展示一些使用数据库的高级技巧。
- ❑ 第 11 章包含 4 个 Hack，展示如何令应用程序运行在不同的 Android 版本上。

❑ 第 12 章通过最后 3 个技巧提供如何构建应用的小窍门。

代码规范和下载

本书所有示例代码都以 monospace 字体显示。注释直接写在代码中，对于较长的注释，使用数字编号标识。

本书所有示例代码都可以从出版社网站下载，出版社网址是 www.manning.com/50AndroidHacks ⊖。读者也可以从 Google 公司的 code 项目中下载源代码，下载最新示例代码的方法列在附录中。此外，示例代码托管在 GitHub 中，读者还可以从 https://github.com/Macarse/50AH-code 下载。

如果要运行本书的示例代码，读者需要安装以下工具：

❑ Eclipse

❑ Android SDK

❑ Eclipse Android 插件

如果读者不知道如何安装，我建议首先访问 http://developer.android.com/sdk/installing/index.html，这里提供了配置开发环境的简单步骤。

作者在线支持

Manning 出版社运营的网上论坛为本书提供免费的在线支持。读者可以通过该论坛发表关于本书的意见和建议，也可以提问技术问题，还可以从作者和其他读者处得到帮助和支持。访问和订阅该论坛的方法很简单，读者只需要在浏览器中输入以下网址：www.manning.com/50AnroidHacks。这个网页提供了论坛注册后的注意事项、读者服务以及论坛规则等信息。

Manning 出版社对读者的承诺是：提供一个在读者和读者之间，以及读者和作者之间可以产生良好互动的交流场所。出版社并不能保证作者有

⊖ 截止本书翻译时，该网址已经更改为 http://www.manning.com/sessa/。——译者注

充足的时间与读者互动，因为作者完全是自愿且免费为论坛服务的。我们建议读者多向作者提问一些有挑战的问题，以激发作者答疑的兴趣。

本书出版后，读者可以从出版社的网站访问作者在线支持论坛查看已有的讨论帖。

关于作者

Carlos Sessa 不仅是一位充满激情的全职 Android 开发者，同时，他也是一家移动开发公司的创始人，公司名称为 NASA Trained Monkeys，位于阿根廷的布宜诺斯艾利斯。他的公司专注于为 Android 和 iOS 等移动开发平台提供解决方案。

关于原书封面插图

本书英文版封面插图中的人物是一个樵夫。这幅插图取自 Sylvain Maréchal 所著的四卷《区域服饰习俗概要》，该书于 19 世纪在法国出版。书中每幅插图都经过精心绘制和手工着色。通过大量丰富多彩的图片，Maréchal 向我们生动地展示了如何从文化上区分 200 年前世界上不同的城镇和地区。由于彼此隔绝，不同地区的人们说着不同的方言和语言。无论是在街道还是在乡间，我们可以仅仅根据服饰区分出人们生活的区域、职业以及状况。

此后，着装要求和区域服饰的多样化发生了变化，当时丰富多彩的服饰也逐渐消失。现在已经很难区分不同大陆的居民，就更不用说区分不同的城镇和地区的居民了。或许，我们以文化的多样性为代价换来了丰富多彩的个人生活，当然，换来的是更多样、更快节奏的科技生活。

当计算机书籍千篇一律，读者很难一次就区分出不同的计算机书籍的时候，Manning 出版社赞赏计算机业务部门通过图书封面呈现多样性的创造思维和主动性，本书便以 Maréchal 描绘的两个世纪前不同区域的丰富多彩和活灵活现的生活写照来体现这种多样性。

目　录

第 1 章

活 用 布 局

本章将介绍 Android 布局相关的一些窍门和建议。通过本章，读者不仅可以学习如何从零开始创建特定类型的布局，还可以学到如何改进和优化现有布局。

Hack 1　使用 weight 属性实现视图的居中显示

Android v1.6+

在给开发者做演讲时，当我解释如何通过 XML 文件创建视图的时候，一个开发者问道：“如果我想将按钮居中显示，并且占据其父视图宽度的一半，应该怎么做呢？”起初，我并没有完全理解他的意思，后来他把想要实现的功能画在了黑板上，我才恍然大悟。他想实现的功能如图 1-1 和图 1-2 所示。

看起来很简单是吗？现在开始，请读者用 5 分钟时间实现这个功能。在这个 Hack 里，我们分析如何结合 LinearLayout 的 android:weightSum 属性和 LinearLayout 的子视图的 android:layout_weight 属性来解决这个问题。这听起来似乎很简单，不过我经常在面试中问到这个问题，很少有面试者知道最佳答案。

图 1-1 居中显示按钮，并占据父视图 50% 宽度（竖屏）

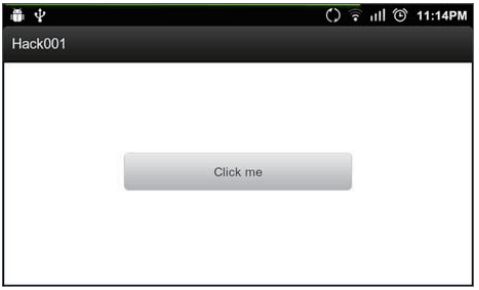

图 1-2 居中显示按钮，并占据父视图 50% 宽度（横屏）

1.1 合用 weightSum 属性和 layout_weight 属性

不同 Android 设备的尺寸往往是不同的。作为开发者，我们需要创建适用于不同尺寸屏幕的 XML 文件。硬编码是不可取的，因此需要其他方法来组织视图。

本节分析如何合用 layout_weight 和 weightSum 这两个属性来填充布局内部的任意剩余空间。android:weightSum（见 1.3 节）的开发文档里的一段描述与我们现在想要实现的功能类似，文档内容

如下：

“定义 weight 总和的最大值。如果未指定该值，以所有子视图的 layout_weight 属性的累加值作为总和的最大值。一个典型的案例是：通过指定子视图的 layout_weight 属性为 0.5，并设置 LinearLayout 的 weightSum 属性为 1.0，实现子视图占据可用宽度的 50%。”

设想一个场景：我们要在盒子里放置其他物体。盒子可用空间的比例就是 weightSum，盒子中每个物体可用空间的比例就是 layout_weight。例如，盒子的 WeightSum 是 1，我们需要往盒子里放置两个物体：物体 *A* 和物体 *B*。物体 *A* 的 layout_weight 为 0.25，物体 *B* 的 layout_weight 为 0.75。那么，物体 *A* 可以占据盒子 25% 的空间，而物体 *B* 可以占据剩下的 75% 的空间。

本章开头所讨论问题的解决方案是与之类似的。我们为父视图指定一个 weightSum，然后指定 Button 的 android:layout_weight 属性为 weightSum 的一半。XML 文件的源码如下所示：

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
android:layout_width="fill_parent"
android:layout_height="fill_parent"
android:background="#FFFFFF"
android:gravity="center"
android:orientation="horizontal"
android:weightSum="1">

<Button
    android:layout_width="0dp"
     android:layout_height="wrap_content"
    android:layout_weight="0.5"
     android:text="Click me"/>
</LinearLayout>
```

❶ 指定 android:weightSum 属性

❷ 指定按钮宽度

❸ 确保按钮占据 50% 可用空间

在 ❶ 中，指定 LinearLayout 的 android:weightSum 属性值为 1，表示其内部所有子视图的 weight 比例总和是 1。LinearLayout 只有唯一一个子视图：Button 控件。在 ❷ 中，指定 Button 的 android:layout_width 属性值为 0dp，因此需要根据 android:weightSum 属性决定 Button 的 width。在 ❸ 中，指定 Button 的 android:layout_weight 属性值为 0.5，最终 Button 将占据 50% 的可用空间。

接下来以宽为200dp，android:weightSum属性值为1的LinearLayout为例分析上述过程。计算Button宽度的公式如下：

```
Button's width + Button's weight * 200 / sum(weight)
```

因为指定Button的宽度为0dp，Button的weight为0.5，sum（weight）等于1，所以结果如下：

```
0 + 0.5 * 200 / 1 = 100
```

1.2 概要

当开发者需要根据比例分配布局可用空间的时候，使用LinearLayout的weight属性是很有必要的，这避免了使用硬编码的方式带来的副作用。如果目标平台是Honeycomb并且使用Fragment，读者会发现绝大多数案例中都是使用weight在布局文件中为Fragment分配空间。深入理解如何使用weight会为读者增添一项重要技能。

1.3 外部链接

http://developer.android.com/reference/android/widget/LinearLayout.html

Hack 2 使用延迟加载以及避免代码重复

Android v1.6+

当创建复杂的布局时，开发者可能会发现添加了很多ViewGroup和View控件。随之而来的问题是View树的层次越来越深，应用程序也越来越慢。优化布局是创建运行速度快，响应灵敏的应用程序的基础。

在这个Hack里，读者会学到如何在XML文件中使用<include />标签来避免代码的重复以及如何使用ViewStub类实现视图的延迟加载。

2.1　使用 <include /> 标签避免代码重复

设想一种情况：我们需要为应用程序中的每个视图都添加一个页脚。为了简化问题，我们假设页脚是一个显示应用程序名的 TextView。通常多个 Activity 会对应多个 XML 文件。难道我们需要把这个 TextView 复制到每个 XML 文件中吗？如果以后需要修改这个 TextView 会出现什么情况？“复制 / 粘贴”的方式固然能够解决这个问题，但并不是高效的方法。解决上述问题的最简单方法是使用 <include /> 标签。接下来分析该标签是如何解决这个难题的。

我们可以通过 <include /> 标签把在其他 XML 文件中定义的布局插入当前布局文件中。在本节的示例代码里，我们将创建一个完整的视图布局文件，在该文件最后的位置，使用 <include /> 标签插入页脚布局。以其中一个 Activity 为例，其 XML 布局文件如下所示：

```
<RelativeLayout
  xmlns:android="http://schemas.android.com/apk/res/android"
  android:layout_width="fill_parent"
  android:layout_height="fill_parent">

  <TextView
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:layout_centerInParent="true"
    android:gravity="center_horizontal"
    android:text="@string/hello"/>

  <include layout="@layout/footer_with_layout_properties"/>

</RelativeLayout/>
```

footer_with_layout_properties 布局文件如下所示：

```
<TextView xmlns:android="http://schemas.android.com/apk/res/android"
  android:layout_width="fill_parent"
  android:layout_height="wrap_content"
  android:layout_alignParentBottom="true"
  android:layout_marginBottom="30dp"
  android:gravity="center_horizontal"
  android:text="@string/footer_text"/>
```

在上述示例代码中，我们使用了 <include /> 标签，并且只需要指定该标签的 layout 属性值。读者可能会想：“这种方式之所以可行是因为 Activity 在 main 布局文件中使用的是 RelativeLayout。如果其中一个 Activity 的布局文件使用的是 LinearLayout 呢？虽

然 android:layout_alignParentBottom="true" 适用于 RelativeLayout，但是并不适用于 LinearLayout。”这个想法是正确的。接下来分析使用 <include /> 标签的第二种方法，在这种方法里，我们直接在 <include /> 标签里使用 android:layout_* 属性。

以下是修改后的 main.xml 文件，其中使用了 <include /> 标签的 android:layout_* 属性，源码如下所示：

```
<RelativeLayout
  xmlns:android="http://schemas.android.com/apk/res/android"
  android:layout_width="fill_parent"
  android:layout_height="fill_parent">

  <TextView
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:layout_centerInParent="true"
    android:gravity="center_horizontal"
    android:text="@string/hello"/>

  <include
    layout="@layout/footer"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:layout_alignParentBottom="true"
    android:layout_marginBottom="30dp"/>

</RelativeLayout/>
```

修改后的页脚布局文件如下：

```
<TextView xmlns:android="http://schemas.android.com/apk/res/android"
  android:layout_width="0dp"
  android:layout_height="0dp"
  android:gravity="center"
  android:text="@string/footer_text"/>
```

在第二种方法中，我们通过 <include /> 标签指定页脚的位置。Android 的缺陷（Issue）跟踪系统中报告过一个缺陷[⊖]，缺陷的标题是：“ <include /> 标签失效了，如果想通过 <include /> 标签的属性覆盖被包含的布局所指定的属性是行不通的。”。这个 issue 描述的问题在一定程度上是正确的，问题出在如果想在 <include /> 标签中覆盖被包含布局所指定的任何 android:layout_* 属性，必须在 <include /> 标签中同时指定 android:layout_width 和

⊖ Issue 原始地址位于 http://code.google.com/p/android/issues/detail?id=2863。——译者注

android:layout_height 这两个属性。

在这个 Hack 中，读者有没有注意到一个小细节？在第二个示例程序中，我们把所有 android:layout_* 属性都移到 <include /> 标签中了，而 footer.xml 文件中的 layout_width 和 layout_height 属性都指定为 0dp。这么做的目的是由 footer.xml 文件的使用者在 <include /> 标签中指定 layout_width 和 layout_height 属性。如果使用者不指定这两个属性，它们的默认值都是 0，我们便看不到页脚。

2.2　通过 ViewStub 实现 View 的延迟加载

设计布局的时候，读者可能想过根据上下文或者用户交互情况显示一个视图。如果想要一个视图只在需要的时候显示，请继续往下阅读，你会尝试使用 ViewStub 这个类。

Android 开发文档中有关于 ViewStub 的介绍（参照 2.4 节），主要内容如下：

“ViewStub 是一种不可视并且大小为 0 的视图，可以延迟到运行时填充（inflate）布局资源。当 ViewStub 设置为可视或者 inflate() 方法被调用后，就会填充布局资源，然后 ViewStub 便会被填充的视图替代。”

既然已经清楚 ViewStub 是什么，接下来看看它能做什么。在下面的示例代码中，我们使用 ViewStub 来延迟加载一个 MapView。假设需要创建一个视图来显示地理位置的详细信息，先看两种可能情况：

❑ 一些场所没有 GPS 信息

❑ 用户可能并不需要地图信息

如果一个场所没有 GPS 信息，开发者不需要在地图上显示标记信息。同样，如果用户不需要地图信息，也就无须加载地图。我们可以把 MapView 放置在 ViewStub 标签中，让用户自己决定是否显示地图信息。

要达到上述目的，需要使用下面的布局：

```
<?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">

  <Button
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="@string/show_map"
    android:onClick="onShowMap"/>

  <ViewStub
        android:id="@+id/map_stub"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
    android:layout="@layout/map"
    android:inflatedId="@+id/map_view"/>
</RelativeLayout>
```

很显然，需要通过 map_stub 这个 ID 从 Activity 中获取 ViewStub。同时，以 android:layout 属性指定需要填充的布局文件。对于本例，需要填充的布局文件是 map.xml 文件，源码如下：

```
<?xml version="1.0" encoding="utf-8"?>
<com.google.android.maps.MapView
  xmlns:android="http://schemas.android.com/apk/res/android"
  android:layout_width="fill_parent"
  android:layout_height="fill_parent"
  android:clickable="true"
  android:apiKey="my_api_key"/>
```

最后一个需要说明的属性是 inflatedId。inflatedId 是调用 ViewStub 的 inflate() 方法或者 setVisibility() 方法时返回的 ID，这个 ID 便是被填充的 View 的 ID。在本例中，我们不需要操作 MapView，只需要调用 setVisibility(View.VISIBLE) 方法即可。如果想获取被填充的视图的引用，inflate() 方法会直接返回该引用，这样避免了再次调用 findViewById() 方法。

Activity 的源码比较简单，如下所示：

```
public class MainActivity extends MapActivity {

  private View mViewStub;

  @Override
  public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);
    mViewStub = findViewById(R.id.map_stub);
  }
```

```
    public void onShowMap(View v) {
      mViewStub.setVisibility(View.VISIBLE);
    }

    ...
}
```

如上述代码所示，只需要改变 ViewStub 的可视性便可控制 map 的显示。

2.3　概要

<include /> 标签是整理布局的有效工具。如果读者使用过 Fragment，会发现它与 <include /> 标签的使用方法几乎是相同的。就像使用 Fragment 一样，完整的视图可以由一系列 <include /> 标签组成。

<include /> 标签提供了合理组织 XML 布局文件的有效方法。如果读者正在创建一个复杂的布局或者布局文件变得很大，那么可以试试创建不同的布局片段，然后通过 <include /> 标签将这些片段组合起来。这样 XML 布局文件就会变得更清晰更易组织。

ViewStub 是实现延迟加载视图的优秀类。无论在什么情况下，只要开发者需要根据上下文选择隐藏或者显示一个视图，都可以用 ViewStub 实现。或许并不会因为一个视图的延迟加载而感觉到性能的明显提升，但是如果视图树的层次很深，便会感觉到性能上的差距了。

2.4　外部链接

http://code.google.com/p/android/issues/detail?id=2863

http://android-developers.blogspot.com.ar/2009/03/android-layout-tricks-3-optimize-with.html

http://developer.android.com/reference/android/view/ViewStub.html

Hack 3 创建定制的 ViewGroup

Android v1.6+

当设计应用程序时，开发者可能需要在不同的 Activity 中显示复杂的视图。假设读者正在开发一款扑克牌游戏，需要创建类似图 3-1 的布局来显示玩家的手牌。应该如何创建这样的布局呢？

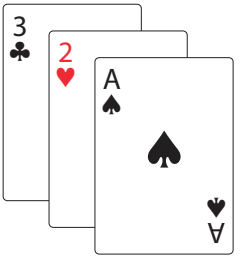

图 3-1 扑克牌游戏中的玩家手牌

或许读者会说，使用 margin 属性便足以实现这种布局。答案是正确的，开发者可以使用 RelativeLayout 布局管理器，然后为其内部 View 控件指定 margin 属性值，这样便可以实现类似上图的的功能，XML 布局文件源码如下：

```
<?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" >

    <View
        android:layout_width="100dp"
        android:layout_height="150dp"
        android:background="#FF0000" />

    <View
        android:layout_width="100dp"
        android:layout_height="150dp"
        android:layout_marginLeft="30dp"
        android:layout_marginTop="20dp"
        android:background="#00FF00" />

    <View
        android:layout_width="100dp"
        android:layout_height="150dp"
        android:layout_marginLeft="60dp"
        android:layout_marginTop="40dp"
        android:background="#0000FF" />

</RelativeLayout>
</FrameLayout>
```

上述布局的显示效果如图 3-2 所示。

在这个 Hack 里，我们分析实现上述功能的另一种方法：创建自定义 ViewGroup。该方法相对于在 XML 文件中手工指定 margin

值有如下优点：

- ❑ 在不同 Activity 中复用该视图时，更易维护。
- ❑ 开发者可以使用自定义属性来定制 ViewGroup 中子视图的位置。
- ❑ 布局文件更简明，更容易理解。
- ❑ 如果需要修改 margin，不必重新手动计算每个子视图的 margin。

图 3-2　使用 Android 默认控件创建的玩家手牌

接下来首先看看 Android 是如何绘制视图的。

3.1　理解 Android 绘制视图的方式

在创建自定义 ViewGroup 前，读者首先需要理解 Android 绘制视图的方式。我不会涉及过多细节，但是需要读者理解 Android 开发文档（见 3.5 节）中的一段话，这段话解释如何绘制一个布局。内容如下：

“绘制布局由两个遍历过程组成：测量过程和布局过程。测量过程由 measure(int, int) 方法完成，该方法从上到下遍历视图树。在递归遍历过程中，每个视图都会向下层传递尺寸和规格。当

measure 方法遍历结束，每个视图都保存了各自的尺寸信息。第二个过程由 layout(int, int, int,int) 方法完成，该方法也是由上而下遍历视图树，在遍历过程中，每个父视图通过测量过程的结果定位所有子视图的位置信息。”

为了理解这个概念，下面分析 ViewGroup 的绘制过程。第一步是测量 ViewGroup 的宽度和高度，在 onMeasure() 方法中完成这步操作。在该方法中，ViewGroup 通过遍历所有子视图计算出它的大小。最后一步操作，在 onLayout() 方法中完成，在该方法中，ViewGroup 利用上一步计算出的测量信息，布局所有子视图。

3.2 创建 CascadeLayout

本节开始为自定义 ViewGroup 编码。读者会看到与图 3-2 一样的结果。将自定义 ViewGroup 命名为 CascadeLayout。CascadeLayout 使用的 XML 布局文件如下所示：

```
<?xml version="1.0" encoding="utf-8"?>
<FrameLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:cascade=
   "http://schemas.android.com/apk/res/com.manning.androidhacks.hack003"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent" >

    <com.manning.androidhacks.hack003.view.CascadeLayout
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"

        cascade:horizontal_spacing="30dp"
        cascade:vertical_spacing="20dp" >

        <View
            android:layout_width="100dp"
            android:layout_height="150dp"
            android:background="#FF0000" />

        <View
            android:layout_width="100dp"
            android:layout_height="150dp"
            android:background="#00FF00" />

        <View
            android:layout_width="100dp"
            android:layout_height="150dp"
            android:background="#0000FF" />
    </com.manning.androidhacks.hack003.view.CascadeLayout>

</FrameLayout>
```

在 XML 中使用自定义属性时指定自定义命名空间

通过 cascade 命名空间，你就可以使用其自定义属性

在 XML 中使用 CascadeLayout 布局，需要指定完全限定类名

现在读者已经理解需要创建什么功能，接下来我们就正式开始了。我们要做的第一件事是定义那些定制的属性。为此，需要在 res/values 目录下创建一个属性文件 attrs.xml，该文件的内容如下：

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <declare-styleable name="CascadeLayout">
        <attr name="horizontal_spacing" format="dimension" />
        <attr name="vertical_spacing" format="dimension" />
    </declare-styleable>
</resources>
```

同时还需要指定水平间距和垂直间距的默认值，以便在未指定这些值时使用。把这些默认值保存在 dimens.xml 文件中，该文件同样位于 res/values 文件夹下。dimens.xml 文件的内容如下：

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <dimen name="cascade_horizontal_spacing">10dp</dimen>
    <dimen name="cascade_vertical_spacing">10dp</dimen>
</resources>
```

理解了 Android 如何绘制 View 之后，读者可能会想到实现一个继承自 ViewGroup 的 CascadeLayout 类，然后在 CascadeLayout 类中重写 ViewGroup 的 onMeasure() 和 onLayout() 方法。接下来的代码有点长，我们分成三部分内容分别予以分析。这三部分是：构造函数、onMeasure() 方法和 onLayout() 方法。先看构造函数，代码如下：

```
public class CascadeLayout extends ViewGroup {

  private int mHorizontalSpacing;
  private int mVerticalSpacing;

  public CascadeLayout(Context context, AttributeSet attrs) {
     super(context, attrs);

    TypedArray a = context.obtainStyledAttributes(attrs,
        R.styleable.CascadeLayout);

  try {
    mHorizontalSpacing = a.getDimensionPixelSize(
    R.styleable.CascadeLayout_horizontal_spacing,
    getResources().getDimensionPixelSize(
        R.dimen.cascade_horizontal_spacing));

 mVerticalSpacing = a.getDimensionPixelSize(
     R.styleable.CascadeLayout_vertical_spacing,
         getResources()
         .getDimensionPixelSize(
```

当通过 XML 文件创建该视图的实例时会调用该构造函数

mHorizontalSpacing 和 mVerticalSpacing 由自定义属性中获取，如果其值未指定，就使用默认值

```
                R.dimen.cascade_vertical_spacing));
    } finally {
      a.recycle();
    }
  }
  ...
```

在编写 onMeasure() 方法之前，先创建自定义 LayoutParams 类，该类用于保存每个子视图的 x、y 轴位置。把 LayoutParams 定义为 CascadeLayout 的内部类，该类的定义如下：

```
public static class LayoutParams extends ViewGroup.LayoutParams {
  int x;
  int y;

  public LayoutParams(Context context, AttributeSet attrs) {
    super(context, attrs);
  }

  public LayoutParams(int w, int h) {
    super(w, h);
  }
}
```

要使用新定义的 CascadeLayout.LayoutParams 类，还需要重写 CascadeLayout 类中的其他一些方法。这些方法是 checkLayoutParams()、generateDefaultLayoutParams()、generateLayoutParams(AttributeSetattrs) 和 generateLayoutParams(ViewGroup.LayoutParams p)。这些方法的代码在不同 ViewGroup 之间往往是相同的。如果读者对这些方法的具体内容感兴趣，可以在示例代码中找到这些内容。

下一步是对 onMeasure() 方法编码。这是该类的核心部分。代码如下所示：

```
@Override
protected void onMeasure(int widthMeasureSpec, int heightMeasureSpec) {
  int width = 0;
  int height = getPaddingTop();

  final int count = getChildCount();
  for (int i = 0; i < count; i++) {
    View child = getChildAt(i);

    measureChild(child, widthMeasureSpec, heightMeasureSpec);

    LayoutParams lp = (LayoutParams) child.getLayoutParams();
    width = getPaddingLeft() + mHorizontalSpacing * i;
```

使用宽和高计算布局的最终大小以及子视图的 x 与 y 轴位置

令每个子视图测量自身

```
            lp.x = width;
            lp.y = height;

            width += child.getMeasuredWidth();
            height += mVerticalSpacing;
        }

        width += getPaddingRight();
        height += getChildAt(getChildCount() - 1).getMeasuredHeight()
            + getPaddingBottom();

        setMeasuredDimension(resolveSize(width, widthMeasureSpec),
            resolveSize(height, heightMeasureSpec));
    }
```

在 LayoutParams 中保存每个子视图的 x 和 y 坐标

使用计算所得的宽和高设置整个布局的测量尺寸

最后一步是对 onLayout() 方法编码，代码如下所示：

```
@Override
protected void onLayout(boolean changed, int l, int t, int r, int b) {

  final int count = getChildCount();
  for (int i = 0; i < count; i++) {
    View child = getChildAt(i);
    LayoutParams lp = (LayoutParams) child.getLayoutParams();

    child.layout(lp.x, lp.y, lp.x + child.getMeasuredWidth(), lp.y
        + child.getMeasuredHeight());
  }
}
```

由以上代码可知，代码逻辑是非常简单的。该方法以 onMeasure() 方法计算出的值为参数循环调用子 View 的 layout() 方法。

3.3　为子视图添加自定义属性

在最后一节，学习如何为子视图添加自定义属性。作为示例，下面将添加为特定子视图重写（override）垂直间距的方法。读者可以看到图 3-3 所示结果。

第一步是向 attrs.xml 文件中添加一个新的属性，代码如下：

```
<declare-styleable name="CascadeLayout_LayoutParams">
    <attr name="layout_vertical_spacing" format="dimension" />
</declare-styleable>
```

因为属性名的前缀是 layout_，没有包含一个视图属性，因此该属性会被添加到 LayoutParams 的属性表中。正如 CascadeLayout 类，在 LayoutParams 类的构造函数中读取这个新属性。源码如下：

```
  public LayoutParams(Context context, AttributeSet attrs) {
    super(context, attrs);

    TypedArray a = context.obtainStyledAttributes(attrs,
        R.styleable.CascadeLayout_LayoutParams);
    try {
      verticalSpacing = a.getDimensionPixelSize(
        R.styleable.CascadeLayout_LayoutParams_layout_vertical_spacing,
        -1);
    } finally {
      a.recycle();
    }
  }
```

图 3-3 使第一个子视图具有不同的垂直间距

verticalSpacing 是一个公共字段（field 成员变量），我们会在 CascadeLayout 类的 onMeasure() 方法中使用到该字段。如果子视图的 LayoutParams 包含 verticalSpacing，就可以使用它。源码如下：

```
verticalSpacing = mVerticalSpacing;

...

LayoutParams lp = (LayoutParams) child.getLayoutParams();

if (lp.verticalSpacing >= 0) {
  verticalSpacing = lp.verticalSpacing;
}

...

width += child.getMeasuredWidth();
height += verticalSpacing;
```

3.4　概要

使用自定义 View 和 ViewGroup 组织应用程序布局是一个好方法。定制组件的同时允许开发者提供自定义行为和功能。以后，开发者在需要创建复杂布局的时候，首先应该考虑使用自定义 ViewGroup 是不是更合适。虽然在开始时，这样做会增加一定的工作量，但是，这是值得的。

3.5　外部链接

http://developer.android.com/guide/topics/ui/how-android-draws.html

http://developer.android.com/reference/android/view/ViewGroup.html

http://developer.android.com/reference/android/view/ViewGroup.LayoutParams.html

Hack 4　偏好设置使用技巧

Android v1.6+

Android SDK 中提供了 Preference（偏好设置）框架，我很喜欢这个功能。相对于 iOS SDK，使用这个功能创建偏好设置界面更容易一些。开发者只需要编辑一个简单的 XML 文件，就能开发出一个易用的偏好设置界面。

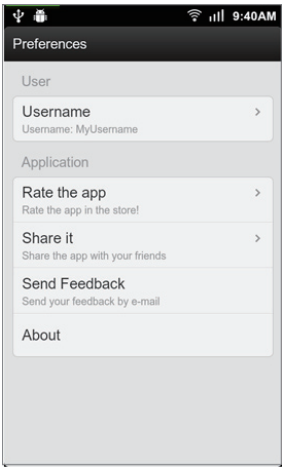

图 4-1　Preference 界面

尽管 Android 提供了很多设置控件供开发者使用，但是很多时候，还是需要开发者自己定制界面。在这个 Hack 提供的示例代码中，读者会学到如何定制设置控件。最终的 Preference 界面如图 4-1 所示：

首先看看下面的 XML 文件，源码如下：

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

    android:key="pref_first_preferencescreen_key"
    android:title="Preferences">

    <PreferenceCategory
        android:title="User">

        <EditTextPreference
            android:key="pref_username"
            android:summary="Username"
            android:title="Username"/>

    </PreferenceCategory>

    <PreferenceCategory
        android:title="Application">

        <Preference
            android:key="pref_rate"
            android:summary="Rate the app in the store!"
            android:title="Rate the app"/>

    <Preference
        android:key="pref_share"
        android:summary="Share the app with your friends"
        android:title="Share it"/>

    <com.manning.androidhacks.hack004.preference.EmailDialog
        android:dialogIcon="@drawable/ic_launcher"
        android:dialogTitle="Send Feedback"
        android:dialogMessage="Do you want to send an email?"
        android:key="pref_sendemail_key"
        android:negativeButtonText="Cancel"
        android:positiveButtonText="OK"
        android:summary="Send your feedback by e-mail"
        android:title="Send Feedback"/>

    <com.manning.androidhacks.hack004.preference.AboutDialog
        android:dialogIcon="@drawable/ic_launcher"
        android:dialogTitle="About"
        android:key="pref_about_key"
        android:negativeButtonText="@null"
        android:title="About"/>

    </PreferenceCategory>

</PreferenceScreen>
```

Figure 4.1 Prefe

为 Preference 指定 android:key 属性是一个很好的习惯，通过该属性可以检索 Preference 对象

使用 PreferenceCategory 可以为 Preference 分组，并指定分组名

选择用户名，需要使用 EditTextPreference 控件。这里指定了 summary 属性的默认值，当用户指定用户名后，该默认值会被替换

为那些需要发送 Intent 的选项指定 Preference

可以创建自定义 Preference 继承已有的控件

上面创建的XML文件负责UI的显示。接下来添加业务逻辑代码，为此，需要创建一个Activity，这个Activity不能继承android.app.Activity，而是需要继承android.preference.PreferenceActivity。源码如下所示：

```
public class MainActivity extends PreferenceActivity implements
    OnSharedPreferenceChangeListener {
```

```
  @Override
  public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    addPreferencesFromResource(R.xml.prefs);

    ...

    Preference ratePref = findPreference("pref_rate");
    Uri uri = Uri.parse("market://details?id=" + getPackageName());
    Intent goToMarket = new Intent(Intent.ACTION_VIEW, uri);
    ratePref.setIntent(goToMarket);
  }

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

    getPreferenceScreen().getSharedPreferences()
        .registerOnSharedPreferenceChangeListener(this);
  }

  @Override
  protected void onPause() {
    getPreferenceScreen().getSharedPreferences()
        .unregisterOnSharedPreferenceChangeListener(this);
   }

  @Override
  public void onSharedPreferenceChanged(
      SharedPreferences sharedPreferences, String key) {

    if (key.equals("pref_username")) {
       updateUserText();
    }
  }

  private void updateUserText() {
    EditTextPreference pref;
    pref = (EditTextPreference) findPreference("pref_username");
    String user = pref.getText();

    if (user == null) {
      user = "?";
    }

    pref.setSummary(String.format("Username: %s", user));
  }
}
```

这里并不调用 setContentView() 方法，而是调用 addPreferencesFromResource() 方法，传入该方法的参数是之前创建的 XML 文件

在 onCreate() 方法中，可以获取 Preference，并为其设置 Intent，在本例中，ratePref 使用 Intent.ACTION_VIEW

注册 Preference 变化通知监听器

注销 Preference 变化通知监听器

当用户名 Preference 改变时，需要更新该 Preference 的 summary

要更新 summary，需要获取该 Preference，然后调用 EditTextPreference 的 getText() 方法

上述代码展示如何创建自定义的 Preference，这个过程与创建自定义视图相似。为了进一步理解 Preference，接下来看看 EmailDialog，源码如下：

```
public class EmailDialog extends DialogPreference {
   Context mContext;

  public EmailDialog(Context context) {
    this(context, null);
  }
```

自定义类需要继承自某个已有的 Preference 控件。在本例中，我们继承自 DialogPreference

```
  public EmailDialog(Context context, AttributeSet attrs) {
    this(context, attrs, 0);
  }

  public EmailDialog(Context context, AttributeSet attrs,
    int defStyle) {

    super(context, attrs, defStyle);
    mContext = context;
  }

  @Override

  public void onClick(DialogInterface dialog, int which) {
     super.onClick(dialog, which);

    if (DialogInterface.BUTTON_POSITIVE == which) {
      LaunchEmailUtil.launchEmailToIntent(mContext);
    }
  }
}
```

构造方法与之前创建的继承自 View 类的自定义视图相同

重写 onClick() 方法。如果用户点击 OK 按钮，我们使用辅助类发送启动 email 的 Intent

4.1 概要

尽管偏好设置框架允许开发者添加自定义功能，但是，有一点必须注意：偏好设置框架的目的是创建简单的偏好设置界面。如果开发者想添加更多复杂的 UI 控件或者逻辑，我建议单独创建一个 Activity 并使用 Dialog 的主题，然后从偏好设置控件上启动它。

4.2 外部链接

http://developer.android.com/reference/android/preference/PreferenceActivity.html

第 2 章

添加悦目的动画效果

本章学习动画相关内容。从本章的示例代码中，读者可以学到如何使用各种 API 为应用程序控件添加动画效果。

Hack 5　使用 TextSwitcher 和 ImageSwitcher 实现平滑过渡

Android v1.6+

假设你需要在 TextView 或者 ImageView 中循环浏览信息。举例如下：

- ❑ 通过向左和向右的导航按钮浏览日期列表
- ❑ 在日期选择控件中改变日期
- ❑ 倒计时时钟
- ❑ 新闻纲要

更改视图中的内容是多数应用程序的基本功能，但是这未必是单调无趣的。如果使用默认的 TextView 控件，你会发现当切换其内容时，并没有良好的视觉体验。因此如果有一种方法可以为内容切换添加动画效果就太好了。为了使过渡过程的视觉效果更自然，

Android 提供了 TextSwitcher 和 ImageSwitcher 这两个类分别替代 TextView 与 ImageView。

TextView 和 TextSwitcher 的机制是类似的。回到上文所述的例子，假设用户正在浏览一个日期列表，每当点击按钮时，就需要更改 TextView 内显示的日期。使用 mTextView.setText("something") 方法更新 TextView 中的内容。具体代码如下所示：

```
private TextView mTextView;

@Override
public void onCreate(Bundle savedInstanceState) {
  super.onCreate(savedInstanceState);
  mTextView = (TextView) findViewById(R.id.your_textview);

  ...

  mTextView.setText("something");
}
```

读者可能会注意到，TextView 的内容是立刻改变的。如果需要添加动画效果以避免这种生硬的内容切换方式，就需要使用 TextSwitcher。TextSwitcher 用于为文本标签添加动画效果。当调用 TextSwitcher 的相关方法时，TextSwitcher 便会以动画的方式换出当前文本，并换入新的文本。要获得这种让用户愉悦的过渡效果，只需要以下几个简单步骤：

1）通过 findViewById() 方法获取 TextSwitcher 对象的引用 switcher，当然也可以直接在代码中构造该对象。

2）通过 switcher.setFactory() 方法指定 TextSwitcher 的 ViewFactory。

3）通过 switcher.setInAnimation() 方法设置换入动画效果。

4）通过 switcher.setOutAnimation() 方法设置换出动画效果。

TextSwitcher 的工作原理是：首先通过 ViewFactory 创建两个用于在 TextSwitcher 中切换的视图，每当调用 setText() 方法时，TextSwitcher 首先移除当前视图并显示 setOutAnimation() 方法设置的动画，然后并将另一个视图切换进来，并显示 setInAnimation() 方法设置的动画。使用方法如下所示：

```
private TextSwitcher mTextSwitcher;
```

```
@Override
public void onCreate(Bundle savedInstanceState) {
  super.onCreate(savedInstanceState);
  setContentView(R.layout.main);
  Animation in = AnimationUtils.loadAnimation(this,
    android.R.anim.fade_in);
  Animation out = AnimationUtils.loadAnimation(this,
    android.R.anim.fade_out);

  mTextSwitcher = (TextSwitcher) findViewById(R.id.your_textview);
  mTextSwitcher.setFactory(new ViewFactory() {

    @Override
    public View makeView() {
      TextView t = new TextView(YourActivity.this);
      t.setGravity(Gravity.CENTER);
      return t;
    }
  });

  mTextSwitcher.setInAnimation(in);
  mTextSwitcher.setOutAnimation(out);
}
```

就这么简单，寥寥几行代码既改变了文本内容，又增加了很酷的动画效果。通过淡入淡出的过渡效果让新的文本内容替换了旧的文本内容。在本例中，使用的是 android.R.anim.fade_in，这是一个淡入效果，也可以使用其他效果，步骤是一样的，因此读者可以尝试使用自定义动画效果或者 android.R.anim 中定义的任意动画效果。ImageSwitcher 与 TextSwitcher 的原理是一样的，只不过前者切换的是图片，后者切换的是文本。

5.1　概要

TextSwitcher 和 ImageSwitcher 提供了添加过渡动画的简单方法。提供这两个控件的目的是为了让过渡更自然，因此不要滥用它们，谁也不想把自己的应用程序点缀得像一颗圣诞树那样不自然。

5.2　外部链接

http://developer.android.com/reference/android/widget/TextSwitcher.html

http://developer.android.com/guide/topics/graphics/view-animation.htm

Hack 6 为 ViewGroup 的子视图添加悦目的动画效果

Android v1.6+

默认情况下，添加到 ViewGroup 中的子视图是直接显示出来的。有一个比较简单的方法可以为这个过程增加动画效果。在这个 Hack 里，我们学习如何为 ViewGroup 的子视图添加动画效果。下面展示如何通过寥寥几行代码便能为应用程序增添悦目的动画效果。

Android 提供了 LayoutAnimationController 类，用于为布局或者 ViewGroup 的子视图添加动画效果。有一点需要强调，开发者不可以为每个子视图分别指定不同的动画效果，但是 LayoutAnimationController 可以决定各个子视图显示动画效果的时间。

通过例子来理解如何使用 LayoutAnimationController 是一个好方法。在下面的例子中，我们将结合透明度渐变动画（alpha animation）和位移动画（translate animation）演示如何给 ListView 的子视图添加动画效果。使用 LayoutAnimationController 的方法有两种：直接在代码中使用或者在 XML 文件中配置。我会演示如何在代码中使用 LayoutAnimationController，感兴趣的读者可以自己试试在 XML 文件中配置的方法。添加动画效果的源码如下所示：

```
mListView = (ListView) findViewById(R.id.my_listview_id);

AnimationSet set = new AnimationSet(true);

Animation animation = new AlphaAnimation(0.0f, 1.0f);
animation.setDuration(50);
set.addAnimation(animation);

animation = new TranslateAnimation(Animation.RELATIVE_TO_SELF, 0.0f,
  Animation.RELATIVE_TO_SELF, 0.0f, Animation.RELATIVE_TO_SELF,
  -1.0f, Animation.RELATIVE_TO_SELF, 0.0f);
animation.setDuration(100);
set.addAnimation(animation);

LayoutAnimationController controller = new LayoutAnimationController(
  set, 0.5f);

mListView.setLayoutAnimation(controller);
```

❶ 获取 ListView 的引用

❷ 创建默认动画集合对象

❸ 创建透明度渐变动画

❹ 创建位移动画

❺ 创建 LayoutAnimationController 对象并设置子视图动画效果持续时间

❻ 将 LayoutAnimationController 对象设置到 ListView 中

首先如❶中所示，需要获得 ListView 的引用。然后如❷

中所示，由于需要添加多个动画效果，因此创建一个动画集合 AnimationSet。传入 AnimationSet 构造函数的布尔型参数决定每个动画是不是使用同一个 interpolator㊀，在本例中，使用默认 interpolator。之后如❸和❹中所示，依次创建透明度渐变动画和过渡动画，并将其添加到动画集合中。再之后，如❺中所示，以动画集合和动画持续时间为参数创建 LayoutAnimationController 对象。最后，如❻中所示，把这个 LayoutAnimationController 对象应用到 ListView 中。

框架层提供的多数动画控件与 TranslateAnimation 相似，下面分析 TranslateAnimation 的源代码，其构造函数定义如下：

```
public TranslateAnimation(int fromXType, float fromXValue, int toXType,
  float toXValue, int fromYType, float fromYValue, int toYType,
  float toYValue) {
```

该接口很简单，需要传入 x 轴和 y 轴的起始与结束坐标。Android 提供了三个选项，用于指定计算坐标的参照：

❑ Animation.ABSOLUTE

❑ Animation.RELATIVE_TO_SELF

❑ Animation.RELATIVE_TO_PARENT

回到上面的例子，我们可以用下面的词汇表示每个子视图的位置：

❑ Initial X：父视图指定的起始 X 坐标

❑ Initial Y：父视图指定的起始 Y 坐标

❑ Final X：父视图指定的结束 X 坐标

❑ Final Y：父视图指定的结束 Y 坐标

上述示例程序的最终效果是：每个子视图㊁沿 Y 轴方向依次滑落到各自的位置上，同时由于子视图之间有一定延迟，所以整体看起来像瀑布一样。

㊀ Interpolator 的概念请参照 http://developer.android.com/reference/android/view/animation/Interpolator.html。——译者注

㊁ 本例中的子视图即 ListView 中的每个 Item。——译者注

6.1　概要

为 ViewGroup 添加动画效果是比较简单的。动画会令应用程序看起来更专业更精练。这个 Hack 只涉及一小部分内容，开发者可以完善本章的示例程序。比如，开发者可以将默认 interpolator 替换为 BounceInterpolator，这样，当子视图滑落到预定位置时，会出现弹跳效果。此外，还可以控制子视图动画的显示顺序。

发挥想象力去创建一些炫酷的效果，但是请记住，过犹不及——开发者应该避免使用太多的动画效果。

6.2　外部链接

http://developer.android.com/reference/android/view/animation/LayoutAnimationController.html

Hack 7　在 Canvas 上显示动画

Android v1.6+

如果读者想为自定义的 UI 控件添加动画效果，会发现动画相关的 API 是很有限的。那有没有 API 可以直接向屏幕绘图呢？答案是肯定的。Android 提供了 Canvas 类满足这一需求。

在这个 Hack 里，我以方块在屏幕上弹跳为例分析如何使用 Canvas 类在屏幕上绘制图形，并为其添加动画效果。应用程序最终效果如图 7-1 所示：

编写应用程序前，需要读者首先确认是否已经理解了 Canvas 类的基本概念，下面的解释摘自开发文档（见 7.2 节），内容如下：

“可以把 Canvas 视为 Surface 的替身或者接口，图形便是绘制在 Surface 上的。Canvas 封装了所有绘图调用。通过 Canvas，绘制到 Surface 上的内容首先存储到与之关联的 Bitmap 中，该 Bitmap 最终会呈现到窗口上。”

基于上述定义，Canvas 类封装了所有绘图调用，我们可以创

建一个 View（视图），重写其 onDraw() 方法，在该方法中便可以绘制基本的图形单元。

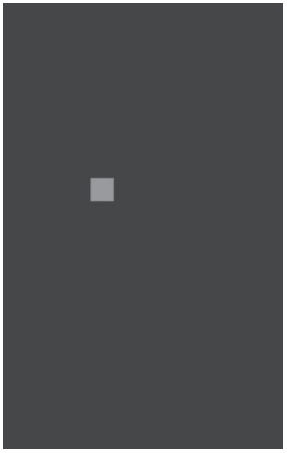

图 7-1　在屏幕内弹跳的方块

为了加深理解，创建 DrawView 类，该类负责在屏幕上绘制一个方块，并不断更新方块的位置，除此之外，屏幕上没有其他控件了。以 DrawView 作为 Activity 的内容视图（Content View），Activity 的代码如下：

```
public class MainActivity extends Activity {

  private DrawView mDrawView;

  @Override
  public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);

    Display display = getWindowManager().getDefaultDisplay();
    mDrawView = new DrawView(this);
    mDrawView.height = display.getHeight();
    mDrawView.width = display.getWidth();

    setContentView(mDrawView);
   }
}
```

❶ 获取屏幕的宽和高

❷ DrawView 占据所有可用空间

在 ❶ 中，通过 WindowManager 获取屏幕的宽和高，这些值用于限制 DrawView 的绘图范围。然后，在 ❷ 中，将 DrawView 设置为 Activity 的内容视图，表示 DrawView 会占据界面的所有可用空间。

接下来看看 DrawView 类的实现代码，源码如下所示：

```
public class DrawView extends View {
  private Rectangle mRectangle;
  public int width;
  public int height;

  public DrawView(Context context) {
    super(context);

    mRectangle = new Rectangle(context, this);
    mRectangle.setARGB(255, 255, 0, 0);
    mRectangle.setSpeedX(3);
    mRectangle.setSpeedY(3);
  }

  @Override
  protected void onDraw(Canvas canvas) {
    mRectangle.move();
    mRectangle.onDraw(canvas);

    invalidate();
   }
}
```

❶ 创建方块对象

❷ 变换方块位置

❸ 将方块绘制到 Canvas 上

❹ 重绘 View

首先，在❶中，创建一个 Rectangle 实例，代表一个方块。Rectangle 类内部实现了将自身绘制到 Canvas 上的逻辑，并且已经包含了正确变换其位置的代码逻辑。当调用 onDraw() 方法时，通过❷，方块的位置就会改变，然后通过❸绘制到 Canvas 上。在❹中，invalidate() 方法本身就是一个小技巧，这个方法强制重绘视图。把这个方法放在 onDraw() 的目的是为了在 View 绘制完自身后，可以立即重新调用 onDraw() 方法。换句话说，通过循环调用 Rectangle 的 move() 和 onDraw() 方法实现一个动画效果。

7.1 概要

在 onDraw() 方法中通过调用 invalidate() 方法变换视图的位置是实现自定义动画的简单方法。如果读者打算开发一款小游戏，使用这个小技巧处理游戏的主循环是一个简单的方法。

7.2 外部链接

http://developer.android.com/reference/android/graphics/Canvas.html

http://developer.android.com/guide/topics/graphics/2d-graphics.html

Hack 8　附加 Ken Burns 特效的幻灯片

Android v1.6+

我们公司早期开发过一款产品叫 FeedTV，这款产品的创意是变革 RSS 源（RSSfeed）的阅读方式。为了避免给用户显示一个冗长的列表，FeedTV 提供类似于相框程序的界面来展示 RSS 源的标题和主图。FeedTV 运行于 iPad 的效果如图 8-1 所示。

图 8-1　FeedTV 运行于 iPad 的效果

为了让界面看起来更酷，我们不显示静态图片，而是通过分析图片的大小和宽高比，加入所谓的 Ken Burns 特效。Ken Burns 特效只不过是视频产品中使用的一种平移和缩放静态图片的特效。理解 Ken Burns 特效最直观的方式是观看一段视频，图 8-2 也可以让读者对其有个初步了解。

图 8-2　维基百科上 Ken Burns 特效的例子

在这个 Hack 里，我会向读者展示如果在图像幻灯片（Image Slideshow）里模拟 Ken Burns 特效。要实现这个功能，需要使用 Jake Wharton 开发的 Nine Old Androids 库。这个库可以让开发者在旧版本上使用 Android 3.0 的动画 API。

要创建 Ken Burns 特效，需要预设一些动画。这些动画将随机应用到 ImageView，当一个动画显示完毕，就开始显示另一个动画和图片。主布局使用 FrameLayout，把 ImageView 置于该布局中。

创建布局的代码如下所示：

```
@Override
public void onCreate(Bundle savedInstanceState) {
  super.onCreate(savedInstanceState);

  mContainer = new FrameLayout(this);
  mContainer.setLayoutParams(new LayoutParams(
     LayoutParams.FILL_PARENT, LayoutParams.FILL_PARENT));

  mView = createNewView();
  mContainer.addView(mView);

  setContentView(mContainer);
}

private ImageView createNewView() {
  ImageView ret = new ImageView(this);
  ret.setLayoutParams(new LayoutParams(LayoutParams.FILL_PARENT,
  LayoutParams.FILL_PARENT));
  ret.setScaleType(ScaleType.FIT_XY);
  ret.setImageResource(PHOTOS[mIndex]);
  mIndex = (mIndex + 1 < PHOTOS.length) ? mIndex + 1 : 0;

  return ret;
}
```

创建布局容器

创建 ImageView，并将其添加到布局容器中

设置要显示的图片，并设置下一个要显示的图片的索引

到目前为止，一切顺利。接下来，首先通过 createNewView() 方法创建 ImageView 对象，以此来引用下一个要显示的图片。然后创建 nextAnimation() 方法，这个方法负责设置动画并启动动画。代码如下所示：

```
private void nextAnimation() {
  AnimatorSet anim = new AnimatorSet();
  final int index = mRandom.nextInt(ANIM_COUNT);

  switch (index) {
    case 0:
      anim.playTogether(
        ObjectAnimator.ofFloat(mView, "scaleX", 1.5f, 1f),
        ObjectAnimator.ofFloat(mView, "scaleY", 1.5f, 1f));
      break;

    ...
    case 3:
    default:
      AnimatorProxy.wrap(mView).setScaleX(1.5f);
      AnimatorProxy.wrap(mView).setScaleY(1.5f);
      anim.playTogether(ObjectAnimator.ofFloat(mView,
        "translationX", 0f, 40f));
    break;
  }

  anim.setDuration(3000);
  anim.addListener(this);
  anim.start();
}
```

随机选择动画

缩放动画

1 位移动画

2 设置动画持续时间，设置动画监听器为当前 Activity，启动动画

在❶中，AnimatorProxy 是定义于 Nine Old Androids 库中的类，用于修改 View 的属性。这个新的动画框架的基础是：视图的属性随着时间的推移可以改变。之所以使用 AnimatorProxy，是因为在 Android 3.0 以下版本，有些属性没有 getters/setters 方法。

余下的代码便是在动画结束的时候调用 nextAnimation() 方法。记住，在❷中将当前 Activity 设置为动画监听器。接下来，我们看看需要重写哪些方法。源码如下所示：

```
@Override
public void onAnimationEnd(Animator animator) {
  mContainer.removeView(mView);
  mView = createNewView();
  mContainer.addView(mView);

  nextAnimation();
 }
```

从布局容器中移除之前的 View，并添加新的 View

开始显示下一个动画

仅此而已，我们便为每幅图片都添加了 Ken Burns 特效。读者可以尝试修改两个地方来改进上面的例子：当切换视图时，添加透明度渐变动画（alpha animation）并且添加一个 AnimationSet 同时显示平移和缩放动画。读者可以从 Nine Old Androids 库的示例代码中获得更多启发。

8.1　概要

新的动画 API 通常可以比旧的 API 实现更多潜在功能。以下是新 API 一些改进的简短列表：

- ❑ 旧版本的 API 只支持视图对象的动画效果。
- ❑ 旧版本的 API 仅限于移动、旋转、缩放、渐变等效果。
- ❑ 旧版本的 API 只改变视图移动时的视觉效果，并未改变其真实位置属性。

事实上，像 Nine Old Androids 这样的库存在就意味着我们有理由在新的 API 上尝试这些特效。

8.2　外部链接

www.nasatrainedmonkeys.com/portfolio/feedtv/

https://github.com/JakeWharton/NineOldAndroids

http://en.wikipedia.org/wiki/Ken_Burns_effect

http://android-developers.blogspot.com.ar/2011/02/animation-in-honeycomb.html

http://android-developers.blogspot.com.ar/2011/05/introducing-viewpropertyanimator.html

第 3 章

使用视图的技巧和窍门

本章介绍使用视图的各种技巧。这些技巧展示了如果通过自定义或者微调 UI 控件来实现特定的功能。

Hack 9　避免在 EditText 中验证日期

Android v1.6+

开发者都知道验证表单[⊖]里的数据是令人厌烦而且容易出错的。我曾经开发过一个应用程序，这个应用程序使用了大量表单并且有多个日期输入框。我不想验证每个日期字段，因此想出一个简洁的方法避免这个验证过程。该方法的思路是：开发一个外观看起来与 EditText 相同的 Button，点击该 Button 后，会显示一个 DatePicker 控件。

要实现上面的思路，需要把 Button 控件的默认背景修改为 EditText 的背景。方法很简单，只需要修改 XML 文件即可，源码如下所示：

```
<Button android:id="@+id/details_date"
    android:layout_width="wrap_content"
```

⊖　英文意思是 form，通常表示界面上的各种输入框——译者注

```
    android:layout_height="wrap_content"
    android:gravity="center_vertical"
    android:background="@android:drawable/edit_text" />
```

注意到没有？我们并没有使用自定义图片指定背景，而是使用了 @android:drawable。在应用程序中使用 Android 内置资源有利有弊。有利的一面是可以使应用程序更好地适应不同设备，不利的一面是应用程序在不同设备上的用户体验有可能不同。一些开发者喜欢使用自定义资源、图片和主题保持应用程序的外观一致。

如果读者在不同设备上测试应用程序，会发现 UI 控件的样式有可能是不一样的，使用 Android 内置资源，会让应用程序选择 Android 的样式。

创建 Button 后，还需要为其设置点击事件监听器。源码如下所示：

```
mDate = (Button) findViewById(R.id.details_date);
mDate.setOnClickListener(new OnClickListener() {

    @Override
    public void onClick(View v) {
        showDialog(DATE_DIALOG_ID);
    }
});
```

在余下的代码逻辑中，会弹出一个 DatePicker 控件，当用户选择一个日期后，该日期的文本会被设置到 Button 上显示出来[⊖]。

9.1 概要

细心的读者内心可能会有这样的疑问：为什么不直接为 EditText 设置一个点击监听器，而非要使用 Button 呢？答案是：使用 Button 更安全，因为用户无法修改 Button 的文本内容。如果使用 EditText，并且只设置了点击监听器，用户可以通过光标获取该控件的焦点，这样便可以绕过 DatePicker 控件直接修改 EditText 的文本内容。

⊖ 截止本书翻译时，Hack009 示例代码中并没有将日期文本设置到 Button 上的逻辑，因此需要在 onDateSet 回调方法中添加这部分逻辑才能正常显示选择的日期。——译者注

读者也可以用 TextWatcher 验证 EditText 中的用户输入信息，但是，这种方法是很烦琐且耗时的。使用本 Hack 提供的方法可以简化代码并避免输入错误。最后，请记住，在应用程序中使用 Android 内置资源是一个借用设备样式（Style）的好方法。

9.2　外部链接

http://developer.android.com/reference/android/widget/DatePicker.html

http://developer.android.com/reference/android/widget/EditText.html

Hack 10　格式化 TextView 的文本

Android v1.6+

假设要在 Twitter 应用程序上显示如图 10-1 所示的推文（tweet）。请注意这条推文里文本样式的不同。你或许会认为这是 Twitter 创建的自定义视图，但是这个 UI 控件是用 TextView 实现的。

图 10-1　Twitter 示例

很多情况下，开发者需要添加一些特殊样式的文本来突出重点内容或者为链接提供视觉反馈，以此提高应用程序的用户友好度。还有其他例子可以说明文本样式的用途，举例如下：

- ❑ 为电话号码显示链接。
- ❑ 为文本的不同部分添加不同的背景色。

在这个 Hack 里，我会分析如果使用 TextView 添加不同样式的文本和链接。

首先，我们添加超链接。可以通过 Html.fromHtml() 方法设置

TextView 的文本内容。文本内容需要简单处理：在 TextView 的文本内容中嵌入 HTML 代码。代码如下所示：

```
mTextView1 = (TextView) findViewById(R.id.my_text_view_html);
String text =
     "Visit <a href=\"http://manning.com/\">Manning home page</a>";
mTextView1.setText(Html.fromHtml(text));
mTextView1.setMovementMethod(LinkMovementMethod.getInstance());
```

在 TextView 中使用 HTML 代码设置样式比较容易理解，那么 Html.fromHtml() 方法做了些什么？它的返回值又是什么？该方法将 HTML 转化为一个 Spanned 对象，并以此为参数调用 TextView 的 setText() 方法。

现在，我们尝试另外一种方法。我们不使用 HTML 格式化文本内容，而是使用 SpannableString ⊖类创建一个 Spanned 对象。源码如下所示：

```
mTextView2 = (TextView) findViewById(R.id.my_text_view_spannable);
Spannable sText = new SpannableString(mTextView2.getText());
sText.setSpan(new BackgroundColorSpan(Color.RED), 1, 4, 0);
sText.setSpan(new ForegroundColorSpan(Color.BLUE), 5, 9, 0);
mTextView2.setText(sText);
```

上述两段代码的运行结果如图 10-2 所示。用法比较简单：我们通过文本中字符的索引指定不同的跨度（span），不同的跨度将文本内容分成不同的部分，然后通过 SpannableString 就可以为不同部分指定不同的样式。

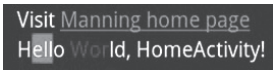

图 10-2 使用 spannable 样式的 TextView

10.1 概要

TextView 是 Android 提供的一个简单却功能强大的 UI 控件。读者可以在应用程序中通过多种方法使用不同样式的文本。尽管

⊖ SpannableString 是 Spanned 的子类，参见 http://developer.android.com/reference/android/text/Spanned.html——译者注

TextView 并不支持所有 HTML 标签，但是用于格式化文本内容已经足够了，放心使用它吧。

10.2　外部链接

http://developer.android.com/reference/android/widget/TextView.html

Hack 11　为文本添加发亮的效果

Android v1.6+

假如读者需要开发一个显示时间的应用程序，还记得那些显示高亮绿灯的数字时钟吗？在这个 Hack 里，我向读者展示如果通过微调 Android 的 TextView 控件来生成这样的效果。最终效果如图 11-1 所示。

图 11-1　数字时钟 demo

首先需要创建继承自 TextView 的 LedTextView 类，通过这个类设置一种特殊的字体，让视图的文本内容看起来与 LED（发光二极管）中显示的一样。源码如下所示：

```
public class LedTextView extends TextView {

  public LedTextView(Context context, AttributeSet attrs) {
    super(context, attrs);

    AssetManager assets = context.getAssets();
    final Typeface font = Typeface.createFromAsset(assets,
        FONT_DIGITAL_7);
    setTypeface(font);
  }
}
```

❶ 设置字体

当创建对象时，我们从 assets 文件夹中获取字体信息，然后在 ❶ 中设置该字体。这样便拥有一个可以使用自定义字体显示文本内容的 UI 控件。接下来需要考虑的事情便是如何在文本上绘制数字。如果读者仔细观察图 11-1，会发现可以通过两个 TextView 实现。第一个 TextView 是背景中显示 88:88:88 的阴影，第二个 TextView 用于显示当前时间。

为了实现发光的效果，TextView 提供了一个方法，其方法签名如下所示：

```
public void setShadowLayer (float radius, float dx, float dy, int color)
```

也可以通过 XML 中的 android:shadowColor, android:shadowDx、android:shadowDy 和 android:shadowRadius 属性使用这种效果。XML 文件内容如下所示：

```
<?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">

    <com.manning.androidhacks.hack011.view.LedTextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerInParent="true"
        android:text="88:88:88"
        android:textSize="80sp"
        android:textColor="#3300FF00"/>

    <com.manning.androidhacks.hack011.view.LedTextView
        android:id="@+id/main_clock_time"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerInParent="true"
        android:text="08:43:02"
        android:textSize="80sp"
        android:textColor="#00FF00"
        android:shadowColor="#00FF00"
        android:shadowDx="0"
        android:shadowDy="0"
        android:shadowRadius="10"/>

</RelativeLayout>
```

❶ 设置较浅的颜色，以使文本看起来透明些

❷ 设置文本和阴影颜色相同

❸ 改变阴影半径，使其看起来更亮些

第一个 LedTextView 用于绘制界面下层的 88:88:88，其目的是模拟数字时钟中的重影效果。我们在 ❶ 中通过把文本颜色设置的稍微透明一点来达到这个目的。第二个 LedTextView 用于显示当前时间，注意在 ❷ 中，字体颜色和阴影颜色是相同的。此外，还可以指定通过 android:alpha 属性指定透明度。

修改 android:shadowDx 和 android:shadowDy 属性的值可以改变阴影与文本之间的偏移。指定 android:shadowRadius 属性可以让用户产生一种文本更亮的错觉。为了产生一种发亮的效果，我们没有使用 android:shadowDx 和 android:shadowDy 属性，而是通过 ❸

中修改 android:shadowRadius 属性的值，使日期的文本内容看起来更亮一些。

11.1　概要

让应用程序看起来很棒，是在 Market 上获得用户好评的最好方法。大多数情况下，修饰应用程序界面会多写一些代码，但是这是值得的。此外，在文本中使用阴影很简单，却可以使应用程序界面看起来更专业。试试这个技巧吧，你不会后悔的。

11.2　外部链接

http://www.styleseven.com/php/get_product.php?product=Digital-7

http://developer.android.com/reference/android/widget/TextView.html

Hack 12　为背景添加圆角边框

Android v1.6+

当需要为应用程序 UI 控件选择背景的时候，开发者通常会使用图片。通常情况下，开发者会添加自定义颜色和形状来替代系统默认样式。

圆角边框是一种看起来很不错的效果，开发者只需要添加几行代码就可以在应用程序中使用这种效果。

我们以一个 Hello World 的示例程序为例，分析如何在应用程序中添加一个带圆角边框的灰色 Button。程序的最终效果如图 12-1 所示：

图 12-1　圆角边框的 Button

首先，需要在布局文件中添加一个 Button 控件，使用的 XML 文件如下所示：

```
<Button android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/hello"
    android:textColor="#000000"
    android:padding="10dp"
    android:background="@drawable/button_rounded_background"/>
```

由以上文件可知，我们并没有使用任何特殊属性。我们只是为背景属性指定了 drawable 值，但是这个值不是一个图片，而是一个 XML 文件。在 drawable 值指定的 XML 文件中有一个 ShapeDrawable 对象，该对象是一个可绘制对象，可以用来绘制各种原始形状，例如方块等。ShapeDrawable 对应的 XML 文件如下所示：

```
<shape xmlns:android="http://schemas.android.com/apk/res/android"
    android:shape="rectangle">
    <solid android:color="#AAAAAA"/>
    <corners android:radius="15dp"/>
</shape>
```

除了 radius 属性，我们还定义了 <shape/> 标签，并且使用 <solid/> 标签为 Button 指定了要填充的颜色值，使用 < corners /> 标签将边框指定为圆角。还有其他属性没有在本例中使用到，有兴趣的读者可以参照开发文档（见 12.2 节）了解这些属性。

12.1 概要

ShapeDrawable 是一个为 UI 控件添加特效的好工具。这个技巧适用于那些可以添加背景的控件。读者也可以在 ListView 上试用这个技巧，这样会使应用程序看起来更专业。

12.2 外部链接

http://developer.android.com/guide/topics/resources/drawable-resource.html#Shape

Hack 13 在 onCreate() 方法中获取 View 的宽度和高度

Android v1.6+

如果需要开发一些依赖于 UI 控件的宽和高的功能，开发者可能会用到 View 的 getHeight() 和 getWidth() 方法。对于新手来说，

这里有一个小陷阱值得注意：试图在 Activity 的 onCreate() 方法中获取控件的宽和高。遗憾的是如果开发者在 onCreate() 方法中调用上述方法，会发现返回值都是 0。本章我会向读者介绍一种避开这个陷阱的方法。

首先分析为什么在 Activity 的 onCreate() 方法中读取视图的尺寸会返回 0 值。当 onCreate() 方法被调用时，会通过 LayoutInflater 将 XML 布局文件填充到 ContentView。填充过程只包括创建视图，却不包括设置其大小。那么，视图的大小是在何时指定的呢?

Android 开发文档（见 13.2 节）的解释如下所示：

“绘制布局由两个遍历过程组成：测量过程和布局过程。测量过程由 measure(int, int) 方法完成，该方法从上到下遍历视图树。在递归遍历过程中，每个视图都会向下层传递尺寸和规格。当 measure 方法遍历结束时，每个视图都保存了各自的尺寸信息。第二个过程由 layout(int, int, int, int) 方法完成，该方法也是由上而下遍历视图树，在遍历过程中，每个父视图通过测量过程的结果定位所有子视图的位置信息。”

结论如下：只有在整个布局绘制完毕后，视图才能得到自身的高和宽，这个过程发生在 onCreate() 方法之后，因此，在此之前调用 getHeight() 和 getWidth() 方法返回的结果都是 0。

把 XML 布局文件比喻成蛋糕食谱：LayoutInflater 类就是购买所有食材的人；测量和布局的过程就是蛋糕师的工作，最终的视图就是蛋糕本身。在 onCreate() 阶段，只是购买了制作蛋糕的食材，但是仅仅知道食材是不足以预知蛋糕最终大小的。

开发者可以使用 View 的 post() 方法解决上述问题。该方法接收一个 Runnable 线程参数，并将其添加到消息队列中。有趣的是 Runnable 线程会在 UI 线程中执行。使用 post() 方法的源码如下所示：

```
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);

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

```
    View view = findViewById(R.id.main_my_view);
    view.post(new Runnable() {

        @Override
        public void run() {

            Log.d(TAG, "view has width: "+view.getWidth() +
               " and height: "+view.getHeight());
        }
    });
}
```

布局绘制后获取 View 的大小

正确的宽和高

13.1 概要

Android 源代码中很多模块都使用了 post() 方法，该方法并不仅限于获取视图的宽和高。如果读者分析过 View 类的源代码，并搜索过 post 关键词，你会惊讶调用 post() 方法的地方竟然如此之多。深入理解框架层运行机制对于避开类似于本例中的小陷阱是很重要的。最后，还是那句话，理解它的用处，不要滥用它。

13.2 外部链接

http://source.android.com/source/downloading.html

http://developer.android.com/guide/topics/ui/how-android-draws.html

Hack 14 VideoView 的转屏处理技巧

Android v1.6+

为应用程序添加视频是提供丰富用户体验的好方法。我曾见过使用包含视频的花式图展示公司信息的应用程序。有时，在复杂视图中使用视频展示信息是一个简单方法，这样就不需要编写动画效果的逻辑代码了。

我注意到，当观看视频时，用户往往喜欢切换到横屏模式。因此，在这个 Hack 里，我会向读者展示当屏幕旋转时，如何使视频全屏显示。

要实现上述功能，需要告诉系统，转屏操作的处理过程由我们

自己来完成。当屏幕旋转后，我们会更改 VideoView 的大小和位置。

首先，为 Activity 创建布局文件。在这个 Hack 里，我们创建的布局文件由上下两部分组成，并以一条横线分割这两部分。上面的部分左侧显示一个稍小的文本，右侧显示一个视频；下面的部分显示一长段描述信息。当创建显示视频的视图时，我并没有使用 VideoView 控件，而是以一个白色背景的视图控件替代。我会使用这个视图控件的大小和位置来正确设置 VideoView 控件的位置。读者可以在图 14-1 中看到该布局文件的最终效果。

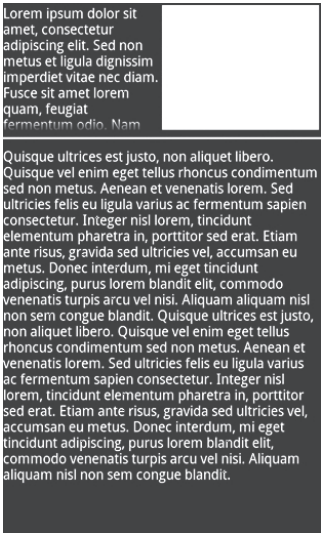

图 14-1　最终布局

从图 14-2 中，可以看出视图树的创建方式。VideoView 控件直接挂在根视图下，并且与 Portrait Content[⊖]控件处于相同的层级。将 VideoView 控件放置在这个位置的用意是：在屏幕旋转时，既不需要创建两个不同的布局，也不需要修改 VideoView 的父视图就可以更改 VideoView 的大小和位置。另外，那个白色背景的

⊖　指的是 ID 为 main_portrait_content 的 LinearLayout。——译者注

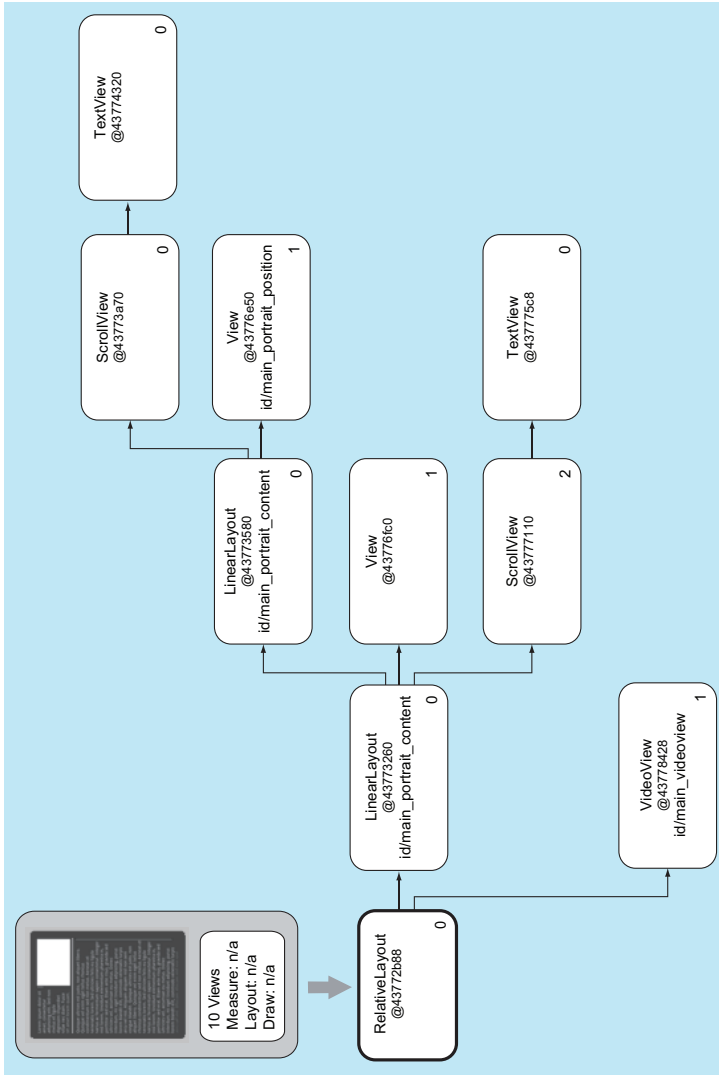
10 Views
Measure: n/a
Layout: n/a
Draw: n/a
RelativeLayout
@43772b88
0
LinearLayout
@43773260
id/main_portrait_content
0
VideoView
@43778428
id/main_videoview
1
LinearLayout
@43773580
id/main_portrait_content
0
View
@43776fc0
1
ScrollView
@43777110
2
ScrollView
@43773a70
0
View
@43776e50
id/main_portrait_position
1
TextView
@43774320
0
TextView
@437775c8
0

图 14-2　视图树

视图则命名为 portrait position ⊖，位于视图树中较深的位置。

现在我们有了布局文件，就可以编写 Activity 的源码了。首先，要让 Activity 能够处理转屏操作。要实现这个功能，需要在 AndroidManifest.xml 文件中相应的 <Activity> 标签中添加 android:configChanges="orientation" 属性。配置该属性后，当屏幕旋转时，系统不会重启 Activity，而是调用该 Activity 的 onConfigurationChanged() 方法。

当屏幕的方向改变后，就需要改变视频的大小和位置。我们通过一个名为 setVideoViewPosition() 的私有方法完成这个功能，其源码如下所示：

```
private void setVideoViewPosition() {
  if (getResources().getConfiguration().orientation ==
    ActivityInfo.SCREEN_ORIENTATION_PORTRAIT) {
    mPortraitContent.setVisibility(View.VISIBLE);
    int[] locationArray = new int[2];
    mPortraitPosition.getLocationOnScreen(locationArray);
    RelativeLayout.LayoutParams params =
      new RelativeLayout.LayoutParams(mPortraitPosition.getWidth(
          mPortraitPosition.getHeight());
    params.leftMargin = locationArray[0];
    params.topMargin = locationArray[1];
    mVideoView.setLayoutParams(params);
  } else {
    mPortraitContent.setVisibility(View.GONE);
    RelativeLayout.LayoutParams params =
      new RelativeLayout.LayoutParams(LayoutParams.FILL_PARENT,
          LayoutParams.FILL_PARENT);
    params.addRule(RelativeLayout.CENTER_IN_PARENT);
    mVideoView.setLayoutParams(params);
  }
}
```

❶ 判断当前横竖屏配置

❷ 使 mPortrait-Content 可视

❸ videoView 的位置

❹ 设置 videoView 的布局参数

❺ 隐藏 mPortrait-Content

❻ 设置 videoView 的布局参数

如 ❶ 中所示，setVideoViewPosition() 方法分为两个分支，分别对应竖屏配置和横屏配置。在竖屏配置的情况下，首先在 ❷ 中令 portrait content 可视。因为在这种情况下，videoView 和白色背景的视图具有相同的大小和位置，因此在 ❸ 中，获取白色背景视

⊖ 指的是 ID 为 main_portrait_position 的 View 控件。——译者注

图的位置信息，并在 ❹ 中将其设置为 videoView 的布局参数。

横屏分支的情况与上述类似。首先，在 ❺ 中，隐藏 portrait content；然后创建一个布局参数使 videoView 可以全屏显示；最后，在 ❻ 中，将该布局参数设置给 videoView。

14.1 概要

正如我在本 Hack 开始时提到的，视频对于丰富应用程序内容是很有用的。开发者可能知道当视频大小改变时，默认的 videoView 类会维持屏幕宽高比，如果你想让视频填充整个可用控件，就需要覆盖自定义视图中的 onMeasure() 方法。

14.2 外部链接

http://developer.android.com/guide/topics/resources/runtime-changes.html

Hack 15 移除背景以提升 Activity 启动速度

Android v1.6+

Android SDK 中提供了 Hierarchy Viewer 工具，该工具可以用来检测未被使用的视图以减少视图树的层次。如果开发者通过该工具浏览一颗视图树，会发现许多不可控的节点。在这个 Hack 里，我们来看看这些节点是什么以及如何微调它们来提升 Activity 的启动速度。

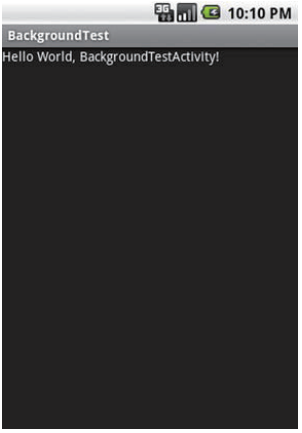

图 15-1 默认 Android 应用程序

如果读者创建一个默认应用程序，并运行该程序，会看到与图 15-1 类似的界面。如果此时运行 Hierarchy Viewer 工具查看该 Activity，会看到

与图 15-2 类似的界面。我们需要削减视图树的层次。

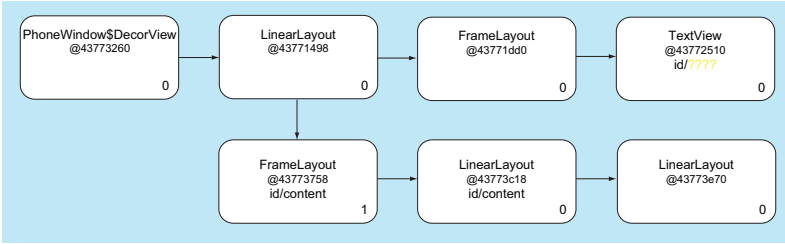

图 15-2　Hierarchy Viewer 显示的 View 树

首先，通过移除标题栏来减少一些节点。标题栏就是界面上方那个显示“ BackgroundTest”的灰条，它由一个 FrameLayout 容器和一个 TextView 控件组成。可以通过在 res/values 目录下创建 theme.xml 文件来删除上述节点。源码如下所示：

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <style name="Theme.NoBackground" parent="android:Theme">
        <item name="android:windowNoTitle">true</item>
    </style>
</resources>
```

可以通过修改 Androidmanifest.xml 中的 <application> 标签，并添加 android:theme="@style/Theme.NoBackground" 属性来使用这个主题。此时，重新运行应用程序，标题栏就消失了，其 View 树如图 15-3 所示。

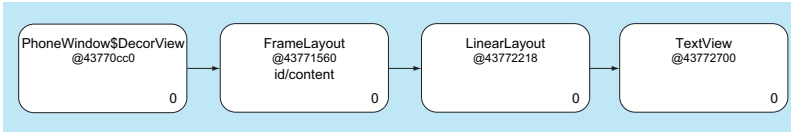

图 15-3　无标题栏时，Hierarchy Viewer 显示的 View 树

读者已经知道 LinearLayout 和 TextView 是什么了，那么 PhoneWindow$DecorView 和 FrameLayout 又是什么呢？

FrameLayout 是在执行 setContentView() 方法时创建的，

DecorView 是视图树的根节点。默认情况下，框架层会以默认背景色填充窗口，而 DecorView 就是持有窗口背景图片的视图。如果使用不透明的界面或者自定义背景，那么绘制默认背景色就完全是在浪费时间。

如果确信要在 Activity 中使用不透明的用户界面，就可以移除默认背景以加快 Activity 启动时间。要实现这个功能，需要在先前的主题中添加一行代码。修改后的代码如下所示：

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <style name="Theme.NoBackground" parent="android:Theme">
        <item name="android:windowNoTitle">true</item>
        <item name="android:windowBackground">@null</item>
    </style>
</resources>
```

15.1 概要

移除窗口默认背景是提升应用程序启动速度的一个简单技巧。判断是否需要移除背景的原则很简单：如果应用程序界面需要占据窗口 100% 的空间，就需要将 windowBackground 属性设置为 null。记住，主题既可以在 <application> 标签中设置，也可以在 <activity> 标签中设置。

15.2 外部链接

http://developer.android.com/guide/developing/debugging/debugging-ui.html#HierarchyViewer

http://stackoverflow.com/questions/6499004/androidwindow-background-null-to-improve-app-speed

Hack 16 更改 Toast 显示位置的技巧

Android v1.6+

在 Android 中，如果开发者需要通知用户发生了什么事情，可以使用 Toast 这个类。Toast 是一种弹出式通知，通常在屏幕下

方中间位置显示一行文本。如果读者从未见过 Toast，可以看看图 16-1，在该图中，Toast 就是那个显示“This alarm is set for 17 hours and 57 minutes from now.”的黑色的方框。

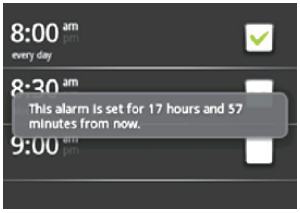

图 16-1　闹钟应用程序中的 Toast 例子

使用 API 创建一个 Toast 是非常简单的，假如要创建一个显示“Hi!”的 Toast，只需要编写以下几行代码：

```
Toast.makeText(this, "Hi!", Toast.LENGTH_SHORT).show();
```

Toast 类提供的接口很不灵活。例如，对于 Toast 显示时间的参数，开发者仅仅可以从 Toast.LENGTH_SHORT 和 Toast.LENGTH_LONG 中选择。虽然 Toast 的变化很少，但是仍然可以改变 Toast 弹出的位置。

图 16-2　在不同位置显示的 Toast

根据应用程序布局的不同，开发者可能需要将 Toast 显示在其他位置，比如，在指定视图的顶部显示一个 Toast。让我们看看如何创建一个 Toast，并使其在不同的位置显示。请看图 16-2 所示的例子，在这个例子中，我们定义了 4 个 Button，分别分布在界面的四角上。当点击 Button 时，会在 Button 所在位置的角落里显示一个 Toast。

要将 Toast 移动到屏幕其他地方，就需要以不同的方式创建

Toast。Toast 中有一个公共方法，其方法签名如下：

```
public void setGravity(int gravity, int xOffset, int yOffset);
```

要显示图 16-2 中所示的 Toast，需要使用如下代码：

```
Toast toast = Toast.makeText(this, "Bottom Right!",          <-- 创建 Toast
        Toast.LENGTH_SHORT);

toast.setGravity(Gravity.BOTTOM | Gravity.RIGHT, 0, 0);      <-- 设置 gravity 属性，以改变默认位置
toast.show();
```

16.1 概要

虽然这个 Hack 看起来挺简单，但是很多 Android 开发者并不知晓本例的解决方案。当界面被分成不同的 Fragment，并且希望 Toast 显示在指定的位置，开发者就会发现更改 Toast 的位置是很有用的。

16.2 外部链接

http://developer.android.com/guide/topics/ui/notifiers/toasts.html

Hack 17 使用 Gallery 创建向导表单

Android v2.1+

当需要用户填充一个较长的表单（form）时，开发者或许会找不到头绪。开发者的目的可能是需要创建一个用户注册表单，或者是需要应用程序通过表单上传一些数据。在其他平台，开发者可以创建向导表单（wizard form）满足上述需求，这种表单可以把表单项分割到不同的视图中。但是 Android 平台没有提供类似控件。在这个 Hack 里，我会使用 Gallery 控件创建一个具有多个表单项的用户注册表单。最终效果如图 17-1 所示。

针对这个例子，在创建的注册表单中，用户需要输入以下信息：

- ❑ 姓名
- ❑ 电子邮件

- ❑ 密码
- ❑ 性别
- ❑ 城市
- ❑ 国家
- ❑ 邮编

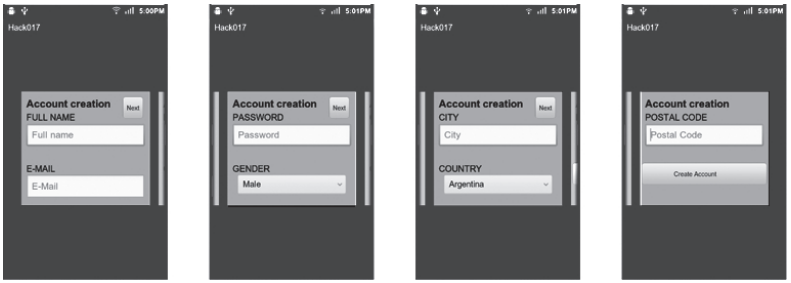

图 17-1　使用 Gallery 控件实现的向导表单

我们在每个页面显示两个表单项，因此一共需要 4 个页面才能完全显示上述信息。要实现这个向导表单，需要创建一个命名为 CreateAccountActivity 的 Activity。为了让表单具备弹出窗口的式样，我们为上述 Activity 使用 Theme.Dialog 样式。在该 Activity 中，我们会创建一个 Gallery 对象，并且用一个 Adapter 填充这个 Gallery。因为该 Adapter 需要与 Activity 交互，因此我们使用 Delegate 委托接口。

先创建每个页面的公用视图，XML 文件内容如下所示：

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="270dp"
    android:layout_height="350dp">

    <LinearLayout android:id="@+id/create_account_form"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_alignParentTop="true"
        android:orientation="vertical"
        android:paddingLeft="10dp"
        android:paddingTop="10dp"
        android:paddingRight="10dp"
        android:background="#AAAAAA">
```

在该 LinearLayout 中放置要显示的表单项

```
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Account creation"
        android:textColor="#000000"
        android:textStyle="bold"
        android:textSize="20sp"/>

</LinearLayout>

<Button
    android:id="@+id/create_account_next"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignParentTop="true"
    android:layout_alignParentRight="true"
    android:textSize="12sp"
    android:gravity="center"
    android:layout_marginTop="10dp"
    android:layout_marginRight="10dp"
    android:text="Next"/>

<Button
    android:id="@+id/create_account_create"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:layout_below="@id/create_account_form"
    android:gravity="center"
    android:paddingRight="45dp"
    android:text="Create Account"
    android:textSize="12sp"/>

</RelativeLayout>
```

在 LinearLayout 第一个子视图中显示表单标题

“Next“按钮用于切换到下一个页面

这个按钮只在最后一个页面显示，用于提交表单

由以上代码可知，我们在布局中设置了一个 LinearLayout 作为表单项的占位符。后续分析中，读者就会看到如何通过 Gallery 的 Adapter 填充它。

既然有了公用视图的布局文件，我们就可以创建 Adapter 的代码。我们将该 Adapter 命名为 CreateAccountAdapter，继承自 BaseAdapter。鉴于该 Adapter 的代码较长，这里只分析重要的方法。第一个要定义的就是用来与上述 Activity 交互的接口，代码如下所示：

```
public static interface CreateAccountDelegate {
    int FORWARD = 1;
    int BACKWARD = -1;

    void scroll(int type);

    void processForm(Account account);
}
```

当用户点击“ Next”按钮时，执行 scroll() 方法；当用户提交表单时，执行 proccessForm() 方法。当按钮被点击时，需要访问委托接口（ delegate），因此我们在 getView() 方法中设置点击事件监听器。源码如下所示：

```
public View getView(int position, View convertView, ViewGroup parent) {

    convertView = mInflator.inflate(
        R.layout.create_account_generic_row, parent, false);

    LinearLayout formLayout = (LinearLayout) convertView
        .findViewById(R.id.create_account_form);

    View nextButton = convertView
        .findViewById(R.id.create_account_next);
    if (position == FORMS_QTY - 1) {
      nextButton.setVisibility(View.GONE);
    } else {
      nextButton.setVisibility(View.VISIBLE);
    }

    if (mDelegate != null) {
      nextButton.setOnClickListener(new OnClickListener() {

        @Override
        public void onClick(View v) {
        mDelegate.scroll(CreateAccountDelegate.FORWARD);
        }
      });
    }

    Button createButton = (Button) convertView
        .findViewById(R.id.create_account_create);
    if (position == FORMS_QTY - 1) {
      createButton.setOnClickListener(new OnClickListener() {

        @Override
        public void onClick(View v) {
          processForm();
        }
      });

      createButton.setVisibility(View.VISIBLE);
    } else {
      createButton.setVisibility(View.GONE);
    }

    switch (position) {
    case 0:
      populateFirstForm(formLayout);
      break;

    ...

    }

    return convertView;
}
```

填充自定义 View

获取放置表单项的 LinearLayout

最后一页不显示“ Next”按钮

仅在最后一页显示该按钮

最后一步，根据页面位置填充 LinearLayout

populateFirstForm() 方法用于创建姓名和标题这两个表单项，

并将其添加到 LinearLayout 中。在示例程序中，我通过代码实现上述逻辑，但是读者也可以通过填充 XML 布局文件的方式实现。

还有一个模块没有讲到，那就是谁来实现 CreateAccount-Delegate 接口。在本例中，我们使用 CreateAccountActivity 实现该接口。

CreateAccountActivity 用于跟踪用户当前所在的页面，并处理页面跳转的逻辑。源码如下所示：

```
public class CreateAccountActivity extends Activity implements
    CreateAccountDelegate {

  private Gallery mGallery;
  private CreateAccountAdapter mAdapter;
  private int mGalleryPosition;

  @Override
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);

    setContentView(R.layout.create_account);
    mGallery = (Gallery) findViewById(R.id.create_account_gallery);

    mAdapter = new CreateAccountAdapter(this);
    mGallery.setAdapter(mAdapter);
    mGalleryPosition = 0;
  }

  @Override
  protected void onResume() {
    super.onResume();
    mAdapter.setDelegate(this);
  }

  @Override
  protected void onPause() {
    super.onPause();
    mAdapter.setDelegate(null);
  }

  @Override
  public void onBackPressed() {

    if (mGalleryPosition > 0) {
      scroll(BACKWARD);
    } else {
      super.onBackPressed();
    }
  }

  @Override
  public void scroll(int type) {

    switch (type) {
    case FORWARD:
      if (mGalleryPosition < mGallery.getCount() - 1) {
        mGallery.onKeyDown(KeyEvent.KEYCODE_DPAD_RIGHT,
            new KeyEvent(0, 0));
        mGalleryPosition++;
```

在 onCreate() 方法中创建 Adapter，并将其设置给 Gallery

在 onResume() 方法中将当前 Activity 设置为 Adapter 的委托，并且在 onPause() 方法中将该委托置空

重写 Activity 的 onBack-Pressed() 方法，用于返回上一个页面

在 scroll() 方法中，Activity 可以根据参数将 Gallery 移动到下一个页面或者上一个页面中

```
            }
            break;
            ...
        }
        ...
    }
```

遗憾的是，我们无法为 Gallery 控件的页面切换添加动画效果。我想到的唯一方法是发送 KeyEvent.KEYCODE_DPAD_RIGHT 事件，虽然这是投机取巧，却还管用。

CreateAccountActivity 中其他代码用于验证数据和错误处理。这些代码并无特别之处，因此留给读者自己阅读。

17.1　概要

使用 Gallery 控件创建向导表单可以简化用户填写较长表单的流程。将表单项放在不同页面中，并且利用 Gallery 控件的默认动画添加悦目的效果，可以使用户在填写表单的过程中更愉悦些。

根据需要，读者也可以使用 ViewPager 开发相同的功能，只是其 Adapter 返回的不是 View 而是 Fragment。

17.2　外部链接

http://developer.android.com/reference/android/widget/Gallery.html